Learning Implicit Bias in Generative Spaces for Accelerating Protein Dynamics Emulation

Kaihui Cheng[1,2] **Zhiqiang Cai**[1] **Wenkai Xiang**[1] **Zhihang Hu**[1]

Siyu Zhu[1,2,3,†] **Tzuhsiung Yang**[1,†] **Yuan Qi**[1,2,†]

[1]**Shanghai Academy of AI for Science** [2]**Fudan University** [3]**Shanghai Innovation Institute**

[†]Corresponding authors

siyuzhu@fudan.edu.cn yangzixiong@sais.org.cn qiyuan@fudan.edu.cn

Abstract

Generative emulators of protein dynamics produce plausible trajectories at a fraction of the cost of molecular dynamics, but they inherit their training distribution and tend to revisit known states rather than reach rare ones under long-horizon extrapolation. Inspired by classical enhanced sampling, we introduce an implicit, history-dependent bias in the generative space of a pretrained emulator. Specifically, a history-aware score estimator augments the frozen emulator with a distance-weighted bias that steers reverse-time sampling away from previously generated structures, regularized by an environment-support term. To preserve structural validity at long horizons, a score-based refinement step re-projects drifted samples onto the data manifold using the frozen emulator. Our experiments demonstrate that the method (i) raises diversity by 35% on DynamicPDB-80; (ii) on 12 zero-shot Fast-Folding proteins, the learned bias alone reaches the unbiased emulator's coverage up to $\sim 15\times$ faster, and pairing it with refinement reaches the coverage up to $\sim 37\times$ faster while covering $\sim 3\times$ as many low-energy states.

1 Introduction

Generative emulators of protein dynamics reproduce short trajectories at a fraction of the cost of molecular dynamics (MD), yet their long-horizon rollouts tend to revisit known states rather than reach rare ones. These rare states underlie biological processes including catalysis, transport, and folding [12, 7], but sit behind high free-energy barriers that direct MD rarely crosses on routine timescales [13, 19]. Classical enhanced-sampling methods, including umbrella sampling [37], metadynamics [20, 4], and replica exchange [36], bias MD toward underexplored regions, but still incur substantial per-system simulation cost. On the generative side, score-based diffusion [14, 32, 33, 34] extended with $SE(3)$-equivariant architectures [42, 6, 41] has driven trajectory-level emulators that generate protein trajectories from sequence and structure conditioning [39, 1, 17, 8, 11]. To encourage exploration in such emulators, recent work injects guidance through collective variables (CVs) or exploration objectives into the reverse process [26, 40, 29].

To accelerate long-horizon exploration, we introduce a history-dependent bias directly in the generative space of a pretrained protein dynamics emulator, leaving the emulator itself frozen. A history-aware score estimator augments the frozen emulator's reverse-time score with a bias conditioned on the rollout history, such that the same noisy state yields different biased reverse scores under different histories and the bias adapts as the history bank updates, analogously to history-dependent biasing in molecular dynamics. During training, the biased reverse step is regularized

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

by an environment-support term that keeps the score on the emulator's data manifold. At inference, a complementary score-based refinement step re-projects samples back onto this manifold through the frozen emulator, correcting drift that accumulates over long-horizon rollouts. The left panel of Figure 1 illustrates the bias steering reverse-time sampling away from previously visited conformations.

Our main contributions are summarized as follows. Both components operate on a pretrained protein dynamics emulator that remains frozen throughout:

- We propose a history-dependent bias in the generative space of a pretrained protein dynamics emulator. A history-aware score estimator augments the reverse-time score with a distance-weighted bias conditioned on past rollout frames, regularized during training by an environment-support term that keeps the biased score on the emulator's data manifold.
- We introduce a score-based refinement step that re-projects samples onto the emulator's data manifold through a short forward-then-reverse diffusion. Applied at inference, it sustains structural validity under strong bias and on horizons well beyond the training timescale.
- We empirically validate the method at both short- and long-horizon scales: on DynamicPDB-80, the bias raises diversity (mDiv-mean) by 35%; on 12 zero-shot Fast-Folding proteins, the bias alone reaches the unbiased emulator's coverage up to $\sim 15\times$ faster, and pairing it with refinement reaches the coverage up to $\sim 37\times$ faster while covering $\sim 3\times$ as many low-energy states.

2 Related Work

2.1 Generative Modeling of Protein Structure and Conformational Ensembles

Generative modeling of protein structure has progressed from single-state prediction to richer conformational distributions, yet the targets remain state-level rather than trajectory-level. Deep learning methods such as AlphaFold2 [18], RoseTTAFold [3], and ESMFold [22] predict protein structures from sequence with high accuracy. Beyond deterministic prediction, generative approaches sample over conformational distributions [2, 25, 16, 43], often parameterized by score-based diffusion [32, 34] on rigid-frame and $SE(3)$-equivariant representations [42, 41, 6]. However, these methods generate conformations independently rather than as time-ordered trajectories.

2.2 Generative Modeling of Protein Dynamics

Generative emulators of protein dynamics fill this gap by learning the transitions between successive conformations directly from simulation trajectories. Among score-based trajectory models, DiffMD [39] estimates score fields over molecular conformations; MDGen [17] models protein motions relative to the input structure; DynaFold [11] operates in a latent trajectory space; and AlphaFolding [8] predicts future conformations from a reference structure and past frames. Force-field learners take a different route: DFF [1] learns a coarse-grained force field without force supervision, with a follow-up [27] enforcing consistency with Langevin dynamics. While these methods improve short-horizon fidelity, repeated rollouts may concentrate in visited basins, reaching new conformations only after many extrapolation steps. Rather than training a new emulator, we freeze the pretrained one and learn a history-dependent bias in the generative space to accelerate long-horizon coverage.

2.3 Enhanced Sampling and Controlled Generative Dynamics

Enhanced sampling has long sought to bias exploration toward underrepresented regions. Classical approaches bias exploration through different mechanisms: umbrella sampling [37] along CVs, replica exchange [36] across parallel temperatures, and metadynamics [20, 4], which builds a history-dependent bias from past visits. Controlled diffusion adopts a similar mechanism in generative modeling, guiding reverse-time sampling by modifying the score or drift [32, 9, 15]. ConfDiff [38] biases the frozen score with a force network learned from MD forces and energies, CV-guided diffusion [26] steers reverse sampling along a chosen reaction coordinate, and recent diffusion-based formulations of enhanced sampling [40, 29] embed exploration objectives into the reverse process. We instead build the bias from the rollout history itself, lifting the history-dependent biasing principle

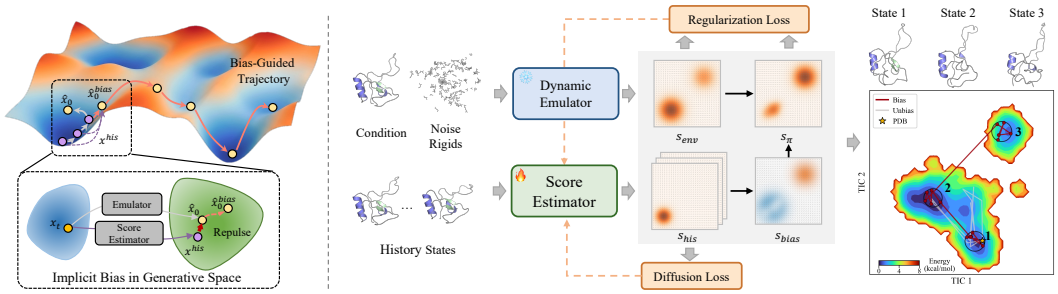


Figure 1: **Illustration of the proposed framework. Left:** concept. A history-aware score estimator adds a history-dependent bias to the reverse score; the generated structure $\hat{x}_0^{\text{bias}}$ is shifted away from the trajectory history x^{his}. **Middle:** architecture. A frozen emulator produces the reverse score s_{emu}; the estimator produces history-aware scores s_{his}, aggregated into a bias s_{bias} that is added to s_{emu} to form the biased reverse score s_π. Color overlays on the score fields encode the bias direction: dark red marks regions of higher biased density, while dark blue marks regions suppressed (penalized by the history). Training combines a diffusion loss with an environment-support regularizer. **Right:** the biased rollout (red) visits three metastable states on the TICA free-energy surface, while the unbiased rollout (gray) stays near the starting structure (yellow star).

of metadynamics into the generative space of a frozen $SE(3)$ emulator, with an environment-support term preventing the learned bias from drifting off the emulator's manifold during training.

3 Preliminaries

3.1 Problem Formulation

We formulate long-horizon protein dynamics generation as an autoregressive trajectory problem. For clarity of exposition, we adopt a single-frame extrapolation setting in which each step generates one new structure conditioned on past frames; repeating this for K steps produces a trajectory $\tau = (\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^{K-1})$ with $\mathbf{x}^k \in SE(3)^N$, where k indexes frames along the trajectory and N is the number of residues. Each step conditions on a residue sequence feature $\mathbf{c}$ shared across all frames and on a history bank $\mathcal{H} = \{\mathbf{h}^0, \mathbf{h}^1, \ldots, \mathbf{h}^{H-1}\}$ of H previously generated clean structures. The frame $\mathbf{x}^k$ is sampled by a reverse diffusion process evolving from $t = 1$ to $t = 0$, with $\mathbf{x}_t^k$ denoting the noisy state at diffusion time t and $\mathbf{x}_1^k$ drawn from the diffusion prior [42]. The frozen emulator parameterizes this process through a score $\mathbf{s}_{\text{emu}}^\theta$ and a one-step denoising kernel $\mathbb{K}_{\text{emu}}(\mathbf{x}_{t-\Delta t}^k \mid \mathbf{x}_t^k, \mathbf{s}_{\text{emu}}^\theta)$; our goal is to learn a history-aware score estimator that biases this reverse process using the rollout history.

3.2 Score-Based Diffusion on $SE(3)$

We build the emulator on the $SE(3)$ diffusion framework of [42]. Following AlphaFold2 [18], each residue is represented by a rigid frame in $SE(3)$, and a protein backbone of N residues lies in $SE(3)^N$. We decompose the noisy state as $\mathbf{x}_t = (\boldsymbol{\omega}_t, \boldsymbol{\nu}_t)$, where $\boldsymbol{\omega}_t \in \mathfrak{so}(3)$ is the rotation vector parameterizing an element of $SO(3)$ and $\boldsymbol{\nu}_t \in \mathbb{R}^3$ is the translation. For the translational component on $\mathbb{R}^3$, the forward perturbation kernel is Gaussian, $q(\boldsymbol{\nu}_t \mid \boldsymbol{\nu}_0) = \mathcal{N}(\boldsymbol{\nu}_t; \alpha_t \boldsymbol{\nu}_0, \sigma_t^2 \mathbf{I})$, with score $\mathbf{s}_{\text{trans}}(\boldsymbol{\nu}_t, t; \boldsymbol{\nu}_0) = \nabla_{\boldsymbol{\nu}_t} \log q(\boldsymbol{\nu}_t \mid \boldsymbol{\nu}_0)$. For the rotational component on $SO(3)$, the forward perturbation kernel is the IGSO(3) density $f_t(\boldsymbol{\omega}_t \mid \boldsymbol{\omega}_0)$ conditional on the clean rotation $\boldsymbol{\omega}_0$ [42], with score $\mathbf{s}_{\text{rot}}(\boldsymbol{\omega}_t, t; \boldsymbol{\omega}_0) = \nabla_{\boldsymbol{\omega}_t} \log f_t(\boldsymbol{\omega}_t \mid \boldsymbol{\omega}_0)$.

The emulator predicts a denoised rigid structure $\hat{\mathbf{x}}_0 = (\hat{\boldsymbol{\omega}}_0, \hat{\boldsymbol{\nu}}_0)$ from $\mathbf{x}_t$, which we substitute as the clean state in the per-component score formulas above. The combined reverse-step score is $\mathbf{s}(\mathbf{x}_t, t) = (\mathbf{s}_{\text{rot}}(\boldsymbol{\omega}_t, t; \hat{\boldsymbol{\omega}}_0),\ \mathbf{s}_{\text{trans}}(\boldsymbol{\nu}_t, t; \hat{\boldsymbol{\nu}}_0))$, and we denote by $\mathbb{K}(\mathbf{x}_{t-\Delta t} \mid \mathbf{x}_t, \mathbf{s})$ the one-step reverse transition kernel obtained by an Euler–Maruyama update under score field $\mathbf{s}$.

4 Methodology

As illustrated in Figure 1, our method keeps the emulator introduced in Section 3 frozen and trains a history-aware score estimator that augments the emulator's reverse score with a history-dependent bias. Because the bias is conditioned on the rollout history $\mathcal{H}$, the same noisy state $\mathbf{x}_t$ yields different biased scores under different histories, and the bias adapts as the history bank updates. For long-horizon rollouts, a complementary score-based refinement step re-projects drifted samples onto the data manifold using the same frozen emulator score, sustaining per-frame structural validity. More details on the derivation are deferred to Appendix C.

4.1 Implicit Bias in Generative Space

Although the frozen emulator's reverse score $\mathbf{s}^{\theta}_{\mathrm{emu}}$ can faithfully reproduce its training trajectories, those trajectories rarely sample the equilibrium distribution: crossing the high free-energy barriers behind rare states requires timescales beyond what direct MD can routinely access [13, 19]. As a result, rollouts driven by this score tend to revisit known states rather than reach rare ones, limiting long-horizon exploration. To address this, we learn a history-aware score estimator that augments $\mathbf{s}^{\theta}_{\mathrm{emu}}$ with a history-dependent bias, yielding a biased reverse score conditioned on the history $\mathcal{H}$.

History-dependent bias score. Let ϕ parameterize the history-aware score estimator. The estimator has H parallel branches, one per structure in the history bank. Given the current noisy state $\mathbf{x}^k_t$, diffusion time t, sequence feature $\mathbf{c}$, and history $\mathcal{H}$, branch j outputs a score $\mathbf{s}^{\phi}_{\mathrm{his},j}(\mathbf{x}^k_t, t, \mathcal{H}, \mathbf{c})$ trained (Section 4.2) to denoise $\mathbf{x}^k_t$ toward the history structure $\mathbf{h}^j \in \mathcal{H}$. The bias aggregates these branch scores as a weighted sum of differences from the emulator score,

$$\mathbf{s}^{\phi}_{\mathrm{bias}}\big(\mathbf{x}^k_t, t, \mathcal{H}, \mathbf{c}\big) = \sum_{j=0}^{H-1} w_j \left(\mathbf{s}^{\theta}_{\mathrm{emu}}(\mathbf{x}^k_t, t) - \mathbf{s}^{\phi}_{\mathrm{his},j}(\mathbf{x}^k_t, t, \mathcal{H}, \mathbf{c})\right). \tag{1}$$

By construction, $\mathbf{s}^{\phi}_{\mathrm{bias}}$ points away from directions that denoise toward previously visited conformations. The biased reverse score is the emulator score augmented with $\mathbf{s}^{\phi}_{\mathrm{bias}}$ at strength $\eta \geq 0$,

$$\mathbf{s}^{\phi}_{\pi}\big(\mathbf{x}^k_t, t, \mathcal{H}, \mathbf{c}\big) = \mathbf{s}^{\theta}_{\mathrm{emu}}(\mathbf{x}^k_t, t) + \eta\, \mathbf{s}^{\phi}_{\mathrm{bias}}(\mathbf{x}^k_t, t, \mathcal{H}, \mathbf{c}). \tag{2}$$

Branch weights. The weights $\{w_j\}$ concentrate the bias on branches whose denoised prediction is closest to the emulator's. We parameterize them as a softmax over Kabsch-aligned Cα root-mean-square deviation (RMSD),

$$w_j = \frac{\exp(-d_j/\kappa)}{\sum_{l=0}^{H-1} \exp(-d_l/\kappa)}, \tag{3}$$

where d_j is the Cα RMSD between the emulator's and branch j's denoised structures (both Kabsch-aligned), and $\kappa > 0$ is the softmax temperature.

Sampling with the biased score. At inference, the estimator enters only through $\mathbf{s}^{\phi}_{\pi}$: we apply the same Euler–Maruyama reverse update as the unbiased emulator, with $\mathbf{s}^{\phi}_{\pi}$ in place of $\mathbf{s}^{\theta}_{\mathrm{emu}}$,

$$\mathbb{K}_{\pi}\big(\mathbf{x}^k_{t-\Delta t} \,\big|\, \mathbf{x}^k_t, \mathcal{H}, t\big) = \mathbb{K}\big(\mathbf{x}^k_{t-\Delta t} \,\big|\, \mathbf{x}^k_t, \mathbf{s}^{\phi}_{\pi}(\mathbf{x}^k_t, t, \mathcal{H}, \mathbf{c})\big)\,, \tag{4}$$

preserving the parametric form of the frozen reverse dynamics.

4.2 Loss Function

We train the history-aware score estimator with two complementary objectives: a per-branch diffusion loss that supervises each branch $\mathbf{s}^{\phi}_{\mathrm{his},j}$ to denoise toward its history target $\mathbf{h}^j$, and an environment-support regularizer that prevents the biased reverse kernel $\mathbb{K}_{\pi}$ from drifting off the frozen emulator's manifold.

Per-branch diffusion loss. Each branch $\mathbf{s}^{\phi}_{\mathrm{his},j}$ is trained to match the reverse-time score that denoises the current noisy state toward its history structure $\mathbf{h}^j$. Given a clean training structure $\mathbf{x}^k$ and a sampled diffusion time t, we perturb $\mathbf{x}^k$ forward into $\mathbf{x}^k_t$. For each $\mathbf{h}^j \in \mathcal{H}$, the target $\mathbf{s}_{\mathrm{target}}(\mathbf{x}^k_t, \mathbf{h}^j, t)$ is the closed-form reverse score that denoises $\mathbf{x}^k_t$ toward $\mathbf{h}^j$, obtained by substituting $\mathbf{h}^j$ as the clean state in the per-component score formulas of Section 3. The loss is the per-branch denoising score-matching objective,

$$\mathcal{L}_{\mathrm{diff}} = \mathbb{E}_{t,\,\mathbf{x}^k_t}\left[\frac{1}{H}\sum_{j=0}^{H-1}\gamma(t)\left\|\mathbf{s}^{\phi}_{\mathrm{his},j}(\mathbf{x}^k_t, t, \mathcal{H}, \mathbf{c}) - \mathbf{s}_{\mathrm{target}}(\mathbf{x}^k_t, \mathbf{h}^j, t)\right\|^2\right], \tag{5}$$

where $\gamma(t)$ is the time-dependent loss weight from [42].

Environment-support regularizer. The per-branch diffusion loss trains each branch individually but does not constrain the resulting biased reverse kernel. At inference, Eq. (4) evaluates the frozen reverse kernel at the biased score, so the biased score must propose next states where $\mathbf{s}^{\theta}_{\mathrm{emu}}$ remains reliable; otherwise the kernel sees out-of-distribution states and sample quality degrades. To prevent this, we regularize the expected KL divergence between the biased and frozen reverse kernels:

$$\mathcal{L}_{\mathrm{support}} = \mathbb{E}_{k,t}\left[\mathrm{KL}\left(\mathbb{K}_{\pi}(\cdot \mid \mathbf{x}^k_t, \mathcal{H}, t)\,\middle\|\,\mathbb{K}_{\mathrm{emu}}(\cdot \mid \mathbf{x}^k_t, \mathbf{s}^{\theta}_{\mathrm{emu}}(\mathbf{x}^k_t, t))\right)\right]. \tag{6}$$

Both kernels share the Gaussian transition form of Eq. (4) and differ only in their reverse-drift score, so the KL admits an unbiased single-sample Monte Carlo estimator: we draw a proposal $\tilde{\mathbf{x}}^k_{t-\Delta t} \sim \mathbb{K}_{\pi}(\cdot \mid \mathbf{x}^k_t, \mathcal{H}, t)$ and evaluate the log-density ratio

$$\mathcal{L}^k_{\mathrm{support}} = \log \mathbb{K}_{\pi}\big(\tilde{\mathbf{x}}^k_{t-\Delta t} \,\big|\, \mathbf{x}^k_t, \mathcal{H}, t\big) - \log \mathbb{K}_{\mathrm{emu}}\big(\tilde{\mathbf{x}}^k_{t-\Delta t} \,\big|\, \mathbf{x}^k_t, \mathbf{s}^{\theta}_{\mathrm{emu}}(\mathbf{x}^k_t, t)\big), \tag{7}$$

averaging $\mathcal{L}_{\mathrm{support}} \approx \mathbb{E}_{k,t}[\mathcal{L}^k_{\mathrm{support}}]$ over training frames and diffusion times.

The total loss is

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{diff}} + \lambda_{\mathrm{support}}\,\mathcal{L}_{\mathrm{support}}, \tag{8}$$

where $\lambda_{\mathrm{support}} \geq 0$ balances the two objectives: $\mathcal{L}_{\mathrm{diff}}$ shapes each branch's score toward its history target $\mathbf{h}^j$, while $\mathcal{L}_{\mathrm{support}}$ keeps the biased score on the frozen emulator's manifold during training.

4.3 Refinement for Long-Horizon Rollouts

Long-horizon biased rollouts gradually drift off the data manifold, degrading per-frame structural validity. Prior work mitigates this drift by augmenting the emulator's conditioning structures with noise during training [31]; we instead repair drifted frames at inference time, without fine-tuning the emulator, by reusing the frozen emulator score $\mathbf{s}^{\theta}_{\mathrm{emu}}$ as a refinement projection. Each generated frame $\mathbf{x}^{\mathrm{gen}}$ is forward-diffused to a small noise level t_{ref} and then denoised by S reverse steps under $\mathbf{s}^{\theta}_{\mathrm{emu}}$: $\hat{\mathbf{x}}^{\mathrm{gen}} \sim \mathbb{K}^S(\cdot \mid q(\mathbf{x}_{t_{\mathrm{ref}}} \mid \mathbf{x}^{\mathrm{gen}}), \mathbf{s}^{\theta}_{\mathrm{emu}})$. The refinement runs as a post-processing step over generated frames, leaving the rollout history $\mathcal{H}$ and the learned bias unchanged (Algorithm 2, Appendix A.3).

5 Experiments

5.1 Experimental Setup

Dataset. We construct training and evaluation sets from DynamicPDB [24], a corpus of over $5{,}300$ protein trajectories curated from the Protein Data Bank (PDB) [5] and simulated with OpenMM [10] under the Amber ff14SB force field for 100 ns each. To prevent information leakage, we cluster all proteins at 40% sequence identity with MMseqs2 [35] and split the clusters $80{:}20$, yielding $4{,}315$ training trajectories for emulator pretraining. For the score estimator, conditioning on a history bank of H structures multiplies the per-frame memory footprint, so we restrict training to the $3{,}578$ trajectories with at most 224 residues. For evaluation, we randomly sample 80 proteins of up to 284 residues from the held-out clusters, hereafter **DynamicPDB-80**; this budget keeps full-trajectory rollout and TICA reconstruction tractable across all evaluated methods. For long-horizon extrapolation beyond the training distribution, we additionally evaluate on 12 **Fast-Folding** proteins simulated on the Anton supercomputer by D. E. Shaw Research [23]; these proteins are held out from DynamicPDB and evaluated zero-shot, with no per-protein fine-tuning.

Table 1: **Structural quality and diversity on DynamicPDB-80.** Metric definitions follow Section 5.1; Div@τ is diversity over frames passing the compliance threshold τ (τ=0 recovers the raw diversity over all frames), and mDiv-mean averages Div@τ over $\tau \in [0.90, 0.99]$. Values are means over 80 proteins. Shaded rows add our implicit bias on top of a frozen baseline.

Method		Structural validity			Diversity			
		CA% ↑	CN% ↑	Clash-Free ↑	Div@.00 ↑	Div@.90 ↑	Div@.95 ↑	mDiv-mean ↑
MD (Oracle)		100.00%	100.00%	100.00%	0.135	0.135	0.135	0.135
Conformation Generation	BioEMU [21]	99.99%	99.97%	99.99%	0.234	0.234	0.234	0.234
	AlphaFlow [16]	100.00%	100.00%	99.99%	0.122	0.122	0.122	0.122
	Str2Str [25]	100.00%	100.00%	100.00%	0.192	0.192	0.192	0.192
Trajectory Generation	MDGEN [17]	84.52%	68.51%	97.57%	0.353	0.193	0.171	0.170
	ConfRover [30]	97.47%	96.89%	99.26%	0.182	0.182	0.181	0.179
	+Implicit Bias	96.91%	96.34%	99.37%	0.221	**0.221**	**0.222**	**0.212**
	Emulator	99.82%	98.43%	99.51%	0.185	0.185	0.184	0.181
	+Implicit Bias	98.37%	95.82%	98.87%	0.313	**0.289**	**0.252**	**0.245**

Training. The emulator is trained from scratch for 700 epochs on 8 NVIDIA H800 GPUs with the joint SE(3) diffusion objective, then frozen. The score estimator is trained for 300 epochs on the same hardware, conditioning on a history bank of H=3 past structures, with branch-weight temperature κ=1.0 and support-regularization weight λ_{support}=0.5. As a transferability test across emulator backbones, we train an additional estimator on top of the frozen pretrained ConfRover [30] under the same λ_{support}. Full architecture and optimization details are in Appendix A.1.

Inference. At rollout time, we sample the initial noisy backbone from the $SE(3)$ diffusion prior and condition on the PDB starting structure and the residue-sequence embedding from Evoformer [18]. Each generated frame uses 100 reverse steps, and each rollout consists of E autoregressive extrapolation segments of 16 frames each, with E varying by benchmark. Unless otherwise noted, we use bias strength η=0.05 on DynamicPDB-80 and η=0.1 on Fast-Folding. Refinement (Section 4.3) uses t_{ref}=0.2 and S=20 steps. Full rollout settings and baseline configurations are in Appendix A.2. The inference algorithms are provided in Appendix A.3.

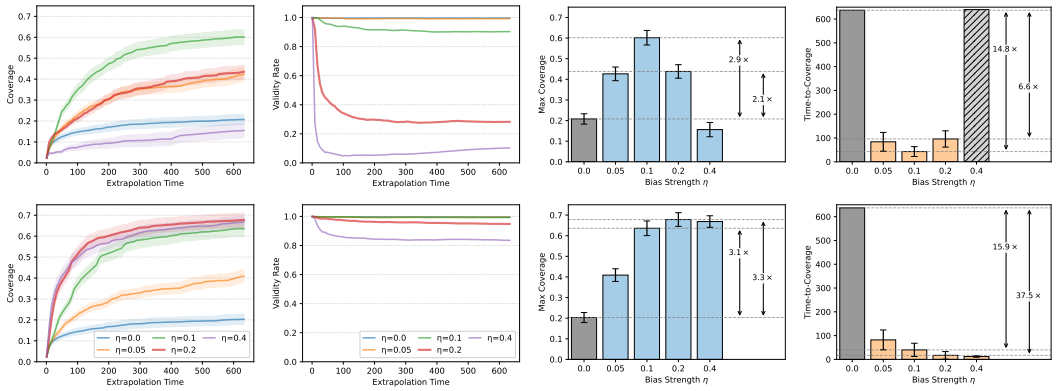
Evaluation. We evaluate in two timescale regimes: a short-timescale regime on DynamicPDB-80 with E=16 extrapolation segments, and a long-timescale regime on Fast-Folding with $E \geq 640$. Refinement is applied only in the long-timescale regime.

Structural validity. For each generated frame, we compute three validity rates and average them over the trajectory. **Cα-Level Validity (CA%)** is the fraction of adjacent Cα–Cα pairs whose distance is below 4.5 Å; **Peptide-Bond Validity (CN%)** is the fraction of adjacent peptide-bond C–N pairs whose distance is below 2.0 Å; and **Steric Validity (Clash-Free)** is the fraction of non-neighboring atom pairs separated by more than 1.0 Å.

Coverage and diversity. All metrics are computed against the MD reference in a two-dimensional TICA space. A frame is compliant at threshold τ when its CA, CN, and Clash-Free rates all satisfy $\geq \tau$. **Ensemble Diversity (Div)** is the mean pairwise $1 - \text{lDDT}_{\text{C}\alpha}$ over the generated ensemble. To trace the validity–diversity trade-off, we restrict Div to compliant frames, denoted $\text{Div}@\tau$, and define **mDiv-mean** as the average of $\text{Div}@\tau$ over $\tau \in [0.90, 0.99]$ on a 10-point grid. **Low-Energy State Coverage (Cov)** [21] is the fraction of low-energy states (TICA regions with free energy below 4 kcal/mol) visited by τ=0.9-compliant frames. To isolate the exploration gain from the bias, we treat each architecture's η=0 rollout as its own reference and report **Time-to-Coverage (TTC)**: the smallest frame count at which the biased coverage first exceeds the η=0 coverage at the full frame budget F. **Speedup** is $\text{TTC}_{\eta=0}/\text{TTC}_{\eta}$, the factor by which the bias reaches the reference coverage faster. Cov and TTC require an MD reference long enough to visit all metastable states; we therefore report them only on the Fast-Folding benchmark (Section 5.3).

5.2 Short-Timescale Trajectory Emulation

We first verify that the bias preserves per-frame structural validity at modest rollout depth, a precondition for any longer-horizon claim. Following Section 5.1, every method generates F=256 frames per



(a) Coverage (b) Validity Rate (c) Max Coverage (d) Time-to-Coverage

Figure 2: **Refinement extends the usable range of bias strength η, recovering validity and low-energy state coverage that raw rollouts lose with long extrapolation.** Each panel stacks raw rollouts (top) above the same trajectories after refinement at t_{ref}=0.2 (bottom): (a) coverage vs. extrapolation depth E, (b) cumulative validity rate, (c) max coverage vs. η, (d) TTC vs. η, with hatched bars marking η where TTC is never reached within the E=640 budget. Curves and bars show mean ±SEM across the 12 Fast-Folding proteins, each averaged over 3 seeds.

protein on DynamicPDB-80. Baselines fall into two classes: IID conformation generation, which samples frames independently, and trajectory generation, which propagates frames autoregressively. Table 1 reports per-frame structural validity and diversity at increasing compliance thresholds τ.

Adding the bias improves diversity within each trajectory architecture while preserving structural validity. On our frozen emulator, the bias raises Div@.00 from 0.185 to 0.313 with a CA drop of 1.45 percentage points, and the gains persist at stricter thresholds (mDiv-mean from 0.181 to 0.245). The same mechanism transfers to ConfRover [30]: adding the bias raises Div@.00 from 0.182 to 0.221 with CA dropping only 0.56 percentage points (from 97.47% to 96.91%), confirming that the bias is not specific to our emulator backbone.

Among trajectory baselines, MDGEN reaches the highest raw diversity (0.353) but at low validity (CA 84.5%, CN 68.5%), reflecting cumulative error in its autoregressive rollouts; once a compliance threshold is applied, its mDiv-mean drops to 0.170. IID conformation generators are flat across τ by construction, since every frame is sampled independently.

5.3 Long-Timescale Trajectory Extrapolation

We extend rollouts to microsecond horizons, far beyond the 100 ns training trajectories, and test whether the proposed method still improves conformational coverage without losing per-frame structural validity. The 12 Fast-Folding proteins from D. E. Shaw Research [23] are held out from the DynamicPDB training set; we evaluate zero-shot, using a single checkpoint across all 12 systems rather than fine-tuning a separate model per protein as in leave-one-out cross-validation. We sweep bias strengths $\eta \in \{0, 0.05, 0.1, 0.2, 0.4\}$ across 3 seeds, with each rollout spanning E=640 autoregressive extrapolation segments, totaling 10,240 frames per protein. For raw biased rollouts, speedup peaks at η=0.1 and decays at higher η as per-frame validity degrades, as shown in the top rows of Figure 2: η=0.1 yields 14.8× over the η=0 baseline, η=0.2 yields 6.6×, and η=0.4 fails to reach baseline coverage within the E=640 budget.

To recover this lost speedup, we apply the refinement step of Section 4.3 to the full biased rollouts. At η=0.2, refinement restores per-frame validity at τ=0.9 from 28.3% to 94.8%, raises max coverage to 3.3× the η=0 baseline, and pushes speedup to 37.5×, as shown in the bottom rows of Figure 2. As a control, refining the unbiased η=0 rollout yields max coverage 0.203 versus 0.208 for the unrefined baseline, isolating the exploration gain to the bias. The full η vs. refinement grid is in Appendix B. Figure 3 shows the biased rollout reaching metastable basins missed by the unbiased baseline on three representative proteins, raising low-energy state coverage from 0.29 to 0.88 on Protein G, 0.44 to 1.00 on WW domain, and 0.46 to 0.73 on α3D.

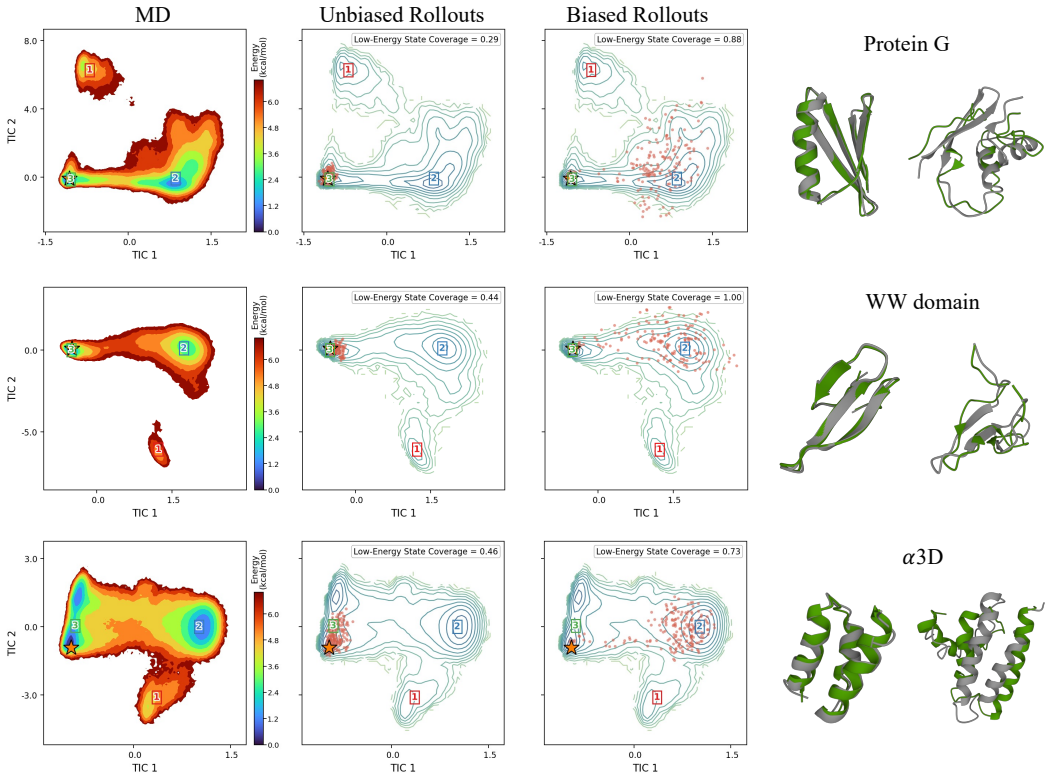


Figure 3: **The bias expands low-energy state coverage on three representative proteins.** Column 1: MD reference free-energy surface with metastable basins labeled 1–3. Column 2: unbiased rollouts. Column 3: biased rollouts. Column 4: 3D structure overlay between MD reference (gray) and a representative biased frame (green). Red dots are TICA projections of generated frames; yellow star marks the input PDB structure.

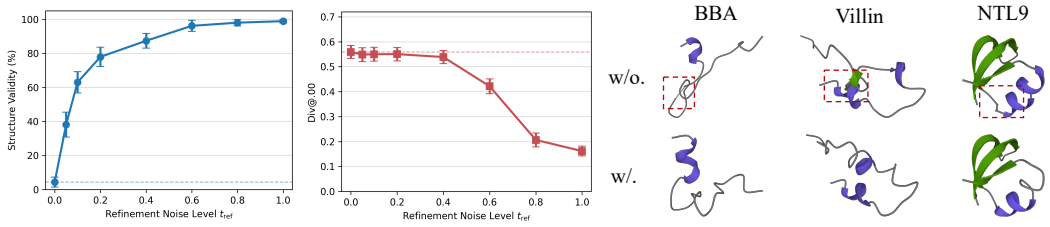


Figure 4: **Refinement noise level t_{ref} controls validity recovery while preserving diversity. Left**: structure validity at τ=0.9 rises sharply with t_{ref} from the pre-refinement baseline (dashed). **Middle**: diversity (Div@.00) stays near the pre-refinement baseline (dashed) at low t_{ref}, then drops as forward noising approaches the prior. **Right**: structure overlays on representative proteins, before (top) and after (bottom) refinement; broken local geometries (red boxes) are repaired.

5.4 Ablation Studies

Refinement Noise Level. We sweep t_{ref} from 0.0 to 1.0 on the 50 worst-validity segments per protein at η=0.2, where refinement is most needed. Figure 4 shows that the per-frame pass-rate at τ=0.9 rises sharply with t_{ref} and saturates near 0.2, climbing from 4.4% to 77.9% while preserving diversity; the right panels show that refinement repairs broken local geometries while keeping structures plausible. We adopt t_{ref}=0.2 at this inflection, where validity has recovered while the forward marginal remains well short of the prior. At small η, refinement also does not degrade coverage (Appendix B).

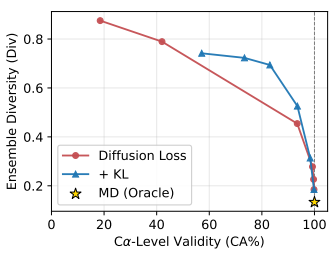


Figure 5: $\mathcal{L}_{\text{support}}$ **preserves validity across the η-sweep on DynamicPDB-80.** Validity–diversity curves trained with and without $\mathcal{L}_{\text{support}}$. Yellow star: MD oracle.

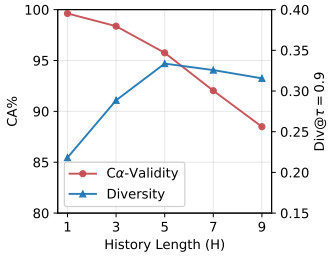


Figure 6: **History length H trades structural fidelity for diversity on DynamicPDB-80.** Increasing H raises diversity while Cα validity decreases.

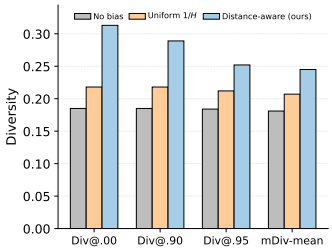


Figure 7: **Weighting Strategies on DynamicPDB-80.** Distance-aware weighting outperforms uniform $1/H$ aggregation across compliance thresholds τ.

Loss Function and Bias Strength. We sweep bias strength $\eta \in \{0.05, \ldots, 1.0\}$ with and without the environment-support regularizer $\mathcal{L}_{\text{support}}$. Figure 5 shows that without $\mathcal{L}_{\text{support}}$, C$\alpha$ validity drops sharply at high η, whereas adding $\mathcal{L}_{\text{support}}$ preserves structural validity across the entire sweep and shifts the curve toward the upper-right, where both validity and diversity are higher.

History Length. We sweep the history length $H \in \{1, 3, 5, 7, 9\}$ to characterize the validity–diversity trade-off. Figure 6 shows that diversity rises with H and saturates while Cα validity drops. We adopt H=3, where diversity is near saturation and validity is still high.

Weighting Strategies. We compare our distance-aware weighting (Eq. 3) against a uniform w_j=$1/H$ aggregation at η=0.05, H=3, using the η=0 rollout as the no-bias reference. Figure 7 shows that uniform aggregation barely improves over the no-bias reference, whereas distance-aware weighting consistently captures most of the diversity gain across compliance thresholds.

6 Discussion

Limitations. Our experiments have validated history-dependent generative-space biasing for accelerated exploration of pretrained protein dynamics emulators. The current bias is unconditional: it explores the free-energy surface but cannot condition on a specific target endpoint. The reachable conformational space is bounded by the frozen emulator's training distribution: the bias redistributes density within this support, not beyond it. Our method accelerates conformational exploration rather than guaranteeing unbiased Boltzmann equilibrium sampling (Appendix D).

Opportunities. Beyond protein dynamics, history-dependent biasing in generative space may apply to long-horizon sampling in other pretrained physical-system generators. Goal-conditioned biasing, in which the score estimator is augmented with target-conformation inputs, would lift the unconditional limitation above by converting exploration bias into a controllable sampler for targeted conformational change. The framework can extend to multi-chain complexes and nucleic acid dynamics with appropriately retrained emulators. Uncertainty-aware sampling could prioritize the bias toward conformations where the emulator's score is most uncertain, allocating compute to under-explored regions of the free-energy surface. Finally, integrating the post-hoc reweighting into a training-time objective could close the loop between biased rollouts and free-energy estimates.

Broader impacts. Accelerated exploration of protein conformational landscapes supports structural biology and drug discovery, including cryptic-pocket detection and allosteric-state characterization. The released models inherit the dual-use profile of public protein-structure generators and introduce no new high-risk capabilities; we release them under a permissive license without additional safeguards.

Acknowledgments and Disclosure of Funding

Funding. This work was supported by the Natural Science Foundation of Shanghai (Grant No. 24ZR1407200), the Shanghai Oriental Talents Project (Grant No. QNKJ2024060), and the Shanghai Municipal Commission of Economy and Informatization (Grant No. 2025-GZL-RGZN-BTBX-01011). We also thank the Shanghai Academy of AI for Science (SAIS) for its support.

Competing interests. The authors declare no competing interests.

References

[1] Marloes Arts, Victor Garcia Satorras, Chin-Wei Huang, Daniel Zugner, Marco Federici, Cecilia Clementi, Frank Noé, Robert Pinsler, and Rianne van den Berg. Two for one: Diffusion models and force fields for coarse-grained molecular dynamics. Journal of Chemical Theory and Computation, 19(18):6151–6159, 2023.

[2] Martina Audagnotto, Werngard Czechtizky, Leonardo De Maria, Helena Käck, Garegin Papoian, Lars Tornberg, Christian Tyrchan, and Johan Ulander. Machine learning/molecular dynamic protein structure prediction approach to investigate the protein conformational ensemble. Scientific Reports, 12(1):10018, 2022.

[3] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. Science, 373(6557):871–876, 2021.

[4] Alessandro Barducci, Giovanni Bussi, and Michele Parrinello. Well-tempered metadynamics: a smoothly converging and tunable free-energy method. Physical review letters, 100(2):020603, 2008.

[5] Helen M Berman, Tammy Battistuz, Talapady N Bhat, Wolfgang F Bluhm, Philip E Bourne, Kyle Burkhardt, Zukang Feng, Gary L Gilliland, Lisa Iype, Shri Jain, et al. The protein data bank. Biological Crystallography, 58(6):899–907, 2002.

[6] Avishek Joey Bose, Tara Akhound-Sadegh, Guillaume Huguet, Kilian Fatras, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. SE(3)-Stochastic Flow Matching for Protein Backbone Generation, April 2024.

[7] Zimei Bu and David JE Callaway. Proteins move! protein dynamics and long-range allostery in cell signaling. Advances in protein chemistry and structural biology, 83:163–221, 2011.

[8] Kaihui Cheng, Ce Liu, Qingkun Su, Jun Wang, Liwei Zhang, Yining Tang, Yao Yao, Siyu Zhu, and Yuan Qi. 4d diffusion for dynamic protein structure prediction with reference and motion guidance. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 93–101, 2025.

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In Advances in neural information processing systems, volume 34, pages 8780–8794, 2021.

[10] 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, Rafal P. Wiewiora, Bernard R. Brooks, and Vijay S. Pande. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. PLoS Comput. Biol., 13(7):1–17, 07 2017.

[11] Zirui Fan, Junjie Zhu, and Hai-Feng Chen. Dynafold: A latent diffusion based generative framework for protein dynamic trajectory. bioRxiv, pages 2025–09, 2025.

[12] Jingjing Guo and Huan-Xiang Zhou. Protein allostery and conformational dynamics. Chemical reviews, 116(11):6503–6515, 2016.

[13] Jérôme Hénin, Tony Lelièvre, Michael R. Shirts, Omar Valsson, and Lucie Delemotte. Enhanced sampling methods for molecular dynamics simulations. Living Journal of Computational Molecular Science, 4(1):1583, December 2022.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Advances in neural information processing systems, volume 33, pages 6840–6851, 2020.

[15] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598, 2022.

[16] Bowen Jing, Bonnie Berger, and Tommi Jaakkola. AlphaFold Meets Flow Matching for Generating Protein Ensembles, September 2024.

[17] Bowen Jing, Hannes Stärk, Tommi Jaakkola, and Bonnie Berger. Generative Modeling of Molecular Dynamics Trajectories, September 2024.

[18] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873):583–589, August 2021.

[19] Jarek Juraszek, Jocelyne Vreede, and Peter G. Bolhuis. Transition path sampling of protein conformational changes. Chemical Physics, 396:30–44, March 2012.

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

[21] Sarah Lewis, Tim Hempel, José Jiménez-Luna, Michael Gastegger, Yu Xie, Andrew YK Foong, Victor García Satorras, Osama Abdin, Bastiaan S Veeling, Iryna Zaporozhets, et al. Scalable emulation of protein equilibrium ensembles with generative deep learning. Science, 389(6761):eadv9817, 2025.

[22] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. BioRxiv, 2022:500902, 2022.

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

[24] Ce Liu, Jun Wang, Zhiqiang Cai, Yingxu Wang, Huizhen Kuang, Kaihui Cheng, Liwei Zhang, Qingkun Su, Yining Tang, Fenglei Cao, et al. Dynamic pdb: a new dataset and a se (3) model extension by integrating dynamic behaviors and physical properties in protein structures. arXiv preprint arXiv:2408.12413, 2024.

[25] Jiarui Lu, Bozitao Zhong, Zuobai Zhang, and Jian Tang. Str2Str: A Score-based Framework for Zero-shot Protein Conformation Sampling. In International Conference on Learning Representations, 2024.

[26] Juno Nam, Bálint Máté, Artur P Toshev, Manasa Kaniselvan, Rafael Gómez-Bombarelli, Ricky TQ Chen, Brandon Wood, Guan-Horng Liu, and Benjamin Kurt Miller. Enhancing diffusion-based sampling with molecular collective variables. arXiv preprint arXiv:2510.11923, 2025.

[27] Michael Plainer, Hao Wu, Leon Klein, Stephan Günnemann, and Frank Noé. Consistent sampling and simulation: Molecular dynamics with energy-based diffusion models. arXiv preprint arXiv:2506.17139, 2025.

[28] Sanjeev Raja, Martin Šípka, Michael Psenka, Tobias Kreiman, Michal Pavelka, and Aditi S. Krishnapriyan. Action-Minimization Meets Generative Modeling: Efficient Transition Path Sampling with the Onsager-Machlup Functional, June 2025.

[29] Daniel D Richman, Jessica Karaguesian, Carl-Mikael Suomivuori, and Ron O Dror. Unlocking hidden biomolecular conformational landscapes in diffusion models at inference time. ArXiv, pages arXiv–2512, 2026.

[30] Yuning Shen, Lihao Wang, Huizhuo Yuan, Yan Wang, Bangji Yang, and Quanquan Gu. Simultaneous modeling of protein conformation and dynamics via autoregression. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

[31] Nima Shoghi, Yuxuan Liu, Yuning Shen, Rob Brekelmans, Pan Li, and Quanquan Gu. Scalable spatio-temporal se (3) diffusion for long-horizon protein dynamics. arXiv preprint arXiv:2602.02128, 2026.

[32] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In International Conference on Learning Representations, 2021.

[33] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Advances in neural information processing systems, volume 32, 2019.

[34] 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, 2021.

[35] Martin Steinegger and Johannes Söding. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nat. Biotechnol., 35(11):1026–1028, 2017.

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

[37] Glenn M Torrie and John P Valleau. Nonphysical sampling distributions in monte carlo free-energy estimation: Umbrella sampling. Journal of computational physics, 23(2):187–199, 1977.

[38] Yan Wang, Lihao Wang, Yuning Shen, Yiqun Wang, Huizhuo Yuan, Yue Wu, and Quanquan Gu. Protein Conformation Generation via Force-Guided SE(3) Diffusion Models, September 2024.

[39] Fang Wu and Stan Z Li. DIFFMD: A geometric diffusion model for molecular dynamics simulations. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 5321–5329, 2023.

[40] Yu Xie, Ludwig Winkler, Lixin Sun, Sarah Lewis, Adam E. Foster, José Jiménez Luna, Tim Hempel, Michael Gastegger, Yaoyi Chen, Iryna Zaporozhets, Cecilia Clementi, Christopher M. Bishop, and Frank Noé. Enhanced Diffusion Sampling: Efficient Rare Event Sampling and Free Energy Calculation with Diffusion Models, February 2026.

[41] Jason Yim, Andrew Campbell, Andrew Y. K. Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S. Veeling, Regina Barzilay, Tommi Jaakkola, and Frank Noé. Fast protein backbone generation with SE(3) flow matching, October 2023.

[42] 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. In International Conference on Machine Learning, pages 40001–40039. PMLR, 2023.

[43] Shuxin Zheng, Jiyan He, Chang Liu, Yu Shi, Ziheng Lu, Weitao Feng, Fusong Ju, Jiaxi Wang, Jianwei Zhu, Yaosen Min, He Zhang, Shidi Tang, Hongxia Hao, Peiran Jin, Chi Chen, Frank Noé, Haiguang Liu, and Tie-Yan Liu. Predicting equilibrium distributions for molecular systems with deep learning. Nature Machine Intelligence, 6(5):558–567, May 2024.

A Implementation Details

A.1 Training

Dataset simulation protocols and preprocessing. Complementing the dataset overview in Section 5.1, the 100 ns all-atom DynamicPDB trajectories [24] are simulated in OpenMM [10] under the Amber ff14SB force field with explicit TIP3P water and periodic boundary conditions, whereas the 12 held-out Fast-Folding trajectories from D. E. Shaw Research [23] are simulated on Anton using a modified CHARMM force field with explicit TIP3P solvent. Prior to training, all frames within each trajectory are rigid-body aligned to the initial frame via optimal Kabsch superimposition on Cα atoms. Following the training protocol of AlphaFolding [8], each 16-frame training window is subsampled from the underlying MD trajectory at a physical interval of 0.4 ns between adjacent frames (corresponding to the stride of 40 below and spanning 6.4 ns per segment).

Emulator backbone. The emulator is a joint $SE(3)$ frame diffusion network with sequence and trajectory-context conditioning. Single-residue and pair tracks have node embedding dimension 384 and edge embedding dimension 128, with 4 Invariant Point Attention (IPA) blocks [18]; each IPA block uses 8 heads, 8 query/key points, 12 value points, and hidden dimension 256. Amino-acid type and residue-index features are projected to 32-dim embeddings before the trunk.

Emulator training. We train from scratch on the DynamicPDB training partition with batch size 4 per GPU on 8 NVIDIA H800 GPUs, giving 32 samples per optimizer step. Optimization uses Adam with AMSGrad at learning rate 10^{-4} without warmup or a learning-rate schedule. The objective is a sum of translation, rotation, backbone-atom, distance-matrix, and torsion losses with weights 1.0, plus an auxiliary loss with weight 0.25, following the joint $SE(3)$ formulation of [42]. Each training window contains 16 frames, with the underlying MD trajectories subsampled at stride 40. We train the emulator for 700 epochs. The pretrained emulator is then frozen for the rest of the pipeline.

Score estimator. The history-aware score estimator adopts the same 4-block IPA architecture and hyperparameters as the emulator, but is instantiated with separate parameters and trained independently. It conditions on a history bank of $H{=}3$ structures via a single shared trunk, where each history frame is folded into the pair track through the same self-conditioning pathway used by the emulator.

Score estimator training. We train the estimator on the same DynamicPDB training partition with batch size 1, frame length 16, history bank $H{=}3$, trajectory-segment stride 40, and maximum residue length 224. Optimization uses AdamW at learning rate 10^{-4} with weight decay 10^{-6} for 300 epochs on 8 NVIDIA H800 GPUs in fully synchronous data-parallel mode in full precision (FP32). The emulator backbone is frozen, so only the estimator weights receive gradients. The objective combines the per-branch diffusion loss of Eq. (5) with the environment-support regularizer of Eq. (6) at weight $\lambda_{\text{support}}{=}0.5$, evaluated by the single-sample MC estimator of Eq. (7). The transferability variant on the frozen pretrained ConfRover [30] backbone uses the same implementation.

A.2 Inference and Baselines

Rollout. At inference time we sample the initial noisy backbone from the joint $SE(3)$ diffusion prior used during training: translations follow a VP-SDE schedule with $\beta_{\min}{=}0.1$, $\beta_{\max}{=}20.0$ and a coordinate scaling factor of 0.1, while rotations follow an IGSO(3) schedule with $\sigma \in [0.1,\ 1.5]$ on a logarithmic grid of 1000 points in both ω and σ. Each generated frame uses $T{=}100$ uniform Euler–Maruyama reverse steps from $t{=}1$ to $t_{\min}{=}0.01$, and each rollout produces E autoregressive extrapolation segments of frame length 16. Sequence conditioning uses MSA-based Evoformer [18] embeddings, feeding the same 384-dim node and 128-dim pair tracks used at training. The bias strength is $\eta{=}0.05$ on DynamicPDB-80 and $\eta{=}0.1$ on Fast-Folding. The branch weighting uses the Kabsch-aligned RMSD with temperature $\kappa{=}1.0$. Refinement (Section 4.3) applies a partial forward noising to $t_{\text{ref}}{=}0.2$ followed by $S{=}20$ reverse denoising steps under the frozen emulator ($\eta = 0$).

Extrapolation rounds versus reverse diffusion steps. To distinguish the two temporal axes in the rollout settings above, E indexes sequential autoregressive extrapolation rounds along the physical trajectory, whereas T indexes the reverse-time denoising steps executed within each round. In each extrapolation round, the model jointly denoises a block of 16 consecutive trajectory frames from the $SE(3)$ prior over T reverse diffusion steps, after which the terminal frame of the generated 16-frame segment serves as the structural conditioning reference for the next round. Consequently, a rollout depth of $E{=}16$ on DynamicPDB-80 yields $16 \times 16 = 256$ trajectory frames (spanning 102.4 ns at the 0.4 ns frame interval, i.e., the $\sim$100 ns horizon below).

Baselines. For protein ensemble generation, we evaluate three baselines from publicly released pretrained checkpoints, each producing 256 ensemble structures:

- **AlphaFlow** [16]: template conditioning enabled, 10 denoising steps (released default).
- **BioEmu** [21]: v1.1, with the unphysical filter disabled.
- **Str2Str** [25]: forward-noise level $\Delta t{=}0.2$.

For trajectory generation, we evaluate the pretrained ConfRover-Base [30] model and retrain MDGEN [17] on our DynamicPDB training set. Our base emulator follows the AlphaFolding [8] architecture, with its OmegaFold sequence embedding replaced by Evoformer. All trajectory-generation methods produce rollouts of approximately 100 ns at a 0.4 ns sampling interval, yielding 256 frames per trajectory. In the Fast-Folding evaluation (Appendix D), we additionally include the force-guided conformation generator **ConfDiff** [38] and the latent trajectory diffusion model **DynaFold** [11].

A.3 Inference Algorithms

Algorithms 1 and 2 detail the inference-time procedures for history-biased trajectory sampling (Section 4.1) and post-hoc manifold refinement (Section 4.3). Following the notation of Section 3, $\mathbf{x}_t^k \in SE(3)^N$ denotes the noisy state of trajectory frame k at diffusion time $t \in [0, 1]$, $\mathbf{c}$ denotes the fixed sequence embedding extracted by Evoformer, $\mathcal{H} = \{\mathbf{h}^j\}$ denotes the history bank of previously generated clean structures, and $\tau = (\mathbf{x}^0, \ldots, \mathbf{x}^{K-1})$ is the resulting K-frame trajectory. For clarity of exposition, Algorithm 1 is written in the single-frame autoregressive notation of the main text;

in our implementation, each step k jointly denoises a 16-frame segment, and both the structural reference $\mathbf{x}^{\mathrm{ref}}$ and history bank $\mathcal{H}$ are updated at segment boundaries.

Score computation and reverse transition kernel. In Algorithm 1, EMULATOR and HISTORYESTIMATOR denote forward passes through the frozen emulator θ and the trained history-aware estimator ϕ, respectively. Each network predicts a denoised structure $\hat{\mathbf{x}}_0 = (\hat{\boldsymbol{\omega}}_0, \hat{\boldsymbol{\nu}}_0)$ from the current noisy state $\mathbf{x}_t = (\boldsymbol{\omega}_t, \boldsymbol{\nu}_t)$. The corresponding $SE(3)$ score is computed as in Section 3:

$$\mathbf{s}(\mathbf{x}_t, t; \hat{\mathbf{x}}_0) = \left(\nabla_{\boldsymbol{\omega}_t} \log f_t(\boldsymbol{\omega}_t \mid \hat{\boldsymbol{\omega}}_0),\ -\frac{\boldsymbol{\nu}_t - \alpha_t \hat{\boldsymbol{\nu}}_0}{\sigma_t^2} \right). \tag{9}$$

We use the subscripts emu and his, j on $\hat{\mathbf{x}}_0$ to distinguish the clean predictions of the frozen emulator and history branch j, and attach the superscript k to the branch distances d_j^k and softmax weights w_j^k to make the rollout index explicit. Here, $\mathrm{RMSD}_{\mathrm{C}\alpha}^{\mathrm{Kabsch}}$ denotes the Cα root-mean-square deviation after optimal Kabsch alignment. The reverse transition kernel $\mathbb{K}$ performs one Euler–Maruyama step on $SO(3)^N \times \mathbb{R}^{3N}$ at diffusion time t_n with step size Δt_n. In the loop, we write $\mathbf{s}_{\mathrm{emu}}^{\theta,k} = \mathbf{s}_{\mathrm{emu}}^{\theta}(\mathbf{x}_{t_n}^k, t_n)$ and $\mathbf{s}_{\mathrm{his},j}^{\phi,k} = \mathbf{s}_{\mathrm{his},j}^{\phi}(\mathbf{x}_{t_n}^k, t_n, \mathcal{H}, \mathbf{c})$, similarly abbreviating $\mathbf{s}_{\mathrm{bias}}^{\phi,k}$ and $\mathbf{s}_{\pi}^{\phi,k}$.

Algorithm 1 Biased Trajectory Sampling

Require: Frozen emulator θ and history estimator ϕ; sequence feature $\mathbf{c}$; initial structure $\mathbf{x}^{\mathrm{ref}}$; initial bank $\mathcal{H}_0$
Require: Rollout length K; bank capacity H; reverse schedule $[t_0, \ldots, t_{T-1}]$; bias strength $\eta \geq 0$; temperature $\kappa > 0$
Ensure: Generated trajectory $\tau = (\mathbf{x}^0, \ldots, \mathbf{x}^{K-1})$

$\tau \leftarrow ()$; $\mathcal{H} \leftarrow \mathcal{H}_0$
for $k = 0, \ldots, K-1$ **do**
  $\mathbf{x}_{t_0}^k \sim p_{\mathrm{prior}}$ ▷ sample from the $SE(3)$ diffusion prior
  for $n = 0, \ldots, T-2$ **do**
    $\Delta t_n \leftarrow t_n - t_{n+1}$
    $(\hat{\mathbf{x}}_{0,\mathrm{emu}}^k, \mathbf{s}_{\mathrm{emu}}^{\theta,k}) \leftarrow$ EMULATOR$(\mathbf{x}_{t_n}^k, t_n, \mathbf{x}^{\mathrm{ref}}, \mathbf{c})$
    for $j = 0, \ldots, |\mathcal{H}| - 1$ **do**
      $(\hat{\mathbf{x}}_{0,\mathrm{his},j}^k, \mathbf{s}_{\mathrm{his},j}^{\phi,k}) \leftarrow$ HISTORYESTIMATOR$(\mathbf{x}_{t_n}^k, t_n, \mathcal{H}, \mathbf{c}, j)$
      $d_j^k \leftarrow \mathrm{RMSD}_{\mathrm{C}\alpha}^{\mathrm{Kabsch}}(\hat{\mathbf{x}}_{0,\mathrm{emu}}^k, \hat{\mathbf{x}}_{0,\mathrm{his},j}^k)$
    end for
    $w_j^k \leftarrow \exp(-d_j^k/\kappa) \big/ \sum_{l=0}^{|\mathcal{H}|-1} \exp(-d_l^k/\kappa)$
    $\mathbf{s}_{\mathrm{bias}}^{\phi,k} \leftarrow \sum_{j=0}^{|\mathcal{H}|-1} w_j^k \big[\mathbf{s}_{\mathrm{emu}}^{\theta,k} - \mathbf{s}_{\mathrm{his},j}^{\phi,k}\big]$
    $\mathbf{s}_{\pi}^{\phi,k} \leftarrow \mathbf{s}_{\mathrm{emu}}^{\theta,k} + \eta \mathbf{s}_{\mathrm{bias}}^{\phi,k}$
    $\mathbf{x}_{t_{n+1}}^k \sim \mathbb{K}\big(\cdot \mid \mathbf{x}_{t_n}^k, \mathbf{s}_{\pi}^{\phi,k}\big)$ ▷ Euler–Maruyama step of size Δt_n
  end for
  $(\hat{\mathbf{x}}_{0,\mathrm{emu}}^k, \cdot) \leftarrow$ EMULATOR$(\mathbf{x}_{t_{T-1}}^k, t_{T-1}, \mathbf{x}^{\mathrm{ref}}, \mathbf{c})$
  $\mathbf{x}^k \leftarrow \hat{\mathbf{x}}_{0,\mathrm{emu}}^k$ ▷ terminal clean prediction at $t_{\min}$
  $\tau \leftarrow \tau \,\|\, (\mathbf{x}^k)$ ▷ append to trajectory
  $\mathcal{H} \leftarrow$ UPDATE$(\mathcal{H}, \mathbf{x}^k, H)$
  $\mathbf{x}^{\mathrm{ref}} \leftarrow \mathbf{x}^k$
end for
return τ

Discretization schedules and history management. Biased sampling (Algorithm 1) discretizes the reverse interval onto a decreasing grid $1 = t_0 > \cdots > t_{T-1} = t_{\min} > 0$ with step sizes $\Delta t_n = t_n - t_{n+1}$ for $n < T-1$, and returns the emulator's clean prediction $\hat{\mathbf{x}}_{0,\mathrm{emu}}^k$ at the terminal time $t_{\min}$. $\mathbf{x}^{\mathrm{ref}}$ tracks the most recent conditioning structure independently of the static sequence feature $\mathbf{c}$. Given an initial bank $\mathcal{H}_0$, the procedure UPDATE maintains a sliding history bank of maximum capacity H; during initial warm-up rounds where $|\mathcal{H}| < H$, branch aggregation runs over the $|\mathcal{H}|$ available entries, reducing to the H-branch sum of Eq. (1) once the bank is full. Post-hoc refinement (Algorithm 2) operates on the completed trajectory τ after rollout termination: each frame is forward-noised to $u_0 = t_{\mathrm{ref}}$ and denoised over a schedule $t_{\mathrm{ref}} = u_0 > \cdots > u_S = 0$ with step sizes $\delta_m = u_m - u_{m+1}$ under the unbiased emulator ($\eta = 0$).

Algorithm 2 Post-Hoc Trajectory Refinement

Require: Raw trajectory $\tau = (\mathbf{x}^0, \ldots, \mathbf{x}^{K-1})$ from Algorithm 1; frozen emulator θ
Require: Sequence feature $\mathbf{c}$; structural conditioning $\mathbf{x}^{\mathrm{ref}}$; refinement schedule $[u_0, \ldots, u_S]$ with $u_0 = t_{\mathrm{ref}}$
Ensure: Refined trajectory $\hat{\tau} = (\hat{\mathbf{x}}^0, \ldots, \hat{\mathbf{x}}^{K-1})$

for $k = 0, \ldots, K-1$ **do**
  $\mathbf{x}^{\mathrm{gen}} \leftarrow \mathbf{x}^k$
  $\mathbf{x}^k_{u_0} \sim q(\cdot \mid \mathbf{x}^{\mathrm{gen}})$ ▷ partial forward diffusion to $u_0 = t_{\mathrm{ref}}$
  for $m = 0, \ldots, S-1$ **do**
    $\delta_m \leftarrow u_m - u_{m+1}$
    $(\cdot, \mathbf{s}^{\theta,k}_{\mathrm{emu}}) \leftarrow \textsc{Emulator}(\mathbf{x}^k_{u_m}, u_m, \mathbf{x}^{\mathrm{ref}}, \mathbf{c})$ ▷ unbiased score ($\eta = 0$)
    $\mathbf{x}^k_{u_{m+1}} \sim \mathbb{K}\big(\cdot \mid \mathbf{x}^k_{u_m}, \mathbf{s}^{\theta,k}_{\mathrm{emu}}\big)$ ▷ Euler–Maruyama step of size δ_m
  end for
  $\hat{\mathbf{x}}^k \leftarrow \mathbf{x}^k_{u_S}$
end for
return $\hat{\tau} = (\hat{\mathbf{x}}^0, \ldots, \hat{\mathbf{x}}^{K-1})$

B Bias and Refinement Decomposition on Fast-Folding

To disentangle the contributions of the learned bias and the score-based refinement step, we evaluate the full η vs. refinement grid on the 12 Fast-Folding proteins at E=640 and 3 seeds, varying the bias strength $\eta \in \{0,\ 0.05,\ 0.1,\ 0.2,\ 0.4\}$ and toggling refinement at t_{ref}=0.2. Table 2 reports peak coverage, time-to-coverage (TTC), and the corresponding speedup over the baseline.

Table 2: **Joint effect of η and refinement on the Fast-Folding benchmark.** Speedup is measured relative to the unbiased baseline (η=0, without refinement); "–" indicates failure to reach the baseline target within the E=640 budget. Values denote pooled means across 12 proteins and 3 random seeds, with SEM in parentheses.

	No refinement			Refinement at t_{ref}=0.2		
η	max coverage ↑	TTC ↓	speedup ↑	max coverage ↑	TTC ↓	speedup ↑
0	0.208 (0.025)	637	1.0×	0.203 (0.024)	637	1.0×
0.05	0.426 (0.033)	84	7.6×	0.409 (0.031)	82	7.8×
0.1	0.602 (0.035)	43	14.8×	0.636 (0.035)	40	15.9×
0.2	0.438 (0.033)	96	6.6×	0.678 (0.033)	17	37.5×
0.4	0.156 (0.035)	–	–	0.669 (0.028)	12	53.1×

C History-Dependent Implicit Bias

C.1 From Enhanced Sampling to Relative Visitation

Enhanced Sampling: Classical enhanced sampling starts from a Boltzmann density

$$p_{\mathrm{eq}}(\mathbf{x}) = Z^{-1} \exp[-\beta U(\mathbf{x})]. \tag{10}$$

Adding a bias potential V_{bias} changes the sampled density to

$$p_{\mathrm{bias}}(\mathbf{x}) \propto p_{\mathrm{eq}}(\mathbf{x}) \exp[-\beta V_{\mathrm{bias}}(\mathbf{x})]. \tag{11}$$

In conventional history-dependent enhanced sampling, such as metadynamics, a positive bias is accumulated in previously visited regions of collective-variable space, thereby discouraging repeated occupation [20, 4]. Motivated by this principle, we construct a generative-space bias that penalizes a region only when its historical occupation is excessive relative to the emulator distribution. The corresponding score correction therefore points away from such relatively over-occupied regions.

Estimation of History Density: Let $p_{\mathrm{env},t}(\mathbf{x}_t)$ denote the marginal density represented by the frozen emulator at time t. Fix $t > 0$, and let all densities below be defined with respect to a common reference measure on the state space. For a history bank $\mathcal{H} = \{\mathbf{h}^0, \ldots, \mathbf{h}^{H-1}\}$, define the smoothed

history-occupation density by

$$p_{\mathrm{hist},t}(\mathbf{x}_t \mid \mathcal{H}) = \sum_{j=0}^{H-1} \pi_j p_t(\mathbf{x}_t \mid \mathbf{h}^j), \qquad \pi_j \geq 0, \quad \sum_j \pi_j = 1, \tag{12}$$

where $p_t(\mathbf{x}_t \mid \mathbf{h}^j)$ is the forward perturbation kernel for the diffusion model.

Estimation of Visitation: A high historical occupation does not by itself imply over-visitation: a basin may be visited frequently simply because it has high probability under the emulator. We therefore quantify over-visitation relative to the emulator's expected occupation via the density ratio:

$$\rho_{\mathcal{H},t}(\mathbf{x}_t) := \frac{p_{\mathrm{hist},t}(\mathbf{x}_t \mid \mathcal{H})}{p_{\mathrm{env},t}(\mathbf{x}_t)}. \tag{13}$$

Regions with $\rho_{\mathcal{H},t} > 1$ and $\rho_{\mathcal{H},t} < 1$ are over- and under-occupied relative to the frozen emulator, respectively.

From Relative Visitation to a Bias Potential: The Boltzmann relationship between population and free energy motivates representing a strictly positive density p by the effective energy $F_p = -\beta^{-1} \log p + C$, defined relative to the chosen reference measure. We therefore map the relative occupation in Eq. (13) directly to the dimensionless bias $\beta V_{\mathcal{H},t}$:

$$\beta V_{\mathcal{H},t}(\mathbf{x}_t) := \eta \log \rho_{\mathcal{H},t}(\mathbf{x}_t) = \eta \log \frac{p_{\mathrm{hist},t}(\mathbf{x}_t \mid \mathcal{H})}{p_{\mathrm{env},t}(\mathbf{x}_t)}, \qquad \eta \geq 0. \tag{14}$$

Regions with $\rho_{\mathcal{H},t} > 1$ receive a positive bias and are suppressed by $\exp(-\beta V_{\mathcal{H},t})$; under-occupied regions receive a negative bias and are promoted. Thus the logarithm converts relative occupation into an additive bias potential, while η controls its strength.

C.2 The Induced Tilt and Score Bias

Applying Eq. (11) to the emulator marginal and substituting Eq. (14) gives the fixed-time tilted density

$$q_{\eta,t}(\mathbf{x}_t \mid \mathcal{H}) = \frac{1}{Z_{\eta,t}(\mathcal{H})} p_{\mathrm{env},t}(\mathbf{x}_t) \left(\frac{p_{\mathrm{env},t}(\mathbf{x}_t)}{p_{\mathrm{hist},t}(\mathbf{x}_t \mid \mathcal{H})} \right)^{\eta}, \tag{15}$$

Throughout, we assume that $p_{\mathrm{env},t}$ and $p_{\mathrm{hist},t}$ are strictly positive on a common support and that $Z_{\eta,t}(\mathcal{H}) < \infty$. This tilt reduces to the frozen emulator at $\eta = 0$ and up-weights regions that have high support under the emulator relative to their occupation in the rollout history.

Define the ideal marginal scores

$$\mathbf{s}_{\mathrm{env}} := \nabla_{\mathbf{x}_t} \log p_{\mathrm{env},t}, \qquad \mathbf{s}_{\mathrm{hist}} := \nabla_{\mathbf{x}_t} \log p_{\mathrm{hist},t}. \tag{16}$$

Here $\mathbf{s}_{\mathrm{env}}$ denotes the exact emulator-marginal score; in implementation, it is approximated by the frozen score network $\mathbf{s}^{\theta}_{\mathrm{emu}}$ of Section 4.1. Differentiating the log-density in Eq. (15) and using the fact that $Z_{\eta,t}$ is independent of $\mathbf{x}_t$ yields

$$\begin{aligned} \mathbf{s}^{\star}_{\pi} &:= \nabla_{\mathbf{x}_t} \log q_{\eta,t} \\ &= (1+\eta)\mathbf{s}_{\mathrm{env}} - \eta \mathbf{s}_{\mathrm{hist}} \\ &= \mathbf{s}_{\mathrm{env}} + \eta \underbrace{(\mathbf{s}_{\mathrm{env}} - \mathbf{s}_{\mathrm{hist}})}_{\mathbf{s}^{\star}_{\mathrm{bias}}}. \end{aligned} \tag{17}$$

Equivalently,

$$\mathbf{s}^{\star}_{\mathrm{bias}} = \nabla_{\mathbf{x}_t} \log \frac{p_{\mathrm{env},t}}{p_{\mathrm{hist},t}}, \qquad -\beta \nabla_{\mathbf{x}_t} V_{\mathcal{H},t} = \eta \mathbf{s}^{\star}_{\mathrm{bias}}. \tag{18}$$

Hence the density-ratio score is the generative-space counterpart of the force induced by the history-dependent bias potential: it subtracts the score component associated with occupied history while retaining the emulator score as a reference.

Eq. (15) is defined directly at each t. In general, forward noising does not commute with products, powers, or ratios of densities, so the family $\{q_{\eta,t}\}_t$ need not be the noised marginals of a single clean-space power-law tilt. Accordingly, we apply the fixed-time score relationship in Eq. (17) directly at each diffusion time t to construct the biased reverse score field.

C.3 History Mixture and Branchwise Estimation

To evaluate the right-hand side of Eq. (17), we need to estimate the history score $\mathbf{s}_{\mathrm{hist}}$. We use the history marginal of Eq. (12), a mixture of forward diffusion kernels $p_t(\mathbf{x}_t \mid \mathbf{h}^j)$ with weights $\{\pi_j\}$ independent of $\mathbf{x}_t$. The history score is

$$\begin{aligned}
\mathbf{s}_{\mathrm{hist}} &= \nabla_{\mathbf{x}_t} \log p_{\mathrm{hist},t}(\mathbf{x}_t \mid \mathcal{H}) \\
&= \nabla_{\mathbf{x}_t} \log \sum_j \pi_j p_t(\mathbf{x}_t \mid \mathbf{h}^j) \\
&= \frac{\sum_j \pi_j \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t \mid \mathbf{h}^j)}{\sum_l \pi_l p_t(\mathbf{x}_t \mid \mathbf{h}^l)} \\
&= \sum_j \underbrace{\frac{\pi_j\, p_t(\mathbf{x}_t \mid \mathbf{h}^j)}{\sum_l \pi_l\, p_t(\mathbf{x}_t \mid \mathbf{h}^l)}}_{\Gamma_j(\mathbf{x}_t, t, \mathcal{H})} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{h}^j) \\
&= \sum_j \Gamma_j \mathbf{s}^\star_{\mathrm{hist},j}.
\end{aligned} \tag{19}$$

Here Γ_j is the posterior responsibility of history branch j, and $\mathbf{s}^\star_{\mathrm{hist},j}$ is the known forward kernel score. Because $\sum_j \Gamma_j = 1$, the ideal bias in Eq. (17) admits the branchwise decomposition

$$\mathbf{s}^\star_{\mathrm{bias}} = \mathbf{s}_{\mathrm{env}} - \sum_j \Gamma_j \mathbf{s}^\star_{\mathrm{hist},j} = \sum_j \Gamma_j \big(\mathbf{s}_{\mathrm{env}} - \mathbf{s}^\star_{\mathrm{hist},j}\big), \tag{20}$$

which matches the structural form of the branchwise estimator in Eq. (1). Because the forward perturbation kernel is known in closed form, each ideal component score $\mathbf{s}^\star_{\mathrm{hist},j} = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t \mid \mathbf{h}^j)$ can be computed analytically and supervised directly via the per-branch objective in Eq. (5). At inference, we approximate the posterior responsibilities Γ_j in Eq. (19) using the RMSD-based weights w_j defined in Eq. (3).

C.4 Variational Characterization

The fixed-time tilt in Eq. (15) admits an equivalent variational characterization. Define the relative exploration reward

$$R_{\mathcal{H},t}(\mathbf{x}_t) := \log \frac{p_{\mathrm{env},t}(\mathbf{x}_t)}{p_{\mathrm{hist},t}(\mathbf{x}_t \mid \mathcal{H})}, \tag{21}$$

and let $\mathcal{P}_t$ denote the set of normalized densities that are absolutely continuous with respect to $p_{\mathrm{env},t}$ and for which the expectation and KL divergence below are finite. For $q \in \mathcal{P}_t$, consider the KL-regularized functional

$$\mathcal{F}_t(q) := \eta\, \mathbb{E}_{\mathbf{x}_t \sim q}\big[R_{\mathcal{H},t}(\mathbf{x}_t)\big] - D_{\mathrm{KL}}(q \,\|\, p_{\mathrm{env},t}). \tag{22}$$

The reward term favors states whose historical occupation is small relative to their emulator probability. The reverse-KL term regularizes this reallocation of probability mass toward the emulator reference; without it, maximizing the linear reward functional would cause q to degenerate toward point masses at the maximizers of $R_{\mathcal{H},t}$.

Introducing a Lagrange multiplier ζ for $\int q(\mathbf{x}_t)\, d\mathbf{x}_t = 1$ gives

$$\begin{aligned}
\mathcal{J}_t[q, \zeta] = \eta \int q(\mathbf{x}_t) R_{\mathcal{H},t}(\mathbf{x}_t)\, d\mathbf{x}_t \\
- \int q(\mathbf{x}_t) \log \frac{q(\mathbf{x}_t)}{p_{\mathrm{env},t}(\mathbf{x}_t)}\, d\mathbf{x}_t + \zeta \left(\int q(\mathbf{x}_t)\, d\mathbf{x}_t - 1 \right).
\end{aligned}$$

Its first variation is

$$\frac{\delta \mathcal{J}_t}{\delta q(\mathbf{x}_t)} = \eta R_{\mathcal{H},t}(\mathbf{x}_t) - \log \frac{q(\mathbf{x}_t)}{p_{\mathrm{env},t}(\mathbf{x}_t)} - 1 + \zeta. \tag{23}$$

The stationarity and normalization conditions yield

$$q_t^\star(\mathbf{x}_t) = \frac{1}{Z_{\eta,t}} p_{\mathrm{env},t}(\mathbf{x}_t) \exp[\eta R_{\mathcal{H},t}(\mathbf{x}_t)] \,, \qquad Z_{\eta,t} = \int p_{\mathrm{env},t}(\mathbf{x}_t) \exp[\eta R_{\mathcal{H},t}(\mathbf{x}_t)] \; d\mathbf{x}_t. \quad (24)$$

The functional in Eq. (22) is the sum of a linear term and the negative KL, which is strictly concave in q; hence this stationary density is the unique maximizer whenever $Z_{\eta,t} < \infty$. Substitution of Eq. (21) into Eq. (24) recovers Eq. (15).

D Supplementary Evaluation on Fast-Folding Proteins

D.1 Expanded Baseline Comparison

We extend the Fast-Folding evaluation (Section 5.3) to include ConfDiff [38] and DynaFold [11], alongside AlphaFlow [16], BioEmu [21], MDGen [17], and our frozen AlphaFolding-based emulator [8]. In addition to the coverage, structural validity, and diversity metrics defined in Section 5.1, Table 3 reports free-energy surface (FES) reconstruction errors (FES-MAE and FES-RMSE, in $\mathrm{kcal/mol}$) relative to the MD reference [21].

Table 3: **Expanded baseline comparison on Fast-Folding.** Structural validity metrics are in percentages and FES errors are in $\mathrm{kcal/mol}$.

			Structural validity			Diversity			FES error	
Category	Method	Cov ↑	CA% ↑	CN% ↑	Clash-Free ↑	Div@.90 ↑	Div@.95 ↑	mDiv-mean ↑	FES-MAE ↓	FES-RMSE ↓
Conformation	AlphaFlow [16]	0.098	100.00	100.00	99.95	0.091	0.091	0.091	1.011	1.347
	BioEmu [21]	0.654	99.96	99.91	99.99	0.296	0.296	0.296	**0.735**	**0.887**
	ConfDiff [38]	**0.688**	100.00	100.00	99.74	0.433	0.433	0.433	0.879	1.068
Trajectory	MDGen [17]	0.051	8.59	3.25	99.65	0.183	0.073	0.086	0.911	1.097
	DynaFold [11]	0.450	99.48	99.73	96.24	0.450	0.442	0.431	1.116	1.395
	Emulator [8]	0.207	100.00	99.79	99.53	0.182	0.181	0.181	0.961	1.263
	+ Implicit Bias	0.640	99.71	98.96	98.95	**0.499**	**0.495**	**0.490**	0.825	1.056

Adding our implicit bias to the frozen emulator raises low-energy state coverage from 0.207 to 0.640 and lowers FES-MAE from 0.961 to 0.825 while maintaining $\geq 98.95\%$ structural validity, outperforming DynaFold across coverage (0.450), steric validity (96.24%), and FES-MAE (1.116). Relative to IID conformation generators, our method achieves the highest compliant diversity overall (mDiv-mean 0.490 vs. 0.433 for ConfDiff and 0.296 for BioEmu) and lower FES error than ConfDiff (0.825 vs. 0.879). Coverage remains slightly below BioEmu (0.654) and ConfDiff (0.688), and BioEmu attains the lowest FES-MAE (0.735).

D.2 Dynamical and Kinetic Diagnostics

To evaluate temporal realism beyond coverage and per-frame validity, Table 4 reports four dynamical diagnostics against the MD reference: **AutoCor-MAE** [31], the mean absolute deviation from MD temporal autocorrelation; **Valid Path Rate** [28], the fraction of generated state-transition paths supported by a reference MD Markov state model (MSM); **Transition NLL**, the average negative log-likelihood of generated transitions under the empirical MD transition matrix; and **ITS-log-MAE**, the log-scale mean absolute error of slow implied timescales relative to the MD MSM.

Table 4: **Dynamical diagnostics on Fast-Folding.** "–" denotes the zero-error MD reference for autocorrelation.

Method	AutoCor-MAE ↓	Valid Path Rate ↑	Transition NLL ↓	ITS-log-MAE ↓
MD (Held-out)	–	69.46%	1.7087	0.878
MDGen [17]	0.537	21.16%	1.5341	1.454
DynaFold [11]	0.251	28.58%	1.6534	1.780
Emulator [8]	**0.186**	45.42%	**1.3234**	3.413
+ Implicit Bias	0.342	**66.50%**	1.8147	**1.696**

Adding the bias raises the Valid Path Rate from 45.42% to 66.50% (approaching the held-out MD reference of 69.46%) and halves ITS-log-MAE from 3.413 to 1.696. Whereas the unbiased emulator remains trapped in local basins—yielding lower $\mathrm{AutoCor\text{-}MAE}$ (0.186) and Transition NLL (1.3234) at the cost of poor slow-mode timescales—the bias promotes barrier crossing and valid transition pathways while perturbing local step-to-step transitions ($\mathrm{AutoCor\text{-}MAE}$ 0.342, Transition NLL 1.8147).

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: The contributions stated in the abstract and introduction are each instantiated in the methodology section and supported by the experimental sections, and the scope of the empirical claims is stated alongside the corresponding results.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: The discussion section contains three limitations: the bias is unconditional, the reachable conformational space is bounded by the frozen emulator's training distribution, and the base emulator does not target the equilibrium distribution.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: Appendix C states the assumptions and provides the derivations for the fixed-time density tilt, its score identity, and its variational characterization.

Guidelines:

- The answer [N/A] 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: The experimental setup section specifies the data splits, training schedule, and rollout settings used to obtain the reported results, and full architecture and optimization details are listed in the appendix.

Guidelines:

- The answer [N/A] 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: All datasets used in this work are publicly available and cited at first use; the training and rollout codebase together with the evaluation scripts will be released publicly.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup section provides the data splits, training duration, and the main rollout hyperparameters; full optimizer schedules and architecture hyperparameters are in the appendix.

   Guidelines:

   - The answer [N/A] 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: [Yes]

   Justification: All entries in the main result tables are population statistics (means) computed across the proteins in the evaluation subset, where each per-protein contribution is itself a per-trajectory mean. Per-cell confidence intervals are omitted from the main tables for compactness; the per-protein scores used to obtain these means are released so that confidence intervals and other variability measures can be reproduced.

   Guidelines:

- The answer [N/A] 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: [Yes]

   Justification: The experimental setup section reports the hardware and training duration, and full optimization details are in Appendix A.1.

   Guidelines:

   - The answer [N/A] 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: The work concerns generative models of protein backbone dynamics and uses publicly available molecular-simulation datasets only; it does not involve human subjects, personally identifiable data, or any application that conflicts with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: The proposed method targets accelerated exploration of protein conformational landscapes, which has positive applications in drug discovery and structural biology; we do not foresee direct negative societal impact, as the released models do not enable malicious or dual-use capabilities beyond those already present in published protein-structure generators.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released models generate protein backbone trajectories conditioned on a query structure and pose no high-risk misuse profile beyond that of existing publicly released structure-prediction models.

Guidelines:

- The answer [N/A] 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: All external datasets, models, and software used in this work are cited at first use and are employed in compliance with their respective licenses.

Guidelines:

- The answer [N/A] 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: The training and evaluation codebase, together with configuration files needed to reproduce the reported results, will be released publicly; trained model checkpoints will be released alongside the code under a permissive license.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The work involves no crowdsourcing and no research with human subjects.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The work involves no human subjects, so IRB review is not applicable.

Guidelines:

- The answer [N/A] 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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core methodology of the paper does not use large language models as an important, original, or non-standard component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.