JAMUN: Bridging Smoothed Molecular Dynamics and Score-Based Learning for Conformational Ensemble Generation

Ameya Daigavane [*1] **Bodhi P. Vani** [*2] **Darcy Davidson** [2] **Saeed Saremi** [2] **Joshua A. Rackers** [†2] **Joseph Kleinhenz** [†2]

Abstract

Conformational ensembles of protein structures are immensely important both to understanding protein function, and for drug discovery in novel modalities such as cryptic pockets. Current techniques for sampling ensembles such as molecular dynamics are computationally inefficient. On the other hand, many recent machine-learning methods do not generalize well outside their training data. We propose JAMUN, that performs MD in a smoothed, noised space of all-atom 3D conformations of molecules by utilizing the framework of walk-jump sampling. JAMUN enables ensemble generation at orders of magnitude faster rates than traditional molecular dynamics or state-of-the-art ML methods. This physical prior enables JAMUN to transfer to systems outside its training data. JAMUN is even able to generalize across length scales it was not trained on.

1. Introduction

Proteins are inherently dynamic entities constantly in motion, and these movements can be vitally important. They are not well characterized as single structures as has traditionally been the case, but rather as ensembles of structures drawn from the Boltzmann distribution (Henzler-Wildman & Kern, 2007). Protein dynamics is required for the function of most proteins, for instance the global tertiary structure motions for myglobin to bind oxygen and move it around the body (Miller & Phillips, 2021), or the beta-sheet transition to a disordered strand for insulin to dissociate and find and bind to its receptor (Antoszewski et al., 2020). Similarly, drug discovery on protein kinases depends on characterizing kinase conformational ensembles (Gough & Kalodimos, 2024). In general the search for druggable 'cryptic pockets' requires understanding protein dynamics (Colombo, 2023), and antibody design is deeply affected by conformational ensembles (Fernández-Quintero et al., 2023). However, while machine learning (ML) methods for molecular structure prediction have experienced enormous success recently, ML methods for dynamics have yet to have similar impact. ML models for generating molecular ensembles are widely considered the 'next frontier' (Bowman, 2024; Miller & Phillips, 2021; Zheng et al., 2023). In this work, we present JAMUN (Walk-**J**ump **A**ccelerated **M**olecular ensembles with **U**niversal **N**oise), a generative ML model which advances this frontier by demonstrating improvements in both speed and transferability over previous approaches.

While the importance of protein dynamics is well-established, it can be exceedingly difficult to sufficiently sample large biomolecular systems. The most common sampling method is molecular dynamics (MD), which will sample the Boltzmann distribution in the limit of infinite sampling time, but is limited by the need for very short timesteps of 1-2 femtoseconds in the numerical integration scheme. Many important protein dynamic phenomena occur on the timescale of milliseconds. As described by Borhani & Shaw (2012), simulating with this resolution is '…equivalent to tracking the advance and retreat of the glaciers of the last Ice Age – tens of thousands of years – by noting their locations each and every second.' Importantly, there is nothing fundamental about this small time-step limitation; it is an artifact of high-frequency motions, such as bond vibrations, that have little effect on protein ensembles (Leimkuhler & Matthews, 2015). Enhanced sampling methods have been developed in an attempt to accelerate sampling, but they often require domain knowledge about relevant collective variables, and, more importantly, do not address the underlying time-step problem (Vitalis & Pappu, 2009).

A large number of generative models have been developed to address the sampling inefficiency problems of MD using machine learning, which we discuss in greater detail in Section 4. The key requirement is that of *transferability*: any model must be able to generate conformational ensembles for molecules that are significantly different from those

[1]Massachusetts Institute of Technology (MIT), Cambridge, MA, USA [2]Prescient Design, Genentech, South San Francisco, CA, USA. Correspondence to: Ameya Daigavane <ameyad@mit.edu>, Bodhi P. Vani <vanib@gene.com>.

Proceedings of the Workshop on Generative AI for Biology at the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

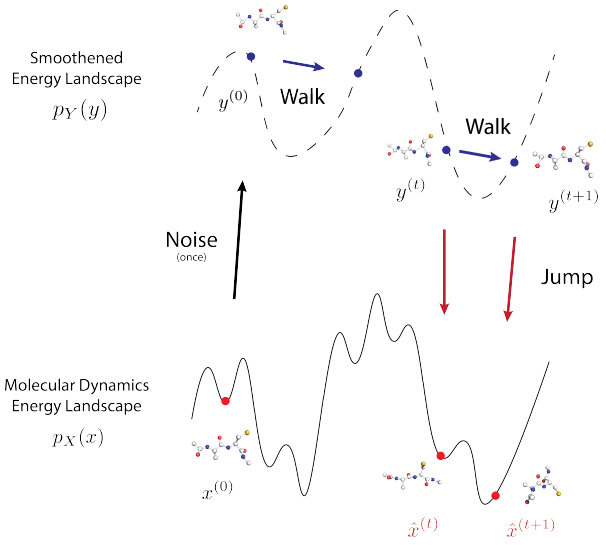


Figure 1. Overview of walk-jump sampling in JAMUN.

in its training set. To benchmark this transferability, we focus on small peptides whose MD trajectories can be run to convergence in a reasonable amount of time, unlike those for much larger proteins (Shaw et al., 2010).

Here, we propose a new method, JAMUN, that bridges molecular dynamics with score-based learning in a latent space. This physical prior enables JAMUN to transfer well – just like force fields for molecular dynamics can – to unseen systems. The key idea is to run Langevin molecular dynamics in a noisy 'latent' space $Y \in \mathbb{R}^{N\times 3}$, instead of the original space $X \in \mathbb{R}^{N\times 3}$ of all-atom 3D positions. Indeed, Y is constructed by adding a small amount of i.i.d Gaussian noise ε to X:

$$Y = X + \sigma\varepsilon \tag{1}$$

To run this Langevin MD over Y, the crucial component is the score function $\nabla_y \log p_Y(y)$, which needs to be modelled. (This is identical to how a force field needs to be parametrized in classical MD). Once the MD trajectory over Y is run, the resulting samples need to be mapped back to clean data via a denoising procedure.

This framework is actually mathematically described by Walk-Jump Sampling (WJS), as first introduced by Saremi & Hyvärinen (2019). WJS has been used in voxelized molecule generation (Pinheiro et al., 2024a;b) and protein sequence generation (Frey et al., 2024). In particular, JAMUN corresponds to a $SE(3)$-equivariant Walk-Jump sampler of point clouds.

The WJS framework actually tells us that the score function $\nabla_y \log p_Y(y)$ can be used for denoising as well, removing the need to learn a separate model. Indeed, this mathematical connection is used to learn the score function, similar to the training of diffusion models. Unlike diffusion models such as DDPM (Ho et al., 2020), however, we only need to learn the score function at a single noise level. The choice of this noise level is important; we aim to simply smooth out the distribution enough to resolve sampling difficulties without fully destroying the information present in the data distribution.

In short, the score function $\nabla_y \log p_Y(y)$ is learned by adding noise to clean data x, and a denoising neural network is trained to recover the clean samples x from y. This denoiser defines the score function of the noisy manifold Y which we sample using Langevin dynamics (walk step) and allows us to periodically project back to the original data distribution (jump step). Crucially, the walk and jump steps are *decoupled* from each other.

Rather than starting over from an uninformative prior for each sample as is commonly done in diffusion (Ho et al., 2020; Song et al., 2022) and flow-matching (Lipman et al., 2023; Klein et al., 2024b), JAMUN is able to simply denoise samples from the slightly noised distribution, enabling much greater sampling efficiency.

We train JAMUN on a large dataset of MD simulations of small peptides. We demonstrate that this model can generalize to a holdout set of unseen peptides. In all of these cases, generation with JAMUN yields converged sampling of the conformational ensemble faster than MD with a standard force field, even outperforming several state-of-the-art baselines. These results suggest that this transferability is a consequence of retaining the physical priors inherent in MD data. Significantly, we find that JAMUN performs well even for peptides longer than the ones seen in the training set.

2. Methods

2.1. Representing Peptides as Point Clouds

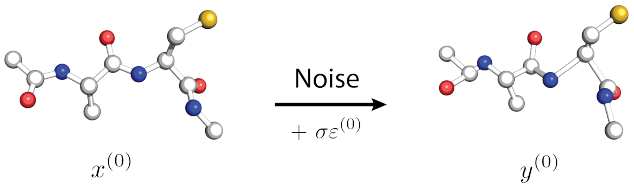
Each point cloud of N atoms can be represented by the tuple (x, h) where $x \in \mathbb{R}^{N\times 3}$ represents the 3D coordinates of each of the N atoms and $h \in \mathbb{R}^D$ represents atom and covalent bonding information. h can be easily computed from the amino acid sequence for each peptide, and hence is not learned or sampled. For clarity of presentation, we omit the conditioning on h in the distributions and models below. We discuss how our model uses h in Section 2.4.

At sampling time, we assume access to an initial sample $x^{(0)} \in \mathbb{R}^{N\times 3}$ sampled from the clean data distribution p_X. Similarly to how MD simulations of small peptides are commonly seeded, we use the `sequence` command in the LEaP program packaged with the Amber force fields to procedurally generate $x^{(0)}$. In theory, $x^{(0)}$ could also be obtained from experimental data, such as crystallized structures from the Protein Data Bank (Berman et al., 2000). We

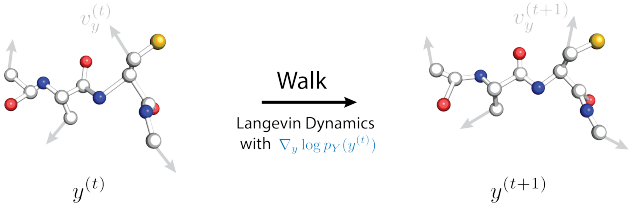
plan to explore this approach in the future to seed JAMUN simulations for larger proteins.

2.2. Walk-Jump Sampling

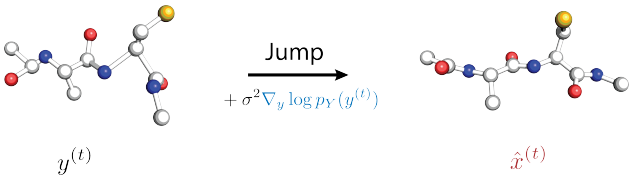
JAMUN operates by performing walk-jump sampling on molecular systems represented as 3D point clouds. A conceptual overview of the process is illustrated in Figure 1, with specific steps depicted in Figure 2.



(a) Adding noise to an initial conformation $x^{(0)}$ to obtain $y^{(0)} \sim p_Y$.



(b) One iteration of BAOAB-discretized Langevin dynamics (Equation 3 and Equation 68) starting from $y^{(t)} \sim p_Y$ leads to a new sample $y^{(t+1)} \sim p_Y$.



(c) Denoising of $y^{(t)}$ according to Equation 7 gives us new samples $\hat{x}^{(t)}$.

Figure 2. Illustrating each of the noise (Figure 2a), walk (Figure 2b) and jump steps (Figure 2c) in walk-jump sampling.

Given the initial sample $x^{(0)} \sim p_X$, walk-jump sampling performs the following steps:

1. **Noise** the initial structure $x^{(0)}$ to create the initial sample $y^{(0)}$ from the noisy data distribution p_Y (Figure 2a):

$$y^{(0)} = x^{(0)} + \sigma\varepsilon^{(0)}, \text{ where } \varepsilon^{(0)} \sim \mathcal{N}(0, \mathbb{I}_{N\times 3}). \quad (2)$$

2. **Walk** to obtain samples $y^{(1)}, \ldots, y^{(N)}$ from p_Y using Langevin dynamics which conists of numerically solving the following Stochastic Differential Equation (SDE) (Figure 2b):

$$dy = v_y dt, \quad (3)$$

$$dv_y = \nabla_y \log p_Y(y)dt - \gamma v_y dt + M^{-\frac{1}{2}}\sqrt{2}dB_t, \quad (4)$$

where v_y represents the particle velocity, $\nabla_y \log p_Y(y)$ is the gradient of the log of the probability density function (called the score function) of p_Y, γ is friction, M is the mass, and B_t is the standard Wiener process in $N \times 3$-dimensions: $B_t \sim \mathcal{N}(0, t\mathbb{I}_{N\times 3})$. In practice, we employ the BAOAB solver (Appendix F) to integrate Equation 3 numerically.

3. **Jump** back to p_X to obtain samples $\hat{x}_1, \ldots, \hat{x}_N$ (Figure 2c) :

$$\hat{x}_i = \hat{x}(y_i) = \mathbb{E}[X \mid Y = y_i], \quad (5)$$

where $\hat{x}(\cdot) \equiv \mathbb{E}[X \mid Y = \cdot]$ is called the denoiser. It corresponds to the minimizer (Section E.1) of the ℓ_2-loss between clean samples X and samples denoised back from $Y = X + \sigma\varepsilon$.

$$\hat{x}(\cdot) = \arg\min_f \mathbb{E}_{\substack{X\sim p_X, \varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3}) \\ Y = X+\sigma\varepsilon}} [\|f(Y) - X\|^2], \quad (6)$$

where $f : \mathbb{R}^{N\times 3} \to \mathbb{R}^{N\times 3}$. As shown by Robbins (1956); Miyasawa (1960) (and Section E.2), the denoiser $\hat{x}$ is closely linked to the score $\nabla_y \log p_Y$:

$$\hat{x}(y) = y + \sigma^2 \nabla_y \log p_Y(y). \quad (7)$$

Importantly, the score function $\nabla_y \log p_Y$ shows up in both the **walk** and **jump** steps, and we need to approximate this quantity.

2.3. Learning to Denoise

In order to run Walk-Jump Sampling as outlined above, we have the choice of modelling either the score $\nabla_y \log p_Y$ or the denoiser $\hat{x}$ as they are equivalent by Equation 7. Following trends in diffusion models (Karras et al., 2022; 2024), we model the denoiser as a neural network $\hat{x}_\theta(y, \sigma) \approx \hat{x}(y)$ parameterized by model parameters θ.

Importantly, we only need to learn a model at a **single, fixed noise level** σ. This is unlike training diffusion or flow-matching models where a wide range of noise levels are required for sampling. In particular, the choice of noise level σ for WJS is important because mode-mixing becomes faster as σ is increased, but the task asked of the denoiser becomes harder.

The denoiser $\hat{x}_\theta$ thus takes in noisy point clouds y formed by adding noise (at a fixed noise level σ) to clean point clouds

x. The denoiser is tasked to reconstruct back x, given y. To be precise, training the denoiser $\hat{x}_\theta$ consists of solving the following optimization problem:

$$\theta^* = \arg\min_\theta \mathbb{E}_{\substack{X \sim p_X, \varepsilon \sim \mathcal{N}(0, \mathbb{I}_{N\times 3}) \\ Y = X + \sigma\varepsilon}} \|\hat{x}_\theta(Y, \sigma) - X\|^2 \quad (8)$$

to obtain θ^*, the optimal model parameters. As is standard in the empirical risk minimization (ERM) (Vapnik, 1991) setting, we approximate the expectation in Equation 8 by sampling $X \sim p_X$ and $\varepsilon \sim \mathcal{N}(0, \mathbb{I}_{N\times 3})$. We minimize the loss as a function of model parameters θ using the first-order optimizer Adam (Kingma & Ba, 2017) in PyTorch 2.0 (Ansel et al., 2024; Falcon & The PyTorch Lightning team, 2019).

2.4. Parametrization of the Denoiser Network

We summarize the key features of the denoiser network $\hat{x}_\theta(y, \sigma)$ which will approximate $\hat{x}(y)$ in this section. Note that σ is fixed in our setting, but we explicitly mention it in this section for clarity. A diagramatic overview of our model along with specific hyperparameters are presented in Appendix C.

We utilize the same parametrization of the denoiser as originally proposed by Karras et al. (2022; 2024) (in the context of image generation):

$$\hat{x}_\theta(y, \sigma) = c_{\text{skip}}(\sigma)y + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)y, c_{\text{noise}}(\sigma)), \quad (9)$$

where F_θ represents a learned network parameterized by parameters θ. In particular, F_θ (Figure 16) is a geometric graph neural network (GNN) model similar to NequIP (Batzner et al., 2022; Thomas et al., 2018). Importantly, F_θ is chosen to be $SE(3)$-equivariant, in contrast to existing methods (Hoogeboom et al., 2022; Klein & Noé, 2024; Klein et al., 2024a) that utilize the $E(3)$-equivariant EGNN model (Satorras et al., 2022). As rightly pointed out by Dumitrescu et al. (2024) and Schreiner et al. (2023), $E(3)$-equivariant models are equivariant under parity, which means that are forced to transform mirrored structures identically. When we experimented with $E(3)$-equivariant architectures, we found symmetric Ramachandran plots which arise from the unnecessary parity constraint of the denoising network. For this reason, TBG (Klein & Noé, 2024) and Timewarp (Klein et al., 2024a) use a 'chirality checker' to post-hoc fix the generated structures from their model. For JAMUN, such post-processing is unnecessary because our model can distinguish between chiral structures.

The coefficients $c_{\text{skip}}(\sigma), c_{\text{out}}(\sigma), c_{\text{in}}(\sigma), c_{\text{noise}}(\sigma)$ in Equation 9 are normalization functions (from $\mathbb{R}^+$ to $\mathbb{R}$) which adjust the effective inputs and outputs to F_θ. They are chosen to encourage re-use of the input y at low noise levels, but the opposite at high noise levels. Importantly, based on the insight that F_θ uses relative vectors in the message passing steps, we adjust the values of these coefficients instead of simply using the choices made in Karras et al. (2022; 2024); Wohlwend et al. (2024); Abramson et al. (2024), as discussed in Appendix D.

In F_θ, edges between atoms are computed using a radial cutoff over the noisy positions in y. The edge features are a concatenation of a one-hot feature indicating bonded-ness and the radial distance embedded using Bessel functions. As obtained from h, atom-level features are computed using the embedding of the atomic number (eg. C and N), and the atom name following PDB notation (eg. CA, CB for alpha and beta carbons). Similarly, residue-level features are obtained using the embedding of the residue code (eg. ALA, CYS) and concatenated to each atom in the residue. Importantly, we *do not use the sequence index of the residues* (eg. $0, 1, \ldots$) as we found that it hurts generalization to longer peptide lengths.

3. Datasets

For development, demonstration, and benchmarking against existing models, we use three different datasets consisting of peptides from 2 to 5 amino acids (AA) long: TIMEWARP 2AA-LARGE and TIMEWARP 4AA-LARGE from Klein et al. (2024a), MDGEN 4AA EXPLICIT from Jing et al. (2024), and our own MD data simulated with OpenMM (Eastman et al., 2017). A summary of these datasets is presented in Table 1. The differing simulation conditions across these datasets allows us to test the broad applicability of our approach.

These TIMEWARP and MDGEN datasets consist of 'uncapped' peptides, whose termini are zwitterionic amino and carboxyl groups, as shown in the left panel of Figure 3. These are not ideal analogues of amino acids in proteins due to local charge interactions as well as lack of steric effects.

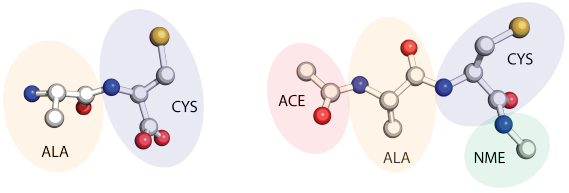


Figure 3. A side-by-side comparison of uncapped (left) compared to capped (right) ALA-CYS. The acetyl (ACE) and N-methyl (NME) capping groups provide steric hindrance and prevent local charge interactions on the N-terminal and C-terminal ends.

We also create a similar dataset called CAPPED 2AA of 2AA peptides by adding ACE (acetyl) and NME (N-methyl amide) caps, a common practice in molecular dynamics simulations of very small peptides. As illustrated in the right panel of Figure 3, these caps introduce additional peptide bonds with the first and last residues. These peptide bonds remove the need for the zwitterion, while the methyl group

Dataset	Peptide Length	Capped?	Force Field	Solvent Model	Temperature	Train Split	Validation Split	Test Split
TIMEWARP 2AA-LARGE	2	✗	amber14	Implicit	310 K	200	80	100
CAPPED 2AA	2	✓	amber14	Explicit	300 K	200	80	100
TIMEWARP 4AA-LARGE	4	✗	amber14	Implicit	310 K	1459	379	182
MDGEN 4AA-EXPLICIT	4	✗	amber14	Explicit	350 K	3109	100	100
UNCAPPED 5AA	5	✗	amber14	Implicit	310 K	–	–	3

Table 1. Simulation conditions of the different datasets investigated in this work.

provides some steric interactions. These capping groups increase the complexity of the modelling task, but ensure a more realistic distribution of conformations. We choose the same splits as in TIMEWARP 2AA-LARGE. Since we simulated this data ourselves, we can also measure the wall-clock speed-ups of JAMUN relative to MD on this dataset.

For the 2AA datasets, the training set consists of 50% of all possible 2AA peptides. For the 4AA datasets, the generalization task is much harder, because the number of 4AA peptides in the training sets is less than 1% and 2% respectively of the total number of possible 4AA peptides.

We also use the implicit MD code from Timewarp to generate trajectories for three randomly picked 5AA peptides with codes NRLCQ, VWSPF and KTYDI. We call this dataset UNCAPPED 5AA. Further details on MD simulation conditions can be found in Appendix B.

4. Related Work and Baseline Models

The goal of building machine learning models that can generate conformational ensembles of molecular systems is not new. While a full overview of this field is beyond the scope of this work – see (Aranganathan et al., 2024) for a recent review – we note a few relevant previous efforts. Boltzmann Generators (Noé et al., 2019) introduced the idea that a neural network could be used to transform the underlying data distribution into an easier-to-sample Gaussian distribution. DiffMD (Wu & Li, 2023) learns a diffusion model over conformations of small organic molecules from MD17 (Chmiela et al., 2017), showing some level of transferability across $C_7O_2H_{10}$ isomers (Schütt et al., 2017). Timewarp (Klein et al., 2024a) uses a normalizing flow as a proposal distribution in MCMC sampling of the Boltzmann distribution to approximate the conditional distribution of future conformational states $x^{(t+\Delta t)}$ conditional on the present state $x^{(t)}$. ITO (Schreiner et al., 2023) modelled this distribution using diffusion with a $SE(3)$-equivariant PaiNN architecture. Equijump (dos Santos Costa et al., 2024) extended this idea with a protein-specific message-passing neural network with reweighting to sample rarer conformations of fast-folding proteins. Importantly, Timewarp was the first truly *transferable* Boltzmann generator, but was still too slow relative to molecular dynamics on unseen systems. Later, Transferable Boltzmann Generators (TBG) (Klein & Noé, 2024) built upon Timewarp by using flow-matching instead of maximum likelihood estimation and a more efficient continuous normalizing flow architecture. 'Two for One' (Arts et al., 2023) showed that the score learned by diffusion models can be used for running molecular dynamics simulations. However, as they choose the noise level for the score function close to 0, the molecular dynamics is effectively run in the original space X, not in the latent space Y as JAMUN does, which again limits the effective timestep of simulation.

MDGen (Jing et al., 2024) creates a $SE(3)$-invariant tokenization of the backbone and sidechain torsion angles, relative to a known initial conformation (here, $x^{(0)}$). Then, they learn a stochastic interpolant (Albergo et al., 2023) (a generalization of diffusion and flow-matching) over the trajectories of these tokens.

BioEmu (Lewis et al., 2024) is a diffusion model for backbone conformations of large proteins built using the EvoFormer stack from AlphaFold2 (Jumper et al., 2021). BioEmu is pretrained on 200 million protein structures from the AlphaFold Protein Structure Database (Varadi et al., 2023) and finetuned on over 200ms of MD data, which are orders of magnitude larger than the datasets we benchmark here. BioEmu is technically only a backbone-only model, but their repository provides an additional side-chain reconstruction step using H-Packer (Visani et al., 2024), allowing comparison to the all-atom models. As seen in Table 2, the side-chain reconstruction can be quite expensive, because of the lack of support for batched inference with H-Packer.

Finally, we also perform some comparisons to Boltz-1 (Wohlwend et al., 2024), an open-source reproduction of AlphaFold3 (Abramson et al., 2024). Boltz-1 was trained exclusively on static crystalline structures of folded states, without any dynamics or conformational information, allowing us to evaluate how effectively the conformational landscape can be inferred from structural data alone.

Concretely, we compare to several of these state-of-the art methods on our benchmark datasets:

- TBG (Klein & Noé, 2024) on TIMEWARP 2AA-LARGE.
- MDGen (Jing et al., 2024) on MDGEN 4AA-EXPLICIT.

Model	Time per Sample	Number of Samples	Total Time
TBG	720 ms	5,000	60 min
MDGen	6 ms	10,000	1 min
Boltz-1	360 ms	10,000	60 min
BioEmu	15 ms	10,000	2.5 min
BioEmu + H-Packer	4320 ms	10,000	720 min
JAMUN (2AA)	2 ms	100,000	3 min
JAMUN (4AA)	3 ms	100,000	5 min
JAMUN (5AA)	8 ms	100,000	12.5 min
CAPPED 2AA	40 ms	60,000	40 min
MDGEN 4AA-EXPLICIT	11 ms	1,000,000	180 min
UNCAPPED 5AA	108 ms	100,000	180 min

Table 2. Comparison of (approximate and batched) sampling times per test peptide for baseline models (top), baseline MD simulations (middle) and JAMUN. All models and baselines were run on a single NVIDIA RTX A100 GPU, except for MDGEN 4AA-EXPLICIT which was simulated on a single NVIDIA T4 GPU, as mentioned in Jing et al. (2024).

- Boltz-1 (Wohlwend et al., 2024) and BioEmu (Lewis et al., 2024) on UNCAPPED 5AA.

In Appendix I, we discuss some preliminary results with JAMUN on cyclic peptides (macrocyles), which are becoming popular as a new drug modality (Garcia Jimenez et al., 2023), but are not modelled well by existing models due to lack of support for non-canonical and methylated residues in their parametrizations.

Table 2 contains a summary of the sampling efficiencies for different models, averaged over the corresponding test sets peptides. While the TBG model can technically produce reweighted samples, we run it in the un-reweighted mode, making it a Boltzmann emulator which is almost $10\times$ faster in practice. This allows for a fair comparison to JAMUN.

For a fair comparison, TBG and MDGen were compared on their representative benchmark datasets. Unfortunately, these models do not transfer to UNCAPPED 5AA due to fixed-length positional embeddings, despite our best efforts. In particular, we initialized the positional embeddings to support longer peptides, but this resulted in broken topologies in the resulting samples. Instead, we choose Boltz-1 and BioEmu which support sampling on UNCAPPED 5AA to compare against JAMUN trained on TIMEWARP 4AA-LARGE.

Due to the different simulation conditions across the datasets shown in Table 1, we train a different JAMUN model for each dataset. However, the same noise level of $\sigma = 0.4$Å is applied for training and sampling on all datasets. In fact, all training hyperparameters are kept identical across datasets. Appendix A contains a discussion of how this noise level was chosen. Our denoiser model is built with the `e3nn` library (Geiger & Smidt, 2022), and contains approximately 10.5M parameters.

5. Metrics

We adopt the analysis methods of MDGen (Jing et al., 2024). In particular, we compare models by projecting their sampled distributions of all-atom positions onto a variety of variables: pairwise distances, dihedral angles of backbone (known as Ramachandran plots) and sidechain torsion angles, TICA (time-lagged independent coordinate analysis) projections, and metastable state probabilities as computed by Markov State Models (MSMs) fit with PyEMMA (Scherer et al., 2015). TICA (Molgedey & Schuster, 1994) is a popular dimensionality reduction method for larger molecules which aims to extract slow collective degrees of freedom from a trajectory (Pérez-Hernández et al., 2013; Schwantes & Pande, 2013). As is standard practice, all TICA projections and MSMs are estimated using the reference MD data.

6. Results

6.1. Results on TIMEWARP 2AA-LARGE and TIMEWARP 4AA-LARGE

First, we show that JAMUN samples similar states as the reference MD data. Indeed, Figure 4 shows that the metastable state probabilities over JAMUN sampled trajectories match very well with those over the reference MD data on the TIMEWARP 2AA-LARGE and TIMEWARP 4AA-LARGE datasets.

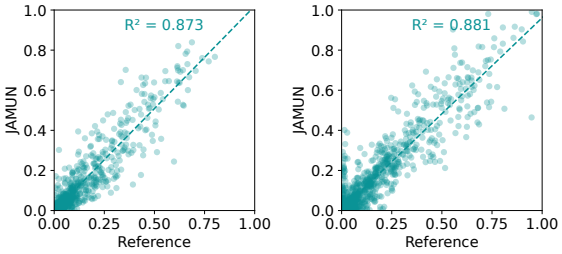


Figure 4. Across both TIMEWARP 2AA-LARGE (left) and TIMEWARP 4AA-LARGE (right), MSM state probabilities for JAMUN samples (on the y-axis) and those for the reference MD trajectories (on the x-axis) across all test peptides are strongly correlated. The perfect sampler will obtain an R^2 of 1.

Table 3 shows that on TIMEWARP 2AA-LARGE, JAMUN outperforms TBG when run for equal amounts of time (based on Table 2), and is only slightly worse when TBG is run for $20\times$ longer.

In Figure 5 and Figure 6 , we visualize the TICA-0,1 projections and Ramachandran plots for randomly chosen test peptides from TIMEWARP 2AA-LARGE, highlighting that TBG misses certain basins that JAMUN is able to sample.

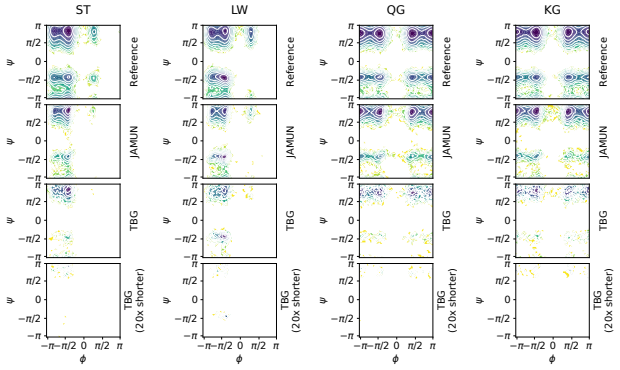


Figure 5. Ramachandran plots for 4 randomly chosen test peptides on TIMEWARP 2AA-LARGE.

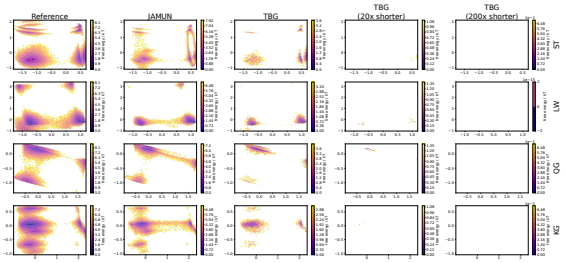


Figure 6. TICA-0,1 projections for 4 randomly chosen test peptides on TIMEWARP 2AA-LARGE.

6.2. Results on CAPPED 2AA

Here, we compute the ratio of the decorrelation times for the backbone and sidechain torsions in JAMUN and the reference MD data. Figure 7 highlights how sampling in the smoothed space Y compared to the original space X enables much faster decorrelation.

In Table 3, we compare JAMUN with the reference MD shortened by a factor of 10. JAMUN outperforms this shortened MD trajectory across all metrics, even though it takes approximately $2\times$ longer to sample than JAMUN, according to Table 2.

6.3. Results on MDGEN 4AA-EXPLICIT

Table 3 shows that JAMUN is very competitive with MDGen on the JSD metrics. In fact, Figure 9 shows that MDGen is missing some basins that JAMUN is able to sample. On the other hand, Figure 10 show an example where JAMUN hallucinates a basin.

Figure 8 shows the significant speedups in backbone and sidechain torsion decorrelation times for JAMUN compared to the reference MD data. This matches our intuition about taking 'larger' integrator steps in the smoothed manifold of the noisy latent space.

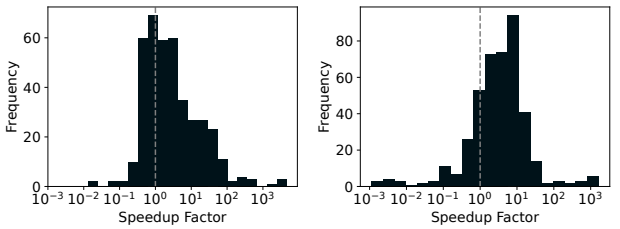


Figure 7. Speedups defined as the ratio between decorrelation times between the reference MD and JAMUN for backbone (left) and sidechain (right) torsions for all test peptides in CAPPED 2AA.

Trajectory	Backbone Torsions	Sidechain Torsions	All Torsions	TICA-0	TICA-0,1	Metastable Probs
TIMEWARP 2AA-LARGE						
JAMUN	0.130 ± 0.020	0.185 ± 0.044	0.165 ± 0.030	0.177 ± 0.053	0.260 ± 0.052	0.155 ± 0.063
TBG	0.083 ± 0.028	0.115 ± 0.045	0.105 ± 0.038	0.122 ± 0.051	0.225 ± 0.070	0.101 ± 0.046
TBG (20× shorter)	0.203 ± 0.070	0.235 ± 0.073	0.225 ± 0.071	0.240 ± 0.072	0.484 ± 0.073	0.124 ± 0.054
CAPPED 2AA						
JAMUN	0.291 ± 0.119	0.320 ± 0.108	0.304 ± 0.112	0.351 ± 0.130	0.438 ± 0.117	0.264 ± 0.108
Reference (10× shorter)	0.447 ± 0.057	0.406 ± 0.071	0.424 ± 0.056	0.557 ± 0.043	0.564 ± 0.041	0.543 ± 0.073
MDGEN 4AA-EXPLICIT						
JAMUN	0.159 ± 0.060	0.210 ± 0.057	0.187 ± 0.054	0.257 ± 0.111	0.353 ± 0.120	0.262 ± 0.118
Reference (10× shorter)	0.100 ± 0.035	0.092 ± 0.027	0.095 ± 0.025	0.234 ± 0.068	0.332 ± 0.067	0.286 ± 0.066
Reference (100× shorter)	0.227 ± 0.062	0.254 ± 0.060	0.240 ± 0.051	0.444 ± 0.131	0.569 ± 0.108	0.482 ± 0.138
MDGen	0.129 ± 0.039	0.089 ± 0.032	0.107 ± 0.028	0.228 ± 0.092	0.320 ± 0.087	0.233 ± 0.093
UNCAPPED 5AA						
JAMUN	0.196 ± 0.027	0.196 ± 0.013	0.197 ± 0.010	0.336 ± 0.049	0.440 ± 0.048	0.250 ± 0.075
Reference (10× shorter)	0.118 ± 0.013	0.150 ± 0.032	0.135 ± 0.015	0.430 ± 0.077	0.504 ± 0.079	0.460 ± 0.051
Reference (100× shorter)	0.272 ± 0.062	0.307 ± 0.023	0.290 ± 0.039	0.555 ± 0.070	0.678 ± 0.034	0.601 ± 0.112
Boltz-1	0.425 ± 0.033	0.402 ± 0.036	0.411 ± 0.029	0.457 ± 0.050	0.584 ± 0.026	0.483 ± 0.047
BioEmu	0.329 ± 0.013	0.489 ± 0.024	0.420 ± 0.018	0.415 ± 0.092	0.597 ± 0.026	0.321 ± 0.018

Table 3. Comparison of Jensen-Shannon distances for different methods on corresponding benchmark datasets.

6.4. Assessing Generalization over Peptide Lengths with UNCAPPED 5AA

JAMUN's graph neural network architecture enables it to operate on molecules of larger sizes than it was originally trained on. Thus, we test whether JAMUN can generalize to peptides of lengths beyond its training set. This is a challenging task, and one that we believe conformational generation models have not been adequately benchmarked on. As we mentioned before in Section 4, the TBG and MDGen models are not compatible with this experiment.

Surprisingly, we find that the JAMUN model *trained only on 4AA peptides can accurately predict ensembles for 5AA peptides*. Figure 11 and Figure 12 show that JAMUN is able to recover most states and even reproduce relative probabilities. Interestingly, the same experiment does not work if we train on TIMEWARP 2AA-LARGE instead, suggesting that the 2AA reference MD data may not be informative enough to generalize from.

On the other hand, we find that Boltz-1 is unable to sample the diversity of peptide conformations. This is not entirely surprising as Boltz-1 was not trained on any MD data, as we noted before. Further, Boltz-1 also utilizes a common pair representation, as computed by its Pairformer stack, across all diffusion samples. The pair representation intuitively represents the residue-wise distance matrix, and thus encodes a significant portion of the geometry. Keeping this

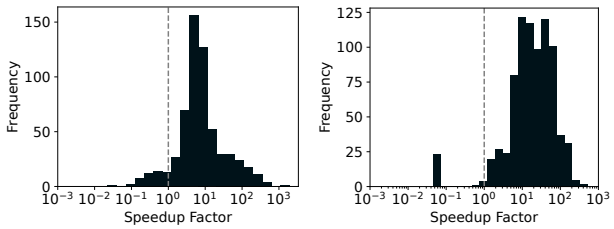


Figure 8. Speedups defined as the ratio between decorrelation times between the reference MD and JAMUN for backbone and sidechain torsions for all test peptides in MDGEN 4AA-EXPLICIT.

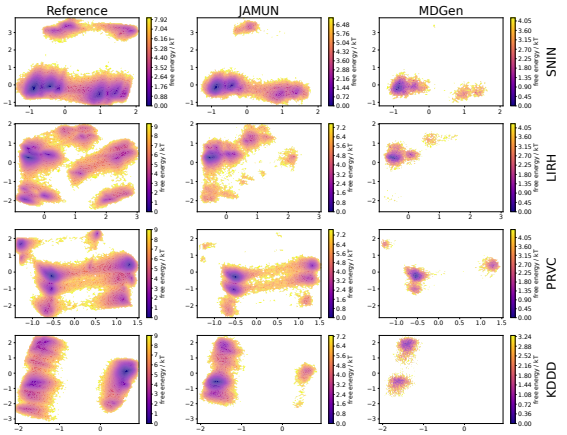


Figure 9. TICA-0,1 projections for 4 randomly chosen test peptides on MDGEN 4AA-EXPLICIT.

representation fixed possibly prevents the sampling of large conformational changes.

Surprisingly, BioEmu also seems to struggle in this setting, even when considering distributions of backbone torsion angles only. This suggests that BioEmu cannot capture the relative flexibility of smaller peptides.

Quantitatively, Table 3 shows that JAMUN significantly outperforms Boltz-1 and BioEmu on the metrics from Section 5. As seen in Table 2, JAMUN is roughly $5\times$ faster than Boltz-1, and is roughly $60\times$ faster than BioEmu when we perform side-chain reconstruction with H-Packer.

7. Conclusion

We present JAMUN, a walk-jump sampling model for generating ensembles of molecular conformations, outperforming the state-of-the-art TBG model, and competitive with the performance of MDGen with no protein-specific parametrization. This represents an important step toward the ultimate goal of a transferable generative model for protein conformational ensembles. Performing MD in the noised space gives the model a clear physics interpretation, and allows faster decorrelation (and hence, sampling) than

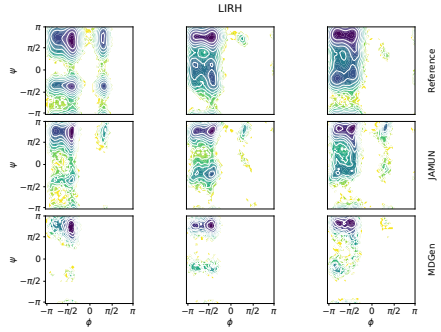


Figure 10. Ramachandran plots for JAMUN and MDGen on a randomly chosen test peptide LIRH in MDGEN 4AA-EXPLICIT.

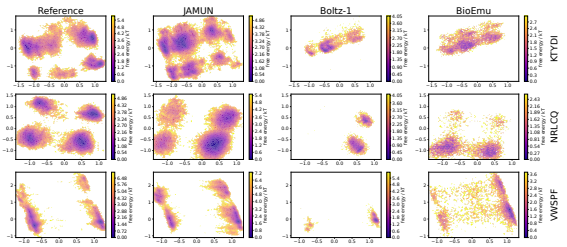


Figure 11. TICA-0,1 projections for the three test peptides NRLCQ, VWSPF and KTYDI in UNCAPPED 5AA.

classical MD.

The model has some limitations that motivate future work. While it is highly transferable in the space of two to five amino acid peptides, we have not tested the model's ability to transfer to larger proteins. We leave exploring this important direction to future work. Additionally, while the current $SE(3)$-equivariant denoiser architecture works well, further development of the denoising network could speed up sampling. Alternative jump methods, such as multiple denoising steps (à la diffusion), could also serve to sharpen generation. Lastly, a promising direction that has not yet been explored is the application of classical enhanced sampling methods, such as metadynamics, for traversing the noisy space.

8. Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., Bodenstein, S. W., Evans, D. A., Hung, C.-

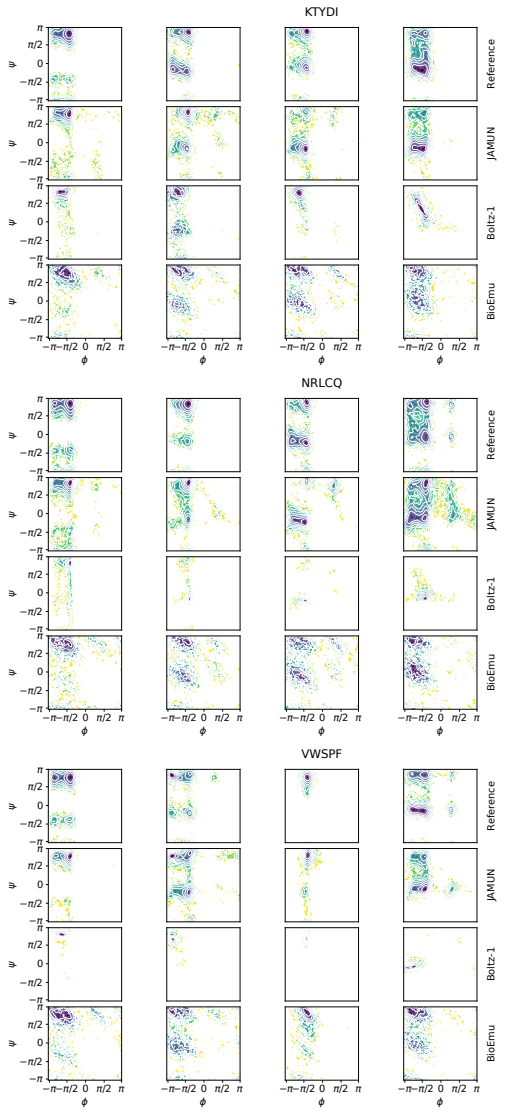


Figure 12. Ramachandran plots for the three test peptides NRLCQ, VWSPF and KTYDI in UNCAPPED 5AA.

C., O'Neill, M., Reiman, D., Tunyasuvunakool, K., Wu, Z., Žemgulytė, A., Arvaniti, E., Beattie, C., Bertolli, O., Bridgland, A., Cherepanov, A., Congreve, M., Cowen-Rivers, A. I., Cowie, A., Figurnov, M., Fuchs, F. B., Gladman, H., Jain, R., Khan, Y. A., Low, C. M. R., Perlin, K., Potapenko, A., Savy, P., Singh, S., Stecula, A., Thillaisundaram, A., Tong, C., Yakneen, S., Zhong, E. D., Zielinski, M., Žídek, A., Bapst, V., Kohli, P., Jaderberg, M., Hassabis, D., and Jumper, J. M. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, Jun 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07487-w. URL https://doi.org/10.1038/s41586-024-07487-w.

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic Interpolants: A Unifying Framework for Flows and Diffusions, 2023. URL https://arxiv.org/abs/2303.08797.

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Suo, M., Tillet, P., Wang, E., Wang, X., Wen, W., Zhang, S., Zhao, X., Zhou, K., Zou, R., Mathews, A., Chanan, G., Wu, P., and Chintala, S. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL https://pytorch.org/assets/pytorch2-2.pdf.

Antoszewski, A., Feng, C.-J., Vani, B. P., Thiede, E. H., Hong, L., Weare, J., Tokmakoff, A., and Dinner, A. R. Insulin Dissociates by Diverse Mechanisms of Coupled Unfolding and Unbinding. *The Journal of Physical Chemistry B*, 124(27):5571–5587, 2020. doi: 10.1021/acs.jpcb.0c03521. URL https://doi.org/10.1021/acs.jpcb.0c03521. PMID: 32515958.

Aranganathan, A., Gu, X., Wang, D., Vani, B., and Tiwary, P. Modeling Boltzmann weighted structural ensembles of proteins using AI based methods. 2024.

Arts, M., Garcia Satorras, V., Huang, C.-W., Zugner, D., Federici, M., Clementi, C., Noé, F., Pinsler, R., and van den Berg, R. Two for one: Diffusion models and force fields for coarse-grained molecular dynamics. *Journal of Chemical Theory and Computation*, 19(18):6151–6159, 2023.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453, May 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-29939-5. URL https://doi.org/10.1038/s41467-022-29939-5.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. The Protein Data Bank. *Nucleic Acids Research*, 28(1): 235–242, 01 2000. ISSN 0305-1048. doi: 10.1093/nar/28.1.235. URL https://doi.org/10.1093/nar/28.1.235.

Borhani, D. W. and Shaw, D. E. The future of molecular dynamics simulations in drug discovery. *Journal of computer-aided molecular design*, 26:15–26, 2012.

Bowman, G. R. AlphaFold and Protein Folding: Not Dead Yet! The Frontier Is Conformational Ensembles. *Annual Review of Biomedical Data Science*, 7, 2024.

Chmiela, S., Tkatchenko, A., Sauceda, H. E., Poltavsky, I., Schütt, K. T., and Müller, K.-R. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017. doi: 10.1126/sciadv.1603015. URL https://www.science.org/doi/abs/10.1126/sciadv.1603015.

Colombo, G. Computing allostery: from the understanding of biomolecular regulation and the discovery of cryptic sites to molecular design. *Current Opinion in Structural Biology*, 83:102702, 2023.

Darden, T., York, D., and Pedersen, L. Particle Mesh Ewald: An N log(N) method for Ewald sums in large systems. *The Journal of Chemical Physics*, 98(12):10089–10092, June 1993. doi: 10.1063/1.464397. URL https://doi.org/10.1063/1.464397.

Dinner, A. R., Mattingly, J. C., Tempkin, J. O. B., Van Koten, B., and Weare, J. Trajectory stratification of stochastic dynamics. *SIAM Rev Soc Ind Appl Math*, 60(4):909–938, November 2018.

dos Santos Costa, A., Mitnikov, I., Pellegrini, F., Daigavane, A., Geiger, M., Cao, Z., Kreis, K., Smidt, T., Kucukbenli, E., and Jacobson, J. EquiJump: Protein Dynamics Simulation via SO(3)-Equivariant Stochastic Interpolants, 2024. URL https://arxiv.org/abs/2410.09667.

Dumitrescu, A., Korpela, D., Heinonen, M., Verma, Y., Iakovlev, V., Garg, V., and Lähdesmäki, H. Field-based Molecule Generation, 2024. URL https://arxiv.org/abs/2402.15864.

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., Wiewiora, R. P., Brooks, B. R., and Pande, V. S. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology*, 13(7):e1005659, July 2017. doi: 10.1371/journal.pcbi.1005659. URL https://doi.org/10.1371/journal.pcbi.1005659.

Falcon, W. and The PyTorch Lightning team. PyTorch Lightning, March 2019. URL https://github.com/Lightning-AI/lightning.

Fernández-Quintero, M. L., Pomarici, N. D., Fischer, A.-L. M., Hoerschinger, V. J., Kroell, K. B., Riccabona, J. R., Kamenik, A. S., Loeffler, J. R., Ferguson, J. A., Perrett, H. R., et al. Structure and Dynamics Guiding Design of Antibody Therapeutics and Vaccines. *Antibodies*, 12(4): 67, 2023.

Frey, N. C., Berenberg, D., Kleinhenz, J., Hotzel, I., Lafrance-Vanasse, J., Kelly, R. L., Wu, Y., Rajpal, A., Ra, S., Bonneau, R., Cho, K., Loukas, A., Gligorijevic, V., and Saremi, S. Protein Discovery with Discrete Walk-Jump Sampling. In *International Conference on Learning Representations*, 2024.

Garcia Jimenez, D., Poongavanam, V., and Kihlberg, J. Macrocycles in Drug Discovery-Learning from the Past for the Future. *Journal of Medicinal Chemistry*, 66 (8):5377–5396, 2023. doi: 10.1021/acs.jmedchem.3c00134. URL https://doi.org/10.1021/acs.jmedchem.3c00134. PMID: 37017513.

Geiger, M. and Smidt, T. e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*, 2022.

Gough, N. R. and Kalodimos, C. G. Exploring the conformational landscape of protein kinases. *Current Opinion in Structural Biology*, 88:102890, 2024.

Grambow, C. A., Weir, H., Cunningham, C. N., Biancalani, T., and Chuang, K. V. RINGER: Rapid inference of macrocyclic peptide conformational ensembles. *arXiv preprint*, arXiv:2310.01234, 2023. URL https://arxiv.org/abs/2310.01234.

Grambow, C. A., Weir, H., Cunningham, C. N., Biancalani, T., and Chuang, K. V. CREMP: Conformer–rotamer ensembles of macrocyclic peptides for machine learning. *Scientific Data*, 11, 2024. doi: 10.1038/s41597-024-03698-y. URL https://doi.org/10.1038/s41597-024-03698-y.

Henzler-Wildman, K. and Kern, D. Dynamic personalities of proteins. *Nature*, 450(7172):964–972, December 2007.

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

Ho, J., Jain, A., and Abbeel, P. Denoising Diffusion Probabilistic Models, 2020. URL https://arxiv.org/abs/2006.11239.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant Diffusion for Molecule Generation in 3D, 2022. URL https://arxiv.org/abs/2203.17003.

Jing, B., Stärk, H., Jaakkola, T., and Berger, B. Generative Modeling of Molecular Dynamics Trajectories. *arXiv preprint arXiv:2409.17808*, 2024.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596 (7873):583–589, Aug 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL https://doi.org/10.1038/s41586-021-03819-2.

Kabsch, W. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, Sep 1976. doi: 10.1107/S0567739476001873. URL https://doi.org/10.1107/S0567739476001873.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the Design Space of Diffusion-Based Generative Models. In *Proc. NeurIPS*, 2022.

Karras, T., Aittala, M., Lehtinen, J., Hellsten, J., Aila, T., and Laine, S. Analyzing and Improving the Training Dynamics of Diffusion Models. In *Proc. CVPR*, 2024.

Kieninger, S. and Keller, B. G. GROMACS Stochastic Dynamics and BAOAB Are Equivalent Configurational Sampling Algorithms. *Journal of Chemical Theory and Computation*, 18(10):5792–5798, 2022.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Klein, L. and Noé, F. Transferable Boltzmann Generators. *arXiv preprint arXiv:2406.14426*, 2024.

Klein, L., Foong, A., Fjelde, T., Mlodozeniec, B., Brockschmidt, M., Nowozin, S., Noé, F., and Tomioka, R. Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. *Advances in Neural Information Processing Systems*, 36, 2024a.

Klein, L., Krämer, A., and Noé, F. Equivariant flow matching. *Advances in Neural Information Processing Systems*, 36, 2024b.

Leimkuhler, B. and Matthews, C. Rational Construction of Stochastic Numerical Methods for Molecular Sampling. *Applied Mathematics Research eXpress*, 2013(1):34–56, June 2012. ISSN 1687-1200. doi: 10.1093/amrx/abs010. URL https://doi.org/10.1093/amrx/abs010. _eprint: https://academic.oup.com/amrx/article-pdf/2013/1/34/397230/abs010.pdf.

Leimkuhler, B. and Matthews, C. Robust and efficient configurational molecular sampling via Langevin dynamics. *The Journal of Chemical Physics*, 138(17): 174102, May 2013. doi: 10.1063/1.4802990. URL https://doi.org/10.1063/1.4802990.

Leimkuhler, B. and Matthews, C. *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Number 39 in Interdisciplinary Applied Mathematics. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2015 edition, 2015. ISBN 978-3-319-16375-8.

Lewis, S., Hempel, T., Jiménez-Luna, J., Gastegger, M., Xie, Y., Foong, A. Y. K., Satorras, V. G., Abdin, O., Veeling, B. S., Zaporozhets, I., Chen, Y., Yang, S., Schneuing, A., Nigam, J., Barbero, F., Stimper, V., Campbell, A., Yim, J., Lienen, M., Shi, Y., Zheng, S., Schulz, H., Munir, U., Clementi, C., and Noé, F. Scalable emulation of protein equilibrium ensembles with generative deep learning. *bioRxiv*, 2024. doi: 10.1101/2024.12.05.626885.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Miller, M. D. and Phillips, G. N. Moving beyond static snapshots: Protein dynamics and the Protein Data Bank. *Journal of Biological Chemistry*, 296, 2021.

Miyasawa, K. An Empirial Bayes Estimator of the Mean of a Normal Population. *Bulletin de l'Institut international de statistique.*, 38(4):181–188, 1960.

Molgedey, L. and Schuster, H. G. Separation of a mixture of independent signals using time delayed correlations.

Phys. Rev. Lett., 72:3634–3637, Jun 1994. doi: 10.1103/PhysRevLett.72.3634. URL https://link.aps.org/doi/10.1103/PhysRevLett.72.3634.

Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

Pérez-Hernández, G., Paul, F., Giorgino, T., De Fabritiis, G., and Noé, F. Identification of slow molecular order parameters for Markov model construction. *J Chem Phys*, 139(1):015102, July 2013.

Pinheiro, P. O., Jamasb, A., Mahmood, O., Sresht, V., and Saremi, S. Structure-based Drug Design by Denoising Voxel Grids. *arXiv preprint arXiv:2405.03961*, 2024a.

Pinheiro, P. O., Rackers, J., Kleinhenz, J., Maser, M., Mahmood, O., Watkins, A., Ra, S., Sresht, V., and Saremi, S. 3D molecule generation by denoising voxel grids. *Advances in Neural Information Processing Systems*, 36, 2024b.

Ponder, J. W. and Case, D. A. Force Fields for Protein Simulations. 66:27–85, 2003. ISSN 0065-3233. doi: https://doi.org/10.1016/S0065-3233(03)66002-X. URL https://www.sciencedirect.com/science/article/pii/S006532330366002X.

Pracht, P. and Grimme, S. Automated exploration of the low-energy chemical space with fast quantum chemical methods. *Physical Chemistry Chemical Physics*, 22 (14):7169–7192, 2020. doi: 10.1039/C9CP06869D. URL https://pubs.rsc.org/en/content/articlelanding/2020/cp/c9cp06869d.

Robbins, H. An Empirical Bayes Approach to Statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 3.1, 1956.

Sachs, M., Leimkuhler, B., and Danos, V. Langevin dynamics with variable coefficients and nonconservative forces: from stationary states to numerical methods. *Entropy*, 19 (12):647, 2017.

Saremi, S. and Hyvärinen, A. Neural Empirical Bayes. *Journal of Machine Learning Research*, 20(181):1–23, 2019.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) Equivariant Graph Neural Networks, 2022. URL https://arxiv.org/abs/2102.09844.

Scherer, M. K., Trendelkamp-Schroer, B., Paul, F., Pérez-Hernández, G., Hoffmann, M., Plattner, N., Wehmeyer, C., Prinz, J.-H., and Noé, F. PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *Journal of Chemical Theory and Computation*, 11(11):5525–5542, 2015. doi: 10.1021/acs.jctc.5b00743. URL https://doi.org/10.1021/acs.jctc.5b00743. PMID: 26574340.

Schreiner, M., Winther, O., and Olsson, S. Implicit Transfer Operator Learning: Multiple Time-Resolution Surrogates for Molecular Dynamics, 2023. URL https://arxiv.org/abs/2305.18046.

Schwantes, C. R. and Pande, V. S. Improvements in Markov State Model Construction Reveal Many Non-Native Interactions in the Folding of NTL9. *J Chem Theory Comput*, 9(4):2000–2009, April 2013.

Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., and Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 8(1), January 2017. ISSN 2041-1723. doi: 10.1038/ncomms13890. URL http://dx.doi.org/10.1038/ncomms13890.

Shaw, D. E., Maragakis, P., Lindorff-Larsen, K., Piana, S., Dror, R. O., Eastwood, M. P., Bank, J. A., Jumper, J. M., Salmon, J. K., Shan, Y., and Wriggers, W. Atomic-Level Characterization of the Structural Dynamics of Proteins. *Science*, 330(6002):341–346, 2010. doi: 10.1126/science.1187409. URL https://www.science.org/doi/abs/10.1126/science.1187409.

Song, J., Meng, C., and Ermon, S. Denoising Diffusion Implicit Models, 2022. URL https://arxiv.org/abs/2010.02502.

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

Umeyama, S. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13 (4):376–380, 1991. doi: 10.1109/34.88573.

Vani, B. P., Weare, J., and Dinner, A. R. Computing transition path theory quantities with trajectory stratification. *J Chem Phys*, 157(3):034106, July 2022.

Vapnik, V. Principles of Risk Minimization for Learning Theory. In Moody, J., Hanson, S., and Lippmann, R. (eds.), *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991. URL https://proceedings.neurips.cc/paper_files/paper/1991/file/ff4d5fbbafdf976cfdc032e3bde78de5-Paper.pdf.

Varadi, M., Bertoni, D., Magana, P., Paramval, U., Pidruchna, I., Radhakrishnan, M., Tsenkov, M., Nair, S., Mirdita, M., Yeo, J., Kovalevskiy, O., Tunyasuvunakool, K., Laydon, A., Žídek, A., Tomlinson, H., Hariharan, D., Abrahamson, J., Green, T., Jumper, J., Birney, E., Steinegger, M., Hassabis, D., and Velankar, S. AlphaFold Protein Structure Database in 2024: providing structure coverage for over 214 million protein sequences. *Nucleic Acids Research*, 52(D1):D368–D375, 11 2023. ISSN 0305-1048. doi: 10.1093/nar/gkad1011. URL https://doi.org/10.1093/nar/gkad1011.

Visani, G. M., Galvin, W., Pun, M., and Nourmohammad, A. H-Packer: Holographic Rotationally Equivariant Convolutional Neural Network for Protein Side-Chain Packing. In Knowles, D. A. and Mostafavi, S. (eds.), *Proceedings of the 18th Machine Learning in Computational Biology meeting*, volume 240 of *Proceedings of Machine Learning Research*, pp. 230–249. PMLR, 30 Nov–01 Dec 2024. URL https://proceedings.mlr.press/v240/visani24a.html.

Vitalis, A. and Pappu, R. V. Methods for Monte Carlo simulations of biomacromolecules. *Annual reports in computational chemistry*, 5:49–76, 2009.

Wohlwend, J., Corso, G., Passaro, S., Reveiz, M., Leidal, K., Swiderski, W., Portnoi, T., Chinn, I., Silterra, J., Jaakkola, T., and Barzilay, R. Boltz-1 Democratizing Biomolecular Interaction Modeling. *bioRxiv*, 2024. doi: 10.1101/2024.11.19.624167. URL https://www.biorxiv.org/content/early/2024/11/20/2024.11.19.624167.

Wu, F. and Li, S. Z. DiffMD: A Geometric Diffusion Model for Molecular Dynamics Simulations, 2023. URL https://arxiv.org/abs/2204.08672.

Zheng, L.-E., Barethiya, S., Nordquist, E., and Chen, J. Machine learning generation of dynamic protein conformational ensembles. *Molecules*, 28(10):4047, 2023.

A. Choice of Noise Scale σ

As we discussed in Section 2.3, the scale of noise σ used to define Y is important. We found that the scale of noise chosen in the paper for all experiments ($\sigma = 0.4$Å) is large enough to result in significant disruption of structure, leading to the smoothed Gaussian convolved 'walk' manifold. However, the scale is also small enough to avoid atoms 'swapping' position, for instance, or pairs of bonded atoms ending up very far from each other with reasonable probability. Indeed, as shown in Figure 13, we find that higher noise levels (such as $\sigma = 0.8$Å) result in samples with broken topologies, while lower noise levels (such as $\sigma = 0.2$Å) require many more sampling steps to explore the entire conformational landscape.

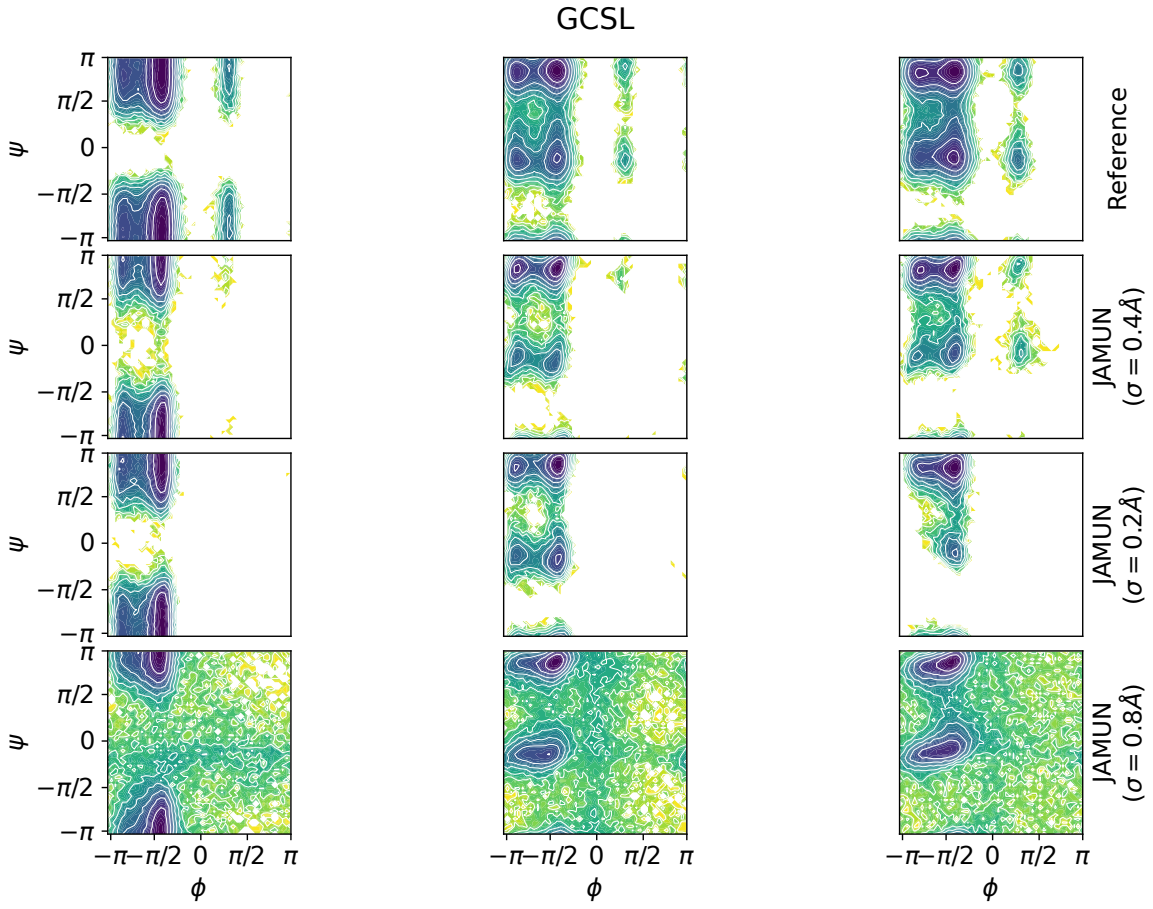


Figure 13. Ramachandran plots for an example test peptide GCSL from TIMEWARP 4AA-LARGE for JAMUN models trained at different noise levels σ and sampled for the same number of steps, showing the tradeoff between smaller noise levels (slower mode mixing at $\sigma = 0.2$Å) and larger noise levels (broken topologies at $\sigma = 0.8$Å).

B. Simulation Details for CAPPED 2AA

We ensure that our unbiased molecular dynamics runs are converged or representative by comparing against biased molecular dynamics runs using Non-Equilibrium Umbrella Sampling (NEUS) (Dinner et al., 2018; Vani et al., 2022), a trajectory stratification based enhanced sampling algorithm. The protein is represented by the AMBER03 force field (Ponder & Case, 2003). The simulations are performed at 300 K with the BAOAB integrator (Leimkuhler & Matthews, 2013) in OpenMM (Eastman et al., 2017); LINCS is used to constrain the lengths of bonds to hydrogen atoms (Hess et al., 1997); Particle Mesh Ewald is used to calculate electrostatics (Darden et al., 1993); the step size was 2 fs. The systems are solvated with TIP3P water models and equilibrated under NVT and NPT ensembles for 100ps each.

C. Overview of Denoiser

The denoiser is a $SE(3)$-equivariant graph neural network. The graph is defined by a radial cutoff of 10Å in y. The overall computation performed by the denoiser is shown in Figure 14, with the initial embedding, message-passing and output head blocks shown in Figure 15, Figure 16 and Figure 17 respectively.

For all datasets, we train and sample with $\sigma = 0.4$ Å. For the Langevin dynamics (Equation 68), we set $M = 1$, friction of $\gamma = 1.0$ and a step size of $\Delta t = \sigma$.

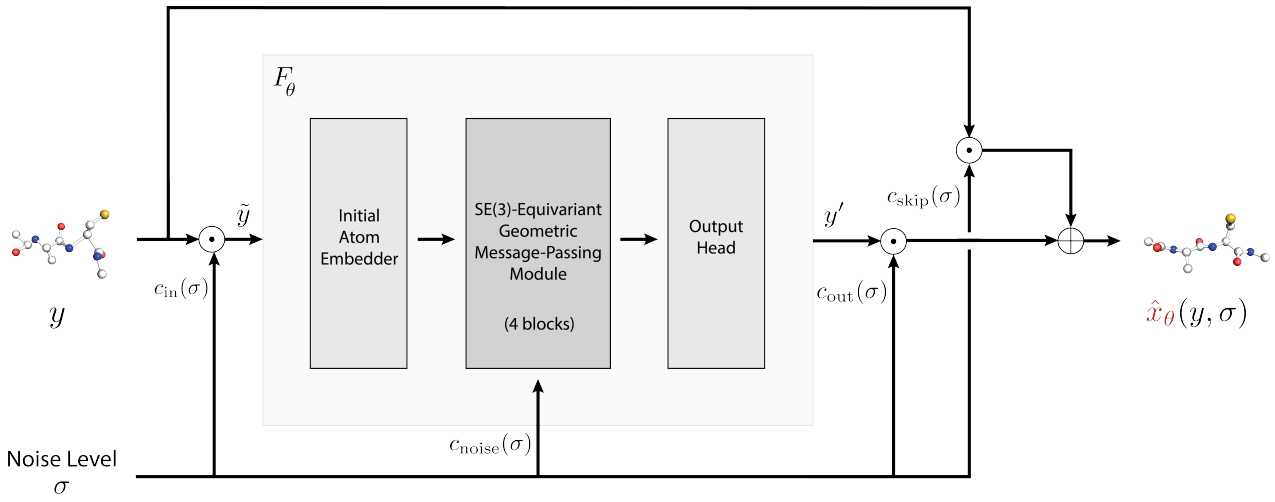


Figure 14. Overview of the denoiser network $\hat{x}_\theta$. The submodule F_θ sees input atom coordinates $\tilde{y} = c_{\text{in}}(\sigma)y$ and outputs predicted atom coordinates y', which gets scaled and added to a noise-conditional skip connection to finally obtain $\hat{x}_\theta(y)$.

The hidden features $h^{(n)}$ for $n = 0, \ldots, 4$ contain 120 scalar and 32 vector features per atom. We use spherical harmonics up to $l = 1$ for the tensor product.

We use the Adam optimizer with learning rate .002. Models are trained with a batch size of 32 over 2 NVIDIA RTX A100 GPUs.

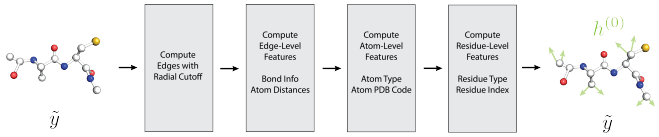


Figure 15. Overview of the initial embedder in the denoiser network, creating initial features $h^{(0)}$ at each atom and edge.

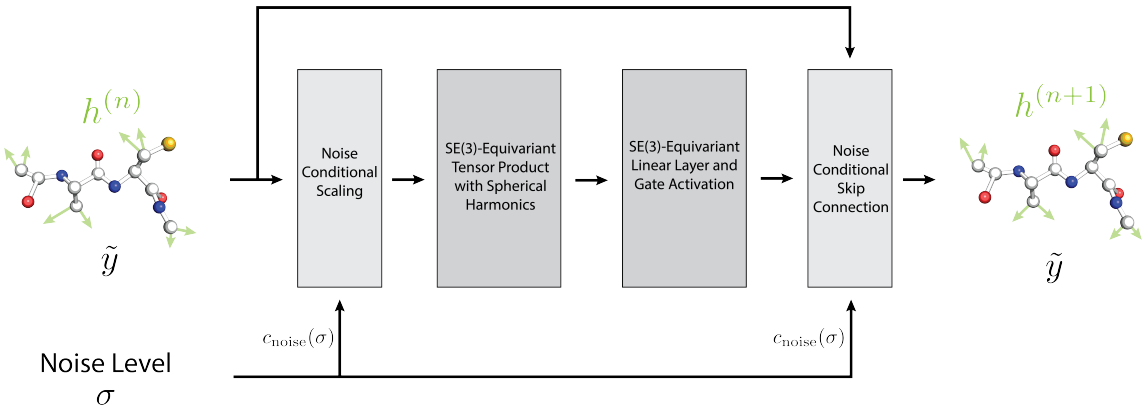


Figure 16. Overview of a single $SE(3)$-equivariant message-passing block (indexed by n) in the denoiser network. There are four such blocks iteratively updating the atom features from $h^{(0)}$ to $h^{(4)}$. The atom coordinates denoted by $\tilde{y} = c_{\text{in}}(\sigma)y$ (and hence, the edge features) are unchanged throughout these blocks.

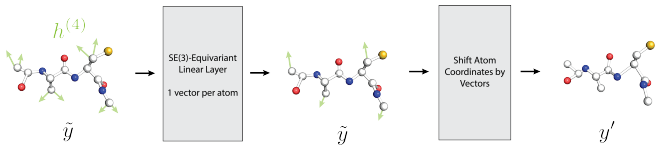


Figure 17. Overview of the output head, which predicts the coordinates $y' = F_\theta(c_{\text{in}}(\sigma)y, c_{\text{noise}}(\sigma))$.

D. Normalization

As the noise level σ is increased, $y = x + \sigma\varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \mathbb{I}_{N\times 3})$ expands in space. Let $\tilde{y}$ represent the 'normalized' input y, as seen by the network F_θ:

$$\tilde{y} = c_{\text{in}}(\sigma) y \tag{10}$$

To control the expansion of y, $c_{\text{in}}(\sigma)$ is chosen such that the following property holds:

$$\mathbb{E}_{\substack{(i,j)\sim\text{Uniform}(E)\\ \varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})}}[\|\tilde{y}_i - \tilde{y}_j\|^2] = 1 \text{ at all noise levels } \sigma. \tag{11}$$

Note that this is distinct from the normalization chosen by (Karras et al., 2022; 2024), which normalizes $\|y\|$ directly. The intuition behind this normalization is that the GNN model F_θ does not operate on atom positions y directly, but instead uses the relative vectors $y_i - y_j$ to account for translation invariance, and controlling this object directly ensures that the topology of the graph does not change with varying noise level σ.

To achieve this, we compute:

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{C + 6\sigma^2}} \tag{12}$$

where $C = \mathbb{E}_{(i,j)\sim\text{Uniform}(E)} \|x_i - x_j\|^2$ can be easily estimated from the true data distribution. The full derivation can be found in Section D.1.

As the input is now appropriately normalized, the target output of the network F_θ should also be appropriately normalized. A full derivation, found in Section D.2, leads to:

$$c_{\text{skip}}(\sigma) = \frac{C}{C + 6\sigma^2} \tag{13}$$

$$c_{\text{out}}(\sigma) = \sqrt{\frac{C \cdot 6\sigma^2}{C + 6\sigma^2}} \tag{14}$$

$$c_{\text{noise}}(\sigma) = \log_{10} \sigma \tag{15}$$

The noise normalization is a scaled version of the recommendation of $\frac{1}{4}\ln\sigma$ for images in Karras et al. (2022; 2024).

D.1. Input Normalization

Fix an $(i, j) \in E$ from Equation 11. As $\varepsilon_i, \varepsilon_j \overset{\text{iid}}{\sim} \mathcal{N}(0, \mathbb{I}_3)$, we have $\varepsilon_i - \varepsilon_j \sim \mathcal{N}(0, 2\mathbb{I}_3)$ from the closure of the multivariate Gaussian under linear combinations. Thus, for each component $d = 1, 2\text{and}3$, we have: $(\varepsilon_i - \varepsilon_j)_{(d)} \sim \mathcal{N}(0, 2)$ and hence:

$$\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})}[(x_i - x_j)^T(\varepsilon_i - \varepsilon_j)] = \sum_{d=1}^{3}(x_i - x_j)_{(d)}\mathbb{E}[(\varepsilon_i - \varepsilon_j)_{(d)}] = 0 \tag{16}$$

$$\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})}[\|\varepsilon_i - \varepsilon_j\|^2] = \sum_{d=1}^{3}\mathbb{E}[(\varepsilon_i - \varepsilon_j)_{(d)}^2] = 6 \tag{17}$$

We can now compute:

$$\begin{aligned}
&\mathbb{E}_z[\|\tilde{y}_i - \tilde{y}_j\|^2] \\
&= c_{\text{in}}(\sigma)^2\mathbb{E}_\varepsilon[\|y_i - y_j\|^2] \\
&= c_{\text{in}}(\sigma)^2\mathbb{E}_\varepsilon[\|x_i - x_j + \sigma(\varepsilon_i - \varepsilon_j)\|^2] \\
&= c_{\text{in}}(\sigma)^2\left(\|x_i - x_j\|^2 + 2\sigma\mathbb{E}_\varepsilon[(x_i - x_j)^T(\varepsilon_i - \varepsilon_j)] + \sigma^2\mathbb{E}_\varepsilon[\|\varepsilon_i - \varepsilon_j\|^2]\right) \\
&= c_{\text{in}}(\sigma)^2\left(\|x_i - x_j\|^2 + \sigma^2\mathbb{E}_\varepsilon[\|\varepsilon_i - \varepsilon_j\|^2]\right) \\
&= c_{\text{in}}(\sigma)^2\left(\|x_i - x_j\|^2 + 6\sigma^2\right).
\end{aligned} \tag{18}$$

Now, taking the expectation over all $(i,j) \in E$ uniformly:

$$\begin{aligned}\mathbb{E}_{\substack{(i,j)\sim\text{Uniform}(E)\\ \varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})}}[\|\tilde{y}_i - \tilde{y}_j\|^2] &= \mathbb{E}_{(i,j)\sim\text{Uniform}(E)}[\mathbb{E}_\varepsilon[\|\tilde{y}_i - \tilde{y}_j\|^2]] \\ &= c_{\text{in}}(\sigma)^2 \left(\mathbb{E}_{(i,j)\sim\text{Uniform}(E)} \|x_i - x_j\|^2 + 6\sigma^2\right) \end{aligned} \tag{19}$$

Let $C = \mathbb{E}_{(i,j)\sim\text{Uniform}(E)} \|x_i - x_j\|^2$, which we estimate from the true data distribution. Then, from Equation 19 and our intended normalization given by Equation 11:

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{C + 6\sigma^2}} \tag{20}$$

D.2. Output Normalization

The derivation here is identical that of (Karras et al., 2022; 2024), but with our normalization. The denoising loss at a single noise level is:

$$\mathcal{L}(\hat{x}_\theta, \sigma) = \mathbb{E}_{X\sim p_X}\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})}[\|\hat{x}_\theta(X + \sigma\varepsilon, \sigma) - X\|^2] \tag{21}$$

which gets weighted across a distribution p_σ of noise levels by (unnormalized) weights $\lambda(\sigma)$:

$$\begin{aligned}\mathcal{L}(\hat{x}_\theta) &= \mathbb{E}_{\sigma\sim p_\sigma}[\lambda(\sigma)\mathcal{L}(\hat{x}_\theta, \sigma)] \\ &= \mathbb{E}_{\sigma\sim p_\sigma}\mathbb{E}_{X\sim p_X}\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})}[\lambda(\sigma)\|\hat{x}_\theta(X + \sigma\varepsilon, \sigma) - X\|^2] \\ &= \mathbb{E}_{\sigma\sim p_\sigma}\mathbb{E}_{X\sim p_X}\mathbb{E}_{Y\sim\mathcal{N}(X,\sigma^2\mathbb{I}_{N\times 3})}[\lambda(\sigma)\|\hat{x}_\theta(Y, \sigma) - X\|^2] \\ &= \mathbb{E}_{\sigma\sim p_\sigma}\mathbb{E}_{X\sim p_X}\mathbb{E}_{Y\sim\mathcal{N}(X,\sigma^2\mathbb{I}_{N\times 3})}[\lambda(\sigma)\|c_{\text{skip}}(\sigma)Y + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)Y, c_{\text{noise}}(\sigma)) - x\|^2] \\ &= \mathbb{E}_{\sigma\sim p_\sigma}\mathbb{E}_{X\sim p_X}\mathbb{E}_{Y\sim\mathcal{N}(X,\sigma^2\mathbb{I}_{N\times 3})}\left[\lambda(\sigma)c_{\text{out}}(\sigma)^2\left\|F_\theta(c_{\text{in}}(\sigma)Y, c_{\text{noise}}(\sigma)) - \frac{x - c_{\text{skip}}(\sigma)Y}{c_{\text{out}}(\sigma)}\right\|^2\right] \\ &= \mathbb{E}_{\sigma\sim p_\sigma}\mathbb{E}_{X\sim p_X}\mathbb{E}_{Y\sim\mathcal{N}(X,\sigma^2\mathbb{I}_{N\times 3})}\left[\lambda(\sigma)c_{\text{out}}(\sigma)^2\|F_\theta(c_{\text{in}}(\sigma)Y, c_{\text{noise}}(\sigma)) - F\|^2\right]\end{aligned} \tag{22}$$

where:

$$F(y,\sigma) = \frac{x - c_{\text{skip}}(\sigma)y}{c_{\text{out}}(\sigma)} \tag{23}$$

is the effective training target for the network F_θ. We want to normalize F similarly as the network input:

$$\mathbb{E}_{\substack{(i,j)\sim\text{Uniform}(E)\\ \varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})}}[\|F_i - F_j\|^2] = 1 \text{ at all noise levels } \sigma. \tag{24}$$

Again, for a fixed $(i,j) \in E$, we have:

$$\begin{aligned}\mathbb{E}_\varepsilon\|F_i - F_j\|^2 &= \frac{\mathbb{E}_\varepsilon\|(x_i - x_j) - c_{\text{skip}}(\sigma)(y_i - y_j)\|^2}{c_{\text{out}}(\sigma)^2} \\ &= \frac{\mathbb{E}_\varepsilon\|(1 - c_{\text{skip}}(\sigma))(x_i - x_j) - c_{\text{skip}}(\sigma)\sigma\cdot(\varepsilon_i - \varepsilon_j)\|^2}{c_{\text{out}}(\sigma)^2} \\ &= \frac{(1 - c_{\text{skip}}(\sigma))^2\|x_i - x_j\|^2 + c_{\text{skip}}(\sigma)^2\cdot 6\sigma^2}{c_{\text{out}}(\sigma)^2}\end{aligned} \tag{25}$$

and hence:

$$\begin{aligned}&\mathbb{E}_{\substack{(i,j)\sim\text{Uniform}(E)\\ \varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})}}[\|F_i - F_j\|^2] = 1 \\ \implies &\frac{(1 - c_{\text{skip}}(\sigma))^2\cdot C + c_{\text{skip}}(\sigma)^2\cdot 6\sigma^2}{c_{\text{out}}(\sigma)^2} = 1 \\ \implies &c_{\text{out}}(\sigma)^2 = (1 - c_{\text{skip}}(\sigma))^2\cdot C + c_{\text{skip}}(\sigma)^2\cdot 6\sigma^2\end{aligned} \tag{26}$$

where C was defined above. Now, to minimize $c_{\text{out}}(\sigma)$ to maximize reuse and avoid amplifying network errors, as recommended by Karras et al. (2022; 2024):

$$\begin{aligned} & \frac{d}{dc_{\text{skip}}(\sigma)} c_{\text{out}}(\sigma)^2 = 0 \\ \implies & -2(1 - c_{\text{skip}}(\sigma)) \cdot C + 2c_{\text{skip}}(\sigma) \cdot 6\sigma^2 = 0 \\ \implies & c_{\text{skip}}(\sigma) = \frac{C}{C + 6\sigma^2} \end{aligned} \tag{27}$$

Substituting into Equation 26, we get after some routine simplification:

$$c_{\text{out}}(\sigma) = \sqrt{\frac{C \cdot 6\sigma^2}{C + 6\sigma^2}} \tag{28}$$

The noise normalization is chosen as $c_{\text{noise}}(\sigma) = \log_{10} \sigma$, a scaled version of the recommendation of $\frac{1}{4} \ln \sigma$ for images in Karras et al. (2022; 2024).

From Equation 22, we set $\lambda(\sigma) = \frac{1}{c_{\text{out}}(\sigma)^2}$ to normalize the loss at at all noise levels, as in Karras et al. (2022; 2024).

D.3. Rotational Alignment

As described in Algorithm 1, we use the Kabsch-Umeyama algorithm (Kabsch, 1976; Umeyama, 1991) to rotationally align y to x before calling the denoiser.

Algorithm 1 Rotational Alignment with the Kabsch-Umeyama Algorithm

Require: Noisy Sample $y \in \mathbb{R}^{N \times 3}$, True Sample $x \in \mathbb{R}^{N \times 3}$.
$H \leftarrow x^T y$ ▷ $H \in \mathbb{R}^{3 \times 3}$
$U, S, V^T \leftarrow \text{SVD}(H)$ ▷ $U, V \in \mathbb{R}^{3 \times 3}$
$\mathbf{R}^* \leftarrow U\text{diag}[1, 1, \det(U)\det(V)]V^T$
return $y(\mathbf{R}^*)^T$

Note that both y and x are mean-centered to respect translational equivariance:

$$\sum_{i=1}^{N} y_i = \vec{0} \in \mathbb{R}^3 \tag{29}$$

$$\sum_{i=1}^{N} x_i = \vec{0} \in \mathbb{R}^3 \tag{30}$$

so there is no net translation.

E. Proofs of Theoretical Results

For completeness, we prove the main theoretical results here, as first established by Robbins (1956); Miyasawa (1960); Saremi & Hyvärinen (2019).

E.1. The Denoiser Minimizes the Expected Loss

Here, we prove Equation 6, rewritten here for clarity:

$$\hat{x}(\cdot) \equiv \mathbb{E}[X \mid Y = \cdot] = \underset{f:\mathbb{R}^{N \times 3} \to \mathbb{R}^{N \times 3}}{\arg\min} \; \mathbb{E}_{\substack{X \sim p_X, \varepsilon \sim \mathcal{N}(0, \mathbb{I}_{N \times 3}) \\ Y = X + \sigma\varepsilon}} [\|f(Y) - X\|^2] \tag{31}$$

First, we can decompose the loss over the domain $\mathbb{R}^{N\times 3}$ of Y:

$$\mathbb{E}_{\substack{X\sim p_X,\varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})\\ Y=X+\sigma\varepsilon}}[\|f(Y)-X\|^2] = \mathbb{E}_{X\sim p_X, Y\sim p_Y}[\|f(Y)-X\|^2] \tag{32}$$

$$= \int_{\mathbb{R}^{N\times 3}}\int_{\mathbb{R}^{N\times 3}} \|f(y)-x\|^2 \, p_{X,Y}(x,y)dxdy \tag{33}$$

$$= \int_{\mathbb{R}^{N\times 3}} \underbrace{\int_{\mathbb{R}^{N\times 3}} \|f(y)-x\|^2 \, p_{Y|X}(y\mid x)p_X(x)dxdy}_{l(f,y)} \tag{34}$$

$$= \int_{\mathbb{R}^{N\times 3}} l(f,y)dy \tag{35}$$

where $l(f,y)\geq 0$ for all functions f and inputs y. Hence, any minimizer f^* must minimize the local denoising loss $l(f^*,y)$ at each point $y\in\mathbb{R}^{N\times 3}$. For a fixed $y\in\mathbb{R}^{N\times 3}$, the loss $l(f,y)$ is convex as a function of $f(y)$. Hence, the global minimizer can be found by finding the critical points of $l(f,y)$ as a function of $f(y)$:

$$\nabla_{f(y)} l(f,y) = 0 \tag{36}$$

$$\implies \nabla_{f(y)} \int_{\mathbb{R}^{N\times 3}} \|f(y)-x\|^2 \, p_{Y|X}(y\mid x)p_X(x)dx = 0 \tag{37}$$

$$\implies \int_{\mathbb{R}^{N\times 3}} 2(f^*(y)-x)p_{Y|X}(y\mid x)p_X(x)dx = 0 \tag{38}$$

Rearranging:

$$f^*(y) = \frac{\int_{\mathbb{R}^{N\times 3}} x\, p_{Y|X}(y\mid x)p_X(x)dx}{\int_{\mathbb{R}^{N\times 3}} p_{Y|X}(y\mid x)p_X(x)dx} \tag{39}$$

$$= \frac{\int_{\mathbb{R}^{N\times 3}} x\, p_{Y|X}(y\mid x)p_X(x)dx}{p_Y(y)} \tag{40}$$

$$= \int_{\mathbb{R}^{N\times 3}} x\, \frac{p_{Y|X}(y\mid x)p_X(x)}{p_Y(y)}dx \tag{41}$$

$$= \int_{\mathbb{R}^{N\times 3}} x\, p_{X|Y}(x\mid y)dx \tag{42}$$

$$= \mathbb{E}[X\mid Y=y] \tag{43}$$

$$= \hat{x}(y) \tag{44}$$

by Bayes' rule. Hence, the denoiser as defined by Equation 5 is indeed the minimzer of the denoising loss:

$$\hat{x}(\cdot)\equiv\mathbb{E}[X\mid Y=\cdot] = \underset{f:\mathbb{R}^{N\times 3}\to\mathbb{R}^{N\times 3}}{\arg\min}\ \mathbb{E}_{\substack{X\sim p_X,\varepsilon\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})\\ Y=X+\sigma\varepsilon}}[\|f(Y)-X\|^2] \tag{45}$$

as claimed.

E.2. Relating the Score and the Denoiser

Here, we rederive Equation 7, relating the score function $\nabla\log p_Y$ and the denoiser $\hat{x}$.

Let $X\sim p_X$ defined over $\mathbb{R}^{N\times 3}$ and $\eta\sim\mathcal{N}(0,\mathbb{I}_{N\times 3})$. Let $Y=X+\sigma\eta$, which means:

$$p_{Y|X}(y\mid x) = \mathcal{N}(y;x,\mathbb{I}_{N\times 3}) = \frac{1}{(2\pi\sigma^2)^{\frac{3N}{2}}}\exp\left(-\frac{\|y-x\|^2}{2\sigma^2}\right) \tag{46}$$

Then:

$$\mathbb{E}[X\mid Y=y] = y+\sigma^2\nabla_y\log p_Y(y) \tag{47}$$

To prove this:

$$\nabla_y p_{Y|X}(y \mid x) = -\frac{y-x}{\sigma^2} p_{Y|X}(y \mid x) \tag{48}$$

$$\implies (x-y) p_{Y|X}(y \mid x) = \sigma^2 \nabla_y p_{Y|X}(y \mid x) \tag{49}$$

$$\implies \int_{\mathbb{R}^{N\times 3}} (x-y) p_{Y|X}(y \mid x)\, p_X(x) dx = \int_{\mathbb{R}^{N\times 3}} \sigma^2 \nabla_y p_{Y|X}(y \mid x)\, p_X(x) dx \tag{50}$$

By Bayes' rule:

$$p_{Y|X}(y \mid x) p_X(x) = p_{X,Y}(x,y) = p_{X|Y}(x|y) p_Y(y) \tag{51}$$

and, by definition of the marginals:

$$\int_{\mathbb{R}^{N\times 3}} p_{X,Y}(x,y) dx = p_Y(y) \tag{52}$$

For the left-hand side, we have:

$$\int_{\mathbb{R}^{N\times 3}} (x-y) p_{Y|X}(y \mid x) p_X(x) dx = \int_{\mathbb{R}^{N\times 3}} (x-y) p_{X,Y}(x,y) dx \tag{53}$$

$$= \int_{\mathbb{R}^{N\times 3}} x p_{X,Y}(x,y) dx - \int_{\mathbb{R}^{N\times 3}} y p_{X,Y}(x,y) dx \tag{54}$$

$$= p_Y(y) \left(\int_{\mathbb{R}^{N\times 3}} x p_{X|Y}(x \mid y) dx - y \int_{\mathbb{R}^{N\times 3}} p_{X|Y}(x \mid y) dx \right) \tag{55}$$

$$= p_Y(y) \left(\mathbb{E}[X \mid Y = y] - y \right) \tag{56}$$

For the right-hand side, we have:

$$\sigma^2 \int_{\mathbb{R}^{N\times 3}} \nabla_y p_{Y|X}(y \mid x) p_X(x) dx = \sigma^2 \nabla_y \int_{\mathbb{R}^{N\times 3}} p_{Y|X}(y \mid x) p_X(x) dx \tag{57}$$

$$= \sigma^2 \nabla_y \int_{\mathbb{R}^{N\times 3}} p_{X,Y}(x,y) dx \tag{58}$$

$$= \sigma^2 \nabla_y p_Y(y) \tag{59}$$

Thus,

$$p_Y(y) \left(\mathbb{E}[X \mid Y = y] - y \right) = \sigma^2 \nabla_y p_Y(y) \tag{60}$$

$$\implies \mathbb{E}[X \mid Y = y] = y + \sigma^2 \frac{\nabla_y p_Y(y)}{p_Y(y)} \tag{61}$$

$$= y + \sigma^2 \nabla_y \log p_Y(y) \tag{62}$$

as claimed.

F. Numerical Solvers for Langevin Dynamics

As mentioned in Section 2.2, solving the Stochastic Differential Equation corresponding to Langevin dynamics is often performed numerically. In particular, BAOAB (Leimkuhler & Matthews, 2012; 2015; Sachs et al., 2017) refers to a 'splitting method' that solves the Langevin dynamics SDE by splitting it into three different components labelled by $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{O}$ below:

$$dy = \underbrace{v_y dt}_{\mathcal{A}} \tag{63}$$

$$dv_y = \underbrace{M^{-1} \nabla_y \log p_Y(y) dt}_{\mathcal{B}} \underbrace{- \gamma v_y dt + \sqrt{2\gamma} M^{-\frac{1}{2}} dB_t}_{\mathcal{O}} \tag{64}$$

where both $y, v_y \in \mathbb{R}^d$. This leads to the following update operators:

$$\mathcal{A}_{\Delta t}\begin{bmatrix} y \\ v_y \end{bmatrix} = \begin{bmatrix} y + v_y \Delta t \\ v_y \end{bmatrix} \tag{65}$$

$$\mathcal{B}_{\Delta t}\begin{bmatrix} y \\ v_y \end{bmatrix} = \begin{bmatrix} y \\ v_y + M^{-1}\nabla_y \log p_Y(y)\Delta t \end{bmatrix} \tag{66}$$

$$\mathcal{O}_{\Delta t}\begin{bmatrix} y \\ v_y \end{bmatrix} = \begin{bmatrix} y \\ e^{-\gamma \Delta t} v_y + M^{-\frac{1}{2}}\sqrt{1 - e^{-2\gamma \Delta t}} B \end{bmatrix} \tag{67}$$

where $B \sim \mathcal{N}(0, \mathbb{I}_d)$ is resampled every iteration. As highlighted by Kieninger & Keller (2022), the $\mathcal{A}$ and $\mathcal{B}$ updates are obtained by simply discretizing the updates highlighted in Equation 63 by the Euler method. The $\mathcal{O}$ update refers to a explicit solution of the Ornstein-Uhlenbeck process, which we rederive for completeness in Appendix G.

Finally, the iterates of the BAOAB algorithm are given by a composition of these update steps, matching the name of the method:

$$\begin{bmatrix} y^{(t+1)} \\ v_y^{(t+1)} \end{bmatrix} = \mathcal{B}_{\frac{\Delta t}{2}}\mathcal{A}_{\frac{\Delta t}{2}}\mathcal{O}_{\Delta t}\mathcal{A}_{\frac{\Delta t}{2}}\mathcal{B}_{\frac{\Delta t}{2}}\begin{bmatrix} y^{(t)} \\ v_y^{(t)} \end{bmatrix} \tag{68}$$

G. The Ornstein-Uhlenbeck Process

For completeness, we discuss the distributional solution of the Ornstein-Uhlenbeck process, taken directly from the excellent Leimkuhler & Matthews (2015). In one dimension, the Ornstein-Uhlenbeck Process corresponds to the following Stochastic Differential Equation (SDE):

$$dv_y = -\gamma v_y dt + \sqrt{2\gamma} M^{-\frac{1}{2}} dB_t \tag{69}$$

Multiplying both sides by the integrating factor $e^{\gamma t}$:

$$e^{\gamma t} dv_y = -\gamma e^{\gamma t}(v_y dt + e^{\gamma t}\sqrt{2\gamma} M^{-\frac{1}{2}} dB_t \tag{70}$$

$$\implies e^{\gamma t}(dv_y + \gamma v_y dt) = e^{\gamma t}\sqrt{2\gamma} M^{-\frac{1}{2}} dB_t \tag{71}$$

and identifying:

$$e^{\gamma t}(dv_y + \gamma v_y dt) = d(e^{\gamma t} v_y) \tag{72}$$

We get after integrating from t_1 to t_2, two adjacent time steps of our integration grid:

$$d(e^{\gamma t} v_y) = e^{\gamma t}\sqrt{2\gamma} M^{-\frac{1}{2}} dB_t \tag{73}$$

$$\implies \int_{t_1}^{t_2} d(e^{\gamma t} v_y) = \int_{t_1}^{t_2} e^{\gamma t}\sqrt{2\gamma} M^{-\frac{1}{2}} dB_t \tag{74}$$

$$\implies e^{\gamma t_2} v_y(t_2) - e^{\gamma t_1} v_y(t_1) = \sqrt{2\gamma} M^{-\frac{1}{2}} \int_{t_1}^{t_2} e^{\gamma t} dB_t \tag{75}$$

Now, for a Wiener process B_t, if $g(t)$ is a deterministic function, $\int_{t_1}^{t_2} g(t) dB_t$ is distributed as $\mathcal{N}\left(0, \int_{t_1}^{t_2} g(t)^2 dt\right)$ by Itô's integral. Thus, applying this result to $g(t) = e^{\gamma t}$, we get:

$$e^{\gamma t_2} v_y(t_2) - e^{\gamma t_1} v_y(t_1) = \sqrt{2\gamma} M^{-\frac{1}{2}} \mathcal{N}\left(0, \frac{e^{2\gamma t_2} - e^{2\gamma t_1}}{2\gamma}\right) \tag{76}$$

$$\implies v_y(t_2) = e^{-\gamma(t_2 - t_1)} v_y(t_1) + \sqrt{2\gamma} M^{-\frac{1}{2}} e^{-\gamma t_2} \mathcal{N}\left(0, \frac{e^{2\gamma t_2} - e^{2\gamma t_1}}{2\gamma}\right) \tag{77}$$

$$= e^{-\gamma(t_2 - t_1)} v_y(t_1) + \sqrt{2\gamma} M^{-\frac{1}{2}} \sqrt{\frac{1 - e^{2\gamma(t_1 - t_2)}}{2\gamma}} \mathcal{N}(0, 1) \tag{78}$$

$$= e^{-\gamma(t_2 - t_1)} v_y(t_1) + M^{-\frac{1}{2}} \sqrt{1 - e^{2\gamma(t_1 - t_2)}} \mathcal{N}(0, 1) \tag{79}$$

In the $N \times 3$ dimensional case, as the Wiener processes are all independent of each other, we directly get:

$$v_y(t_2) = e^{-\gamma(t_2-t_1)} v_y(t_1) + M^{-\frac{1}{2}} \sqrt{1 - e^{2\gamma(t_1-t_2)}} \mathcal{N}\left(0, \mathbb{I}_{N\times 3}\right) \tag{80}$$

Setting $\Delta t = t_2 - t_1$, we get the form of the $\mathcal{O}$ operator (Equation 65) of the BAOAB integrator in Appendix F.

H. Parallelizing Sampling with Multiple Independent Chains

Our sampling strategy batches peptides in order to increase throughput. Another potential method to increase throughput (but which we did not employ for the results in this paper) is to sample multiple chains in parallel.

This can be done by initializing multiple chains: $y_1^{(0)}, \ldots, y_{N_{\text{ch}}}^{(0)}$, where:

$$y_{\text{ch}}^{(0)} = x^{(0)} + \sigma \varepsilon_{\text{ch}}^{(0)} \tag{81}$$

where $\varepsilon_{\text{ch}}^{(0)} \overset{\text{iid}}{\sim} \mathcal{N}(0, \mathbb{I}_{N\times 3})$ for $\text{ch} = 1, \ldots, N_{\text{ch}}$ are all independent of each other. Then, the chains can be evolved independently withindependent walk steps (Equation 3) and denoised with independent jump steps (Equation 7). This independence allows batching over the $y_{\text{ch}}^{(t)}$ over all chains ch at each iteration t.

Note that at $t = 0$, the chains are correlated as they are all initialized from the same $x^{(0)}$. However, if the number of samples per chain is large enough, the chains are no longer correlated, as they have now mixed into the stationary distribution.

I. Preliminary Results on Macrocycles

One of the most important aspects of JAMUN is its highly general input, a point cloud, unlike other protein ensemble models that are frequently more bespoke, using dihedral or frame representations. While this may be a slight disadvantage for us in the protein space, it also makes us extremely flexible and easily transferable to other modalities, unlike existing models in this space. Here, we demonstrate that on macrocyclic peptides with several non-canonical residues.

The exploration of conformational ensembles in macrocyclic peptides is crucial due to their emerging role as therapeutic modalities. These molecules present significant challenges in computational modeling because of their conformational diversity and inherent geometric constraints. In fact, molecular dynamics trajectories for these molecules are particularly slow as good classical forcefields are unavailable and it is necessary to use quantum mechanical calculations to compute forces. Macrocyclic peptides are also extremely unwieldy in their "open", most common conformations, forming hydrogen bond networks with water. However, those macrocycles that are able to occupy smaller "crumpled" conformations are greasy and able to permeate through biological membranes, making them more suitable for biodelivery. Here, as an example, we use macrocycles from the CREMP dataset (Grambow et al., 2024), generate ensembles with the CREST protocol (Pracht & Grimme, 2020), and benchmark the resulting conformers against the RINGER model (Grambow et al., 2023). Figure 18 illustrates the transferability of JAMUN to macrocyclic peptides.

It is clear that we are able to recover most basins sampled, even though it does seem like there are new basins uncovered. We note that our outputs look significantly more diffusive than the ground truth. The main reason for this is that the CREST data is clustered and filtered to represent the local minima, whereas JAMUN is designed to sample entire distributions. In some sense, the data is strictly not complete for the task JAMUN is designed for. It is impressive that in spite of this JAMUN learns enough from the denoising "jump" step with very local data to still perform the Langevin dynamics "walk" step and sample multiple basins.

We find that for 4-mers, which is what we trained our model for, we are able to recover all the sampled basins. We also attempt to run inference for 5 and 6-mers with the same model to test generalizability.

While this is a preliminary study, it points to the potential of JAMUN being used universally, not just for proteins, and in particular shows that it is effective even in a low-data regime.

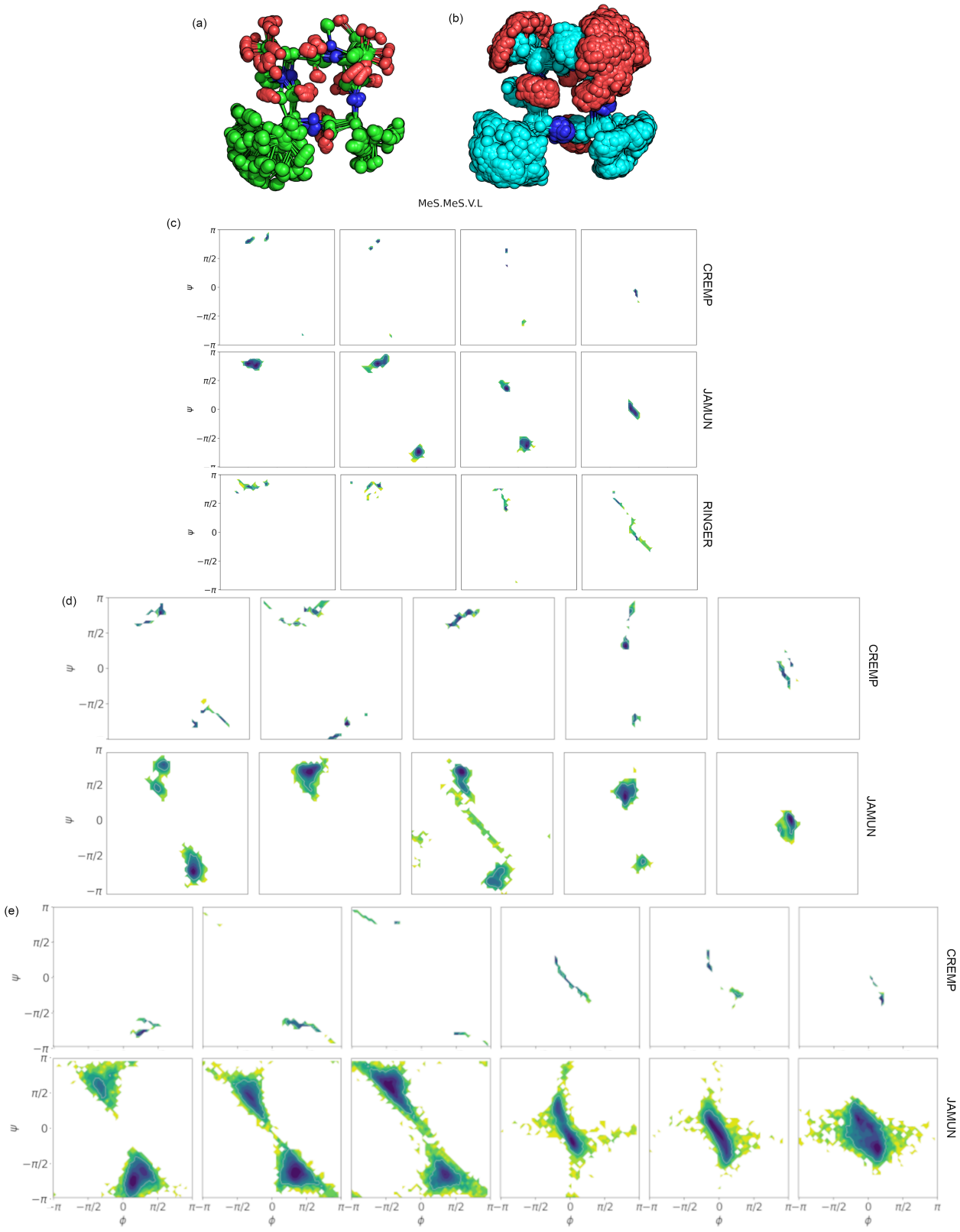


Figure 18. Macrocycle results trained on 4AA: 3D image of the (a) CREMP (green) and (b) JAMUN (cyan) MeS.MeS.V.L macrocycle. (c) Ramachandran plot for CREMP, JAMUN, and RINGER samples of the 4AA MeS.MeS.V.L macrocycle. (d) Ramachandran plots for CREMP and JAMUN samples of the 5AA F.Q.L.G.Met macrocycle.(e) Ramachandran plots for CREMP and JAMUN samples of the 6AA Mes.T.Q.Mei.V.W macrocycle.