Atomic Trajectory Modeling with State Space Models for Biomolecular Dynamics

Liang Shi[*]
School of Computer Science
Peking University
liang.shi@stu.pku.edu.cn

Jiarui Lu
Mila - Québec AI Institute
Université de Montréal
jiarui.lu@umontreal.ca

Junqi Liu[*]
School of Computer Science
Peking University
liujunqi@pku.edu.cn

Chence Shi
BioGeometry
chence.shi@umontreal.ca

Zhi Yang[†]
School of Computer Science
Peking University
yangzhi@pku.edu.cn

Jian Tang[†]
Mila - Québec AI Institute
HEC Montréal
CIFAR AI Chair
jian.tang@hec.ca

Abstract

Understanding the dynamic behavior of biomolecules is fundamental to elucidating biological function and facilitating drug discovery. While Molecular Dynamics (MD) simulations provide a rigorous physical basis for studying these dynamics, they remain computationally expensive for long timescales. Recent deep generative models accelerate conformation generation but often either discard temporal correlations entirely or struggle to condition on the extended history required for faithful kinetics, a consequence of the effectively non-Markovian nature of partially observed subsystem coordinates. To bridge this gap, we introduce ATMOS, a novel generative framework based on State Space Models (SSM) designed to generate atom-level MD trajectories for biomolecular systems. ATMOS integrates a Pairformer-based state transition mechanism to capture temporal dependencies, with a diffusion-based module to decode trajectory frames autoregressively. We demonstrate that ATMOS achieves state-of-the-art performance in generating conformation trajectories for both protein monomers and protein-ligand systems. This work provides a unified and computationally efficient framework for biomolecular trajectory generation, taking a step toward dynamics foundation models.

1 Introduction

Elucidating the functional mechanisms of biomolecules requires a comprehensive understanding of protein dynamics, rather than merely analyzing static crystal structures. Traditionally, Molecular Dynamics (MD) simulations are the gold standard for this task, relying on fundamental physical principles to reveal critical thermodynamic and kinetic properties. However, despite rigorous adherence to physical laws, traditional MD methods are computationally expensive, often struggling to access biologically relevant timescales or sample rare conformational transitions efficiently.

To alleviate these computational burdens, deep generative models have emerged as powerful surrogates. Broadly, these approaches fall into two categories. Static ensemble generators, such as AlphaFlow [11] and BioEmu [16], learn to approximately sample conformations from the equilibrium Boltzmann distribution, yet they do so in an *i.i.d.* fashion, discarding temporal correlations and

[*]Work conducted during their internships at BioGeometry.
[†]Corresponding authors.

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

kinetic information. In contrast, trajectory generation methods explicitly model temporal evolution: MDGen [12] applies flow matching to time-series of frames in MD simulations; TEMPO [39] decouples slow and fast motions hierarchically to sample trajectories; ConfRover [30] employs a causal transformer with an SE(3) diffusion decoder; and STAR-MD [31] leverages joint spatio-temporal attention for more scalable autoregressive modeling. Despite these advances, existing methods still face challenges. Methods that employ temporal attention scale quadratically with trajectory length. More efficient non-attention approaches [28, 15, 4] face a different challenge: MD trajectories of a subsystem (e.g., protein-only coordinates) are effectively non-Markovian due to implicit solvent coupling and coarse-graining [30]. Consequently, accurate kinetic modeling requires conditioning on an extended prefix of past frames, which these methods often struggle to incorporate efficiently.

In this work, we introduce **A**tomic **T**rajectory **MO**deling with **S**SMs (**ATMOS**), which formulates biomolecular dynamics generation as a sequence modeling problem. ATMOS employs a state space model (SSM) as a learnable propagator for MD trajectories. The SSM maintains a recurrent latent state that accumulates information from past frames, enabling the model to condition on an extended temporal context without the quadratic cost of attention. This design directly addresses the non-Markovian nature of the observed subsystem coordinates, as the latent state can efficiently summarize the history required for accurate kinetic predictions. We parameterize the SSM transition function with a Pairformer module [1], and decode the updated latent state back to full atomic coordinates via a diffusion model. A hierarchical coarse-to-fine sampling strategy, augmented with chunk stitching, ensures global consistency and temporal smoothness across long roll-outs. Being trained end-to-end on MD trajectories, ATMOS achieves state-of-the-art trajectory generation performance on both mdCATH [23] and MISATO [32] benchmarks. ATMOS provides a unified and computationally efficient framework for biomolecular trajectory generation, taking a step toward protein dynamics foundation models.

Main contributions. (i) We propose ATMOS, a unified generative framework for biomolecular dynamics that handles diverse systems, ranging from monomeric proteins to protein-ligand complexes. (ii) We innovatively adapt State Space Models to the trajectory generation, enabling the capture of temporal dependencies with improved inference efficiency compared to traditional simulation or purely attention-based approaches. A hierarchical coarse-to-fine sampling strategy with chunk stitching further ensures long-range temporal smoothness. (iii) ATMOS achieves state-of-the-art performance on the large-scale mdCATH [23] and MISATO [32] benchmarks for trajectory generation, validating the model's ability to generalize across diverse biological systems.

2 Related Work

Generative Modeling of Protein Ensembles. Deep generative models have been developed to sample conformations *i.i.d.*. Seminal work such as Boltzmann Generators [24] utilized normalizing flows to sample from the Boltzmann distribution, though scaling to systems beyond training remained challenging. Recent advances leverage diffusion and flow matching on large-scale MD datasets; for instance, AlphaFlow [11] fine-tunes pre-trained folding models to capture ensemble diversity, while EBA [18] enhances this framework with physical energy-based alignment. Other approaches focus on specific physical constraints or data regimes: Distributional Graphormer (DiG) [40] and ConfDiff [35] incorporate energy or force guidance to improve physical validity, Str2Str [20] employs a zero-shot translation framework without reliance on MD training data, ESMDiff [19] adopts latent language modeling to efficiently sample conformation ensembles, and BioEmu [16] integrates experimental stability data to predict thermodynamic properties.

Generative Modeling of Molecular Trajectories. While ensemble models capture time-independent distributions, trajectory generation methods aim to emulate the temporal evolution of molecular systems, serving as surrogates for expensive Molecular Dynamics (MD) simulations. Initial efforts, such as ITO [28], TimeWarp [15] and EquiJump [4], focused on learning time-coarsened transition operators to jump over large time steps. More recent frameworks model the joint distribution of entire trajectories to capture long-range temporal dependencies. MDGen [12] treats trajectories as time-series of 3D structures, utilizing flow matching to enable forward simulation, transition path interpolation, and upsampling. To handle the multi-scale nature of protein motions, TEMPO [39] introduces a hierarchical trajectory generation framework that separates slow collective motions from fast local fluctuations. ConfRover [30] employs a causal transformer with an SE(3) diffusion decoder

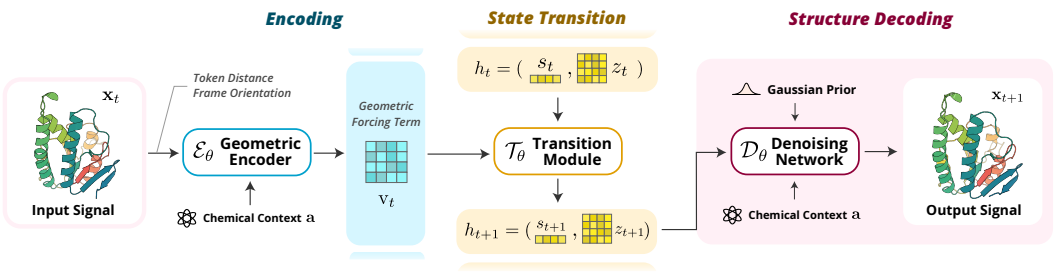


Figure 1: **Overview of the Proposed Atomic Trajectory Modeling Framework.**

to simultaneously learn conformational distributions and dynamics. STAR-MD [31] introduces an efficient joint spatio-temporal attention mechanism, enabling more scalable autoregressive dynamics modeling. Other specialized architectures include BioMD [6] for protein-ligand unbinding pathways, and PTraj-Diff [38] for protein-protein complex dynamics.

3 Method

3.1 Preliminaries

Problem Formulation. We conceptualize the generation of biomolecular trajectory as a conditional sequence modeling task. Consider a biomolecular system consisting of N atoms. We represent the dynamic evolution of this system as a trajectory of coordinates $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) \in \mathbb{R}^{T\times N\times 3}$, where $\mathbf{x}_t \in \mathbb{R}^{N\times 3}$ denotes the 3D Cartesian coordinates of all N atoms at time step t, and T represents the trajectory length, i.e., the number of frames. In addition to coordinates, the system is defined by time-invariant atomic features $\mathbf{a}$, where $\mathbf{a} \in \mathbb{R}^{N\times d_a}$ represents atom-level attributes including atom type, residue type, molecule-level identifiers that indicate whether an atom belongs to a protein or ligand, etc. Our objective is to learn a generative model p_θ that approximates the true distribution of trajectories given the initial conformation $\mathbf{x}_1$ and the context $\mathbf{a}$. Following standard autoregressive practices, we factorize the joint probability over the temporal dimension:

$$p_\theta(\mathbf{X}|\mathbf{x}_1, \mathbf{a}) = \prod_{t=1}^{T-1} p_\theta(\mathbf{x}_{t+1}|\mathbf{x}_{1:t}, \mathbf{a}) \tag{1}$$

This formulation allows the model to generate physically consistent dynamics by conditioning the next state on the history of the trajectory and the chemical identity of the system.

State Space Models (SSM). State Space Models [7, 8] provide a unified framework for modeling sequential data by mapping a 1-dimensional input signal $u(t) \in \mathbb{R}$ to an output signal $y(t) \in \mathbb{R}$ through a latent state $h(t) \in \mathbb{R}^D$. In their continuous-time form, linear SSMs are defined by the following ordinary differential equations (ODEs):

$$\dot{h}(t) = \mathbf{A}h(t) + \mathbf{B}u(t), \quad y(t) = \mathbf{C}h(t) \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{D\times D}$, $\mathbf{B} \in \mathbb{R}^{D\times 1}$, and $\mathbf{C} \in \mathbb{R}^{1\times D}$ are learnable projection matrices. To apply this framework to discrete sequence data (such as MD trajectories sampled at fixed intervals Δ), the continuous system is discretized, typically via the Zero-Order Hold method, yielding the recurrence:

$$h_t = \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}u_t, \quad y_t = \mathbf{C}h_t \tag{3}$$

Here, $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ are the discretized state parameters. This recurrent formulation allows for efficient $O(1)$ inference per step, making SSM particularly well-suited for generating long molecular trajectories compared to the quadratic complexity of standard attention mechanisms.

Notations. Throughout this paper, we use bold lowercase letters (e.g., $\mathbf{x}, \mathbf{h}$) for vectors and atom-level representations, and bold or calligraphic uppercase letters (e.g., $\mathbf{A}, \Phi, \mathcal{E}$) for matrices and operations. We denote the coordinate of the i-th atom at time t as $\mathbf{x}_{t,i} \in \mathbb{R}^3$. The set of all atoms is indexed by $i \in \{1, \ldots, N\}$, and discrete time steps are indexed by $t \in \{1, \ldots, T\}$.

3.2 An Atomic Trajectory Modeling Framework

SSM as Learnable Biomolecular Simulators. We frame the generation of biomolecular trajectories as the learning of a time-continuous evolution function, functionally analogous to the integrator used in classical MD simulations. In traditional MD simulations, the system evolves according to an integrator that explicitly calculates future positions based on current coordinates, momenta, and interatomic forces derived from a force field. While rigorous, this process is computationally intensive for long timescales. To overcome this, ATMOS adopts SSM as efficient, learnable biomolecular propagators. Unlike standard attention-based sequence models that must revisit the entire history for every prediction, SSM maintains a latent recurrent state. This aligns with the physical reality of molecular dynamics: the future evolution of a system is governed by its current *phase-space* state (positions and momenta). By replacing the expensive numerical integrator with a neural SSM, we achieve a framework that is both physically intuitive and computationally efficient.

Specifications. Formally, we instantiate the SSM framework for atomic trajectory modeling by defining the input signal, latent state, and output prediction as follows:

- **Input Signal ($\mathbf{x}_t$):** The time-varying input is the configuration of the system at time t, represented by the atomic coordinates $\mathbf{x}_t \in \mathbb{R}^{N\times 3}$. This serves as the observable states of the sequence.
- **Latent State ($\mathbf{h}_t$):** The recurrent hidden state $\mathbf{h}_t$ acts as an implicit representation of the system's complete phase-space state. While the input $\mathbf{x}_t$ provides only the geometric configuration, the latent state $\mathbf{h}_t$ captures the auxiliary dynamic variables necessary for propagation.
- **Output Signal ($\mathbf{x}_{t+1}$):** Conditioning on the updated latent $\mathbf{h}_{t+1}$, the model operates autoregressively to predict the updated coordinates $\mathbf{x}_{t+1} \in \mathbb{R}^{N\times 3}$ for the subsequent time step.

Learning Objective. This framework provides two critical advantages. First, the constant inference complexity of SSMs per step (in recurrent mode) allows ATMOS to generate trajectories of arbitrary length T with linear scaling, addressing the bottleneck of existing autoregressive methods. Second, the continuous-time formulation of SSMs naturally handles the temporal correlations inherent in physical motion, smoothing the modeling of conformational transitions. Under this framework, ATMOS can be trained to maximize the likelihood of the ground-truth trajectory $\mathbf{X}$ given the chemical context $\mathbf{a}$. The learning objective is:

$$\mathcal{L}_{\text{SSM}} = -\sum_{t=1}^{T-1} \log p_\theta(\mathbf{x}_{t+1} \mid \mathbf{h}_t, \mathbf{x}_t, \mathbf{a}) \tag{4}$$

where p_θ is parameterized by the predicted shift in coordinates.

3.3 Model Architecture

Compact Coupled Latent Representation. We adopt a compact *coupled latent* widely used in modern structure prediction architectures [13, 1]. Specifically, we define the latent SSM state $\mathbf{h}_t$ as a coupled representation operating on a tokenized sequence of length L (where $L \leq N$):

$$\mathbf{h}_t \triangleq (\mathbf{s}_t, \mathbf{z}_t). \tag{5}$$

Here, the system is tokenized such that atom-level representations of protein are grouped into residue-level latents, otherwise the atomic resolution is retained. Consequently, $\mathbf{s}_t \in \mathbb{R}^{L\times d_s}$ denotes the *single representation*, capturing the state of each residue or ligand atom, and $\mathbf{z}_t \in \mathbb{R}^{L\times L\times d_z}$ denotes the *pair representation*, encoding the interactions between tokens.

Neural SSM Parametrization. While classical SSMs rely on linear projections (Eq. 2), modeling complex molecular interactions requires high-capacity non-linear transitions. Based on the coupled latent space, one can effectively parameterize the state evolution using neural networks. The architecture is composed of three functional modules:

Context and Input Encoding. We first initialize the latent state by tokenizing the chemical context $\mathbf{a}$. An embedding layer maps contexts such as atom types and residue indices to the initial state $\mathbf{h}_0 \triangleq (\mathbf{s}_0, \mathbf{z}_0) = \mathbf{e}_\theta(\mathbf{a})$, establishing the context prior of the system before dynamics begin. At each time step t, the input signal $\mathbf{x}_t$ (current atomic coordinates) must be injected into the latent space

to update the system's trajectory. We employ a geometric encoder $\mathcal{E}_\theta$ to extract rot-trans invariant features from $\mathbf{x}_t$:

$$\mathbf{v}_t = \mathcal{E}_\theta(\mathbf{x}_t) \tag{6}$$

This term $\mathbf{v}_t$ acts as the forcing term (analogous to $\mathbf{B}\mathbf{u}_t$ in Eq. 3), representing the instantaneous geometric update derived from the current frame $\mathbf{x}_t$.

State Transition. The core propagator of ATMOS is the state transition function $\mathcal{T}_\theta$, which evolves the latent state $\mathbf{h}$ from t to $t+1$. We instantiate this transition using the Pairformer module [1], denoted as $\Phi_\theta : \mathbf{s}, \mathbf{z} \mapsto \mathbf{s}', \mathbf{z}' \in \mathbb{R}^{L\times d_s}, \mathbb{R}^{L\times L\times d_z}$. The discretized update rule is defined as ($\Delta t > 0$ indicates the time step):

$$\mathbf{h}_{t+1} \triangleq (\mathbf{s}_{t+1}, \mathbf{z}_{t+1}) = \mathcal{T}_\theta(\mathbf{h}_t, \mathbf{v}_t, \Delta t). \tag{7}$$

Here, the transition kernel $\mathcal{T}_\theta$ is defined as

$$\mathcal{T}_\theta\big(\mathbf{h}_t, \mathbf{v}_t, \Delta t\big) \triangleq \mathcal{T}_\theta\big(\mathbf{s}_t, \mathbf{z}_t; \mathbf{v}_t; \Delta t\big) = \Phi_\theta\big(\mathbf{s}_t + \tau_\theta(\Delta t),\ \mathbf{z}_t + \mathbf{v}_t + \tau_\theta(\Delta t)\big),$$

where $\tau_\theta(\cdot)$ is broadcasted learnable timestep embeddings. Intuitively, the Pairformer Φ_θ functions as a learnable integrator, calculating the next latent state by reasoning over the previous memory $\mathbf{h}_t$ and the current geometric update $\mathbf{v}_t$.

Structure Decoding. Finally, to map the evolved latent state back to the coordinate space, we employ a diffusion-based decoder. We model the generation of the next frame $\mathbf{x}_{t+1}$ as a reverse diffusion process. The core component is a denoising network $\mathcal{D}_\theta$ that predicts the clean, denoised structure from a noisy state. Specifically, for a given trajectory step $t+1$, we introduce a diffusion time variable $\gamma \in [0, +\infty]$ to distinguish the diffusion time from the trajectory index t. The decoder is conditioned on the noisy structure $\tilde{\mathbf{x}}^{(\gamma)}$, the chemical context $\mathbf{a}$, and the evolved latent representations $(\mathbf{h}_{t+1})$. The denoising operation is defined as:

$$\hat{\mathbf{x}}_{t+1} = \mathcal{D}_\theta(\tilde{\mathbf{x}}^{(\gamma)}, \mathbf{a}, \mathbf{h}_{t+1}, \gamma) \tag{8}$$

where $\hat{\mathbf{x}}_{t+1}$ represents the predicted denoised coordinates.

3.4 Training Objectives

Training Regime. To efficiently train ATMOS on MD trajectory data, we employ a teacher-forcing [37] with subsampling strategy during the training stage. Rather than unrolling the full trajectory, we optimize the model on sub-trajectory windows and input the prefix states as the ground truth. Specifically, for a given training trajectory $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ of length T, we sample a start index t_0, a temporal stride Δt, and a window size k such that $1 \le t_0 < t_0 + k\Delta t \le T$. The model processes the first $k-1$ frames as context to predict the target state $\mathbf{x}_{t_0+k\Delta t}$ at the strided k-th step. For simplicity, we adopt a fixed stride in this work, whereas we believe it is worth exploring learn dynamics across varied timescales by randomizing Δt.

Loss Functions. We train ATMOS end-to-end with two important loss terms: (1) latent fidelity loss (distogram) and (2) reconstruction loss (diffusion), which are detailed as follows.

Latent Fidelity Loss (Distogram). To enforce geometric consistency in the latent space, we impose a distogram-based latent fidelity loss. Following AlphaFold 3 [1], a lightweight prediction head $\mathbf{P}_\theta$ is applied to the pair representation $\mathbf{z}_t$, which outputs a 64-bin probability distribution over discretized distances for each token pair (i, j). The distogram loss is defined as the cross-entropy between the predicted distribution and the target bin (one-hot) derived from the target coordinates:

$$\mathcal{L}_{\text{disto}} = -\mathbb{E}_{\mathbf{x}_t}\big[\frac{1}{L^2}\sum_{i,j=1}^{L}\sum_{b=1}^{64}\delta^{(b)}_{(i,j)}(\mathbf{x}_t)\ \log \mathbf{P}^{(b)}_{\theta,(i,j)}(\mathbf{z}_{t,(i,j)})\big], \tag{9}$$

where $\delta^{(b)}_{(i,j)}(\mathbf{x}_t)$ indicates whether the distance between tokens i and j in the target structure falls into the bin b, and the target frame $\mathbf{x}_t$ is sampled via the training regime above.

Reconstruction Loss (Denoising Diffusion). This objective optimizes the diffusion-based decoder $\mathcal{D}_\theta$ to (conditionally) recover the target frame $\mathbf{x}_t$ from a noise-corrupted state $\tilde{\mathbf{x}}^{(\gamma)}$. Firstly, we parameterize the decoder $\mathcal{D}_\theta$ using an EDM-style diffusion process [14]. During training, the noisy state is sampled following the diffusion time schedule $\gamma \sim p(\gamma) \in \mathbb{R}^+$:

$$\tilde{\mathbf{x}}^{(\gamma)} = \mathbf{x}_t + \gamma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}). \tag{10}$$

To ensure the model focuses on non-trivial structural deviations rather than global rotations or translations, we apply the Kabsch algorithm to rigid align the target frame to the denoise coordinates. The loss is defined as:

$$\mathcal{L}_{\text{recon}} = \mathbb{E}_{\mathbf{x}_t,\gamma,\epsilon}\left[\sum_{i=1}^{N} \omega(i, \sigma_{\text{data}}, \gamma) \left\|\hat{\mathbf{x}}_i - \text{Align}(\mathbf{x}_{t,i})\right\|^2\right] \tag{11}$$

where $\hat{\mathbf{x}} = \mathcal{D}_\theta(\tilde{\mathbf{x}}^{(\gamma)}, \mathbf{a}, \mathbf{h}_t, \gamma)$ is the predicted denoised structure. The weighting term $\omega(i, \sigma_{\text{data}}, \gamma) = w_{\text{entity}}(i) \cdot w_{\text{diff}}(\gamma, \sigma_{\text{data}})$ balances the loss by accounting for: atom entities w_{entity} (distinguishes atoms from protein, ligand, etc.), dataset-specific noise scales $\sigma_{\text{data}} > 0$, and diffusion weighting $w_{\text{diff}}(\gamma, \sigma_{\text{data}}) \triangleq (\gamma^2 + \sigma_{\text{data}}^2)/(\gamma \cdot \sigma_{\text{data}})^2$.

The final training objective is a linear combination as:

$$\mathcal{L} = \lambda_{\text{recon}} \cdot \mathcal{L}_{\text{recon}} \; + \; \lambda_{\text{disto}} \cdot \mathcal{L}_{\text{disto}}, \tag{12}$$

with $\lambda_{\text{recon}} > 0, \lambda_{\text{disto}} > 0$ as tunable hyperparameters.

3.5 Trajectory Sampling

Bi-Level Sampling Protocol. During inference, ATMOS samples trajectories in a specific temporal granularity Δt. However, autoregressive generation of high-dimension conformation over long horizons can lose consistency due to accumulated drift. Motivated by prior work [9, 39], we adopt a hierarchical, coarse-to-fine sampling strategy that decouples conformation changes from local fluctuations. In practice, we specialize two variants of ATMOS: a *keyframe generator* $\mathcal{G}$ trained with a large temporal stride ($\Delta t^{\text{kf}} \triangleq u \cdot \Delta t$, where $u > 1$ is an upsampling factor) to predict distinct anchor conformations, and an *interpolator* $\mathcal{I}$ trained with a fine temporal stride (Δt) to fill in the dynamics between anchors; in other words, the interpolator upsamples (with the factor $u > 1$) the trajectory generated from $\mathcal{G}$.

Conditioned Interpolation with Chunk Stitching. While the keyframe generator operates via standard autoregressive rollout, the interpolator must produce smooth transitions both between and across anchor intervals. To achieve this, the interpolator variant $\mathcal{I}$ is conditioned not only on the starting anchor frame $\mathbf{x}_t$ and the destination anchor frame $\mathbf{x}_{t'}$ (with $t' = t + \Delta t^{\text{kf}}$), but also on a short prefix of fine-grained frames from the preceding interpolation interval. Concretely, let $\mathbf{x}_{t-p\Delta t:t}$ denote the p most recent generated frames up to and including the anchor frame $\mathbf{x}_t$. The interpolator then generates the u frames of the current interval via

$$p_\theta(\mathbf{x}_{t:t'} \mid \mathbf{x}_{t-p\Delta t:t}, \mathbf{x}_{t'}, \mathbf{a}, t_{\text{rem}}) = \prod_{k=1}^{u} p_\theta\big(\mathbf{x}_{t+k\Delta t} \mid \mathbf{x}_{t-p\Delta t:t+(k-1)\Delta t}, \mathbf{x}_{t'}, \mathbf{a}, t_{\text{rem}}\big), \tag{13}$$

where $t_{\text{rem}} \equiv (u - k + 1)\Delta t$ is the temporal distance to the next anchor frame. For the very first interpolation interval (where no prior fine-grained frames exist), we simply use a prefix consisting only of the initial anchor frame (i.e., $p = 1$). This prefix mechanism ensures temporal continuity across anchor boundaries and eliminates discontinuities that would otherwise arise from restarting interpolation at each anchor frame independently.

4 Experiments

In this section, we provide details on the application of ATMOS to molecular dynamics (MD) trajectory generation.

Dataset. We evaluate ATMOS on two modalities: protein and protein-ligand. For protein dynamics, we adopt the mdCATH dataset [23], which provides MD simulations for a wide range of protein domains. mdCATH features simulations of 5,398 domains at five different temperatures, each with five replicas. The production frames are sampled at 1ns intervals from simulations. We follow the dataset construction of TEMPO [39], which focuses on the 320K trajectories and uses 1,000 domains for training, 50 for validation, and 64 for testing, with an average sequence similarity of 18.93% between the training and test sets, ensuring a challenging generalization benchmark.

Table 1: **Evaluation on mdCATH**. The best scores are highlighted in **bold** and the second-best scores are underlined. r: Pearson correlation; J: Jaccard similarity; $\mathcal{W}_2$: 2-Wasserstein distance. The MD (400ns) column reports oracle scores computed using one 400 ns MD trajectory from the reference set for each test target.

Metrics	AlphaFlow	EBA	MDGen	TEMPO	ConfRover	ATMOS	MD (400ns)
Pairwise RMSD r ↑	0.88	0.80	0.65	0.73	0.69	**0.89**	0.96
Global RMSF r ↑	0.86	0.74	0.67	0.61	0.62	**0.87**	0.95
Per-target RMSF r ↑	0.87	0.80	0.77	0.52	0.81	**0.88**	0.95
Root mean $\mathcal{W}_2$-dist. ↓	**3.06**	4.04	3.86	4.98	4.52	3.30	1.89
MD PCA $\mathcal{W}_2$-dist. ↓	2.39	2.67	2.75	2.38	3.04	**2.01**	1.67
Joint PCA $\mathcal{W}_2$-dist. ↓	2.82	3.42	3.54	4.71	4.15	**2.79**	1.75
% PC-sim > 0.5 ↑	30.7	36.7	3.17	7.94	22.2	**45.3**	61.9
Weak contacts J ↑	0.56	0.40	0.57	0.48	0.43	**0.67**	0.84
Transient contacts J ↑	**0.34**	0.19	0.22	0.20	0.22	**0.34**	0.55
t-RMSD Error ↓	-	-	2.71	2.42	2.50	**1.79**	0.00
Decorr-TIC0 r ↑	-	-	0.07	-0.14	-0.05	**0.25**	1.00
MSM-Trans Error ↓	-	-	0.28	0.31	0.30	**0.24**	0.23
VAL-Cα ↑	0.89	0.91	0.07	0.01	0.53	**0.97**	0.94

For protein-ligand dynamics, we use the MISATO dataset [32], which provides trajectories capturing the coupled dynamics of proteins and ligands. The simulations are performed at 300 K for 8 ns, yielding 100 snapshots per trajectory. We preprocess MISATO following a procedure similar to Boltz-2 [25], resulting in 12,662 systems for training, 1,258 for validation, and 1,276 for testing (see Appendix E for preprocessing details). Due to the computational cost of trajectory generation, we randomly sample 100 systems from the test set for final evaluation, aligning the test scale with Boltz-2. The PDB IDs of the sampled test systems are provided in Appendix E.

Implementation details. We initialize the context embedding layer and the diffusion decoder with weights from Protenix [3]. During training on MD trajectories, the diffusion decoder is fine-tuned end-to-end together with the newly introduced and randomly initialized geometric encoder $\mathcal{E}_\theta$ and the state-transition kernel $\mathcal{T}_\theta$. Architectural details of these modules are provided in the Appendix D. We adopt the dual-scale generation strategy described in Section 3.5. On the mdCATH dataset, we train two separate models: one for coarse-scale keyframe generation at a 20ns interval and another for fine-scale interpolation at a 1ns interval. Similarly, for MISATO, we train a coarse model at 0.8ns and a fine model at 0.08ns. This yields four trained models in total; we leave the exploration of a unified model for future work. In the interpolation stage, the chunk stitching prefix length is set to $p = 5$ for both datasets. Comprehensive training hyperparameter settings are listed in the Appendix F.

4.1 Protein Dynamics

In Table 1, we compare ATMOS against several recent trajectory generation approaches. For trajectory-level modeling, we evaluate MDGEN [12], TEMPO [39], and ConfRover [30]. MDGEN and TEMPO are trained on the mdCATH dataset, while for ConfRover, which only provides inference code, we use the author-released checkpoint trained on the ATLAS dataset [33]. We exclude BioMD [6] and STAR-MD [31] from the comparison because neither codebase was publicly available at the time of writing. We additionally compare ATMOS against ensemble generators, we include AlphaFlow [11] and EBA [18], both of which produce equilibrium ensembles without modeling temporal correlations. For a fair comparison, we use the AlphaFlow-MD + Templates variant of AlphaFlow, which, like the trajectory methods, conditions on a single input structure.

For each protein in the test set, we condition on the initial structure and generate a trajectory of 400 frames at 1ns intervals, which we then compare to the corresponding MD reference trajectory. For AlphaFlow and EBA, we generate an independent set of conformations of equal size for comparison.

Our evaluation framework spans three complementary aspects: ensemble-level properties, trajectory-specific kinetics, and structural validity. **For ensemble-level properties**, following AlphaFlow [11], we assess three aspects: flexibility (Table 1, rows 2-4), distributional accuracy (Table 1 rows 5-8), and ensemble observables (Table 1 rows 9-10). **For trajectory-specific kinetics**, we adopt three metrics: (i) t-RMSD Error: For each frame, we compute the RMSD to the initial structure; the final metric is the root mean squared error between the predicted and reference RMSD sequences. It

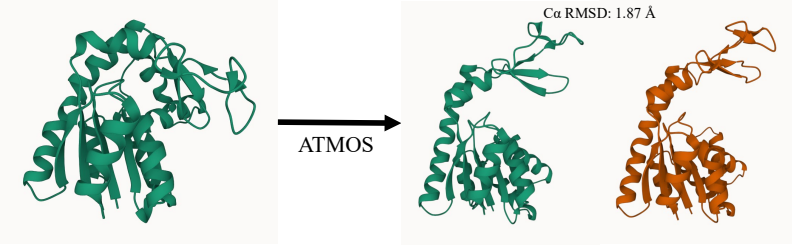


Figure 2: Case study on the adenylate kinase (AdK). ATMOS simulates the domain motion of the adenylate kinase from close (0 ns) to open (165 ns). This protein is not included in the training set.

directly measures how well the model reproduces the magnitude of conformational change over time. (ii) Decorr-TIC0 r: The TIC0 decorrelation time is defined as the first lag at which the normalized autocorrelation of the first TICA component falls below 0.5. We report the Pearson correlation between predicted and reference decorrelation times. This tests whether the relaxation timescale is captured. (iii) MSM-Transition Error: The metric is the root mean squared error between the transition matrices of the reference and predicted Markov state models (MSMs), defined on the same MSM state space. It quantifies how faithfully the model reproduces state-to-state kinetic probabilities. **For structural validity**, we report VAL-Cα, the fraction of generated conformations in which no pair of Cα atoms is either clashing or dissociated. Full details of all metrics are given in Appendix G.

As shown in Table 1, ATMOS consistently outperforms prior trajectory models across all evaluation dimensions, while matching or surpassing dedicated ensemble generators on most static distribution metrics. On ensemble-level properties, ATMOS reaches the highest Pearson correlations for both per-target RMSF and aggregate flexibility measures (pairwise RMSD, global RMSF). The Joint PCA $\mathcal{W}_2$ distance and the fraction of principal component similarities above 0.5 are the best among all methods, indicating better reproduction of dominant collective motions [11]. Additionally, ATMOS matches or surpasses AlphaFlow-MD + Templates variant of AlphaFlow on both weak and transient contact Jaccard similarities. Overall, this demonstrates that ATMOS generates conformational ensembles of comparable diversity and physical accuracy to dedicated ensemble generators.

The advantage of ATMOS is most pronounced on trajectory-specific kinetic metrics, which are inaccessible to static ensemble generators. ATMOS reduces the t-RMSD error to 1.79Å, a significant improvement over the next best trajectory model. On the Decorr-TIC0 r, ATMOS reaches $r = 0.25$, markedly better than the weak or even negative correlations of prior methods, though still far from the strong correlation, highlighting the challenge of capturing slow kinetic modes. Similarly, the MSM transition error of 0.24 is the lowest among trajectory models and approaches the reference value of 0.23. Finally, ATMOS achieves a structural validity score of 0.97, outperforming all baselines, indicating that the generated trajectories are essentially free of steric clashes or chain dissociation.

We compare trajectory-generation throughput on 1gnla01 for a 400 ns trajectory with 400 output frames at 1 ns intervals. All methods are run on a single RTX 4090 GPU. As shown in Table 2, ATMOS takes 0.174 hours, compared with 0.467 hours for ConfRover and more than 10 hours for MD, and uses about 15% of ConfRover's peak GPU memory. This comparison measures wall-clock time at a fixed trajectory length and output frequency. The speedup at matched thermodynamic or kinetic accuracy remains to be established.

Table 2: **Efficiency evaluation on the 1gnla01 protein (139 AA).** MD simulation is performed with Amber24 on GPU [27], employing an integration time step of 4 fs.

	Inference Time (Hour)	Peak Mem (GB)
MD	>10	<1
ConfRover	0.467	22.53
ATMOS	0.174	3.445

We further assess all-atom physical plausibility through geometric validation and force-field energy evaluation on 16 randomly sampled mdCATH test systems. The evaluation protocol and results are provided in Appendix H (Table 7). Additionally, we evaluate ATMOS on Adenylate Kinase (AdK), a canonical system exhibiting large-scale, functionally relevant conformational changes between a closed (PDB: 1AKE) and an open (PDB: 4AKE) state. As shown in Figure 2, starting from the closed

Table 3: **Evaluation on MISATO**. The best scores are highlighted in **bold** and the second-best scores are underlined. r: Pearson correlation; $\mathcal{W}_2$: 2-Wasserstein distance.

	Method	per-target RMSF $r \uparrow$	Root mean $\mathcal{W}_2$-dist $\downarrow$	Steric Clashes $\downarrow$	t-RMSD Error $\downarrow$
Ligand	NeuralMD	0.574	0.698	7.667	0.698
	DynamicBind	0.678	1.470	7.113	-
	Boltz-2	0.613	1.237	0.210	-
	ATMOS	**0.769**	**0.693**	**0.000**	**0.522**
Protein-Ligand	DynamicBind	0.193	2.288	52.74	-
	Boltz-2	0.534	1.992	0.020	-
	ATMOS	**0.654**	**1.370**	**0.000**	**0.435**

Table 4: **Contribution of pretrained Protenix components on mdCATH.** The best scores are highlighted in **bold** and the second-best scores are underlined.

Metrics	ATMOS (full)	ATMOS (ablated)	MDGen	ConfRover
Pairwise RMSD $r \uparrow$	0.89	**0.90**	0.65	0.69
Global RMSF $r \uparrow$	**0.87**	0.86	0.67	0.62
Root mean $\mathcal{W}_2$-dist. $\downarrow$	3.30	**2.99**	3.86	4.52
% PC-sim $> 0.5 \uparrow$	**45.3**	40.3	3.17	22.2
Weak contacts $J \uparrow$	**0.67**	0.54	0.57	0.43
Transient contacts $J \uparrow$	**0.34**	0.30	0.22	0.22
t-RMSD Error $\downarrow$	**1.79**	2.01	2.71	2.50
Decorr-TIC0 $r \uparrow$	**0.25**	0.134	0.07	-0.05
MSM-Trans Error $\downarrow$	**0.24**	0.26	0.28	0.30
VAL-C$\alpha \uparrow$	**0.97**	0.76	0.07	0.53

state, trajectories generated by our model evolve toward the open state without explicit biasing forces, indicating that the model captures the intrinsic conformational landscape and coordinated domain motions underlying AdK function.

4.2 Protein-Ligand Dynamics

For protein-ligand dynamics, we compare ATMOS against NeuralMD [17], DynamicBind [21], and Boltz-2 [25]. NeuralMD models only ligand dynamics within a fixed protein environment, whereas DynamicBind and Boltz-2 generate protein-ligand conformational ensembles. Following the same protocol used for mdCATH, we condition each method on the initial complex structure and generate a complete trajectory. For ensemble generators, we sample an independent set of conformations of equal size. Given the short 8ns duration of MISATO trajectories and the localized nature of binding pocket motions, we evaluate four aspects: (1) flexibility prediction via per-target RMSF Pearson correlation, (2) distributional accuracy via root mean 2-Wasserstein distance, (3) structural quality via steric clashes, and (4) trajectory fidelity via t-RMSD error. Full metric definitions are provided in Appendix G. We report separate results for ligand-only and whole-system evaluations where applicable. We note that Boltz-2 was trained on the complete MISATO dataset, which may constitute data leakage for our test set; we include it as a strong reference point.

As shown in Table 3, ATMOS achieves strong performance across both ligand-only and full protein-ligand evaluations. For ligand dynamics, ATMOS attains the best scores in all four metrics. These results demonstrate that our model generates ligand motions closely matching the reference distribution and temporal evolution. On the full protein–ligand system, ATMOS delivers the best scores in all four metrics. Although Boltz-2 also shows competitive performance on some metrics, it is trained on the entire dataset (potential data leakage) and does not produce trajectories. In summary, ATMOS not only excels at modeling ligand-in-pocket dynamics but also provides a strong framework for generating trajectories of the entire protein-ligand system.

4.3 Ablation Studies

Contribution of pretrained Protenix components. To assess the contribution of pretrained Protenix components, we evaluate an ablated variant in which the 48-layer pretrained Pairformer context trunk is removed and the diffusion decoder is trained from scratch. These changes are applied to both the coarse-scale keyframe generator and the fine-scale interpolator.

As shown in Table 4, this variant outperforms MDGen on nine of the ten reported metrics and ConfRover on all ten. The full model further improves structural validity (VAL-Cα: 0.76 to 0.97) and kinetic accuracy (Decorr-TIC0 r: 0.134 to 0.25; t-RMSD error: 2.01 to 1.79), whereas the ablated variant achieves better pairwise RMSD correlation and root mean $\mathcal{W}_2$ distance. These results suggest that the gains of ATMOS over these baselines cannot be attributed solely to pretrained Protenix components.

Table 5: **Ablation study on Chunk Stitching**.

p	Pairwise RMSD $r \uparrow$	Root mean $\mathcal{W}_2$-dist. $\downarrow$	t-RMSD Error $\downarrow$	Decorr-TIC0 $r \uparrow$	MSM-Trans Error $\downarrow$
0	0.87	2.55	1.83	0.10	0.25
5	0.89	3.30	1.79	0.25	0.24
10	0.90	2.99	1.80	0.20	0.24

Effect of chunk stitching. We examine the impact of the chunk stitching prefix length p (defined in Section 3.5) on trajectory quality, comparing $p = 0$ (i.e., restarting interpolation independently at each anchor frame), $p = 5$, and $p = 10$. As shown in Table 5, removing the prefix entirely ($p = 0$) causes a sharp drop in the Decorr-TIC$_0$ correlation (r falls from 0.25 to 0.10), confirming that temporal continuity across anchor boundaries is essential for faithfully capturing slow relaxation timescales. In contrast, the pairwise RMSD correlation and t-RMSD error are largely robust to changes in p, indicating that overall flexibility and conformational drift are primarily determined by the keyframe generator.

Interestingly, $p = 0$ yields a better Root mean $\mathcal{W}_2$ distance (2.55 vs 3.30 for $p = 5$). A possible explanation is that restarting interpolation at each anchor frame allows the interpolator to explore a broader conformational space between keyframes, inflating ensemble diversity at the cost of kinetic coherence. With $p = 5$ and $p = 10$, performance is nearly identical across all metrics, demonstrating that a short prefix of five fine-grained frames already provides sufficient temporal context.

5 Conclusion

We present ATMOS, a unified generative framework for simulating biomolecular dynamics that generalizes from monomeric proteins to complex biomolecular systems. The model achieves state-of-the-art performance on large-scale MD benchmarks, demonstrating its ability to generalize across diverse biological systems and paving the way toward foundational dynamics models. We provide further discussion of broader impacts and limitations in Appendix B and Appendix C, respectively.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their valuable comments and suggestions.

Funding Statement. This work was supported by BioGeometry through internship funding, computational resources, and research guidance. Liang Shi acknowledges financial aid from NeurIPS 2026 for conference participation.

Competing Interests Disclosures. The authors declare no competing interests.

References

[1] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

[2] D. Arthur, S. Vassilvitskii, et al. k-means++: The advantages of careful seeding. In *Soda*, volume 7, pages 1027–1035, 2007.

[3] X. Chen, Y. Zhang, C. Lu, W. Ma, J. Guan, C. Gong, J. Yang, H. Zhang, K. Zhang, S. Wu, et al. Protenix-advancing structure prediction through a comprehensive alphafold3 reproduction. *bioRxiv*, pages 2025–01, 2025.

[4] A. D. S. Costa, I. Mitnikov, F. Pellegrini, A. Daigavane, M. Geiger, Z. Cao, K. Kreis, T. Smidt, E. Kucukbenli, and J. Jacobson. Equijump: Protein dynamics simulation via so (3)-equivariant stochastic interpolants. *arXiv preprint arXiv:2410.09667*, 2024.

[5] P. Eastman, R. Galvelis, R. P. Peláez, C. R. Abreu, S. E. Farr, E. Gallicchio, A. Gorenko, M. M. Henry, F. Hu, J. Huang, et al. Openmm 8: molecular dynamics simulation with machine learning potentials. *The Journal of Physical Chemistry B*, 128(1):109–116, 2024.

[6] B. Feng, J. Zhang, X. Zhang, Z. Liu, and Y. Li. Biomd: All-atom generative model for biomolecular dynamics simulation. *arXiv preprint arXiv:2509.02642*, 2025.

[7] A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[8] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021.

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

[10] M. Hoffmann, M. K. Scherer, T. Hempel, A. Mardt, B. de Silva, B. E. Husic, S. Klus, H. Wu, J. N. Kutz, S. Brunton, and F. Noé. Deeptime: a python library for machine learning dynamical models from time series data. *Machine Learning: Science and Technology*, 2021.

[11] B. Jing, B. Berger, and T. Jaakkola. AlphaFold meets flow matching for generating protein ensembles. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22277–22303. PMLR, 21–27 Jul 2024.

[12] B. Jing, H. Stärk, T. Jaakkola, and B. Berger. Generative modeling of molecular dynamics trajectories. *Advances in Neural Information Processing Systems*, 37:40534–40564, 2024.

[13] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[14] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[15] L. Klein, A. Foong, T. Fjelde, B. Mlodozeniec, M. Brockschmidt, S. Nowozin, F. Noé, and R. Tomioka. Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. *Advances in Neural Information Processing Systems*, 36:52863–52883, 2023.

[16] S. Lewis, T. Hempel, J. Jiménez-Luna, M. Gastegger, Y. Xie, A. Y. Foong, V. G. Satorras, O. Abdin, B. S. Veeling, I. Zaporozhets, et al. Scalable emulation of protein equilibrium ensembles with generative deep learning. *Science*, 389(6761):eadv9817, 2025.

[17] S. Liu, W. Du, H. Xu, Y. Li, Z. Li, V. Bhethanabotla, D. Yan, C. Borgs, A. Anandkumar, H. Guo, et al. A multi-grained symmetric differential equation model for learning protein-ligand binding dynamics. *Nature Communications*, 2025.

[18] J. Lu, X. Chen, S. Z. Lu, A. Lozano, V. Chenthamarakshan, P. Das, and J. Tang. Aligning protein conformation ensemble generation with physical feedback. In A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 40436–40451. PMLR, 13–19 Jul 2025.

[19] J. Lu, X. Chen, S. Z. Lu, C. Shi, H. Guo, Y. Bengio, and J. Tang. Structure language models for protein conformation generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[20] J. Lu, B. Zhong, Z. Zhang, and J. Tang. Str2str: A score-based framework for zero-shot protein conformation sampling. In *The Twelfth International Conference on Learning Representations*, 2024.

[21] W. Lu, J. Zhang, W. Huang, Z. Zhang, X. Jia, Z. Wang, L. Shi, C. Li, P. G. Wolynes, and S. Zheng. Dynamicbind: predicting ligand-specific protein-ligand complex structure with a deep equivariant generative model. *Nature Communications*, 15(1):1071, 2024.

[22] R. T. McGibbon, K. A. Beauchamp, M. P. Harrigan, C. Klein, J. M. Swails, C. X. Hernández, C. R. Schwantes, L.-P. Wang, T. J. Lane, and V. S. Pande. Mdtraj: a modern open library for the analysis of molecular dynamics trajectories. *Biophysical journal*, 109(8):1528–1532, 2015.

[23] A. Mirarchi, T. Giorgino, and G. De Fabritiis. mdcath: A large-scale md dataset for data-driven computational biophysics. *Scientific Data*, 11(1):1299, 2024.

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

[25] S. Passaro, G. Corso, J. Wohlwend, M. Reveiz, S. Thaler, V. R. Somnath, N. Getz, T. Portnoi, J. Roy, H. Stark, et al. Boltz-2: Towards accurate and efficient binding affinity prediction. *BioRxiv*, 2025.

[26] S. Röblitz and M. Weber. Fuzzy spectral clustering by pcca+: application to markov state models and data classification. *Advances in Data Analysis and Classification*, 7(2):147–179, 2013.

[27] R. Salomon-Ferrer, A. W. Gotz, D. Poole, S. Le Grand, and R. C. Walker. Routine microsecond molecular dynamics simulations with amber on gpus. 2. explicit solvent particle mesh ewald. *Journal of chemical theory and computation*, 9(9):3878–3888, 2013.

[28] M. Schreiner, O. Winther, and S. Olsson. Implicit transfer operator learning: Multiple time-resolution models for molecular dynamics. *Advances in Neural Information Processing Systems*, 36:36449–36462, 2023.

[29] S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[30] Y. Shen, L. Wang, H. Yuan, Y. Wang, B. Yang, and Q. Gu. Simultaneous modeling of protein conformation and dynamics via autoregression. In *ICML 2025 Generative AI and Biology (GenBio) Workshop*, 2025.

[31] N. Shoghi, Y. Liu, Y. Shen, R. Brekelmans, P. Li, and Q. Gu. Scalable spatio-temporal SE(3) diffusion for long-horizon protein dynamics. In *The Fourteenth International Conference on Learning Representations*, 2026.

[32] T. Siebenmorgen, F. Menezes, S. Benassou, E. Merdivan, K. Didi, A. S. D. Mourão, R. Kitel, P. Liò, S. Kesselheim, M. Piraud, et al. Misato: machine learning dataset of protein–ligand complexes for structure-based drug discovery. *Nature computational science*, 4(5):367–378, 2024.

[33] Y. Vander Meersche, G. Cretin, A. Gheeraert, J.-C. Gelly, and T. Galochkina. Atlas: protein flexibility description from atomistic molecular dynamics simulations. *Nucleic acids research*, 52(D1):D384–D392, 2024.

[34] R. Wang, X. Fang, Y. Lu, C.-Y. Yang, and S. Wang. The pdbbind database: methodologies and updates. *Journal of medicinal chemistry*, 48(12):4111–4119, 2005.

[35] Y. Wang, L. Wang, Y. Shen, Y. Wang, H. Yuan, Y. Wu, and Q. Gu. Protein conformation generation via force-guided se (3) diffusion models. *arXiv preprint arXiv:2403.14088*, 2024.

[36] C. J. Williams, J. J. Headd, N. W. Moriarty, M. G. Prisant, L. L. Videau, L. N. Deis, V. Verma, D. A. Keedy, B. J. Hintze, V. B. Chen, et al. Molprobity: more and better reference data for improved all-atom structure validation. *Protein science*, 27(1):293–315, 2018.

[37] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[38] K. Xu, J. Wang, M. Liu, K. Zhou, S. Lin, W. Li, L. Shi, P. Zhou, H. Liu, and X. Yao. Efficient generation of protein and protein–protein complex dynamics via se(3)-parameterized diffusion models. *Journal of Chemical Information and Modeling*, 65(22):12366–12376, 2025.

[39] Y. Xu, D. Wang, Z. Zhou, T. Yu, and M. Chen. TEMPO: Temporal multi-scale autoregressive generation of protein conformational ensembles. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[40] S. Zheng, J. He, C. Liu, Y. Shi, Z. Lu, W. Feng, F. Ju, J. Wang, J. Zhu, Y. Min, et al. Predicting equilibrium distributions for molecular systems with deep learning. *Nature Machine Intelligence*, 6(5):558–567, 2024.

A Lay Summary

Proteins are constantly moving, and their movements help determine how they work and interact with medicines. Conventional computer simulations can reveal this motion but are costly to run. We introduce ATMOS, an AI model that learns from existing simulations to generate molecular movies. Starting from a molecular structure, it keeps a running memory of earlier motion to guide what comes next. It first generates snapshots at widely spaced times, then fills in the motion between them. In tests on proteins and proteins interacting with small molecules, ATMOS reproduced several properties of simulated motion better than competing AI methods. It could help researchers explore molecular behavior more efficiently, although it does not yet replace conventional simulations or reliably capture very slow molecular changes.

B Broader Impacts

ATMOS has significant broader impacts for computational biology and drug discovery. It provides a general-purpose framework for modeling biomolecular dynamics that could accelerate therapeutic discovery and enhance biological understanding. However, ethical concerns, such as potential misuse in drug development, warrant careful consideration. Responsible deployment and transparency are essential to maximize benefits and mitigate risks.

C Limitations

While ATMOS achieves strong performance across diverse benchmarks, several limitations point to promising directions for future work. First, our current approach trains separate models for different temporal scales with a fixed stride, rather than learning dynamics across a continuum of timescales within a single unified model. Second, for protein-ligand systems, the largest publicly available benchmark, MISATO, provides trajectories of only 8 ns, which primarily probe local pocket flexibility rather than multi-state binding transitions or slow dissociation events. The absence of large, transferable, long-timescale protein-ligand MD datasets currently prevents a rigorous evaluation of such global kinetic phenomena. Third, we train and evaluate on each benchmark independently, as varying simulation protocols and temperature ranges across datasets hinder straightforward unification. Establishing a single dynamics foundation model that generalizes across systems, temperatures, and timescales remains an important objective that we leave for future investigation.

Our throughput benchmark fixes the trajectory length and output frequency on a single protein. A fidelity-matched comparison with MD is an important direction for future work. Such an evaluation would measure the wall-clock cost of reaching comparable accuracy on specified thermodynamic and kinetic observables.

D Architectural Details

This section provides implementation details for the key modules introduced in Section 3.3.

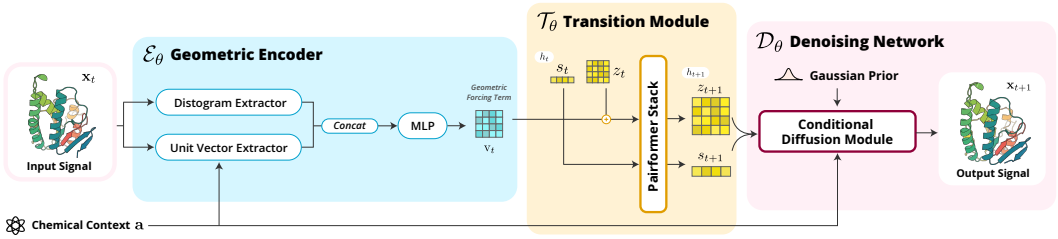


Figure 3: **Detailed architecture of our framework.**

D.1 Context and Input Encoding

We initialize the context embedding layer $\mathbf{e}_\theta$ with the pre-trained trunk module of Protenix [3] (an open-source reproduction of AlphaFold3 [1]), which includes the Input Feature Embedder, MSA Module and Pairformer. This module is frozen during training, similar to ConfRover [30].

The geometric encoder $\mathcal{E}_\theta$, which is a MLP, maps the current conformation X_t into a forcing term $\mathbf{v}_t$. $\mathbf{v}_t$ encodes pairwise geometric features.

Pairwise geometric features. For each token pair (i, j), we compute invariant features that encode their relative geometry. For protein residues, the token center is taken as the Cβ atom (Cα for glycine); for ligand atoms, the atom itself serves as the center. We compute (1) a distance histogram between the two centers, and (2) unit vectors that express the displacement vector in the local coordinate frame of each token (constructed from backbone atoms for residues and neighboring atoms for ligands). These features are concatenated and projected:

$$\mathbf{v}_t[i, j] = \mathrm{MLP}\big(\mathrm{concat}(\mathrm{Distogram}_{ij},\ \mathrm{UnitVectors}_{ij})\big). \tag{14}$$

D.2 State Transition

The core propagator of ATMOS is the state transition function $\mathcal{T}_\theta$, which evolves the latent state $\mathbf{h}$ from t to $t + 1$. We instantiate $\mathcal{T}_\theta$ using the Pairformer module [1]. We use 4 Pairformer blocks in our implementation. The architectural hyperparameters are listed in Table 6.

Table 6: Architectural hyperparameters for the Pairformer.

Hyperparameter	Value
Number of Pairformer blocks	4
Single representation dimension	384
Pair representation dimension	128
Triangle multiplication hidden dimension	128
Number of triangle attention heads	4
Transition layer expanding factor	4
Pair attention dropout rate	0.25

Algorithm 1 Conditional Diffusion Sampling

Input: Diffusion steps N_{step}, latent state $\mathbf{h}$, chemical context $\mathbf{a}$, sampling hyperparameters $\lambda, \eta, \gamma_0, \gamma_{\text{min}}$.
Output: Generated coordinates $\mathbf{x}$.
noise_schedule $\leftarrow$ inference_noise_scheduler(N_{step})
$\mathbf{x} \leftarrow$ noise_schedule[0] $*\mathcal{N}(0, I)$ # Initialize with noise
for $i \leftarrow 1$ to len(noise_schedule) **do**
  $\gamma^{\text{last}}, \gamma \leftarrow$ noise_schedule[$i-1$], noise_schedule[i]
  $\mathbf{x} \leftarrow$ centre_random_augmentation($\mathbf{x}$) # Apply random rigid augmentation
  $\gamma' \leftarrow \gamma_0$ if $\gamma > \gamma_{\text{min}}$ else 0
  $t_{\text{hat}} \leftarrow \gamma^{\text{last}} * (\gamma' + 1)$
  $\mathbf{x}_{\text{noisy}} \leftarrow \mathbf{x} + \lambda\sqrt{t_{\text{hat}}^2 - (\gamma^{\text{last}})^2} * \mathcal{N}(0, I)$
  $\mathbf{x}_{\text{denoised}} \leftarrow \mathcal{D}_\theta(\mathbf{x}_{\text{noisy}}, \mathbf{a}, \mathbf{h}, t_{\text{hat}})$ # Conditional denoising
  $\delta \leftarrow (\mathbf{x}_{\text{noisy}} - \mathbf{x}_{\text{denoised}})/t_{\text{hat}}$
  $\Delta t \leftarrow \gamma - t_{\text{hat}}$
  $\mathbf{x} \leftarrow \mathbf{x}_{\text{noisy}} + \eta * \Delta t * \delta$
end for
Return $\mathbf{x}$

D.3 Structure Decoding

We model the generation of the next frame $\mathbf{x}_{t+1}$ as a reverse diffusion process, with a denoising network $\mathcal{D}_\theta$ that predicts the clean structure from a noisy state. The diffusion decoder $\mathcal{D}_\theta$ is initialized from the weights of Protenix [3]. Together with the context embedding layer $\mathbf{e}_\theta$, this provides a prior on plausible biomolecular geometries, enabling data-efficient learning of dynamics. We parameterize the decoder $\mathcal{D}_\theta$ using an EDM-style diffusion process [14]. The conditional diffusion sampling procedure is detailed in Algorithm 1. We use $N_{\text{step}} = 50$ diffusion steps, a step scaling factor $\eta = 1.5$, and a noise scaling schedule parameter $\gamma_0 = 0.8$ with a minimum $\gamma_{\text{min}} = 1.0$. The noise scaling factor λ is set to 1.5 to encourage diversity.

E MISATO Dataset Preprocessing

We preprocess the MISATO dataset following a procedure similar to Boltz-2 [25], resulting in 12,662 systems for training, 1,258 for validation, and 1,276 for testing.

The preprocessing consists of several steps. First, we download the MD restart file (to extract bond topology) and the MD result file (which contains trajectory coordinates and can be used to infer chain_ids). Next, we generate a (.prmtop, .nc) pair for each protein–ligand complex. The .prmtop file holds topology information and the .nc file stores the trajectory coordinates. These files are loaded with MDTraj [22], and each frame is exported as an individual PDB file. This method preserves complete topology (including bonds) and ensures a robust ligand representation, whereas directly constructing a PDB from the MD result file can lose atoms and introduce parsing inconsistencies.

We then add CCD (Chemical Component Dictionary) information to the loaded (.prmtop, .nc) pair, and convert them to (.pdb, .xtc) format. Systems that do not contain a MOL (ligand) residue are excluded. We also remove systems where the ligand's atomic elements differ from those in the corresponding PDBBind MOL2 files, ensuring chemical consistency. Additionally, trajectories in which the ligand drifts more than 12Å from the protein at any frame are filtered out.

Furthermore, we remove approximately 500 systems that cannot be processed by the Protenix [3] data pipeline due to unexpected errors, indicating potential format, content, or geometry issues. (We first converts the PDB file to mmCIF format and then applies Protenix's standard processing.) The processed MISATO data and preprocessing code are publicly available; see Appendix I.

The PDB IDs of the randomly sampled test systems are as follows: `16PK, 1AO8, 1D2S, 1DHJ, 1DRK, 1F73, 1IK4, 1JQD, 1NPO, 1WE2, 1XK9, 1Y2F, 1YYS, 2BET, 2EXG, 2GKL, 2HXM, 2I6B, 2LTO, 2NQI, 2PZE, 2QIC, 2RA6, 2RR4, 2V54, 2WJG, 2Y4K, 2Y4M, 2ZA3, 3BLU, 3C39, 3D9M, 3EOR, 3F69, 3G3N, 3GPO, 3HII, 3JUQ, 3MAG, 3ME9, 3MEU, 3QVU, 3S0B, 3TNE, 3ZOT, 4A22, 4ASK, 4AYR, 4B0C, 4DPU, 4DZY, 4G0Q,`

4H7Q, 4I7B, 4KM2, 4KQO, 4KQR, 4KXL, 4L0L, 4MRE, 4NA4, 4NRQ, 4O1L, 4O24, 4ODL, 4UFH, 4X11, 4XAQ, 4YXD, 4ZB6, 5C13, 5C4K, 5ECT, 5EFC, 5F1R, 5H5Q, 5J6D, 5MXX, 5NPB, 5NPC, 5NPD, 5TPG, 5TT8, 5ZDC, 6BOE, 6C3U, 6CQ5, 6CW4, 6DYN, 6DYS, 6FAC, 6FAM, 6FNR, 6FNT, 6MIN, 6NP3, 6OA3, 6OIO, 6P12, 6PYA.

F Training

All experiments are conducted on 8 NVIDIA RTX 4090 GPUs with distributed data parallelism.

Batching. We employ a three-level batching strategy. First, each GPU processes one independent trajectory per training step. Second, for each trajectory we sample four subsequences (with random start indices and fixed stride) and compute the loss over all four in parallel. Third, for each target frame in a subsequence we denoise 12 independent noise samples (diffusion batch size=12) to stabilize the diffusion objective.

Optimization. We use the Adam optimizer with a learning rate of 5×10^{-4} and train for 40k steps.

Cropping. During training we crop each system to a fixed size of 256 tokens. For mdCATH we take a contiguous crop of 256 residues; for MISATO we apply a spatial interface crop that includes the ligand and its surrounding protein residues.

Loss weighting. The training objective is a weighted sum of the reconstruction loss $\mathcal{L}_{\text{recon}}$ and the latent fidelity loss $\mathcal{L}_{\text{disto}}$ (defined in Section 3.4). We set $\lambda_{\text{recon}} = 4.0$ and $\lambda_{\text{disto}} = 0.03$. The reconstruction loss also includes an atom-type weight $w_{\text{entity}}(i)$, set to 11.0 for ligand atoms and 1.0 for protein atoms. We use $\sigma_{\text{data}} = 16$ following AlphaFold 3 [1].

G Evaluation Metrics

G.1 mdCATH

G.1.1 Ensemble-level Properties

Following AlphaFlow [11], we evaluate the generated ensembles from three aspects: predicting flexibility, distributional accuracy, and ensemble observables. For each protein, the generated trajectory is compared against all replicas of the corresponding MD simulations.

Predicting flexibility. We assess the ability of a model to reproduce the dynamic fluctuations observed in MD simulations using three correlation-based metrics:

- **Pairwise RMSD** r: For each protein, we compute the average Cα-RMSD between every pair of conformations in the ensemble, which serves as a scalar measure of overall flexibility. The Pearson correlation between these pairwise RMSD values and those of the reference MD ensemble is reported.
- **Global RMSF** r: We calculate the root-mean-square fluctuation (RMSF) of each residue (represented by its Cα atom) across the ensemble. All per-residue RMSF values from all test proteins are concatenated, and the Pearson correlation with the reference RMSF values is computed.
- **Per-target RMSF** r: For each individual protein, we compute the Pearson correlation between the predicted and reference per-residue RMSF vectors. The median of these per-protein correlations across the test set is reported.

Distributional accuracy. We assess the model's ability to reproduce the correct distribution of atomic positions using four metrics based on the 2-Wasserstein distance ($\mathcal{W}_2$):

- **Root mean $\mathcal{W}_2$-dist**: The root mean Wasserstein distance (RMWD) between ensembles is defined as:

$$\text{RMWD}(\mathcal{X}, \mathcal{Y}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \mathcal{W}_2^2(\mathcal{N}[\mathcal{X}_i], \mathcal{N}[\mathcal{Y}_i])}, \tag{15}$$

  where $\mathcal{N}[\mathcal{X}_i]$ denotes a 3D Gaussian fitted to the positional distribution of the i-th atom in ensemble $\mathcal{X}$. We report the median value over the test set.

- **MD PCA $\mathcal{W}_2$-dist**: To assess the similarity of collective motions, we project the Cα atom positions of the generated ensemble onto the first two principal components (PCs) derived from the reference MD ensemble. The 2-Wasserstein distance (in Angstrom units of RMSD) between the projected distributions of the generated and reference ensembles in this 2D PCA space is computed and reported.
- **Joint PCA $\mathcal{W}_2$-dist**: We perform PCA on the combined set of Cα atom positions from the generated and reference conformations (equally weighted) to obtain a joint set of principal components. Both ensembles are then projected onto the first two joint PCs, and the 2-Wasserstein distance between their projected Cα distributions is computed.
- **% PC-sim > 0.5**: For each protein, we compute the unsigned cosine similarity between the top principal component (derived from Cα positions) of the generated ensemble and that of the reference MD ensemble. A value exceeding 0.5 indicates that the dominant collective motion is successfully captured. We report the percentage of test proteins for which this condition holds.

Ensemble observables. To evaluate whether the model captures biologically relevant contact dynamics, we compute two sets of inter-residue contacts and compare them to the reference MD ensemble:

- **Weak contacts J:** We identify pairs of Cα atoms that are in contact (distance ≤ 8Å) in the crystal structure but dissociate (distance > 8Å) in more than 10% of the ensemble structures. The Jaccard similarity between the set of such weak contacts in the generated ensemble and that in the reference MD ensemble is reported.
- **Transient contacts J:** Conversely, we identify pairs of Cα atoms that are not in contact (distance > 8Å) in the crystal structure but associate (distance ≤ 8Å) in more than 10% of the ensemble structures. The Jaccard similarity between the generated and reference sets of transient contacts is reported.

G.1.2 Trajectory-specific Kinetics

t-RMSD Error. We report the t-RMSD Error, a trajectory-level metric that quantifies a model's ability to reproduce the magnitude of conformational change over time. This metric is computed by (1) calculating the RMSD of each frame to the initial frame, and (2) computing the RMSE between the resulting RMSD sequences of the predicted and ground-truth trajectories.

Decorr-TIC0 r. To apply TICA, each frame of a trajectory is featurized by the sine and cosine of the backbone ϕ and ψ dihedral angles. These features are then projected onto the first two time-lagged independent components (TIC0 and TIC1) using the `deeptime` [10] library, with a lag time of 1ns.For each trajectory, we compute the normalized autocorrelation function of the first component (TIC0) via `statsmodels` [29]. The TIC0 decorrelation time is defined as the smallest lag (in frames) at which this autocorrelation drops below 0.5. Finally, we report the Pearson correlation coefficient between the decorrelation times of the predicted and reference trajectories across the test set, denoted as Decorr-TIC0 r. A higher correlation indicates better agreement in the timescale of the dominant slow relaxation process.

MSM-Transition Error. We first cluster the two-dimensional features (TIC0 and TIC1) of the reference trajectory into 100 microstates using the KMeans++ [2] algorithm. A Markov state model (MSM) is then constructed from the microstate assignments with a lag time of 1 ns, and the PCCA+ [26] algorithm coarse-grains these microstates into 10 metastable states. To ensure full connectivity, any microstates classified as inactive are reassigned to the nearest active metastable cluster. The resulting clustering and PCCA+ models, derived exclusively from the reference trajectory, define a fixed mapping from the TIC space into the 10 metastable states.

We apply this mapping to both the generated and reference trajectories. For each trajectory, transition counts between metastable states are tallied over consecutive frames at a 1 ns lag, producing a transition count matrix, which is then normalized to a row-stochastic transition probability matrix, denoted $\mathbf{T}_{\text{pred}}$ and $\mathbf{T}_{\text{ref}}$, respectively. The MSM-Transition Error is the root mean squared error between these two matrices. A lower error indicates that the generated dynamics reproduce the reference state-to-state transition probabilities more faithfully.

Table 7: **Additional physical plausibility metrics on mdCATH.** Evaluation is performed on 16 randomly sampled test systems, and all values are reported as means. The best scores among the evaluated generative methods, excluding the MD reference, are highlighted in **bold**. Rama. denotes Ramachandran. Force-field energies are evaluated after a short energy minimization with `maxIterations=100`.

Method	Rama. outliers (%) ↓	Rama. favored (%) ↑	Rotamer outliers (%) ↓	RMS (bonds) ↓	RMS (angles) ↓	Clashscore ↓	Force-field energy ↓
MD reference	1.100	92.36	2.863	0.036	3.654	0.953	-7962
AlphaFlow	0.617	96.72	0.776	0.031	2.350	25.13	-7464
EBA	0.202	**97.61**	1.328	0.100	4.394	9.413	1.5×10^8
MDGen	7.944	78.70	4.799	0.085	6.112	247.7	2×10^{21}
ConfRover	3.955	89.48	4.143	0.053	4.165	147.3	3×10^{26}
TEMPO	11.66	20.05	N/A	0.372	22.61	476.1	N/A
ATMOS	**0.192**	97.48	**0.394**	**0.014**	**1.786**	**6.204**	$\mathbf{-7931}$

G.1.3 Structural Validity

VAL-Cα. We assess the structural stability of generated protein conformations over long timescales using a geometric validity criterion. Following ConfDiff [35], a conformation is deemed valid if two conditions hold: (I) no pair of Cα atoms is closer than 3.0 (steric clash check), and (II) the Cα–Cα distance of each pair of consecutive residues does not exceed 4.19 (backbone continuity check). We report the fraction of frames in a trajectory that satisfy both conditions, denoted VAL-Cα.

G.2 MISATO

The MISATO dataset captures the coupled dynamics of protein–ligand complexes, with the ligand's motion confined to the protein's binding pocket. For evaluation, we crop the pocket–ligand region. The pocket is defined as all residues where any heavy atom lies within 10Å of any ligand heavy atom, following the threshold used in AlphaFold 3 [1].

We assess four aspects: flexibility prediction (per-target RMSF Pearson correlation), distributional accuracy (root mean 2-Wasserstein distance), steric clashes, and trajectory accuracy (t-RMSD Error). Except for steric clashes, the definitions align with those in Section G.1, but extended to include all atoms (protein and ligand).

Steric clashes are reported as the average number of clashes per generated trajectory. For ligand-only evaluation, we count clashes between ligand heavy atoms. For whole-complex evaluation, we count clashes between protein heavy atoms and ligand heavy atoms. A clash is defined as two heavy atoms being closer than 1.1Å again following AlphaFold 3 [1].

H Additional Physical Plausibility Evaluation

Evaluation protocol. To complement the structural validity evaluation in the main paper, we assess the all-atom physical plausibility of generated structures using direct geometric and energetic checks. Because these checks are computationally expensive when applied frame by frame, we evaluate them on 16 randomly sampled mdCATH test systems. All reported values are means.

We use MolProbity [36] to evaluate Ramachandran outliers and favored regions, rotamer outliers, root-mean-square deviations of bond lengths and bond angles from ideal geometry, and clashscore, which measures severe steric overlaps per 1,000 atoms. In addition, we use OpenMM [5] to compute force-field energies. For energy evaluation, hydrogens and OXT atoms are added, and a short energy minimization with `maxIterations=100` is performed before computing the energies. The same protocol is applied to all methods and the MD reference.

Results and interpretation. As shown in Table 7, ATMOS achieves the best scores among the evaluated generative methods on most geometric metrics, including Ramachandran outliers, rotamer outliers, bond-length and bond-angle deviations, and clashscore. Its mean force-field energy after minimization is close to that of the MD reference (-7931 versus -7962), although its clashscore remains higher (6.204 versus 0.953).

These results provide complementary evidence that ATMOS improves all-atom physical plausibility in terms of local geometry, steric consistency, and force-field energy under the same evaluation protocol. These checks do not, by themselves, establish full thermodynamic correctness, accurate long-timescale sampling, or the ability to replace explicit MD simulations.

I Code, Data, and Model Availability

Released assets. The ATMOS source code and instructions for installation, training, inference, and evaluation are available at `https://github.com/VectorShi/ATMOS`. Four trained model checkpoints and their configurations are available at `https://modelscope.cn/models/SmartVector/ATMOS`. Processed mdCATH and MISATO trajectories, loader indices, data splits, and precomputed protein MSAs are available at `https://modelscope.cn/datasets/SmartVector/ATMOS-dataset`. The repository documents the asset layout, preprocessing procedures, and evaluation conventions in `docs/assets.md` and the dataset-specific guides.

Licenses and attribution. ATMOS builds on Protenix [3] and uses its v0.5.0 initialization weights. The Protenix code and these model parameters are distributed under Apache-2.0. The source mdCATH [23] and MISATO [32] datasets are distributed under CC BY 4.0; our processed subsets retain attribution to their original creators. ATMOS source code is distributed under Apache-2.0, except for components covered by their own licenses. In particular, the trajectory conversion script adapted from MISATO is distributed under LGPL-2.1-or-later. Original copyright and license notices are retained in the repository and summarized in `THIRD_PARTY_NOTICES.md`.

PDBbind v2020 [34] is used during MISATO preprocessing and must be obtained separately under the data provider's terms. The release does not redistribute the original PDBbind files. Source-code licensing does not replace the applicable terms for separately distributed datasets, MSAs, model weights, or common chemical data.

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: Yes. Our main contributions are detailed in Sec. 1. Also see Sec. 4 for more experimental evidence.

   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: Yes, please see Appendix. C for limitations.

   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: [N/A]

Justification: The paper does not include theoretical results.

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: Yes. We provide a clear and detailed description of the architecture in Sec. 3 and Appendix D, along with the necessary training configurations in Sec. F. All datasets used in our experiments are also publicly accessible.

   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: The ATMOS code and execution instructions are publicly available on GitHub, and the trained checkpoints and processed datasets are available on ModelScope. Appendix I provides the corresponding links. The supplied scripts cover ATMOS training, inference, and benchmark evaluation on mdCATH and MISATO.

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: Please see Sec. 4 and Sec. F.

   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: [No]

   Justification: Following previous work, we do not report error bars.

   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: Yes. Please see Sec. F.

   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: We followed 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: Yes. Please see Appendix. B

    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: We do not foresee any high risk for misuse of this work.

    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: We cite the original creators and specify the licenses and access conditions of the principal assets in Appendix I. The code repository retains upstream copyright and license notices and documents third-party components.

    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: We release the ATMOS source code, four trained model checkpoints with configurations, and processed mdCATH and MISATO data with MSA features and split indices. The accompanying documentation describes asset organization, preprocessing, training, inference, and evaluation. See Appendix I for access and attribution, and Appendix C for limitations.

    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 paper does not involve crowdsourcing nor 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 paper does not involve crowdsourcing nor research with human subjects.

    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 method development in this research does not involve LLMs as any important, original, or non-standard components.

    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.