Simultaneous Modeling of Protein Conformation and Dynamics via Autoregression

Yuning Shen[1⋆], **Lihao Wang**[1⋆], **Huizhuo Yuan**[1], **Yan Wang**[2†],
Bangji Yang[3†], **Quanquan Gu**[1‡]

[1]ByteDance Seed
[2]School of Mathematical Sciences, Tongji University
[3]Department of Automation, Tsinghua University
{yuning.shen,quanquan.gu}@bytedance.com,

Abstract

Understanding protein dynamics is critical for elucidating their biological functions. The increasing availability of molecular dynamics (MD) data enables the training of deep generative models to efficiently explore the conformational space of proteins. However, existing approaches either fail to explicitly capture the temporal dependencies between conformations or do not support direct generation of time-independent samples. To address these limitations, we introduce CONFROVER, an autoregressive model that simultaneously learns protein conformation and dynamics from MD trajectories, supporting both time-dependent and time-independent sampling. At the core of our model is a modular architecture comprising: (i) an *encoding layer*, adapted from protein folding models, that embeds protein-specific information and conformation at each time frame into a latent space; (ii) a *temporal module*, a sequence model that captures conformational dynamics across frames; and (iii) an SE(3) diffusion model as the *structure decoder*, generating conformations in continuous space. Experiments on ATLAS, a large-scale protein MD dataset of diverse structures, demonstrate the effectiveness of our model in learning conformational dynamics and supporting a wide range of downstream tasks. CONFROVER is the first model to sample both protein conformations and trajectories within a single framework, offering a novel and flexible approach for learning from protein MD data.

1 Introduction

Proteins are flexible molecules that can adopt multiple structures, called *conformations*. Their ability to transition between different conformations enables biological processes critical to life. Characterizing the behavior of a protein, including its *(1) dynamic motions, (2) conformational distribution, and (3) transitions between different states*, is crucial for understanding its function and guiding the design of novel proteins [5, 12, 34]. Molecular dynamics (MD) simulations are the "gold standard" for studying protein conformational changes [8, 21, 33]. These simulations use physical models to describe the energy of a protein conformation and the forces acting on its atoms. By simulating atomic motion through classical mechanics and iteratively sampling conformations over time, MD enables researchers to explore different conformations, approximating the conformational distribution at equilibrium, and gain mechanistic insights in protein functions. However, MD simulations are both computationally expensive and technically challenging due to long simulation times and the tendency to become trapped in local energy minima.

These challenges have motivated the use of deep generative models to study proteins, leveraging the rich conformational and dynamic information provided by large-scale MD datasets [28, 43]:

⋆Equal contribution. †Work done during their internship at ByteDance. ‡Corresponding Author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

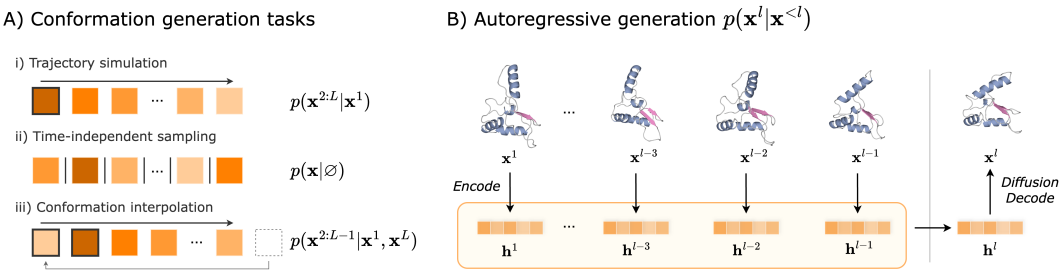

Figure 1: Key ideas of CONFROVER. (A) Conformation generation tasks with various conditioning configurations. Each block denotes a frame and arrows indicates the sequential dependencies among frames from autoregressive formulation. Initial conditioning frames are outlined in black. In conformation interpolation, the last frame is repositioned and prepended to the first frame for proper sequential dependencies. (B) CONFROVER models each frame as a conditional distribution given preceding frames. Sequential dependencies are captured through latent variables $\mathbf{h}$, and conformations are sampled from a diffusion decoder, conditioned on the updated latent.

(1) *Generating the dynamic motions of proteins is a direct analog to MD simulation.* Pioneering works modeled this by learning transition probabilities of future conformations from the current state [9, 22, 38]. However, MD trajectories are often non-Markovian due to partially observed coordinates (e.g., protein-only atoms) and environmental coupling (e.g., using Langevin thermostats). To mitigate this, Cheng et al. [7] incorporated higher-order information using multiple context frames, though this requires fix context windows and limits flexibility. Jing et al. [19] instead modeled the joint distribution over the entire trajectory, capturing complex dependencies among frames. Due to training on fixed-length trajectories and the non-autoregressive design, their model has limited inference-time flexibility that cannot generate variable-length trajectories. Li et al. [27] introduced an autoregressive approach for flexible trajectory extension, but its deterministic formulation cannot capture trajectory distributions or generate diverse samples.
(2) *Learning the conformational distribution enables sampling time-independent conformations.* Several methods [18, 25, 35, 44, 49] train diffusion- or flow-based generative models on conformation ensembles from MD simulation data, bypassing the need for sequential sampling. While effective for generating samples in parallel, they disregard temporal information in MD trajectories and therefore cannot simulate physical motion of proteins.
(3) *Conformation interpolation generate transition pathways between different states.* Recent works [11, 19] have extended generative modeling to conformation interpolation, where the goal is to generate plausible intermediate samples between known start and end conformational states. Jing et al. [19] framed interpolation as a conditional trajectory generation task, but it requires training task-specific model and has not been evaluated on large proteins.

As these generative problems all stem from the same underlying physical principles and involve sampling from a protein's conformational space, a natural question arises: *can we develop a general framework to learn all of these objectives?*

We present CONFROVER, a framework for simultaneous learning protein conformation distribution and dynamics from MD trajectory data (Figure 1). Our key observation is that, for an MD trajectory $\mathbf{x}^{1:L}$ of length L, by adopting a general autoregressive formulation, $p(\mathbf{x}^{1:L}) = \prod_{l=1}^{L} p(\mathbf{x}^l|\mathbf{x}^{<l})$, where $\mathbf{x}^{<l}$ denotes all preceding frames of $\mathbf{x}^l$ in the sequence, we can unify multiple generation objectives as instances of frame generation: (1) generating future frame conditioned on all previous frames, suitable for simulating non-Markovian dynamics; (2) unconditional single-frame generation, $p(\mathbf{x}|\varnothing)$, corresponding to time-independent conformation sampling; (3) flexible frame sequence ordering redefines the dependency structure, enabling tasks such as conformation interpolation.

Our contributions are summarized as follows:

- We introduce a simple yet general framework to learn both the conformational distribution and dynamics from MD data, supporting multiple generation tasks including trajectory simulation, time-independent conformation sampling, and conformation interpolation.
- We design a modular architecture that captures temporal dependencies in latent space using efficient causal transformers (i.e., Llama [42]), and directly models conformations in continuous SE(3) space using a diffusion decoder, avoiding discretization structure into tokens.

- Experiments show strong capabilities of CONFROVER: it outperforms MDGEN [19] in trajectory simulation, matches the performance of ALPHAFLOW[18] and CONFDIFF[44] in time-dependent generation, and can effectively sample conformations interpolating two endpoints.

2 Background

2.1 Data Generation from Molecular Dynamics

Molecular dynamics describes the motion of molecules through Newtonian mechanics $M\ddot{\mathbf{x}} = -\nabla U(\mathbf{x})$, where $\mathbf{x}$ denotes coordinates of atoms in the system, M is the atomic mass, $U(\mathbf{x})$ is the potential energy of the configuration and $-\nabla U(\mathbf{x})$ represents the forces acting on atoms. In practice, stochastic and frictional forces are integrated to model energy exchange with the environment and maintain temperature control of the system, converting the equations of motion to a Langevin process:

$$M\ddot{\mathbf{x}} = -\nabla U(\mathbf{x}) - \gamma M\dot{\mathbf{x}} + \sqrt{2M\gamma k_B T}\boldsymbol{\eta}(t),$$

where γ is the friction coefficient, k_B is the Boltzmann constant, T is the temperature, and $\boldsymbol{\eta}(t)$ is a Gaussian noise term delta-correlated in time $\langle \eta_i(t)\eta_j(t')\rangle = \delta_{ij}\delta(t-t')$. Sampling from this stochastic process generates a time evolution of system configurations. Over time, the samples converges to the Boltzmann distribution $p(\mathbf{x}) \propto \exp(-U(\mathbf{x})/k_B T)$. The trajectory of protein coordinates, $\mathbf{x}^{1:L}_{\text{prot}} = (\mathbf{x}^1_{\text{prot}}, \mathbf{x}^2_{\text{prot}}, \ldots, \mathbf{x}^L_{\text{prot}})$, is extracted and saved at prescribed simulation intervals, providing both distributional and kinetic information in the protein conformational dynamics. For simplicity, we omit the subscript 'prot' and use $\mathbf{x}$ to denote protein coordinates throughout the paper.

2.2 Protein Representations

Proteins are chain-like molecules composed of amino acid *residues*, each selected from 20 standard amino acid types. We parameterize the coordinates of heavy atoms in a protein using the SE(3)-torsional convention [20]: the backbone atoms (`N-C`α`-C`) of each residue define a local coordinate via a Gram-Schmidt process, referred to as a *rigid*. The position and orientation of each rigid relative to the global coordinate system are described by a translation-rotation transformation in SE(3) space. The backbone conformation of a protein with N residues can then be represented as $\mathbf{x} = (\mathbf{T}, \mathbf{R}) \in \mathrm{SE}(3)^N$, where $\mathbf{T} \in \mathbb{R}^{N\times 3}$ and $\mathbf{R} \in \mathrm{SO}(3)^N$ are the translation and rotation components. The coordinates of the oxygen atom of the backbone and the side chain atoms can be determined with the addition of up to 7 torsional angles $(\phi, \psi, \omega, \chi_1, \ldots, \chi_4)$ describing the bond rotation. Therefore, the complete configuration of a protein structure is parameterized in the space: $\mathbf{x} = (\mathbf{T}, \mathbf{R}, \phi, \psi, \omega, \chi_1, \ldots, \chi_4) \in (\mathrm{SE}(3) \times \mathbb{T}^7)^N$.

2.3 SE(3)-Diffusion Models for Protein Conformation Generation

Diffusion generative models are capable of learning complex data distributions. Training involves progressively corrupting data with noise and learning to reverse this process through denoising, thereby modeling the original data distribution [14, 39]. Recently, diffusion models operating in SE(3) space have been proposed to model protein backbone structures [44, 48]. Below, we provide a brief overview of diffusion model and defer the detailed SE(3) formulation to Appendix B:

Given a protein backbone conformation as $\mathbf{x}_0 = (\mathbf{T}_0, \mathbf{R}_0) \in \mathrm{SE}(3)^N$, and conditioned on the protein identity (omitted in the equations for clarity), we aim to train a neural network to jointly estimate the score functions of the reverse-time marginal distributions at varying diffusion time t, $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$. This model is trained using the *denoising score matching* (DSM) loss:

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_t, t}\left[\lambda(t)\|s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\|^2\right].$$

Here $p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)$ is the forward transition kernel defined in the SE(3) space, $\mathbf{x}_t = (\mathbf{T}_t, \mathbf{R}_t)$ is the noisy data at time t, and $\lambda(t)$ is a time-dependent weight. During inference, DPM generates clean conformations from random noise by simulating the reverse diffusion process with the learned score network $s_\theta(\mathbf{x}_t, t)$.

3 CONFROVER

3.1 Modeling MD Trajectories through Autoregression

Autoregressive generative models factorize the distribution of a sequence as a series of conditional generations over frames. We cast this idea to MD trajectories, modeling a sequence of L frames as:

$$p(\mathbf{x}^{1:L}|\mathcal{P}) = \prod_{l=1}^{L} p(\mathbf{x}^l|\mathbf{x}^{<l}, \mathcal{P}), \tag{1}$$

where $\mathbf{x}^{<l}$ is the preceding frames and $\mathcal{P}$ denotes the protein-specific conditioning input.

Despite its simplicity, this formulation naturally supports multiple learning objectives in protein conformation modeling. In its base form, it models temporal dependencies among frames by learning to generate the trajectory. When $L = 1$, it removes the frame context and reduces to a single-frame distribution $p(\mathbf{x}|\mathcal{P})$, learning to direct sample time-independent conformations. In addition, the sequential dependency in Equation (1) can be extended to any desired frame-conditioned generation tasks. By prepending conditioning frames $\mathcal{K}$ to the sequence, we train the model to learn any conditional generation $p(\mathbf{x}^{1:L}|\mathcal{K}, \mathcal{P})$, including conformation interpolation by setting $\mathcal{K} = \{\mathbf{x}^1, \mathbf{x}^L\}$. A similar idea was applied in text infilling tasks by shuffling the order of text contexts [4].

After defining the main learning objectives for trajectory simulation, single-frame (time-independent), and conformation interpolation, we describe how to effectively model autoregressive dependencies over protein conformations in Section 3.2. More critically, in Section 3.3, we explain how to adapt sequence models, traditionally designed for discrete tokens, to the continuous space of protein conformations. Lastly, we introduce a specific choice of architectures of CONFROVER in Section 3.4.

3.2 Latent Causal Modeling

We propose a modular design composed of an encoder, a latent sequence model, and a stochastic decoder. This design enables the use of modern causal transformers, such as in Llama [42], to efficiently capture sequential dependencies between frames in the latent space (Figure 2). During training, the input sequence is shifted by one frame and a mask token "[M]" is prepended; the generation process also begins with the mask token and conditioning frames (e.g., $\mathbf{x}^1$).

To model $p(\mathbf{x}^l|\mathbf{x}^{<l}, \mathcal{P})$, the context frames $\mathbf{x}^{<l}$ are first encoded into intermediate latent states $\mathbf{h}^{<l} = (\mathbf{h}^1, \ldots, \mathbf{h}^{l-1})$ using a shared encoder network with protein-specific condition $\mathcal{P}$:

$$\mathbf{h}^i = f_\eta^{\text{enc}}(\mathbf{x}^i, \mathcal{P}), \quad i = 1, 2, \ldots, l-1. \tag{2}$$

A temporal module is then used to capture the sequential dependencies between the frame latents. Causal attention is used to ensure the latent for frame $\mathbf{h}^l$ is only updated by its preceding frames:

$$\mathbf{h}^l_{\text{updated}} = f_\xi^{\text{temp}}\left(\mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^{l-1}\right). \tag{3}$$

Since both the encoder and the temporal module are chosen to be deterministic, $p(\mathbf{x}^l|\mathbf{x}^{<l})$ reduces to a conditional generation over the updated latent, realized through a probabilistic decoder:

$$p(\mathbf{x}^l|\mathbf{x}^{<l}, \mathcal{P}) = p_\theta^{\text{dec}}(\mathbf{x}^l|\mathbf{h}^l_{\text{updated}}). \tag{4}$$

Details on this latent modeling are provided in Appendix C. As a result, we train a model (η, ξ, θ) by jointly optimizing the three modules f_η^{enc}, f_ξ^{temp}, and p_θ^{dec}.

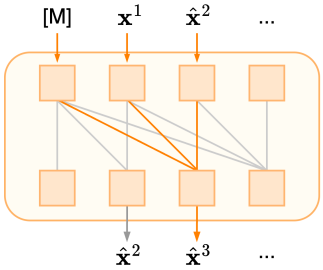

Figure 2: Causal sequence model to generate trajectory $(\hat{\mathbf{x}}^2, \ldots)$ from the mask token "[M]" and the conditioning frame $\mathbf{x}^1$. Each frame only attend to its previous frames. Attention activations for $\hat{\mathbf{x}}^3$ are highlighted in orange.

This setup easily accommodates learning single-frame distribution: by replacing all input frames with a mask token and using identity attentions, where each frame only attends to itself, we effectively disable inter-frame information flow. This trains the model to directly sample conformations. A similar strategy has been used in image-video training [16, 29].

3.3 Training Autoregressive Model with SE(3) Diffusion Loss

The main challenge in applying autoregressive modeling to conformation trajectories lies in representing the continuous distribution of protein conformations within a framework typically used

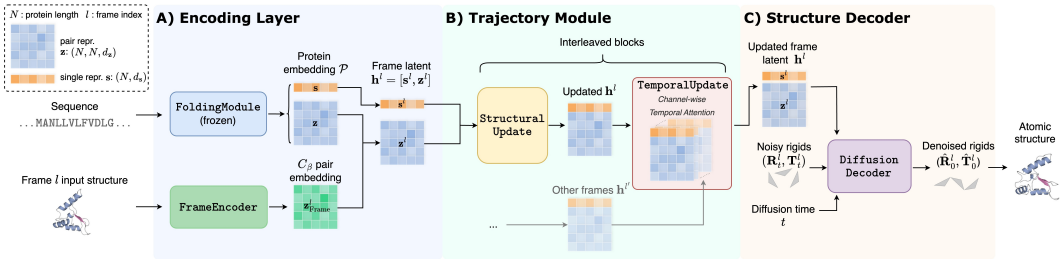

Figure 3: Architecture overview. (A) *Encoding Layer* embeds protein sequence and input structure to each frame as a frame latent representation $\mathbf{h}^l$, comprised of single and pair embeddings; (B) The *Trajectory Module* then updates frame latent $\mathbf{h}^l$ using interleaved structural and temporal update blocks; (C) A diffusion-based *Structure Decoder* learns to denoise noisy conformations conditioned on the updated frame latent $\mathbf{h}^l$; during inference, it samples conformations from the prior distribution. See Appendix D.1 for details.

for discrete token sequences. While some studies have approached this by discretizing the protein structural space into discrete tokens [13, 30, 31], such approaches inherently suffer from discretization error, which can lead to suboptimal performance in modeling protein conformations.

Instead, we propose to directly model the continuous conformational space using diffusion probabilistic models and employ the DSM loss for autoregressive model training, similar to Li et al. [26]. Specifically, we perform DSM loss training in SE(3) space.

Given a clean frame $\mathbf{x}_0^l = (\mathbf{T}_0^l, \mathbf{R}_0^l)$, its latent embedding with temporal context $\mathbf{h}^l$ (omit subscript "update" for clarity), the forward transition kernels $p_{t|0}(\mathbf{T}_t^l|\mathbf{T}_0^l)$ and $p_{t|0}(\mathbf{R}_t^l|\mathbf{R}_0^l)$ for the translation and rotation component of SE(3), and a score network to jointly estimate the translation and rotation scores $s_\theta(\mathbf{T}_t^l, \mathbf{h}^l, t)$ and $s_\theta^r(\mathbf{R}_t^l, \mathbf{h}^l, t)$, the loss is defined as

$$\begin{aligned}\mathcal{L}_{\text{DSM}}^{\text{SE(3)}} = \mathbb{E}\left[\lambda(t)\|s_\theta(\mathbf{T}_t^l, \mathbf{h}^l, t) - \nabla_{\mathbf{T}_t^l} \log p_{t|0}(\mathbf{T}_t^l|\mathbf{T}_0^l)\|^2\right] \\ + \mathbb{E}\left[\lambda^r(t)\|s_\theta^r(\mathbf{R}_t^l, \mathbf{h}^l, t) - \nabla_{\mathbf{R}_t^l} \log p_{t|0}(\mathbf{R}_t^l|\mathbf{R}_0^l)\|^2\right],\end{aligned} \tag{5}$$

where the expectation is taken over the diffusion time t and noisy structure $\mathbf{x}_t^l = (\mathbf{T}_t^l, \mathbf{R}_t^l)$ sampled from the forward process. Gradients with respect to $\mathbf{h}^l$ are then backpropagated to update the weights in the temporal module f_ξ^{temp} and encoder f_η^{enc}. During inference, we decode each frame autoregressively by performing reverse sampling as in Equation (6), replacing the scores with estimated values from $s_\theta(\mathbf{T}_t^l, \mathbf{h}^l, t)$ and $s_\theta^r(\mathbf{R}_t^l, \mathbf{h}^l, t)$.

3.4 Model Architecture

An overview of model architecture is shown in Figure 3, with detailed illustrations of each module provided in Appendix D.1.

Encoding Layer. A `FoldingModule`, parameterized by a pretrained OpenFold model [2], extracts protein-specific embeddings $\mathcal{P}$ consisting of a single representation ($\mathbf{s}$) and a pair representation ($\mathbf{z}$), shared across frames. For each frame, a `FrameEncoder`, adapted from the template module used in prior works [18, 20], encodes pairwise distance of pseudo–C_β atoms via triangular updates and merges this frame pair representations $\mathbf{z}_{\text{Frame}}^l$ with the protein pair representation $\mathbf{z}$. The resulting frame latent embedding, $\mathbf{h}^l = [\mathbf{s}^l, \mathbf{z}^l]$, is invariant to global translation and rotation of the conformations, and are passed to the *Trajectory Module*. Following causal sequence modeling, a masked frame token "[M]" is introduced by zeroing out the pseudo-C_β pairwise distances to remove structural information.

Trajectory Module. The *Trajectory Module* models structural and temporal dependencies across frames, updating each frame's embedding based on its preceding frames. It consists of interleaved `StructuralUpdate` and `TemporalUpdate` layers that operate on the frame-wise latent embeddings. `StructuralUpdate` incorporates *Pairformer* layers, a core architecture in protein structure modeling, to update single and pair embeddings through triangular operations [2]. `TemporalUpdate` employs a Llama-based causal transformer layer for channel-wise self-attention over the sequence of frame embeddings. Each channel in the single and pair embeddings is updated independently. Frame indices

are encoded using Rotary Position Embedding [41]. This interleaved design enables efficient updates while maintaining flexibility in modeling sequential dependencies.

Structure Decoder. The updated latent embeddings from *Trajectory Module* serve as conditioning signals for generating the conformation at each frame. For the SE(3) diffusion model described in Section 2.3, we adopt CONFDIFF [44] as the `DiffusionDecoder` to generate 3D conformations. CONFDIFF composes of layers of Invariant Point Attention and Transformer (on single embeddings) to collectively update the residue SE(3) *rigids*, as well as single and pair embeddings. Trained with denoising score matching in Equation (5), the `DiffusionDecoder` learns to iteratively denoise noisy frame structures drawn from a prior SE(3) distribution, conditioned on the frame latent embeddings, to generate accurate backbone conformations of the frame. To reconstruct full-atom geometry, we additionally predict the 7 torsional angles $(\phi, \psi, \omega, \chi_1, \ldots, \chi_4)$ using a light-weight `AngleResNet`, for the coordinates of backbone oxygen atom and side-chain atoms.

4 Experiments

Dataset. We evaluate model performance on ATLAS [43], a large-scale protein MD dataset covering $\sim$1300 proteins with diverse sizes and structures. For each protein, it contains triplicated 100 ns simulation trajectories. All models are trained on training trajectories and evaluated on test trajectories split by protein identity [18, 19, 44]. This presents a challenging task for assessing the generalization to unseen protein structures and dynamics.

Model training. We use OPENFOLD (with frozen pretrained weights) as CONFROVER's `FoldingModule` and initialized the weights of the `DiffusionDecoder` from CONFDIFF, while training the remaining parts of the network from scratch. During training, trajectories of length $L = 8$ with varying timesteps (strides), corresponding to $1 \sim 1024$ MD snapshots saved at 10 ps intervals, are sampled to enable learning across multiple timescales. For the base CONFROVER, we adopt a *hybrid training* strategy with 1:1 ratio between trajectory and single-frame training objectives. To further enable conformation interpolation, we continue training the base model with a 1:1:1 ratio of trajectory, single-frame and interpolation objectives, denotes the model as CONFROVER-INTERP. See Appendix D.2 for training details.

Baselines. We compare CONFROVER with state-of-the-art deep learning models for each task: For *trajectory simulation*, we compare against MDGEN [19], a flow-based non-autoregressive trajectory model trained on ATLAS; For *time-independent generation*, we evaluate against ALPHAFLOW [18] and CONFDIFF [44], flow- and diffusion-based conformation generation models finetuned on ATLAS; For *conformation interpolation*, no existing baseline is available for large proteins, therefore, we focus on analyzing results of our model. See Appendix A for more details on the availability of baseline models.

4.1 Trajectory Simulation

Since trajectories from both MD and models are stochastic samples, directly comparing them using frame-wise error, such as root-mean-square-deviation (RMSD) between atomic coordinates, is not appropriate. Therefore, we evaluate the model's ability to recover trajectory dynamics from two perspectives: (1) how well it captures the magnitude of conformation changes across varying start conformations and timescales; (2) how well it recovers the conformational states and principal dynamic modes observed in long-time MD simulations.

Evaluating conformation change accuracy on *multi-start* benchmark. We curated a test benchmark consisting of short trajectories with $L = 9$ frames, extracted from 82 ATLAS test proteins. For each protein, we choose from varying starting frames (snapshot index $1000 \sim 7000$) and strides ($128 \sim 1024$ snapshots), resulting in a total of $\sim 2,700$ generation conditions. For each trajectory, we measure three aspects of conformational changes: *Trajectory*, the total changes over the entire sequence $\sum_{l=1}^{L-1} d(\mathbf{x}^l, \mathbf{x}^{l+1})$; *Frame*, the changes of each frame relative to the starting frame $d(\mathbf{x}^l, \mathbf{x}^1)$; Δ *Frame*, the changes between consecutive frames $d(\mathbf{x}^l, \mathbf{x}^{l+1})$. Here, $d(\cdot, \cdot)$ measures the distance between two conformations. We report both the L^2-distance in projected 2D PCA space and the RMSD (in Å) of alpha-carbon (Cα) atoms. This benchmark captures diverse dynamics at both the trajectory and frame levels and enables comprehensive evaluation across varying conditions and timescales (see Appendix E.1 for details).

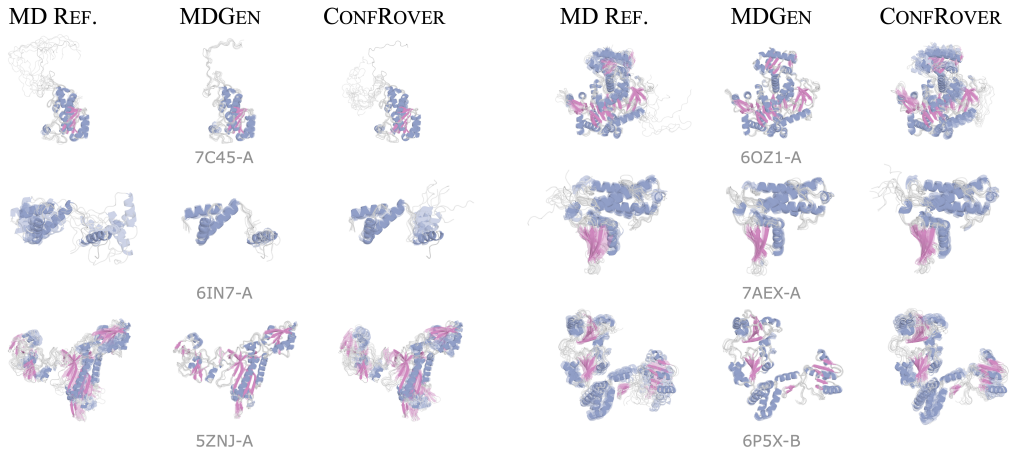

Figure 4: Visualization of six proteins from *multi-start*. Trajectory conformations are colored by their secondary structures and superposed to show the dynamic ensemble. MDGEN primarily exhibits local movements, whereas CONFROVER captures conformations changes similar to MD simulations.

ConfRover shows superior performance in recovering the magnitude of conformational changes. We report the Pearson correlation of measured conformation changes between model-generated and reference trajectories in Table 1, with additional results in Appendix E.1. Compared with MDGEN, CONFROVER shows a significant improvement in correlation scores, mean absolute error and structural quality (Table 9), indicating its stronger ability to recover the magnitude of conformation changes across different starting conditions in the conformational space. The greater difference observed in PCA highlights that CONFROVER more accurately captures conformational changes along the feature dimensions most relevant to the structural variance observed in MD. Figure 4 visualizes ensembles of conformations in generated trajectories, with additional examples in Figure 11. CONFROVER exhibit more notable conformation changes than MDGEN, and reflects the major movements in the structured and loop domains observed in the MD reference.

Table 1: Pearson correlations between conformation changes in model-generated and reference trajectories under the *multi-start* setting. The mean and standard deviations are calculated from five independent runs. Models with higher correlations are highlighted in **bold**.

	Cα coordinates		
	Traj.	Frame	ΔFrame
MDGEN	0.56±0.03	0.47±0.03	0.41±0.02
CONFROVER	**0.75**±0.01	**0.63**±0.01	**0.53**±0.01
	PCA 2D		
	Traj.	Frame	ΔFrame
MDGEN	0.18±0.01	0.15±0.01	0.10±0.01
CONFROVER	**0.73**±0.01	**0.50**±0.01	**0.43**±0.00

MDGEN is a non-autoregressive model trained on trajectories of length $L = 250$. Adjusting its inference setup results in degraded conformation. To ensure fair comparison, we use the original inference setting ($S = 40, L = 250$) and downsampled the trajectories for evaluation. To confirm that this post-processing step does not introduce artifacts, we also trained MDGEN models under the evaluation setups. The results are consistent with the downsampled version (see Appendix E.5).

Assessing long trajectory generation on 100 ns simulation. We further evaluate model's ability to recover conformational states and principal dynamics of proteins. For each of protein, we simulate a trajectory of $L = 80$ frames at stride $S = 120$, approximating the 100 ns MD simulation in ATLAS. To assess state recovery, model-generated conformations are projected into a reduced PCA space and compared with the reference trajectory. Specifically, we discretize each principal component into 10 evenly sized "states" and measure the distribution similarity using Jensen-Shannon Distance (JSD). We also compute precision, recall, and F1-score on whether sample conformations fall within these known states [32, 44, 49]. To evaluate dynamic mode recovery, we perform time-

Table 2: Recovery of conformational states in the 100 ns simulation experiment. The mean and standard deviation are computed over five independent runs, and the better results are highlighted in **bold**. MD 100NS serves as the oracle and is excluded from the comparison.

	JSD (↓)	Recall (↑)	F1 (↑)
MD 100NS	0.31	0.67	0.79
MDGEN	0.56±0.01	0.29±0.01	0.42±0.01
CONFROVER	**0.51**±0.01	**0.42**±0.00	**0.58**±0.00

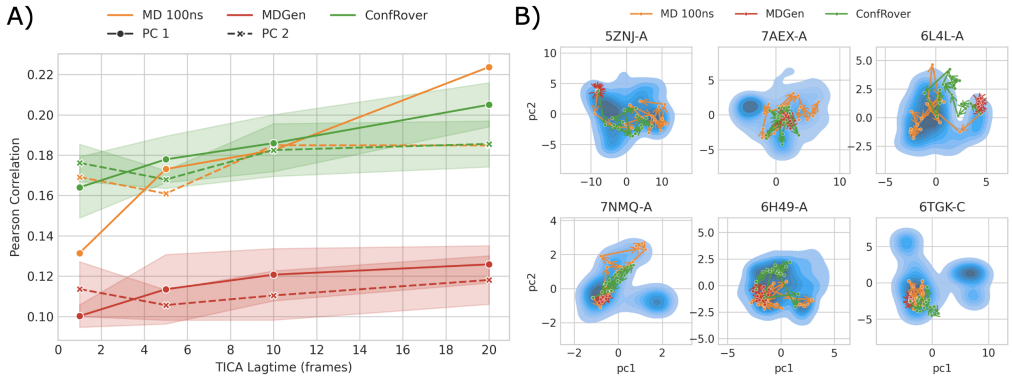

Figure 5: Results from 100 ns simulation. (A) Correlations of principal dynamic modes between sample and reference trajectories, evaluated at varying lag time. The mean and standard deviation are shown as line and shadowed area, computed from five individual runs for MDGEN and CONFROVER. (B) Examples trajectories illustrating the states explored by different methods (downsampled by 5 frames for visualization). The blue background indicates the density of the ground-truth conformation distribution from MD reference.

lagged independent component analysis (tICA) at varying lag times on both reference and sample trajectories. We then compute Pearson correlations between the per-residue contribution to the leading components, based on the tICA coefficients. See Appendix E.2 for details. We include one of the triplicate MD trajectories—excluded from ground-truth evaluation—as an "oracle" reference, denoted as MD 100NS, representing the performance expected if the model were as accurate as an MD simulation run.

ConfRover recovers more conformational states than MDGen and accurately captures the principal dynamics. As shown in Table 2, CONFROVER outperforms MDGEN in state recovery, achieving lower JSD, higher recall and F1 scores, showing its improved ability to capture diverse conformations. Additionally, CONFROVER shows clear advantage in capturing the principal dynamic modes across varying lag times, performing even comparably to the MD oracle (Figure 5A). This results suggest CONFROVER can learn and generalize dynamics to unseen proteins and still capture the most important dynamic modes. We visualize simulated trajectories in the PCA space in Figure 5B and Figure 12. These examples again confirm that CONFROVER is more capable of sampling over the conformational space of the proteins and covering diverse conformations. Yet, we also observe some cases where MD 100NS overcame the energy barrier and achieved more remote states while CONFROVER did not (e.g., 7NMQ-A in Figure 5B).

Summary. These experiments demonstrate that CONFROVER outperforms the current state-of-the-art model in trajectory simulation, effectively learning protein dynamics from MD data and generalizing well to unseen proteins. While a gap remains compared to the oracle MD 100NS, particularly in state recovery, the improvement narrows the gap between deep generative models and established simulation methods.

4.2 Time-independent Conformation Sampling

We evaluate the time-independent sampling performance of CONFROVER, following the benchmark setup in Ye et al. [47]. For each protein, 250 independent conformations are sampled and compared the model generated ensembles with MD reference ensembles. We summarize the mean and standard deviations from five independent runs with key metrics are summarized in Table 3 with full results in Appendix E.3.

ConfRover matches the performance of state-of-the-art ensemble generation models. Compared with ALPHAFLOW and CONFDIFF, CONFROVER demonstrate overall comparable performance and outperforms at least one of the SOTA models in five evaluation criteria. This demonstrate that CONFROVER, despite being a general-purpose model capable of trajectory generation, also performs strongly in sampling independent conformations that approximate the equilibrium distribution from MD simulation. In contrast, MDGEN, which is trained solely for trajectory generation, shows suboptimal results with sequentially sampled conformations.

Effect of hybrid training. Without explicit single-frame training, the model primarily learns time-dependent generation, with only the first frame of each trajectory learning to generate conformation

Table 3: Results from the time-independent generation experiment. ALPHAFLOW and CONFDIFF are state-of-the-art models for direct conformation ensemble generation. CONFROVER is the base model trained for both trajectory and time-independent generation, where as CONFROVER-TRAJ and MDGEN are trained exclusively for trajectory generation. The mean and standard deviation are computed from five independent runs. The best scores are highlighted in **bold**, and the second-best scores are underlined.

	Pairwise RMSD r (↑)	Per target RMSF r (↑)	RMWD (↓)	MD PCA $\mathcal{W}_2$ (↓)	Joint PCA $\mathcal{W}_2$ (↓)	Weak contacts J (↑)	Transient contacts J (↑)	Exposed residue J (↑)
ALPHAFLOW	**0.56±0.06**	**0.85±0.01**	**2.62±0.03**	1.52±0.05	2.26±0.03	0.62±0.00	**0.41±0.00**	**0.69±0.01**
CONFDIFF	0.54±0.00	**0.85±0.00**	2.70±0.01	1.44±0.00	**2.22±0.04**	**0.64±0.00**	0.40±0.00	0.67±0.00
MDGEN	0.47±0.04	0.72±0.02	2.78±0.04	1.86±0.03	2.44±0.04	0.51±0.01	0.28±0.01	0.57±0.01
CONFROVER-TRAJ	0.48±0.00	0.84±0.01	2.85±0.02	**1.43±0.01**	2.30±0.01	0.53±0.01	0.36±0.00	0.58±0.01
CONFROVER	0.51±0.01	**0.85±0.00**	2.66±0.02	1.47±0.03	2.23±0.04	0.62±0.01	0.37±0.01	0.66±0.01

unconditionally (i.e., from a masked token input). To test the importance of hybrid training, we ablated the single-frame objective and trained a variant, CONFROVER-TRAJ, solely on trajectory generation. As shown in Table 3, while this variant still outperforms time-dependent results from MDGEN, it shows decreased performance across several metrics compared to CONFROVER. This highlights the importance of hybrid training in balancing the learning objectives and enhancing the model's for generating independent conformations.

4.3 Conformation Interpolation

To enable CONFROVER for conformation interpolation, we continue training CONFROVER with a hybrid objective combining trajectory, single-frame and interpolation, referred as CONFROVER-INTERP. We select 38 short trajectories from *multi-start* for evaluation where the reference MD trajectories for these cases exhibit significant conformation changes and clear state transitions, see Appendix E.4 for details. To condition on both start and end frames, we prepend the end frame to the start frame and autoregressively generate the remaining (intermediate) frames. To evaluate whether the model generate smooth transitions towards the target end state, we measure Cα-RMSD and L^2 distance in the PCA space between each intermediate frame and the start/end frames.

Training on the interpolation objective enables smooth interpolation between conformations. As shown in Figure 6A, the distance to the start frame increases while the distance to the end frame decreases with frame index, indicating smooth and directed transitions. Without explicit interpolation training, the original CONFROVER (dashed lines in Figure 6A) generates trajectory that do not progress towards the end state. Figure 6B visualizes intermediate structures and transition pathways in PCA space, showing that intermediate conformations from CONFROVER-INTERP closely resemble those in the MD reference. In contrast, as shown in Figure 14, the original CONFROVER can miss key transitions and fails to reach the end state. Additional results and visualizations are provided in Appendix E.4. These results highlight the effectiveness of our interpolation training strategy: by adjusting the dependency order in the sequence model, CONFROVER-INTERP learns to generate smooth transitions between two conformations.

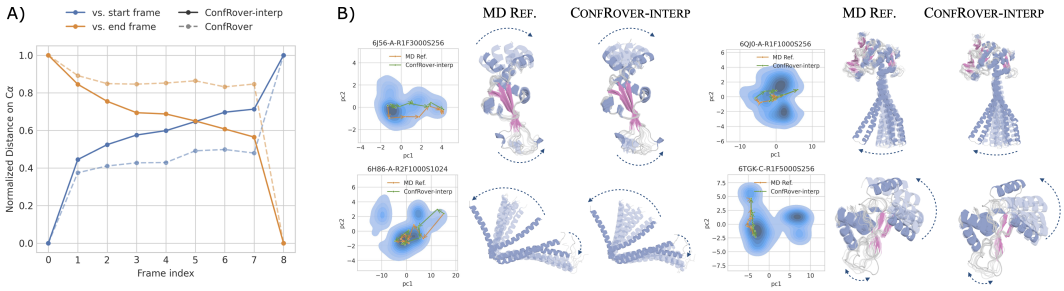

Figure 6: Results from *conformation interpolation.* (A) Cα-RMSD distance of intermediate frames to the start and end frames, normalized by the distance between start and end frames. Reported values are averaged over 38 cases selected from the *multi-start* benchmark. (B) Example interpolations results. CONFROVER-INTERP generates smooth pathways between the start and end frames, capturing the dynamics observed with the MD reference. Start and end frames are shown as solid structures; intermediate conformations are shown in fading colors. Main motions are indicated by blue arrows.

Table 4: Quality of model generated conformations. Conformations sampled from 38 trajectories are evaluated. Geometric metrics are reported as mean and standard deviations across 38 cases, and energy values are reported as mean with 95% percentiles. The best scores are highlighted in **bold**.

	Ramachandran outliers % (↓)	Rotamer outliers % (↓)	Clash score (↓)	RMS bonds (↓)	RMS angles (↓)	MolProbity score (↓)	MadraX energy (↓)
MD REFERENCE	0.38±0.49	1.02±0.89	0.04±0.16	0.01±0.00	1.88±0.05	0.72±0.18	-519.3 (-1793.0,-53.4)
MDGEN	0.93±0.86	2.86±1.59	16.14±20.05	0.02±0.02	2.13±0.30	2.24±0.40	-314.7 (-1483.8,263.6)
CONFROVER	**0.58±0.63**	1.98±1.48	7.81±6.52	**0.01±0.01**	**1.88±0.25**	1.72±0.38	**-522.2 (-1858.9, -53.4)**
CONFROVER-INTERP	0.71±0.94	**1.86±1.46**	**7.25±8.74**	0.02±0.01	1.91±0.32	**1.61±0.51**	-469.7 (-1712.3, -42.8)

4.4 Conformation Quality

To ensure that CONFROVER generates physically plausible conformations, we further evaluate the quality using geometric assessments from MolProbity package [46] and energy profiles using a coarse-grained force field MadraX [36]. We compared conformations across 38 trajectories shared between the forward simulation and interpolation experiments, including results from MDGEN, CONFROVER, and an oracle MD REFERENCE. All structures are relaxed using the refinement pipeline in OpenFold [2] to enable energy comparison.

As shown in Table 4, conformations generated by MD simulation exhibit the hightest overall quality, as expected. As a generative model, CONFROVER also produces high-quality conformations with fewer backbone (Ramachandran) and side-chain (rotamer) outliers, more accurate covalent lengths and angles, and achieving energy levels comparable to those of MD REFERENCE; outperforming MDGEN across all metrics. Furthermore, we compare conformations generated from forward simulation and interpolation tasks, where the latter includes additional constraints on terminal conformations. CONFROVER-INTERP shows similar geometric and energetic metrics, indicating that intermediate conformations maintains physical plausible when CONFROVER tries to interpolate between two conformational states.

5 Conclusions and Limitations

We introduce CONFROVER, a general framework for learning protein conformational dynamics from MD trajectory data. Through autoregressive factorization, CONFROVER supports three tasks in a unified manner: trajectory simulation, time-independent sampling, and conformation interpolation. This formulation reflects the temporal nature of MD while naturally encompassing conditional and unconditional frame-level generation. Extensive experiments and analyses highlight several empirical advantages: (1) CONFROVER outperforms the current state-of-the-art in trajectory simulation, accurately capturing dynamic magnitude, state recovery, and principal motions; (2) Despite being a multi-purpose model, it achieves competitive performance in time-independent sampling compared to specialized methods; (3) With simple sequence reordering, CONFROVER effectively learns to interpolate between conformations.

Nevertheless, several limitations still remain: (1) Trajectory simulation and interpolation are emerging tasks with few available baseline models. We hope that this work, together with future developments in the field, will contribute to establishing more comprehensive benchmarks; (2) The dataset and evaluation metrics used in this study are limited and preliminary. The ATLAS dataset contains 100 ns simulations of single-chain proteins, which may not capture large conformational changes or the dynamics of protein complexes. Although we curated interpolation cases from ATLAS for demonstration purposes, future benchmarks reflecting realistic functional state transitions would be more meaningful. (3) Although CONFROVER narrows the gap with classical MD, it still falls short in fully capturing the conformational space with high structural fidelity. Future gains may come from scaling training data, using more efficient architectures, and leveraging additional information from MD, such as energy information. (4) Finally, while the triangular updates in the structural modules ensure high accuracy, their computational cost limits scalability to larger proteins and longer trajectories. Despite these challenges, CONFROVER demonstrates the promise of autoregressive models in molecular simulation, offering a unified, efficient, and extensible approach to modeling protein dynamics.

6 Acknowledgments

We would like to thank Dr. Hang Li for his invaluable support of this project. We also thank Zaixiang Zheng for insightful discussions, and Wesley Hsieh, Yi Zhou, Nima Shoghi, Yuxuan Liu, Xiaolu Shen, Jing Yuan, Yilai Li, Fei Ye, and Wei Qu for their valuable feedback.

References

[1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024. doi: 10.1038/s41586-024-07487-w.

[2] Gustaf Ahdritz, Nazim Bouatta, Christina Floristean, Sachin Kadyan, Qinghui Xia, William Gerecke, Timothy J O'Donnell, Daniel Berenberg, Ian Fisk, Niccolò Zanichelli, Bo Zhang, Arkadiusz Nowaczynski, Bei Wang, Marta M Stepniewska-Dziubinska, Shang Zhang, Adegoke Ojewole, Murat Efe Guney, Stella Biderman, Andrew M Watkins, Stephen Ra, Pablo Ribalta Lorenzo, Lucas Nivon, Brian Weitzner, Yih-En Andrew Ban, Peter K Sorger, Emad Mostaque, Zhao Zhang, Richard Bonneau, and Mohammed AlQuraishi. OpenFold: Retraining AlphaFold2 yields new insights into its learning mechanisms and capacity for generalization. *bioRxiv*, 2022.

[3] Marloes Arts, Victor Garcia Satorras, Chin-Wei Huang, Daniel Zügner, Marco Federici, Cecilia Clementi, Frank Noé, Robert Pinsler, and Rianne van den Berg. Two for one: Diffusion models and force fields for coarse-grained molecular dynamics. *Journal of Chemical Theory and Computation*, 19(18):6151–6159, 2023. doi: 10.1021/acs.jctc.3c00702.

[4] Mo Bavarian, Heewoo Jun, Nikolas A. Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *ArXiv*, abs/2207.14255, 2022.

[5] Herman JC Berendsen and Steven Hayward. Collective protein dynamics in relation to function. *Current opinion in structural biology*, 10(2):165–169, 2000.

[6] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[7] Kaihui Cheng, Ce Liu, Qingkun Su, Jun Wang, Liwei Zhang, Yining Tang, Yao Yao, Siyu Zhu, and Yuan Qi. 4d diffusion for dynamic protein structure prediction with reference guided motion alignment. *arXiv preprint arXiv:2408.12419*, 2024.

[8] Matthew Carter Childers and Valerie Daggett. Insights from molecular dynamics simulations for computational protein design. *Mol. Syst. Des. Eng.*, 2:9–33, 2017.

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

[10] Diego del Alamo, Davide Sala, Hassane S Mchaourab, and Jens Meiler. Sampling alternative conformational states of transporters and receptors with alphafold2. *eLife*, 11:e75751, mar 2022. doi: 10.7554/eLife.75751.

[11] Yuanqi Du, Michael Plainer, Rob Brekelmans, Chenru Duan, Frank Noe, Carla P Gomes, Alan Aspuru-Guzik, and Kirill Neklyudov. Doob's lagrangian: A sample-efficient variational approach to transition path sampling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

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

[13] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf A. Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, 2025.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

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

[16] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022. URL `http://arxiv.org/abs/2204.03458`.

[17] Bowen Jing, Ezra Erives, Peter Pao-Huang, Gabriele Corso, Bonnie Berger, and Tommi Jaakkola. Eigenfold: Generative protein structure prediction with diffusion models. *arXiv preprint arXiv:2304.02198*, 2023.

[18] Bowen Jing, Bonnie Berger, and Tommi Jaakkola. Alphafold meets flow matching for generating protein ensembles. In *Proceedings of the 41st International Conference on Machine Learning*, pages 22277–22303, 2024.

[19] Bowen Jing, Hannes Stärk, Tommi Jaakkola, and Bonnie Berger. Generative modeling of molecular dynamics trajectories. *arXiv preprint arXiv:2409.17808*, 2024.

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

[21] Martin Karplus and John Kuriyan. Molecular dynamics and protein function. *Proceedings of the National Academy of Sciences*, 102(19):6679–6685, 2005.

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

[23] Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International conference on machine learning*, pages 5361–5370. PMLR, 2020.

[24] Dávid Péter Kovács, J. Harry Moore, Nicholas J. Browning, Ilyes Batatia, Joshua T. Horton, Yixuan Pu, Venkat Kapil, William C. Witt, Ioan-Bogdan Magdău, Daniel J. Cole, and Gábor Csányi. Mace-off: Short-range transferable machine learning force fields for organic molecules. *Journal of the American Chemical Society*, 147(21):17598–17611, 2025. doi: 10.1021/jacs.4c07099.

[25] Sarah Lewis, Tim Hempel, José Jiménez Luna, Michael Gastegger, Yu Xie, Andrew YK Foong, Victor García Satorras, Osama Abdin, Bastiaan S Veeling, Iryna Zaporozhets, et al. Scalable emulation of protein equilibrium ensembles with generative deep learning. *bioRxiv*, pages 2024–12, 2024.

[26] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*, 2024.

[27] Zongzhao Li, Jiacheng Cen, Liming Wu, Hao Sun, Hangyu Mao, zhangfuzheng, Di Zhang, and Wenbing Huang. Geometric spatiotemporal transformer to simulate long-term physical dynamics, 2025. URL https://openreview.net/forum?id=LOBhVTtVnc.

[28] Ce Liu, Jun Wang, Zhiqiang Cai, Yingxu Wang, Huizhen Kuang, Kaihui Cheng, Liwei Zhang, Qingkun Su, Yining Tang, Fenglei Cao, Limei Han, Siyu Zhu, and Yuan Qi. Dynamic pdb: A new dataset and a se(3) model extension by integrating dynamic behaviors and physical properties in protein structures, 2024.

[29] Haozhe Liu, Shikun Liu, Zijian Zhou, Mengmeng Xu, Yanping Xie, Xiao Han, Juan C Pérez, Ding Liu, Kumara Kahatapitiya, Menglin Jia, et al. Mardini: Masked autoregressive diffusion for video generation at scale. *arXiv preprint arXiv:2410.20280*, 2024.

[30] Yufeng Liu, Linghui Chen, and Haiyan Liu. Diffusion in a quantized vector space generates non-idealized protein structures and predicts conformational distributions. *bioRxiv*, 2023. doi: 10.1101/2023.11.18.567666.

[31] Jiarui Lu, Xiaoyin Chen, Stephen Zhewen Lu, Chence Shi, Hongyu Guo, Yoshua Bengio, and Jian Tang. Structure language models for protein conformation generation. *arXiv preprint arXiv:2410.18403*, 2024.

[32] Jiarui Lu, Bozitao Zhong, Zuobai Zhang, and Jian Tang. Str2str: A score-based framework for zero-shot protein conformation sampling. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[34] JA McCammon. Protein dynamics. *Reports on Progress in Physics*, 47(1):1, 1984.

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

[36] Gabriele Orlando, Luis Serrano, Joost Schymkowitz, and Frederic Rousseau. Integrating physics in deep learning algorithms: a force field as a pytorch module. *Bioinformatics*, 40(4):btae160, 2024.

[37] Benjamin Rhodes, Sander Vandenhaute, Vaidotas Šimkus, James Gin, Jonathan Godwin, Tim Duignan, and Mark Neumann. Orb-v3: atomistic simulation at scale, 2025. URL https://arxiv.org/abs/2504.06231.

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

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

[40] Richard A. Stein and Hassane S. Mchaourab. Speach_af: Sampling protein ensembles and conformational heterogeneity with alphafold2. *PLOS Computational Biology*, 18(8):1–16, 08 2022. doi: 10.1371/journal.pcbi.1010483.

[41] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

[42] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

[44] Yan Wang, Lihao Wang, Yuning Shen, Yiqun Wang, Huizhuo Yuan, Yue Wu, and Quanquan Gu. Protein conformation generation via force-guided se (3) diffusion models. In *Forty-first International Conference on Machine Learning*, 2024.

[45] Hannah K. Wayment-Steele, Adedolapo Ojoawo, Renee Otten, Julia M. Apitz, Warintra Pitsawong, Marc Hömberger, Sergey Ovchinnikov, Lucy Colwell, and Dorothee Kern. Predicting multiple conformations via sequence clustering and alphafold2. *Nature*, 625(7996):832–839, November 2023.

[46] Christopher J Williams, Jeffrey J Headd, Nigel W Moriarty, Michael G Prisant, Lizbeth L Videau, Lindsay N Deis, Vishal Verma, Daniel A Keedy, Bradley J Hintze, Vincent B Chen, et al. Molprobity: more and better reference data for improved all-atom structure validation. *Protein Science*, 27(1):293–315, 2018.

[47] Fei Ye, Zaixiang Zheng, Dongyu Xue, Yuning Shen, Lihao Wang, Yiming Ma, Yan Wang, Xinyou Wang, Xiangxin Zhou, and Quanquan Gu. Proteinbench: A holistic evaluation of protein foundation models. *arXiv preprint arXiv:2409.06744*, 2024.

[48] Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. SE(3) diffusion model with application to protein backbone generation. In *Proceedings of the 40th International Conference on Machine Learning*, pages 40001–40039, 2023.

[49] Shuxin Zheng, Jiyan He, Chang Liu, Yu Shi, Ziheng Lu, Weitao Feng, Fusong Ju, Jiaxi Wang, Jianwei Zhu, Yaosen Min, et al. Predicting equilibrium distributions for molecular systems with deep learning. *Nature Machine Intelligence*, 6(5):558–567, 2024.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims are centered around a new framework (see Section 3) and its claimed performance supported by empirical results across three key tasks (see Section 4).

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See Section 5.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

Justification: N/A

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Detailed information about model framework, architectures, and experimental setups are included in the main text and appendices.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The ATLAS data is open access (https://www.dsimb.inserm.fr/ATLAS/index.html). Code and model checkpoints for this work will be released on GitHub.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: See Section 4, Appendix D and E.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: For main experiments, we evaluated the results through repeated independent runs and report the statistics (e.g., mean and standard deviations). Calculation details are included in the caption of each Table.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: See Appendix D.3 and Appendix D.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: N/A

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: See Appendix E.7

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [No]

    Justification: We consider our work to pose a low risk of misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have properly cited the dataset, model, and related references.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: N/A

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

    Justification: N/A

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: N/A

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: N/A

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Additional Background

A.1 Related Work

Deep Generative Models for MD Trajectories. Recent works have explored generating protein trajectories as a surrogate for MD simulations. Models such as TIMEWARP [22], ITO [38], and EQUIJUMP [9] learn stochastic transport functions to sample future conformations at a lagged time (longer than MD intervals), reducing the computational costs of long-timescale simulations. However, these methods assume Markovian dynamics by relying solely on the current state for prediction, which may not be suitable for non-Markovian dynamics common in protein MD data. To capture higher-order dependencies between the frames, ALPHAFOLDING [7] incorporates history frames via "motion nodes", but it requires a fixed context window. MDGEN [19] takes a different approach by directly modeling the joint distributions of frames in a trajectory and learning frame dependencies through "masked frame modeling", similar to masked language modeling. However, its key-frame parameterization requires separate models for different tasks, and its non-autoregressive paradigm limits flexible generation (e.g., not compatible for generating trajectories with variable lengths). GST [27] applies autoregression for future frame prediction, enabling variable-length conditioning context and prediction horizons. While the autoregressive approach is conceptually similar to CONFROVER, their work differs in several key aspects: it performs deterministic prediction rather than generative trajectory sampling; it employs a graph-based architecture with fixed structural priors from an adjacency graph, instead of full attention across all residues; it is trained and evaluated on a single protein instead of diverse proteins from ATLAS under a transferable setting.

Beyond forward trajectory simulation, generative models have also been applied to conformation interpolation, that is, generating the intermediate trajectories between two conditioned states. The non-autoregressive framework of MDGEN [19] can be extended to sample transition pathways between such states; however, its key-frame parameterization requires training a separate model for this task, and it has not been tested on large proteins like those in ATLAS. Du et al. [11] proposed a simulation-free objective for transition-pathway sampling based on Doob's h-transform, but their approach has only been validated on numerical models and the small protein Chignolin. Its generalizability to larger, more diverse proteins remains unassessed.

Notably, the above models focus on learning temporal dependencies between frames and do not support direct, time-independent conformation sampling from the learned distribution. In contrast, CONFROVER is a general framework that learns both the trajectory generation tasks as well as direct sampling of independent protein conformations.

Deep Learning Models for Conformation Ensemble Generation. Another line of work focuses on direct sampling of conformations in a time-independent manner. Early efforts include perturbing the input to folding models (e.g., AlphaFold) [10, 45, 40] or perturbing the conformation using a structural diffusion model [32]. However, these models are trained solely on static PDB structures and do not explicitly model the conformational distribution. Recent works have shifted to deep generative paradigms that directly learn protein-specific conformational distributions [17, 49, 18, 44, 31, 25]. Several models in this category, including ALPHAFLOW, CONFDIFF, and BIOEMULATOR, fine-tune pretrained structure models on large-scale MD datasets, enhancing their ability to capture conformational distributions. A related approach trains normalizing flow models to approximate the Boltzmann distribution [35, 23], but their invertibility constraints limit scalability and transferability beyond small molecules and peptides. While these methods can generate time-independent conformations, they overlook temporal relationships and do not capture the kinetic aspects of protein dynamics.

Deep Learning Enhanced Molecular Dynamics. Another direction integrates deep learning with molecular conformation modeling through machine learning force field (MLFF) models [24, 37]. These models aim to incorporate higher-level accuracy (e.g., from *ab initio* calculations) into classical molecular dynamics simulations, improving fidelity while maintaining scalability. However, they still rely on sequential MD sampling with small integration steps and can be more computationally expensive than conventional MD. Two-for-One [3] does not directly learn an MLFF but instead trains a diffusion model for protein conformations and uses the resulting score function as an approximate coarse-grained force field for MD simulation. Although it still depends on sequential MD sampling, this work provides an interesting perspective that connects conformation distribution modeling and trajectory generation. In contrast, different from these approaches, CONFROVER explicitly models

and generates both individual conformation and trajectories, without relying on force field-based MD simulations.

Image-Video Generation. The challenge of modeling protein dynamics conceptually parallels tasks in image and video generation, requiring both data distribution learning and temporal modeling. Recent advances in video generation offer valuable insights in addressing these challenges. Given limited video data, extending image generative models to video has proven effective. Several works [16, 6, 15] achieve this by incorporating temporal attention layers, enabling frame-to-frame communication. Disabling temporal attention reverts them to image models, allowing flexible training across both modalities. These approaches efficiently model time correlations without explicitly tracking offsets between frames. Meanwhile, the extension of autoregressive language models to image and video domains has shown strong potential for sequential generation in different data modalities. Li et al. [26] integrates language models' sequential modeling with diffusion models' ability to model continuous distributions, showing that discrete tokens are not essential for autoregressive models. MARDINI [29] extends the concept to video generation with efficient llama-style temporal planning and high-resolution video generation via a diffusion decoder. By applying masked "frame" modeling, MARDINI allows the model to learn flexible temporal relationships and enables diverse tasks such as frame interpolation. CONFROVER differs from these works from video models in several aspects: it employs SE(3) diffusion for 3D structure generation; by using an autoregressive paradigm, it explicitly decouples the diffusion generation from temporal modeling, unlike the spatiotemporal denoising process in MARDINI; in addition, the causal autoregressive framework enables more flexible trajectory generation with variable lengths.

A.2 Baseline Limitation

Modeling MD trajectories using deep generative models is still an emerging research area, few models currently support learning protein dynamics in transferable settings. Existing approaches based on forward transport operators have been primarily trained and evaluated on small peptides (e.g., Timewarp [22]) or small fast-folding proteins (e.g., EquiJump [9]). Although AlphaFolding [7] was also trained using ATLAS, the model weights are not publicly available at the time of this work. (We attempted an internal reproduction and included its result in Appendix E.6.) Due to these limitations, we use MDGEN as the only available model for the main experiment. For conformation interpolation, neither sampling-based method [11] nor video-like method MDGEN [19] have been trained and evaluated on large proteins. Therefore, we focus on demonstrating the interpolation results of CONFROVER-INTERP.

B Diffusion Models on SE(3) Space

Diffusion Probabilistic Models (DPM) model complex distributions through iterative denoising. In the context of protein conformations, DPMs defined over $\mathrm{SE}(3)$ translation-rotation space have been applied for protein backbone structural generation [48, 44]. Following Section 2.2, $\mathbf{x}_0 = (\mathbf{T}_0, \mathbf{R}_0) \in \mathrm{SE}(3)^N$ denotes the translations and rotations of backbone *rigids* in data. The diffusion processes defined in the translation and rotation subspace add noise to corrupt the data:

$$\begin{aligned}
\mathrm{d}\mathbf{T}_t &= -\frac{1}{2}\beta_t \mathrm{P}\mathbf{T}_t \,\mathrm{d}t + \sqrt{\beta_t}\mathrm{P}\mathrm{d}\mathbf{w}_t, \\
\mathrm{d}\mathbf{R}_t &= \sqrt{\frac{\mathrm{d}}{\mathrm{d}t}\sigma_t^2}\,\mathrm{d}\mathbf{w}_t^{\mathrm{SO}(3)},
\end{aligned}$$

where $t \in [0, 1]$ is the diffusion time, β_t and σ_t are predefined time-dependent noise schedules and P is a projection operator removing the center of mass. $\mathbf{w}_t$ and $\mathbf{w}_t^{\mathrm{SO}(3)}$ are the standard Wiener processes in $\mathcal{N}(0, I_3)^{\otimes N}$ and $\mathcal{U}(\mathrm{SO}(3))^{\otimes N}$ respectively.

The transition kernel of $\mathbf{T}$ satisfies $p_{t|0}(\mathbf{T}_t|\mathbf{T}_0) = \mathcal{N}(\mathbf{T}_t; \sqrt{\alpha_t}\mathbf{T}_0, (1-\alpha_t)\mathbf{I})$, where $\alpha_t = e^{-\int_0^t \beta_s ds}$. The transition kernel of $\mathbf{R}$ satisfies $p_{t|0}(\mathbf{R}_t|\mathbf{R}_0) = \mathcal{IGSO}_3(\mathbf{R}_t; \mathbf{R}_0, t)$, where $\mathcal{IGSO}_3$ is the isotropic Gaussian distribution on $\mathrm{SO}(3)$ [48].

The associated reverse-time stochastic differential equation (SDE) follows:

$$\begin{aligned}
\mathrm{d}\mathbf{T}_t &= \mathbf{P}\left[-\frac{1}{2}\beta_t\mathbf{T}_t - \beta_t\nabla\log p_t(\mathbf{T}_t)\right]\mathrm{d}t + \sqrt{\beta_t}\mathrm{P}\mathrm{d}\bar{\mathbf{w}}_t, \\
\mathrm{d}\mathbf{R}_t &= -\frac{\mathrm{d}}{\mathrm{d}t}\sigma_t^2\nabla\log p_t(\mathbf{R}_t)\mathrm{d}t + \sqrt{\frac{\mathrm{d}}{\mathrm{d}t}\sigma_t^2}\mathrm{d}\bar{\mathbf{w}}_t^{\mathrm{SO}(3)},
\end{aligned} \tag{6}$$

where $\bar{\mathbf{w}}_t$ and $\bar{\mathbf{w}}_t^{\mathrm{SO}(3)}$ denote standard Wiener processes in the reverse time.

The reverse process can be approximated by a neural network through the *denoising score matching* loss for translation and rotation:

$$\begin{aligned}
\mathcal{L}(\theta) =& \mathcal{L}^{\mathbf{T}}(\theta) + \mathcal{L}^{\mathbf{R}}(\theta) \\
=& \mathbb{E}\left[\lambda(t)\|s_\theta(\mathbf{T}_t, t) - \nabla_{\mathbf{T}_t}\log p_{t|0}(\mathbf{T}_t|\mathbf{T}_0)\|^2\right] \\
&+ \mathbb{E}\left[\lambda^r(t)\|s_\theta^r(\mathbf{R}_t, t) - \nabla_{\mathbf{R}_t}\log p_{t|0}(\mathbf{R}_t|\mathbf{R}_0)\|^2\right],
\end{aligned} \tag{7}$$

where $\lambda(t)$ and $\lambda^r(t)$ are time-dependent weights, $s_\theta(\mathbf{T_t}, \mathbf{t})$ and $s_\theta^r(\mathbf{R}_t, t)$ are the score networks commonly parameterized with shared weights. The expectations are taken over diffusion time $t \sim \mathcal{U}[t_{\min}, 1]$, and over noisy and clean data pairs from the forward process $(\mathbf{T}_0, \mathbf{T}_t)$ and $(\mathbf{R}_0, \mathbf{R}_t)$.

C Derivation of Equations in Section 3.2

Here we provide a more rigorous derivation of equations in Section 3.2. For clarity, we omit the conditioning variable $\mathcal{P}$ in the intermediate steps.

As defined in Equation (1), our goal is to model the frame-level conditional distribution $p(\mathbf{x}^l|\mathbf{x}^{<l})$. To achieve this, we encode the previous frames $\{\mathbf{x}^i\}_{i=1}^{l-1}$ into a sequence of latent embeddings $\{\mathbf{h}^i\}_{i=1}^{l-1}$, and model inter-frame dependencies in this latent space.

By applying the Bayes' rule, we can factor the joint distribution over the current conformation $\mathbf{x}^l$ and intermediate latent embeddings $\mathbf{h} = (\mathbf{h}^l, \mathbf{h}^{<l})$ as:

$$p(\mathbf{x}^l, \mathbf{h}^l, \mathbf{h}^{<l} \mid \mathbf{x}^{<l}) = p(\mathbf{x}^l \mid \mathbf{h}^l, \mathbf{h}^{<l}, \mathbf{x}^{<l})\, p(\mathbf{h}^l \mid \mathbf{h}^{<l}, \mathbf{x}^{<l})\, p(\mathbf{h}^{<l} \mid \mathbf{x}^{<l}).$$

Integrating both sides over the latent variables $\mathbf{h}$ yields:

$$p(\mathbf{x}^l \mid \mathbf{x}^{<l}) = \int_{\mathbf{h}} p(\mathbf{x}^l \mid \mathbf{h}^l, \mathbf{h}^{<l}, \mathbf{x}^{<l})\, p(\mathbf{h}^l \mid \mathbf{h}^{<l}, \mathbf{x}^{<l})\, p(\mathbf{h}^{<l} \mid \mathbf{x}^{<l})\, \mathrm{d}\mathbf{h}. \tag{8}$$

In our approach, both $p(\mathbf{h}^l \mid \mathbf{h}^{<l}, \mathbf{x}^{<l})$ and $p(\mathbf{h}^{<l} \mid \mathbf{x}^{<l})$ are modeled using deterministic neural networks: an encoder $f_\eta^{\text{enc}}(\mathbf{x}^i)$ and an autoregressive temporal module $f_\xi^{\text{temp}}(\mathbf{h}^{<l})$, respectively.

These mappings reduce both $p(\mathbf{h}^l \mid \mathbf{h}^{<l}, \mathbf{x}^{<l})$ and $p(\mathbf{h}^{<l} \mid \mathbf{x}^{<l})$ to Dirac delta functions, and the conditional dependencies can be simplified as:

$$\begin{aligned} p(\mathbf{h}^l|\mathbf{h}^{<l}, \mathbf{x}^{<l}) &= p(\mathbf{h}^l|\mathbf{h}^{<l}) \\ p(\mathbf{x}^l|\mathbf{h}^l, \mathbf{h}^{<l}, \mathbf{x}^{<l}) &= p(\mathbf{x}^l|\mathbf{h}^l). \end{aligned}$$

Substituting into Equation (8) gives:

$$p(\mathbf{x}^l|\mathbf{x}^{<l}) = \int_{\mathbf{h}} p(\mathbf{x}^l|\mathbf{h}^l) \cdot p(\mathbf{h}^l|\mathbf{h}^{<l}) \cdot p(\mathbf{h}^{<l}|\mathbf{x}^{<l}) \mathrm{d}\mathbf{h}. \tag{9}$$

Again, due to the deterministic nature of the encoder and temporal module, there is no marginalization involved in Equation (9), yielding:

$$\begin{aligned} p(\mathbf{x}^l|\mathbf{x}^{<l}) &= p(\mathbf{x}^l|\mathbf{h}^l) \\ \text{where} \quad \mathbf{h}^i &= f_\eta^{\text{enc}}(\mathbf{x}^i, \mathcal{P}), \quad i = 1, 2, \ldots, l-1 \\ \text{and} \quad \mathbf{h}^l &= f_\xi^{\text{temp}}\left(\mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^{l-1}\right). \end{aligned}$$

Finally, we approximate $p(\mathbf{x}^l|\mathbf{h}^l)$ with a parameterized model $p_\theta^{\text{enc}}(\mathbf{x}^l|\mathbf{h}^l)$.

D Method Details

D.1 Detailed Module Architectures

Encoding Layer. The protein-specific single and pair representations are obtained from the Evoformer stack of a pretrained OpenFold model (with frozen weights), after three recycle iterations. In addition, we encode residue-level sequence information by combining sinusoidal positional embeddings of residue indices with learnable embeddings for the 20 standard amino acid types. These features are concatenated with the single representation from the `FoldingModule`.

To encode the structural information of each conformation frame, we introduced the `FrameEncoder`, a pseudo-beta-carbon (C_β) coordinate encoder similar to the `InputEmbedding` module from ALPHAFLOW [18] (without diffusion time embedding). Specifically, this module first compute the pairwise distances between residues using C_β coordinates. These distances are then binned, embedded into latent embedding, and further refined through triangular update blocks including triangle attention and multiplication updates [20]. See Algorithm 1 for the specifics. The resulting per-frame C_β pair embedding $\mathbf{z}^l_{\text{Frame}}$ is concatenated with the pair representation from `FoldingModule`.

Both single and pair embeddings are projected into the same dimension of d for simplicity, forming the latent embedding $\mathbf{h}^l = [\mathbf{s}^l, \mathbf{z}^l]$ for each frame. See detailed illustration in Figure 7.

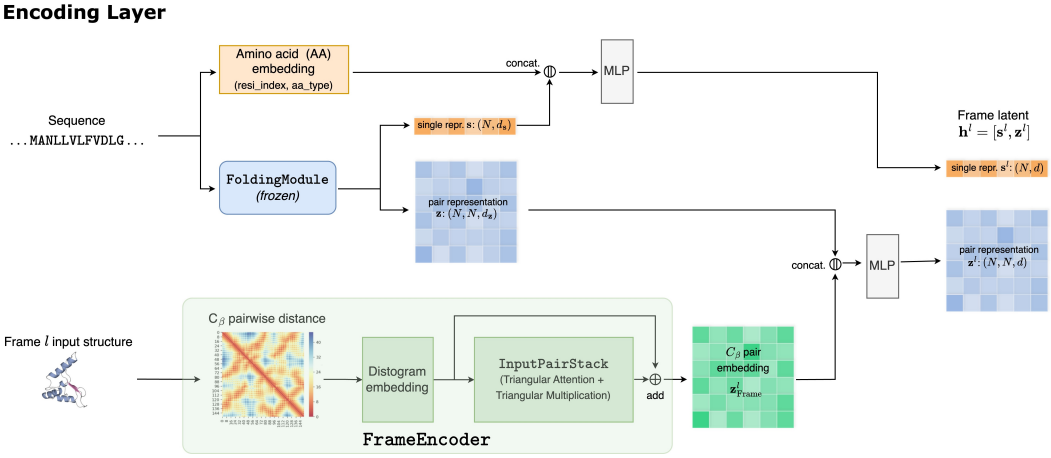

Figure 7: Architecture details of the *Encoding Layer*. A frozen `FoldingModule` encodes the protein-specific information from its sequence, containing prior knowledge on its chemical environment and folding structures. The single representation is further concatenated with additional amino acid embeddings and projected to a hidden dimension of size d; The pair representation is concatenated with frame conformation information, encoded in C_β pair embedding, and projected to a hidden dimension of size d. Both frame-level single and pair embeddings form the frame-level latent for downstream modules.

Algorithm 1 FRAMEENCODER

Input: Pseudo beta carbon (C_β) coordinates $\mathbf{x} \in \mathbb{R}^{N\times 3}$, time $t \in [0, 1]$
Output: Input pair embedding $\mathbf{z} \in \mathbb{R}^{N\times N\times 64}$
$\mathbf{z}_{ij} \leftarrow \|\mathbf{x}_i - \mathbf{x}_j\|$
$\mathbf{z}_{ij} \leftarrow \text{Bin}(\mathbf{z}_{ij}, \text{min} = 3.25\ \text{Å}, \text{max} = 50.75\ \text{Å}, N_{\text{bins}} = 39)$
$\mathbf{z}_{ij} \leftarrow \text{Linear}(\text{OneHot}(\mathbf{z}_{ij}))$
for $l \leftarrow 1$ to $N_{\text{blocks}} = 4$ **do**
  $\{\mathbf{z}\}_{ij}$ += TriangleAttentionStartingNode$(\mathbf{z}_{ij}, c = 64, N_{\text{head}} = 4)$
  $\{\mathbf{z}\}_{ij}$ += TriangleAttentionEndingNode$(\mathbf{z}_{ij}, c = 64, N_{\text{head}} = 4))$
  $\{\mathbf{z}\}_{ij}$ += TriangleMultiplicationOutgoing$(\mathbf{z}_{ij}, c = 64)$
  $\{\mathbf{z}\}_{ij}$ += TriangleMultiplicationIncoming$(\mathbf{z}_{ij}, c = 64)$
  $\{\mathbf{z}\}_{ij}$ += PairTransition$(\mathbf{z}_{ij}, n = 2)$
end for
$\mathbf{z}_{ij} = \text{LayerNorm}(\mathbf{z}_{ij})$

Trajectory Module. In the *Trajectory Module*, we interleave layers of `StructuralUpdate` and `TemporalUpdate` to iteratively update the latent $[\mathbf{s}^l, \mathbf{z}^l]$, enabling the temporal reasoning across frames and structural refinement within each frame.

For the `StructuralUpdate`, we adopt a `Pairformer` block from AlphaFold 3 [2], which jointly updates the single and pair embeddings of the current frame through structural reasoning. After the `StructuralUpdate`, the pair embedding is flattened from $[N, N, d]$ to $[N \times N, d]$ and concatenated with the single embedding before being passed into the `TemporalUpdate`.

To model temporal dependencies between frames, we use a lightweight Llama architecture [42]. We transpose the input such that the temporal dimension is treated as the sequence axis for channel-wise self-attention across time. Rotary positional encoding [41] is applied to encode the temporal position for each frame. A causal attention mask is applied to restrict each frame to only attend to previous frames. After the temporal update, the latent embedding are reshaped and split back into single and pair embeddings.

Figure 8 and Table 5 provide the detailed module architecture and hyperparameter configurations, respectively. A `StructuralUpdate` block is included for every two `TemporalUpdate` layers.

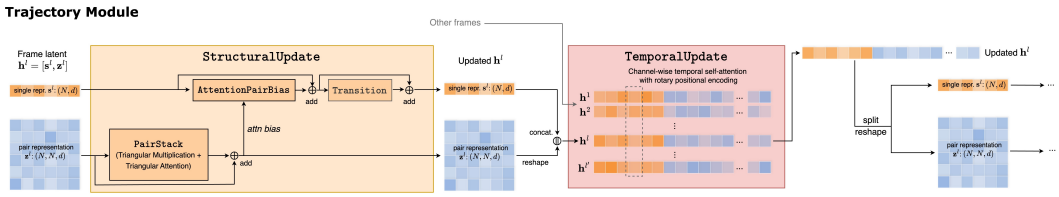

Figure 8: Architecture details of the *Trajectory Module*. *Trajectory Module* contains interleaving blocks of `StructuralUpdate` and `TemporalUpdate` (only one block of each is shown). `StructuralUpdate` leverages the *Pairformer* architecture from Abramson et al. [1], updating the pair embeddings with triangular updates and the single embeddings using with pair bias from the updated pair. The updated pair embeddings are flatten and concatenate with single embedding for channel-wise temporal update. The attention is applied along the temporal dimension and update each single and pair embedding channels independently. The embeddings from `TemporalUpdate` are split and reshape back into single and pair embeddings.

Table 5: Hyperparameter choices of *Trajectory Module*

Hyperparameters	Values
Lightweight Llama (`TemporalUpdate`)	
Number of layers	8
Dimension of the MLP embeddings	256
Dimension of the hidden embeddings	128
Number of attention heads	4
Pairformer (`StructuralUpdate`)	
Dimension of single embeddings	128
Dimension of pair embeddings	128
Dimension of triangle multiplication hidden embeddings	128
Number of triangle attention heads	4
Dimension of pair attention embeddings	32
Transition layer expanding factor	4
Pair attention dropout rate	0.25

Structure Decoder. Conformation generation, conditioned on the temporal signals from the *Trajectory Module*, is performed using the model architecture from CONFDIFF [44], As shown in Figure 9, following CONFDIFF, the inputs to the denoising model include diffusion time t, pairwise distance between residue rigids, and residue indices (not shown). They are encoded and concatenated with the single and pair embeddings from `FoldingModule`. In addition, the single and pair embeddings from

the *Trajectory Module* are projected back to the latent dimension $d_{\mathbf{s}}$ and $d_{\mathbf{z}}$, respectively, to modify the single and pair embeddings used by CONFDIFF.

The core of the *Structure Decoder* consists of multiple `IPA-transformer` blocks, which update single and pair embeddings as well as the SE(3) rigids of noisy conformations. In the final block, torsional angles are predicted by `TorsionPred` and, together with denoised rigids, to reconstruct the atomic structure of generated conformation. The corresponding hyperparameter settings are summarized in Table 6.

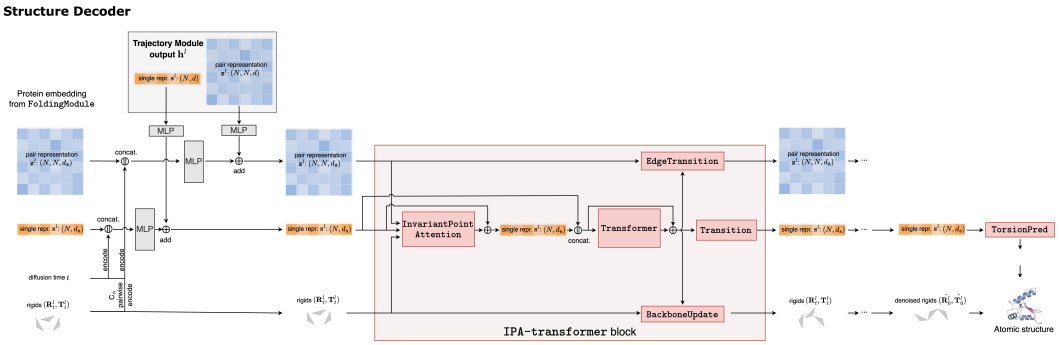

Figure 9: Architecture details of the *Structure Decoder*. Single and pair embeddings from *Trajectory Module* is used to update the original embeddings from `FoldingModule`. The resulting embeddings are fed into blocks of `IPA-transformer` to update single, pair embeddings and denoise rigids, SE(3) representation of protein backbone conformations. Denoised rigids together with torsion angles predicted by `TorsionPred` recovers the atomic structure of protein conformation at this frame.

Table 6: Hyperparameter choices of the *Structure Decoder*

Hyperparameters	**Values**
Neural network	
Number of IPA blocks	4
Dimension of single embedding ($d_{\mathbf{s}}$)	256
Dimension of pair embedding ($d_{\mathbf{z}}$)	128
Dimension IPA hidden embedding	256
Number of IPA attention heads	4
Number of IPA query points	8
Number of IPA value points	12
Number of transformer attention heads	4
Number of transformer layers	2
SE(3)-diffusion SDE	
Number of diffusion steps	200
Translation scheduler	Linear
Translation $\beta_{\min}$	0.1
Translation $\beta_{\max}$	20
Rotation scheduler	Logarithmic
Rotation $\sigma_{\min}$	0.1
Rotation $\sigma_{\max}$	1.5

D.2 Training and Inference Details

We train all CONFROVER models on the trajectories from the ATLAS training set, following the train-validation-test split of previous works [18, 44, 19]. Specifically, we exclude the training proteins longer than 384 amino acid residues, leading to 1080 training proteins.

Most components of CONFROVER models were trained from scratch, except for the `FoldingModule`, where we used frozen weights from OpenFold to extract the single and pair representations from three recycling iterations, and the `DiffusionDecoder`, which was initialized from `ConfDiff-OF-r3-MD` checkpoint provided by the authors[1].

During each training epoch, we randomly sample stride length from 2^0 to 2^{10} to extract sub-trajectories of length $L = 9$ at varying time scales. With the use of causal transformers, input frames were shifted forward by one frame with a `[MASK]` token padded at the beginning of the trajectory. Combined with the use of a causal mask in temporal attention, the design ensures that each frame is trained to sample conditioned only on previous frames and the first frame is generated unconditionally using only the `[MASK]` token as input.

We trained main CONFROVER model 180 epochs ($\sim$ 37 hrs) and CONFROVER–INTERP model for additional 220 epochs ($\sim$ 45 hrs). Additional training hyperparameters can be found in Table 7. All model training and sampling were carried out using 8 NVIDIA H100 GPUs.

Table 7: Training hyperparameters

Hyperparameters	**Values**
Batch Size	1
Frames Num	8
Gradient Clip	1.0
Learning Rate	1×10^{-4}
Optimizer	Adam (weight decay = 0.)

D.3 Training and Inference Cost

CONFROVER models in this work contains 19.6 M trainable parameters. All model training and sampling were carried out using 8 NVIDIA H100 GPUs with Distributed Data Parallel. We trained the main CONFROVER model for 180 epochs ($\sim$ 37 hrs) and CONFROVER–INTERP model for additional 220 epochs ($\sim$ 45 hrs).

Inference cost varies with protein size and benchmark setups. To measure the potential speedups from using CONFROVER compared with classic MD simulation, we measured the wall-clock time required to generate 100 ns trajectories (80 frames) for ATLAS proteins of varying sizes and report the average inference time per size bucket. For comparison, we selected a representative protein from each bucket and estimated the time required to simulate 100 ns using OpenMM with implicit solvent. Both are performed on a single NVIDIA H100-80G GPU. As shown in the Table 8, CONFROVER provides significant speedup for 100 ns simulation, with even more pronounced acceleration for larger proteins.

Table 8: Runtime comparison (in minutes) across different protein sequence lengths. Speedup is computed as MD runtime divided by CONFROVER runtime

Sequence length	(0,150)	[150,300)	[300,450)	[450,600)	[600,724]
ConfRover	6.99	7.53	10.92	15.88	20.83
MD	104.54	207.92	386.69	651.29	1099.13
Speedup	**14.95**×	**27.61**×	**35.41**×	**41.01**×	**52.77**×

For our large-scale experimental benchmarks, *multi-start* trajectory simulation (2,700+ trajectories) took 8 hours and 30 minutes, while time-independent sampling (250 conformations per protein) took 3 hours and 20 minutes.

[1] https://github.com/bytedance/ConfDiff

E Additional Experimental Results

E.1 Trajectory Simulation: *multi-start*

Benchmark Curation. In *multi-start*, we sample short trajectories from varying starting point while ensuring the generation within the scope of the reference trajectory. For example, we select frame index of 1000, 3000, 5000, and 7000 as starting frames for stride $S = 128/256$, resulting in 12 test trajectories from triplicates; frame index 1000, 3000, 5000 as starting frames for stride $S = 512$, and frame index 1000 for stride $S = 1024$ to avoid exceeding total of 10000 frames. This provided us 2,706 different starting conditions from 82 proteins from the ATLAS test set for evaluation.

PCA Projection. Following previous works [18, 44], we project the Cα coordinates of proteins into a reduced PCA space to focus on the principal dimensions that best capture the structural variations observed in MD simulations. Briefly, for each protein, conformations from triplicate MD simulations in ATLAS are all aligned to the reference conformation (input structure for simulations). The coordinates of each Cα atoms are then flattened and used to fit a per-protein PCA model. For all subsequent analyses, sampled conformation are aligned to the reference structure before computing their PCA projections.

Additional Results. Here we also include the scatterplot of Pearson correlation in Figure 10 and additional metrics from the *multi-start* experiments at different strides in Table 9 (one experimental included). Across different strides, CONFROVER models consistently outperforms MDGEN in recovering the correct level of dynamics.

Table 9: Additional metrics from the *multi-start* benchmark. Results for different strides are shown in separate blocks. The better score in each block is highlighted in **bold** (excluding diversity). One inference run per model is used for this comparison.

	Diversity	MAE on PCA-2D (↓)			MAE on CA coordinates (↓)			Quality
	Pairwise RMSD	Trajectory	Frame	Frame Next	Trajectory	Frame	Frame Next	PepBond Break % (↓)
Stride=128								
MDGEN	1.26	6.53	1.28	0.91	5.02	0.96	0.71	27.9
CONFROVER	1.63	**3.10**	**1.10**	**0.74**	**3.83**	**0.81**	**0.64**	**17.3**
Stride=256								
MDGEN	1.34	7.66	1.54	1.06	6.00	1.14	0.85	27.9
CONFROVER	1.78	**3.91**	**1.28**	**0.87**	**4.60**	**0.91**	**0.75**	**16.6**
Stride=512								
MDGEN	1.40	9.12	1.94	1.25	7.27	1.41	1.01	28.0
CONFROVER	1.89	**4.84**	**1.53**	**1.01**	**5.66**	**1.07**	**0.89**	**16.7**
Stride=1024								
MDGEN	1.51	11.48	2.62	1.55	9.04	1.80	1.24	28.1
CONFROVER	2.04	**6.75**	**1.89**	**1.25**	**7.39**	**1.26**	**1.09**	**16.7**

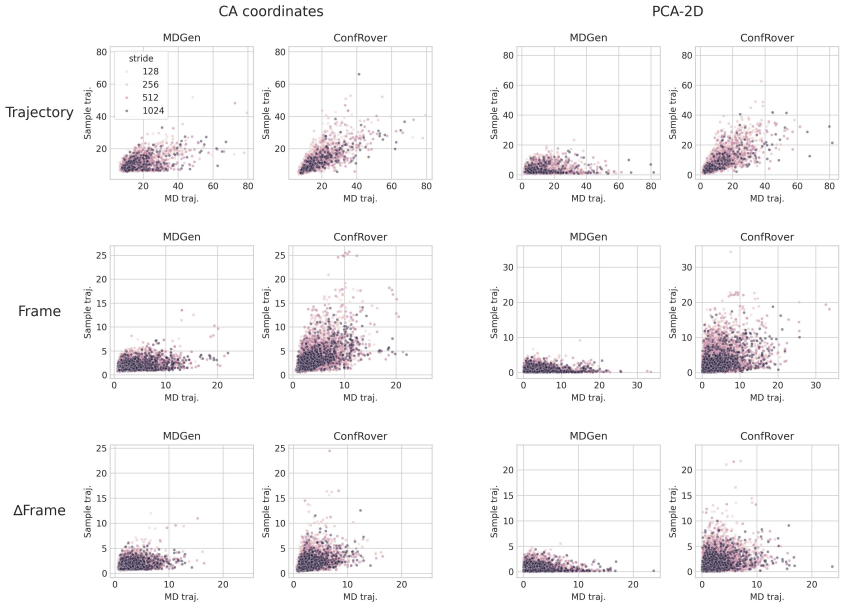

Figure 10: Scatterplots of conformation changes in sample trajectories versus those in the reference trajectories, measured by trajectory-level conformation changes, frame-level conformation changes, and next-frame difference (ΔFrame), measured by the RMSD of alpha carbons (unit: Å) or L^2 distance in the projected PCA space. MDGEN tends underestimate the magnitude of conformation changes while CONFROVER generate samples at similar level as the MD reference. The exact match of measured conformation changes is not possible due to stochastic sampling in both MD simulation and generative models.

Additional Visualization of Trajectory. We additional unfiltered examples (randomly selected) for visual comparison of conformations generated by different models in the *multi-start* experiments, as shown in Figure 11.

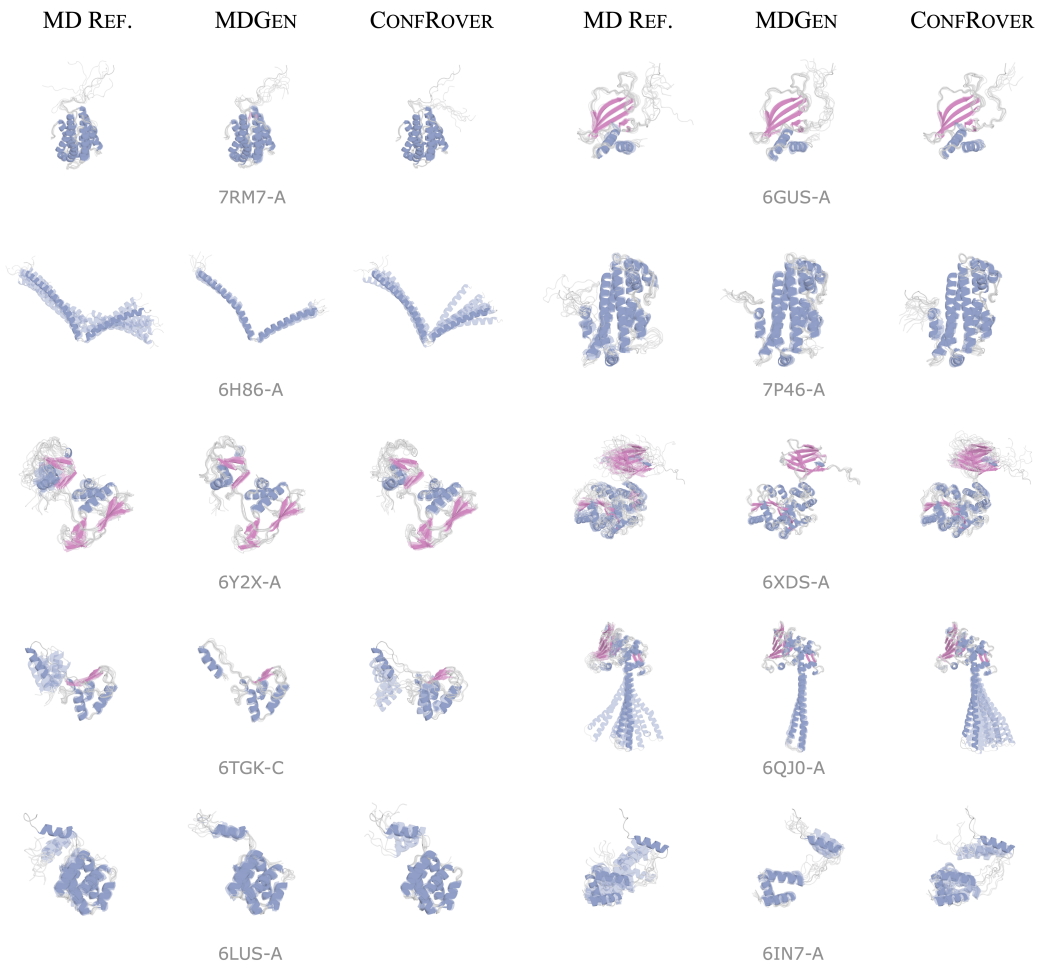

Figure 11: Visualization of 10 trajectories randomly selected from the *Multi-start* benchmark. Trajectory conformations are colored by their secondary structures and superposed to show the dynamic ensemble. MDGEN exhibits primarily local motions while CONFROVER better reflects the motions observed in MD reference.

E.2 Trajectory Simulation: 100 ns Long Trajectory Simulation on ATLAS

Details on Evaluation Metrics. Conformational state recovery is evaluated by comparing the distribution of model-generated and reference conformations in a PCA space. Same as in the *multi-start* benchmark, each conformation is projected into the PCA space parameterized by the 3D coordinates of Cα atoms. To compare distributions, each principal component is discretized into 10 evenly spaced bins. After projecting the conformations into this space, we count their occurrences in each bin and compute the distribution similarity using Jensen–Shannon Distance (JSD). We also binarize the occupancy counts to compute precision, recall, and F1-score—evaluating whether sampled conformations fall within known states, following prior work [32, 44, 49].

Dynamic mode recovery is assessed using time-lagged independent component analysis (tICA) applied separately to reference and generated trajectories across varying lag times. After fitting, we extract tICA coefficients for each Cα atom and compute Pearson correlations between the per-residue contributions to the leading components, evaluating alignment of dynamic modes.

Additional Visualizations. We additionally provide unfiltered (randomly selected) PCA plots in Figure 12.

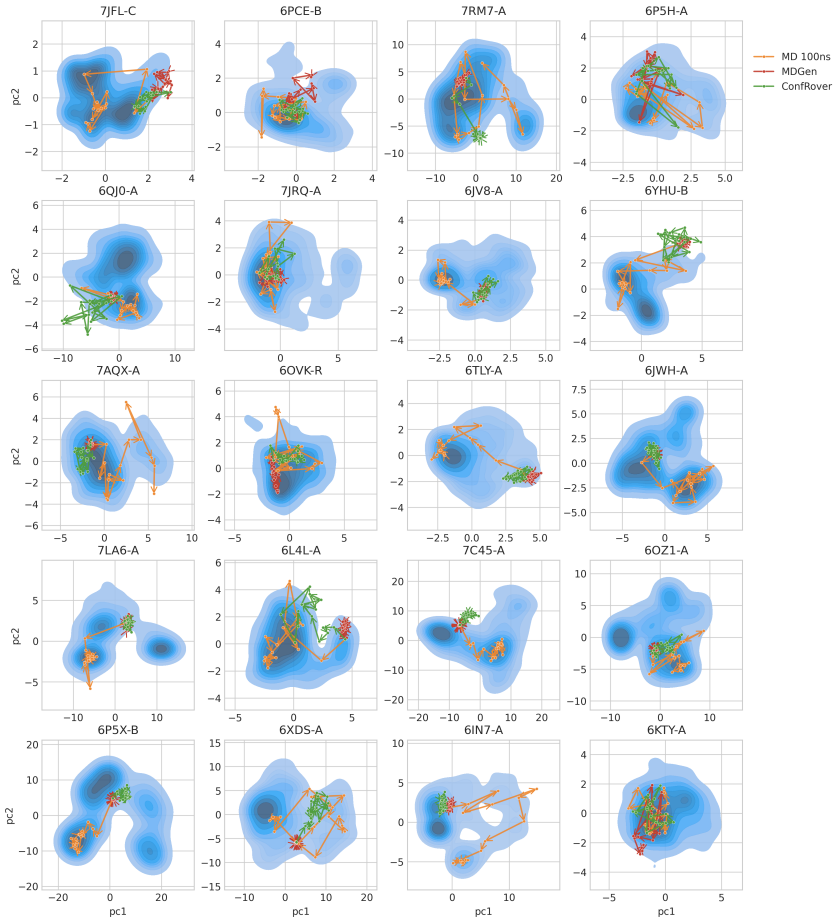

Figure 12: Visualization the ATLAS-100ns trajectories from 20 randomly selected cases. The blue background indicates the density of the ground-truth conformation distribution from MD reference. CONFROVER shows improved conformation state recovery in several cases (e.g., 7JRQ-A, 6YHU-B, 7AQX-A, 6L4L-A, 6OZ1-A, etc), sampling more diverse conformations. Yet, the gap between the oracle MD 100NS and deep learning models is evident in some cases (e.g., 7JFL-C, 6TLY-A, 6JWH-A, etc)

E.3 Time-independent Conformation Sampling

We follow the evaluation protocol of Ye et al. [47] to assess time-independent conformation sampling on the ATLAS test set. For each protein, 250 independent samples are generated. Since MDGEN does not support time-independent sampling, we approximate its performance using samples from its 100-ns trajectory, serving as a sequential-sampling baseline. The performance of state-of-the-art models, ALPHAFLOW and CONFDIFF, is taken from Table 10 of Ye et al. [47], using their best-performing variants: ALPHAFLOW-MD and CONFDIFF-OPEN-MD. Full results are shown in Table 10.

Table 10: Performance on time-independent conformation generation on ATLAS. A total of 250 conformations were sampled for each protein, and the mean and standard deviation of metrics are computed from five independent runs. The best performance is highlighted in **bold**, and the second-best is underlined.CONFROVER-TRAJ and MDGEN are trained to exclusive generate trajectories. MDGEN does not support time-independent sampling and the metrics are evaluated on sequential sampling result.

	Diversity		Flexibility: *Pearson* r on			Distributional accuracy			
	Pairwise RMSD	RMSF	Pairwise RMSD ↑	Global RMSF ↑	Per target RMSF ↑	RMWD ↓	MD PCA $\mathcal{W}_2$ ↓	Joint PCA $\mathcal{W}_2$↓	PC sim > 0.5 %↑
ALPHAFLOW	2.85±0.05	1.63±0.01	**0.56±0.06**	**0.66±0.04**	**0.85±0.01**	**2.62±0.03**	1.52±0.05	2.26±0.03	39.52±3.22
CONFDIFF	3.59±0.02	2.18±0.02	0.54±0.00	0.65±0.00	**0.85±0.00**	2.70±0.01	1.44±0.00	**2.22±0.04**	**41.00±1.12**
MDGEN	1.34±0.05	0.77±0.00	0.47±0.04	0.50±0.01	0.72±0.02	2.78±0.04	1.86±0.03	2.44±0.04	10.24±3.18
CONFROVER-TRAJ	3.19±0.04	1.74±0.00	0.48±0.00	0.62±0.01	0.84±0.01	2.85±0.02	**1.43±0.01**	2.30±0.01	37.08±2.83
CONFROVER	3.68±0.04	2.25±0.01	0.51±0.01	0.64±0.01	**0.85±0.00**	2.66±0.02	1.47±0.03	2.23±0.04	38.28±4.44

	Ensemble observables				Quality	
	Weak contacts J ↑	Transient contacts J↑	Exposed residue J ↑	Exposed MI matrix ρ ↑	CA clash % ↓	PepBond break % ↓
ALPHAFLOW	0.62±0.00	**0.41±0.00**	**0.69±0.01**	**0.35±0.01**	**0.00±0.00**	22.00±0.19
CONFDIFF	**0.64±0.00**	0.40±0.00	0.67±0.00	0.33±0.00	0.50±0.00	**6.20±0.00**
MDGEN	0.51±0.01	0.28±0.01	0.57±0.01	0.26±0.01	0.24±0.05	30.84±1.69
CONFROVER-TRAJ	0.53±0.01	0.36±0.00	0.58±0.01	0.27±0.00	0.32±0.04	11.78±0.24
CONFROVER	0.62±0.01	0.37±0.01	0.66±0.01	0.32±0.01	0.50±0.00	19.18±0.08

E.4 Conformation Interpolation

In *conformation interpolation* experiment, we selected trajectories from *multi-start*. These trajectories exhibit sufficient conformation changes (e.g., RMSD between the start and end frames > 4 Å) and clear interpolation path in the PCA space. The list of selected cases are in Table 11.

The L^2 distance of generated intermediate frames to the start and end frames in the PCA spaces are reported in Figure 13. Distances are normalized by the distance between start and end frames. Similar to the results measured by Cα-RMSD, CONFROVER-INTERP shows smooth interpolation between the start and end frames while CONFROVER does not. This result shows that by continue training the model on interpolation objective, CONFROVER can learn to generated interpolating trajectories conditioned on the end state.

Table 11: List of 38 selected cases from *multi-init* for interpolation test. Naming conventions: "{PDB_ID}_{Chain_ID}_R{ATLAS repeat}F{Starting index}S{Stride}"

0	5ZNJ-A-R2F1000S512	6L4L-A-R1F3000S256	6TGK-C-R1F5000S256	7JFL-C-R2F3000S256
1	6E7E-A-R3F3000S512	6LRD-A-R1F1000S1024	6TLY-A-R2F1000S256	7JRQ-A-R1F1000S1024
2	6GUS-A-R2F7000S256	6LUS-A-R2F3000S128	6XDS-A-R1F1000S128	7LA6-A-R1F1000S512
3	6H49-A-R2F1000S1024	6OVK-R-R2F7000S128	6XRX-A-R3F5000S128	7LP1-A-R2F5000S256
4	6H86-A-R2F1000S1024	6OZ1-A-R1F1000S512	6Y2X-A-R2F3000S512	7P41-D-R2F3000S256
5	6IN7-A-R3F5000S128	6P5H-A-R1F1000S1024	7AEX-A-R3F5000S256	7P46-A-R3F5000S256
6	6J56-A-R1F3000S256	6P5X-B-R1F3000S256	7AQX-A-R2F3000S512	7RM7-A-R3F3000S128
7	6JPT-A-R2F5000S512	6Q9C-A-R3F3000S256	7ASG-A-R1F7000S128	7S86-A-R3F5000S256
8	6JV8-A-R3F1000S256	6QJ0-A-R1F1000S256	7BWF-B-R3F3000S128	
9	6KTY-A-R3F1000S1024	6RRV-A-R1F1000S256	7C45-A-R1F5000S512	

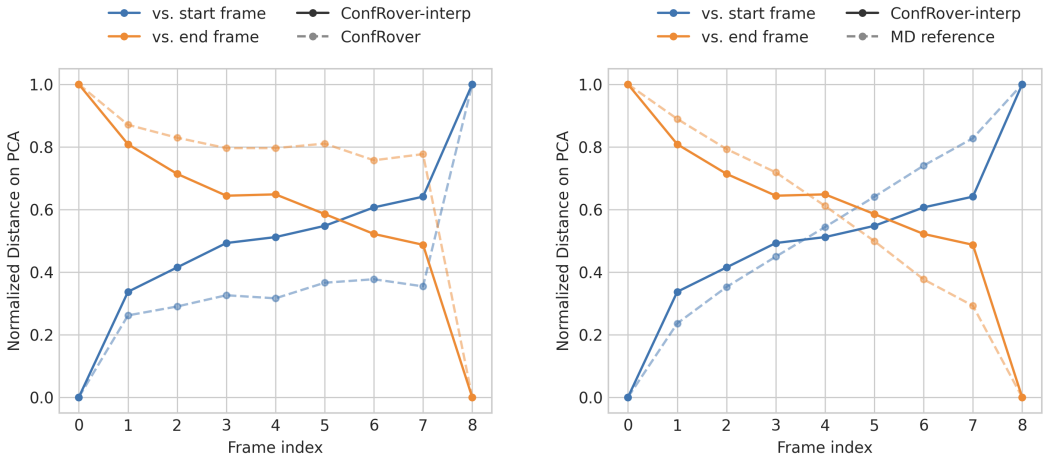

Figure 13: Normalized PCA distance of intermediate frames to the start and end frames, averaged over 38 cases selected from the *multi-start* benchmark. [Left] a comparison between CONFROVER-INTERP and CONFROVER, where CONFROVER-INTERP generates smooth pathways while CONFROVER does not; [Right] a comparison between CONFROVER-INTERP and reference trajectories.

Additional Visualizations. Here we include additional visualization on interpolation results.

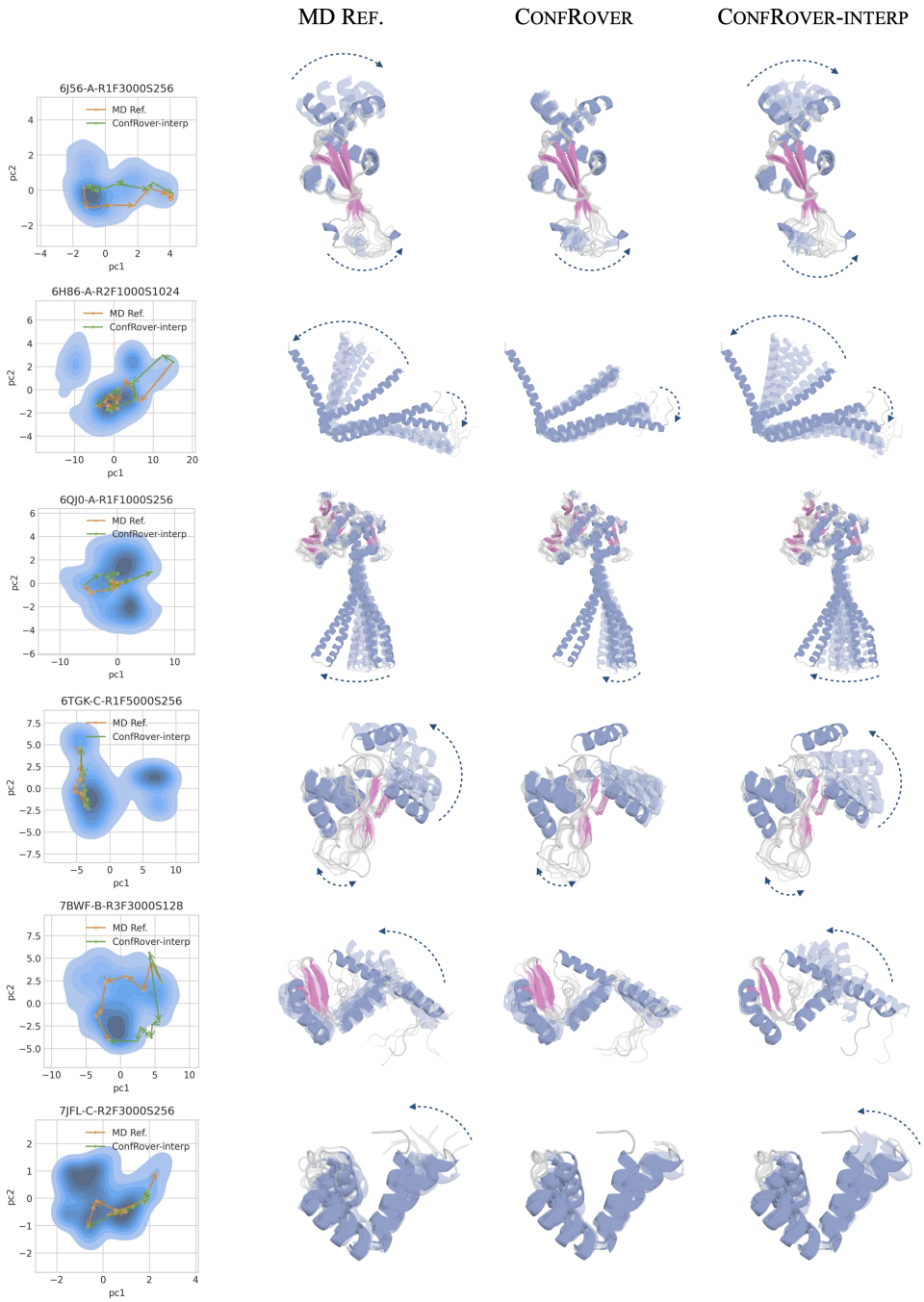

Figure 14: Example interpolations results. CONFROVER-INTERP generates smooth pathways between the start and end frames, capturing the dynamics observed with the MD reference while CONFROVER does not show the correct intermediate conformations. Start and end frames are shown as solid structures; intermediate conformations are shown in fading colors. Main motions are indicated by blue dashed arrows. These examples highlight the difference between the original CONFROVER and CONFROVER-INTERP that further trained on the interpolation objective. The original CONFROVER can miss key motions of the transition while CONFROVER-INTERP correctly capture these motions.

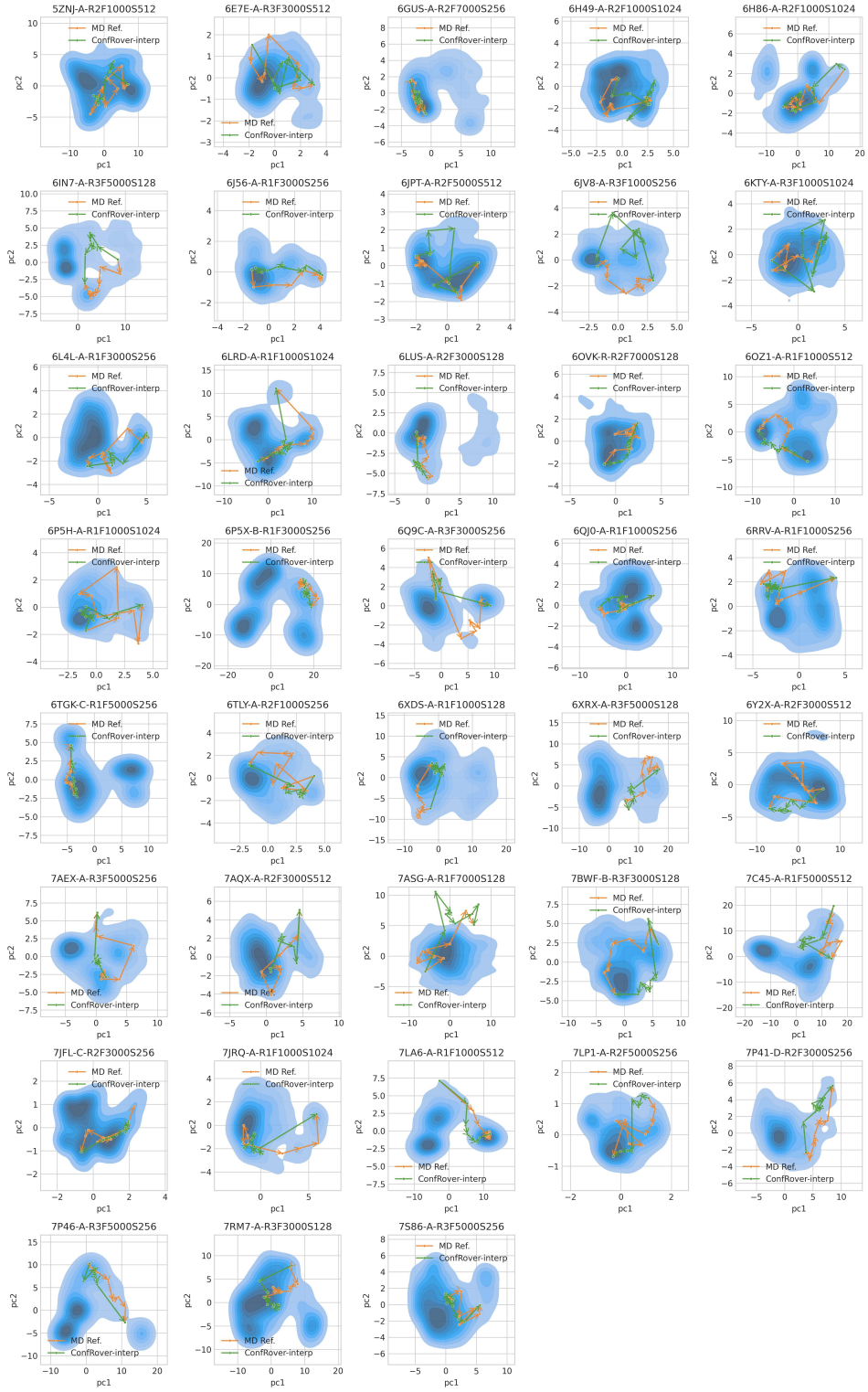

Figure 15: PCA plot of 38 selected interpolation cases. MD reference trajectories and results from CONFROVER-INTERP are shown in each plot.

E.5 Retraining MDGen for Experiment-Specific Setups

Due to its non-autoregressive design and training on fixed-length trajectories, MDGEN cannot directly generate sequences of varying lengths. Therefore, in our evaluation, we generate trajectories using the original settings (named MDGEN-S{stride}F{length}) and subsample them to match the evaluation setup. However, this post-processing may introduce artifacts. To address this concern, we retrain MDGEN under the evaluation settings and compare the results on the *Multi-start* benchmark (stride = 256) and 100 ns long trajectory simulations, as shown below in Table 12, Table 13, Table 14 and Figure 16. Experimental results show no significant difference of performance for key metrics observed comparing MDGEN with post-processed results and models specifically trained at the evaluation settings, suggesting no evident decrease of trajectory quality from the subsampling post-process. We use one inference run per model for this experiment.

Table 12: Compare MDGEN-S256F9 with MDGEN from subsampling post-process. Here is the table summarizing the Pearson correlations of conformation changes between sampled and reference trajectories in *multi-start*. MDGEN-S256F9 is trained and sampled with stride of 256 MD snapshots and length of 9 frames. The best scores are highlighted in **bold**.

	Cα coordinates		
	Trajectory	Frame	ΔFrame
MDGEN	0.57	0.46	0.41
MDGEN-S256F9	0.56	0.45	0.38
CONFROVER	**0.77**	**0.62**	**0.53**
	PCA 2D		
	Trajectory	Frame	ΔFrame
MDGEN	0.18	0.13	0.11
MDGEN-S256F9	0.21	0.19	0.11
CONFROVER	**0.75**	**0.5**	**0.44**

Table 13: Compare MDGEN-S256F9 with MDGEN from subsampling post-process. Here is the table summarizing additional metrics in *multi-start* benchmark. MDGEN-S256F9 is trained and sampled with stride of 256 MD snapshots and length of 9 frames. The best scores are highlighted in **bold**.

	Diversity	MAE on PCA-2D (↓)			MAE on CA coordinates (↓)			Quality
	Pairwise RMSD	Traj.	Frame	ΔFrame	Traj.	Frame	ΔFrame	PepBond Break %(↓)
MDGEN	1.34	7.66	1.54	1.06	6.00	1.14	0.85	27.9
MDGEN-S256F9	1.59	6.90	1.47	1.03	5.33	1.13	0.82	**16.2**
CONFROVER	1.78	**3.91**	**1.28**	**0.87**	**4.60**	**0.91**	**0.75**	16.6

Table 14: Compare MDGEN-S256F9 with MDGEN from subsampling post-process. Here is the table summarizing the state recovery performance in 100 ns long trajectory simulation. MDGEN-S120F80 is trained and sampled with stride of 120 MD snapshots and length of 80 frames. The best scores are highlighted in **bold**. MD 100NS is included as the oracle.

	JSD (↓)	Recall (↑)	F1 (↑)
MD 100NS	0.31	0.67	0.79
MDGEN	0.56	0.30	0.44
MDGEN-S120F80	0.57	0.29	0.42
CONFROVER	**0.51**	**0.42**	**0.58**

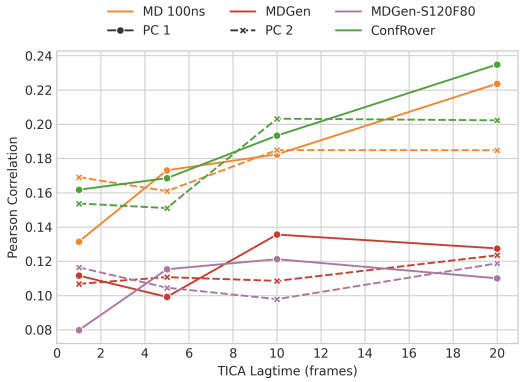

Figure 16: Compare MDGEN-S256F9 with MDGEN from subsampling post-process. This figure shows correlations of main dynamic modes between sampled trajectory and reference trajectory. MDGEN-S120F80 is trained and sampled with stride of 120 MD snapshots and length of 80 frames.

E.6 Reproducing AlphaFolding

Given the limited available baseline models for protein trajectory generation, we attempted to reproduce AlphaFolding [7], a diffusion-based model that generate short trajectories (i.e., blocks) and can be extended to longer through iterative generation. We followed the authors' official implementation [2], setting the motion token count to 2 and the generation horizon (block length) to 16 frames. To improve generation efficiency, we increased the stride from the default 1 to 40, matching the setup in MDGEN. Training ALPHAFOLDING on the full ATLAS dataset resulted in out-of-memory error on NVIDIA A100-80GB GPU. Therefore, we adopted the authors' filtering criterion of a maximum sequence length of 256 residues. Similarly, we encountered the out-of-memory error for the five largest proteins during inference. The model was trained for 65K steps where we saw convergence. For evaluation, we iteratively extended the 16-frame outputs to generate 256 frames and retained the first 250 frames to match the 100 ns simulation setting.

We compared ALPHAFOLDING with MDGEN and CONFROVER on the ATLAS 100 ns simulation task, using one inference run per model. As shown in Table 15, both ALPHAFOLDING and CONFROVER outperform MDGEN in capturing the principal coordinates of the dominant dynamics, although ALPHAFOLDING slightly lags behind CONFROVER. However, as shown in Table 16, ALPHAFOLDING produces lower-quality conformations, exhibiting inflated backbone and side-chain rotamer outliers as well as higher clash rates. We found more evident degradations from error accumulation (Table 17), likely due to its iterative block-wise extension, which conditions only on the last frame of the preceding block. In contrast, MDGEN employs non-autoregressive attention across all frames, while CONFROVER maintains a full attention history via KV cache. Owing to the noisy nature of its generated trajectories, ALPHAFOLDING also shows inflated recall in conformational state recovery (Table 18), as the increased structural noise likely leads to artificially higher coverage in state space.

Although our implementation is a reproduction of [7] and may not faithfully reflect the authors' original model, this analysis provides valuable insights into the performance and potential limitations of block-extension-based trajectory models such as ALPHAFOLDING.

[2] https://github.com/fudan-generative-vision/dynamicPDB/tree/main/applications/4d_diffusion

Table 15: Pearson correlations of principal dynamic modes (PC1 and PC2) between sampled and reference trajectories, evaluated at varying lag times (Δt, in frames). The best scores are highlighted in **bold**.

		$\Delta t = 1$	$\Delta t = 5$	$\Delta t = 10$	$\Delta t = 20$
PC 1	MDGEN	0.11	0.12	0.10	0.12
PC 1	ALPHAFOLDING	0.15	0.15	0.16	0.18
PC 1	CONFROVER	**0.19**	**0.17**	**0.19**	**0.19**
PC 2	MDGEN	0.12	0.10	0.10	0.11
PC 2	ALPHAFOLDING	0.18	0.15	0.16	**0.20**
PC 2	CONFROVER	**0.19**	**0.17**	**0.18**	0.17

Table 16: Conformation geometric quality evaluated using MolProbity. Unrelaxed conformations are used for evaluation. The best scores are highlighted in **bold**.

	Ramachandran outliers % (↓)	Rotamer outliers % (↓)	Clash score (↓)	RMS bonds (↓)	RMS angles (↓)	MolProbity score (↓)
ALPHAFOLDING	2.91	15.37	151.35	0.07	4.76	3.94
MDGEN	1.87	**2.85**	128.08	**0.04**	**3.14**	3.28
CONFROVER	**1.75**	3.30	**76.89**	0.05	3.69	**3.00**

Table 17: Average Ramachandran outliers across trajectory frame ranges. Compared with MDGEN and CONFROVER, ALPHAFOLDING exhibits an increased level of backbone outliers when simulating longer trajectories.

Frame range	(0,15]	(15,31]	(31,47]	(47,63]	(63,79]
ALPHAFOLDING	1.77	2.78	3.21	3.34	3.57
MDGEN	1.59	1.85	1.96	1.95	2.05
CONFROVER	1.34	1.61	1.77	1.93	2.08

Table 18: Recovery of conformational states in the ATLAS 100 ns simulation experiment. Although ALPHAFOLDING shows the highest coverage, this may due to lower sample quality that artificially inflates diversity.

	JS-Dist ↓	Recall ↑	F1 ↑
ALPHAFOLDING	0.47	0.51	0.65
MDGEN	0.55	0.29	0.43
CONFROVER	0.51	0.42	0.58

E.7 Broader Impacts

This work aims to advance machine learning for protein modeling, with broad applications in biology and drug discovery. By enabling efficient simulation of protein dynamics and conformational changes, our method has the potential to accelerate research in structural biology and therapeutic development. Beyond proteins, the approach may also be adapted to other domains of computer-aided design, including small molecule design, materials science, and chip design. While the potential benefits are significant, it is important to ensure responsible use and prevent misuse of the technology. Developing appropriate safeguards and regulatory frameworks will be essential to mitigate any potential negative impacts.