

A Data details

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 1b. These are not ideal analogues of amino acids in proteins due to local charge interactions as well as lack of steric effects.

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 1b, 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 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.

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) [17, 82], a trajectory stratification based enhanced sampling algorithm. The protein is represented by the AMBER03 force field [63]. The simulations are performed at 300 K with the BAOAB integrator [49] in OpenMM [20]; LINCS is used to constrain the lengths of bonds to hydrogen atoms [30]; Particle Mesh Ewald is used to calculate electrostatics [15]; the step size was 2 fs. The systems are solvated with TIP3P water models and equilibrated under NVT and NPT ensembles for 100ps each.

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.

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 2: Simulation conditions vary significantly across the different datasets, allowing us to demonstrate the broad applicability of our approach.

B Results for CAPPED 2AA

B.1 Further Results on internally simulated data

Here, we compute the ratio of the decorrelation times for the backbone and sidechain torsions in JAMUN and the reference MD data. Figure 6 highlights how sampling in the smoothed space Y compared to the original space X enables much faster decorrelation.

In Table 3, we compare JAMUN to the reference MD, shortened by a factor of 10 along the variables mentioned in Section 5, using the Jensen-Shannon distance to the full reference MD data. JAMUN outperforms this shortened MD trajectory across all metrics, even though it takes approximately $2\times$ longer to sample than JAMUN, from Figure 2a.

Trajectory	Backbone Torsions	Sidechain Torsions	All Torsions	TICA-0	TICA-0,1	Metastable Probs
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 ($10\times$ 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

Table 3: Comparison of Jensen-Shannon distances between JAMUN and the reference MD (shortened by a factor of 10), averaged over the test peptides in CAPPED 2AA. Note that this shortened reference MD takes $2\times$ longer to sample as JAMUN.

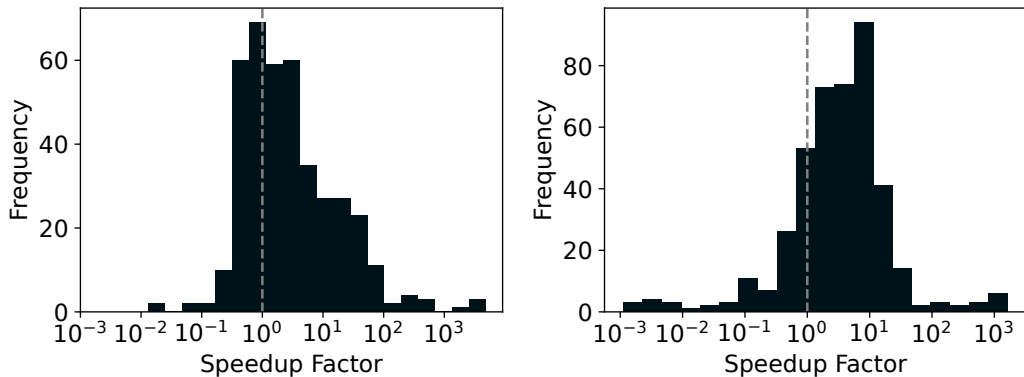


Figure 6: Speedups defined as the ratio between decorrelation times between the reference MD and JAMUN for backbone (left) and sidechain (right) torsions for all test peptides in CAPPED 2AA.

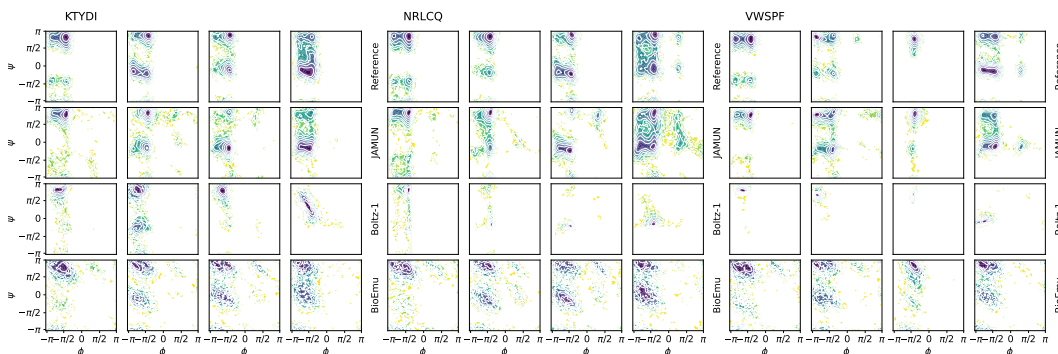


Figure 7: Ramachandran plots for test peptides NRLCQ, VWSPF and KTYDI in UNCAPPED 5AA.

C Related Work

Protein structure prediction only requires prediction of a few folded states, and is usually trained on crystallized structures of proteins which can be quite different from their native states. The size of these proteins are significantly larger than the peptides we study here. On the other hand, conformational ensemble generation requires many samples from the Boltzmann distribution to capture the effects of solvent atoms and intramolecular interactions, even if dynamics is not modelled explicitly. Research has found that protein structure prediction models tend to struggle with capturing this diversity, especially when sampling the conformations of flexible domains [68, 12, 2] and rarer conformations [47] not found in the Protein Data Bank (PDB) [9]. Another issue is that the quality of these models' predictions can be dependent on multiple sequence alignment (MSA) information, a preprocessing step where sequence databases are queried for similar protein sequences which indicate evolutionary conservation patterns as a supplementary input to the model. In fact, some of the first attempts to sample alternative conformations of proteins with AlphaFold2 were performed by subsampling [16], clustering [87] or manipulating MSA information [28]. Sequence-to-structure models such as ESMFold [52] which do not require MSA information can be faster but tend to produce less physically accurate structures, as noted by Lu et al. [54]. Unfortunately, MSA information can be quite unreliable for small peptides due to the presence of many hits. In fact, popular MSA software suites such as MMseqs2 [79] querying will, by default, simply return an empty MSA for the peptides we study here. Nevertheless, previous research [66] has shown that AlphaFold2 can still learn an accurate energy function for protein structures without MSA co-evolution information.

Jussupow and Kaila [38] finds that AlphaFold2's predicted local distance difference test (pLDDT) and predicted aligned error (PAE) scores correlate with local protein dynamics and global conformational flexibility respectively. They use these scores to parametrize an additional harmonic potential for coarse-grained MD with the MARTINI [55] force field. AlphaFlow [35] develops flow-matching over the quotient space of 3D positions modulo rotations to learn a distribution over 3D positions of

the β -carbon atoms, outperforming MSA subsampling with AlphaFold2 over the Protein Data Bank (PDB) [9]. Distributional Graphormer [91] parametrizes a diffusion model over α -carbon positions, also trained on the PDB. Str2Str [54] proposes a noising-denoising process along a range of noise levels for sampling backbone atom coordinates, followed by regression of side-chain coordinates with the rotamer-based FASPR [34] package. BioEmu [51] is a diffusion model built using the EvoFormer stack from AlphaFold2 [37]. BioEmu is pretrained on 200 million protein structures from the AlphaFold Protein Structure Database [84] and finetuned on over 200ms of MD data, which are orders of magnitude larger than the datasets we benchmark here.

A predecessor of many of the models discussed in 4, ‘Two for One’ [7] 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.

There have also been efforts to build ML models for taking longer MD time-steps [45, 73, 33] and for approximating conformations of large proteins [91, 35]. These methods rely on hand-crafted featurizations (eg. backbone torsion angles). In practice, this has made generalization to unseen molecules challenging for these models as well. The above models are often classified either as Boltzmann Generators or Boltzmann Emulators. Boltzmann Generators are guaranteed to draw unbiased samples from the Boltzmann distribution, while Boltzmann Emulators do not have this guarantee. Strictly speaking, JAMUN is a Boltzmann Emulator.

JAMUN is a walk-jump sampling method which uses an $SE(3)$ -equivariant neural network for denoising. WJS is built on the seminal work of Neural Empirical Bayes [69], and has been used in voxelized molecule generation [61, 62] and protein sequence generation [23]. Our work is the first to our knowledge to apply walk-jump sampling to point clouds.

D Overview of Denoiser

The denoiser is a $SE(3)$ -equivariant graph neural network. The graph is defined by a radial cutoff of 10\AA in y . The overall computation performed by the denoiser is shown in Figure 8, with the initial embedding, message-passing and output head blocks shown in Figure 9, Figure 10 and Figure 11 respectively.

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

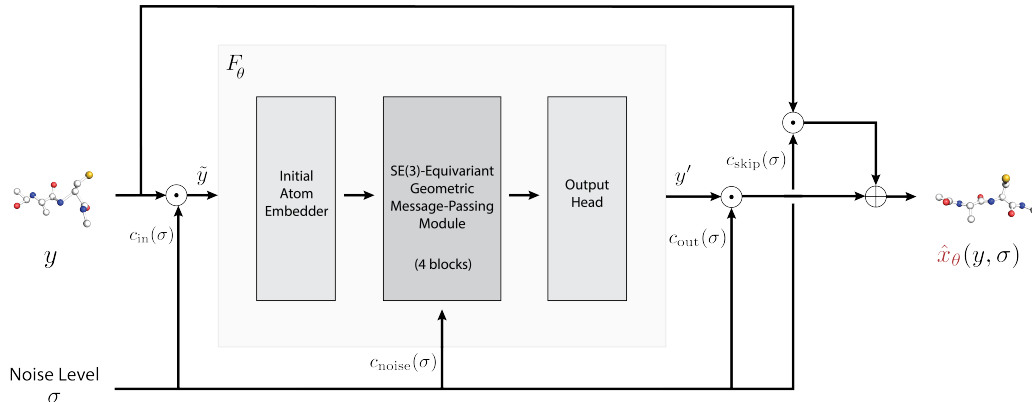


Figure 8: Overview of the denoiser network \hat{x}_θ . The submodule F_θ sees input atom coordinates $\tilde{y} = c_{\text{in}}(\sigma)y$ and outputs predicted atom coordinates y' , which gets scaled and added to a noise-conditional skip connection to finally obtain $\hat{x}_\theta(y)$.

The hidden features $h^{(n)}$ for $n = 0, \dots, 4$ 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.

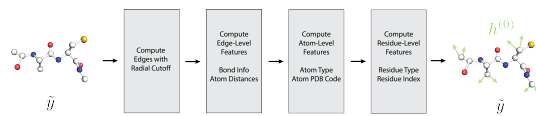


Figure 9: Overview of the initial embedder in the denoiser network, creating initial features $h^{(0)}$ at each atom and edge.

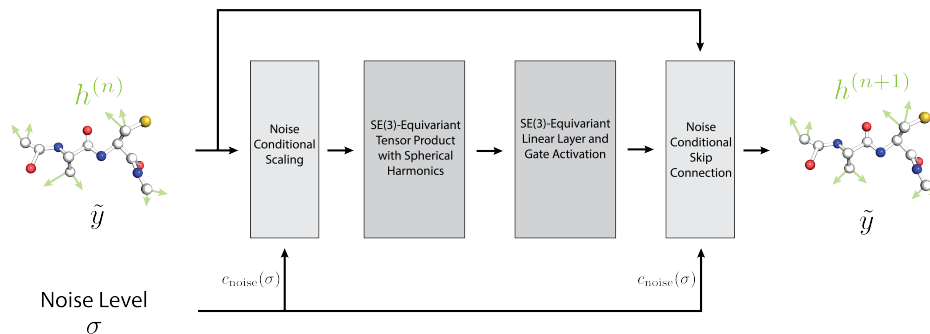


Figure 10: Overview of a single $SE(3)$ -equivariant message-passing block (indexed by n) in the denoiser network. There are four such blocks iteratively updating the atom features from $h^{(0)}$ to $h^{(4)}$. The atom coordinates denoted by $\tilde{y} = c_{\text{in}}(\sigma)y$ (and hence, the edge features) are unchanged throughout these blocks.

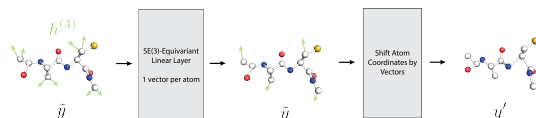


Figure 11: Overview of the output head, which predicts the coordinates $y' = F_{\theta}(c_{\text{in}}(\sigma)y, c_{\text{noise}}(\sigma))$.

788 E Normalization

789 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}
 790 represent the ‘normalized’ input y , as seen by the network F_θ :

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

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

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

792 Note that here, we define E such that Note that this is distinct from the normalization chosen by
 793 [40, 41], which normalizes $\|y\|$ directly. The intuition behind this normalization is that the GNN
 794 model F_θ does not operate on atom positions y directly, but instead uses the relative vectors $y_i - y_j$
 795 to account for translation invariance, and controlling this object directly ensures that the topology of
 796 the graph does not change with varying noise level σ .

797 To achieve this, we compute:

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

798 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
 799 full derivation can be found in Section E.1.

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

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

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

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

804 The noise normalization is a scaled version of the recommendation of $\frac{1}{4} \ln \sigma$ for images in Karras
 805 et al. [40, 41].

806 E.1 Input Normalization

807 Fix an $(i, j) \in E$ from Equation 11. As $\varepsilon_i, \varepsilon_j \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \mathbb{I}_3)$, we have $\varepsilon_i - \varepsilon_j \sim \mathcal{N}(0, 2\mathbb{I}_3)$ from
 808 the closure of the multivariate Gaussian under linear combinations. Thus, for each component
 809 $d = 1, 2, 3$, we have: $(\varepsilon_i - \varepsilon_j)_{(d)} \sim \mathcal{N}(0, 2)$ and hence:

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

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

810 We can now compute:

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

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

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

812 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
813 Equation 19 and our intended normalization given by Equation 11:

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

814 E.2 Output Normalization

815 The derivation here is identical that of [40, 41], but with our normalization. The denoising loss at a
816 single noise level is:

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

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

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

818 where:

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

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

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

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

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

822 and hence:

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

823 where C was defined above. Now, to minimize $c_{\text{out}}(\sigma)$ to maximize reuse and avoid amplifying
824 network errors, as recommended by Karras et al. [40, 41]:

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

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

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

826 The noise normalization is chosen as $c_{\text{noise}}(\sigma) = \log_{10} \sigma$, a scaled version of the recommendation of
827 $\frac{1}{4} \ln \sigma$ for images in Karras et al. [40, 41].

828 From Equation 22, we set $\lambda(\sigma) = \frac{1}{c_{\text{out}}(\sigma)^2}$ to normalize the loss at all noise levels, as in Karras et al.
829 [40, 41].

830 E.3 Rotational Alignment

831 As described in Algorithm 1, we use the Kabsch-Umeyama algorithm [39, 81] to rotationally align y
832 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 $x \in \mathbb{R}^{N \times 3}$.

$H \leftarrow x^T y$ $\triangleright H \in \mathbb{R}^{3 \times 3}$
 $U, S, V^T \leftarrow \text{SVD}(H)$ $\triangleright U, V \in \mathbb{R}^{3 \times 3}$
 $\mathbf{R}^* \leftarrow U \text{diag}[1, 1, \det(U) \det(V)] V^T$
return $y(\mathbf{R}^*)^T$

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

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

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

834 so there is no net translation.

835 F Proofs of Theoretical Results

836 For completeness, we prove the main theoretical results here, as first established by Robbins [65],
837 Miyasawa [57], Saremi and Hyvärinen [69].

838 F.1 The Denoiser Minimizes the Expected Loss

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

$$\hat{x}(\cdot) \equiv \mathbb{E}[X \mid Y = \cdot] = \arg \min_{f: \mathbb{R}^{N \times 3} \rightarrow \mathbb{R}^{N \times 3}} \mathbb{E}_{X \sim p_{X, \varepsilon} \sim \mathcal{N}(0, \mathbb{I}_{N \times 3})} [\|f(Y) - X\|^2] \quad (31)$$

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

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

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

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

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

841 where $l(f, y) \geq 0$ for all functions f and inputs y . Hence, any minimizer f^* must minimize the local
 842 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
 843 a function of $f(y)$. Hence, the global minimizer can be found by finding the critical points of $l(f, y)$
 844 as a function of $f(y)$:

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

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

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

845 Rearranging:

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

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

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

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

$$= \mathbb{E}[X | Y = y] \quad (43)$$

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

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

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

848 as claimed.

849 F.2 Relating the Score and the Denoiser

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

851 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:

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

852 Then:

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

853 To prove this:

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

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

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

854 By Bayes' rule:

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

855 and, by definition of the marginals:

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

856 For the left-hand side, we have:

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

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

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

$$= p_Y(y) (\mathbb{E}[X | Y = y] - y) \quad (56)$$

857 For the right-hand side, we have:

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

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

$$= \sigma^2 \nabla_y p_Y(y) \quad (59)$$

858 Thus,

$$p_Y(y) (\mathbb{E}[X | Y = y] - y) = \sigma^2 \nabla_y p_Y(y) \quad (60)$$

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

$$= y + \sigma^2 \nabla_y \log p_Y(y) \quad (62)$$

859 as claimed.

860 **G Numerical Solvers for Langevin Dynamics**

861 As mentioned in Section 2.2, solving the Stochastic Differential Equation corresponding to Langevin
 862 dynamics is often performed numerically. In particular, BAOAB [48, 50, 67] refers to a ‘splitting
 863 method’ that solves the Langevin dynamics SDE by splitting it into three different components
 864 labelled by \mathcal{A} , \mathcal{B} and \mathcal{O} below:

$$dy = \underbrace{v_y dt}_{\mathcal{A}} \quad (63)$$

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

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

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

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

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

where $B \sim \mathcal{N}(0, \mathbb{I}_d)$ is resampled every iteration. As highlighted by Kieninger and Keller [42], 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 H.

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

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

H The Ornstein-Uhlenbeck Process

For completeness, we discuss the distributional solution of the Ornstein-Uhlenbeck process, taken directly from the excellent Leimkuhler and Matthews [50]. In one dimension, the Ornstein-Uhlenbeck Process corresponds to the following Stochastic Differential Equation (SDE):

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

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

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

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

and identifying:

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

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

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

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

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

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:

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

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

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

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

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

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

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

885 I Parallelizing Sampling with Multiple Independent Chains

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

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

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

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

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

897 J Model transferability exploration using Macrocycles

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

903 The exploration of conformational ensembles in macrocyclic peptides is crucial due to their emerging
 904 role as therapeutic modalities. These molecules present significant challenges in computational
 905 modeling because of their conformational diversity and inherent geometric constraints. In fact,
 906 molecular dynamics trajectories for these molecules are particularly slow as good classical forcefields
 907 are unavailable and it is necessary to use quantum mechanical calculations to compute forces.
 908 Macrocyclic peptides are also extremely unwieldy in their "open", most common conformations,
 909 forming hydrogen bond networks with water. However, those macrocycles that are able to occupy
 910 smaller "crumpled" conformations are greasy and able to permeate through biological membranes,
 911 making them more suitable for biodelivery. Here, as an example, we use macrocycles from the
 912 CREMP dataset [27], generate ensembles with the CREST protocol [64], and benchmark the resulting
 913 conformers against the RINGER model [26]. Figure 12 illustrates the transferability of JAMUN to
 914 macrocyclic peptides.

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

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

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

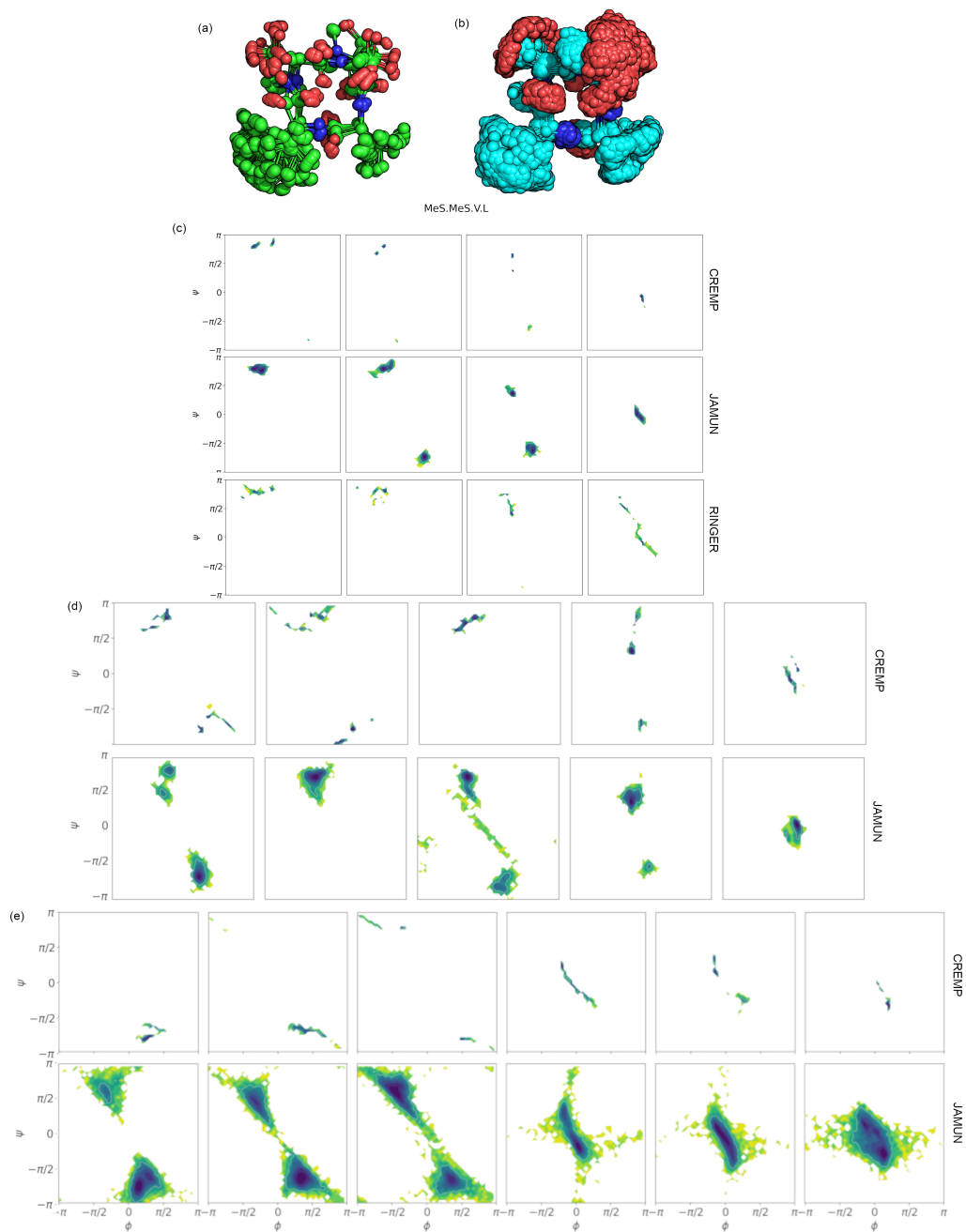


Figure 12: Macrocycle results trained on 4AA: 3D image of the (a) CREMP (green) and (b) JAMUN (cyan) MeS.MeS.V.L macrocycle. (c) Ramachandran plot for CREMP, JAMUN, and RINGER samples of the 4AA MeS.MeS.V.L macrocycle. (d) Ramachandran plot for CREMP, JAMUN, and RINGER samples of the 5AA F.Q.L.G.Met macrocycle. (e) Ramachandran plot for CREMP and JAMUN the 6AA Mes.T.Q.Mei.V.W macrocycle.