Generative Modeling of Molecular Dynamics Trajectories

Bowen Jing[*1] **Hannes Stärk**[*1] **Tommi Jaakkola**[1] **Bonnie Berger**[1 2]
[1]CSAIL, Massachusetts Institute of Technology
[2]Dept. of Mathematics, Massachusetts Institute of Technology
{bjing, hstark}@mit.edu, tommi@csail.mit.edu, bab@mit.edu

Abstract

Molecular dynamics (MD) is a powerful technique for studying microscopic phenomena, but its computational cost has driven significant interest in the development of deep learning-based surrogate models. We introduce generative modeling of molecular trajectories as a paradigm for learning flexible multi-task surrogate models of MD from data. By conditioning on appropriately chosen frames of the trajectory, we show such generative models can be adapted to diverse tasks such as forward simulation, transition path sampling, and trajectory upsampling. By alternatively conditioning on part of the molecular system and inpainting the rest, we also demonstrate the first steps towards dynamics-conditioned molecular design. We validate the full set of these capabilities on tetrapeptide simulations and show preliminary results on scaling to protein monomers. Altogether, our work illustrates how generative modeling can unlock value from MD data towards diverse downstream tasks that are not straightforward to address with existing methods or even MD itself. Code is available at https://github.com/bjing2016/mdgen.

1 Introduction

Numerical integration of Newton's equations of motion at atomic scales, known as *molecular dynamics* (MD), is a widely-used technique for studying diverse molecular phenomena in chemistry, biology, and other molecular sciences (Alder and Wainwright, 1959; Rahman, 1964; Verlet, 1967; McCammon et al., 1977). While general and versatile, MD is computationally demanding due to the large separation in timescales between integration steps and relevant molecular phenomena. Thus, a vast body of literature spanning several decades aims to accelerate or enhance the sampling efficiency of MD simulation algorithms (Ryckaert et al., 1977; Darden et al., 1993; Sugita and Okamoto, 1999; Laio and Parrinello, 2002; Anderson et al., 2008; Shaw et al., 2009). More recently, learning surrogate models of MD has become an active area of research in deep generative modeling (Noé et al., 2019; Zheng et al., 2023; Klein et al., 2024; Schreiner et al., 2024; Jing et al., 2024). However, existing training paradigms fail to fully leverage the rich dynamical information in MD training data, restricting their applicability to a limited set of downstream problems.

In this work, we propose MDGEN, a novel paradigm for fast, general-purpose surrogate modeling of MD based on *direct generative modeling of simulated trajectories*. Different from previous works, which learn the autoregressive transition density or equilibrium distribution of MD, we formulate end-to-end generative modeling of full trajectories viewed as *time-series* of 3D molecular structures. Akin to how image generative models were extended to videos (Ho et al., 2022), our framing of the problem augments single-structure generative models with an additional time dimension, opening the door to a larger set of forward and inverse problems to which our model can be applied. When

[*]Equal contribution.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

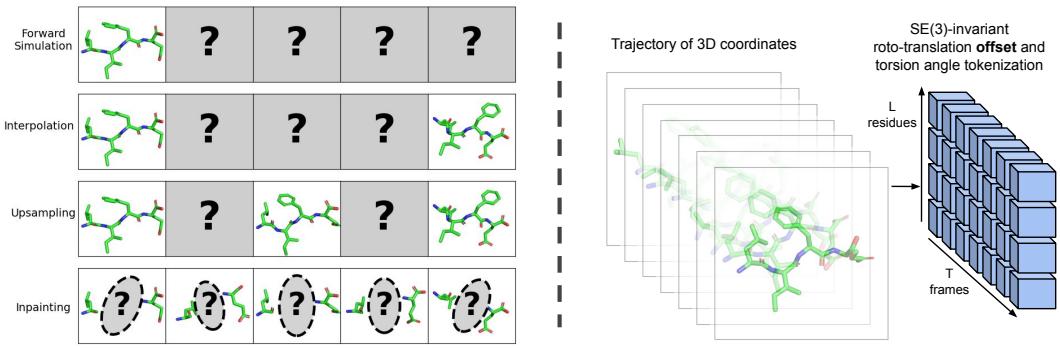

Figure 1: (*Left*) Tasks: generative modeling of MD trajectories addresses several tasks by conditioning on different parts of a trajectory. (*Right*) Method: We tokenize trajectories of T frames and L residues into an $(T \times L)$-array of SE(3)-invariant tokens encoding roto-translation offsets from *key frames* and torsion angles. Using *stochastic interpolants*, we generate arrays of such tokens from Gaussian noise.

provided (and conditioned on) the initial "frame" of a given system, such generative models serve as familiar surrogate forward simulators of the reference dynamics. However, by providing other kinds of conditioning, these "molecular video" generative models also enable highly flexible applications to a variety of inverse problems not possible with existing surrogate models. In sum, we formulate and showcase the following novel capabilities of MDGEN:

- *Forward simulation*—given the initial frame of a trajectory, we sample a potential time evolution of the molecular system.
- *Interpolation*—given the frames at the *two endpoints* of a trajectory, we sample a plausible path connecting the two. In chemistry, this is known as *transition path sampling* and is important for studying reactions and conformational transitions.
- *Upsampling*—given a trajectory with timestep Δt between frames, we upsample the "frame-rate" by a factor of M to obtain a trajectory with timestep $\Delta t/M$. This infers fast motions from trajectories saved at less frequent intervals.
- *Inpainting*—given part of a molecule and its trajectory, we *generate the rest of the molecule* (and its time evolution) to be consistent with the known part of the trajectory. This ability could be applied to design molecules to scaffold desired dynamics.

These tasks are conceptually illustrated in Figure 1. While the forward simulation task aligns with the typical modeling paradigm of approximating the data-generating process, the others represent novel capabilities on scientifically important inverse problems *not straightforward to address even with MD itself*. As such, our framework presents a new perspective on how to unlock value from MD simulation with machine learning towards diverse downstream objectives. We highlight further exciting possibilities opened up by our framework in Section 5.

We demonstrate our framework on MD simulations of tetrapeptides (i.e., length-4 peptides), with preliminary extensions to full-sized protein monomers. To do so, we parameterize molecular trajectories in terms of sidechain torsions and residue offsets with respect to conditioning *key frames*, obtaining a generative modeling task over a 2D array of $SE(3)$-invariant tokens rather than residue frames or point clouds. In this parameterization, we can then employ a Scalable Interpolant Transformer (SiT) (Ma et al., 2024) as our flow-based generative backbone, avoiding the more restrictive geometric architectures commonly used for molecular structure. Furthermore, by replacing the time-wise attention in SiT with the long-context architecture Hyena (Poli et al., 2023), we provide proof-of-concept of scaling up to trajectories of 100k frames, enabling a wide range of timescales and dynamical processes to be captured with a single model generation.

We evaluate MDGEN on the forward simulation, interpolation, upsampling, and inpainting tasks on tetrapeptides in a *transferable* setting (i.e., unseen test peptides). Our model accurately reproduces free energy surfaces and dynamical content such as torsional relaxation and Markov state fluxes, provides realistic transition paths between arbitrary pairs of metastable states, and recovers fast dynamical phenomena below the sampling threshold of coarse-timestep trajectories. In preliminary

steps toward dynamics-scaffolded design, we show that molecular inpainting with MDGEN obtains much higher sequence recovery than inverse folding methods based on one or two static frames. Finally, we evaluate MDGEN on simulation of proteins and find that it outperforms MSA subsampling with AlphaFold (Del Alamo et al., 2022) in terms of recovering ensemble statistical properties.

2 Background

Molecular dynamics. At a high level, the aim of molecular dynamics is the integrate the equations of motion $M_i \ddot{\mathbf{x}}_i = -\nabla_{\mathbf{x}_i} U(\mathbf{x}_1 \dots \mathbf{x}_N)$ for each particle i in a molecular configuration $(\mathbf{x}_1 \dots \mathbf{x}_N) \in \mathbb{R}^{3N}$, where M_i is the mass and U is the potential energy function (or *force field*) $U : \mathbb{R}^{3N} \to \mathbb{R}$. However, these equations of motion are often modified to include a *thermostat* in order to model contact with surroundings at a given temperature. For example, the widely-used *Langevin thermostat* transforms the equations of motion into a stochastic diffusion process:

$$d\mathbf{x}_i = \mathbf{p}_i / M_i \, dt, \quad d\mathbf{p}_i = -\nabla_{\mathbf{x}_i} U \, dt - \gamma \mathbf{p}_i \, dt + \sqrt{2 M_i \gamma k T} \, d\mathbf{w} \tag{1}$$

where $\mathbf{p}_i$ are the momenta. By design, this process converges to the *Boltzmann distribution* of the system $p(\mathbf{x}_1 \dots \mathbf{x}_N) \propto e^{-U/kT}$. To incorporate interactions with solvent molecules—ubiquitous in biochemistry—one includes a box of surrounding solvent molecules as part of the molecular system (explicit solvent) or modifies the force field U to model their effects (implicit solvent). In either case, only the positions $\mathbf{x}_i$ of non-solvent atoms are of interest, and their time evolution constitutes (for our purposes) the *MD trajectory*.

Deep learning for MD. An emerging body of work seeks to approximate the distributions over configurations $\mathbf{X} = (\mathbf{x}_1 \dots \mathbf{x}_N)$ arising from Equation 1 with deep generative models. Fu et al. (2023), Timewarp (Klein et al., 2024), and ITO (Schreiner et al., 2024) learn the *transition density* $p(\mathbf{X}_{t+\Delta t} \mid \mathbf{X}_t)$ and emulate MD trajectories via simulation rollouts of the learned model. On the other hand, *Boltzmann generators* (Noé et al., 2019; Köhler et al., 2021; Garcia Satorras et al., 2021; Midgley et al., 2022, 2024) directly approximate the stationary Boltzmann distribution, forgoing any explicit modeling of dynamics. In particular, Boltzmann-targeting diffusion models trained with frames from MD trajectories have demonstrated promising scalability and generalization to protein systems (Zheng et al., 2023; Jing et al., 2024). However, these works have focused exclusively on forward simulation and have not explored joint modeling of entire trajectories $(\mathbf{X}_t \dots \mathbf{X}_{t+N\Delta t})$ or the inverse problems accessible under such a formulation.

Stochastic interpolants. We build our MD trajectory generative model under the *stochastic interpolants* framework: Given a continuous distribution $p_1 \equiv p_{\text{data}}$ over $\mathbb{R}^n$, stochastic interpolants (Albergo and Vanden-Eijnden, 2022; Albergo et al., 2023; Lipman et al., 2022; Liu et al., 2022), provide a method for learning continuous flow-based models $d\mathbf{x} = v_\theta(\mathbf{x}, t) \, dt$ transporting a prior distribution p_0 (e.g., $p_0 \equiv \mathcal{N}(0, \mathbf{I})$) to the data p_1. To do so, one defines intermediate distributions $\mathbf{x}_t \sim p_t, t \in (0, 1)$ via $\mathbf{x}_t = \alpha_t \mathbf{x}_1 + \sigma_t \mathbf{x}_0$ where $\mathbf{x}_0 \sim p_0$ and $\mathbf{x}_1 \sim p_1$ and the interpolation path satisfies $\alpha_0 = \sigma_1 = 0$ and $\alpha_1 = \sigma_0 = 1$. A neural network $v_\theta : \mathbb{R}^n \times [0, 1] \to \mathbb{R}^n$ is trained to approximate the time-evolving flow field

$$v_\theta(\mathbf{x}_t, t) \approx v(\mathbf{x}_t, t) \equiv \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1 | \mathbf{x}_t}[\dot{\alpha}_t \mathbf{x}_1 + \dot{\sigma}_t \mathbf{x}_0] \tag{2}$$

which satisfies the transport equation $\partial p_t / \partial t + \nabla \cdot (p_t v_t) = 0$. Hence, at convergence, noisy samples $\mathbf{x}_0 \sim p_0$ can be evolved under v_θ to obtain data samples $\mathbf{x}_1 \sim p_1$. When parameterized with transformers (Vaswani et al., 2017), stochastic interpolants are state-of-the-art in image generation (Esser et al., 2024). In particular, we adopt the notation, architecture, and training framework of Scalable Interpolant Transformer (SiT) (Ma et al., 2024), to which we refer for further exposition.

3 Method

3.1 Tokenizing Peptide Trajectories

Given a chemical specification of a molecular system with N atoms, our aim is to learn a generative model over time-series $\boldsymbol{\chi} \equiv [\mathbf{X}_1, \dots \mathbf{X}_T]$ of corresponding molecular structures $\mathbf{X}_i \in \mathbb{R}^{3N}$ for some trajectory length T. In this work, we specialize to MD trajectories of short peptides (Sections 4.1–4.4) or single-chain proteins (4.4). Thus, our chemical specifications are amino acid sequences

$A = \{1 \dots 20\}^L$, and we adopt an $SE(3)$-based parameterization of peptide structures (Jumper et al., 2021; Yim et al., 2023). In this parameterization, the all-atom coordinates of each amino acid residue are *implicitly* described by a roto-translation (i.e., element of $SE(3)$) corresponding to the rigid body motion of the residue, and seven torsion angles describing its conformation:

$$\boldsymbol{\chi}_t^l = [g, \tau_1, \dots \tau_7], \quad g \in SE(3), \tau \in \mathbb{T}, \quad \boldsymbol{\chi} \in \left(\left[SE(3) \times \mathbb{T}^7 \right]^L \right)^T \tag{3}$$

Throughout, subscripts indicate time and superscripts residue indices. The undefined torsion angles can be randomized and are unsupervised for residues with fewer than seven torsion angles.

Traditionally, equivariant architectures have been required for geometry-aware processing of polypeptide structures. However, to learn a scalable generative model over this space of roto-translations and torsion angles, we seek to represent each $\boldsymbol{\chi}_t^l$ in terms of an $SE(3)$-invariant feature vector—a *token* suitable for processing by vanilla transformers. To obtain such a vector, we leverage the fact that we are concerned with *conditional trajectory generation*—meaning that there always exists at least one frame in the trajectory with un-noised roto-translations, which we do not need to generate and can reference in the modeling process. Inspired by analogy to video compression, we refer to such frames as *key frames*. We can then obtain $SE(3)$-invariant tokens by parameterizing the roto-translations of remaining structures *relative* to the key frames.

In more detail, given K key frames at times $t_1 \dots t_K$ we tokenize residue j in frame t as:

$$\boldsymbol{\chi}_t^j = \left[\phi\left([g_{t_1}^j]^{-1} g_t^j \right), \dots, \phi\left([g_{t_K}^j]^{-1} g_t^j \right), \psi([\tau_t^j]_1), \dots \psi([\tau_t^j]_7) \right] \subset \mathbb{R}^{7K+14} \tag{4}$$

where $g_t^j \in SE(3)$ represents the roto-translation and $[\tau_t^j]_i$ the torsion angles of residue j at frame t. Here, $\phi : SE(3) \to \mathbb{R}^7$ parameterizes an element of $SE(3)$ in terms of a unit quaternion and translation vector, and $\psi : \mathbb{T} \to \mathbb{R}^2$ converts torsion angles to points on the unit circle. We thus obtain a $(7K + 14)$-dimensional array for each residue in every frame. Because the relative roto-translations and torsion angles are both $SE(3)$-invariant, in this manner we can represent a polypeptide molecular trajectory as an $(T \times L)$-array of $SE(3)$-invariant tokens.

To *untokenize* a generated trajectory of tokens to all-atom coordinates $\mathbf{X}_t \in \mathbb{R}^{3N}$, we first convert each predicted quaternion and translation vector to a relative roto-translation and apply it to the key frame(s), obtaining absolute roto-translations. We then read off the torsion angles from the unit circle and assemble the all-atom coordinates as implemented in Jumper et al. (2021), averaging the reconstructions from different key frames if needed.

3.2 Flow Model Architecture

Our base modeling task is to generate a distribution over $\mathbb{R}^{T \times L \times (7K+14)}$ conditioned on roto-translations of one or more key frames $g_{t_1} \dots g_{t_K}$, and (in most settings) amino acid identities A. To do so, we learn a flow-based model via the stochastic interpolant framework described in SiT (Ma et al., 2024) and parameterize a velocity network $v_\theta(\cdot \mid g_{t_1} \dots g_{t_K}, A) : \mathbb{R}^{T \times L \times (7K+14)} \times [0, 1] \to \mathbb{R}^{T \times L \times (7K+14)}$. To condition on the key frames and amino acids, we first provide the sequence embedding to several IPA layers (Jumper et al., 2021) that embed the key frame roto-translations; these conditioning representations (which are $SE(3)$-invariant) are broadcast across the time axis and added to the input embeddings. The main trunk of the network consists of alternating attention blocks across the residue index and across time, with the construction of each block closely resembling DiT (Peebles and Xie, 2023). Sidechain torsions and roto-translation offsets, when available, are directly provided to the model as conditioning tokens. Further details are provided in Appendix A.1.

In the molecular inpainting setting where we also *generate* the amino acid identities, we additionally require a generative framework over these discrete variables. While several formulations of discrete diffusion or flow-matching are available (Hoogeboom et al., 2021; Austin et al., 2021; Campbell et al., 2022, 2024), we select Dirichlet flow matching (Stark et al., 2024) as it is most compatible with the continuous-space, continuous-time stochastic interpolant framework used for the positions. Specifically, we place the amino acid identities on the 20-dimensional probability simplex (one per amino acid), augment the token representations with these variables, and regress against a $T \times L \times (7K + 14 + 20)$-dimensional vector field. Further details are provided in Appendix A.2.

3.3 Conditional Generation

We present the precise specifications of the various conditional generation settings in Table 1. Depending on the task, we choose the key frames to be the first frame g_1 or the first and last frames g_1, g_T. Each conditional generation task is further characterized by providing the ground-truth tokens of known frames or residues as additional inputs to the velocity network. Meanwhile, mask tokens are provided for the unknown frames and residues that the model generates. For example, in the upsampling setting, we provide ground-truth tokens every M frames, while mask tokens are provided for all other frames. We note that in the inpainting setting, the model accesses the roto-translations g of designed residues at the trajectory endpoints via the key frames, constituting a slight departure from the full inpainting setting. However, these residues are not observed for intermediate frames, and their identities are never provided to the model.

Table 1: Conditional generation settings. g: roto-translations, $\boldsymbol{\tau}$: torsions, A: residue identities M: upsampling factor. Superscripts indicate residue index and subscripts indicate frame (time) index. For inpainting, we find that excluding identities and torsions reduces overfitting.

Setting	Key frames	Generate	Conditioned on	Token dim.
Forward simulation	g_1	$g_{1\cdots T}, \boldsymbol{\tau}_{1\cdots T}$	$g_1, \boldsymbol{\tau}_1, A$	21
Interpolation	g_1, g_T	$g_{1\cdots T}, \boldsymbol{\tau}_{1\cdots T}$	$g_{1,T}, \boldsymbol{\tau}_{1,T}, A$	28
Upsampling	g_1	$g_{1\cdots T}, \boldsymbol{\tau}_{1\cdots T}$	$g_{1+\{1,2,\cdots\}M}, \boldsymbol{\tau}_{1+\{1,2,\cdots\}M}, A$	21
Inpainting	g_1, g_T	$g_{1\cdots T}, A$	$g^{\text{known}}_{1\cdots T}$	7 (+20)

4 Experiments

We evaluate MDGEN on its ability to learn from MD simulations of training molecules and then sample trajectories for unseen molecules. We focus on *tetrapeptides* as our main molecule class for evaluation as they provide nontrivial chemical diversity while remaining small enough to tractably simulate to equilibrium (Klein et al., 2024). Sections 4.1–4.3 thoroughly evaluate our model on forward simulations, interpolation / transition path sampling, and trajectory upsampling on test peptides. Section 4.4 provides proof-of-concept and preliminary exploration of additional tasks—namely, inpainting for dynamics-conditioned design, long trajectories with Hyena (Poli et al., 2023), and scaling to simulations of protein monomers. Separate models are trained for each setting.

To obtain tetrapeptide MD trajectories for training and evaluation, we run implicit- and explicit-solvent, all-atom simulations of ≈3000 training, 100 validation, and 100 test tetrapeptides for 100 ns. For proteins, we use explicit-solvent, all-atom simulations from the ATLAS dataset (Vander Meersche et al., 2024), which provides three 100 ns trajectories for each of 1390 structurally diverse proteins. Unless otherwise specified, models are trained with trajectory timesteps of $\Delta t = 10$ ps. Our default baselines consist of *replicate MD simulations* ranging from 10 ps to 100 ns, with additional comparisons in each section as appropriate.

Our experiments make extensive use of Markov State Models (MSMs), a widely used coarse-grained representation of molecular dynamics (Prinz et al., 2011; Noé et al., 2013). We obtain an MSM to represent a system by discretizing its MD trajectory (parameterized with torsion angles) into 10 metastable states and estimating the transition probabilities between them. Appendix B provides further details on constructing MSMs and other experimental settings. Additional results, including structural validations and further comparisons with related methods, can be found in Appendix C.

4.1 Forward Simulation

In the *forward simulation* setting, we train a model to sample 10 ns trajectories conditioned on the first frame. By chaining together successive model rollouts at inference time, we obtain 100 ns trajectories for each peptide to compare with ground-truth simulations. We evaluate if these sampled trajectories (1) match the structural distribution of trajectories from MD, (2) accurately capture the dynamical content of MD, and (3) traverse the state space in less wall-clock time than MD.

Distributional similarity. We report the Jensen-Shannon divergence (JSD) between the ground-truth and emulated trajectories along various collective variables shown in Figure 2 and Table 2. The first

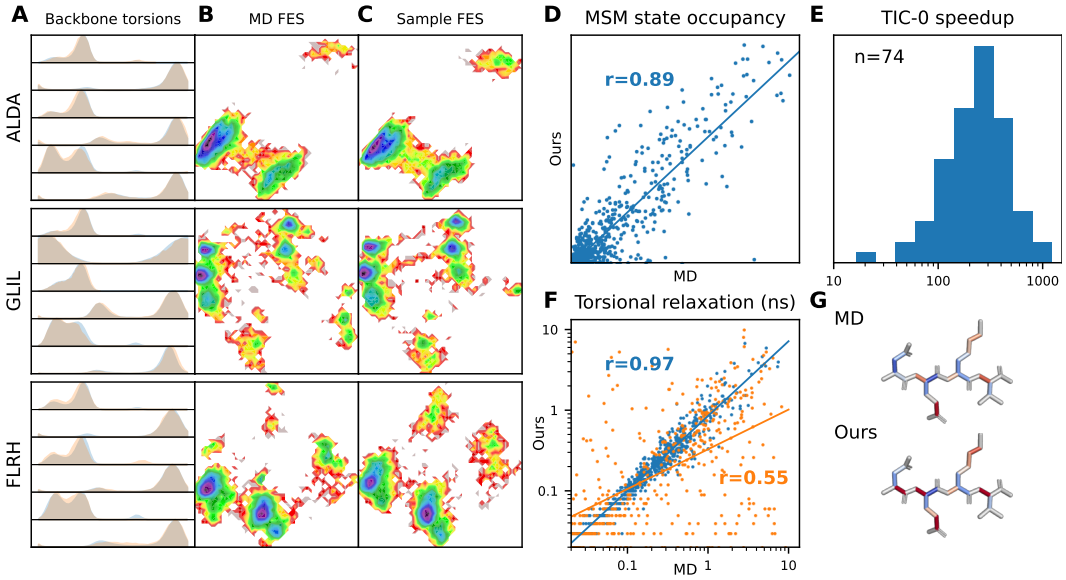

Figure 2: **Forward simulation evaluations on test peptides.** (**A**) Torsion angle distributions for the six backbone torsion angles from MD trajectories (orange) and sampled trajectories (blue). (**B, C**) Free energy surfaces along the top two TICA components computed from backbone and sidechain torsion angles. (**D**) Markov State Model occupancies computed from MD trajectories versus sampled trajectories, pooled across all test peptides ($n = 1000$ states total). (**E**) Wall-clock decorrelation times of the first TICA component under MD versus our model rollouts. (**F**) Relaxation times of torsion angles computed from MD versus sampled trajectories, pooled across all test peptides—508 backbone (blue) and 722 sidechain (orange) torsions in total. (**G**) Torsion angles in the tetrapeptide `AAAA` colored by the decorrelation time computed from MD (top) and from rollout trajectories (bottom).

set of these are the individual torsion angles (backbone and sidechains) in each tetrapeptide. The second set of variables are the top independent components obtained from *time-lagged independent components analysis* (TICA), representing the slowest dynamic modes of the peptide. By each of these collective variables, MDGEN demonstrates excellent distributional similarity to the ground truth, approaching the accuracy of replicate 100-ns simulations. To more stringently assess the ability to locate and populate modes in the joint distribution over state space, we build Markov State Models (MSMs) for each test peptide using the MD trajectory, extract the corresponding metastable states, and compare the ground-truth and emulated distributions over metastable states. Our model captures the relative ranking of states reasonably well and rarely misses important states or places high mass on rare states (Figure 2D).

Table 2: JSD between sampled and ground-truth distributions, with replicate simulations as baselines. 100 ns represents oracle performance.

C.V.	Ours	10 ns	1 ns	100 ps	100 ns
Torsions (bb)	**.130**	.145	.212	.311	.103
Torsions (sc)	**.093**	.111	.261	.403	.055
Torsions (all)	**.109**	.125	.240	.364	.076
TICA-0	**.230**	.323	.432	.477	.201
TICA-0,1 joint	**.316**	.424	.568	.643	.268
MSM states	**.235**	.363	.493	.527	.208
Runtime	60s	1067s	107s	11s	3h

Dynamical content. We compute the dynamical properties of each tetrapeptide in terms of the *decorrelation time* of each torsion angle from the MD simulation and from our sampled trajectory. Intuitively, this assesses if our model can discriminate between slow- and fast-relaxing torsional barriers. The correlation between true and predicted relaxation timescales is plotted in Figure 2F, showing excellent agreement for sidechain torsions and reasonable agreement for backbones. To assess coarser but higher-dimensional dynamical content, we compute the *flux matrix* between all pairs of distinct metastable states using ground-truth and sampled trajectories and find substantial Spearman correlation between their entries (mean $\rho = 0.67 \pm 0.01$; Figure 8). Thus, our simulation rollouts can accurately identify high-flux transitions in the peptide conformational landscape.

Sampling speed. Averaged across test peptides, our model samples 100 ns-equivalent trajectories in ≈60 GPU-seconds, compared to ≈3 GPU-hours for MD. To quantify the speedup more rigorously,

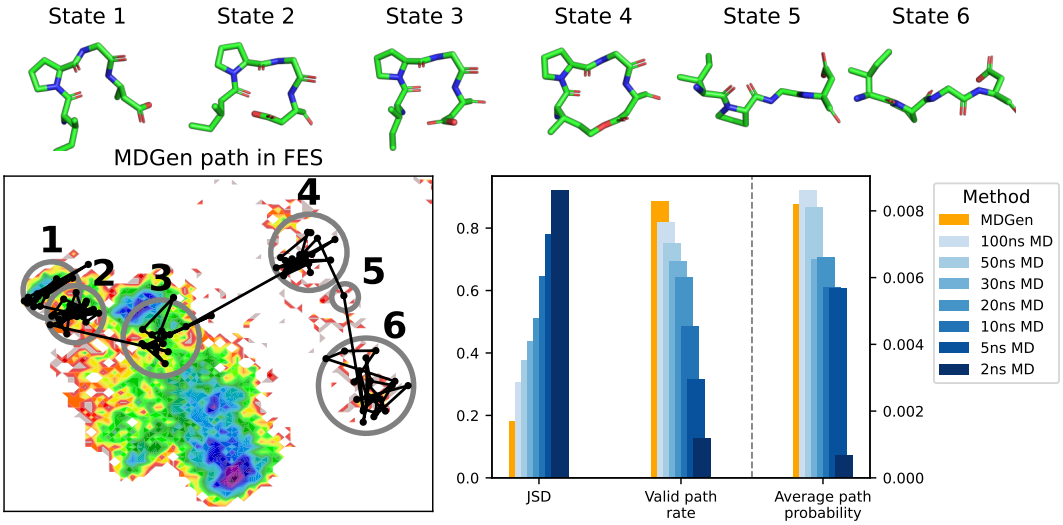

Figure 3: **Transition path sampling results.** (**Top**) Intermediate states of one of the 1-nanosecond interpolated trajectories between two metastable states for the test peptide IPGD. (**Bottom Left**) The corresponding trajectory on the 2D free energy surface of the top two TICA components (more examples in Figure 9). (**Bottom Right**) Statistics averaged over 100 test peptides and 1000 paths for each of them. Shown are JSD, fraction of drawn paths that are valid transition paths, and average path likelihood of our discretized transitions under the reference MSM compared to discrete transitions drawn from the reference MSM or alternative MSMs built from replica simulations of varying lengths.

we compute the decorrelation wall-clock times along the slowest independent component from TICA, capturing how quickly the simulation traverses the highest barriers in state space. These times are plotted in Figure 2E, showing that our model achieves a speedup of 10x–1000x over the MD simulation for 78 out of 100 peptides (the other 22 peptides did not fully decorrelate).

4.2 Interpolation

In the *interpolation* or *transition path sampling* setting, we train a model to sample 1 ns trajectories conditioned on the first and last frames. For evaluation, we identify the two most well-separated states (i.e., with the least flux between them) for each test peptide and sample an ensemble of 1000 transition paths between them. Figure 3 shows an example of such a sampled path, which passes through several intermediate states on the free energy surface to connect the two endpoints.

To evaluate the accuracy of these sampled transitions, we cannot directly compare with MD trajectories since, in most cases, there are zero or very few 1-ns transitions between the two selected states (by design, the transition is a *rare event*). Thus, we instead discretize the trajectory over MSM metastable states and evaluate the *path likelihood* under the transition path distribution from the reference MSM (details in Appendix B.3). We also report the fraction of valid paths (i.e., non-zero probability) and the JSD between the distribution of visited states from our path distribution versus the transition path distribution of the reference MSM. For baselines, we sample transition paths from MSMs constructed from replicate MD simulations of varying lengths and compute the same metrics for these (discrete) path ensembles under the reference MSM.

As shown in Figure 3, our paths have higher likelihoods than those sampled from any replicate MD MSM shorter than 100ns, which is the length of the reference MD simulation itself. Moreover, MDGEN's ensembles have the best JSDs to the distribution of visited states of the reference MD MSM and the highest fraction on valid non-zero probability paths. Hence, our model enables zero-shot sampling of trajectories corresponding to arbitrary rare transitions for unseen peptides.

4.3 Upsampling

Molecular dynamics trajectories are often saved at relatively long time intervals (10s–100s of picoseconds) to reduce disk storage; however, some molecular motions occur at faster timescales and

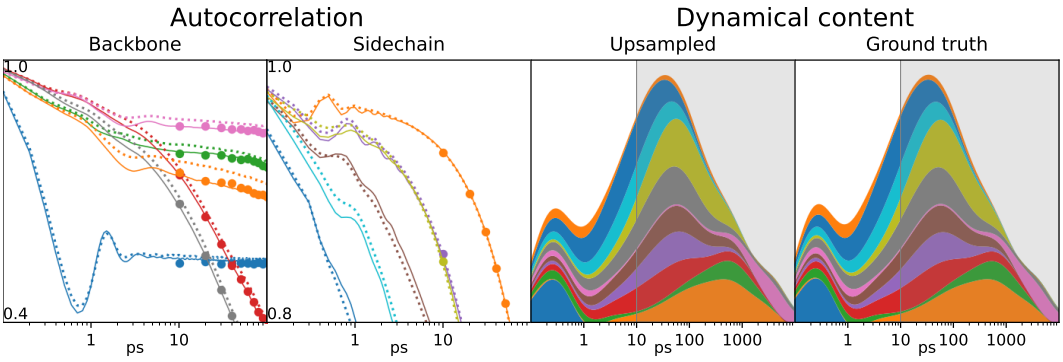

Figure 4: **Recovery of fast dynamics via trajectory upsampling** for peptide GTLM. (*Left*) Autocorrelations of each torsion angle from (—) the original 100 fs-timestep trajectory, (•) the subsampled 10 ns-timestep trajectory, and (⋯) the reconstructed 100 fs-timestep trajectory (all length 100 ns). (*Right*) Dynamical content as a function of timescale from the upsampled vs. ground truth trajectories, stacked for all torsion angles (same color scheme). The subsampled trajectory contains only the shaded region and our model recovers the unshaded region. Further examples in Figure 10.

Table 3: Sequence recovery for the inner two peptides when conditioning on the partial trajectory (MDGEN), the two terminal frames (DynMPNN), or a single frame (S-MPNN).

Method	High Flux	Random Path
MDGen	**52.1%**	**62.0%**
DynMPNN	17.4%	24.5%
S-MPNN	16.3%	13.5%

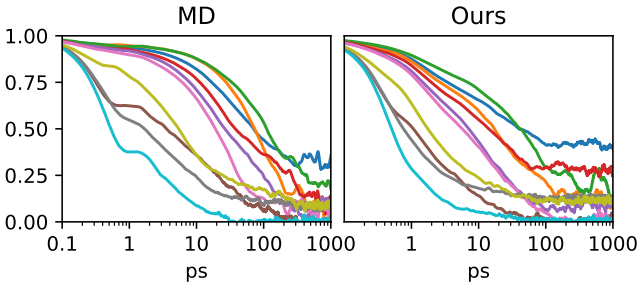

Figure 5: Autocorrelation functions of MDGEN sidechain torsion angles computed from a 10-ns MD trajectory (*left*) versus a single **100k-frame model sample** with Hyena (*right*), capturing dynamics spanning four orders of magnitude.

would be missed by downstream analysis of the saved trajectory. In the *upsampling* setting, we train MDGEN to upsample trajectories saved with timestep 10 ps to a finer timestep of 100 fs, representing a 100x upsampling factor. To evaluate if the upsampled trajectories accurately capture the fastest dynamics, we compute the *autocorrelation function* $\langle \cos(\theta_t - \theta_{t+\Delta t}) \rangle$ of each torsion angle in the test peptides as a function of lag time Δt ranging from 100 fs to 100 ps.

Representative examples of ground truth, subsampled, and reconstructed autocorrelation functions for two test peptides are shown in Figure 4 (further examples in Figure 10). We further compute the *dynamical content* as the negative derivative of the autocorrelation with respect to log-timescale, which captures the extent of dynamic relaxations occurring at that timescale (Shaw et al., 2009). These visualizations highlight the significant dynamical information absent from the subsampled trajectory and which are accurately recovered by our model. In particular, our model distinctly recovers the *oscillations* of certain torsion angles as seen in the non-monotonicity of the autocorrelation function at sub-picosecond timescales; these features are completely missed at the original sampling frequency.

4.4 Additional Tasks

Inpainting Design. We aim to sample trajectories conditioned on the dynamics of the two flanking residues of the tetrapeptide; in particular, the model determines the identities and dynamics of the two inner residues. We focus on *dynamics scaffolding* as one possible higher-level objective of inpainting: given the conformational transition of the observed residues, we hope to design peptides that support flux between the corresponding Markov states. Thus, for each test peptide, we select a 100-ps transition between the two most well-connected Markov states, mask out the inner residue identities and dynamics, and inpaint them with our model. To evaluate the designs, we compute the fraction of

Table 4: Median results on test protein ensembles ($n = 82$); evaluations from Jing et al. (2024). Runtimes are reported per sample structure or frame.

	MDGEN	AlphaFlow	MSA sub.
Pairwise RMSD r ↑	**0.48**	**0.48**	0.22
Global RMSF r ↑	0.50	**0.60**	0.29
Per-target RMSF r ↑	0.71	**0.85**	0.55
Root mean $\mathcal{W}_2$ dist. ↓	2.69	**2.61**	3.62
MD PCA $\mathcal{W}_2$ dist. ↓	1.89	**1.52**	1.88
% PC-sim > 0.5 ↑	10	**44**	21
Weak contacts J ↑	0.51	**0.62**	0.40
Exposed residue J ↑	0.29	**0.41**	0.27
Runtime (s)	**0.2**	70	4

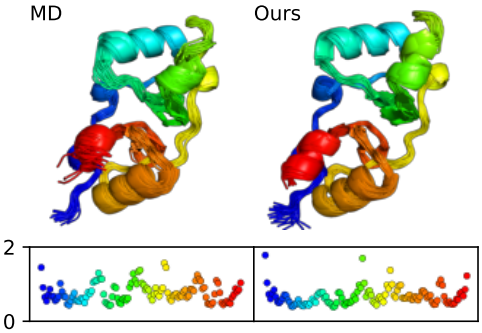

Figure 6: MD vs generated ensembles for `6uof_A`, with Cα RMSFs plotted by residue index (Pearson $r = 0.74$).

generated residue types that are identical to the tetrapeptide in which the target transition is known to occur. We compare MDGEN with a bespoke inverse folding baseline that is provided the two terminal states (i.e., two fully observed MD frames), and thus designs peptides that support the two modes (rather than additionally a partially-observed *transition* between them). We call this baseline DYNMPNN, and it otherwise has the same architecture and settings as MDGEN (more details in Appendix B.3). We find (Table 3) that MDGEN recovers the ground-truth peptide substantially more often than DynMPNN when conditioned on a high-flux path or (as a sanity check) a random path from the reference simulatiom.

Scaling to Long Trajectories. Although Section 4.1 showed that our model can emulate long trajectories, this was limited to rollouts of 1000 frames at a time with coarse 10 ps timesteps, potentially missing faster dynamics or disrupting slower dynamics. Thus, we investigate generating extremely long consistent trajectories that capture timescales spanning several orders of magnitude *within a single model sample*. To do so, we replace the time attention in our baseline SiT architecture with a non-causal Hyena operator (Poli et al., 2023), which has $O(N \log N)$ rather than $O(N^2)$ overhead. We overfit on 100k-frame, 10-ns trajectories of the pentapeptide `MDGEN` and compare the torsional autocorrelation functions computed from a *single* generated trajectory with a *single* ground truth trajectory (Figure 5). Although not yet comparable to the main set of forward simulation experiments due to data availability and architectural expressivity reasons, these results demonstrate proof-of-concept for longer context lengths in future work.

Protein Simulation. To demonstrate the applicability of our method for larger systems such as proteins, we train a model to emulate all-atom simulations of proteins from the ATLAS dataset (Vander Meersche et al., 2024) conditioned on the first frame (i.e., forward simulation). We follow the same splits as Jing et al. (2024). Due to the much larger number of residues, we generate samples with 250 frames and 400 ps timestep, such that a single sample emulates the 100 ns ATLAS reference trajectory. The difficulty of running fully equilibrated trajectories for proteins prevents the construction of Markov state models used in our main evaluations. Instead, we compare statistical properties of forward simulation ensembles following Jing et al. (2024). Our ensembles successfully emulate the ground-truth ensembles at a level of accuracy between AlphaFlow and MSA subsampling while being orders of magnitude faster per generated structure than either (Table 4; Figure 6).

5 Discussion

Limitations. Our experiments have validated the model and architecture for peptide simulations; however, a few limitations provide opportunities for future improvement. Due to the reliance on key frames, the model is not capable of unconditional generation or inpainting of residue roto-translations. The weaker performance on protein monomers relative to peptides suggests that scaling to larger systems will likely require additional data or methodological innovations. Fine-tuning of single structure models for co-generation of the key frames and trajectory tokens, similar to the content-frame decomposition of video diffusion models (Yu et al., 2024), may provide improvement. Since our tokenization scheme is specific to polypeptides, alternative strategies will be needed to model all-atom trajectories of more general systems, such as organic ligands, materials, or explicit

solvent. More ambitious applications (see below) may require the ability to model trajectories not of a predefined set of atoms but over a region of space in which atoms may enter and exit. As such, we anticipate advancements in tokenization and architecture to be a fruitful direction of future work.

Opportunities. Similar to the foundational role of video generative models for understanding the macroscopic world (Yang et al., 2024), MD trajectory generation could serve as a multitask, unifying paradigm for deep learning over the microscopic world. Interpolation can be more broadly framed as *hypothesis generation* for mechanisms of arbitrary molecular phenomena, especially when only partial information about the end states is supplied. Molecular inpainting could be a general technique to design molecular machinery by scaffolding more fine-grained and complex dynamics, for example, redesigning proteins to enhance rare transitions observed only once in a simulation or (with *ab initio* trajectories) *de novo* design of enzymatic mechanisms and motifs. Other types of conditioning not explored in this work may lead to further applications, such as conditioning over textual or experimental descriptors of the trajectory. Future availability of significantly more ground truth MD trajectory data for diverse chemical systems could be a chief enabler of such work. Lastly, considerations unique to molecular trajectories, such as equilibrium vs non-equilibrium processes, Markovianity, and the reversibility of the microscopic world contrasted with the macroscopic world (e.g., the missing arrow of time), could provide ripe areas for theoretical exploration.

Acknowledgments and Disclosure of Funding

We thank Felix Faltings, Jason Yim, Mateo Reveiz, Gabriele Corso, and anonymous NeurIPS reviewers for helpful feedback and discussions.

This work was supported by the National Institute of General Medical Sciences of the National Institutes of Health under award number 1R35GM141861; the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DESC0022158; the National Science Foundation under Grant No. 1918839; the Machine Learning for Pharmaceutical Discovery and Synthesis (MLPDS) consortium; the Abdul Latif Jameel Clinic for Machine Learning in Health; the DTRA Discovery of Medical Countermeasures Against New and Emerging (DOMANE) threats program; and the DARPA Accelerated Molecular Discovery program. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a Department of Energy Office of Science User Facility using NERSC awards ASCR-ERCAP0027302 and ASCRERCAP0027818

References

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2022.

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

Joshua A Anderson, Chris D Lorenz, and Alex Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of computational physics*, 227(10):5342–5359, 2008.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*, 2024.

John D Chodera and Frank Noé. Markov state models of biomolecular conformational dynamics. *Current opinion in structural biology*, 25:135–144, 2014.

Tom Darden, Darrin York, and Lee Pedersen. Particle mesh ewald: An n log (n) method for ewald sums in large systems. *The Journal of chemical physics*, 98(12):10089–10092, 1993.

Diego Del Alamo, Davide Sala, Hassane S Mchaourab, and Jens Meiler. Sampling alternative conformational states of transporters and receptors with alphafold2. *Elife*, 11:e75751, 2022.

Peter Eastman, Jason Swails, John D Chodera, Robert T McGibbon, Yutong Zhao, Kyle A Beauchamp, Lee-Ping Wang, Andrew C Simmonett, Matthew P Harrigan, Chaya D Stern, et al. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS computational biology*, 13(7):e1005659, 2017.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

Xiang Fu, Tian Xie, Nathan J Rebello, Bradley Olsen, and Tommi S Jaakkola. Simulate time-integrated coarse-grained molecular dynamics with multi-scale graph networks. *Transactions on Machine Learning Research*, 2023.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E (n) equivariant normalizing flows. *Advances in Neural Information Processing Systems*, 34:4181–4192, 2021.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021.

Brooke E Husic and Vijay S Pande. Markov state models: From an art to a science. *Journal of the American Chemical Society*, 140(7):2386–2396, 2018.

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

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

Leon Klein, Andrew Foong, Tor Fjelde, Bruno Mlodozeniec, Marc Brockschmidt, Sebastian Nowozin, Frank Noé, and Ryota Tomioka. Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. *Advances in Neural Information Processing Systems*, 36, 2024.

Jonas Köhler, Andreas Krämer, and Frank Noé. Smooth normalizing flows. *Advances in Neural Information Processing Systems*, 34:2796–2809, 2021.

Alessandro Laio and Michele Parrinello. Escaping free-energy minima. *Proceedings of the national academy of sciences*, 99(20):12562–12566, 2002.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024.

J Andrew McCammon, Bruce R Gelin, and Martin Karplus. Dynamics of folded proteins. *nature*, 267(5612):585–590, 1977.

Laurence Midgley, Vincent Stimper, Javier Antorán, Emile Mathieu, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Se (3) equivariant augmented coupling flows. *Advances in Neural Information Processing Systems*, 36, 2024.

Laurence Illing Midgley, Vincent Stimper, Gregor NC Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. *arXiv preprint arXiv:2208.01893*, 2022.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

Frank Noé, Hao Wu, Jan-Hendrik Prinz, and Nuria Plattner. Projected and hidden Markov models for calculating kinetics and metastable states of complex molecules. *The Journal of Chemical Physics*, 139(18):184114, 11 2013.

Vijay S Pande, Kyle Beauchamp, and Gregory R Bowman. Everything you wanted to know about markov state models but were afraid to ask. *Methods*, 52(1):99–105, 2010.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

Guillermo Pérez-Hernández, Fabian Paul, Toni Giorgino, Gianni De Fabritiis, and Frank Noé. Identification of slow molecular order parameters for markov model construction. *The Journal of chemical physics*, 139(1), 2013.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023.

Jan-Hendrik Prinz, Hao Wu, Marco Sarich, Bettina Keller, Martin Senne, Martin Held, John D. Chodera, Christof Schütte, and Frank Noé. Markov models of molecular kinetics: Generation and validation. *The Journal of Chemical Physics*, 134(17):174105, 2011.

Aneesur Rahman. Correlations in the motion of atoms in liquid argon. *Physical review*, 136(2A): A405, 1964.

Susanna Röblitz and Marcus Weber. Fuzzy spectral clustering by pcca+: application to markov state models and data classification. *Advances in Data Analysis and Classification*, Jun 2013.

Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman JC Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of computational physics*, 23(3):327–341, 1977.

Martin K. Scherer, Benjamin Trendelkamp-Schroer, Fabian Paul, Guillermo Pérez-Hernández, Moritz Hoffmann, Nuria Plattner, Christoph Wehmeyer, Jan-Hendrik Prinz, and Frank Noé. PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *Journal of Chemical Theory and Computation*, 11:5525–5542, 2015.

Mathias Schreiner, Ole Winther, and Simon Olsson. Implicit transfer operator learning: Multiple time-resolution models for molecular dynamics. *Advances in Neural Information Processing Systems*, 36, 2024.

David E Shaw, Ron O Dror, John K Salmon, JP Grossman, Kenneth M Mackenzie, Joseph A Bank, Cliff Young, Martin M Deneroff, Brannon Batson, Kevin J Bowers, et al. Millisecond-scale molecular dynamics simulations on anton. In *Proceedings of the conference on high performance computing networking, storage and analysis*, pages 1–11, 2009.

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design. *arXiv preprint arXiv:2402.05841*, 2024.

Yuji Sugita and Yuko Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical physics letters*, 314(1-2):141–151, 1999.

Yann Vander Meersche, Gabriel Cretin, Aria Gheeraert, Jean-Christophe Gelly, and Tatiana Galochkina. Atlas: protein flexibility description from atomistic molecular dynamics simulations. *Nucleic Acids Research*, 52(D1):D384–D392, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Loup Verlet. Computer" experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Physical review*, 159(1):98, 1967.

Christoph Wehmeyer, Martin K Scherer, Tim Hempel, Brooke E Husic, Simon Olsson, and Frank Noé. Introduction to markov state modeling with the pyemma software—v0. 3.

Sherry Yang, Jacob Walker, Jack Parker-Holder, Yilun Du, Jake Bruce, Andre Barreto, Pieter Abbeel, and Dale Schuurmans. Video as the new language for real-world decision making. *arXiv preprint arXiv:2402.17139*, 2024.

Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023.

Sihyun Yu, Weili Nie, De-An Huang, Boyi Li, Jinwoo Shin, and Anima Anandkumar. Efficient video diffusion models via content-frame motion-latent decomposition. *arXiv preprint arXiv:2403.14148*, 2024.

Shuxin Zheng, Jiyan He, Chang Liu, Yu Shi, Ziheng Lu, Weitao Feng, Fusong Ju, Jiaxi Wang, Jianwei Zhu, Yaosen Min, et al. Towards predicting equilibrium distributions for molecular systems with deep learning. *arXiv preprint arXiv:2306.05445*, 2023.

A Method Details

A.1 Flow Model Architecture

Notation. Here, as in the main text, we use the following notation:

- T: number of trajectory frames
- L: number of amino acids
- K: number of key frames, with indicies $t_1 \dots t_K$

In Algorithms 1–3 below, we modify the architectures of DiffusionTransformerAttentionLayer and DiffusionTransformerFinalLayer from DiT (Peebles and Xie, 2023). Elements from these layers are also then incorporated into our custom InvariantPointAttentionLayer.

Algorithm 1: Velocity network

Input: noisy tokens $\boldsymbol{\chi} \in \mathbb{R}^{T\times L\times(7K+14)}$, conditioning tokens $\boldsymbol{\chi}_{\text{cond}} \in \mathbb{R}^{T\times L\times(7K+14)}$,
key frame roto-translations $g_{t_1} \dots g_{t_K} \in \left(SE(3)^L\right)^K$, flow matching time t,
amino acid identities $A \in \{1, \dots 20\}^L$, conditioning mask $\mathbf{m} \in \{0,1\}^{T\times L\times(7K+14)}$

Output: flow velocity $v \in \mathbb{R}^{T\times L\times(7K+14)}$

$t \leftarrow \text{Embed}(t)$;
for $k \leftarrow 1$ **to** K **do**
  $\mathbf{x}_k = \text{Embed}(A) + \sum_{k'} \text{Linear}(g_{t_k}^{-1} g_{t_{k'}})$;
  for $l \leftarrow 1$ **to** num_ipa_layers **do**
    $\mathbf{x}_k = \text{InvariantPointAttentionLayer}(\mathbf{x}, g_k, t)$
$\mathbf{x} = \sum_k \mathbf{x}_k + \text{Linear}(\boldsymbol{\chi}) + \text{Linear}(\boldsymbol{\chi}_{\text{cond}} \odot \mathbf{m}) + \text{Embed}(\mathbf{m})$;
for $l \leftarrow 1$ **to** $num\_transformer\_layers$ **do**
  $\mathbf{x} = \text{DiffusionTransformerAttentionLayer}(\mathbf{x}, t)$
return $\text{DiffusionTransformerFinalLayer}(\mathbf{x}, t)$

Algorithm 2: DiffusionTransformerAttentionLayer

Input: $\mathbf{x} \in \mathbb{R}^{T\times L\times C}$, time conditioning t

$(\alpha, \beta, \gamma)_{t,\ell,f} = \text{Linear}(t)$;
$\mathbf{x} \mathrel{+}= g_\ell \odot \text{AttentionWithRoPE}(\gamma_\ell \odot \text{LayerNorm}(\mathbf{x}) + \beta_\ell, \text{dim} = 1)$;
$\mathbf{x} \mathrel{+}= g_t \odot \text{AttentionWithRoPE}(\gamma_t \odot \text{LayerNorm}(\mathbf{x}) + \beta_t, \text{dim} = 0)$;
$\mathbf{x} \mathrel{+}= g_m \odot \text{MLP}(\gamma_m \odot \text{LayerNorm}(\mathbf{x}) + \beta_m)$;
return $\mathbf{x}$

Algorithm 3: InvariantPointAttentionLayer

Input: $\mathbf{x} \in \mathbb{R}^{L\times C}$, time conditioning t, roto-translations $g \in SE(3)^L$

$(\alpha, \beta, \gamma)_{\ell,f} = \text{Linear}(t)$;
$\mathbf{x} \mathrel{+}= \text{InvariantPointAttention}(\text{LayerNorm}(\mathbf{x}), g)$;
$\mathbf{x} \mathrel{+}= g_\ell \odot \text{AttentionWithRoPE}(\gamma_\ell \odot \text{LayerNorm}(\mathbf{x}) + \beta_\ell)$;
$\mathbf{x} \mathrel{+}= g_m \odot \text{MLP}(\gamma_m \odot \text{LayerNorm}(\mathbf{x}) + \beta_m)$;
return $\mathbf{x}$

A.2 Integrating Dirichlet Flow Matching

To additionally generate amino acid identities along with the trajectory dynamics, we integrate our SiT flow matching framework with Dirichlet flow matching (Stark et al., 2024). Specifically, we now parameterize a velocity network $v_\theta : \left(\mathbb{R}^{7K} \oplus \mathbb{R}^{20}\right)^{T\times L} \times [0,1] \to \left(\mathbb{R}^{7K} \oplus \mathbb{R}^{20}\right)^{T\times L}$. No

architecture modifications are necessary other than augmenting the tokens with one-hot tokens of residue identity, broadcasted across time. At training time, we sample from the Dirichlet probability path (rather than the Gaussian path) for those token elements. However, the parameterization is subtle as Dirichlet FM trains with cross-entropy loss, contrary to the standard flow-matching MSE loss. Thus, during *training time* we minimize the loss

$$\mathcal{L} = \mathbb{E}\left[\|v_\theta\texttt{[...,:-20]} - u_t(\boldsymbol{\chi}_t \mid \boldsymbol{\chi}_1)\|^2 + \text{CrossEntropy}(\text{Softmax}(v_\theta\texttt{[...,-20:]}), A)\right] \quad (5)$$

That is, we interpret the last 20 outputs in the channel dimension as logits over the 20 residue types. At *inference time*, on the other hand, we convert these logits to the Dirichlet FM flow field:

$$v'_\theta = \text{Concat}\left(v_\theta\texttt{[...,:-20]}, \sum_i \text{Softmax}(v_\theta\texttt{[...,-20:]})_i \cdot u_{\text{DFM}}(\cdot \mid x_1 = i)\right) \quad (6)$$

where u_{DFM} is the appropriate Dirichlet vector field from Stark et al. (2024).

A.3 Conditional Generation

We control the conditional generation settings by simply setting appropriate entries of the conditioning mask $\mathbf{m}$ in Algorithm 1 to 1 or 0. Specifically,

- For the forward simulation setting, $\mathbf{m}[t, \ell, c] = \begin{cases} 1 & t = 1 \\ 0 & t \neq 1 \end{cases}$
- For the inpainting setting, $\mathbf{m}[t, \ell, c] = \begin{cases} 1 & t \in \{1, T\} \\ 0 & t \notin \{1, T\} \end{cases}$
- For the upsampling setting, $\mathbf{m}[t, \ell, c] = \begin{cases} 1 & t \,\%\, M = 1 \\ 0 & t \,\%\, M \neq 1 \end{cases}$ where M is the upsampling factor.
- For the inpainting setting, $\mathbf{m}[t, \ell, c] = \begin{cases} 1 & \ell \in \mathcal{S}_{\text{known}} \\ 0 & \ell \notin \mathcal{S}_{\text{known}} \end{cases}$ where $\mathcal{S}_{\text{known}}$ is the set of residues in the known part of the trajectory.

We use 1 indexing to be consistent with the main text. In practice, in the inpainting setting we also mask out all torsion angles and withhold the amino acid identities for all residues. Further, we do not train the model to generate the torsions as all, such that the tokenization yields $\boldsymbol{\chi} \in \mathbb{R}^{T \times L \times 7K}$. These interventions were observed to be necessary to prevent overfitting.

B Experimental Details

B.1 Markov State Models

A Markov State Model (MSM) is a representation of a system's dynamics discretized into r states $s \in \{1 \dots r\}$ and a discrete timesteps separated by *time lag* τ such that the dynamics are approximately Markovian (Husic and Pande, 2018; Chodera and Noé, 2014; Pande et al., 2010). An MSM is parameterized with a vector $\boldsymbol{\pi}$ that assigns each state a stationary probability and a matrix T containing the probabilities for transitioning from state s_t to s_{t+1} after one timestep, i.e., $T_{i,j} = p(s_{t+1} = j \mid s_t = i)$.

To build a Markov state model, we use PyEMMA (Scherer et al., 2015; Wehmeyer et al.) and its accompanying tutorials. Briefly, we first featurize molecular trajectories with all torsion angles as points on the unit circle, obtaining a $2m$-dimensional invariant trajectory where m is the number of torsion angles. We run TICA on these trajectories with kinetic scaling and then run k-means clustering with $k = 100$ over the first few (5–10 chosen by PyEMMA) TICA coordinates. We then estimate an MSM over these 100 states and use PCCA+ spectral clustering (Röblitz and Weber, 2013) to further group these into 10 metastable states. Our final MSM is built from the discrete trajectory over these 10 metastable states. In all cases we use timelag $\tau = 100$ ps.

Unconditionally sampling an MSM. To unconditionally sample a trajectory of length N from an MSM, we first sample the start state from the stationary distribution, i.e., $s_1 \sim \boldsymbol{\pi}$. We then iteratively sample each subsequent state as $s_{t+1} \sim T_{s_t,:}$.

Sampling an MSM conditioned on a start state. To sample a trajectory of length N conditioned on a starting state s_1, we iteratively sample each subsequent state as $s_{t+1} \sim T_{s_t,:}$.

Sampling an MSM conditioned on a start and end state. For our transition path sampling evaluations in Section 4.2, we employ replica transition paths sampled from an MSM by conditioning on a start state s_1 and end state s_N. To do so, we iteratively sample each state between the conditioning states by utilizing the probability

$$p(s_{t+1} = j \mid s_t = i, s_N = k) = \frac{p(s_N = k \mid s_{t+1} = j, s_t = i)p(s_{t+1} = j \mid s_t = i)}{p(s_N = k \mid s_t = i)}. \tag{7}$$

Firstly, the term $p(s_{t+1} = j \mid s_t = i)$ is available in out transition matrix as $T_{i,j}$. Secondly, we obtain $p(s_N = k \mid s_t = i)$ as an entry of the $(N-t)th$ matrix exponential of the transition matrix. Specifically $p(s_N = k \mid s_t = i) = T_{i,k}^{(N-t)}$ where the superscript denotes a matrix exponential. Lastly, we obtain the term $p(s_N = k \mid s_{t+1} = j, s_t = i)$ by realizing that under the Markov assumption $p(s_N = k \mid s_{t+1} = j, s_t = i) = p(s_N = k \mid s_{t+1} = j)$. Further, $p(s_N = k \mid s_{t+1} = j) = T_{j,k}^{(N-t)-1}$.

Replacing the terms in Equation 7 results in

$$p(s_{t+1} = j \mid s_t = i, s_N = k) = \frac{T_{j,k}^{(N-t-1)} T_{i,j}}{T_{i,k}^{(N-t)}}. \tag{8}$$

Thus, we sample states $s_2 \dots s_{N-1}$ iteratively as

$$s_{t+1} \sim \frac{T_{:,s_N}^{(N-t-1)} T_{s_t,:}}{T_{s_t,s_N}^{(N-t)}}. \tag{9}$$

B.2 Tetrapeptide Molecular Dynamics

We run all-atom molecular dynamics simulations in OpenMM (Eastman et al., 2017) using the `amber14` force field parameters with `gbn2` implicit solvent or `tip3pfb` water model. Initial structures are generated with PyMOL, prepared with `pdbfixer`, and protonated at neutral pH. For explicit solvent, we prepare a solvent box with 10 Å padding and neutralize the system with sodium or chloride ions. All simulations are integrated with Langevin thermostat at 350K with hydrogen bond constraints, timestep 2 fs, and friction coefficient 0.3 ps^{-1} (explicit) or 0.1 ps^{-1} (implicit). For explicit solvent, nonbonded interactions are cut off at 10 Å with long-range particle-mesh Ewald. We first minimize the energy with L-BFGS and then equilibrate the system in the NVT ensemble for 20 ps. We then run 100 ns of production simulation in the NVT ensemble (implicit) or NPT ensemble with Monte Carlo barostat at 1 bar (explicit). We write heavy atom positions every 100 fs.

For explicit-solvent settings (forward simulation, interpolation, inpainting), we run simulations for 3109 training, 100 validation, and 100 test peptides. For implicit-solvent settings (upsampling), we run simulations for 2646 training, 100 validation, and 100 test peptides. All peptides are randomly chosen and split. Additionally, 5195 training and 100 validation implicit solvent simulations are run for the pentapeptide `MDGEN`.

B.3 Evaluation Details

Trajectory Featurization We featurize trajectories by selecting the sine and cosine of all torsion angles as the collective variables. Specifically, we featurize ψ, ϕ backbone angles and all χ sidechain torsion angles for each peptide. We then reduce dimensionality with Time-lagged Independent Components Analysis (TICA) (Pérez-Hernández et al., 2013) in PyEMMA (Scherer et al., 2015).

Jensen-Shannon Divergence We compute the JSD as implemented in `scipy`, i.e.,

$$\sqrt{\frac{D(p \mid m) + D(q \mid m)}{2}} \tag{10}$$

where $m = (p+q)/2$. For the 1-dimenional JSD over torsion angles, we discretize the range $[-\pi, \pi]$ into 100 bins. For the 1-dimensional JSD over TIC-0, we discretize the range spanning the maximum and minimum values into 100 bins. For the 2-dimensional JSD over TIC-0,1 we discretize the space into 50×50 bins.

Autocorrelation The autocorrelation of torsion angle θ at time lag Δt is defined as $\langle \cos(\theta_t - \theta_{t+\Delta t}) \rangle$, corresponding to the inner product of $\theta_t, \theta_{t+\Delta t}$ on the unit circle. To compute the *decorrelation time* of a torsion angle, we subtract the baseline inner product $\langle \cos\theta \rangle^2 + \langle \sin\theta \rangle^2$, this is analogous to removing the mean of a real-valued time series before computing the autocorrelation. The decorrelation time is then defined as the time required for the autocorrelation to fall below $1/e$ of its initial value (which is always unity), with the subtracted baseline computed from the reference trajectory. In a small number of cases (21 torsions), the MDGEN trajectory did not decorrelate within 1000 frames (10 ns), and we exclude the angle from Figure 2F.

To compute the decorrelation time for TIC-0, we now define the autocorrelation as

$$\mathbb{E}[(y_t - \mu)(y_{t+\Delta t} - \mu)]/\sigma^2 \tag{11}$$

where μ, σ are computed from the *reference* trajectory. Hence, when computed for a sampled MDGEN trajectory, the autocorrelation may not start at unity and may not decay to zero. We report a *decorrelation time* if starts above and falls below 0.5 within 1000 frames (10 ns), which happens in 74 out of 100 cases as shown in Figure 2E.

Interpolation In our interpolation or transition path sampling experiments, we sample 1000 trajectories of length 1ns for each of our 100 test tetrapeptides. We first select a start state s_1 and an end state s_N that exhibits non-trivial transitions. To do so, we consider a reference MD simulation of 100 ns for the tetrapeptide and obtain an MSM as described in Appendix B.1. From the MSM's transition matrix T and stationary distribution $\boldsymbol{\pi}$, we compute the flux matrix $F = T \odot Pi$ where Pi is the square matrix with $\boldsymbol{\pi}$ in each column. The chosen start and end state is the row and column index of the smallest non-zero entry in F.

With the start state s_1 and end state s_N selected, we sample 1000 start frames $\mathbf{x}_1$ and end frames $\mathbf{x}_N$ from the states. The 1000 start frames are sampled from all frames in the reference MD simulation that belong to state s_1. Analogously, the end frames are sampled from all frames belonging to state s_N. Using the 1000 pairs of start and end frames, we condition MDGEN on them and generate trajectories of 100 frames (1 ns). For evaluation, we discretize these trajectories under the 10-state clustering determined by the MSM of the reference MD simulation as described in Appendix B.1. Note that with the MSM lag time of 100 ps, these discrete trajectories are of length 10.

MD baselines. To sample transition paths of 1 ns between our selected start and end states, we employ MSMs built from replica MD simulations of varying lengths. For instance, for a replica MD simulation of 100ns, we first discretize its trajectory with the cluster assignments of the reference MD simulation (the same cluster assignments as we use to discretize the MDGEN ensemble and that we use for evaluation). Next, we estimate an MSM from the discretized trajectory. We then proceed to sample 1000 transition paths from the MSM as described in B.1 where the path is conditioned on an end and start state. In the event that the replica MSM has zero transition probability for transitioning out of the start state or zero probability for transitioning into the end state (this occurs if the replica MD simulation never visited the start or end state), we treat all 1000 paths of the replica MD as having zero probability for our evaluation metrics which are further detailed in the following.

Computing TPS metrics. As described above, we obtain ensembles of 1000 discretized 1ns paths of 100 frames for both MDGEN and the replica MD simulations. For these, in Figure 3, we show a JSD, the rate of valid paths, and the average path probability. These metrics are computed with respect to the MSM of the reference MD simulation of length 100 ns.

- To compute the JSD, we draw 1000 discrete transition paths from the reference MD simulation and compute the probability of visiting each state from the frequency with which each state is visited. We do the same for the transition path ensemble of MDGEN (or the baseline) and compute the JSD between the categorical distributions as described above.
- The average path probability for an ensemble is the average of its paths' probabilities for transitioning from the start to the end state under the reference MSM. This probability can be computed as described in Appendix B.1.

- The valid path rate is the fraction of paths that have a non-zero probability.

Inpainting In our inpainting experiments, we set out to design tetrapeptides that transition between two states. Considering the residue indices $1, 2, 3$ and 4, we call the residues $1, 4$ the flanking residues which we condition on and $2, 3$ the inner residues which we aim to design. Specifically, we condition MDGEN on the trajectory of the flanking residues' backbone coordinates and generate the residue identities of the inner two residues. To carry out this design for a single tetrapeptide, we draw 1000 samples from MDGEN to estimate the mode of its joint distribution over the inner two residues.

The conditioning information (the trajectories of the outer two residues' backbone coordinates) is different for the two evaluation settings of designing transitions with *high flux* or for designing arbitrary transitions. However, for both of them, the start and end frames are provided as conditioning information via the key frames. In the high flux setting, the conditioning information is obtained by sampling 1000 paths from the reference MD simulation of length 10 ps with 100 frames that start and end in the desired states. These states are determined as those with the maximum flux between them (see the paths about interpolation above for a description of flux). When designing residues that give rise to arbitrary random paths, the trajectories are randomly sampled from the reference simulation.

After sampling 1000 pairs of residues for the inner two residues, we select the most frequently occurring pair as the final design. For this design, we report the sequence recovery (the fraction of residues that match the original sequence of the MD simulations from which the conditioning information was sampled).

Inpainting Baselines. We aim to assess the benefit that is obtained by the trajectory-based inference of MDGEN over a baseline that only takes the start frame or the start and end frame as input for designing residues that transition between two states. Thus, we construct DYNMPNN and S-MPNN. These baselines use the same architecture as MDGEN in the inpainting setting, but DYNMPNN only obtains the start and end frames as key frames and via their roto-translation offsets for the first and last frames. S-MPNN is the analog with only the first frame.

Notably, in the inpainting setting, **MDGen** and the baselines do not treat torsion angles, and all torsion angle entries of the SE(3)-invariant tokens are set to 0. Furthermore, the model does not take the amino acids of the flanking residues as input. We make this choice of withholding all information about amino acid identities since otherwise, the models overfit on the arbitrary identities of the flanking residues and do not generalize to the test set.

Protein Simulations For training and evaluation on proteins, we use trajectories from the ATLAS dataset (Vander Meersche et al., 2024), which includes 3 replicates of 100 ns explicit-solvent, all-atom simulations for each of 1390 non-membrane protein monomers. The proteins are chosen from the PDB as representatives of all available ECOD domains and are thus structurally non-redundant. We split the dataset into 1265 training, 39 validation, and 82 test proteins by PDB release date following Jing et al. (2024). At training time, we randomly select a protein, select one of the three replicates, subsample every 40 frames, obtaining a training target with 250 frames. We train with random crops of up to 256 residues, but draw samples for the full protein at inference time. To compute statistical similarity of the MDGEN ensembles with the ground truth MD ensembles, we compare the 250 frames with 30k pooled frames from all three trajectories. Baseline metrics and runtimes for AlphaFlow and MSA subsampling are taken directly from Jing et al. (2024). Analysis and visualization code for Table 4 and Figure 6 are provided courtesy of Jing et al. (2024).

Runtime MD runtimes in Table 2 are tabulated on a NVIDIA T4 GPU. All MDGEN experiments are carried out on NVIDIA A6000 GPUs. AlphaFlow and MSA subsampling runtimes in Table 4 are tabulated on NVIDIA A100 GPUs by Jing et al. (2024).

C Additional Results

C.1 Forward Simulation

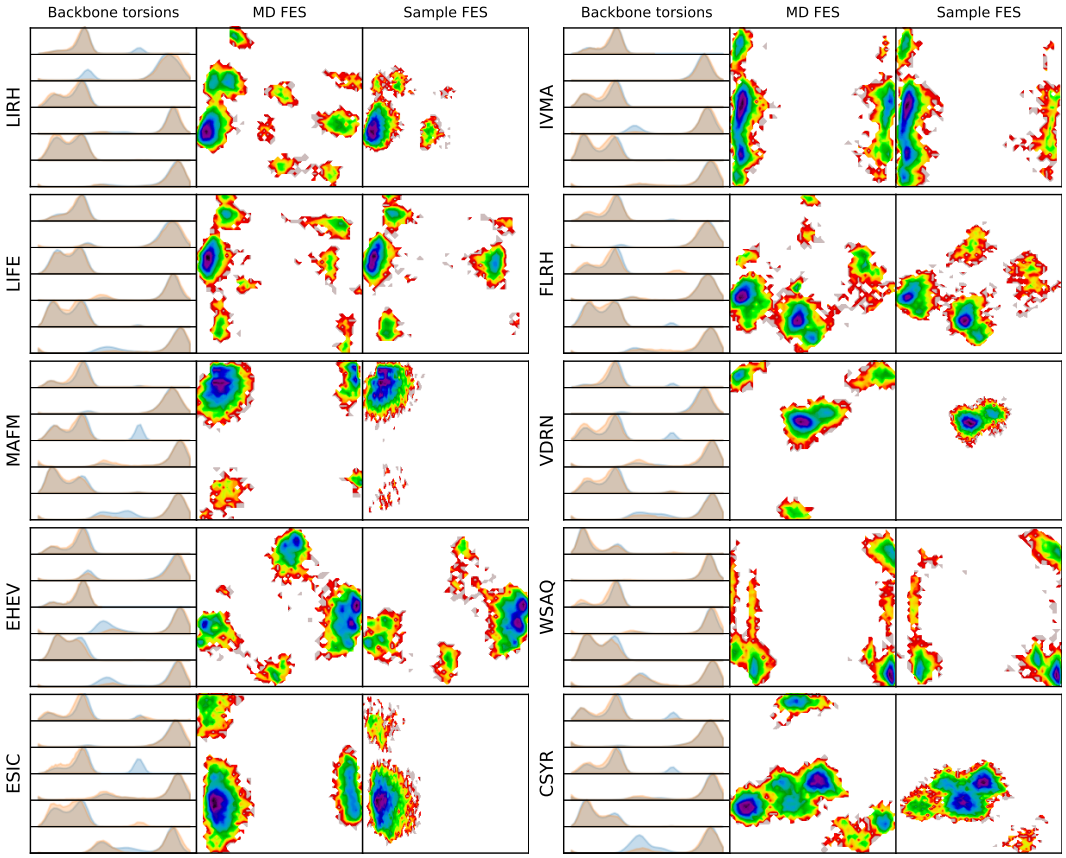

Figure 7: Additional backbone torsion angle distributions (orange from MD, blue from samples) and free energy surfaces along the top two TICA components for 10 randomly chosen test peptides.

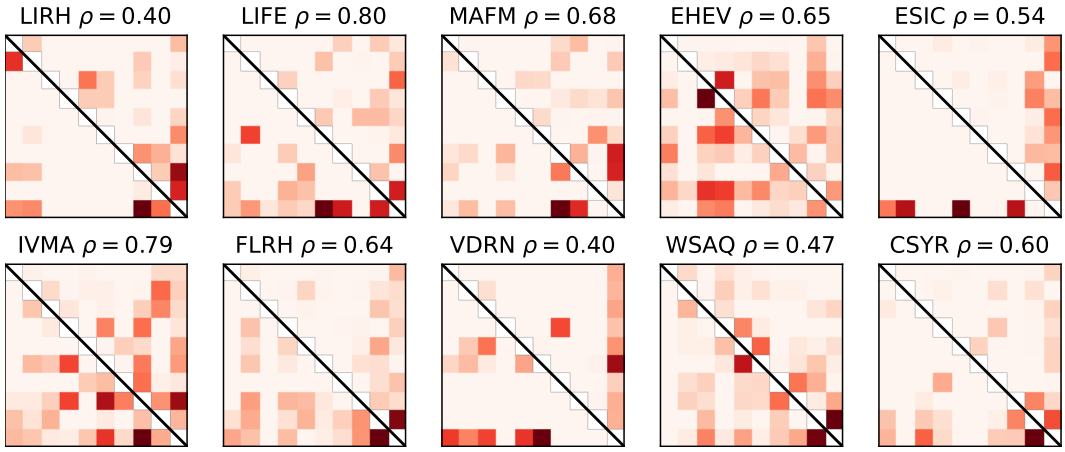

Figure 8: Flux matrices between MSM metastable states computed from reference MD trajectories (upper right) and MDGEN trajectories (bottom left) for 10 random test peptides (the matrices are symmetric). Cells are colored by the square root of the flux, with darker indicating high flux. The Spearman correlation between the entries is shown.

C.2 Interpolation

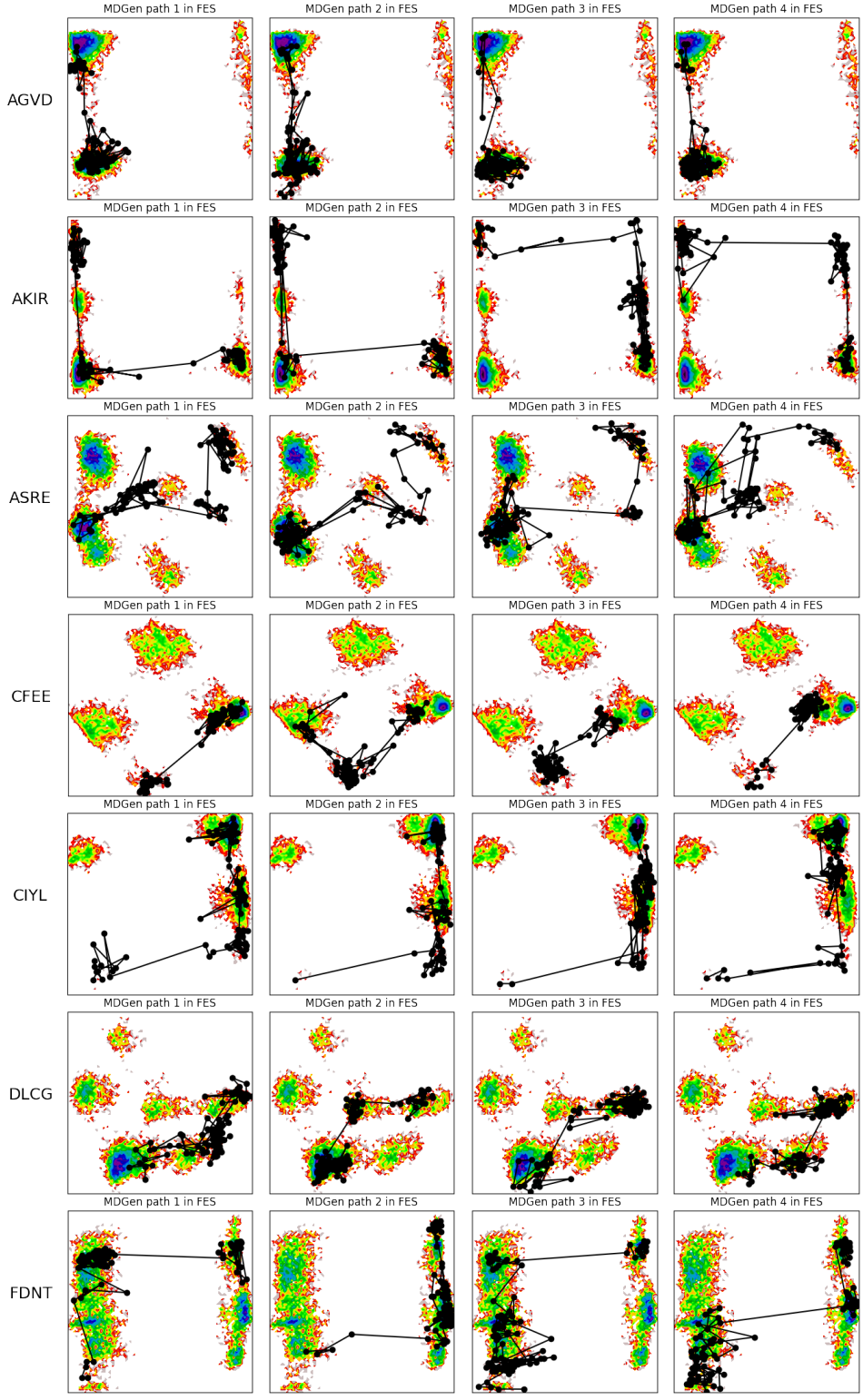

Figure 9: Four of 1000 transition paths of MDGEN for several tetrapeptides in the test set.

C.3 Upsampling

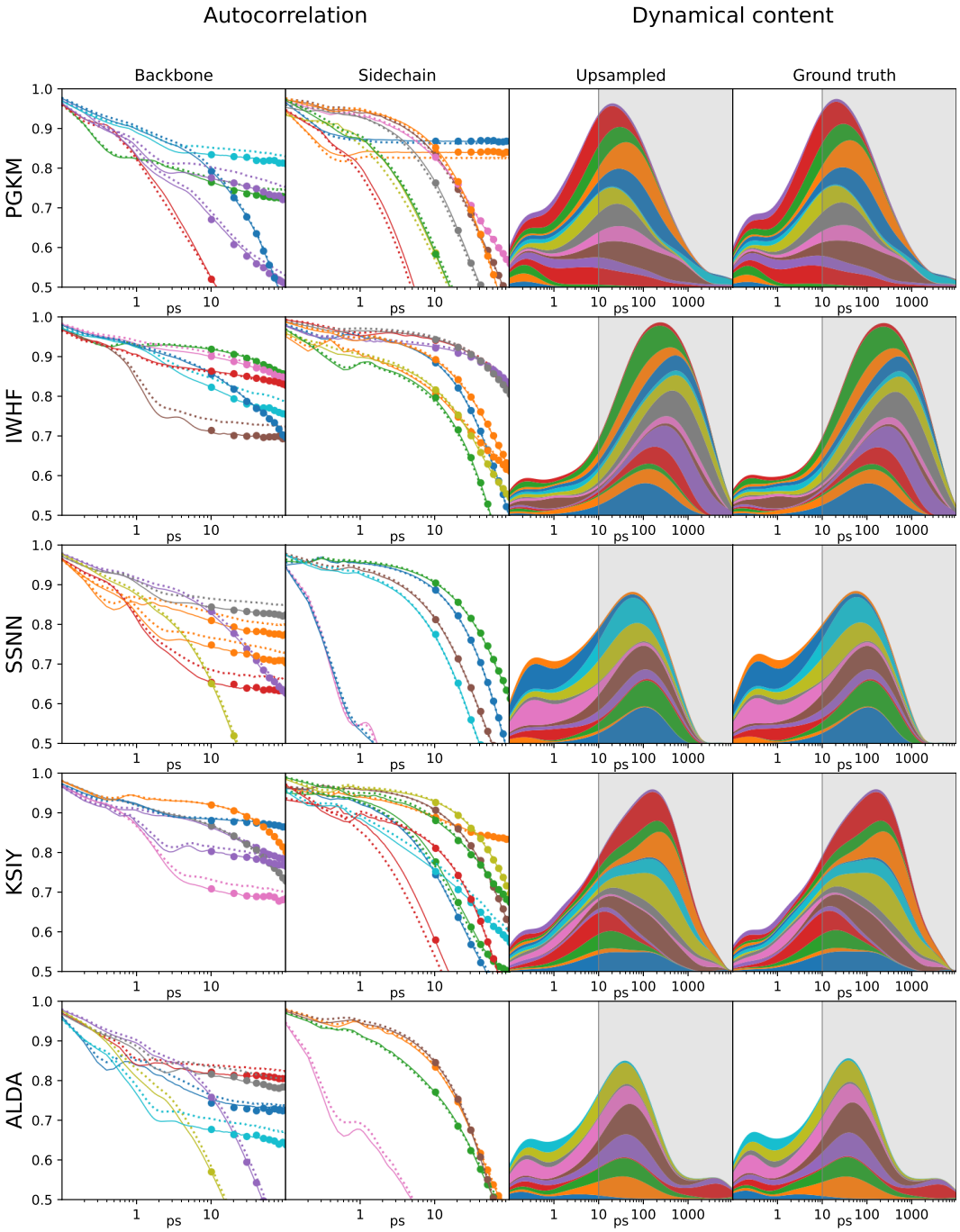

Figure 10: **Recovery of fast dynamics via trajectory upsampling** for random test peptides. (*Left*) Autocorrelations of each torsion angle from (—) the original 100 fs-timestep trajectory, (•) the subsampled 10 ns-timestep trajectory, and (⋯) the reconstructed 100 fs-timestep trajectory (all length 100 ns). (*Right*) Dynamical content as a function of timescale from the upsampled vs. ground truth trajectories, stacked for all torsion angles (same color scheme). The subsampled trajectory contains only the shaded region and our model recovers the unshaded region.

C.4 Inpainting

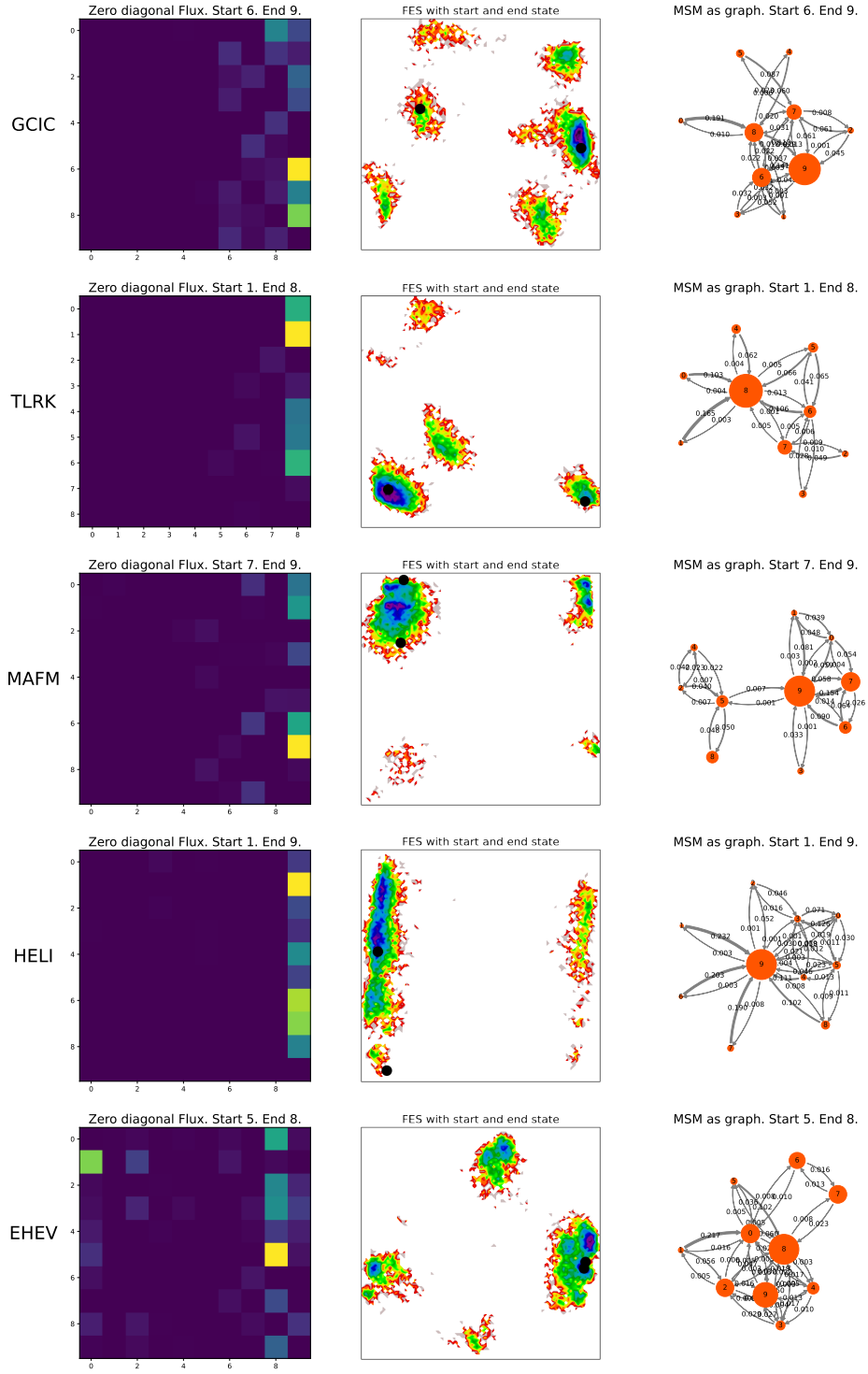

Figure 11: For six tetrapeptides, we show the states that we chose in our design experiments when designing transitions between the highest flux states. Column 1 shows the flux matrix with zeros on the diagonal. Column 2, the free energy surface of a 100 ns simulation and the selected start and end states based on the highest flux in the flux matrix. Column 3, the MSM that was built from the MD simulation.

C.5 Structural Validation

In addition to distributional similarity and dynamical content, we also assess the frequency of clashes or high-energy structures in MDGEN forward simulation rollouts. Specifically, we compute the distributions of:

- The closest distance between any pair of nonbonded atoms
- Nonbonded energy (Coulomb + Lennard-Jones)
- Torsional energy
- Heavy atom bond lengths
- Radius of gyration

These distributions are shown and compared to the ground truth in Figure 12. We find that the vast majority of MDGEN structures are of high quality (i.e., clashes are rare) and adhere closely to the ground truth distributions.

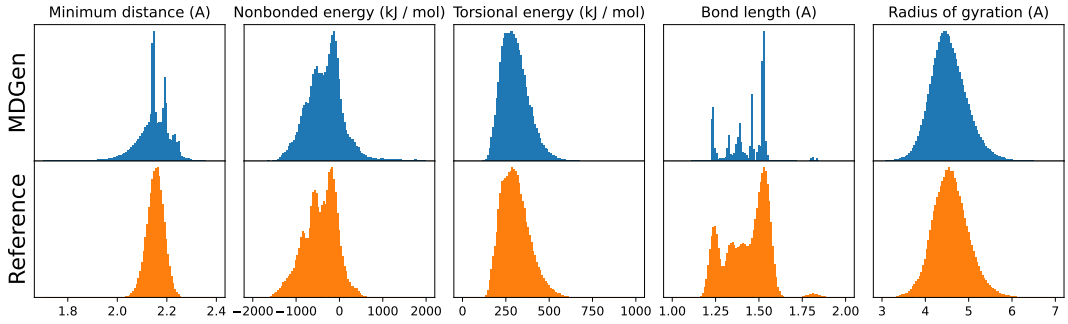

Figure 12: Histograms of various structural validation metrics between MDGEN forward simulation and reference trajectories, pooled across all test tetrapeptides.

C.6 Additional Comparisons

In this section, we compare MDGEN to Timewarp (Klein et al., 2024) and ITO (Schreiner et al., 2024), generative models for autoregressively rolling out surrogate MD trajectories. Note that these comparisons are limited to the forward simulation task as Timewarp and ITO are not capable of solving the other tasks.

- For **Timewarp** (Klein et al., 2024), we use the 4AA model with weights from the authors and sample 100 ns trajectories by running 2000 inference steps with timestep 50 ps. We do not use MH acceptance steps as the authors found exploration of the energy landscape to be much more effective without them.
- For **ITO** (Schreiner et al., 2024), transferable models across tetrapeptides are not available. We therefore re-train ITO on our tetrapeptide dataset with timesteps of 500 ps. We then run 200 inference steps to sample 100 ns trajectories.

For both methods, we observe that trajectories are unstable without further intervention. To bolster the baselines, we run OpenMM (Eastman et al., 2017) relaxation steps between each timestep to proceed with further analysis. We note that the Timewarp authors, in lieu of relaxation, rejected steps with an energy increase of 300 kJ / mol (Klein et al., 2024); however, we found that this strategy would reject the majority of proposed steps on generic tetrapeptides. In Figure 13, we visualize the free energy surfaces and torsion angle distributions for several peptides from Timewarp and ITO compared with MDGen. Table 5 shows the Jensen-Shannon divergences from the forward simulation rollouts across all test peptides (c.f. Table 2). Qualitatively and quantitatively, our model obtains better consistency with the ground-truth free energy surfaces.

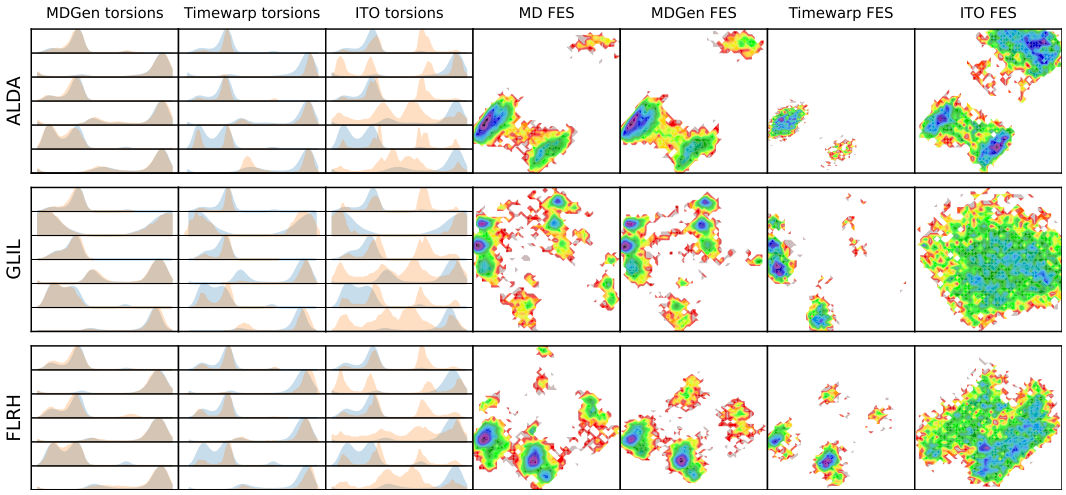

Figure 13: Comparison of MDGEN, Timewarp, and ITO in terms of forward simulation evaluations on test peptides. (*Left*) Torsion angle distributions for the six backbone torsion angles from MD trajectories (orange) and sampled trajectories (blue). (*Right*) Free energy surfaces along the top two TICA components computed from backbone and sidechain torsion angles.

Table 5: Comparison of MDGEN, Timewarp, and ITO in terms of the JSD between sampled and ground-truth distributions along various collective variables in the forward simulation setting.

C.V.	MDGEN	Timewarp	ITO
Torsions (bb)	**.130**	.325	.564
Torsions (sc)	**.093**	.427	.462
Torsions (all)	**.109**	.383	.505
TICA-0	**.230**	.265	.538
TICA-0,1 joint	**.316**	.419	.756
MSM states	.235	**.222**	.414
Runtime	60s	599s	2083s

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We outline all contributions of the paper, emphasizing the ones with robust support and qualifying the ones for which the support is preliminary.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 5 for a discussion of limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide details sufficient to reproduce our model and experiments in Appendix A and Appendix B. In particular, the model architecture is detailed in Algorithm 1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have released our code and data under MIT license.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and test details are provided throughout the main text, figure and table captions, and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our results are reported across a large test set for the task ($n = 100$) and focus more on the new qualitative capabilities that we introduce rather than quantitative improvements on benchmarks. Thus, we did not feel it was necessary to report error bars but would be happy to provide them upon request.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Our models are small by modern standards and we did not feel it was necessary to explicitly highlight the compute time required for experiments. We provide some information on computational resources in Appendix B. Further, the runtimes reported in Tables 2 and 4 provide an idea of the efficiency of our models.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: All authors have read and adhered to, in every respect, the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

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

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not release data or models that have a high risk for misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide the original citations of major open-source software packages and datasets used in our work. We did not feel it was necessary to explicitly mention their open-source licenses as all are publicly available.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Model code and weights for reproducibility are provided via a public repository with instructions for replicating training and evaluation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.