An Equivariant Flow Matching Framework for Learning Molecular Crystallization

Shengchao Liu [1] **Divin Yan** [1] **Hongyu Guo** [2 3] **Anima Anandkumar** [1 3]

Abstract

Molecular crystallization is the transformation of molecules from weakly-correlated structures to strongly-correlated structures, *e.g.*, liquid water freezing into solid ice. It is essential in determining the functionalities of compounds across various fields, from pharmaceuticals to materials science. However, existing simulation methods for crystallization, which primarily rely on numerical techniques, can be exceedingly time-consuming. In this work, we build up a novel ML paradigm for solving the crystallization problem, *i.e.*, learning a distribution mapping from weakly-correlated structures to strongly-correlated structures. First, we construct two datasets, coined COD-Cluster17, for benchmarking and also design two packing matching metrics for crystallization evaluation. Second, we propose CrystalFlow, an SE(3)-equivariant Flow Matching framework, for modeling the crystallization trajectories of molecular clusters. The data and code are available on this GitHub link.

1. Introduction

In the past years, advanced machine learning (ML) techniques have proven adept at effectively narrowing the chemical search space and sufficiently expediting the molecular discovery process across critical tasks, including molecular property prediction (Raccuglia et al., 2016; Ward et al., 2016), de novo molecule design (Sanchez-Lengeling & Aspuru-Guzik, 2018), molecular dynamics simulation, and quantum chemistry calculation (Wang et al., 2018), amongst many others. Despite its significance, machine learning for crystallization represents a challenging and fundamental aspect of the pipeline, yet it remains relatively underexplored. This paper aims to fill this gap.

Crystallization. This is a process where atoms, ions, or molecules arrange themselves from weakly-correlated structures to strongly-correlated structures, with a repetitive pattern known as the **crystal lattice**. One classic example is water-to-ice crystallization as depicted in Figure 1.

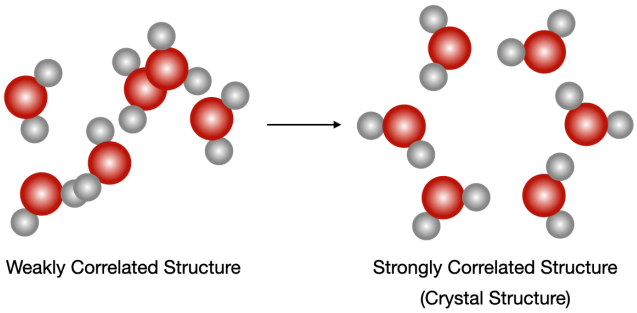


Figure 1. Demonstration of crystallization on a cluster of six molecules: liquid water freezing into solid ice. In CrystalFlow, we adopt an equivariant Conditional Flow Matching framework to learn such a process.

For the experimental chemist, employing ML for crystallization to generate reasonable crystal structures can be helpful in many scenarios. For instance, X-ray diffraction is commonly used to determine crystal structures post-crystallization. However, the diffraction signal can sometimes be weak due to the poor crystal quality, making it impractical to solve the crystal structure directly. ML for crystallization can serve as a validation tool providing plausible crystal structures which can be used to simulate diffraction pattern and compare with the experiment results. Besides, such a tool also has the potential for virtual screening databases where experimentally determined crystal structures are not available. Reasonable virtual crystal structures can be generated based on the molecular entities on which calculations can be run for screening target properties.

For the computational chemist, methods like molecular dynamics simulation and structure minimization (or energy minimization) have been used for estimating the crystallization process. Nonetheless, these methods present notable challenges. In particular, the energy landscape is complex due to the presence of multiple local minima. Additionally, the computational demands of these methods are substantial. Moreover, the intricacies of crystallization, encompassing both kinetics and thermodynamics, add further challenges to achieving accurate estimations.

[1]Caltech [2]NRC [3]Joint Advising. Correspondence to: Shengchao Liu <shengchao1224@gmail.com>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at 41 st International Conference on Machine Learning, Vienna, Austria.

We also want to highlight that from the ML aspect, existing tools have shown great promise in tackling complex tasks. For instance, AlphaFold has made significant strides in predicting the structures of macromolecules by accurately simulating the protein folding process (Jumper et al., 2021). Both protein folding and molecule crystallization involve predicting how molecules arrange themselves into stable structures, which is critical for understanding their function and properties. However, while ML for folding has demonstrated success in computational biology, the application of ML to molecular crystallization remains underdeveloped.

Group symmetry and SE(3)-equivariance. When modeling geometric data, such as atomic coordinates in 3D Euclidean space (also known as conformation), the representation/density function must be equivariant with respect to translations and rotations. These rigid motions in 3D Euclidean space form the SE(3) group, and functions that maintain the equivariance property under these transformations are called SE(3)-equivariant. Existing SE(3)-equivariant models can be categorized into two families: projecting the vector variables onto either spherical harmonics basis or the vector frame basis (Liu et al., 2023). In addition to SE(3)-equivariance, crystal structures also possess a periodic lattice, so modeling them needs to be periodic-invariant as well. We note that since crystal structures only appear after the crystallization process, we do not need to consider periodic-invariance in this work.

Dataset curation. In this work, we construct a dataset COD-Cluster17 from the Crystallography Open Database (COD). COD-Cluster17 has two versions, with and without the molecule inversion when creating the dataset, and both include 106,672 molecules. Each crystal molecule in COD-Cluster17 includes a cluster of weakly-correlated structures as the initial position and a cluster of strongly-correlated structures as the final position after crystallization. The inversion dataset introduces a fifty percent chance of inverting each molecule to account for achiral phenomena in crystallization. Also, these datasets are in a format that is easily accessible and user-friendly for machine learning researchers. The overall concept for constructing the dataset is illustrated in Appendix C.

Diffusion and Flow Matching. Denoising diffusion probabilistic model (Ho et al., 2020) or denoising score matching model (Song et al., 2020; Vincent, 2011) has been widely used for learning the target data distribution from a standard Gaussian prior. This has shown its advantage in learning the data distribution in the continuous space, *e.g.*, the geometric coordinates in the Euclidean space. More recently, Flow Matching (Albergo & Vanden-Eijnden, 2022; Lipman et al., 2022; Liu et al., 2022) is proposed as a more general framework that learns the mapping from an arbitrary prior distribution to a target distribution, following a linear interpolation (LERP) path. Additionally, as shown in (Lipman et al., 2022), Diffusion is a special case of Flow Matching, and we will benchmark both for the crystallization task.

CrystalFlow framework. In our work, we propose CrystalFlow, the first SE(3)-equivariant flow matching framework for learning molecular crystallization, to the best of our knowledge. In specific, (1) CrystalFlow leverages Conditional Flow Matching techniques to effectively map weakly-correlated distributions to strongly-correlated distributions. We benchmark three main variants of CrystalFlow, with Diffusion or LERP path. Experiments reveal that LERP is more appealing because it can learn distributions from arbitrary prior distributions as long as the probability path follows Gaussian transformation, while FM with Diffusion path requires the prior distribution to be Gaussian. (2) CrystalFlow aims at molecular clusters for crystallization. Modeling molecular clusters excludes the need for periodic information during the crystallization process. (3) CrystalFlow benchmarks on the two versions of COD-Cluster17, each with three subsets. Results are quantitatively measured with two packing matching metrics. We further qualitatively illustrate that CrystalFlow can resemble the crystallization of clusters of molecules.

2. CrystalFlow Framework

For each molecule, we have its composition (atom types) as $\boldsymbol{a}$, weakly-correlated geometry as $\boldsymbol{x}_w$, and strongly-correlated geometry as $\boldsymbol{x}_s$. The goal of ML for crystallization is to predict the crystal structure given the weakly-correlated geometry, namely $p(\boldsymbol{x}_s|\boldsymbol{x}_w, \boldsymbol{a})$. In this section, we propose CrystalFlow, an SE(3)-equivariant Flow Matching framework for learning the crystallization process.

2.1. SE(3)-equivariant Modeling

Interactions between molecules significantly influence the nucleation, growth, and stabilization processes inherent in crystallization. Consequently, CrystalFlow aims at simultaneously modeling a cluster of B molecules. The key modeling challenge here is the absence of lattice information during the crystallization process.

To be more concrete, CrystalFlow models the crystallization as a conditional generative process, with the learning and inference scaled to a time range of $t \in [0, 1]$. In it, the core module is an SE(3)-equivariant function $f(\cdot)$. The inputs are atomic types $\boldsymbol{a}$ (composition) and atomic coordinates $\boldsymbol{x}_t$ at timestep t, and the outputs are the coordinates $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{a})$ at timestep $t + 1$.

Ideally, any expressive SE(3)-equivariant model can be applied here. However, because we are modeling clusters of $B = 17$ molecules simultaneously for each molecule in COD-Cluster17, we prefer the SE(3)-equivariant model that

offers a great balance between efficiency and effectiveness. (The distribution on # atoms in each molecule is illustrated in Appendix D.) As extensively verified in Geom3D (**?**), PaiNN is adopted here for geometric modeling (Schütt et al., 2021). The high-level idea is that for each atom i, we have a corresponding scalar representation $\boldsymbol{h}_i^s \in \mathbb{R}^d$ and $\boldsymbol{h}_i^v \in \mathbb{R}^{3\times d}$ where d is the latent dimension. At the initial layer, $\boldsymbol{h}^s$ is the one-hot encoding of atomic number $\boldsymbol{a}$, and $\boldsymbol{h}^v$ is initialized as zero-tensor. Then we apply the message passing between each atom i and its neighborhood atoms j within a cutoff threshold c, *i.e.*, $\|\boldsymbol{x}_{ij}\| \leq c$. There are two stages for message passing in PaiNN, the first stage is from scalar to scalar and vector:

$$\begin{aligned}\boldsymbol{h}_i^s &= \boldsymbol{h}_i^s + \sum_j \text{MLP}(\boldsymbol{h}_j^s)\cdot\text{MLP}(\|\boldsymbol{x}_{ij}\|),\\ \boldsymbol{h}_i^v &= \boldsymbol{h}_i^v + \sum_j \boldsymbol{h}_j^v\cdot\text{MLP}(\boldsymbol{h}_j^s)\cdot\text{MLP}(\|\boldsymbol{x}_{ij}\|)\\ &\quad + \sum_j \frac{\boldsymbol{x}_{ij}}{\|\boldsymbol{x}_{ij}\|}\cdot\text{MLP}(\boldsymbol{h}_j^s)\cdot\text{MLP}(\|\boldsymbol{x}_{ij}\|),\end{aligned} \tag{1}$$

where $\text{MLP}(\cdot)$ is the multi-layer perceptron, and $\cdot$ is the element-wise product. The second stage is atom-wise update, from vector to scalar and vector:

$$\begin{aligned}\boldsymbol{h}_i^s &= \boldsymbol{h}_i^s + \text{MLP}(\boldsymbol{h}_i^s \oplus \|V\boldsymbol{h}_i^v\|)\\ &\quad + \text{MLP}(\boldsymbol{h}_i^s \oplus \|V\boldsymbol{h}_i^v\|)\cdot\langle U\boldsymbol{h}_i^v, V\boldsymbol{h}_i^v\rangle,\\ \boldsymbol{h}_i^v &= \boldsymbol{h}_i^v + \text{MLP}(\boldsymbol{h}_i^s \oplus \|V\boldsymbol{h}_i^v\|)\cdot U\boldsymbol{h}_i^v,\end{aligned} \tag{2}$$

where $\oplus$ is the concatenation, $U = \text{MLP}(\boldsymbol{h}_j^v)$, $V = \text{MLP}(\boldsymbol{h}_j^v)$, $\|\cdot\|$ means taking the norm along the three Euclidean bases, and $\langle\cdot\rangle$ is the inner product. By repeating Equations (1) and (2) for L layers, and we can take the $\boldsymbol{x}_i = \text{MLP}(\boldsymbol{h}_L^v)$ as the coordinate shift from time t to $t+1$.

The resulting atom displacements from this geometric modeling are then utilized by a Flow Matching algorithm to capture the transitional trajectories from weakly-correlated to strongly-correlated geometries during the crystallization process, which is detailed in Section 2.3.

2.2. Conditional Flow Matching.

Conditional Flow Matching (CFM) (Lipman et al., 2022) and two parallel works (Rectified Flow (Liu et al., 2022) and Stochastic Interpolants (Albergo & Vanden-Eijnden, 2022)) formulate the distribution modeling problem as learning a vector field that can generate a probability path mapping from simple distribution at $t=0$ to the target distribution at $t=1$. We provide a brief introduction here and the detailed derivatives can be found in Appendix B.

First, we need to bring in four key concepts. Our geometric data are atomic coordinates in the 3D Euclidean points $\boldsymbol{x} \in \mathbb{R}^3$, and the atomic type is fixed during the whole crystallization process, so we may as well ignore that. Then we define time-dependent vector field $\boldsymbol{v} : [0,1]\times\mathbb{R}^3 \to \mathbb{R}^3$. A time-dependent vector field defines a time-dependent diffeomorphic map, called flow, $\phi : [0,1]\times\mathbb{R}^3 \to \mathbb{R}^3$. The vector field defines flow via an ordinary differential equation as

$$\frac{d}{dt}\phi_t(\boldsymbol{x}) = u_t(\phi_t(\boldsymbol{x})). \tag{3}$$

A probability density path is denoted as $p : [0,1]\times\mathbb{R}^3 \to \mathbb{R}_{>0}$. Existing flow model (Chen et al., 2018) maps a prior distribution p_0 to another distribution p_t with push-forward equation or change of variable rule:

$$p_t(\boldsymbol{x}) = [\phi_t]_* p_0(\boldsymbol{x}) = p_0(\phi_t^{-1}(\boldsymbol{x}))\det\left|\frac{d\phi_t(\boldsymbol{x})}{dx}\right|^{-1}. \tag{4}$$

Thus, modeling the likelihood of data distribution at $t=1$ can be transformed into modeling the velocity field matching problem with parameterized velocity field v_θ, *i.e.*, Flow Matching:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\boldsymbol{x}}\left\|u_t(\boldsymbol{x}) - v_\theta(\boldsymbol{x},t)\right\|^2. \tag{5}$$

With the continuity equation (Villani et al., 2009), we can further get an equivalent objective by considering the conditional vector field conditioned on the empirical data $\boldsymbol{x}_1$, *i.e.*, $u_t(\boldsymbol{x}|\boldsymbol{x}_1)$, and the objective is the Conditional Flow Matching:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,\boldsymbol{x},\boldsymbol{x}_1}\left\|u_t(\boldsymbol{x}|\boldsymbol{x}_1) - v_\theta(\boldsymbol{x},\boldsymbol{x}_1,t)\right\|^2. \tag{6}$$

2.3. CrystalFlow Framework

We briefly introduce how to utilize Conditional Flow Matching to solve the general distribution estimation problem (Equation (6)). To adapt it in our setting, the question is how to construct the conditional velocity field $u_t(\boldsymbol{x}|\boldsymbol{x}_s)$ and with a preference for a closed-form formulation.

Fortunately, Lipman et al. (Lipman et al., 2022) show that when considering a class of Gaussian conditional probability paths, one can analytically compute the conditional vector field provided the means and the standard deviations are differentiable. This has inspired the following forms of vector fields by defining the means and deviations.

A simple example provided in (Lipman et al., 2022) is a Gaussian conditional probability path $p_t(\boldsymbol{x}|\boldsymbol{x}_s) = \mathcal{N}(\boldsymbol{x}|\mu_t(\boldsymbol{x}_s), \sigma_t(\boldsymbol{x}_s)^2 I)$, and its corresponding objective function is:

$$\mathcal{L}_{\text{CFN}} = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_s}\left\|u_t(\phi(\boldsymbol{x}_w)|\boldsymbol{x}_s) - v_\theta(\boldsymbol{x}_t,\boldsymbol{x}_w,t)\right\|^2, \tag{7}$$

where $u_t(\phi_t(\boldsymbol{x})|\boldsymbol{x}_s) = \frac{\sigma_t'(\boldsymbol{x}_s)}{\sigma_t(\boldsymbol{x}_s)}\big(\phi_t(\boldsymbol{x}) - \mu_t(\boldsymbol{x}_s)\big) + \mu_t'(\boldsymbol{x}_s)$.

Linear interpolation (LERP) path. First let us consider the linear interpolation (LERP) path between $\boldsymbol{x}_w$ and $\boldsymbol{x}_s$, which is one of the most straightforward path designs inspired by

Table 1. Results for CrystalFlow on COD-Cluster17 without inversion. The evaluation metrics are packing matching (PM) on the atom-level and mass-center-level, and achieving a smaller PM indicates better performance.

	COD-5K		COD-10K		COD-100K	
	PM (atom) ↓	PM (center) ↓	PM (atom) ↓	PM (center) ↓	PM (atom) ↓	PM (center) ↓
GNN-MD	13.668 ± 0.062	13.799 ± 0.069	13.831 ± 0.061	13.902 ± 0.052	22.296 ± 12.044	14.511 ± 0.816
CrystalSDE-VE	15.517 ± 1.484	16.456 ± 0.987	17.252 ± 2.459	17.863 ± 1.113	17.285 ± 0.730	18.918 ± 0.030
CrystalSDE-VP	18.154 ± 3.023	19.147 ± 4.463	22.200 ± 3.292	21.388 ± 1.496	18.032 ± 4.561	20.018 ± 3.699
CrystalFlow-LERP	$\mathbf{13.589 \pm 0.087}$	$\mathbf{13.256 \pm 0.089}$	$\mathbf{13.539 \pm 0.033}$	$\mathbf{13.197 \pm 0.032}$	$\mathbf{13.608 \pm 0.005}$	$\mathbf{13.278 \pm 0.009}$

Table 2. Results for CrystalFlow on COD-Cluster17 with inversion. The evaluation metrics are packing matching (PM) on the atom-level and mass-center-level, and achieving a smaller PM indicates better performance.

	COD-5K		COD-10K		COD-100K	
	PM (atom) ↓	PM (center) ↓	PM (atom) ↓	PM (center) ↓	PM (atom) ↓	PM (center) ↓
GNN-MD	15.356 ± 0.089	14.054 ± 0.094	15.390 ± 0.024	14.085 ± 0.017	15.512 ± 0.013	14.191 ± 0.010
CrystalSDE-VE	18.775 ± 0.517	19.438 ± 0.617	20.769 ± 2.634	19.282 ± 1.816	15.651 ± 0.303	16.786 ± 0.387
CrystalSDE-VP	14.600 ± 5.088	14.066 ± 4.628	20.390 ± 2.141	18.497 ± 1.733	21.335 ± 0.217	20.209 ± 0.116
CrystalFlow-LERP	$\mathbf{13.630 \pm 0.065}$	$\mathbf{13.315 \pm 0.069}$	$\mathbf{13.599 \pm 0.024}$	$\mathbf{13.269 \pm 0.031}$	$\mathbf{13.580 \pm 0.025}$	$\mathbf{13.252 \pm 0.027}$

the flow matching framework. We design the mean and variance at time t as:

$$\mu_t(\boldsymbol{x}) = (1-t)\boldsymbol{x}_w + t\boldsymbol{x}_s, \sigma_t(x) = 1-(1-\sigma_{\min})t. \quad (8)$$

Then we have the conditional velocity as $u_t(\phi_t(\boldsymbol{x})|\boldsymbol{x}_s) = \frac{\boldsymbol{x}_s - \sigma_{\min}\boldsymbol{x}_w - (1-\sigma_{\min})\boldsymbol{x}}{1-(1-\sigma_{\min})t}$. Plugging this into Equation (7), the objective function is:

$$\mathcal{L}_{\text{LERP}} = \mathbb{E}_{\boldsymbol{x}_w, \boldsymbol{x}_s, t}\left[\left\|\boldsymbol{x}_s - \boldsymbol{x}_w - v_\theta(\boldsymbol{x}_t, \boldsymbol{x}_w, t)\right\|^2\right]dt. \quad (9)$$

For more detailed derivations, please check Appendix B.

3. Experiment

3.1. Experiment Setup

Baselines. GNN-MD (Liu et al., 2024) is a simple baseline method where the task is to predict the strongly correlated structures directly from weakly correlated structures using a geometric model. Then we adopt CrystalSDE-VE and CrystalSDE-VP, the two SDE variations of SDE (Song et al., 2020), to model the crystallization as a conditional distribution estimation problem.

Metrics. We consider applying the packing matching (Chisholm & Motherwell, 2005) on the atom-level and mass-center-level for evaluation. The packing matching measures the degree of resemblance between the spatial arrangements of our predicted strongly correlated geometries and the ground-truth crystal geometries.

Hyperparameters. In CrystalFlow, one of the most important hyper-parameters is the option of the interpolation path, and our experiment carefully compared this in the main results. For the cutoff threshold, we consider $c \in \{20, 50\}$.

3.2. Main Results

The main results are in Tables 1 and 2. An interesting observation is the Flow Matching framework performs much better compared to the DDPM framework. We test one CrystalFlow-LERP and two Diffusion models, VE and VP. Among all six datasets and 12 metrics, LERP reaches consistently better packing matching than the two Diffusion variants. This aligns with our intuition that the diffusion path, which estimates the probabilistic trajectory from a random Gaussian to the target data distribution, is not as suitable as Flow Matching for the crystallization task.

4. Conclusion

This study begins with a comprehensive and concise review of the motivation and definition of machine learning for crystallization. We then introduce a dataset named COD-Cluster17. Based on this dataset, we propose CrystalFlow, an SE(3)-equivariant framework aiming for estimating the crystallization process. We thoroughly evaluate the CrystalFlow framework on COD-Cluster17 and demonstrate its potential for both the machine learning and chemistry communities. Although the generalization performance remains an open challenge, we assert that COD-Cluster17 and the CrystalFlow framework serve as valuable benchmarks. Enhancing the CrystalFlow framework for practical applications will require collaborative efforts from both the machine learning and chemistry communities.

References

Albergo, M. S. and Vanden-Eijnden, E. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Chen, R. T. and Lipman, Y. Riemannian flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023.

Chen, R. T. and Lipman, Y. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Chisholm, J. A. and Motherwell, S. Compack: a program for identifying crystal structure similarity using distances. *Journal of applied crystallography*, 38(1):228–231, 2005.

Davtyan, A., Sameni, S., and Favaro, P. Efficient video prediction via sparsely conditioned flow matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23263–23274, 2023.

Downs, R. T. and Hall-Wallace, M. The american mineralogist crystal structure database. *American Mineralogist*, 88(1):247–250, 2003.

Dunn, I. and Koes, D. R. Mixed continuous and categorical flow matching for 3d de novo molecule generation. *arXiv preprint arXiv:2404.19739*, 2024.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jiao, R., Huang, W., Lin, P., Han, J., Chen, P., Lu, Y., and Liu, Y. Crystal structure prediction by joint equivariant diffusion. *Advances in Neural Information Processing Systems*, 36, 2024.

Jing, B., Berger, B., and Jaakkola, T. Alphafold meets flow matching for generating protein ensembles. *arXiv preprint arXiv:2402.04845*, 2024.

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.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Liu, S., Du, W., Li, Y., Li, Z., Zheng, Z., Duan, C., Ma, Z.-M., Yaghi, O. M., Anandkumar, A., Borgs, C., Chayes, J. T., Guo, H., and Tang, J. Symmetry-informed geometric representation for molecules, proteins, and crystalline materials. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=ygXSNrIU1p.

Liu, S., Du, W., Li, Y., Li, Z., Bhethanabotla, V., Rampal, N., Yaghi, O., Borgs, C., Anandkumar, A., Guo, H., et al. A multi-grained symmetric differential equation model for learning protein-ligand binding dynamics. *arXiv preprint arXiv:2401.15122*, 2024.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Luo, Y., Liu, C., and Ji, S. Towards symmetry-aware generation of periodic materials. *Advances in Neural Information Processing Systems*, 36, 2024.

Raccuglia, P., Elbert, K. C., Adler, P. D., Falk, C., Wenny, M. B., Mollo, A., Zeller, M., Friedler, S. A., Schrier, J., and Norquist, A. J. Machine-learning-assisted materials discovery using failed experiments. *Nature*, 533(7601): 73–76, 2016.

Sanchez-Lengeling, B. and Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

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

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Vaitkus, A., Merkys, A., Sander, T., Quirós, M., Thiessen, P. A., Bolton, E. E., and Gražulis, S. A workflow for deriving chemical entities from crystallographic data and its application to the crystallography open database. *Journal of Cheminformatics*, 15(1):123, 2023.

Villani, C. et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

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

Wang, H., Zhang, L., Han, J., and Weinan, E. Deepmd-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Computer Physics Communications*, 228:178–184, 2018.

Ward, L., Agrawal, A., Choudhary, A., and Wolverton, C. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2(1):1–7, 2016.

Wu, L., Wang, D., Gong, C., Liu, X., Xiong, Y., Ranjan, R., Krishnamoorthi, R., Chandra, V., and Liu, Q. Fast point cloud generation with straight flows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9445–9454, 2023.

Yang, M., Cho, K., Merchant, A., Abbeel, P., Schuurmans, D., Mordatch, I., and Cubuk, E. D. Scalable diffusion for materials generation. *arXiv preprint arXiv:2311.09235*, 2023.

Yim, J., Campbell, A., Foong, A. Y., Gastegger, M., Jiménez-Luna, J., Lewis, S., Satorras, V. G., Veeling, B. S., Barzilay, R., Jaakkola, T., et al. Fast protein backbone generation with se (3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023.

A. Related Work

Flow Matching. Flow Matching (Lipman et al., 2022), Rectified Flow (Liu et al., 2022), and Stochastic Interpolants (Albergo & Vanden-Eijnden, 2022) were originally proposed for modeling data distribution. It has shown two advantages: better convergence performance and flexible interpolation between any two distributions. In this work, our proposed CrystalFlow is mainly focusing on the second utilization. Meanwhile, Flow Matching has been widely used in many different applications, including but not limited to image generation (Chen & Lipman, 2024; Lipman et al., 2022), video prediction (Davtyan et al., 2023), point cloud generation (Wu et al., 2023), manifold data generation (Chen & Lipman, 2023), and molecule and protein generation (Dunn & Koes, 2024; Jing et al., 2024; Yim et al., 2023).

ML for crystal structure prediction. We would like to highlight a closely related but fundamentally different research line, ML for crystal structure prediction (CSP). ML for CSP utilizes generative models to predict strongly correlated positions. Existing ML works can be split into two main venues: (1) Generate the crystal structure from scratch (Luo et al., 2024; Yang et al., 2023). (2) Generate the crystal structure given the molecular compositions (Jiao et al., 2024; Yang et al., 2023). In contrast, this work introduces **a novel third category**: (3) Generate the crystal structure given the weakly correlated structures, *i.e.*, the crystallization process.

ML for crystallization. To the best of our knowledge, our work is the first to apply machine learning algorithms (Flow Matching) to tackle the distribution shift in molecular crystallization problems, specifically the transition from a distribution of weakly correlated structures to strongly correlated structures. Along the ML for CSP research line, our CrystalFlow framework possesses three fundamental differences from existing ML methods. (1) The first fundamental difference is the task design. In existing ML methods for CSP, the condition is the composition or without any condition; while in ML for crystallization, the condition is the weakly correlated geometry. (2) Existing ML for CSP works are modeling the atom-level geometry inside each crystal structure, and our work utilizes the geometry on clusters of molecules instead of the periodic information in the crystals. (3) Existing ML methods either generate the atomic coordinates conditioned on the crystal lattice (Luo et al., 2024; Yang et al., 2023), or generate the atomic coordinates and crystal lattice iteratively during the training and inference process (Jiao et al., 2024). However, in reality, the periodic structure of a lattice only emerges after the crystallization process itself. In other words, no crystal lattice information should be considered during the crystallization process, and we follow this criterion in our design of ML for crystallization.

B. Background: Flow Matching and Conditional Flow Matching

Flow Matching and Conditional Flow Matching (Lipman et al., 2022) essentially formulate the density estimation problem as learning a vector field that determines a probability path: it maps a simple distribution to the target data distribution. In this section, we provide a brief description of the algorithm and recommend referring to the original paper for more details. We also acknowledge two parallel works that offer similar ideas: Rectified Flow (Liu et al., 2022), or Stochastic Interpolants (Albergo & Vanden-Eijnden, 2022).

B.1. Key Concepts

Let us first introduce several key concepts.

Data. Assume the each data point has dimension of d, *i.e.*, $\boldsymbol{x} \in \mathbb{R}^d$.

Vector Field. A time-dependent vector field $\boldsymbol{v} : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$.

Flow. A time-dependent vector field defines a time-dependent diffeomorphic map, called flow, $\phi : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$. The vector field defines flow via an ordinary differential equation as

$$\begin{aligned} \frac{d}{dt}\phi_t(\boldsymbol{x}) &= v_t(\phi_t(\boldsymbol{x})) \\ \phi_0(\boldsymbol{x}) &= \boldsymbol{x}. \end{aligned} \tag{10}$$

To put this under the physics context, this is essentially saying that at time 0, we have a data point at position $\boldsymbol{x}$. Then at time t, we move the data point to position $\phi_t(\boldsymbol{x})$.

Probability density path. A probability density path $p : [0, 1] \times \mathbb{R}^d \to \mathbb{R}_{>0}$. Existing flow model (Chen et al., 2018) maps a prior distribution p_0 to the target distribution p_t with push-forward equation or change of variable rule:

$$p_t(\boldsymbol{x}) = [\phi_t]_* p_0(\boldsymbol{x}) = p_0(\phi_t^{-1}(\boldsymbol{x}))\text{det}\Big|\frac{d\phi_t(\boldsymbol{x})}{dx}\Big|^{-1}. \tag{11}$$

Connection. Suppose we have a vector field $\boldsymbol{v}_t$ determines a flow ϕ_t as Equation (10), and the flow ϕ_t and probability path p_t satisfy Equation (11), then we say this vector field $\boldsymbol{v}_t$ generates a probability density p_t.

B.2. Flow Matching

Based on these definitions, a straightforward way is to learn the vector field directly, which gives us the objective function of Flow Matching:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\boldsymbol{x}}\Big[\Big\|u_t(\boldsymbol{x}) - v_\theta(\boldsymbol{x}, t)\Big\|^2\Big], \tag{12}$$

where v_θ is the learnable neural network.

B.3. Conditional Flow Matching

Suppose we have data sample $\boldsymbol{x}_1$, and we define a probability path $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$ s.t. when $t = 0$, p_t is a simple prior distribution like Gaussian, and when $t = 1$, p_t is a Gaussian distribution centered around the target data. Then we can have a marginal probability path as:

$$p_t(\boldsymbol{x}_t) = \int p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)q(\boldsymbol{x}_1)d\boldsymbol{x}_1, \tag{13}$$

where at time $t = 1$, the marginal probability p_1 approximates the data distribution q, as:

$$p_1(\boldsymbol{x}_1) = \int p_1(\boldsymbol{x}_1|\boldsymbol{x}_1)q(\boldsymbol{x}_1)d\boldsymbol{x}_1 \approx q(\boldsymbol{x}). \tag{14}$$

We can also define a marginal vector field:

$$u_t(\boldsymbol{x}_t) = \int u_t(\boldsymbol{x}_t|\boldsymbol{x}_1)\frac{p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)q(\boldsymbol{x}_1)}{p_t(\boldsymbol{x}_t)}. \tag{15}$$

where $u(\cdot|\boldsymbol{x}_1)$ is a conditional vector field that generates $p_t(\cdot|\boldsymbol{x}_1)$.

We want to prove that: Given a conditional vector field $u_t(\boldsymbol{x}|\boldsymbol{x}_1)$ that generates conditional probability paths $p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)$. Then we can see that the marginal vector field $u(\boldsymbol{x}_t)$ (in Equation (15)) can generate the marginal probability path $p(\boldsymbol{x}_t)$ (in Equation (13)).

Proof. We first need to recall the *continuity equation* (Villani et al., 2009). It provides a necessary and sufficient condition to ensure that a vector field u_t generates probability path p_t as long as they satisfy the continuity equation:

$$\frac{\partial p_t(\boldsymbol{x}_t)}{\partial t} = -\text{div}_{\boldsymbol{x}_t}\big(p_t(\boldsymbol{x}_t)u_t(\boldsymbol{x}_t)\big), \tag{16}$$

where div is the divergence operator. Similarly, a condition vector field generates a conditional probability path as long as they satisfy the following:

$$\frac{\partial p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)}{\partial t} = -\text{div}_{\boldsymbol{x}_t}\big(p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)u_t(\boldsymbol{x}_t|\boldsymbol{x}_1)\big). \tag{17}$$

To adapt this to our proof, we have:

$$\begin{aligned}
\frac{\partial p_t(\boldsymbol{x}_t)}{\partial t} &= \frac{\partial \int p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)q(\boldsymbol{x}_1)d\boldsymbol{x}_1}{\partial t} \\
&= \int \frac{\partial p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)}{\partial t} q(\boldsymbol{x}_1)d\boldsymbol{x}_1 && \text{// Leibniz Rule} \\
&= -\text{div}_{\boldsymbol{x}_t} \int p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)u_t(\boldsymbol{x}_t|\boldsymbol{x}_1)q(\boldsymbol{x}_1)d\boldsymbol{x}_1 && \text{// Equation (17)} \\
&= -\text{div}_{\boldsymbol{x}_t}\big(u_t(\boldsymbol{x}_t)p_t(\boldsymbol{x}_t)\big). && \text{// Leibniz Rule}
\end{aligned} \tag{18}$$

Thus, we can verify that the vector field $u_t(\boldsymbol{x}_t)$ generates the probability $p_t(\boldsymbol{x}_t)$. □

This reveals that constructing the conditional vector field is equivalent to constructing the vector field. And by plugging this into Equation (12), we have the following objective for Conditional Flow Matching:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\boldsymbol{x},\boldsymbol{x}_1}\Big[\Big\|u_t(\boldsymbol{x}|\boldsymbol{x}_1) - v_\theta(\boldsymbol{x},\boldsymbol{x}_1,t)\Big\|^2\Big]. \tag{19}$$

B.4. Gaussian Probability Path

A simple example provided in (Lipman et al., 2022) is a Gaussian conditional probability path, namely having conditional probability paths that maintain the shape of a Gaussian:

$$p_t(\boldsymbol{x}|\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}|\mu_t(\boldsymbol{x}_1), \sigma_t(\boldsymbol{x}_1)^2 I). \tag{20}$$

For $t = 0$, we set $\mu_0(\boldsymbol{x}_1) = 0$ and $\sigma_0(\boldsymbol{x}_1) = 1$, *i.e.*, all probability paths converge to a same Gaussian distribution, $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|0, I)$. For $t = 1$, we set $\mu_1(\boldsymbol{x}_1) = \boldsymbol{x}_1$ and $\sigma_1(\boldsymbol{x}_1) = \sigma_{\text{min}}$ with a sufficiently small value.

A simple way to define the flow by affine transformation: $\phi_t(\boldsymbol{x}) = \sigma_t(\boldsymbol{x}_1)\boldsymbol{x} + \mu_t(\boldsymbol{x}_1)$, where $\boldsymbol{x} \in \mathcal{N}(0, I)$. This means the flow ϕ_t

$$p_t(\boldsymbol{x}|\boldsymbol{x}_1) = [\phi_t]_* p_0(\boldsymbol{x}|\boldsymbol{x}_1) = [\phi_t]_* p(\boldsymbol{x}). \tag{21}$$

The corresponding vector field is:

$$\frac{d}{dt}\phi_t(\boldsymbol{x}) = u_t(\phi_t(\boldsymbol{x})|\boldsymbol{x}_1). \tag{22}$$

Thus, after reparameterization using $\boldsymbol{x}_0$, the objective of Conditional Flow Matching is

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,\boldsymbol{x},\boldsymbol{x}_0}\Big[\Big\|u_t(\phi_t(\boldsymbol{x}_0)|\boldsymbol{x}_1) - v_\theta(\boldsymbol{x},\boldsymbol{x}_0,t)\Big\|^2\Big]. \tag{23}$$

Notice that here ϕ_t is an affine transformation, thus, there is an analytical form of u_t in Equation (22) as

$$u_t(\phi_t(\boldsymbol{x})|\boldsymbol{x}_1) = \frac{\sigma_t'(\boldsymbol{x}_1)}{\sigma_t(\boldsymbol{x}_1)}\big(\phi_t(\boldsymbol{x}) - \mu_t(\boldsymbol{x}_1)\big) + \mu_t'(\boldsymbol{x}_1). \tag{24}$$

Proof. According to Equation (10), we have

$$\phi_t'(\boldsymbol{x}) = u_t(\phi_t(\boldsymbol{x})|\boldsymbol{x}_1). \tag{25}$$

Recall that $\boldsymbol{x} = \frac{\phi_t(\boldsymbol{x}) - \mu_t(\boldsymbol{x}_1)}{\sigma_t(\boldsymbol{x}_1)}$, and differentiating ϕ_t w.r.t. t leads to $\phi_t'(\boldsymbol{x}) = \sigma_t'(\boldsymbol{x}_1)\boldsymbol{x} + \mu_t'(\boldsymbol{x}_1)$. Injecting both into Equation (25) can give us

$$u_t(\phi_t(\boldsymbol{x})|\boldsymbol{x}_1) = \frac{\sigma_t'(\boldsymbol{x}_1)}{\sigma_t(\boldsymbol{x}_1)}\big(\phi_t(\boldsymbol{x}) - \mu_t(\boldsymbol{x}_1)\big) + \mu_t'(\boldsymbol{x}_1). \tag{26}$$

□

B.5. Diffusion with Variance Exploding

The denoising diffusion probabilistic model (DDPM) (Ho et al., 2020) or denoising score matching model (Vincent, 2011) has been well studied in learning data distributions. More recently, (Song et al., 2020) summarized that both methods can be formulated as solving a stochastic differential equation (SDE), coined *Variance Exploding* and *Variance Preserving*, respectively. Along this line, (Lipman et al., 2022) provides another aspect that both can be formulated under the Conditional Flow Matching framework.

The diffusion model gradually adds noise to the data distribution until it reaches a Gaussian distribution. The Gaussian probability path of Variance Exploding (VE) is:

$$p_t(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_1, \sigma_{1-t}^2 I), \tag{27}$$

where $0 = \sigma_0 \leq ... \leq \sigma_t \leq ...\sigma_1$. For time t, the mean and standard deviation are $\mu_t(\boldsymbol{x}_1) = \boldsymbol{x}_1$ and $\sigma_t(\boldsymbol{x}_1) = \sigma_{1-t}$, respectively. Plugging this into Equation (24) gives us:

$$u_t(\boldsymbol{x}|\boldsymbol{x}_1) = -\frac{\sigma_{1-t}'}{\sigma_{1-t}}(\boldsymbol{x} - \boldsymbol{x}_1). \tag{28}$$

Plugging this into Equation (23), the objective is:

$$\mathcal{L}_{\text{Diffusion-VE}} = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_1}\left[\left\|\frac{\sigma_{1-t}'}{\sigma_{1-t}}(\boldsymbol{x}_1 - \boldsymbol{x}_0) - v(\boldsymbol{x}_t, \boldsymbol{x}_0, t)\right\|^2\right]. \tag{29}$$

B.6. Diffusion with Variance Preserving

For Variance Preserving, the probability path is

$$p_t(\boldsymbol{x}|\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}|\alpha_{1-t}\boldsymbol{x}_1, (1 - \alpha_{1-t}^2)I), \tag{30}$$

where $\alpha_t = e^{-T(t)/2}$, $T(t) = \int_0^t \beta(s)ds$, and β is the noise scale function. With mean $\mu_t(\boldsymbol{x}_1) = \alpha_{1-t}\boldsymbol{x}_1$ and standard deviation $\sigma_t(\boldsymbol{x}_1) = \sqrt{1 - \alpha_{1-t}^2}$. Plugging this into Equation (24) gives us:

$$\begin{aligned} u_t(\boldsymbol{x}|\boldsymbol{x}_1) &= \frac{\alpha_{1-t}\alpha_{1-t}'}{1 - \alpha_{1-t}^2}(\boldsymbol{x} - \alpha_{1-t}\boldsymbol{x}_1) - \alpha_{1-t}'\boldsymbol{x}_1 \\ &= \frac{\alpha_{1-t}'}{1 - \alpha_{1-t}^2}(\alpha_{1-t}\boldsymbol{x} - \boldsymbol{x}_1) \\ &= -\frac{T'(1-t)}{2}\frac{e^{-T(1-t)}\boldsymbol{x} - e^{-T(1-t)/2}\boldsymbol{x}_1}{1 - e^{-T(1-t)}}. \end{aligned} \tag{31}$$

Plugging this into Equation (23), the objective is:

$$\mathcal{L}_{\text{Diffusion-VP}} = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_1}\left[\left\|\frac{T'(1-t)}{2}\frac{e^{-T(1-t)/2}\boldsymbol{x}_1 - e^{-T(1-t)}\boldsymbol{x}_0}{1 - e^{-T(1-t)}} - v(\boldsymbol{x}_t, \boldsymbol{x}_0, t)\right\|^2\right]. \tag{32}$$

For implementation, we follow (Ho et al., 2020; Lipman et al., 2022) by taking $\beta(s) = \beta_{min} + s(\beta_{max} - \beta_{min})$, we can have

$$T(t) = t\beta_{min} + \frac{1}{2}t^2(\beta_{max} - \beta_{min}). \tag{33}$$

B.7. Linear interpolation (LERP)

The optimal transport-based conditional probability path (Lipman et al., 2022) that linearly interpolates between the base and the conditional target distribution has shown promising results. In specific, First let us take the linear interpolation between $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, then we have:

$$\begin{aligned} \mu_t(\boldsymbol{x}) &= (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1 \\ \sigma_t(x) &= 1 - (1-\sigma_{\text{min}})t. \end{aligned} \tag{34}$$

This gives us

$$\begin{aligned} \mu_t'(\boldsymbol{x}) &= \boldsymbol{x}_1 - \boldsymbol{x}_0 \\ \sigma_t'(x) &= \sigma_{\text{min}} - 1. \end{aligned} \tag{35}$$

Plugging this into Equation (24), then we have the conditional velocity as

$$\begin{aligned} u_t(\boldsymbol{x}|\boldsymbol{x}_1) &= \frac{\sigma_t'(\boldsymbol{x}_1)}{\sigma_t(\boldsymbol{x}_1)}\big(\boldsymbol{x} - \mu_t(\boldsymbol{x}_1)\big) + \mu_t'(\boldsymbol{x}_1) \\ &= \frac{(\sigma_{\text{min}}-1)(\boldsymbol{x} - (1-t)\boldsymbol{x}_0 - t\boldsymbol{x}_1) + (1 - t + \sigma_{\text{min}}t)(x_1 - x_0)}{1-(1-\sigma_{\text{min}})t} \\ &= \frac{\boldsymbol{x}_1 - \sigma_{\text{min}}\boldsymbol{x}_0 - (1-\sigma_{\text{min}})\boldsymbol{x}}{1-(1-\sigma_{\text{min}})t}. \end{aligned} \tag{36}$$

Plugging this into Equation (23), the objective is:

$$\mathcal{L}_{\text{LERP}} = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_1}\Big[\Big\|\boldsymbol{x}_1 - \boldsymbol{x}_0 - v(\boldsymbol{x}_t, \boldsymbol{x}_0, t)\Big\|^2\Big]. \tag{37}$$

C. Dataset Construction

To construct the dataset for crystallization modeling, we consider experimentally-verified crystal structures stored in Crystallographic Information File (CIF) from Crystallography Open Database (COD) with open access (Downs & Hall-Wallace, 2003; Vaitkus et al., 2023).

Crystal selection criteria. In this work, we consider a simple setting: how a molecule of a given geometry will pack itself in its crystallized form. We start with 225K CIFs in COD supplied with SMILES strings and the following criteria were applied to the selection of the qualified molecular crystals. (1) Crystals of one type of molecule and a z_prime value no larger than one: z_prime value denotes how many chemical formulas (molecules) are there in one asymmetric unit. This criterion considers molecules that have only one local environment in their crystallized form, where the local environment refers to the spatial relationship between a central molecule and its neighboring molecules. While crystals can contain multiple components and molecules in various local environments, which can be handled by designing more sophisticated models, this criterion simplifies the problem to a manageable level. Applying this criterion results in 133K qualified crystals. (2) Crystals having no disordered atoms: In crystallography, disorder occurs when different atoms can occupy the same position or the same atom can occupy multiple positions. In both scenarios, disordered atoms exhibit occupancy values smaller than one. Such uncertainty in atomic positions and atomic types introduces challenges for machine learning modeling and crystal structure generation. (3) Non-polymeric crystals: Some crystal formations involve polymerization reactions between molecules, resulting in polymers that can extend across infinite unit cells in one, two, or three dimensions. Modeling polymeric crystals requires additional information about the polymerization process, such as which bonds are newly formed and which atoms act as leaving groups. This level of detail is beyond the scope of the current study.

Preprocessing of atomic coordinates The crystal structure information stored in a CIF (Crystallographic Information File) comprises two essential components: symmetry information and atomic coordinates. The symmetry information includes the periodicity (unit cell parameters) and space group. It is important to note that the atomic coordinates in a CIF pertain only to one asymmetric unit. The atomic coordinates of the entire crystal can be derived by applying symmetry operations to those in the asymmetric unit. For machine learning modeling, the atomic coordinates of the asymmetric unit must first be converted to those of the molecules within one unit cell, ensuring the integrity of each molecule remains intact. This conversion is accomplished using a Perl program called cod-tools. Moreover, the above-mentioned detection of polymeric crystals is also achieved by cod-tools.

The effect of conditions. Crystallization conditions are crucial in the molecule crystallization process. Molecules can crystallize from their liquid state, gas state, or from a solution. Depending on the specific conditions, the same molecule can form different crystal structures, known as polymorphs. In our preliminary trials of applying machine learning to crystallization, we do not include crystallization conditions in the dataset. This is because these conditions are typically described textually in the original literature rather than being formatted in CIF files, making them difficult to extract. Instead, we generate a set of plausible crystal structures for a given molecular geometry and verify if any of them match the actual crystal structure.

Cluster of molecules. There are various methods to determine whether two crystal structures are identical. In our approach, we compare the local environment surrounding a central molecule. Since we focus on molecules that have only one local environment in their crystallized form, if the local environments are the same, the crystal structures are considered identical.

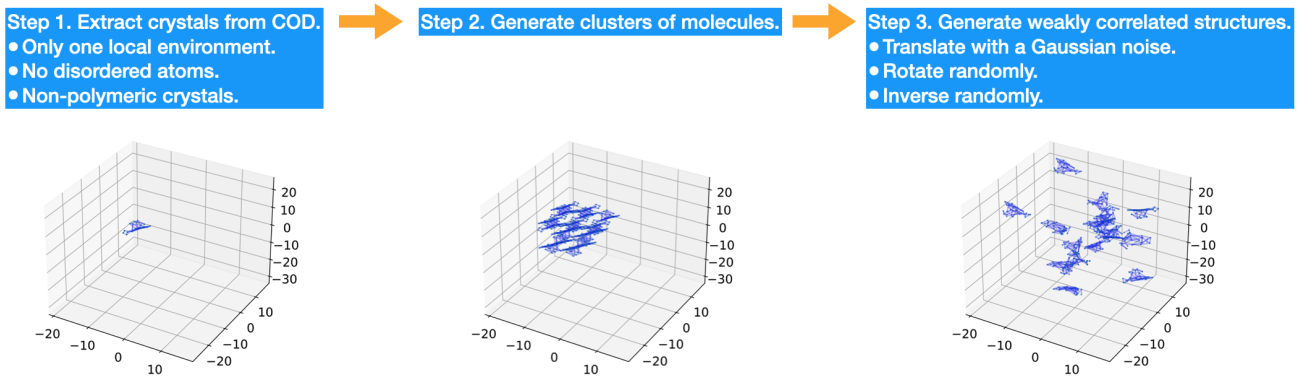


Figure 2. Dataset curation of COD-Cluster17.

To achieve this, we extract a cluster of molecules representing the local environment. Specifically, we select a single molecule and its 16 nearest neighbors, extracting their coordinates in Euclidean space, to create the dataset of final positions.

Two types of COD-Cluster17: with and without inversion. In addition to translation and rotation, another critical symmetry element in crystal structures is inversion. An inversion center can be located either at the center of a molecule or between two molecules. In the first scenario, the molecule itself is symmetric. In the second scenario, the two molecules have identical connectivity but are mirror images of each other. These can be either left-handed and right-handed chiral molecules or the left-handed and right-handed geometries of an achiral molecule. The former are known as optical isomers and cannot interconvert during crystallization, while the latter arise from flexibility in geometry and can interconvert during crystallization. To account for second type of inversion, when noise is added to the final positions, we also introduce a fifty percent chance of inverting each molecule, creating an additional dataset incorporating inversion noise.

D. Dataset Statistics

In this section, we list certain key statistics of COD-Cluster17.

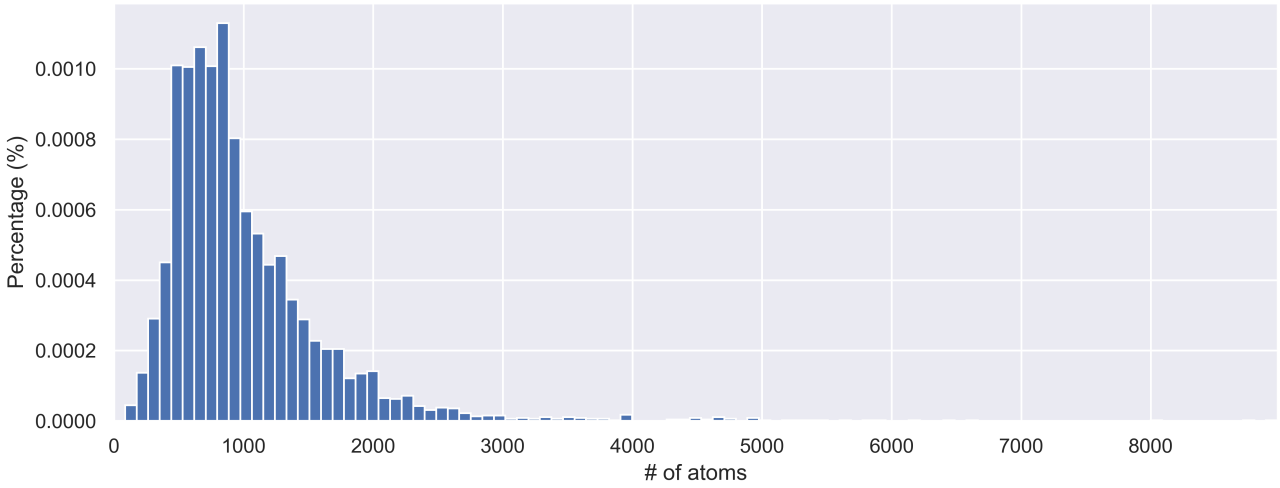
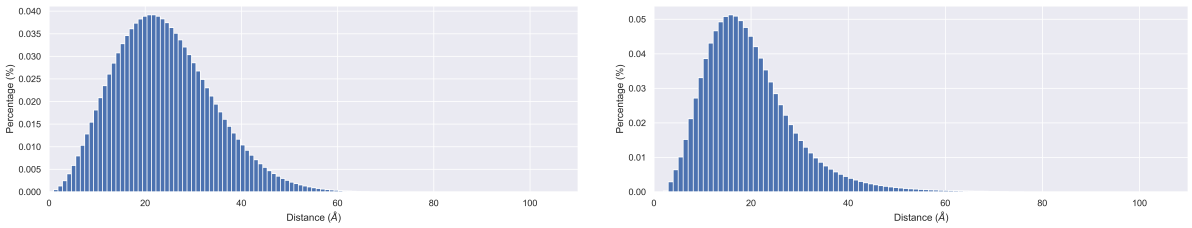
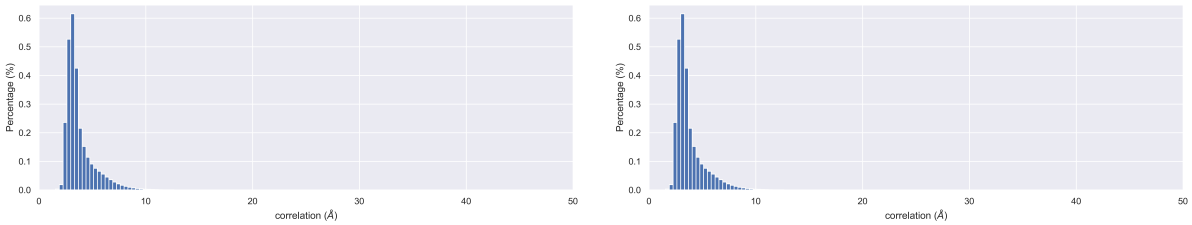


Figure 3. Distribution on the number of atoms in COD-Cluster17.



(a) Weakly correlated structures. (b) Strongly correlated structures.

Figure 4. Atom pairwise distance distribution on COD-Cluster17.



(a) Weakly correlated structures. (b) Strongly correlated structures.

Figure 5. Correlation distribution on the atom level. (sampled 5K molecules)