SCALE-RECURSIVE GENERATION OF LONG-HORIZON MOLECULAR DYNAMICS TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Molecular dynamics (MD) simulations are a standard atomistic tool for studying protein folding, ligand binding, and other biomolecular processes, but their reliance on femtosecond-scale integration steps makes the simulation of millisecond-scale events prohibitively expensive. Generative surrogates offer a promising acceleration route, yet long-horizon trajectory generation poses a memory–accuracy trade-off: autoregressive models accumulate errors across successive predictions, while full-horizon temporal attention evaluates quadratically many frame pairs. Generating shorter segments reduces the jointly modeled sequence length but can affect agreement with reference trajectory distributions. We introduce RESCALE, a scale-recursive factorization and sampling procedure for conditional molecular trajectories. RESCALE jointly denoises separately maintained temporal states through recursive cross-scale conditioning, using a common generator architecture with level-specific parameters. This enables longer jointly generated sequences under practical memory constraints without sequential short-segment rollouts. We evaluate forward simulation on tetrapeptide and fast-folding protein systems, and transition path sampling on tetrapeptides. RESCALE improves agreement with reference torsional distributions, populations along TICA coordinates, and MSM state occupations. Memory-capacity tests demonstrate $4.5\times$ longer jointly generated sequences than MDGen at a 27.5 GiB peak-memory budget, while forward-simulation benchmarks show up to a $6\times$ inference speedup over MDGen. Results on held-out proteins further support scale-recursive generation as a promising approach to scalable MD trajectory modeling.

1 INTRODUCTION

Molecular Dynamics (MD) simulations provide atomistic insights into fundamental biological processes, from protein folding (Lane et al., 2013) to ligand binding (Liu et al., 2017). Many properties of biological relevance, such as equilibrium populations (Slatkin, 1993) and transition rates between metastable states (Langer, 1969), are governed by rare events that unfold over long timescales, making sufficiently long and converged trajectories essential for reliable modeling (Ormeño & General, 2024). Conventional MD relies on explicit force-field integration, leading to a profound temporal gap: while biological function often occurs on millisecond scales, the governing equations of motion must be integrated at femtosecond intervals to maintain numerical stability (Alder & Wainwright, 1959; Frenkel & Smit, 2023). This 10^{12} order-of-magnitude discrepancy makes the brute-force simulation computationally prohibitive: even reaching millisecond-scale folding trajectories required Anton, a custom-built molecular-dynamics supercomputer (Lindorff-Larsen et al., 2011).

Machine-learning interatomic potentials (MLIPs), including ANI (Smith et al., 2017a), NequIP (Batzner et al., 2022), and MACE (Batatia et al., 2022), approximate reference potential-energy surfaces at substantially lower cost than direct electronic-structure calculations, while retaining step-by-step numerical integration. A complementary line of work, on which we focus, instead learns the distribution of MD trajectories directly from existing simulation data. These models seek to accelerate trajectory generation while reproducing statistical properties of reference simulations (Jing et al., 2024b). Autoregressive (AR) models (Klein et al., 2023) can accumulate modeling errors through repeated conditional predictions over long horizons.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Diffusion models (Ho et al., 2020; Jing et al., 2022) and flow-matching frameworks (Lipman et al., 2022; Albergo & Vanden-Eijnden, 2023) in principle support joint trajectory generation. Full-horizon temporal attention evaluates $\mathcal{O}(T^2)$ frame pairs, which can limit the jointly generated sequence length under a fixed GPU-memory budget. For the attention-based MDGen configuration evaluated here (Jing et al., 2024b), extending beyond the jointly generated segment length requires successive rollouts. Repeated rollouts condition subsequent segments on generated states and can accumulate modeling errors, motivating our evaluation of joint-generation capacity and distributional accuracy under matched rollout schedules (section 3.5).

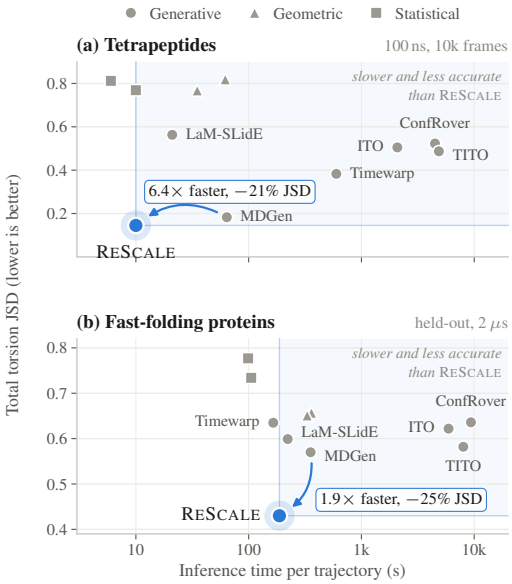

Figure 1: **Accuracy versus inference cost** (Table 1). ReScale denotes ReScale-L3. The shaded region contains methods that are both slower and less accurate than ReScale.

Molecular trajectories exhibit multi-scale temporal structure, combining local fluctuations, collective rearrangements, and conformational transitions. Their characteristic timescales overlap and vary with the system and conditions. The depth of this hierarchy is system-dependent: a tetrapeptide and a multi-domain protein can exhibit different numbers of well-separated kinetic regimes. This motivates a scale-recursive modeling strategy, in which the same coarse-to-fine architectural structure is repeated across temporal resolutions. At each level, a generator updates its noisy temporal representation using context from the coarser level. This recurrence is an architectural inductive bias; it does not require the physical dynamics at different scales to follow identical distributions. The hierarchy depth determines how many cross-scale conditioning stages connect the coarsest representation to the full-resolution trajectory within each denoising step.

Based on this perspective, we introduce **ReScale**, a scale-recursive generative framework for long-horizon molecular dynamics trajectories. ReScale explicitly factorizes the temporal axis into K levels with $T = \prod_{k=1}^{K} t_k$, where T is the total number of frames, and models each level with a flow-matching backbone. The generator architecture is reused across levels, while each level has its own parameters ω_k to specialize to its temporal granularity. Each level maintains its own noisy state. All levels advance through denoising together: within each step, a coarser state is updated and then supplies context for the finer-level updates. We refer to this nested temporal factorization as *fractalization*. Scale recursion organizes cross-scale conditioning within each denoising step, while fragments within a level can be updated in parallel given the noisy coarse context. This factorization replaces full-horizon attention with attention over shorter sequences at each level, enabling longer trajectories within the available memory budget.

Each fine fragment receives a bidirectional window containing its parent state and neighboring coarse states. The Mori–Zwanzig formalism motivates accounting for memory in reduced descriptions of dynamics (Lin et al., 2023). Our context mechanism uses this motivation without explicitly deriving or approximating a memory kernel. Hierarchy depth determines the number of cross-scale conditioning stages within each denoising step.

We evaluate scale-recursive generation through forward simulation on tetrapeptide and fast-folding protein systems and transition-path sampling on tetrapeptides. Our experiments assess torsional distributions, populations along TICA coordinates, MSM state occupations, and inference cost. ReScale improves distributional accuracy over trajectory-generation baselines while achieving approximately 6× and 1.9× inference speedups over MDGen on tetrapeptides and proteins, respectively. Memory-capacity and matched-rollout comparisons further examine the achievable joint-generation length and distributional accuracy (fig. 1).

Contributions. (i) We introduce **ReScale**, a scale-recursive parameterization of conditional molecular-trajectory generation across temporal resolutions. (ii) We develop a coupled denoising procedure with bidirectional cross-scale conditioning and parallel fragment updates, reduc-

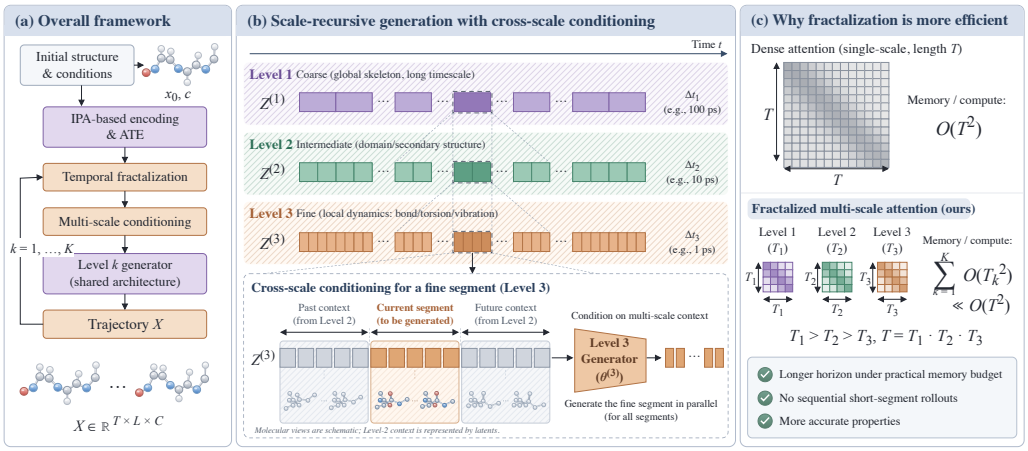

Figure 2: **Scale-recursive trajectory generation with ReScale. (a):** Generation pipeline: IPA-based encoding and ATE feed into K recursive levels of temporal fractalization and multi-scale conditioning, producing latents $\hat{\mathbf{Z}}^{(K)}$ that decode to the trajectory $\mathbf{X}$. **(b):** Cross-scale conditioning connects L1→L2→L3 within each denoising step, with boundary conditioning at the root. All levels retain noisy states throughout generation. **(c):** Schematic attention blocks with local factors t_k. The balanced-hierarchy bound assumes fixed K and context lengths $w_k = \mathcal{O}(t_k)$; section 2.8 gives the full attention-score count. In (a), the loop denotes the coarse-to-fine sweep within each denoising step; levels are not generated sequentially. In (c), T_k denotes the local factor t_k of section 2.3. Time intervals in (b) are illustrative.

ing attention cost and enabling full-horizon generation without sequential physical-time rollouts. (iii) We demonstrate improved distributional accuracy and computational efficiency on tetrapeptide and fast-folding protein benchmarks, including evaluation on held-out proteins.

2 Methodology

2.1 Problem Setup and Notation

A molecular system has atomic positions $\mathbf{q} \in \mathbb{R}^{3N_{\mathrm{at}}}$ and potential energy $U(\mathbf{q})$, where N_{at} is the number of atoms. Langevin dynamics (Ermak & McCammon, 1978) samples the Boltzmann distribution $p(\mathbf{q}) \propto e^{-U(\mathbf{q})/k_B\mathcal{T}}$ (Chandler, 1987), with Boltzmann constant k_B and temperature $\mathcal{T}$. Metastability creates a timescale gap between rapid atomic motions and slow transitions between conformational states (Frauenfelder et al., 1991; Noé et al., 2009; Shaw et al., 2010).

We represent a trajectory by descriptors $\mathbf{X}_{1:T} \in \mathbb{R}^{T\times L\times C}$, where T is the number of frames, L the number of residues, and C the per-residue descriptor dimension. Following Jing et al. (2024b), the descriptors comprise roto-translational offsets relative to conditioning frames and torsion angles (section D.2). Hidden representations have width D. ReScale supports *forward simulation*, $p(\mathbf{X}_{2:T} \mid \mathbf{x}_1)$, and *transition path sampling (TPS)*, $p(\mathbf{X}_{2:T-1} \mid \mathbf{x}_1, \mathbf{x}_T)$. We denote these boundary conditions by $\mathcal{B}$ and distinguish physical frame index t from generative flow time $s \in [0, 1]$.

2.2 Method Overview

ReScale organizes generation into K temporal levels by factorizing $T = \prod_{k=1}^{K} t_k$ (fig. 2). A common architecture is used at every level, with level-specific parameters ω_k. Absolute time embedding (ATE) is applied before folding temporal blocks into latent representations (sections 2.4 and 2.5).

Each level maintains its own noisy state throughout generation. All levels evolve over a shared flow-time schedule; within each denoising step, the coarser state is updated and passed as context to the next finer level. Thus, cross-scale conditioning proceeds from coarse to fine at every step, while the levels denoise jointly. Fragments within a level can be evaluated in parallel given this noisy coarse context. The hierarchy replaces full-horizon attention with shorter, context-conditioned blocks. Section 2.8 analyzes their attention cost and distinguishes it from measured peak GPU memory.

2.3 SCALE-RECURSIVE TEMPORAL DECOMPOSITION

Let $T_0 = 1$, $T_k = \prod_{\ell=1}^{k} t_\ell$, and $S_k = T/T_k$. Level k contains T_k temporal states, each associated with a block of S_k original frames. The levels range from coarse ($k = 1$) to full temporal resolution ($k = K, T_K = T$). At flow time s, their separately maintained noisy states are

$$\mathbf{Z}_s^{(k)} \in \mathbb{R}^{T_k \times L \times D}, \qquad k = 1, \dots, K. \tag{1}$$

The finest state has one latent per frame; the output readout in fig. 2 maps it to the molecular descriptors. Accordingly, latent states $\mathbf{Z}^{(K)}$ and descriptors $\mathbf{X}$ have distinct feature dimensions.

Joint generation across scales. The coupled sampler defines a joint endpoint distribution $p_\omega(\mathbf{Z}_1^{(1)}, \dots, \mathbf{Z}_1^{(K)} \mid \mathcal{B})$. Its finest-level marginal and trajectory readout are

$$\begin{aligned} p_\omega(\mathbf{Z}_1^{(K)} \mid \mathcal{B}) &= \int p_\omega(\mathbf{Z}_1^{(1)}, \dots, \mathbf{Z}_1^{(K)} \mid \mathcal{B}) \prod_{k=1}^{K-1} \mathrm{d}\mathbf{Z}_1^{(k)}, \\ \widehat{\mathbf{X}}_{1:T} &= \mathrm{Dec}(\mathbf{Z}_1^{(K)}, \mathcal{B}). \end{aligned} \tag{2}$$

Here Dec denotes the descriptor readout. The recursion is imposed on the coupled updates below: finer levels receive evolving noisy coarse states throughout denoising, rather than a fixed, completed coarse sample.

Fragment-wise updates. At level k, partition the state into T_{k-1} fragments with index sets $I_{k,j} = \{(j-1)t_k + 1, \dots, jt_k\}$. Each fragment's velocity is computed by

$$\left[\mathbf{V}_s^{(k)}\right]_{I_{k,j}} = v_{\omega_k}\left(\left[\mathbf{Z}_s^{(k)}\right]_{I_{k,j}}, s \mid \mathcal{C}_{s,j}^{(k)}\right), \quad j = 1, \dots, T_{k-1}. \tag{3}$$

For the root, $\mathcal{C}_{s,1}^{(1)} = \mathcal{B}$; finer-level context comes from the noisy parent state (section 2.6). Parameters are shared across fragments within a level and separate across levels. This block structure allows parallel fragment updates given the current coarse context; it does not assume independence conditioned only on the final coarse state.

2.4 PRE-FOLDING ABSOLUTE TIME EMBEDDING (ATE)

We augment each frame with an A-dimensional absolute temporal embedding before folding (Vaswani et al., 2017):

$$\widehat{\mathbf{x}}_t = [\mathbf{x}_t \parallel \mathrm{ATE}(t)] \in \mathbb{R}^{L \times (C+A)}, \qquad t = 1, \dots, T. \tag{4}$$

The embedding is broadcast over residues and concatenated along the feature dimension. This supplies the global physical-time position of each frame before temporal compression.

2.5 LATENT FRACTALIZATION

Fractalization constructs the level-wise clean targets from a training trajectory. At level k, concatenate blocks of S_k ATE-augmented frames along the feature dimension and project them to width D:

$$\mathbf{y}_j^{(k)} = \mathrm{concat}_{\mathrm{feat}}\left[\widehat{\mathbf{x}}_{S_k(j-1)+1:S_kj}\right] \mathbf{W}^{(k)}, \quad \mathbf{W}^{(k)} \in \mathbb{R}^{S_k(C+A) \times D}. \tag{5}$$

For $j = 1, \dots, T_k$, these targets form $\mathbf{Y}^{(k)} \in \mathbb{R}^{T_k \times L \times D}$. Since $S_K = 1$, the finest target preserves the frame count. We use $\mathbf{Y}^{(k)}$ for clean targets and $\mathbf{Z}_s^{(k)}$ for evolving noisy states. Folding defines the target representations; at inference, each level maintains its own state rather than rebuilding it from the noisy finest-level trajectory.

2.6 BIDIRECTIONAL CROSS-SCALE CONDITIONING

Temporal compression motivates conditioning on a neighborhood of coarse states. The Mori–Zwanzig formalism describes how projecting dynamics onto reduced variables can introduce memory effects

(Lin et al., 2023). We use this as motivation for cross-scale context. For a fine fragment j at level $k > 1$, the context window is

$$\mathcal{C}_{s,j}^{(k)} = \mathrm{Win}_{k,j}(\mathbf{Z}_s^{(k-1)}) = \left[\mathbf{z}_{s,j-u_k}^{(k-1)} : \mathbf{z}_{s,j+v_k}^{(k-1)}\right] \in \mathbb{R}^{w_k \times L \times D}, \tag{6}$$

where $w_k = u_k + v_k + 1$ includes the central coarse state, and out-of-bounds entries are zero-padded. Past and future refer to physical-time neighbors within the generated trajectory. Both belong to the evolving noisy parent state. The window size w_k and fine-fragment length t_k are distinct hyperparameters. Context is refreshed during denoising to coordinate neighboring fragments; the discrete sampler uses the parent's newly updated state as specified below.

2.7 Flow-Matching Generators Across Scales

Each level uses a Scalable Interpolant Transformer (SiT) (Ma et al., 2024). The continuous-time description is a coupled flow on the collection of states:

$$\frac{\mathrm{d}\mathbf{Z}_s^{(k)}}{\mathrm{d}s} = \mathbf{V}_s^{(k)}, \qquad k = 1, \ldots, K, \tag{7}$$

with fragment velocities and noisy context given by eqs. (3) and (6). All levels evolve from $s = 0$ to $s = 1$ together. For training, with Gaussian noise $\boldsymbol{\epsilon}^{(k)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, a shared flow time defines the level-wise interpolants $\mathbf{Z}_s^{(k)} = (1 - s)\boldsymbol{\epsilon}^{(k)} + s\mathbf{Y}^{(k)}$ and velocity targets $\mathbf{Y}^{(k)} - \boldsymbol{\epsilon}^{(k)}$. The flow-matching losses are summed across levels without an auxiliary loss (algorithm 1).

Cross-scale updates within a denoising step. Let s_n be a solver time, $h_n = s_{n+1} - s_n$, and $\mathbf{Z}_n^{(k)} = \mathbf{Z}_{s_n}^{(k)}$. The parent-updated Euler sweep illustrates the sampling order:

$$\begin{aligned} \widetilde{\mathcal{C}}_{n,j}^{(k)} &= \begin{cases} \mathcal{B}, & k = 1, \\ \mathrm{Win}_{k,j}(\mathbf{Z}_{n+1}^{(k-1)}), & k > 1, \end{cases} \\ \left[\mathbf{Z}_{n+1}^{(k)}\right]_{I_{k,j}} &= \left[\mathbf{Z}_n^{(k)}\right]_{I_{k,j}} + h_n v_{\omega_k}\left(\left[\mathbf{Z}_n^{(k)}\right]_{I_{k,j}}, s_n \mid \widetilde{\mathcal{C}}_{n,j}^{(k)}\right). \end{aligned} \tag{8}$$

Within each step, levels are updated in order $1, \ldots, K$, so finer updates use the already updated noisy parent at s_{n+1}. After this sweep, every level has advanced one step. This discrete conditioning convention is distinct from conditioning on a fully denoised coarse trajectory. The continuous description in eq. (7) uses a common s; section C.3 gives the discrete procedure.

For $k > 1$, the transformer concatenates each noisy fragment with its context, forming a tensor of shape $T_{k-1} \times (t_k + w_k) \times L \times D$. Only the t_k fragment positions contribute to that level's state update; context positions are supplied by the parent. One-shot generation means covering the target physical horizon without sequential short-segment rollouts, while retaining the repeated, coupled denoising steps.

2.8 Computational and Memory Efficiency

ReScale reduces the temporal attention cost of joint trajectory generation through shorter, context-conditioned blocks. At level $k > 1$, each of the T_{k-1} fragments contains t_k target tokens and $w_k = u_k + v_k + 1$ context tokens. Omitting fixed boundary-conditioning tokens and common batch, residue, and head factors, the total number of temporal attention-score entries per layer and denoising step is

$$A(T) = t_1^2 + \sum_{k=2}^{K} T_{k-1}(t_k + w_k)^2, \tag{9}$$

compared with T^2 for full-horizon attention. For fixed hierarchy depth K, balanced factors $t_k = \Theta(T^{1/K})$, and context sizes $w_k = \mathcal{O}(t_k)$, this gives

$$A(T) = \mathcal{O}\left(T^{1+1/K}\right). \tag{10}$$

This scaling concerns the temporal attention component and assumes that the hierarchy factors are balanced as the sequence length increases.

Table 1: **Forward-simulation results on tetrapeptide and protein systems.** Torsional and TICA population JSD and MSM occupation error; lower is better. Best values are bold; second-best values are underlined. Inference times are in seconds. Baseline protocols are given in section D.5.

	Category	Method	Torsions (JSD) ↓			TICA populations (JSD) ↓		MSM	Infer.
			Backbone	Sidechain	Total	$TICA_0$	$TICA_{0,1}$	occ. error ↓	Time
Tetrapeptide	Statistical	ARIMA	0.816	0.809	0.812	0.812	0.829	0.701	**6s**
		W-ARIMA	0.766	0.772	0.769	0.761	0.810	0.577	10s
	Geometric	SchNet	0.822	0.816	0.819	0.816	0.831	0.731	62s
		EGNN	0.772	0.764	0.768	0.763	0.810	0.582	35s
	Generative	Timewarp	0.325	0.427	0.383	0.265	0.419	0.222	599s
		ITO	0.564	0.462	0.505	0.538	0.756	0.414	2083s
		MDGen	0.263	**0.103**	0.183	0.273	0.380	0.218	64s
		ConfRover	0.337	0.706	0.523	0.411	0.596	0.402	4490s
		TITO	0.530	0.445	0.487	0.471	0.601	0.368	4864s
		LaM-SLidE	0.608	0.519	0.563	0.586	0.713	0.442	21s
	Ours	RESCALE-L2	0.220±0.003	0.136±0.005	0.178±0.002	**0.242**±0.005	**0.340**±0.003	**0.217**±0.007	11s
		RESCALE-L3	**0.182**±0.004	0.108±0.005	**0.145**±0.004	0.271±0.009	0.353±0.008	0.287±0.010	10s
Protein	Statistical	ARIMA	0.766	0.789	0.777	0.775	0.825	0.299	**98.9s**
		W-ARIMA	0.713	0.755	0.734	0.768	0.809	0.407	105.3s
	Geometric	SchNet	0.647	0.667	0.657	0.661	0.783	0.365	360.4s
		EGNN	0.662	0.640	0.651	0.562	0.722	0.285	330.4s
	Generative	Timewarp	0.682	0.589	0.635	0.557	0.732	0.422	165.3s
		ITO	0.638	0.606	0.622	0.601	0.734	0.363	5922.4s
		MDGen	0.634	0.533	0.570	0.606	0.773	0.260	354.3s
		ConfRover	0.583	0.688	0.636	0.534	0.749	0.259	9362s
		TITO	0.590	0.573	0.582	0.540	0.671	0.270	8010s
		LaM-SLidE	0.591	0.607	0.599	0.550	0.675	0.296	222s
	Ours	RESCALE-L2	0.639±0.003	0.374±0.002	0.524±0.002	0.604±0.005	0.768±0.002	0.259±0.023	200.7s
		RESCALE-L3	**0.509**±0.024	**0.327**±0.006	**0.430**±0.015	**0.506**±0.012	**0.669**±0.013	**0.190**±0.021	187.4s

At each denoising step, the levels are updated from coarse to fine, while fragments within a level can be evaluated in parallel given the updated parent state. A sampler with N steps therefore contains NK dependent level-update stages. Each stage still performs sequence-length-dependent work, and its runtime depends on fragment batching and available hardware.

Peak GPU memory additionally includes model parameters, the separately maintained noisy states, and intermediate activations. The noisy states alone contain $LD\sum_{k=1}^{K} T_k$ entries. Consequently, attention-score counts provide an architectural comparison, while practical memory capacity is assessed experimentally. Under the measured 27.5 GiB peak-memory budget, RESCALE jointly generates 10,000 frames compared with 2,212 for MDGen, approximately a $4.5\times$ increase in demonstrated sequence length (fig. 3).

3 EXPERIMENTS

We conduct experiments to evaluate RESCALE through the following four questions:

- **RQ1.** Does RESCALE outperform statistical, geometric, and iterative baselines on both small tetrapeptide and larger protein systems, jointly in distributional accuracy and inference cost?
- **RQ2.** How does hierarchy depth affect torsional distributions, TICA populations, and MSM state occupations across tetrapeptide and protein systems?
- **RQ3.** How do memory constraints and rollout schedules affect joint-generation capacity and distributional accuracy?
- **RQ4.** How do temporal embeddings, cross-scale conditioning, parameter sharing, and attention structure affect RESCALE's distributional accuracy?

3.1 EXPERIMENTAL SETUP

Datasets and tasks. We evaluate on the tetrapeptide dataset of Jing et al. (2024b) and the 12 fast-folding proteins of Lindorff-Larsen et al. (2011). The tetrapeptide dataset contains approxi-

mately 3,000 training systems and 100 systems each for validation and testing, with 100 ns reference trajectories. The proteins span 10–80 residues and diverse fold topologies; we use a 9/1/2 training/validation/test split. Dataset preparation and implementation details are given in sections D.1 and D.3. We evaluate 10,000-frame forward simulation on both primary datasets and transition path sampling on tetrapeptides, as defined in section 2.1.

We additionally evaluate forward simulation on ATLAS (Vander Meersche et al., 2024), a structurally diverse protein benchmark providing three independent 100 ns all-atom MD replicas per protein. The ATLAS results are reported in section 3.3.

Baselines. We compare RESCALE with statistical baselines ARIMA (Shumway & Stoffer, 2017) and Wavelet ARIMA (W-ARIMA) (Kriechbaumer et al., 2014); geometric models such as SchNet (Schütt et al., 2018) and EGNN (Satorras et al., 2021); and generative trajectory or transition models ITO (Schreiner et al., 2023), Timewarp (Klein et al., 2023), MDGen (Jing et al., 2024b), ConfRover (Shen et al., 2026), TITO (Diez et al., 2026), and LaM-SLidE (Sestak et al., 2026). The latter three provide autoregressive, lag-conditioned transition, and latent block-generation comparisons, respectively. Their training and sampling protocols are described in section D.5.

Metrics. Following Jing et al. (2024b), we evaluate RESCALE in terms of distributional accuracy and computational efficiency: (1) Torsional distributions: Jensen–Shannon distance (JSD) between reference and generated backbone (ϕ, ψ) and sidechain (χ) torsion distributions. (2) Conformational populations: marginal (TICA_0) and joint ($\text{TICA}_{0,1}$) population JSD in reference-fitted time-lagged independent component analysis (TICA) coordinates, together with Markov state model (MSM) occupation error. (3) Computational efficiency: wall-clock inference time. The distributional metrics assess agreement with reference conformational distributions; they do not by themselves establish accurate transition rates or other kinetic quantities. Further details are given in section D.4; MD-replica reference scores are reported in section E.5.

3.2 MAIN RESULTS: FORWARD SIMULATION (RQ1, RQ2)

Table 1 compares 10,000-frame forward simulation on tetrapeptides and proteins. RESCALE-L2 and RESCALE-L3 use two and three temporal levels, respectively.

On tetrapeptides, the RESCALE family achieves the best score on five of the six distributional metrics. RESCALE-L3 reduces backbone and total torsion JSD relative to MDGen by 30.8% and 20.8%, respectively. RESCALE-L2 achieves the lowest TICA_0 and $\text{TICA}_{0,1}$ population JSD (0.242 and 0.340) and MSM occupation error (0.217). On proteins, RESCALE-L3 has the lowest error on all six metrics, including comparisons with ConfRover, TITO, and LaM-SLidE. The joint TICA population errors are close: 0.669 for RESCALE-L3 and 0.671 for TITO. Across both datasets, the RESCALE family leads on 11 of the 12 distributional metrics.

RESCALE-L3 is the fastest learned generator on tetrapeptides, with a reported inference time of 10 s, compared with 64 s for MDGen and 21 s for LaM-SLidE. On proteins, its runtime is 187.4 s, compared with 354.3 s for MDGen, 222 s for LaM-SLidE, and 8,010 s for TITO and 9,362 s for ConfRover. The speedups over MDGen are $6.4\times$ and $1.9\times$ on tetrapeptides and proteins, respectively. Timewarp is faster on proteins (165.3 s), while RESCALE-L3 has lower error on all six distributional metrics.

The effect of hierarchy depth differs across datasets and observables. On tetrapeptides, increasing depth from L2 to L3 reduces backbone, sidechain, and total torsion JSD by 17–21%, while TICA population JSD and MSM occupation error increase. On proteins, L3 reduces all six distributional errors relative to L2 by 13–27%. These results indicate that the benefits of additional refinement levels depend on the system and the distributional quantity being evaluated.

3.3 GENERALIZATION TO ATLAS

We additionally evaluate forward simulation on ATLAS to examine performance on a broader protein benchmark. RESCALE achieves the lowest error on five of the six reported distributional metrics. Compared with MDGen, it reduces total torsion JSD from 0.364 to 0.318 and joint TICA population JSD from 0.796 to 0.600. MDGen achieves slightly lower MSM occupation error (0.287 versus

0.288). These results extend the evidence for improved distributional fidelity beyond the fast-folding protein benchmark. Table 2 summarizes three metrics; the full six-metric comparison is provided in section E.3.

Table 2: **ATLAS forward-simulation results.** Total torsion and joint TICA population JSD, and MSM occupation error; lower is better. Best values are bold; second-best values are underlined. The full six-metric comparison is reported in Table 5.

Metric ↓	ARIMA	W-ARIMA	SchNet	EGNN	Timewarp	MDGen	RESCALE
Total torsion JSD	0.808	0.675	0.766	0.687	0.685	<u>0.364</u>	**0.318**
$TICA_{0,1}$ population JSD	0.830	0.827	0.805	<u>0.754</u>	0.760	0.796	**0.600**
MSM occupation error	0.561	0.449	0.376	0.303	0.291	**0.287**	<u>0.288</u>

3.4 TRANSITION PATH SAMPLING (RQ1, RQ3)

Table 3 reports distributional accuracy on the Alanine-4 transition-path sampling task over the stated 10,000-step horizon. The three reported MDGen settings generate 5,000, 2,000, and 1,000 frames, while RESCALE generates all 10,000 frames. RESCALE achieves the lowest backbone and total torsion JSD and the lowest marginal and joint TICA population JSD. It ties the 2,000-frame MDGen setting on MSM occupation error (0.228), while the 5,000-frame MDGen setting achieves the lowest sidechain JSD.

Table 3: **Transition Path Sampling (TPS) Results.** Performance comparison on Alanine-4 over a 10,000-step interpolation horizon. Baselines are evaluated at varying strides to satisfy the physical time constraint, while RESCALE generates the full resolution path one-shot.

Method	#Rollout	#Frames	Torsions (JSD) ↓			TICA populations (JSD) ↓		MSM	Infer.
			Backbone	Sidechain	Total	$TICA_0$	$TICA_{0,1}$	occ. error ↓	Time (s)
MDGen (Jing et al., 2024b)	2	5,000	0.271	**0.119**	0.194	0.295	0.420	0.243	62s
	5	2,000	0.394	0.202	0.297	0.343	0.470	**0.228**	62s
	10	1,000	0.317	0.195	0.256	0.363	0.525	0.367	60s
RESCALE (Ours)	**1**	**10,000**	**0.258**	<u>0.144</u>	**0.192**	**0.257**	**0.408**	**0.228**	**11s**

No MDGen setting is best on all six distributional metrics. The separate matched-rollout experiment in section 3.5 examines the effect of rollout schedules. Projected free-energy surfaces are provided in section E.1 as qualitative comparisons of the generated conformational distributions.

3.5 MEMORY CAPACITY AND MATCHED-ROLLOUT GENERATION (RQ3)

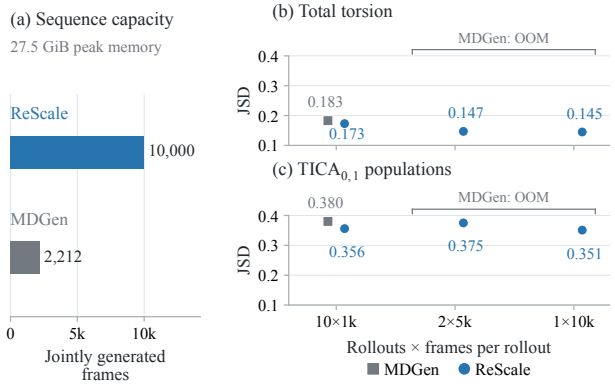

Figure 3: **Memory capacity and matched-rollout accuracy.** (a) Jointly generated sequence lengths under a 27.5 GiB peak-memory budget. (b–c) Total torsion and joint TICA population JSD under three rollout schedules, each targeting 10,000 total frames; lower is better. OOM denotes out-of-memory failure. The memory budget in (a) applies to the capacity comparison.

Figure 3 evaluates joint-generation capacity and distributional accuracy under matched rollout schedules. At a peak-memory budget of 27.5 GiB, RESCALE generates 10,000 frames jointly, compared with 2,212 for MDGen—a $4.5\times$ increase in demonstrated sequence length. Under the shared feasible schedule of 10 rollouts $\times$ 1,000 frames, RESCALE reduces total torsion JSD from 0.183 to 0.173 and joint TICA population JSD from 0.380 to 0.356. RESCALE also completes the 5,000- and 10,000-frame-per-rollout settings, where MDGen encounters out-of-memory errors. These results support longer joint generation while maintaining competitive distributional accuracy.

3.6 Ablation Study (RQ4)

We compare seven ReScale-L3 configurations and two MDGen variants under the same 30-epoch training and evaluation protocol (Table 4). Ours uses ATE, bidirectional conditioning, and per-level generator weights, without auxiliary loss; each other ReScale variant changes one component. This budget is separate from the fully trained main results.

Table 4: **Controlled 30-epoch ablations on tetrapeptides.** Torsional and TICA population JSD and MSM occupation error; lower is better. Best values are bold; second-best values are underlined.

Metric ↓	ReScale-L3 variants							MDGen variants	
	Ours	−ATE	Uni-dir. cond.	−Cond.	Shared (171M)	−RoPE	Window attn.	MDGen	+ATE
Backbone JSD	<u>0.177</u>	**0.169**	0.199	0.392	0.233	0.196	0.345	0.236	0.230
Sidechain JSD	**0.111**	<u>0.114</u>	0.117	0.171	0.120	0.128	0.130	0.136	0.133
Total torsion JSD	<u>0.145</u>	**0.142**	0.158	0.281	0.177	0.156	0.220	0.178	0.174
TICA_0 JSD	**0.266**	<u>0.271</u>	0.274	0.380	0.286	0.281	0.385	0.307	0.319
$\text{TICA}_{0,1}$ JSD	**0.351**	<u>0.359</u>	<u>0.359</u>	0.484	0.378	0.378	0.481	0.413	0.416
MSM occ. error	**0.277**	0.293	0.285	0.305	0.287	0.301	0.295	<u>0.282</u>	0.321

Uni-dir.: unidirectional cross-scale conditioning; −Cond.: no context-window conditioning; Shared: generator weights shared across levels. Window attn.: global attention replaced by window attention.

Under the shared training budget, ATE improves sidechain, TICA population, and MSM occupation scores, while the no-ATE configuration has lower backbone and total torsion JSD. Restricting conditioning to one direction or removing context increases all six errors relative to bidirectional conditioning. Sharing generator parameters also increases every reported error, including when the shared generator is deepened to match the parameter count (section E.6).

4 Limitations and Future Work

ReScale-L3's multi-scale factorization helps most on long, structurally rich trajectories; gains on short tetrapeptide rollouts are modest. While ReScale consistently outperforms prior generative baselines on proteins, a substantial absolute gap to the oracle reference persists (section E.5), reflecting both the limited training set (9 proteins) and the difficulty of capturing diverse fold topologies from such scale. Recovery of NTL9 folding transitions and the associated rare-event kinetics remains unresolved in the evaluated models (section E.4). The recursion depth and per-level strides are chosen by hand, and the trajectory length must be exactly factorizable into the chosen strides. Compute limits also prevented us from running protein transition-path sampling at the scales reported for forward simulation. Future work includes choosing the temporal hierarchy automatically from data-driven signals (physical timescales, dominant TICA components, or autocorrelation structure), relaxing factorizability via overlap-allowed fractal blocks, and scaling up both training data and the TPS study.

5 Conclusion

We introduced ReScale, a scale-recursive factorization and sampling procedure for conditional molecular dynamics trajectories. ReScale jointly denoises separately maintained temporal states, passing updated noisy coarse states to finer levels through cross-scale conditioning. It uses a common generator architecture with level-specific parameters. This organization reduces attention cost and enables parallel refinement of fine-scale fragments without sequential physical-time rollouts. Experiments on tetrapeptide and fast-folding protein systems demonstrate improved distributional accuracy and computational efficiency over trajectory-generation baselines. These results support scale recursion as a practical approach to generating longer molecular trajectories under memory constraints.

AI use statement

During preparation of this version, generative AI tools assisted with literature searches, manuscript editing, formatting, figure and table presentation, and checks of mathematical exposition and the interpretation of reported results. The authors remain responsible for the final content.

REFERENCES

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations (ICLR)*, 2023.

Berni J Alder and Thomas E Wainwright. Studies in molecular dynamics. i. general method. *The Journal of Chemical Physics*, 1959.

Yaxin An and Sanket A Deshmukh. Machine learning approach for accurate backmapping of coarse-grained models to all-atom models. *Chemical communications*, 56(65):9312–9315, 2020.

Ilyes Batatia, David P Kovacs, Gregor Simm, Christoph Ortner, and Gábor Csányi. Mace: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in neural information processing systems*, 35:11423–11436, 2022.

Ilyes Batatia, Philipp Benner, Yuan Chiang, Alin M Elena, Dávid P Kovács, Janosh Riebesell, Xavier R Advincula, Mark Asta, Matthew Avaylon, William J Baldwin, et al. A foundation model for atomistic materials chemistry. *The Journal of chemical physics*, 163(18), 2025.

Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022.

Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007.

Filippo Bigi, Sanggyu Chong, Agustinus Kristiadi, and Michele Ceriotti. Flashmd: long-stride, universal prediction of molecular dynamics. *arXiv preprint arXiv:2505.19350*, 2025.

Peter G Bolhuis, David Chandler, Christoph Dellago, and Phillip L Geissler. Transition path sampling: Throwing ropes over rough mountain passes, in the dark. *Annual Review of Physical Chemistry*, 53(1):291–318, 2002.

Gregory R Bowman, Vijay S Pande, and Frank Noé. *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*. Springer Science & Business Media, 2013.

JW Cannon. The fractal geometry of nature. by benoit b. mandelbrot. *The American Mathematical Monthly*, 91(9):594–598, 1984.

David Chandler. *Introduction to Modern Statistical Mechanics*. Oxford University Press, 1987.

Allan Dos Santos Costa, Ilan Mitnikov, Franco Pellegrini, Ameya Daigavane, Mario Geiger, Zhonglin Cao, Karsten Kreis, Tess Smidt, Emine Kucukbenli, and Joseph Jacobson. Equijump: Protein dynamics simulation via so (3)-equivariant stochastic interpolants. *arXiv preprint arXiv:2410.09667*, 2024.

Christoph Dellago, Peter G Bolhuis, Félix S Csajka, and David Chandler. Transition path sampling and the calculation of rate constants. *The Journal of chemical physics*, 108(5):1964–1977, 1998.

Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International conference on machine learning*, pp. 1174–1183. PMLR, 2018.

Juan Viguera Diez, Mathias Schreiner, and Simon Olsson. Transferable generative models bridge femtosecond to nanosecond time-step molecular dynamics. *Science advances*, 12(15):eaed2333, 2026.

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.

Donald L Ermak and J Andrew McCammon. Brownian dynamics with hydrodynamic interactions. *The Journal of Chemical Physics*, 69(4):1352–1360, 1978.

Kenneth Falconer. *Fractal geometry: mathematical foundations and applications*. John Wiley & Sons, 2013.

Jens Feder. *Fractals*. Springer Science & Business Media, 2013.

Hans Frauenfelder, Stephen G Sligar, and Peter G Wolynes. The energy landscapes and motions of proteins. *Science*, 254(5038):1598–1603, 1991.

Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Academic press, 2023.

Xiang Fu, Brandon M Wood, Luis Barroso-Luque, Daniel S Levine, Meng Gao, Misko Dzamba, and C Lawrence Zitnick. Learning smooth and expressive interatomic potentials for physical property prediction. *arXiv preprint arXiv:2502.12147*, 2025.

Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.

Fuchun Ge, Yuxinxin Chen, and Pavlo O Dral. Artificial intelligence for direct prediction of molecular dynamics across chemical space. *Journal of Chemical Theory and Computation*, 22(4):1549–1562, 2026.

Charles J Geyer. Introduction to markov chain monte carlo. *Handbook of markov chain monte carlo*, 20116022(45):22, 2011.

Jiaqi Han, Minkai Xu, Aaron Lou, Haotian Ye, and Stefano Ermon. Geometric trajectory diffusion models. *Advances in Neural Information Processing Systems*, 37:25628–25662, 2024.

Lim Heo and Michael Feig. One bead per residue can describe all-atom protein structures. *Structure*, 32(1):97–111, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

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, 2022b.

Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24240–24253, 2022.

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

Bowen Jing, Hannes Stärk, Tommi Jaakkola, and Bonnie Berger. Generative modeling of molecular dynamics trajectories. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

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: 52863–52883, 2023.

Thomas Kriechbaumer, Andrew Angus, David Parsons, and Monica Rivas Casado. An improved wavelet–arima approach for forecasting metal prices. *Resources Policy*, 39:32–41, 2014.

Thomas J Lane, Diwakar Shukla, Kyle A Beauchamp, and Vijay S Pande. To milliseconds and beyond: challenges in the simulation of protein folding. *Current opinion in structural biology*, 23 (1):58–65, 2013.

James S Langer. Statistical theory of the decay of metastable states. *Annals of Physics*, 54(2): 258–275, 1969.

Zhen Li, Xin Bian, Xiantao Li, and George Em Karniadakis. Incorporation of memory effects in coarse-grained modeling via the mori-zwanzig formalism. *The Journal of chemical physics*, 143 (24), 2015.

Yen Ting Lin, Yifeng Tian, Danny Perez, and Daniel Livescu. Regression-based projection for learning mori–zwanzig operators. *SIAM Journal on Applied Dynamical Systems*, 22(4):2890–2926, 2023.

Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, and David E Shaw. How fast-folding proteins fold. *Science*, 334(6055):517–520, 2011.

Yaron Lipman et al. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Kai Liu, Etsurou Watanabe, and Hironori Kokubo. Exploring the stability of ligand binding modes to proteins by molecular dynamics simulations. *Journal of computer-aided molecular design*, 31(2): 201–211, 2017.

Liyao Lyu and Huan Lei. Construction of coarse-grained molecular dynamics with many-body non-markovian memory. *Physical Review Letters*, 131(17):177301, 2023.

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. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

James A Maier, Carmenza Martinez, Koushik Kasavajhala, Lauren Wickstrom, Kevin E Hauser, and Carlos Simmerling. ff14sb: improving the accuracy of protein side chain and backbone parameters from ff99sb. *Journal of chemical theory and computation*, 11(8):3696–3713, 2015.

Benoit B Mandelbrot and John W Van Ness. Fractional Brownian motions, fractional noises and applications. *SIAM Review*, 10(4):422–437, 1968.

Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1):579, 2023.

Mark Neumann, James Gin, Benjamin Rhodes, Steven Bennett, Zhiyi Li, Hitarth Choubisa, Arthur Hussey, and Jonathan Godwin. Orb: A fast, scalable neural network potential. *arXiv preprint arXiv:2410.22570*, 2024.

Hai Nguyen, Daniel R Roe, and Carlos Simmerling. Improved generalized born solvent model parameters for protein simulations. *Journal of chemical theory and computation*, 9(4):2020–2034, 2013.

Frank Noé, Christof Schütte, Eric Vanden-Eijnden, Lothar Reich, and Thomas R Weikl. Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations. *Proceedings of the National Academy of Sciences*, 106(45):19011–19016, 2009.

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.

Franco Ormeño and Ignacio J General. Convergence and equilibrium in molecular dynamics simulations. *Communications Chemistry*, 7(1):26, 2024.

Yutack Park, Jaesun Kim, Seungwoo Hwang, and Seungwu Han. Scalable parallel algorithm for graph neural network interatomic potentials in molecular dynamics simulations. *Journal of chemical theory and computation*, 20(11):4857–4868, 2024.

Magnus Petersen, Gemma Roig, and Roberto Covino. Dynamicsdiffusion: Generating and rare event sampling of molecular dynamic trajectories using diffusion models. In *NeurIPS 2023 AI for Science Workshop*, 2023.

Stefano Piana, Kresten Lindorff-Larsen, and David E Shaw. How robust are protein folding simulations with respect to force field parameterization? *Biophysical journal*, 100(9):L47–L49, 2011.

Jan-Hendrik Prinz et al. Markov models of molecular kinetics: Generation and validation. *The Journal of Chemical Physics*, 2011.

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. ISSN 1089-7690. doi: 10.1063/1.4811489. URL `http://dx.doi.org/10.1063/1.4811489`.

Benjamin Rhodes, Sander Vandenhaute, Vaidotas Šimkus, James Gin, Jonathan Godwin, Tim Duignan, and Mark Neumann. Orb-v3: atomistic simulation at scale. *arXiv preprint arXiv:2504.06231*, 2025.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Mathias Schreiner, Ole Winther, and Simon Olsson. Implicit transfer operator learning: Multiple time-resolution models for molecular dynamics. *Advances in Neural Information Processing Systems*, 36:36449–36462, 2023.

Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International conference on machine learning*, pp. 9377–9388. PMLR, 2021.

Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of chemical physics*, 148(24), 2018.

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

Florian Sestak, Artur Toshev, Andreas Fürst, Günter Klambauer, Andreas Mayr, and Johannes Brandstetter. Lam-slide: Latent space modeling of spatial dynamical systems via linked entities. *Advances in Neural Information Processing Systems*, 38:140166–140218, 2026.

David E Shaw, Paul Maragakis, Kresten Lindorff-Larsen, Stefano Piana, Ron O Dror, Michael P Eastwood, Joseph A Bank, John M Jumper, J Kevin Salmon, Yibing Moraes, et al. Atomic-level characterization of the structural dynamics of proteins. *Science*, 330(6002):341–346, 2010.

Lifeng Shen, Weiyu Chen, and James Kwok. Multi-resolution diffusion models for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuning Shen, Lihao Wang, Huizhuo Yuan, Yan Wang, Bangji Yang, and Quanquan Gu. Simultaneous modeling of protein conformation and dynamics via autoregression. *Advances in Neural Information Processing Systems*, 38:109782–109822, 2026.

Robert H Shumway and David S Stoffer. Arima models. In *Time series analysis and its applications: with R examples*, pp. 75–163. Springer, 2017.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

Montgomery Slatkin. Isolation by distance in equilibrium and non-equilibrium populations. *Evolution*, 47(1):264–279, 1993.

Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. Ani-1, a data set of 20 million calculated off-equilibrium conformations for organic molecules. *Scientific data*, 4(1):170193, 2017a.

Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. Ani-1: an extensible neural network potential with dft accuracy at force field computational cost. *Chemical science*, 8(4):3192–3203, 2017b.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Fang Sun, Zijie Huang, Haixin Wang, Huacong Tang, Xiao Luo, Wei Wang, and Yizhou Sun. Graph fourier neural odes: Modeling spatial-temporal multi-scales in molecular dynamics. *arXiv preprint arXiv:2411.01600*, 2024.

Luke Thompson, Davy Guan, Dai Shi, Slade Matthews, Junbin Gao, and Andi Han. Atom: A pretrained neural operator for multitask molecular dynamics. *arXiv preprint arXiv:2510.05482*, 2025.

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. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Ruben Villegas, Dumitru Erhan, Honglak Lee, et al. Hierarchical long-term video prediction without supervision. In *International Conference on Machine Learning*, pp. 6038–6046. PMLR, 2018.

Lee-Ping Wang, Todd J Martinez, and Vijay S Pande. Building force fields: An automatic, systematic, and reproducible approach. *The journal of physical chemistry letters*, 5(11):1885–1891, 2014.

Fang Wu and Stan Z Li. Diffmd: A geometric diffusion model for molecular dynamics simulations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 5321–5329, 2023.

Liming Wu, Zhichao Hou, Jirui Yuan, Yu Rong, and Wenbing Huang. Equivariant spatio-temporal attentive graph networks to simulate physical dynamics. *Advances in Neural Information Processing Systems*, 36:45360–45380, 2023.

Chenxin Xu, Robby T Tan, Yuhong Tan, Siheng Chen, Yu Guang Wang, Xinchao Wang, and Yanfeng Wang. Eqmotion: Equivariant multi-agent motion prediction with invariant interaction reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1410–1420, 2023.

Han Yang, Chenxi Hu, Yichi Zhou, Xixian Liu, Yu Shi, Jielan Li, Guanzhi Li, Zekun Chen, Shuizhou Chen, Claudio Zeni, et al. Mattersim: A deep learning atomistic model across elements, temperatures and pressures. *arXiv preprint arXiv:2405.04967*, 2024.

Soojung Yang and Rafael Gómez-Bombarelli. Chemically transferable generative backmapping of coarse-grained proteins. *arXiv preprint arXiv:2303.01569*, 2023.

Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. *arXiv preprint arXiv:2403.01742*, 2024.

APPENDIX CONTENTS

A Theoretical Background **16**

A.1 Formal Definition of Self-Similar Fractals 16

A.2 Flow Matching and Stochastic Interpolants 16

A.3 Forward Simulation and Transition Path Sampling 16

B Related Work **17**

B.1 Generative Models for Molecular Dynamics 17

B.2 Machine-Learning Interatomic Potentials 18

B.3 Multi-Scale and Hierarchical Generative Models 18

C Method Details **18**

C.1 Two-Level Instantiation 18

C.2 Training Procedure 19

C.3 Inference Procedure 19

D Experimental Details **20**

D.1 Dataset Description 20

D.2 Trajectory Representation 21

D.3 Implementation Details and Hyperparameters 21

D.4 Evaluation Metrics 21

D.5 Baseline Configuration 22

D.6 Code 23

E Additional Results **23**

E.1 TICA Free Energy Surface Comparisons 24

E.2 Training Dynamic Comparisons 25

E.3 ATLAS Forward-Simulation Results 25

E.4 NTL9 at Varying Sampling Intervals 26

E.5 MD-Replica Reference for Forward Simulation 26

E.6 Generator Parameter Sharing Across Levels 27

A Theoretical Background

A.1 Formal Definition of Self-Similar Fractals

In this paper, *scale recursion* denotes the repeated use of a coarse-to-fine generative architecture across temporal resolutions, with separate parameters at each level. Coarse-to-fine denotes the direction of conditioning within each denoising step; all levels evolve jointly in flow time. The term *fractalization* describes the nested temporal factorization used to build that hierarchy. The mathematical notion of self-similarity below provides background for this terminology; exact geometric self-similarity of MD trajectories is not an assumption of the model.

A set S is a fractal if its Hausdorff dimension (Falconer, 2013) strictly exceeds its topological dimension; the hallmark property is *self-similarity*, whereby the set exhibits structurally similar patterns at different scales. Formally, a compact set $S \subset \mathbb{R}^d$ is strictly self-similar if it can be decomposed into N non-overlapping copies of itself, each scaled by a constant factor $s > 1$. The fractal dimension D_f then satisfies the power-law relation $N = s^{D_f}$, giving:

$$D_f = \frac{\log N}{\log s} \tag{11}$$

Unlike standard Euclidean objects (Feder, 2013) where D_f takes integer values, fractals possess non-integer dimensions that reflect their space-filling properties across scales.

A.2 Flow Matching and Stochastic Interpolants

We adopt the Scalable Interpolant Transformer (SiT) framework (Ma et al., 2024) for all generators in our hierarchy. SiT unifies continuous-time probability flow and diffusion models, offering efficient few-step generation that avoids the expensive SDE simulation required by score-based diffusion (Song et al., 2021; Ho et al., 2020).

For a target distribution p_1 and Gaussian base distribution $p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$, we define a time-dependent interpolant for generative flow-time $s \in [0, 1]$:

$$\mathbf{z}_s = \alpha_s \mathbf{z}_1 + \sigma_s \boldsymbol{\epsilon}, \quad \mathbf{z}_1 \sim p_1, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{12}$$

where α_s, σ_s are smooth interpolant schedules satisfying $(\alpha_0, \sigma_0) = (0, 1)$ and $(\alpha_1, \sigma_1) = (1, 0)$. The generative process is governed by the probability flow ODE (Lipman et al., 2022; Albergo & Vanden-Eijnden, 2023):

$$\dot{\mathbf{z}}_s = v_\theta(\mathbf{z}_s, s) \tag{13}$$

where v_θ is a learned velocity field. The model is trained by minimizing the conditional flow matching objective:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{\boldsymbol{\epsilon}, \mathbf{z}_1, s} \left[\| v_\theta(\mathbf{z}_s, s) - \dot{\mathbf{z}}_s \|^2 \right] \tag{14}$$

For a regular continuous flow, the change-of-variables identity gives

$$\log p_\theta(\mathbf{z}_1 \mid \mathcal{B}) = \log p_0(\mathbf{z}_0 \mid \mathcal{B}) - \int_0^1 \operatorname{Tr}\left(\frac{\partial v_\theta(\mathbf{z}_s, s \mid \mathcal{B})}{\partial \mathbf{z}_s} \right) \mathrm{d}s. \tag{15}$$

For coupled levels, $\mathbf{z}$ represents the complete collection of noisy states and the Jacobian is with respect to that joint state. This identity does not imply a product of endpoint densities conditioned only on completed parent samples.

In ReScale, the root velocity is conditioned on boundary states $\mathcal{B}$, and finer velocities receive bidirectional windows of the noisy parent state. The continuous description couples levels at a common flow time. The discrete sampler updates them coarse to fine within every step, passing each newly updated noisy state to the next level (eq. (8)). The finest-level marginal and descriptor readout determine the generated trajectory (eq. (2)).

A.3 Forward Simulation and Transition Path Sampling

Forward Simulation. Forward simulation predicts the future trajectory $\mathbf{X}_{2:T} = \{\mathbf{x}_2, \ldots, \mathbf{x}_T\}$ conditioned on an initial state $\mathbf{x}_1$. Standard autoregressive surrogates factorize this as $p(\mathbf{X}_{2:T} \mid \mathbf{x}_1) = \prod_{t=1}^{T-1} p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$, where each step is sampled from a learned transition kernel. While efficient for short-term prediction, this iterative rollout is the primary source of error accumulation addressed in this work.

Transition Path Sampling (TPS). TPS (Dellago et al., 1998; Bolhuis et al., 2002) addresses the boundary-value problem of sampling rare dynamical trajectories connecting two metastable states A and B. The goal is to sample from the conditional path ensemble:

$$P_{AB}(\mathbf{X}_{1:T}) \propto p(\mathbf{X}_{1:T})\, \mathbb{I}(\mathbf{x}_1 \in A)\, \mathbb{I}(\mathbf{x}_T \in B) \tag{16}$$

where $p(\mathbf{X}_{1:T})$ is the unbiased dynamical distribution and $\mathbb{I}$ denotes indicator functions. Traditional approaches such as shooting moves (Bolhuis et al., 2002) rely on expensive MCMC (Geyer, 2011) iterations in path space. RESCALE amortizes TPS by learning a generative model that directly targets this constrained distribution through coupled denoising over the full horizon, conditioning on both endpoints $\{\mathbf{x}_1, \mathbf{x}_T\}$.

B Related Work

B.1 Generative Models for Molecular Dynamics

Two complementary paradigms have emerged for learning MD trajectory distributions directly from data, each with distinct trade-offs between generation quality and scalability.

Iterative and rollout-based surrogates. One family of methods learns a transition kernel or short-segment predictor and extends trajectories through repeated application. Classical time-series baselines such as ARIMA (Shumway & Stoffer, 2017) model temporal correlations via autoregressive integration, while Wavelet-ARIMA (Kriechbaumer et al., 2014) decomposes the signal into frequency components via wavelet transform before applying ARIMA to each scale independently, providing a rudimentary multi-scale forecast. Among learned surrogates, Timewarp (Klein et al., 2023) trains a normalizing-flow conditional density model $p_\theta(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t)$ that proposes large time-coarsened jumps, deployed inside Metropolis–Hastings MCMC to sample the Boltzmann distribution with transferability to unseen peptides. ITO (Schreiner et al., 2023) learns stochastic transfer operators at multiple time resolutions using equivariant diffusion models. DiffMD (Wu & Li, 2023) uses score-based geometric diffusion conditioned on prior frames and accelerations for next-conformation prediction. Deterministic spatiotemporal predictors such as ESTAG (Wu et al., 2023) and EqMotion (Xu et al., 2023), and more recent direct-prediction models FlashMD (Bigi et al., 2025), MDtrajNet-1 (Ge et al., 2026), and ATOM (Thompson et al., 2025) target rollout cost or long-stride prediction but do not stochastically sample the conformational trajectory distribution. Regardless of the underlying generative mechanism: normalizing flow, diffusion, or deterministic regression, the common limitation is that repeated iterative application accumulates error, causing generated trajectories to drift off the physical manifold and corrupting long-timescale kinetic statistics.

Full-trajectory generative models. A complementary approach generates entire trajectory segments in a single pass rather than extending them step by step. Boltzmann Generators (Noé et al., 2019) pioneered invertible generative models for sampling equilibrium molecular states, though they target static conformations rather than temporally coherent trajectories. Torsional Diffusion (Jing et al., 2022) introduced geometry-aware diffusion over molecular torsion space for conformer generation. Flow matching (Lipman et al., 2022) and stochastic interpolants (Albergo & Vanden-Eijnden, 2023; Albergo et al., 2025) provide the modern continuous-time generative framework that underpins many recent molecular generators, including protein ensemble methods such as AlphaFlow (Jing et al., 2024a) and EquiJump (Costa et al., 2024). For full MD trajectories, MDGen (Jing et al., 2024b) formulates generative modeling of molecular dynamics trajectories and evaluates forward simulation, transition-path sampling, and trajectory upsampling on tetrapeptides and protein monomers. GeoTDM (Han et al., 2024) applies diffusion to geometric trajectory generation, while Dynamics-Diffusion (Petersen et al., 2023) targets rare-event sampling and transition-path inpainting. GF-NODE (Sun et al., 2024) decomposes molecular configurations into spatial frequency modes via the graph Fourier transform and evolves each mode with Neural ODEs, jointly capturing multiple spatial scales dynamics within a single forward pass. However, full-sequence models with transformer-style temporal attention face $\mathcal{O}(T^2)$ memory in the number of frames, while Neural ODE approaches require expensive numerical solvers whose cost scales with trajectory length, making direct generation of long horizons infeasible in either case. In practice, MDGen produces short segments and stitches them via iterative rollouts, reintroducing boundary discontinuities that degrade slow-mode fidelity.

B.2 Machine-Learning Interatomic Potentials

Machine-learning interatomic potentials (MLIPs) replace the expensive potential-energy and force evaluation in MD with a learned surrogate, achieving near-density-functional-theory accuracy at classical force-field cost. The lineage begins with Behler–Parrinello neural network potentials (Behler & Parrinello, 2007), followed by transferable potentials such as ANI (Smith et al., 2017b) and continuous-filter and directional message-passing architectures such as SchNet (Schütt et al., 2018) and DimeNet (Gasteiger et al., 2020). The 2021–2024 wave is dominated by equivariant GNN potentials: PaiNN (Schütt et al., 2021), NequIP (Batzner et al., 2022), MACE (Batatia et al., 2022), and Allegro (Musaelian et al., 2023) exploit E(3)-equivariant representations for high-accuracy force fields, while scalable deployments like SevenNet (Park et al., 2024) and universal foundation potentials like MACE-MP-0 (Batatia et al., 2025), MatterSim (Yang et al., 2024), ORB (Neumann et al., 2024; Rhodes et al., 2025), and eSEN (Fu et al., 2025) extend coverage across chemical space. While MLIPs accelerate per-step force evaluation by orders of magnitude, the resulting trajectories are still produced through step-by-step numerical integration at small timesteps, leaving the long-horizon timescale gap fundamentally unsolved. They are best understood as rollout-based simulation accelerators rather than long-horizon trajectory generators.

B.3 Multi-Scale and Hierarchical Generative Models

The challenge of generating long temporal sequences has been extensively studied outside molecular simulation. In video generation, stochastic latent video models (Denton & Fergus, 2018; Villegas et al., 2018) established that long-horizon generation benefits from latent or structural abstraction, while modern video diffusion approaches (Ho et al., 2022b;a; Singer et al., 2022) employ cascaded coarse-to-fine refinement that separates base generation from temporal and spatial super-resolution. In time-series modeling, multi-resolution diffusion (Shen et al., 2024) and decomposition-guided diffusion (Yuan & Qiao, 2024) explicitly decompose signals into hierarchical temporal components.

For molecular systems, the closest conceptual relatives are coarse-graining and backmapping methods (An & Deshmukh, 2020; Yang & Gómez-Bombarelli, 2023; Heo & Feig, 2024), which reconstruct atomistic detail from coarse representations but focus on spatial rather than temporal resolution. The theoretical foundation for hierarchical temporal modeling in MD comes from the Mori–Zwanzig formalism (Li et al., 2015; Lyu & Lei, 2023), which shows that projecting full atomistic dynamics onto coarse variables introduces memory kernels coupling evolution across temporal neighbors. This formalism justifies cross-scale conditioning but has not previously been instantiated as a recursive generative architecture for long-horizon trajectory synthesis. Our work fills this gap: rather than decomposing spatial resolution or generating independent equilibrium conformations, ReScale uses scale-recursive temporal factorization with cross-scale conditioning to model long-horizon trajectories. Its nested architecture is motivated by fractal structure (Cannon, 1984; Mandelbrot & Van Ness, 1968); the distinction between architectural recurrence and exact mathematical self-similarity is discussed in section A.1.

C Method Details

C.1 Two-Level Instantiation

For $K = 2$, write $T = t_1 t_2$, with cumulative lengths $T_1 = t_1$ and $T_2 = T$. The coarse state $\mathbf{Z}_s^{(1)}$ contains t_1 latent nodes, while the fine state $\mathbf{Z}_s^{(2)}$ contains T nodes organized into t_1 fragments of length t_2. Both are maintained throughout denoising. Their joint endpoint law induces the finest-level marginal

$$p_\omega(\mathbf{Z}_1^{(2)} \mid \mathcal{B}) = \int p_\omega(\mathbf{Z}_1^{(1)}, \mathbf{Z}_1^{(2)} \mid \mathcal{B})\, \mathrm{d}\mathbf{Z}_1^{(1)}, \qquad \widehat{\mathbf{X}} = \mathrm{Dec}(\mathbf{Z}_1^{(2)}, \mathcal{B}). \tag{17}$$

At each denoising step, the coarse state is advanced first. Each fine fragment then advances using its own noisy state and a window of the newly updated noisy coarse state. Both levels reach the final flow time together. ATE supplies physical-time indices before constructing the clean latent targets, as in eq. (5).

Illustrative Attention Accounting. For the documented two-level configuration, $T = 10{,}000$, $t_1 = 1{,}000$, and $t_2 = 10$. With $u = v = 5$, each fine fragment receives $w = 11$ coarse context tokens. Omitting fixed boundary-conditioning tokens and common batch, residue, head, and layer factors, full-horizon temporal attention contains $T^2 = 10^8$ score entries. RESCALE instead contains

$$1{,}000^2 + 1{,}000(10 + 11)^2 = 1.441 \times 10^6 \tag{18}$$

entries across its two levels, approximately 69 times fewer. This calculation quantifies the temporal attention interactions for the specified hierarchy. Peak GPU memory also depends on parameters, noisy states, intermediate activations, fragment batching, and the attention implementation; the measured capacity comparison is reported separately in fig. 3.

C.2 TRAINING PROCEDURE

The clean targets $\mathbf{Y}^{(k)}$ are constructed by the ATE-augmented block projection in eq. (5). At a shared flow time s, each level has a noisy interpolant and a flow-matching target. Finer levels receive noisy parent context at this time, rather than the clean parent target. The final objective sums the per-level flow-matching losses without an auxiliary loss. Training evaluates contexts at the sampled flow time; the parent-updated rule in eq. (8) specifies the numerical sampling procedure.

Algorithm 1 RESCALE Training Procedure

Input: Training trajectory $\mathbf{X}_{1:T}$, boundaries $\mathcal{B}$, factors $t_1, \ldots, t_K$
Append ATE to each frame to obtain $\widehat{\mathbf{X}} \in \mathbb{R}^{T \times L \times (C+A)}$
for $k = 1$ **to** K **do**
    $T_k \leftarrow \prod_{\ell=1}^{k} t_\ell$, $S_k \leftarrow T/T_k$
    Fold $\widehat{\mathbf{X}}$ into T_k blocks, each of shape $L \times S_k(C + A)$
    Project each block with $\mathbf{W}^{(k)}$ to obtain $\mathbf{Y}^{(k)} \in \mathbb{R}^{T_k \times L \times D}$
end for
Sample one flow time $s \sim \mathcal{U}[0, 1]$ and noise tensors $\{\boldsymbol{\epsilon}^{(k)}\}_{k=1}^{K}$
for $k = 1$ **to** K **do**
    $\mathbf{Z}_s^{(k)} \leftarrow (1 - s)\boldsymbol{\epsilon}^{(k)} + s\mathbf{Y}^{(k)}$
    $\mathbf{U}^{(k)} \leftarrow \mathbf{Y}^{(k)} - \boldsymbol{\epsilon}^{(k)}$
end for
$\mathcal{L}_{\text{total}} \leftarrow 0$
for $k = 1$ **to** K **do**
    for each fragment $j = 1, \ldots, T_{k-1}$ **do**
        $\mathcal{C} \leftarrow \mathcal{B}$ if $k = 1$, else $\text{Win}_{k,j}(\mathbf{Z}_s^{(k-1)})$
        $\widehat{\mathbf{V}}_{I_{k,j}}^{(k)} \leftarrow v_{\omega_k}([\mathbf{Z}_s^{(k)}]_{I_{k,j}}, s \mid \mathcal{C})$
    end for
    $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{total}} + \|\widehat{\mathbf{V}}^{(k)} - \mathbf{U}^{(k)}\|^2$
end for
Update the model using $\mathcal{L}_{\text{total}}$

C.3 INFERENCE PROCEDURE

All levels start from noisy states and advance along a shared flow-time grid. Each denoising step contains a coarse-to-fine sweep: update a level, pass its updated noisy state to the next level, and continue until every level has advanced. This repeats until $s = 1$. Algorithm 2 gives the parent-updated Euler form; the documented two-level configuration uses $N = 50$ uniformly spaced steps.

Algorithm 2 RESCALE Coupled Sampling (Parent-Updated Euler)

Input: Boundaries $\mathcal{B}$, hierarchy, time grid $0 = s_0 < \cdots < s_N = 1$
Initialize each $\mathbf{Z}_0^{(k)}$ with Gaussian noise of shape $T_k \times L \times D$
for $n = 0$ **to** $N - 1$ **do**
  $h_n \leftarrow s_{n+1} - s_n$
  for $k = 1$ **to** K **do**
    for each fragment $j = 1, \ldots, T_{k-1}$ **in parallel do**
      $\mathcal{C} \leftarrow \mathcal{B}$ if $k = 1$, else $\mathrm{Win}_{k,j}(\mathbf{Z}_{n+1}^{(k-1)})$
      $\mathbf{V} \leftarrow v_{\omega_k}([\mathbf{Z}_n^{(k)}]_{I_{k,j}}, s_n \mid \mathcal{C})$
      $[\mathbf{Z}_{n+1}^{(k)}]_{I_{k,j}} \leftarrow [\mathbf{Z}_n^{(k)}]_{I_{k,j}} + h_n \mathbf{V}$
    end for
  end for
end for
Return: $\widehat{\mathbf{X}}_{1:T} = \mathrm{Dec}(\mathbf{Z}_N^{(K)}, \mathcal{B})$

The sampler propagates updated noisy states between levels, not predicted velocities or fully denoised parent trajectories. The ordering within each step enforces cross-scale dependencies, while the fragments of a given level can be evaluated in parallel. Physical-time rollouts are unnecessary to cover the target horizon.

D EXPERIMENTAL DETAILS

D.1 DATASET DESCRIPTION

Tetrapeptide (Ala4). We use the tetrapeptide MD trajectories released by Jing et al. (2024b), downloadable from `https://huggingface.co/datasets/bjing-mit/tetrapeptide-sims`. Simulations were performed in OpenMM (Eastman et al., 2017) with the `amber14` force field parameters (Maier et al., 2015). Initial peptide structures were generated with PyMOL, prepared with PDBFixer, and protonated at neutral pH. The released dataset contains both solvent variants described below. Our main benchmark uses the explicit-solvent trajectories for RESCALE and all baselines.

- *Explicit solvent* (3,109 training / 100 validation / 100 test peptides): TIP3P-FB water (Wang et al., 2014) in a 10 Å padded box, neutralized with Na^+/Cl^- counterions; Langevin thermostat at 350 K with friction $\gamma = 0.3$ ps^{-1}; nonbonded cutoff 10 Å with particle-mesh Ewald; production in NPT at 1 bar with a Monte Carlo barostat.
- *Implicit solvent* (2,646 training / 100 validation / 100 test peptides): GB-Neck2 generalized-Born model (Nguyen et al., 2013); Langevin thermostat at 350 K with friction $\gamma = 0.1$ ps^{-1}; production in NVT.

For both variants, hydrogen bonds were constrained, the integration timestep was 2 fs, and each system was energy-minimized with L-BFGS followed by 20 ps NVT equilibration before 100 ns of production. Heavy-atom positions were written every 100 fs, which we subsample to a 10 ps stride, yielding $T = 10{,}000$ frames per 100 ns trajectory.

Fast-folding proteins. We use the 12 fast-folding protein trajectories of Lindorff-Larsen et al. (2011), generated on the Anton supercomputer with the CHARMM22* (Piana et al., 2011) force field and TIP3P explicit water at temperatures near each protein's experimental folding midpoint (290–373 K depending on the system), with a 2.5 fs integration timestep. The 12 proteins span 10-80 residues and cover α-helical, β-sheet, and mixed $\alpha\beta$ folds. The reference trajectories range from 100 μs to 3 ms, saved at 200 ps intervals. We split the 12 proteins as 9/1/2 for train/validation/test and extract non-overlapping 10,000-frame (≈ 2 μs) windows for training and evaluation.

Preprocessing. For both datasets, trajectories are aligned by removing global rotation and translation against a reference frame, and each residue is tokenized into the SE(3)-invariant representation described in Section 2 (per-residue rigid-body offsets to key frames + seven dihedral angles encoded

on the unit circle). Trajectory windows of length $T = 10{,}000$ are sampled uniformly at random during training; for evaluation, windows are taken from a fixed seed.

D.2 Trajectory Representation

Following Jing et al. (2024b), we represent each residue using an SE(3)-invariant internal coordinate system so that the descriptor is independent of global rotations and translations. For each residue j at time-step t, the descriptor $\mathbf{x}_t^j \in \mathbb{R}^C$ comprises two components:

Roto-translational offsets. Let $g_t^j \in \mathrm{SE}(3)$ denote the backbone rigid-body frame of residue j at time t. Given K_{key} conditioning key frames (e.g., $\mathbf{x}_1$ for forward simulation, or $\{\mathbf{x}_1, \mathbf{x}_T\}$ for TPS), each residue frame is expressed as a relative offset:

$$\Delta g_{t,i}^j = \left(g_{t_i}^j\right)^{-1} g_t^j, \quad i = 1, \ldots, K_{\text{key}}.$$

Each offset is parameterized as a unit quaternion (4 dimensions) concatenated with a translation vector (3 dimensions), yielding $7K_{\text{key}}$ dimensions per residue.

Torsion angles. Seven backbone and sidechain dihedral angles $(\psi, \phi, \omega, \chi_1, \chi_2, \chi_3, \chi_4)$ are extracted for each residue. Each angle α is represented on the unit circle as $(\cos\alpha, \sin\alpha)$, contributing 14 dimensions per residue.

The full per-residue descriptor dimension is therefore $C = 7K_{\text{key}} + 14$.

D.3 Implementation Details and Hyperparameters

Architecture. The generators use a common 6-layer Scalable Interpolant Transformer (SiT) architecture, with separate parameters at each hierarchy level. Parameters are shared across fragments within a level. The latent dimension is $D = 384$, aligned with MDGen (Jing et al., 2024b) for a fair comparison. The ATE dimension is $A = 8$. For 2-level ReScale, the local factors are $t_1 = 1{,}000$ and $t_2 = 10$, giving cumulative sequence lengths $T_1 = 1{,}000$ and $T_2 = 10{,}000$. The bidirectional context uses $u = 5$ past and $v = 5$ future neighbors; including the central node, its length is $w = u + v + 1 = 11$. For 3-level ReScale, the local factors are $(t_1, t_2, t_3) = (100, 10, 10)$, so $(T_1, T_2, T_3) = (100, 1{,}000, 10{,}000)$. At each finer level $k > 1$, $u_k = v_k = t_k/2 = 5$, again giving 11 context tokens.

Training. The final ReScale configuration uses ATE and the hierarchical flow-matching objective in algorithm 1, without an auxiliary loss. For the main experiments, we train with the AdamW optimizer using a learning rate of 1×10^{-4} and a batch size of 8 for 400 epochs. All models are trained on 4 NVIDIA A6000 GPUs.

The controlled ablations in sections 3.6 and E.6 use a separate budget of 30 epochs for every compared variant, including MDGen and MDGen with ATE. The NTL9 sampling-interval sweep in section E.4 evaluates both models after 50,000 training steps on identical data. These fixed-budget comparisons are distinct from the main training protocol above.

Inference. For the two-level configuration, we use $N = 50$ uniformly spaced Euler steps on a shared flow-time grid. Within each step, the fine level receives the newly updated noisy coarse state (algorithm 2). The experiments report both ReScale-L2 and ReScale-L3 using the hierarchies specified above; section C.1 provides an illustrative two-level instantiation.

D.4 Evaluation Metrics

Following MDGen (Jing et al., 2024b), we use the Jensen–Shannon *distance*, retaining the abbreviation JSD used in its evaluation. For normalized discrete distributions P and Q,

$$\mathrm{JSD}(P, Q) = \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P \| M) + \frac{1}{2} D_{\mathrm{KL}}(Q \| M)}, \quad M = \frac{P + Q}{2}. \tag{19}$$

The KL terms use natural logarithms, as in MDGen's evaluation implementation. The square root distinguishes this distance from the Jensen–Shannon divergence; lower values indicate closer distributions.

Torsion Distribution JSD. For each individual torsion, we form reference and generated histograms over time using 100 equal-width bins on $[-\pi, \pi]$, normalize them, and compute their JSD. Backbone scores use ϕ and ψ; sidechain scores use the available χ torsions. The backbone, sidechain, and total-torsion metrics summarize these per-angle comparisons.

TICA Population JSD. Time-lagged Independent Component Analysis (TICA) (Pérez-Hernández et al., 2013; Schwantes & Pande, 2013) identifies slow collective coordinates from time-lagged covariances. Following MDGen, we fit TICA with kinetic-map scaling to sine/cosine features of backbone and sidechain torsions from reference MD, then apply the same transformation to generated trajectories. TICA_0 uses a 100-bin histogram along the first component; $\text{TICA}_{0,1}$ uses a 50×50 joint histogram along the first two components. For each comparison, reference and generated histograms share bin edges spanning their combined range. We normalize the counts and compute JSD, flattening the joint histogram into a probability vector. These population scores do not directly measure transition rates or relaxation times.

MSM Occupation Error. We construct a reference Markov state model (MSM) (Prinz et al., 2011; Bowman et al., 2013) using k-means clustering in reference-fitted TICA coordinates and PCCA+ coarse-graining into metastable states, following MDGen. Reference and generated frames use the same state-assignment map. Let $p_{\text{ref},i}$ and $p_{\text{gen},i}$ denote the fractions of evaluated reference and generated frames assigned to state i. The reported occupation error is the JSD between these empirical state-population vectors:

$$\text{MSM occupation error} = \text{JSD}(\mathbf{p}_{\text{ref}}, \mathbf{p}_{\text{gen}}). \tag{20}$$

This metric measures agreement in state occupations and does not by itself establish the accuracy of transition probabilities or rates. The *Oracle* reference is computed by applying the same procedure to an independent held-out MD replica, providing an empirical comparison between finite MD samples.

D.5 Baseline Configuration

Training data and solvent conditions. For the main benchmark, the learned baselines—MDGen, SchNet, EGNN, Timewarp, ITO, ConfRover, TITO, and LaM-SLidE—are retrained from scratch on the same trajectories and data splits as ReScale, using their published architectures and hyperparameters. All generative baselines (e.g., MDGen) are trained for 400 epochs, matching the ReScale training budget in section D.3. All methods use the explicit-solvent TIP3P-FB tetrapeptide data and TIP3P/Anton protein data described in section D.1. In particular, ITO (Schreiner et al., 2023) is retrained on these data rather than evaluated using published weights trained under different solvent conditions. ARIMA and W-ARIMA are fitted to the seed frames of each test trajectory without pretraining. Evaluation uses the same distributional-metric pipeline for all methods. Table 1 reports ConfRover, TITO, and LaM-SLidE results on both datasets.

Sampling and physical duration. MDGen (Jing et al., 2024b) uses a 6-layer SiT architecture with $D = 384$ and generates 1,000-frame segments stitched via iterative rollouts in the main forward-simulation comparison. The varied rollout schedules are specified separately in sections 3.4 and 3.5. Timewarp (Klein et al., 2023) uses the default architecture and step size from its official implementation. For the primary forward-simulation benchmark, all methods target the same physical duration within each dataset. At the model sampling interval, 10,000 frames correspond to a nominal duration of $10{,}000 \times 10$ ps $= 100$ ns for tetrapeptides and $10{,}000 \times 200$ ps $= 2$ μs for fast-folding proteins. These model sampling intervals are distinct from the MD integration timesteps and, where subsampling is applied, the stored-frame intervals. The separate NTL9 sweep uses the sampling intervals specified in section E.4.

Additional generative baselines. ConfRover (Shen et al., 2026) generates conformations autoregressively. TITO (Diez et al., 2026) models transitions conditioned on a time lag. Our TITO evaluation uses the retrained checkpoint, with sampling arguments `num_frames=10000`, `ode_steps=20`,

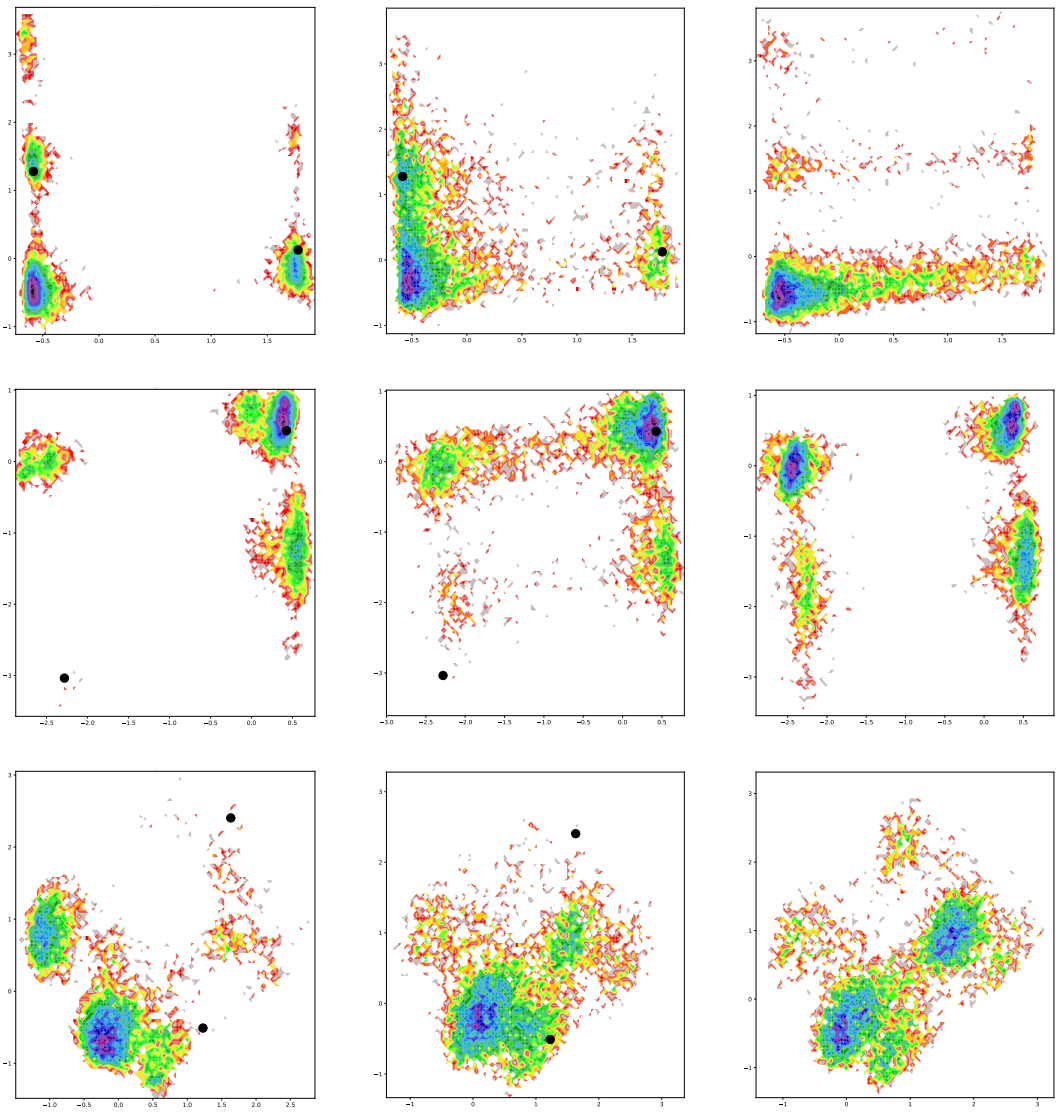

Figure 4: **TICA free energy surface comparisons (backbone and sidechain pair 1).** Columns from left to right: Reference MD, RESCALE (10,000 frames, stride = 1), MDGen (1,000 frames, stride = 10). Rows 1–2: backbone torsion pairs; row 3: sidechain torsion pair 1.

`lag=5.0`, and `chunk=100`. This configuration uses 20 ODE integration steps per generated frame and a lag of five stored frames. Our LaM-SLidE (Sestak et al., 2026) evaluation generates 250-frame blocks per forward pass. The ConfRover tetrapeptide runtime and the TITO and LaM-SLidE runtimes on both datasets were measured on a single NVIDIA L40S GPU and averaged over three seeds.

D.6 CODE

We open-sourced our code with sufficient instructions in our anonymous code repository.

E ADDITIONAL RESULTS

Proteins. On the protein dataset the separation is visible from the very first epochs: RESCALE-L3 sits consistently below the BASELINE on both training and validation throughout the run, and the absolute gap widens as training progresses. The relative improvement is substantially larger than

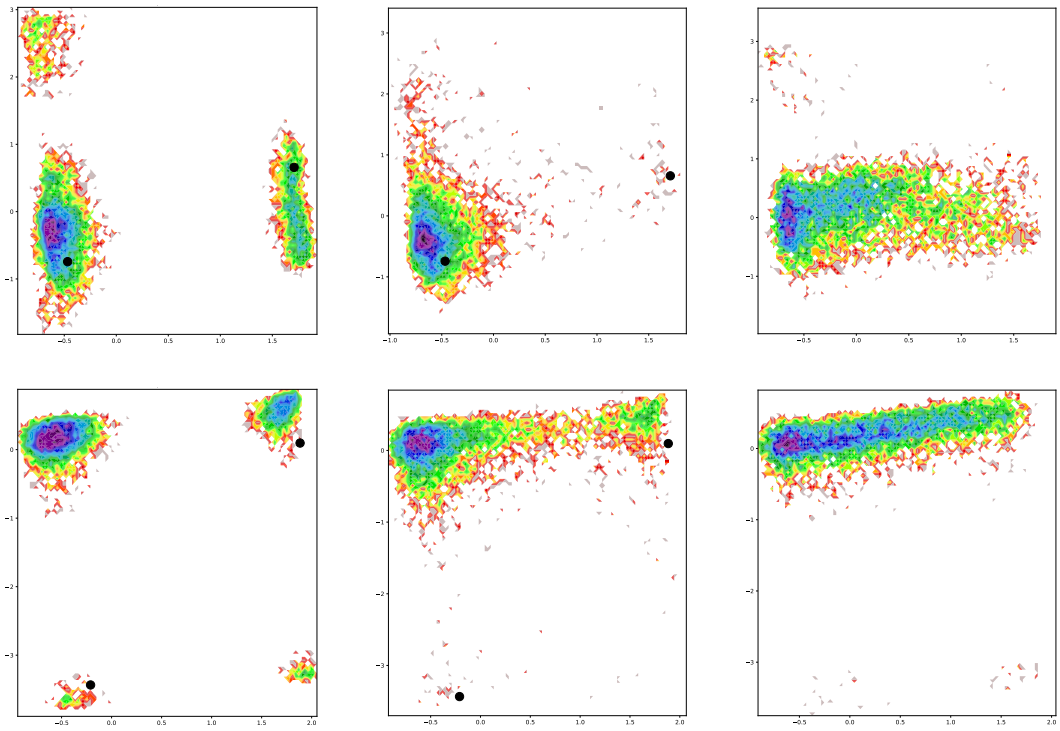

Figure 5: **TICA free energy surface comparisons (sidechain pairs 2–3).** Same column layout as above. The projected landscapes provide a visual comparison of reference and generated conformational distributions.

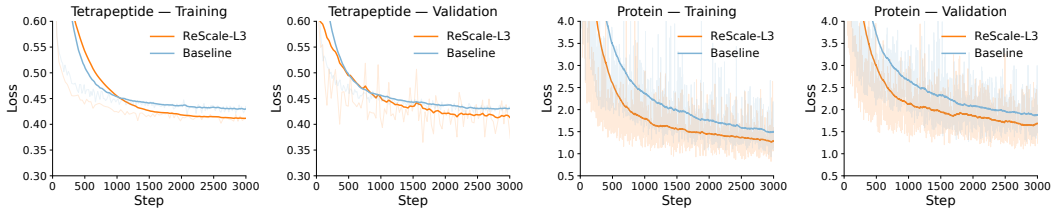

Figure 6: Training and validation loss for RESCALE-L3 versus the baseline on the 4AA tetrapeptide dataset (left two panels, EMA weight 0.9) and the protein dataset (right two panels, EMA weight 0.99). Faint lines show raw values; solid lines are EMA-smoothed. Curves are clipped to 3,000 optimizer steps for legibility.

in the tetrapeptide setting, suggesting that the coarse–fine factorization in RESCALE-L3 pays off most when the per-frame token sequence is long enough that the BASELINE's flat attention pattern becomes a bottleneck.

E.1 TICA FREE ENERGY SURFACE COMPARISONS

fig. 4 and fig. 5 show free energy surfaces projected onto the top two TICA components computed from backbone and sidechain torsion angles, following the protocol of Jing et al. (2024b). Each row corresponds to a different torsion angle pair; columns compare Reference MD, RESCALE (10,000 frames, stride $= 1$), and MDGen (1,000 frames, stride $= 10$). These plots provide a qualitative comparison of the projected conformational distributions through their free energy landscapes.

E.2 TRAINING DYNAMIC COMPARISONS

We compare RESCALE-L3 against the BASELINE on tetrapeptide and protein training (Fig. 6), plotting EMA-smoothed training and validation losses against gradient-step count. The two settings are matched in optimizer, batch size, and effective sequence length per step; differences in the loss curves are therefore attributable to the architecture rather than to the optimization recipe.

Tetrapeptides. On the 4AA corpus the two models track each other closely during the first $\sim 1{,}000$ steps, after which RESCALE-L3 pulls ahead on both the training and validation curves and continues to descend while the baseline flattens. By step $3{,}000$ RESCALE-L3 reaches a training loss of roughly 0.41 versus 0.43 for the baseline, with an essentially identical gap on validation. The validation curve closely shadows the training curve for both models, indicating that the additional capacity exposed by the fractal factorization is absorbed without overfitting on this data scale.

E.3 ATLAS FORWARD-SIMULATION RESULTS

Table 5 reports all six distributional metrics for the ATLAS forward-simulation comparison summarized in section 3.3. RESCALE achieves the lowest JSD for backbone, sidechain, and total torsions and for the marginal and joint TICA populations. MDGen achieves the lowest MSM occupation error. These metrics evaluate the generated distributions rather than transition rates or other kinetic quantities.

Table 5: **Full ATLAS forward-simulation results.** Torsional and TICA population JSD and MSM occupation error; lower is better. Best values are bold.

Method	Torsions (JSD) ↓			TICA populations (JSD) ↓		MSM
	Backbone	Sidechain	Total	TICA_0	$\text{TICA}_{0,1}$	occ. error ↓
ARIMA	0.808	0.807	0.808	0.809	0.830	0.561
W-ARIMA	0.652	0.698	0.675	0.764	0.827	0.449
SchNet	0.769	0.762	0.766	0.655	0.805	0.376
EGNN	0.702	0.673	0.687	0.573	0.754	0.303
Timewarp	0.707	0.662	0.685	0.576	0.760	0.291
MDGen	0.355	0.373	0.364	0.632	0.796	**0.287**
RESCALE	**0.330**	**0.306**	**0.318**	**0.519**	**0.600**	0.288

E.4 NTL9 AT VARYING SAMPLING INTERVALS

We evaluate NTL9 across nominal trajectory durations from 10 μs to 1 ms, keeping the sequence length fixed at 10,000 frames and increasing the sampling interval. Both models are evaluated after 50,000 training steps on identical data. RESCALE achieves lower backbone torsion and joint TICA population JSD than MDGen at all four intervals (Table 6). At 20 μs, these errors are 0.365 and 0.352, compared with 0.504 and 0.417 for MDGen. Errors remain above the MD-replica reference values, particularly at longer horizons. This experiment evaluates longer physical durations through coarser temporal sampling at a fixed frame count.

Evaluation protocol. For this sweep, we fix the output length to $N = 10{,}000$ frames and use sampling intervals $\Delta t \in \{1, 2, 20, 100\}$ ns. The nominal duration $H = N\Delta t$ is therefore 10 μs, 20 μs, 200 μs, or 1 ms; the elapsed time between the first and last frames is $(N-1)\Delta t$. Generated distributions are evaluated against a held-out reference MD replica. The MD-replica comparison reports the corresponding discrepancy between MD replicas at each nominal duration, providing an empirical reference for the distributional metrics. These metrics do not establish the accuracy of folding rates or other rare-event kinetics.

Table 6: **NTL9 at varying sampling intervals.** Each sequence contains 10,000 frames. Nominal duration increases by coarsening the sampling interval Δt. Backbone torsion and joint TICA population JSD are reported; lower is better. Among the learned models, best values are bold and second-best values are underlined. MD ref. denotes the MD-replica comparison.

Nominal duration	Δt (ns)	Backbone torsion JSD ↓			$\text{TICA}_{0,1}$ population JSD ↓		
		MD ref.	MDGen	**RESCALE**	MD ref.	MDGen	**RESCALE**
10 μs	1	0.085	<u>0.462</u>	**0.389**	0.229	<u>0.394</u>	**0.367**
20 μs	2	0.114	<u>0.504</u>	**0.365**	0.337	<u>0.417</u>	**0.352**
200 μs	20	0.108	<u>0.544</u>	**0.509**	0.231	<u>0.630</u>	**0.629**
1 ms	100	0.050	<u>0.638</u>	**0.608**	0.170	<u>0.665</u>	**0.650**

E.5 MD-REPLICA REFERENCE FOR FORWARD SIMULATION

Independent MD replicas provide the *Oracle* scores in Table 7, measuring variation between finite MD samples of the same system and duration.

Table 7: **MD-replica reference (Oracle) for Table 1.** Simulation times are in hours.

Dataset	Torsions (JSD) ↓			TICA (JSD) ↓		MSM ↓	Time
	Backbone	Sidechain	Total	TICA_0	$\text{TICA}_{0,1}$	occ. error	(h)
Tetrapeptide	0.103	0.055	0.076	0.201	0.268	0.208	3
Protein	0.043	0.038	0.040	0.089	0.159	0.068	43.2

E.6 Generator Parameter Sharing Across Levels

We compare separate generator parameters at each hierarchy level with two weight-sharing variants under the same 30-epoch training and evaluation protocol as Table 4. The standard ReScale-L3 configuration uses 432M parameters. Sharing generator weights across levels reduces the parameter count to 171M. A second shared variant triples the generator depth to match the 432M parameter count. All three configurations retain ATE and bidirectional conditioning and use no auxiliary loss.

Table 8: **Generator parameter-sharing ablation on tetrapeptides.** Torsional and TICA population JSD and MSM occupation error; lower is better. Best values are bold.

Generator parameters	Params	Torsion JSD ↓			TICA JSD ↓		MSM ↓
		Backbone	Sidechain	Total	$TICA_0$	$TICA_{0,1}$	occ. error
Per-level (Ours)	432M	**0.177**	**0.111**	**0.145**	**0.266**	**0.351**	**0.277**
Shared	171M	0.233	0.120	0.177	0.286	0.378	0.287
Shared (3× depth)	432M	0.270	0.136	0.204	0.307	0.402	0.292

Per-level weights achieve lower error on all six metrics than either shared variant. Increasing the shared generator's depth to restore the parameter count does not close this gap. These results favor level-specific generator parameters under the evaluated training budget.