LaM-SLidE: Latent Space Modeling of Spatial Dynamical Systems via Linked Entities

Florian Sestak[1] **Artur P. Toshev**[2] **Andreas Fürst**[1]
Günter Klambauer[*,1,4] **Andreas Mayr**[*,1] **Johannes Brandstetter**[*,1,3]
* Equal contribution
[1] ELLIS Unit, LIT AI Lab, Institute for Machine Learning, JKU Linz, Austria
[2] Department of Engineering Physics and Computation, TUM, Germany
[3] Emmi AI GmbH, Linz, Austria, [4] Clinical Research Institute for Medical AI, JKU Linz, Austria
{klambauer,mayr,brandstetter}@ml.jku.at

Abstract

Generative models are spearheading recent progress in deep learning, showcasing strong promise for trajectory sampling in dynamical systems as well. However, whereas latent space modeling paradigms have transformed image and video generation, similar approaches are more difficult for most dynamical systems. Such systems – from chemical molecule structures to collective human behavior – are described by interactions of entities, making them inherently linked to connectivity patterns, entity conservation, and the traceability of entities over time. Our approach, LAM-SLIDE (Latent Space Modeling of Spatial Dynamical Systems via Linked Entities), bridges the gap between: (1) keeping the traceability of individual entities in a latent system representation, and (2) leveraging the efficiency and scalability of recent advances in image and video generation, where pre-trained encoder and decoder enable generative modeling directly in latent space. The core idea of LAM-SLIDE is the introduction of identifier representations (IDs) that enable the retrieval of entity properties and entity composition from latent system representations, thus fostering traceability. Experimentally, across different domains, we show that LAM-SLIDE performs favorably in terms of speed, accuracy, and generalizability. Code is available at https://github.com/ml-jku/LaM-SLidE.

1 Introduction

Understanding dynamical systems represents a fundamental challenge across numerous scientific and engineering domains [47, 46, 73]. In this work, we address *spatial* dynamical systems, characterized by scenes of distinguishable entities at defined spatial coordinates. Modeling temporal trajectories of such entities quickly becomes challenging, especially when stochasticity is involved. A prime example is molecular dynamics [47], where trajectories of individual atoms are modeled via Langevin dynamics, which accounts for omitted degrees of freedom by using stochastic differential equations. Consequently, the trajectories of the atoms themselves become non-deterministic.

A conventional approach to predict spatial trajectories of entities is to represent scenes as neighborhood graphs and to subsequently process these graphs with graph neural networks (GNNs). When using GNNs [83, 64, 32, 11, 86], each entity is usually represented by a node, and the spatial entities nearby are connected by an edge in the neighborhood graph. Neighborhood graphs have extensively been used for trajectory prediction tasks [51], especially for problems with a large number of indistinguishable entities, [e.g., 81, 63]. Recently, GNNs have been integrated into generative modeling frameworks to effectively capture the behavior of stochastic systems [98, 22].

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

 https://doi.org/10.52202/085713-4684

Despite their widespread use in modeling spatial trajectories, GNNs hardly follow recent trends in latent space modeling, where unified representations [45] together with universality and scalability of transformer blocks [91] offer simple application across datasets and tasks, a behavior commonly observed in computer vision and language processing [24, 25]. Notably, recent breakthroughs in image and video generation can be attributed to latent space generative modeling [38, 15]. In such paradigms, pre-trained encoders and decoders are employed to map data into a latent space, where subsequent modeling is performed, leveraging the efficiency and expressiveness of this representation. This poses the question:

> Can we leverage recent techniques from **generative latent space modeling** to boost the **modeling of stochastic trajectories of dynamical systems** with a varying number of entities?

Recently, it has been shown [5] that it is possible to model the bulk behavior of large particle systems purely in the latent space, at the cost of sacrificing the traceability of individual particles, which is acceptable or even favorable for systems where particles are indistinguishable, but challenging for, e.g., molecular dynamics, where understanding the dynamics of individual atoms is essential.

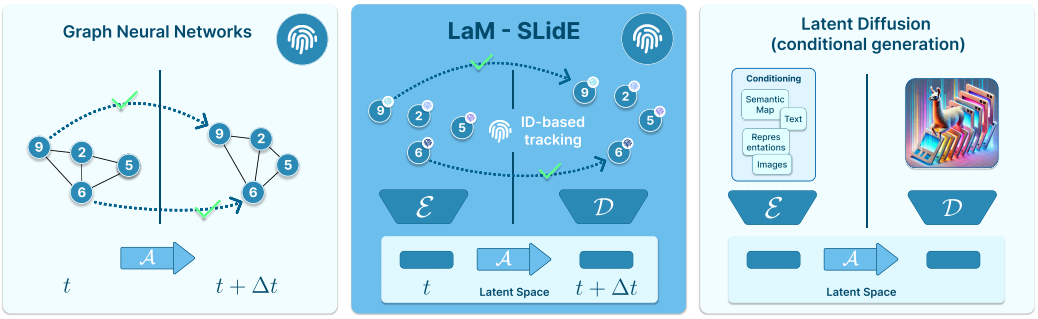

Figure 1: Overview of our approach. **Left:** Conventional graph neural networks (GNNs) model time-evolving systems (e.g., molecular dynamics) by representing entities as nodes and iteratively updating node embeddings and positions to capture system dynamics across timesteps. **Right:** Latent diffusion models employ an encoder-decoder architecture to compress input data into a lower-dimensional latent space where generative modeling is performed. Latent diffusion models, frequently enhanced with conditional information such as text, excel at generative tasks; however, due to their fixed input/output structure, they are not directly adaptable to physical systems with a varying number of entities. **Middle:** Our proposed approach LAM-SLIDE bridges these paradigms by: (1) introducing identifiers that allow traceability of individual entities, and (2) leveraging a latent system representation.

To leverage a latent system's representation, we need to be able to trace individual entities within the system. The core idea of LAM-SLIDE is the introduction of *identifiers* (IDs) that allow for the retrieval of entity properties, e.g., entity coordinates, from the latent system representation. Consequently, we can train generative models, like flow-matching [54, 57, 2], purely in latent space, where pre-trained decoder blocks map the generated representations back to the physics domain. An overview is given in Section 1. Qualitatively, LAM-SLIDE demonstrates flexibility and performs favorably across a variety of different modeling tasks.

Summarizing our contributions:

- **Latent system representation**: We propose LAM-SLIDE for generative modeling of stochastic trajectories, leveraging a latent system representation.
- **Entity structure preservation**: We introduce entity structure preservation to recover the encoded system states from the latent space representation via assignable identifiers.
- **Cross-domain generalization**: We perform experiments in different domains with varying degrees of difficulty, focusing on molecular dynamics, human motion behavior, and particle systems. LAM-SLIDE performs favorably with respect to all other architectures and showcases scalability with model size.

https://doi.org/10.52202/085713-4684

2 Background & Related Work

Dynamical systems. Formally, we consider a random dynamical system to be defined by a state space $\mathcal{S}$, representing all possible configurations of the system, and an evolution rule $\Phi : \mathbb{R} \times \mathcal{S} \mapsto \mathcal{S}$ that determines how a state $\mathbf{s} \in \mathcal{S}$ evolves over time, and which exhibits the following properties for the time differences $0, \hat{t}_1$, and, $\hat{t}_2$:

$$\Phi(0, \mathbf{s}) = \mathbf{s} \tag{1}$$

$$\Phi(\hat{t}_2, \Phi(\hat{t}_1, \mathbf{s})) = \Phi(\hat{t}_1 + \hat{t}_2, \mathbf{s}) \tag{2}$$

We note that Φ does not necessarily need to be defined on the whole space $\mathbb{R} \times \mathcal{S}$, but we assume this for notational simplicity. The exact formal definition of random dynamical systems is more involved and consists of a base flow (noise) and a cocycle dynamical system defined on a physical phase space [8]. We skip the details, but assume that we deal with random dynamical systems for the remainder of the paper. The non-deterministic behavior of such dynamical systems suggests the use of generative modeling approaches.

Generative modeling. Recent developments in generative modeling have captured widespread interest. The breakthroughs of the last years were mainly driven by diffusion models [87, 88, 37], a paradigm that transforms a simple distribution into a target data distribution via iterative refinement steps. Flow Matching [54, 57, 2] has emerged as a powerful alternative to diffusion models, enabling simulation-free training between arbitrary start and target distributions [55]. It has also been extended to data manifolds [19]. This approach comes with straighter paths, offering faster integration, and has been successfully applied across different domains like images [27], audio [92], videos [71], protein design [41], and robotics [13].

Latent space modeling. Latent space modeling has achieved remarkable success at image and video generation [15, 27], where pre-trained encoders and decoders map data into a latent space, and back into the physics space. The latent space aims to preserve the essential structure and features of the original data, often following a compositional structure $\mathcal{D} \circ \mathcal{A} \circ \mathcal{E}$ [85, 4, 5], where the encoder $\mathcal{E}$ maps the input signal into the latent space, the approximator $\mathcal{A}$ models a process, and the decoder maps back to the original space. Examples of approximators include conditional generative modeling techniques, such as generating an image given a text prompt (condition) [77]. This framework was recently used for 3D shape generation, where the final shape in the spatial domain is then constructed by querying the latent representations over a fixed spatial grid [101, 100].

3 LaM - SLidE

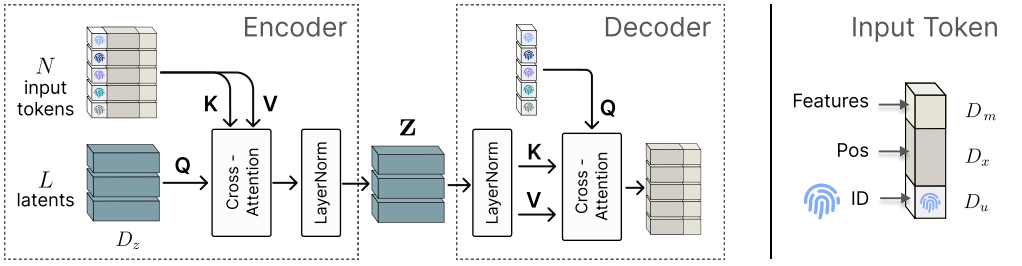

Figure 2: Architecture of our encoder-decoder structure (First Stage): **Left:** The encoder maps N input tokens to a latent system representation by cross-attending to L learned latent query tokens. The decoder reconstructs the input data from the latent representation using the assigned IDs. **Right:** Structure of the input token, consisting of an ID, spatial information, and features (see also Figure 3).

We introduce an *identifier pool* and an *identifier assignment function* which allows us to effectively map and retrieve entities to and from a latent system representation. The ID components preserve the relationships between entities, making them traceable across time-steps. LAM-SLIDE follows an encoder $\mathcal{E}$ - approximator $\mathcal{A}$ - decoder $\mathcal{D}$ paradigm.

https://doi.org/10.52202/085713-4684

3.1 Problem Formulation

State space. We consider spatial dynamics. Our states $\mathbf{s} \in \mathcal{S}$ describe the configuration of entities within the scene together with their individual features. We assume that a scene consists of N entities e_i with $i \in 1, \ldots, N$. An entity e_i is described by its spatial location $\mathbf{x}_i \in \mathbb{R}^{D_x}$ and some further properties $\mathbf{m}_i \in \mathbb{R}^{D_m}$ (e.g., atom type, etc.). We denote the set of entities as $E = \{e_1, \ldots, e_n\}$. We consider states $\mathbf{s}^t$ at discretized timepoints t. Analogously, we use $\mathbf{x}_i^t$, $\mathbf{m}_i^t$ to describe coordinates and properties at time t, respectively. We refer to the coordinate concatenation $[\mathbf{x}_1^t, .., \mathbf{x}_N^t]$ of the N entities in $\mathbf{s}^t$ as $\mathbf{X}^t \in \mathbb{R}^{N \times D_x}$. Analogously, we use $\mathbf{M}^t \in \mathbb{R}^{N \times D_m}$ to denote $[\mathbf{m}_1^t, .., \mathbf{m}_N^t]$. When properties are conserved over time, i.e., $\mathbf{M}^t = \mathbf{M}^1$, we just skip the time index and the time-wise repetition of states and use $\mathbf{M} \in \mathbb{R}^{N \times D_m}$. We concatenate sequences of coordinate states $\mathbf{X}^t$ with $t \in 1 \,..\, T$ to a tensor $\mathbf{X} \in \mathbb{R}^{T \times N \times D_x}$, which describes a whole sampled coordinate trajectory of a system with T time points and N entities. An example of such trajectories from dynamical systems are molecular dynamics trajectories (e.g. Figure 15). Notation is summarized in Table 6.

Prediction task. We predict a trajectory of entity coordinates $\mathbf{X}^{[T_o+1\,:\,T]} = [\mathbf{X}^{T_o+1}, \ldots, \mathbf{X}^t, \ldots, \mathbf{X}^T] \in \mathbb{R}^{(T-T_o) \times N \times D_x}$, given a short (observed) initial trajectory $\mathbf{X}^{[1\,:\,T_o]} = [\mathbf{X}^1, \ldots, \mathbf{X}^t, \ldots, \mathbf{X}^{T_o}] \in \mathbb{R}^{T_o \times N \times D_x}$ together with general (time-invariant) entity properties $\mathbf{M}$. Here, T_o denotes the length of the observed trajectory and $T_f = T - T_o$ denotes the prediction horizon.

3.2 Entity Structure Preservation

We aim to preserve the integrity of individual entities in a latent system representation. More specifically, we aim to preserve both the number of entities as well as their structure. For example, in the case of molecules, we want to preserve both the number of atoms and the atom composition. We therefore assign identifiers from an identifier pool to each entity, allowing us to trace the entities by their assigned identifiers. The two key components are: (i) creating a fixed, finite pool of *identifiers (IDs)* and (ii) defining a unique mapping between entities and identifiers.

Definition 3.1. For a fixed $u \in \mathbb{N}$, let $\mathcal{I} = \{0, 1, \ldots, u-1\}$ be the *identifier pool*. An *identifier* i is an element of the set $\mathcal{I}$.

Definition 3.2. Let E be a finite set of entities and $\mathcal{I}$ an identifier pool. The *identifier assignment pool* is the set of all injective functions from E to $\mathcal{I}$:

$$I = \{\texttt{ida}(\cdot) : E \mapsto \mathcal{I} \mid \forall e_i, e_j \in E : e_i \neq e_j \implies \texttt{ida}(e_i) \neq \texttt{ida}(e_j)\}, \tag{3}$$

An *identifier assignment function* $\texttt{ida}(\cdot)$ is an element of the set I.

Proposition 3.3. *Given an identifier pool $\mathcal{I}$ and a finite set of entities E, an identifier assignment pool I as defined by Definition 3.2 is non-empty if and only if $|E| \leqslant |\mathcal{I}|$.*

Proof, see App. F.1.

Proposition 3.4. *Given an identifier pool $\mathcal{I}$ and a finite set of entities E such that $|E| \leqslant |\mathcal{I}|$, the identifier assignment pool I as defined by Definition 3.2 contains finitely many injective functions.*

Proof, see App. F.2.

Notably, since $\texttt{ida}(\cdot)$ may be selected randomly – the specific choice of the mapping $\texttt{ida}(\cdot)$ can be arbitrary – the only requirement is that an injective mapping between entities and identifiers is established, i.e., each entity is uniquely assigned to an identifier, but not all identifiers need to be assigned to an entity. Further, Proposition 3.3 suggests using an identifier pool that is large enough, such that a model learned on this identifier pool can generalize across systems with varying numbers of entities.

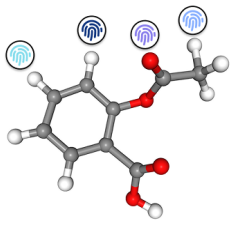

Figure 3: **Example aspirin**: IDs are assigned to the atoms of the molecule.

Example aspirin. Aspirin $C_9H_8O_4$ consists of 21 atoms, thus the identifier pool $\mathcal{I}$ needs to have at least 21 unique identifiers. We select an assignment function $\texttt{ida}(\cdot)$, arbitrary, and use it to assign each atom a unique identifier. Notably, e.g., for molecules, we do not explicitly model molecular bond information, as the spatial relationship between atoms (interatomic distances) implicitly captures this information. Figure 3 shows an arbitrary but fixed identifier assignment for aspirin; we illustrate different IDs by colored fingerprint symbols.

3.3 Model Architecture

Since predicting continuations of system trajectories is a conceptually similar task to generating videos from an initial sequence of images, we took inspiration from Blattmann et al. [15] in using a latent diffusion architecture. We also took inspiration from Jaegle et al. [43] to decompose our model architecture as follows: To map the state of the system composed of N entities to a latent space containing L learned latent tokens ($\in \mathbb{R}^{D_z}$), we use a cross-attention mechanism. In the resulting latent space, we aim to train an approximator to predict future latent states based on the embedded initial states. Inversely to the encoder, we again use a cross-attention mechanism to retrieve the physical information of the individual entities of the system from the latent system representation. To wrap it up, LAM-SLIDE , is built up by an encoder $\mathcal{E}$ - approximator $\mathcal{A}$ - decoder $\mathcal{D}$ architecture, $\mathcal{D} \circ \mathcal{A} \circ \mathcal{E}$. A detailed composition of $\mathcal{E}$ and $\mathcal{D}$ is shown in Figure 2.

Encoder. The encoder $\mathcal{E}$ aims to encode a state of the system such that the properties of each entity e_n can be decoded (retrieved) later. At the same time, the structure of the latent state representation $\mathbf{Z}^t \in \mathbb{R}^{L \times D_z}$ is constant and should not depend on the different number N of entities. This contrasts with GNNs, where the number of latent vectors depends on the number of nodes.

To allow for traceability of the entities, we first embed each identifier i in the space $\mathbb{R}^{D_u}$ by a learned embedding $\texttt{IDEmb} : \mathcal{I} \mapsto \mathbb{R}^{D_u}$. We map all ($n = 1, \ldots, N$) system entities e_n to $\mathbf{u}_n \in \mathbb{R}^{D_u}$ as follows:

$$\texttt{ida}(\cdot) \qquad \text{(arbitrary identifier assignment)} \tag{4}$$

$$\mathbf{u}_n = \texttt{IDEmb}(\texttt{ida}(e_n)) \qquad \forall n \in 1, \ldots, N \tag{5}$$

The inputs to the encoder comprise the time dependent location $\mathbf{x}_n^t \in \mathbb{R}^{D_x}$, properties $\mathbf{m}_n \in \mathbb{R}^{D_x}$, and identity representation $\mathbf{u}_n \in \mathbb{R}^{D_x}$ of each entity e_n, as visualized in the right part of Figure 2. We concatenate the different types of features across all entities of the system: $\mathbf{X}^t = [\mathbf{x}_1^t, .., \mathbf{x}_N^t]$, $\mathbf{M} = [\mathbf{m}_1, .., \mathbf{m}_N]$ and $\mathbf{U} = [\mathbf{u}_1, .., \mathbf{u}_N]$.

The encoder $\mathcal{E}$ maps the input to a latent system representation via

$$\mathcal{E} : [\mathbf{X}^t, \mathbf{M}, \mathbf{U}] \in \mathbb{R}^{N \times (D_x + D_m + D_u)} \mapsto \mathbf{Z}^t \in \mathbb{R}^{L \times D_z},$$

realized by cross-attention [91, 75] between the input tensor $\in \mathbb{R}^{N \times (D_x + D_m + D_u)}$, which serves as keys and values, and a fixed number of L learned latent vectors $\in \mathbb{R}^{D_z}$ [43], which serve as the queries. The encoding process is depicted on the left side of Figure 2.

Decoder. The aim of the decoder $\mathcal{D}$ is to retrieve the system state information $\mathbf{X}^t$ and $\mathbf{M}$ from the latent state representation $\mathbf{Z}^t$ using the encoded entity identifier embeddings $\mathbf{U}$. The decoder $\mathcal{D}$ maps the latent system representation back into the coordinates and properties of each entity via

$$\mathcal{D} : \mathbf{Z}^t \in \mathbb{R}^{L \times D_z} \times \mathbf{U} \in \mathbb{R}^{L \times D_u} \mapsto [\mathbf{X}^t, \mathbf{M}] \in \mathbb{R}^{N \times (D_x + D_m)}.$$

As shown in the middle part of Figure 2, $\mathcal{D}$ is realized by cross-attention layers. The latent space representation $\mathbf{Z}^t$ serves as the keys and values in the cross-attention mechanism, while the embedded identifier $\mathbf{u}_n$ acts as the query. Applied to to all ($n = 1, \ldots, N$) system entities e_n, this results in the retrieved system state information $\mathbf{X}^t$ and $\mathbf{M}$. Using the learned identifier embeddings as queries can be interpreted as a form of content-based retrieval and associative memory [6, 40, 75].

Approximator. Finally, the approximator models the system's time evolution in latent space, i.e., it predicts a series of future latent system states $\mathbf{Z}^{[T_o+1:\,T]} = [\mathbf{Z}^{T_o+1}, \ldots, \mathbf{Z}^t, \ldots, \mathbf{Z}^T]$, given a series of initial latent system states $\mathbf{Z}^{[1:\,T_o]} = [\mathbf{Z}^1, \ldots, \mathbf{Z}^t, \ldots, \mathbf{Z}^{T_o}]$,

$$\mathcal{A} : \mathbf{Z}^{[1:\,T_o]} \in \mathbb{R}^{T_o \times L \times D_z} \mapsto \mathbf{Z}^{[T_o+1:\,T]} \in \mathbb{R}^{T_f \times L \times D_z}.$$

Given the analogy of predicting the time evolution of a dynamical system to the task of synthesizing videos, we realized $\mathcal{A}$ by a flow-based model. Specifically, we constructed it based on the stochastic interpolants framework [2, 60].

We are interested in time-dependent processes, which interpolate between data $\mathbf{o}_1 \sim p_1$ from a target data distribution p_1 and noise $\epsilon \sim p_0 \coloneqq \mathcal{N}(\mathbf{0}, \boldsymbol{I})$:

$$\mathbf{o}_\tau = \alpha_\tau \mathbf{o}_1 + \sigma_\tau \epsilon, \tag{6}$$

 https://doi.org/10.52202/085713-4684

where $\tau \in [0,1]$ is the time parameter of the flow (to be distinguished from system times t). α_τ and σ_τ are differentiable functions in τ, which have to fulfill $\alpha_\tau^2 + \sigma_\tau^2 > 0 \ \forall \tau \in [0,1]$, and, further $\alpha_0 = \sigma_1 = 0$, and, $\alpha_1 = \sigma_0 = 1$. The goal is to learn a parametric model $\boldsymbol{v}_\theta(\mathbf{o}, \tau)$, s.t., $\int_0^1 \mathbb{E}[||\boldsymbol{v}_\theta(\mathbf{o}_\tau, \tau) - \dot{\alpha}_\tau \mathbf{o}_1 - \dot{\sigma}_\tau \epsilon||^2] \, d\tau$ is minimized. Within the stochastic interpolants framework, we identify $\mathbf{o}_1$ with a whole trajectory $\mathbf{Z} = \mathbf{Z}^{[1:T]} = [\mathbf{Z}^{[1:T_o]}, \mathbf{Z}^{[T_o+1:T]}] \in \mathbb{R}^{T \times L \times D_z}$.

Since the generated trajectories should be conditioned on the latent system representations of initial time frames $\mathbf{Z}^{[1:T_o]}$, we extend $\boldsymbol{v}_\theta$ with a conditioning argument $\mathbf{C} \in \mathbb{R}^{T \times L \times D_z}$, making it effectively a conditional vector field $\boldsymbol{v}_\theta : \mathbb{R}^{T \times L \times D_z} \times [0,1] \times \mathbb{R}^{T \times L \times D_z} \mapsto \mathbb{R}^{T \times L \times D_z}$. The tensor structure of $\mathbf{C}$ is the same as the one for $\mathbf{Z}$. For the first time steps, both tensors have equal values, i.e., $\mathbf{C}^{[1:T_o]} = \mathbf{Z}^{[1:T_o]}$. The remaining tensor entries $\mathbf{C}^{[T_o+1:T]}$ are filled up with mask tokens, see Figure 4. The latent model is structured as series transformer blocks [90, 67], alternating between the spatial and temporal dimensions (see App. B.4). We parametrized our model using a data prediction objective [55], see App. C.1. For architectural details see App. B.

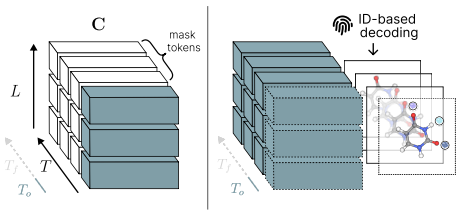

Figure 4: **Left:** The latent model receives conditioning via known tokens (observed timesteps) and mask tokens (for prediction). This example shows conditioning on one timeframe to predict three future ones. **Right:** ID-based decoding, where the predicted atom positions are decoded by the assigned IDs.

3.4 Training Procedure

The training process follows a two-stage approach similar to latent diffusion models [77]. **First Stage:** We train the encoder $\mathcal{E}$ and decoder $\mathcal{D}$, to reconstruct entities from latent space using assigned IDs, see Figure 2). **Second Stage:** We train the approximator $\mathcal{A}$ on the latent system representations produced by the frozen encoder $\mathcal{E}$ (details in App. E.3 and Figure 5).

4 Experiments

In order to evaluate LAM-SLIDE we focus on three key aspects: (i) **Robust generalization in diverse domains**. We examine LAM-SLIDE 's generalization in different data domains in relation to other methods, for which we utilize tracking data from human motion behavior, trajectories of particle systems, and data from *molecular dynamics* (MD) simulations. (ii) **Temporal adaptability**. We evaluate temporal adaptability through various conditioning/prediction horizons, considering single/multi-frame conditioning and short/long-term predictions; (iii) **Computational efficiency and scalability**. Finally, we assess the inference time of LAM-SLIDE , and evaluate performance with respect to model size. The subsequent sections show our key findings. For implementation details and additional results see App. E and G, for information on the datasets see App. E.1.

Metrics. We utilized the Average Discrepancy Error (ADE) and the Final Discrepancy Error (FDE), defined as $\text{ADE}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{(T-T_o)N} \sum_{t=T_o+1}^{T} \sum_{i=1}^{N} \|\mathbf{X}_i^t - \hat{\mathbf{X}}_i^t\|_2$, $\text{FDE}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{X}_i^T - \hat{\mathbf{X}}_i^T\|_2$, capturing model performance across predicted future time steps and the model performance specifically for the last predicted frame, respectively. These metrics represent well-established evaluation criteria in trajectory forecasting [94, 95].

For the MD experiments on peptides (Tetrapeptides), we use the Jensen-Shannon divergence (JSD) over the distribution of torsion angles, considering both backbone (BB) and side chain (SC) angles. To capture long temporal behavior, we use *Time-lagged Independent Component Analysis* (TICA) [70], focusing on the slowest components TIC 0 and TIC 1. To investigate metastable state transitions, we make use of *Markov State Models* (MSMs) [74, 65].

For inference time and scalability, we assess the *number of function evaluations* (NFE) and report the performance of our method for different model sizes. A comprehensive overview of the conditioning and prediction horizons for each experiment is shown in Table 7.

Table 1: **Results on generation on N-body dataset**, in terms of ADF/FDE averaged over 5 runs.

	Particle		Spring		Gravity	
	ADE	FDE	ADE	FDE	ADE	FDE
RF [52][a]	0.479	1.050	0.0145	0.0389	0.791	1.630
TFN [89][a]	0.330	0.754	0.1013	0.2364	0.327	0.761
SE(3)-Tr [30][a]	0.395	0.936	0.0865	0.2043	0.338	0.830
EGNN [82][a]	0.186	0.426	0.0101	0.0231	0.310	0.709
EqMotion [94][a]	0.141	0.310	0.0134	0.0358	0.302	0.671
SVAE [95][a]	0.378	0.732	0.0120	0.0209	0.582	1.101
GeoTDM [35][a]	0.110	0.258	**0.0030**	**0.0079**	0.256	0.613
LAM-SLIDE	**0.104**	**0.238**	0.0070	0.0135	**0.157**	**0.406**

[a] Results from Han et al. [35].

Table 2: **Results on the ETH-UCY** dataset for pedestrian forecasting. Results are in terms of minADE/minFDE out of 20 runs.

	ETH	Hotel	Univ	Zara1	Zara2	Average
Linear[a]	1.07/2.28	0.31/0.61	0.52/1.16	0.42/0.95	0.32/0.72	0.53/1.14
SGAN [34][a]	0.64/1.09	0.46/0.98	0.56/1.18	0.33/0.67	0.31/0.64	0.46/0.91
SoPhie [78][a]	0.70/1.43	0.76/1.67	0.54/1.24	0.30/0.63	0.38/0.78	0.54/1.15
PECNet [61][a]	0.54/0.87	0.18/0.24	0.35/0.60	0.22/0.39	0.17/0.30	0.29/0.48
Traj++ [80][a]	0.54/0.94	0.16/0.28	0.28/0.55	0.21/0.42	0.16/0.32	0.27/0.50
BiTraP [97][a]	0.56/0.98	0.17/0.28	0.25/0.47	0.23/0.45	0.16/0.33	0.27/0.50
MID [33][a]	0.50/0.76	0.16/0.24	0.28/0.49	0.25/0.41	0.19/0.35	0.27/0.45
SVAE [95][a]	0.47/0.76	0.14/0.22	0.25/0.47	**0.20**/0.37	**0.14/0.28**	**0.24**/0.42
GeoTDM [35][a]	0.46/**0.64**	**0.13**/0.21	**0.24/0.45**	0.21/0.39	0.16/0.30	**0.24/0.40**
LAM-SLIDE	**0.45**/0.75	**0.13/0.19**	0.26/0.47	0.21/**0.35**	0.17 / 0.30	**0.24**/ 0.41

[a] Results from Han et al. [35].

4.1 Pedestrian Trajectory Forecasting (ETH-UCY)

Experimental setup. For human motion behavior, we first consider the ETH-UCY dataset [68, 53], which provides pedestrian movement behavior, over five different scenes: ETH, Hotel, Univ, Zara1, and Zara2. We use the same setup as Han et al. [35], Xu et al. [94, 95], in which the methods obtain the first 8 frames as conditioning information and have to predict the next 12 frames. We report the minADE/minFDE, computed across 20 sampled trajectories.

Compared methods. We compare LAM-SLIDE to eight state-of-the-art generative methods covering different model categories, including: GANs : SGAN [34], SoPhie [78]; VAEs: PECNet [61], Traj+ + [80], BiTrap [97], SVAE [95]; diffusion models: MID [33] and GeoTDM [35] and a linear baseline. The baseline methods predominantly target pedestrian trajectory prediction, with GeoTDM and Linear being the exceptions.

Results. As shown in Table 2, our model performs competitively across all five scenes, achieving lower minFDE for Zara1 and Hotel scene, and the lowest minADE on the ETH scene. Notably, in contrast to the compared baselines, we did not create additional features like velocity and acceleration or imply any kind of connectivity between entities.

4.2 Basketball Player Trajectory Forecasting (NBA)

Experimental setup. Our second experiment examines human motion behavior in the context of basketball gameplay. We utilize the SportVU NBA movement dataset [99], which contains player movement data from the 2015-2016 NBA season. Each recorded frame includes ten player positions (5 for each team) and the ball position. We consider two different scenarios: a) *Rebounding:* containing scenes of missed shots; b) *Scoring:* containing scenes of a team scoring a basket. The interaction patterns – both in frequency and in their adversarial versus cooperative dynamics – exhibit different challenges as those in Section 4.1. The evaluation procedure by Xu et al. [95], which we use, provides 8 frames as input conditioning and the consecutive 12 frames for prediction. The performance is reported by the minADE/minFDE metrics, which are computed across 20 sampled trajectories. For the reported metrics, only the trajectories of the players are considered.

Table 3: **Results on the NBA dataset:** Compared methods have to predict player positions for 12 frames and are given the initial 8 frames as input. Results in terms of minADE/minFDE for the Rebounding and Scoring scenarios.

	Rebounding	Scoring
Linear[a]	2.14/5.09	2.07/4.81
Traj++ [80][a]	0.98/1.93	0.73/1.46
BiTraP [97][a]	0.83/1.72	0.74/1.49
SGNet-ED [93][a]	0.78/1.55	0.68/1.30
SVAE [95][a]	**0.72/1.37**	**0.64**/1.17
LAM-SLIDE	0.79/1.42	**0.64/1.09**

[a] Results from Xu et al. [95].

Compared methods. The method comparison for this human motion forecasting task includes methods based on VAEs, Traj++ [80], BiTrap [97], SGNet-ED [93], SVAE [95], and a linear baseline.

Results. As illustrated in Table 3, our model shows robust performance across both scenarios, Rebounding and Scoring. For the Scoring scenario, LAM-SLIDE achieves parity with SocialVAE [34]

https://doi.org/10.52202/085713-4684

in terms of minADE and surpasses the performance in terms of minFDE. In the Rebounding scenario, we observe comparable but slightly lower performance of LAM-SLIDE compared to SocialVAE. Trajectories sampled by our model are shown in Figure 13.

4.3 N-Body System Dynamics (Particle Systems)

Experimental setup. We evaluate our method across three distinct N-Body simulation scenarios: a) *Charged Particles*: comprising particles with randomly assigned charges $+1/-1$ interacting via Coulomb forces [51, 82]; b) *Spring Dynamics*: consisting of $N = 5$ particles with randomized masses connected by springs with a probability 0.5 between particle pairs [51]; and c) *Gravitational Systems*: containing $N = 10$ particles with randomized masses and initial velocities governed by gravitational interactions [17]. For all three scenarios, we consider 10 conditioning frames and 20 prediction frames. In line with Han et al. [35], we use 3000 trajectories for training and 2000 trajectories for validation and testing, and we report ADE/FDE averaged over $K = 5$ runs.

Compared methods. We compare LAM-SLIDE to seven different methods, including six equivariant GNN based methods: Tensor Field Network [89], Radial Field [52], SE(3)-Transformer [30], EGNN [82], EqMotion [94], GeoTDM [35], and a non-equivariant method: SVAE [95].

Results. LAM-SLIDE achieves the best performance in terms of ADE/FDE for the Charged Particles and Gravity scenarios and competitive second-rank performance in the Spring Dynamics scenario, see Table 1. Unlike compared methods, LAM-SLIDE achieves these results without computing intermediate physical quantities such as velocities or accelerations. We present sampled trajectories in Figure 14.

4.4 Molecular Dynamics - Small Molecules (MD17)

Experimental setup. In this experiment, we evaluate LAM-SLIDE on the well-established MD17 [21] dataset, containing simulated molecular dynamics trajectories of 8 small molecules. The size of those molecules ranges from 9 atoms (Ethanol and Malonaldehyde) to 21 atoms (Aspirin). We use 10 frames as conditioning, 20 frames for prediction, and report ADE/FDE averaged over $K = 5$ runs.

Compared methods. Similar to Section 4.3, we compared LAM-SLIDE to: Tensor Field Network [89], Radial Field [52], SE(3)-Transformer [30], EGNN [82], EqMotion [94], GeoTDM [35], and SVAE [95].

Results. The results in Table 4 show the performance on the MD17 benchmark. LAM-SLIDE achieves the lowest ADE/FDE of all methods and for all molecules. These results are particularly remarkable considering that: (1) our model operates without an explicit definition of molecular bond information, and (2) it surpasses the performance of all equivariant baselines, an inductive bias we intentionally omitted in LAM-SLIDE .

Notably, we train a single model on all molecules – a feat that is structurally encouraged by the design of LAM-SLIDE . For ablation, we also train GeoTDM [35] on all molecules and evaluate the performance on each one of them ("all→each" in the Table 13). Interestingly, we also observe consistent improvements in the GeoTDM performance. However, GeoTDM's performance does not reach that of LAM-SLIDE . We also note that our latent model is trained for 2000 epochs, while GeoTDM was trained for 5000 epochs. Trajectories are shown in Figure 15.

4.5 Molecular Dynamics - Tetrapeptides (4AA)

Experimental setup. To investigate LAM-SLIDE on long prediction horizons, we utilize the Tetrapeptide dataset from Jing et al. [44], containing explicit-solvent molecular dynamics trajectories simulated using OpenMM [26]. The dataset comprises 3,109 training, 100 validation, and 100 test peptides. We use a single conditioning frame to predict 10,000 consecutive frames. The predictions are structured as a sequence of ten cascading 1,000-step rollouts, where each subsequent rollout is conditioned on the final frame of the previous. Note that, in contrast to the MD17 dataset, the methods predict trajectories of *unseen* molecules.

Table 4: **Results on the MD17 dataset**. Compared methods have to predict the atom positions of 20 frames, conditioned on 10 input frames. Results in terms of ADE/FDE, averaged over 5 runs.

	Aspirin		Benzene		Ethanol		Malonaldehyde		Naphthalene		Salicylic		Toluene		Uracil	
	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
RF [52][a]	0.303	0.442	0.120	0.194	0.374	0.515	0.297	0.454	0.168	0.185	0.261	0.343	0.199	0.249	0.239	0.272
TFN [89][a]	0.133	0.268	0.024	0.049	0.201	0.414	0.184	0.386	0.072	0.098	0.115	0.223	0.090	0.150	0.090	0.159
SE(3)-Tr. [30][a]	0.294	0.556	0.027	0.056	0.188	0.359	0.214	0.456	0.069	0.103	0.189	0.312	0.108	0.184	0.107	0.196
EGNN [82][a]	0.267	0.564	0.024	0.042	0.268	0.401	0.393	0.958	0.095	0.133	0.159	0.348	0.207	0.294	0.154	0.282
EqMotion [94][a]	0.185	0.246	0.029	0.043	0.152	0.247	0.155	0.249	0.073	0.092	0.110	0.151	0.097	0.129	0.088	0.116
SVAE [95][a]	0.301	0.428	0.114	0.133	0.387	0.505	0.287	0.430	0.124	0.135	0.122	0.142	0.145	0.171	0.145	0.156
GeoTDM [35] [a]	0.107	0.193	0.023	0.039	0.115	0.209	0.107	0.176	0.064	0.087	0.083	0.120	0.083	0.121	0.074	0.099
LAM-SLIDE	**0.059**	**0.098**	**0.021**	**0.032**	**0.087**	**0.167**	**0.073**	**0.124**	**0.037**	**0.058**	**0.047**	**0.074**	**0.045**	**0.075**	**0.050**	**0.074**

[a] Results from Han et al. [35].

Compared methods. We compare LAM-SLIDE to the recently proposed method MDGen [44], which is geared towards protein MD simulations, and to a replicate of the ground truth MD simulation as a baseline, which is a lower bound for the performance.

Results. Table 5 shows performance metrics of the methods (for details on those metrics see App. E.6). Figure 16 shows the distribution of backbone torsion angles, and the free energy surfaces of the first two TICA components, for ground truth vs simulated trajectories. LAM-SLIDE performs competitively with the current state-of-the-art method MDGen with respect to torsion angles, which is a notable achievement given that MDGen operates in torsion space only. With respect to the TICA and MSM metrics, LAM-SLIDE even outperforms MDGen. Sampled trajectories are shown in Figure 17.

4.6 Computational Efficiency and Scaling Behavior

To assess computational efficiency, we compare the NFEs of our model to the second-best method, GeoTDM. Our results show that LAM-SLIDE requires up to 10x-100x fewer function evaluations depending on the domain. For a detailed discussion, see App. G.2.

We further assess the scalability of LAM-SLIDE across different model sizes. Our results indicate that performance consistently improves with increasing parameter count, suggesting that our method benefits from larger model capacity and could potentially achieve even better results with additional computational resources. Comprehensive details of this analysis can be found in App. G.

Table 5: **Results on the Tetrapeptide dataset:** Columns denote the JSD between distributions of *torsion angles* (backbone (BB), side-chain (SC), and all angles), the TICA, and the MSM metric.

	Torsions			TICA		MSM	Time
	BB	SC	All	0	0,1 joint		
100ns [a]	.103	.055	.076	.201	.268	.208	$\sim$ 3h
MDGen[44][a]	.130	**.093**	**.109**	.230	.316	.235	$\sim$ 60s
LAM-SLIDE	**.128**	.122	.125	**.227**	**.315**	**.224**	$\sim$ 53s

[a] Results from Jing et al. [44].

4.7 Ablations

Identifier Pool Size. To understand how the size of the identifier pool affects the performance of the first stage model, we conducted additional experiments on the MD17 dataset. Maintaining a fixed number of update steps across different configurations results in a modest increase in the reconstruction error for larger pool sizes. We attribute this to the reduced number of updates per individual identifier embedding. A comprehensive analysis is provided in App. G.4.

Identifier Assignment. We assessed the sensitivity of our model to the specific entity-identifier assignment by evaluating five random assignments on the MD17 dataset. Results do not indicate a negative impact on the performance of our model. This demonstrates the robustness of our approach with respect to the specific assignment between entities and identifiers, see App. G.5 for more details.

Identifier Latent Utilization. We analyzed decoder attention patterns to understand how identifiers access latent information across different compression ratios. Our findings show that each identifier maintains a consistent addressing scheme across different molecule conformations, and the model adaptively utilizes attention heads based on the capacity of the latent space. In an overparameterized

 https://doi.org/10.52202/085713-4684

setting, the model favors single-head retrieval, while in the compressed setting utilizes both heads to avoid potential identifier collisions. See Appendix G.6 for detailed analysis.

5 Discussion

Limitations. Our experiments indicate that our architecture applies to a diverse set of problems; however, a few limitations provide opportunities for future improvement. While our current approach successfully allows for compressing entities with beneficial reconstruction performance, our experiments indicate a tradeoff between the number of latent space vectors to encode system states and the performance of our latent model; for more details, see App. G.3.

Conclusion. We have introduced LAM-SLIDE, a novel approach for modeling spatial dynamical systems that consist of a variable number of entities within a fixed-size latent system representation, where assignable identifiers enable the traceability of individual entities. Across diverse domains, LAM-SLIDE matches or exceeds specialized methods and offers promising scalability properties. Its minimal reliance on prior knowledge makes it suitable for many tasks, suggesting its potential as a foundational architecture for dynamical systems.

Acknowledgments

We thank Niklas Schmidinger, Anna Zimmel, Benedikt Alkin, and Arturs Berzins for helpful discussions and feedback. The ELLIS Unit Linz, the LIT AI Lab, and the Institute for Machine Learning are supported by the Federal State Upper Austria. We thank the projects FWF AIRI FG 9-N (10.55776/FG9), AI4GreenHeatingGrids (FFG- 899943), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01), FWF Bilateral Artificial Intelligence (10.55776/COE12). We thank NXAI GmbH, Audi AG, Silicon Austria Labs (SAL), Merck Healthcare KGaA, GLS (Univ. Waterloo), TÜV Holding GmbH, Software Competence Center Hagenberg GmbH, dSPACE GmbH, TRUMPF SE + Co. KG.

References

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

[2] Albergo, M., Boffi, N., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions, 2023. *ArXiv preprint ArXiv230308797*, 2023.

[3] Albergo, M. S. and Vanden-Eijnden, E. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

[4] Alkin, B., Fürst, A., Schmid, S. L., Gruber, L., Holzleitner, M., and Brandstetter, J. Universal physics transformers: A framework for efficiently scaling neural operators. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[5] Alkin, B., Kronlachner, T., Papa, S., Pirker, S., Lichtenegger, T., and Brandstetter, J. Neuraldem-real-time simulation of industrial particulate flows. *arXiv preprint arXiv:2411.09678*, 2024.

[6] Amari, S.-I. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on computers*, 100(11):1197–1206, 1972.

[7] Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Suo, M., Tillet, P., Wang, E., Wang, X., Wen, W., Zhang, S., Zhao, X., Zhou, K., Zou, R., Mathews, A., Chanan, G., Wu, P., and Chintala, S. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference*

on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24). ACM, April 2024. doi: 10.1145/3620665.3640366. URL https://docs.pytorch.org/assets/pytorch2-2.pdf.

[8] Arnold, L. *Random Dynamical Systems*. Monographs in Mathematics. Springer, 1998. ISBN 9783540637585.

[9] Arriola, M., Gokaslan, A., Chiu, J. T., Yang, Z., Qi, Z., Han, J., Sahoo, S. S., and Kuleshov, V. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

[10] Ba, J. L. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[11] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[12] Biewald, L. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

[13] Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[14] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2023.

[15] Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S. W., Fidler, S., and Kreis, K. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22563–22575, 2023.

[16] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[17] Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J., and Welling, M. Geometric and physical quantities improve e (3) equivariant message passing. *arXiv preprint arXiv:2110.02905*, 2021.

[18] Case, D., Aktulga, H., Belfon, K., Ben-Shalom, I., Berryman, J., Brozell, S., Cerutti, D., Cheatham, T., III, Cisneros, G., Cruzeiro, V., Darden, T., Forouzesh, N., Ghazimirsaeed, M., Giambaşu, G., Giese, T., Gilson, M., Gohlke, H., Goetz, A., Harris, J., Huang, Z., Izadi, S., Izmailov, S., Kasavajhala, K., Kaymak, M., Kovalenko, A., Kurtzman, T., Lee, T., Li, P., Li, Z., Lin, C., Liu, J., Luchko, T., Luo, R., Machado, M., Manathunga, M., Merz, K., Miao, Y., Mikhailovskii, O., Monard, G., Nguyen, H., O'Hearn, K., Onufriev, A., Pan, F., Pantano, S., Rahnamoun, A., Roe, D., Roitberg, A., Sagui, C., Schott-Verdugo, S., Shajan, A., Shen, J., Simmerling, C., Skrynnikov, N., Smith, J., Swails, J., Walker, R., Wang, J., Wang, J., Wu, X., Wu, Y., Xiong, Y., Xue, Y., York, D., Zhao, C., Zhu, Q., and Kollman, P. Amber 2024, 2024.

[19] Chen, R. T. and Lipman, Y. Riemannian flow matching on general geometries. *arXiv e-prints*, pp. arXiv–2302, 2023.

[20] Chen, R. T. Q. torchdiffeq, 2018. URL https://github.com/rtqichen/torchdiffeq.

[21] Chmiela, S., Tkatchenko, A., Sauceda, H. E., Poltavsky, I., Schütt, K. T., and Müller, K.-R. Machine learning of accurate energy-conserving molecular force fields. *Science advances*, 3 (5):e1603015, 2017.

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

[23] Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A. P., Caron, M., Geirhos, R., Alabdulmohsin, I., et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pp. 7480–7512. PMLR, 2023.

 https://doi.org/10.52202/085713-4684

[24] Devlin, J. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[25] Dosovitskiy, A. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[26] Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., Simmonett, A. C., Harrigan, M. P., Stern, C. D., et al. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS computational biology*, 13(7): e1005659, 2017.

[27] Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al. Scaling rectified flow transformers for high-resolution image synthesis, 2024. *URL https://arxiv. org/abs/2403.03206*, 2, 2024.

[28] Falcon, W. and The PyTorch Lightning team. PyTorch Lightning, March 2019. URL `https://github.com/Lightning-AI/lightning`.

[29] Ford, G., Kac, M., and Mazur, P. Statistical mechanics of assemblies of coupled oscillators. *Journal of Mathematical Physics*, 6(4):504–515, 1965.

[30] Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020.

[31] Gardner Jr, E. S. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1): 1–28, 1985.

[32] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

[33] Gu, T., Chen, G., Li, J., Lin, C., Rao, Y., Zhou, J., and Lu, J. Stochastic trajectory prediction via motion indeterminacy diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17113–17122, 2022.

[34] Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2255–2264, 2018.

[35] Han, J., Xu, M., Lou, A., Ye, H., and Ermon, S. Geometric trajectory diffusion models. *arXiv preprint arXiv:2410.13027*, 2024.

[36] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[37] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[38] Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[39] Hoel, H. and Szepessy, A. Classical langevin dynamics derived from quantum mechanics. *arXiv preprint arXiv:1906.09858*, 2019.

[40] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[41] Huguet, G., Vuckovic, J., Fatras, K., Thibodeau-Laufer, E., Lemos, P., Islam, R., Liu, C.-H., Rector-Brooks, J., Akhound-Sadegh, T., Bronstein, M., et al. Sequence-augmented se (3)-flow matching for conditional protein backbone generation. *arXiv preprint arXiv:2405.20313*, 2024.

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

[43] Jaegle, A., Borgeaud, S., Alayrac, J.-B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[44] Jing, B., Stärk, H., Jaakkola, T., and Berger, B. Generative modeling of molecular dynamics trajectories. *arXiv preprint arXiv:2409.17808*, 2024.

[45] Joshi, C. K., Fu, X., Liao, Y.-L., Gharakhanyan, V., Miller, B. K., Sriram, A., and Ulissi, Z. W. All-atom diffusion transformers: Unified generative modelling of molecules and materials. *arXiv preprint arXiv:2503.03965*, 2025.

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

[47] Karplus, M. and Petsko, G. A. Molecular dynamics simulations in biology. *Nature*, 347(6294): 631–639, 1990.

[48] Kingma, D. and Gao, R. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36, 2024.

[49] Kingma, D. P. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[50] Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[51] Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *International conference on machine learning*, pp. 2688–2697. PMLR, 2018.

[52] Köhler, J., Klein, L., and Noé, F. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019.

[53] Lerner, A., Chrysanthou, Y., and Lischinski, D. Crowds by example. In *Computer graphics forum*, volume 26, pp. 655–664. Wiley Online Library, 2007.

[54] Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[55] Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R. T., Lopez-Paz, D., Ben-Hamu, H., and Gat, I. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

[56] Liu, J., Yang, C., Lu, Z., Chen, J., Li, Y., Zhang, M., Bai, T., Fang, Y., Sun, L., Yu, P. S., et al. Towards graph foundation models: A survey and beyond. *arXiv preprint arXiv:2310.11829*, 2023.

[57] Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[58] Loshchilov, I., Hutter, F., et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5, 2017.

[59] Lou, A., Meng, C., and Ermon, S. Discrete diffusion modeling by estimating the ratios of the data distribution. *URL https://arxiv. org/abs/2310.16834*, 2024.

[60] Ma, N., Goldstein, M., Albergo, M. S., Boffi, N. M., Vanden-Eijnden, E., and Xie, S. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024.

[61] Mangalam, K., Girase, H., Agarwal, S., Lee, K.-H., Adeli, E., Malik, J., and Gaidon, A. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 759–776. Springer, 2020.

 https://doi.org/10.52202/085713-4684

[62] Mao, H., Chen, Z., Tang, W., Zhao, J., Ma, Y., Zhao, T., Shah, N., Galkin, M., and Tang, J. Position: Graph foundation models are already here. In *Forty-first International Conference on Machine Learning*, 2024.

[63] Mayr, A., Lehner, S., Mayrhofer, A., Kloss, C., Hochreiter, S., and Brandstetter, J. Boundary graph neural networks for 3d simulations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):9099–9107, Jun. 2023. doi: 10.1609/aaai.v37i8.26092.

[64] Micheli, A. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009. doi: 10.1109/TNN.2008.2010350.

[65] Noé, F., Wu, H., Prinz, J.-H., and Plattner, N. Projected and hidden markov models for calculating kinetics and metastable states of complex molecules. *The Journal of chemical physics*, 139, 2013.

[66] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[67] Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

[68] Pellegrini, S., Ess, A., Schindler, K., and Van Gool, L. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th international conference on computer vision*, pp. 261–268. IEEE, 2009.

[69] Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[70] Pérez-Hernández, G., Paul, F., Giorgino, T., De Fabritiis, G., and Noé, F. Identification of slow molecular order parameters for markov model construction. *The Journal of chemical physics*, 139(1), 2013.

[71] Polyak, A., Zohar, A., Brown, A., Tjandra, A., Sinha, A., Lee, A., Vyas, A., Shi, B., Ma, C.-Y., Chuang, C.-Y., et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

[72] Ponder, J. W. and Case, D. A. Force fields for protein simulations. *Advances in protein chemistry*, 66:27–85, 2003. ISSN 0065-3233. doi: 10.1016/S0065-3233(03)66002-X.

[73] Price, I., Sanchez-Gonzalez, A., Alet, F., Andersson, T. R., El-Kadi, A., Masters, D., Ewalds, T., Stott, J., Mohamed, S., Battaglia, P., et al. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025.

[74] Prinz, J.-H., Wu, H., Sarich, M., Keller, B., Senne, M., Held, M., Chodera, J. D., Schütte, C., and Noé, F. Markov models of molecular kinetics: Generation and validation. *The Journal of chemical physics*, 134(17), 2011.

[75] Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Adler, T., Kreil, D., Kopp, M. K., G, K., and Hochreiter, S. Hopfield networks is all you need. *International Conference on Learning Representations*, 2021.

[76] Rogozhnikov, A. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2021.

[77] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

[78] Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., and Savarese, S. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1349–1358, 2019.

[79] Sahoo, S., Arriola, M., Schiff, Y., Gokaslan, A., Marroquin, E., Chiu, J., Rush, A., and Kuleshov, V. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

[80] Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pp. 683–700. Springer, 2020.

[81] Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pp. 8459–8468. PMLR, 2020.

[82] Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9323–9332. PMLR, 18–24 Jul 2021.

[83] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

[84] Scherer, M. K., Trendelkamp-Schroer, B., Paul, F., Pérez-Hernández, G., Hoffmann, M., Plattner, N., Wehmeyer, C., Prinz, J.-H., and Noé, F. Pyemma 2: A software package for estimation, validation, and analysis of markov models. *Journal of chemical theory and computation*, 11(11):5525–5542, 2015.

[85] Seidman, J., Kissas, G., Perdikaris, P., and Pappas, G. J. Nomad: Nonlinear manifold decoders for operator learning. *Advances in Neural Information Processing Systems*, 35:5601–5613, 2022.

[86] Sestak, F., Schneckenreiter, L., Brandstetter, J., Hochreiter, S., Mayr, A., and Klambauer, G. Vn-egnn: E (3)-equivariant graph neural networks with virtual nodes enhance protein binding site identification. *arXiv preprint arXiv:2404.07194*, 2024.

[87] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

[88] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[89] Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[90] Van Den Oord, A., Vinyals, O., et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[91] Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[92] Vyas, A., Shi, B., Le, M., Tjandra, A., Wu, Y.-C., Guo, B., Zhang, J., Zhang, X., Adkins, R., Ngan, W., et al. Audiobox: Unified audio generation with natural language prompts. *arXiv preprint arXiv:2312.15821*, 2023.

[93] Wang, C., Wang, Y., Xu, M., and Crandall, D. J. Stepwise goal-driven networks for trajectory prediction. *IEEE Robotics and Automation Letters*, 7(2):2716–2723, 2022.

[94] Xu, C., Tan, R. T., Tan, Y., Chen, S., Wang, Y. G., Wang, X., and Wang, Y. Eqmotion: Equivariant multi-agent motion prediction with invariant interaction reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1410–1420, 2023.

 https://doi.org/10.52202/085713-4684

[95] Xu, P., Hayet, J.-B., and Karamouzas, I. Socialvae: Human trajectory prediction using timewise latents. In *European Conference on Computer Vision*, pp. 511–528. Springer, 2022.

[96] Yadan, O. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL `https://github.com/facebookresearch/hydra`.

[97] Yao, Y., Atkins, E., Johnson-Roberson, M., Vasudevan, R., and Du, X. Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation. *IEEE Robotics and Automation Letters*, 6(2):1463–1470, 2021.

[98] Yu, Z., Huang, W., and Liu, Y. Force-guided bridge matching for full-atom time-coarsened dynamics of peptides. *arXiv preprint arXiv:2408.15126*, 2024.

[99] Yue, Y., Lucey, P., Carr, P., Bialkowski, A., and Matthews, I. Learning fine-grained spatial models for dynamic sports play prediction. In *IEEE International Conference on Data Mining*, pp. 670–679, 2014.

[100] Zhang, B. and Wonka, P. Lagem: A large geometry model for 3d representation learning and diffusion. *arXiv preprint arXiv:2410.01295*, 2024.

[101] Zhang, B., Tang, J., Niessner, M., and Wonka, P. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)*, 42 (4):1–16, 2023.

[102] Zwanzig, R. Nonlinear generalized langevin equations. *Journal of Statistical Physics*, 9(3): 215–220, 1973.

Appendix

Table of Contents

 https://doi.org/10.52202/085713-4684

A Notation

Table 6: Overview of used symbols and notations.

Definition	Symbol/Notation	Type
continuous time	$\hat{t}$	$\mathbb{R}$
overall number of (sampled) time steps	T	$\mathbb{N}$
number of observed time steps (when predicting later ones)	T_o	$\mathbb{N}$
number of future time steps (prediction horizon)	$T_f = T - T_o$	$\mathbb{N}$
time index for sequences of time steps	t	$\mathbb{N}$
system state space	$\mathcal{S}$	application-dependent set, to be further defined
system state	$\mathbf{s}$	$\mathcal{S}$
entity	e	symbolic
number of entities	N	$\mathbb{N}$
entity index	n	$1\,..\,N$
set of entities	E	$\{e_1, \ldots, e_n\}$
spatial entity dimensionality	D_x	$\mathbb{N}$
entity feature dimensionality	D_m	$\mathbb{N}$
entity location (coordinate)	$\mathbf{x}$	$\mathbb{R}^{D_x}$
entity properties (entity features)	$\mathbf{m}$	$\mathbb{R}^{D_m}$
identifier representation dimensionality	D_u	$\mathbb{N}$
number of latent vectors	L	$\mathbb{N}$
latent vector dimensionality	D_z	$\mathbb{N}$
trajectory of a system (locations of entities over time)	$\mathbf{X}$	$\mathbb{R}^{T_o \times N \times D_x}$
entity locations at t	$\mathbf{X}^t$	$\mathbb{R}^{N \times D_x}$
entity i of trajectory at t	$\mathbf{X}_i^t$	$\mathbb{R}^{D_x}$
trajectory in latent space	$\mathbf{Z}$	$\mathbb{R}^{T_o \times L \times D_z}$
latent system state at t	$\mathbf{Z}^t$	$\mathbb{R}^{L \times D_z}$
time invariant features of entities	$\mathbf{M}$	$\mathbb{R}^{N \times D_m}$
matrix of identifier embeddings	$\mathbf{U}$	$\mathbb{R}^{N \times D_u}$
projection matrices	$\mathbf{Q}, \mathbf{K}, \mathbf{V}$	not specified; depends on number of heads etc.
identifier assignment function	$\texttt{ida}(\cdot)$	$E \mapsto \mathcal{I}$
encoder	$\mathcal{E}(.)$	$\mathbb{R}^{N \times (D_u + D_x + D_m)} \mapsto \mathbb{R}^{L \times D_z}$
decoder	$\mathcal{D}(.)$	$\mathbb{R}^{L \times D_z} \times \mathbb{R}^{N \times D_u} \mapsto \mathbb{R}^{N \times (D_x + D_m)}$
approximator (time dynamics model)	$\mathcal{A}(.)$	$\mathbb{R}^{T \times L \times D_z} \mapsto \mathbb{R}^{T \times L \times D_z}$
loss function	$\mathcal{L}(.,.)$	var.
time parameter of the flow-based model	τ	$[0, 1]$
noise distribution	$\mathbf{o}_0$	$\mathbb{R}^{T \times L \times D_z}$
de-noised de-masked trajectory	$\mathbf{o}_1 = \mathbf{Z}$	$\mathbb{R}^{T \times L \times D_z}$
flow-based model "velocity prediction" (neural net)	$\boldsymbol{v}_\theta(\mathbf{o}_\tau, \tau)$	$\mathbb{R}^{T \times L \times D_z} \times \mathbb{R} \mapsto \mathbb{R}^{T \times L \times D_z}$
flow-based model "data prediction" (neural net)	$\boldsymbol{o}_\theta(\mathbf{o}_\tau, \tau)$	$\mathbb{R}^{T \times L \times D_z} \times \mathbb{R} \mapsto \mathbb{R}^{T \times L \times D_z}$
neural network parameters	θ	undef.

https://doi.org/10.52202/085713-4684

B Architecture Details

B.1 Architecture Overview in Detail

Figure 5 shows an expanded view of our architecture, and how the different components of LAM-SLIDE interact.

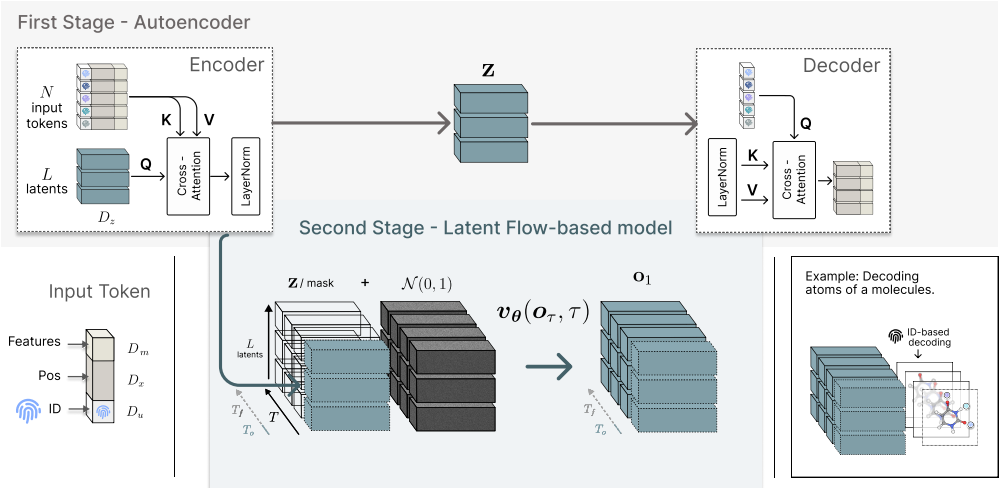

Figure 5: Expanded architectural overview. **First Stage**: The model is trained to reconstruct the encoded system by querying the latent system representation by IDs. **Second Stage**: Latent flow-based model is trained to predict multiple masked future timesteps. The predicted system states are decoded by the frozen decoder.

B.2 Identifier Assignment

We provide pseudocode for the identifier creation in Algorithm 1. This algorithm prevents the reuse of already assigned IDs, maintaining unique IDs across all entities. From a practical perspective, we sample IDs randomly so that all entity embeddings receive gradient updates.

Algorithm 1: Identifier Construction

Input : number of entities N; identifier pool size $|\mathcal{I}|$ where $N \leqslant |\mathcal{I}|$; embedding dimension D_u
Output : $\mathbf{U} \in \mathbb{R}^{N \times D_u}$

$\mathbf{U} \leftarrow$ empty matrix of size $N \times D_u$
$S \leftarrow \{\}$ `// track assigned identifiers`
for $i \leftarrow 1$ **to** N **do**
  $r \leftarrow \text{RandomSample}(\mathcal{I} \setminus S)$
  $S \leftarrow S \cup \{r\}$
  $\mathbf{e}_r \leftarrow \text{Embedding}(r)$ `// learnable embeddings`
  $\mathbf{U}[i] \leftarrow \mathbf{e}_r$
return $\mathbf{U}$

B.3 Encoder and Decoder

We provide pseudocode of the forward passes for encoding ($\mathcal{E}$) to and decoding ($\mathcal{D}$) from the latent system space of LAM-SLIDE in Algorithm 2 and Algorithm 3 respectively. In general, encoder and decoder blocks follow the standard Transformer architecture [91] with feedforward and normalization layers. To simplify the explanation, we omitted additional implementation details here and refer readers to our provided source code.

https://doi.org/10.52202/085713-4684

Algorithm 2: Encoder Function $\mathcal{E}$ (Cross-Attention)

Input : input data $\mathbf{XMU} = [\mathbf{X}, \mathbf{M}, \mathbf{U}] \in \mathbb{R}^{N \times (D_x + D_m + D_u)}$
Output : latent system state $\mathbf{Z} \in \mathbb{R}^{L \times D_z}$
Internal parameters : learned latent queries $\mathbf{Z}_{\text{init}} \in \mathbb{R}^{L \times D_z}$

$\mathbf{K} \leftarrow \text{Linear}(\mathbf{XMU})$
$\mathbf{V} \leftarrow \text{Linear}(\mathbf{XMU})$
$\mathbf{Q} \leftarrow \text{Linear}(\mathbf{Z}_{\text{init}})$
return $\text{LayerNorm}\big(\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})\big)$ `// without learnable affine parameters`

Algorithm 3: Decoder Function $\mathcal{D}$ (Cross-Attention)

Input : latent system representation $\mathbf{Z} \in \mathbb{R}^{L \times D_z}$; entity representation $\mathbf{u} \in \mathbb{R}^{D_u}$ drawn from $\mathbf{U} \in \mathbb{R}^{N \times D_u}$
Output : $[\mathbf{x}, \mathbf{m}] \in \mathbb{R}^{D_x + D_m}$

$\mathbf{Z} \leftarrow \text{LayerNorm}(\mathbf{Z})$ `// without learnable affine parameters`
$\mathbf{K} \leftarrow \text{Linear}(\mathbf{Z})$
$\mathbf{V} \leftarrow \text{Linear}(\mathbf{Z})$
$\mathbf{q} \leftarrow \text{Linear}(\mathbf{u})$
return $\text{Attention}\big([\mathbf{q}],\, \mathbf{K},\, \mathbf{V}\big)$

For the decoding functionality presented in Algorithm 3, we made use of multiple specific decoder blocks depending on the actual task (e.g., for the molecules dataset, we use one decoder block for atom positions and one decoder block for atom types).

B.4 Latent Flow Model

We provide pseudocode of the data prediction network $\boldsymbol{o}_\theta$ forward pass in Algorithm 4. The latent layer functionality is given by Algorithm 5. The architecture of the latent layers (i.e., our flow model) is based on Dehghani et al. [23], with the additional usage of adaptive layer norm (adaLN) [69] as also used for Diffusion Transformers [67]. The implementation is based on ParallelMLP block code from Black Forest Labs [14], which was adapted to use it along the latent dimension as well as along the temporal dimension (see Figure 6). The velocity model is obtained via reparameterization as outlined in App. C.1.

Algorithm 4: Latent Flow Model $\boldsymbol{o}_\theta$ (data prediction network)

Input : noise-interpolated data $\mathbf{o}_{\text{inter}} \in \mathbb{R}^{T \times L \times D_z}$; diffusion time τ used for interpolation; conditioning $\mathbf{C} \in \mathbb{R}^{T \times L \times D_z}$; conditioning mask $\mathbf{B} \in \{0, 1\}^{T \times L \times D_z}$
Output : prediction of original data (not interpolated with noise) $\mathbf{o} \in \mathbb{R}^{T \times L \times D_z}$

$\boldsymbol{\tau} \leftarrow \text{Embed}(\tau)$
$\mathbf{o} \leftarrow \text{Linear}(\mathbf{o}_{\text{inter}}) + \text{Linear}(\mathbf{C}) + \text{Embed}(\mathbf{B})$
for $i \leftarrow 1$ **to** *num_layers* **do**
  $\mathbf{o} \leftarrow \text{LatentLayer}(\mathbf{o}, \boldsymbol{\tau})$
$\alpha, \beta, \gamma \leftarrow \text{Linear}(\text{SiLU}(\boldsymbol{\tau}))$
return $\mathbf{o} + \gamma \odot \text{MLP}\big(\alpha \odot \text{LayerNorm}(\mathbf{o}) + \beta\big)$

Algorithm 5: LatentLayer

Input : $\mathbf{o} \in \mathbb{R}^{T \times L \times C}$; diffusion time embedding $\boldsymbol{\tau}$
Output : updated $\mathbf{o} \in \mathbb{R}^{T \times L \times C}$

$\mathbf{o} \mathrel{+}= \text{ParallelMLPAttentionWithRoPE}(\mathbf{o}, \boldsymbol{\tau}, \text{dim} = 0)$
$\mathbf{o} \mathrel{+}= \text{ParallelMLPAttentionWithRoPE}(\mathbf{o}, \boldsymbol{\tau}, \text{dim} = 1)$
return $\mathbf{o}$

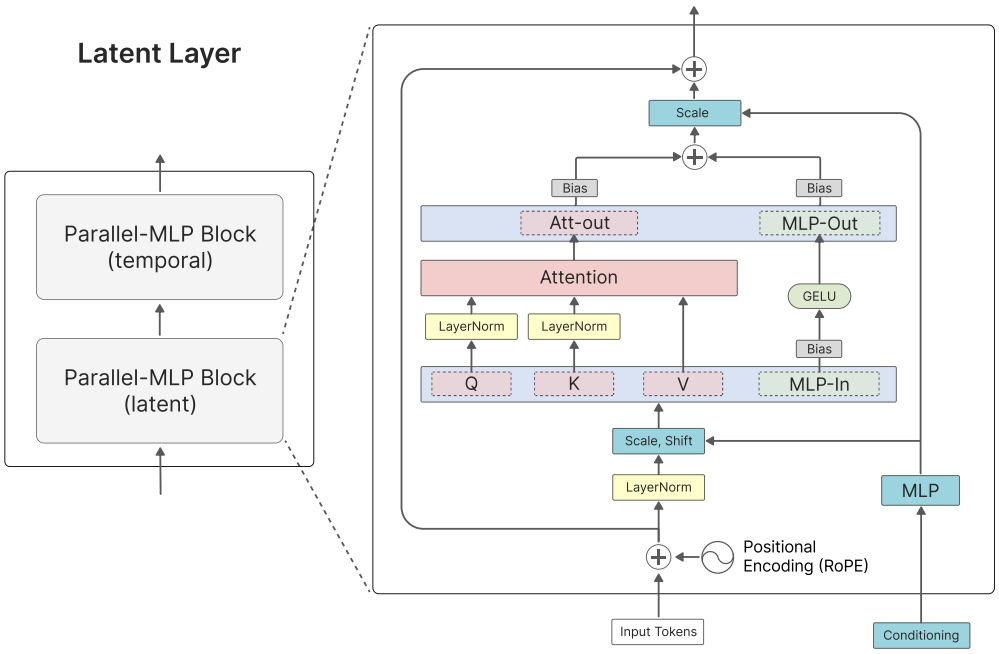

Figure 6: **Left**: LatentLayer of our method, consisting of a latent and a temporal ParallelMLP block. **Right**: Zoomed in view of the ParalellMLP block

Using `einops` [76] notation, the latent layer in Figure 6 can be expressed as:

$$\begin{aligned}
\mathbf{o}' &\leftarrow \texttt{rearrange}(\mathbf{o},\ (B\ \ L)\ \ T\ \ D_z \rightarrow (B\ \ T)\ \ L\ \ D_z) \\
\mathbf{o}' &\leftarrow l^i_\psi(\mathbf{o}', \tau) \\
\mathbf{o}' &\leftarrow \texttt{rearrange}(\mathbf{o}',\ (B\ \ T)\ \ L\ \ D_z \rightarrow (B\ \ L)\ \ T\ \ D_z) \\
\mathbf{o}' &\leftarrow l^i_\phi(\mathbf{o}', \tau)
\end{aligned}$$

with parameters sets ψ and ϕ, where for the latent block, the time dimension gets absorbed into the batch dimension, and for the temporal block, the latent dimension gets absorbed into the batch dimension.

B.5 Python Pseudocode

This section shows Python pseudocode for training and inference for the latent approximator model.

```
def latent_layer(z, t, z_cond):
    ## Latent layer of the approximator (high level)
    z = z + embed_cond(z_cond)
    t_embed = embed_time(t)

    # T x L x Dz, attention is calculated over L     | ref [4,5]
    z = attention_spatial(z, t_embed)

    z = rearrange(z, "T L Dz -> L T Dz")

    # L x T x Dz, attention is calculated over T     | ref [4,5]
    z = attention_temporal(z, t_embed)

    z = rearrange(z, "L T Dz -> T L Dz")

    return z
```


https://doi.org/10.52202/085713-4684

```
##################
### Training ####
##################

# T x N x Dn (N=number of entities, D=[x,y,z,atom_type])
# e.g. Aspirin 21x4
x
ids
# T binary mask [1, 0, 0, ...], e.g., one conditioning frame
mask

# Encoding and conditioning setup

# T x L x Dz Encoder state into latent vector
z1 = encode(x, ids)

# T x L x Dz Contains only the conditioning frames,
# the other frames are masked out and set to 0.
z_cond = mask_frames(z1, mask)

# Sample z0 from a normal distribtion
z0 = torch.randn_like(z1)

# Sample t uniformly
t = torch.uniform(0, 1)

# Create interpolation
zt = t * z1 + (1 - t) * z0

# Loss
z1_hat = approximator(zt, t, z_cond)
loss = torch.mean((z1_hat - z1)**2)
loss.backward()


##################
### Inference ####
##################

# T x N x Dn, but non-conditioning frames are zero
x_cond

# Transform model trained on data prediction into velocity
# prediction model.
velocity_model = data_to_velocity_model(approximator)

# T x L x Dz, contains only the conditioning
# frames/ missing frames are set to 0
z_cond = encode(x_cond, ids)
z0 = torch.randn_like(z1)

# Solve ODE torchdiffeq (https://github.com/rtqichen/torchdiffeq)
z1_pred = solver(velocity_model, z0, z_cond)

x_hat = decoder(z1_pred)
```

C Additional Information on Stochastic Interpolants

General interpolants. The stochastic interpolants framework [2, 3, 60] is defined without reference to a forward SDE, which allows a lot of flexibility; any choice of α_t and σ_t, satisfying the following conditions, is possible:

1. $\alpha_\tau^2 + \sigma_\tau^2 > 0$;
2. α_τ and σ_τ are differentiable for all $\tau \in [0, 1]$
3. $\alpha_0 = \sigma_1 = 0$ and $\alpha_1 = \sigma_0 = 1$

Interpolants. Two common choices for α_t and σ_t are the Linear and a Generalized Variance-Preserving (GVP) path:

$$\text{Linear:} \quad \alpha_\tau = \tau, \qquad \sigma_\tau = 1 - \tau \tag{7}$$

$$\text{GVP:} \quad \alpha_\tau = \sin\left(\frac{1}{2}\pi\tau\right), \qquad \sigma_\tau = \cos\left(\frac{1}{2}\pi\tau\right) \tag{8}$$

C.1 Parametrization

Our latent flow-based model is implemented via data prediction objective [55, 48], with the aim to have small differences:

$$||\hat{\boldsymbol{o}}_\theta(\mathbf{o}; \tau) - \mathbf{o}_1||^2 \, . \tag{9}$$

The velocity model $\hat{\boldsymbol{v}}_\theta$ is obtained by reparameterization according to Lipman et al. [55]:

$$\hat{\boldsymbol{v}}_\theta(\mathbf{o}, \tau) = \hat{\boldsymbol{s}}_\theta(\mathbf{o}; \tau)\left(\frac{\dot{\alpha}_\tau \sigma_\tau^2}{\alpha_\tau} - \sigma_\tau \dot{\sigma}_\tau\right) + \frac{\dot{\alpha}_\tau}{\alpha_\tau}\mathbf{o} \tag{10}$$

where

$$\hat{\boldsymbol{s}}_\theta(\mathbf{o}; \tau) = -\sigma_\tau^{-2}(\mathbf{o} - \alpha_\tau \hat{\boldsymbol{o}}_\theta(\mathbf{o}; \tau)) \, . \tag{11}$$

For integration, we employed the `torchdiffeq` package [20], which provides solvers for differential equations.

 https://doi.org/10.52202/085713-4684

D Additional related work

D.1 Molecular Dynamics (MD)

The most fundamental concepts used to describe the dynamics of molecules nowadays are those given by the laws of quantum mechanics. The SchrÃűdinger equation is a partial differential equation that gives the evolution of the complex-valued wave function ψ over time t: $i\hbar\frac{\partial\psi}{\partial t} = \hat{H}(t)\psi$. Here i is the imaginary unit with $i^2 = -1$, $\hbar$ is the reduced Planck constant, and $\hat{H}(t)$ is the Hamiltonian operator at time t, which is applied to a function ψ and maps to another function. It determines how a quantum system evolves with time, and its eigenvalues correspond to measurable energy values of the quantum system. The solution to SchrÃűdinger's equation in the many-body case (particles $1, \ldots, N$) is the wave function $\psi(\mathbf{x}_1, \ldots, \mathbf{x}_N, t) : \bigtimes_{i=1}^{N} \mathbb{R}^3 \times \mathbb{R} \to \mathbb{C}$ which we abbreviate as $\psi(\{\mathbf{x}\}, t)$. It's the square modulus $|\psi(\{\mathbf{x}\}, t)|^2 = \psi^*(\{\mathbf{x}\}, t)\psi(\{\mathbf{x}\}, t)$ is usually interpreted as a probability density to measure the positions $\mathbf{x}_1, \ldots, \mathbf{x}_N$ at time t, whereby the normalization condition $\int \ldots \int |\psi(\{\mathbf{x}\}, t)|^2 d\mathbf{x}_1 \ldots d\mathbf{x}_N = 1$ holds for the wave function ψ.

Analytic solutions of ψ for specific operators $\hat{H(t)}$ are hardly known and are only available for simple systems like free particles or hydrogen atoms. In contrast to that are proteins, which consist of many thousands of atoms. However, already for much smaller quantum systems, approximations are needed. A famous example is the BornâĂŞOppenheimer approximation, where the wave function of the multi-body system is decomposed into parts for heavier atom nuclei and the light-weight electrons, which usually move much faster. In this case, one obtains a SchrÃűdinger equation for the electron's movement and another SchrÃűdinger equation for the nucleus's movement. A much faster option than solving a second SchrÃűdinger equation for the motion of the nuclei is to use the laws from classical Newtonian dynamics. The solution of the first SchrÃűdinger equation defines an energy potential, which can be utilized to obtain forces $\mathbf{F}_i$ on the nuclei and to update nuclei positions according to Newton's equation of motion: $\mathbf{F}_i = m_i\, \ddot{\mathbf{q}}_i(t)$ (with m_i being the mass of particle i and $\mathbf{q}_i(t)$ describing the motion trajectory of particle i over time t).

Additional complexity in studying molecule dynamics is introduced by the environmental conditions surrounding molecules. Maybe the most important property is temperature. For biomolecules, it is often of interest to assume that they are dissolved in water. To model temperature, a usual strategy is to assume a system of coupled harmonic oscillators to model a heat bath, from which Langevin dynamics can be derived [29, 102]. The investigation of the relationship between quantum-mechanical modeling of heat baths and Langevin dynamics remains a current research topic, with various aspects, including the coupling of oscillators and the introduction of Markovian properties with stochastic forces. For instance, Hoel & Szepessy [39] studies how canonical quantum observables are approximated by molecular dynamics. This includes the definition of density operators, which behave according to the quantum Liouville-von Neumann equation.

The forces in molecules are usually given as the negative derivative of the (potential) energy: $\mathbf{F}_i = -\nabla E$. In the context of molecules, E is usually assumed to be defined by a force field, which is a parameterized sum of intra- and intermolecular interaction terms. An example is the Amber force field [72, 18]:

$$E = \sum_{\text{bonds } r} k_b(r - r_0)^2 + \sum_{\text{angles } \theta} k_\theta(\theta - \theta_0)^2 + \tag{12}$$
$$\sum_{\text{dihedrals } \phi} V_n(1 + cos(n\phi - \gamma)) + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\epsilon R_{ij}} \right)$$

Here $k_b, r_0, k_\theta, \theta_0, V_n, \gamma, A_{ij}, B_{ij}, \epsilon, q_i, q_j$ serve as force field parameters, which are found either empirically or which might be inspired by theory.

Newton's equations of motion for all particles under consideration form a system of ordinary differential equations (ODEs), to which various numerical integration schemes, such as Euler, Leapfrog, or Verlet, can be applied to obtain particle position trajectories for given initial positions and velocities. If temperature is included, the resulting Langevin equations form a system of stochastic differential equations (SDEs), and Langevin integrators can be employed. It should be mentioned

that it is often necessary to use very small integration timesteps to avoid large approximation errors. This, however, increases the time needed to find new stable molecular configurations.

D.2 Relationship to Graph Foundation Models

From our perspective, LAM-SLIDE bears a relationship to graph foundation models [GFMs; 56, 62]. Bommasani et al. [16] consider foundation models to be *trained on broad data at scale* and to be *adaptable to a wide range of downstream tasks*. Mao et al. [62] argue that graphs are more diverse than natural language or images, and therefore, there are unique challenges for GFMs. Especially, they mention that *none of the current GFM have the capability to transfer across all graph tasks and datasets from all domains*. It is for sure true that LAM-SLIDE is not a GFM in this sense. However, it might be debatable whether LAM-SLIDE might serve as a domain- or task-specific GFM. While we primarily focused on a trajectory prediction task and are, from that point of view, task-specific, we observed that our trained models can generalize across different molecules or scenes, which may seem quite remarkable given that it is common practice to train specific trajectory prediction models for individual molecules or scenes. Nevertheless, it was not our aim in this research to provide a GFM, as we believe that this would require further investigation into additional domains and could also necessitate, for instance, examining whether emergent abilities might arise with larger models and more training data [56].

D.3 Relationship to Video and Language Diffusion Models

We want to elaborate our perspective on the relationship between LAM-SLIDE and recent advances in video [15] and language diffusion models [79, 59]. At their core, these approaches share a fundamental similarity: they can be conceptualized as a form of unmasking.

In video diffusion models, the model unmasks future frames; in language diffusion models, the model unmasks unknown tokens. Both paradigms learn to recover information that is initially obscured in the sequence, and importantly, both methods do so in parallel over the entire input sequence [9], compared to autoregressive models, which predict a single frame or token at a time.

Similarly, LAM-SLIDE represents each timestep as a set of latent tokens (or as a single token when concatenated). This perspective allows us to seamlessly incorporate recent advances from both video and language diffusion research into our modeling paradigm.

 https://doi.org/10.52202/085713-4684

E Experimental Details

E.1 Datasets

Pedestrian Movement. The pedestrian movement dataset, along with its data processing, is available at `https://github.com/MediaBrain-SJTU/EqMotion`.

Basketball Player Movement. The dataset, along with its predefined splits, is available at `https://github.com/xupei0610/SocialVAE`. Data processing is provided in our source code.

N-Body. The dataset creation scripts, along with their predefined splits, are available at `https://github.com/hanjq17/GeoTDM`.

Small Molecules (MD17). The MD17 dataset is available at `http://www.sgdml.org/#datasets`. Preprocessing and dataset splits follow Han et al. [35] and can be accessed through their GitHub repository at `https://github.com/hanjq17/GeoTDM`. The dataset comprises 5,000 training, 1000 validation, and 1000 test trajectories for each molecule.

Tetrapeptides. The dataset, including the full simulation parameters for ground truth simulations, is sourced from Jing et al. [44] and is publicly available in their GitHub repository at `https://github.com/bjing2016/mdgen`. The dataset comprises 3,109 training, 100 validation, and 100 test peptides.

E.2 Condition and Prediction Horizon

Table 7 shows the conditioning and prediction horizon for the individual experiments. For the Tetrapeptides experiments, we predicted 1000 steps in parallel and reconditioned the model ten times on the last frame for each predicted block, this concept is similar to Arriola et al. [9].

Table 7: Number of conditioning and predicted frames for the different experiments.

Experiment	Conditioning Frames	Predicted Frames	Total Frames
Pedestrian trajectory forecasting (ETH-UCY)	8	12	20
Basketball (NBA)	8	12	20
N-Body	10	20	30
Molecular Dynamics (MD17)	10	20	30
Molecular Dynamics - Tetrapeptides (4AA)	1	9 999	10 000

E.3 Implementation Details

Training procedure. (i) **First Stage**. In the first stage, we train the encoding and decoding functions $\mathcal{E}$ and $\mathcal{D}$ in an auto-encoding fashion, i.e., we optimize for a precise reconstruction of the original system state representation from its latent representation. For discrete features (e.g., atom type, residue type), we tend to use a cross-entropy loss, whereas for continuous features we use a regression loss (e.g., position, distance). The loss functions for each task are summarized in Appendix E.5. Notably, the entity identifier assignment is also random. (ii) **Second Stage**. In the second stage, we freeze the encoder and train the approximator to model the temporal dynamics via the encoded latent system representations. To learn a consistent behavior over time, we pass $\mathbf{U}$ from the encoder $\mathcal{E}$ to the decoder $\mathcal{D}$. To avoid high variance latent spaces, we used layer-normalization [10] (see Appendix E.3).

Data Augmentation. To compensate for the absence of built-in inductive biases such as equivariance/invariance with respect to spatial transformations, we apply random rotations and translations to the input coordinates.

Identifiers. For the embedding of the identifiers we use a `torch.nn.Embedding` [66] layer, where we assign a random subset of the possible embeddings to the entities in each training step. See also Algorithm 1.

Latent space regularization. To avoid high variance latent spaces, Rombach et al. [77] relies on KL-reg., imposing a small KL-penalty towards a standard normal on the latent space, as used in VAE [49]. Recent work [100] has shown that layer normalization [10] can achieve similar regulatory effects without requiring an additional loss term and simplifying the training procedure. We adapt this approach in our method (see the left part of Figure 2).

Latent Model. For the latent Flow Model we additionally apply auxiliary losses for the individual tasks, as shown in App. E.5. Where we decode the predicted latent system representations and back-propagate through the frozen decoder to the latent model.

MD17. We train a single model on all molecules – a feat that is structurally encouraged by the design of LAM-SLIDE . For ablation, we also train GeoTDM [35] on all molecules and evaluate the performance on each one of them ("all→each" in the Table 13). Interestingly, we also observe consistent improvements in the GeoTDM performance. However, GeoTDM's performance does not reach that of LAM-SLIDE .

Tetrapeptides. For the experiments on tetrapeptides in Section 4.5, we employ the Atom14 representation as used in AlphaFold [1]. In this representation, each entity corresponds to one amino acid of the tetrapeptide, where multiple atomic positions are encoded into a single vector of dimension $D_x = 3 \times 14$. Masked atomic positions are excluded from gradient computation during model updates. This representation is computationally more efficient.

E.4 Loss Functions

This section defines the losses, which we use throughout training:

Position Loss.

$$\mathcal{L}_{\text{pos}}(\mathbf{X}^t, \hat{\mathbf{X}}^t) = \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{X}_i^t - \hat{\mathbf{X}}_i^t||_2^2 \tag{13}$$

Inter-distance Loss.

$$\mathcal{L}_{\text{int}}(\mathbf{X}^t, \hat{\mathbf{X}}^t) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (D_{ij}(\mathbf{X}^t) - D_{ij}(\hat{\mathbf{X}}^t))^2 \tag{14}$$

with

$$D_{ij}(\mathbf{X}^t) = ||\mathbf{X}_i^t - \mathbf{X}_j^t||_2 \tag{15}$$

Cross-Entropy Loss. Depending on the experiment, we have different CE losses depending on the problem, see App. E.5.

$$\mathcal{L}_{CE} = \frac{1}{N} \left(- \sum_{k=1}^{K} y_k \log(p_k) \right) \tag{16}$$

Frame and Torsion Loss. For the Tetrapeptide experiments, we employ two additional auxiliary loss functions tailored to better capture unique geometric constraints of proteins, complementing our primary optimization objectives: a frame loss $\mathcal{L}_{frame}$, which is based on representing all atoms withing a local reference frame [1, Algorithm 29], and a torsion loss torsion loss $\mathcal{L}_{\text{tors}}$ inspired by Jumper et al. [46].

E.5 Hyperparameters

Tabs. 8 to 12 show the hyperparameters for the individual tasks, loss functions are as defined in App. E.4. For all trained models, we use the AdamW [50, 58] optimizer and use EMA [31] in each update step with a decay parameter of $\beta = 0.999$.

 https://doi.org/10.52202/085713-4684

E.6 Evaluation Details

Tetrapeptides. Our analysis of the Tetrapeptide trajectories utilized PyEMMA [84] and followed the procedure as Jing et al. [44], incorporating both Time-lagged Independent Component Analysis (TICA) [70] and Markov State Models (MSM) [42]. For the evaluation of these metrics, we relied on the implementations provided by [44].

E.7 Computational Resources

Our experiments were conducted using a system with 128 CPU cores and 2048GB of system memory. Model training was performed on 4 NVIDIA H200 GPUs, each equipped with 140GB of VRAM. In total, roughly 5000 GPU hours were used in this work.

E.8 Software

We used `PyTorch 2` [7] for the implementation of our models. Our training pipeline was structured with `PyTorch Lightning` [28]. We used `Hydra` [96] to run our experiments with different hyperparameter settings. Our experiments were tracked with `Weights & Biases` [12].

Table 8: Hyperparameter configuration for the pedestrian movement experiments (Section 4.1).

First Stage	
Network	
Encoder	
Number of latents L	2
Number of entity embeddings	8
Number of attention heads	2
Number of cross attention layers	1
Dimension latents D_z	32
Dimension entity embedding	128
Dimension attention head	16
Decoder	
Number of attention heads	2
Number of cross attention layers	1
Dimension attention head	16
Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	1
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	1
Training	
Learning rate	1e-4
Learning rate scheduler	CosineAnnealing(min_lr=1e-7)
Batch size	256
Epochs	3K
Precision	32-Full
Batch size	1024
Second Stage	
Setup	
Condition	8 Frames
Prediction	12 Frames
Network	
Hidden dimension	128
Number of Layers	6
Auxiliary - Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	0.25
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	0.25
Training	
Learning rate	1e-3
Batch size	64
Epochs	1K
Precision	BF16-Mixed
Inference	
Integrator	Euler
ODE steps	10

https://doi.org/10.52202/085713-4684

Table 9: Hyperparameter configuration for the basketball player movement experiments (Section 4.2).

First Stage	
Network	
Encoder	
Number of latents L	32
Number of entity embeddings	11
Number of attention heads	2
Number of cross attention layers	1
Dimension latents D_z	32
Dimension entity embedding	128
Dimension attention head	16
Decoder	
Latent dimension	32
Number of attention heads	8
Number of cross attention layers	1
Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	1
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	1
$\mathcal{L}_{CE}(\cdot, \cdot)$ – Group	0.01
$\mathcal{L}_{CE}(\cdot, \cdot)$ – Team	0.01
Training	
Learning rate	1e-4
Learning rate scheduler	CosineAnnealing(min_lr=1e-7)
Optimizer	AdamW
Batch size	16
Second Stage	
Setup	
Condition	8 Frames
Prediction	12 Frames
Network	
Hidden dimension H	128
Number of Layers	6
Auxiliary - Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	0.25
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	0.25
Training	
Learning rate	1e-3
Learning rate scheduler	CosineAnnealing(min_lr=1e-7)
Batch size	64
Epochs	500
Precision	BF16-Mixed
Inference	
Integrator	Euler
ODE steps	10

Table 10: Hyperparameter configuration for the N-Body experiments (Section 4.3).

First Stage	
Network	
Encoder	
Number of latents L	16
Number of entity embeddings	10
Number of attention heads	2
Number of cross attention layers	1
Dimension latents D_z	32
Dimension entity embedding	128
Dimension attention head	16
Decoder	
Number of cross attention layers	1
Number of attention heads	2
Number of cross attention layers	16
Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	1
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	1
Training	
Learning rate	1e-3
Batch size	128
Epochs	2K
Precision	32-Full
Second Stage	
Setup	
Condition	10 Frames
Prediction	20 Frames
Network	
Hidden dimension	256
Number of Layers	6
Auxiliary - Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	0.0
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	0.0
Training	
Learning rate	1e-3
Learning rate scheduler	CosineAnnealing(min_lr=1e-7)
Batch size	64
Epochs	1K
Precision	BF16-Mixed
Inference	
Integrator	Euler
ODE steps	10

https://doi.org/10.52202/085713-4684

Table 11: Hyperparameter configuration for the small molecule (MD17) experiments (Section 4.4).

First Stage	
Network	
Encoder	
Number of latents L	32
Number of entity embeddings	8
Number of attention heads	2
Number of cross attention layers	1
Dimension latents D_z	32
Dimension entity embedding	128
Dimension attention head	16
Decoder	
Number of cross attention layers	1
Number of attention heads	2
Number of cross attention layers	16
Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	1
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	1
$\mathcal{L}_{CE}(\cdot, \cdot)$ – Atom type	1
Training	
Learning rate	1e-4
Batch size	256
Epochs	3K
Precision	32-Full
Second Stage	
Setup	
Condition	10 Frames
Prediction	20 Frames
Network	
Hidden dimension	128
Number of Layers	6
Auxiliary - Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	0.25
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	0.25
Training	
Learning rate	1e-3
Learning rate scheduler	CosineAnnealing(min_lr=1e-7)
Batch size	64
Epochs	2K
Precision	BF16-Mixed
Inference	
Integrator	Euler
ODE steps	10

Table 12: Hyperparameter configuration for the Tetrapeptides experiments (Section 4.5).

First Stage	
Network	
Encoder	
Number of latents L	5
Number of entity embeddings	8
Number of attention heads	2
Number of cross attention layers	1
Dimension latents D_z	96
Dimension entity embedding	128
Dimension attention head	16
Decoder	
Number of attention heads	2
Number of cross attention layers	1
Dimension attention head	16
Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	1
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	1
$\mathcal{L}_{frame}(\mathbf{X}, \hat{\mathbf{X}})$	1
$\mathcal{L}_{tors}(\mathbf{X}, \hat{\mathbf{X}})$	0.1
$\mathcal{L}_{CE}(\cdot, \cdot)$ – Residue type	0.001
Training	
Learning rate	1e-4
Batch size	16
Epochs	200K
Precision	32-Full
Second Stage	
Setup	
Condition	1 Frame
Prediction	10,000 Frames (10x rollouts)
Network	
Hidden dimension	384
Number of Layers	6
Auxiliary - Loss	**Weight**
$\mathcal{L}_{pos}(\mathbf{X}, \hat{\mathbf{X}})$	0.25
$\mathcal{L}_{int}(\mathbf{X}, \hat{\mathbf{X}})$	0.25
$\mathcal{L}_{frame}(\mathbf{X}, \hat{\mathbf{X}})$	0.25
Training	
Learning rate	1e-3
Optimizer	AdamW
Batch size	64
Epochs	1.5K
Precision	BF16-Mixed
Inference	
Integrator	Dopri5 [20]
ODE steps	adaptive

https://doi.org/10.52202/085713-4684

F Proofs

F.1 Proof of Proposition 3.3

The proof is rather simple, but we include it for completeness.

Proposition 3.3. *Given an identifier pool $\mathcal{I}$ and a finite set of entities E, an identifier assignment pool I as defined by Definition 3.2 is non-empty if and only if $|E| \leqslant |\mathcal{I}|$.*

Proof. Assume $|E| > |\mathcal{I}|$. By pigeonhole principle, any function $f : E \mapsto \mathcal{I}$ must map at least two distinct elements of E to the same element in I. Therefore, f cannot be injective.

Conversely, if $|E| \leqslant |\mathcal{I}|$, we can construct an injective function from E to $\mathcal{I}$ by assigning each element in E a unique element in $\mathcal{I}$, which is possible because $\mathcal{I}$ has at least as many elements as E.

Therefore, an injective identifier assignment function $\texttt{ida}(\cdot) \in I$ only exists if $|E| \leqslant |\mathcal{I}|$. Hence, the set I is non-empty in this case and empty otherwise.

F.2 Proof of Proposition 3.4

Proposition 3.4. *Given an identifier pool $\mathcal{I}$ and a finite set of entities E such that $|E| \leqslant |\mathcal{I}|$, the identifier assignment pool I as defined by Definition 3.2 contains finitely many injective functions.*

Let $n = |E|$ and $m = |\mathcal{I}|$, then the set of infective functions I is bounded and finite:

$$|I| = (m-1)\dots(m-n+1) = \frac{m!}{(m-n)!} = (m)_n \leqslant \inf \tag{17}$$

Where $(m)_n$ is commonly referred to as *falling factorials*, the number of injective functions from a set of size n to a set of size m.

G Additional Experiments

To investigate the sensitivity of our model with respect to different hyperparameter settings, we conducted additional experiments.

G.1 Number of Parameter

We conducted scaling experiments on both the MD17 and the Tetrapeptides (4AA) datasets to evaluate how LAM-SLIDE 's performance scales with model size. On MD17, we evaluate LAM-SLIDE using model variants with 1.7M, 2.1M, and 2.5M parameters. Our results show that, for nearly all molecules, performance improves with parameter count in terms of ADE/FDE, see Table 13. Similarly, on the Tetrapeptides dataset, we evaluate using model variants with 4M, 7M, 11M, and 28M parameters. All performance metrics show consistent improvement with increased model capacity, see Table 14. These findings indicate the favorable scaling behavior of our method.

Table 13: **Method comparison for forecasting MD trajectories of small molecules**. Compared methods predict atom positions for 20 frames, conditioned on 10 input frames. Results are reported in terms of ADE/FDE, averaged over 5 sampled trajectories.

	Aspirin		Benzene		Ethanol		Malonaldehyde		Naphthalene		Salicylic		Toluene		Uracil	
	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
RF [52][a]	0.303	0.442	0.120	0.194	0.374	0.515	0.297	0.454	0.168	0.185	0.261	0.343	0.199	0.249	0.239	0.272
TFN [89][a]	0.133	0.268	0.024	0.049	0.201	0.414	0.184	0.386	0.072	0.098	0.115	0.223	0.090	0.150	0.090	0.159
SE(3)-Tr. [30][a]	0.294	0.556	0.027	0.056	0.188	0.359	0.214	0.456	0.069	0.103	0.189	0.312	0.108	0.184	0.107	0.196
EGNN [82][a]	0.267	0.564	0.024	0.042	0.268	0.401	0.393	0.958	0.095	0.133	0.159	0.348	0.207	0.294	0.154	0.282
EqMotion [94][a]	0.185	0.246	0.029	0.043	0.152	0.247	0.155	0.249	0.073	0.092	0.110	0.151	0.097	0.129	0.088	0.116
SVAE [95][a]	0.301	0.428	0.114	0.133	0.387	0.505	0.287	0.430	0.124	0.135	0.122	0.142	0.145	0.171	0.145	0.156
GeoTDM 1.9M[a]	0.107	0.193	0.023	0.039	0.115	0.209	0.107	0.176	0.064	0.087	0.083	0.120	0.083	0.121	0.074	0.099
GeoTDM 1.9M (all→each)	0.091	0.164	0.024	0.040	0.104	0.191	0.097	0.164	0.061	0.092	0.074	0.114	0.073	0.112	0.070	0.102
LAM-SLIDE 2.5M	**0.059**	**0.098**	**0.021**	**0.032**	**0.087**	**0.167**	**0.073**	**0.124**	**0.037**	**0.058**	**0.047**	**0.074**	**0.045**	**0.075**	**0.050**	**0.074**
LAM-SLIDE 2.1M	0.064	0.104	0.023	0.033	0.097	0.182	0.084	0.141	0.044	0.067	0.053	0.081	0.054	0.086	0.054	0.079
LAM-SLIDE 1.7M	0.074	0.117	0.025	0.037	0.110	0.195	0.097	0.159	0.053	0.074	0.063	0.091	0.064	0.094	0.064	0.089

[a] Results from Han et al. [35].

Table 14: **Method comparison for predicting MD trajectories of tetrapeptides**. The columns denote the JSD between distributions of *torsion angles* (backbone (BB), side-chain (SC), and all angles), the TICA, the MSM metric, and the number of parameters.

	Torsions			TICA		MSM	Params	Time
	BB	SC	All	0	0,1 joint		(M)	
100 ns[a]	.103	.055	.076	.201	.268	.208		$\sim$ 3h
MDGen[a]	.130	**.093**	**.109**	.230	.316	.235	34	$\sim$ 60s
LAM-SLIDE	**.128**	0.122	0.125	**.227**	**.315**	**.224**	28	$\sim$ 53s
LAM-SLIDE	.152	.151	.152	.239	.331	.226	11	
LAM-SLIDE	.183	.191	.187	.26	.356	.235	7	
LAM-SLIDE	.284	.331	.311	.339	.461	.237	**4**	

[a] Results from Jing et al. [44].

G.2 Number of Function Evaluations (NFEs)

We conduct a comparative analysis on computational efficiency by measuring the number of function evaluations (NFEs) required to achieve the performance results reported in the main section of our publication. As shown in Table 15, our approach demonstrates remarkable efficiency compared to the previous state-of-the-art method, GeoTDM [35], across N-Body simulations, MD17 molecular dynamics, and ETH pedestrian motion forecasting experiments. Our model consistently requires significantly fewer NFEs than GeoTDM to reach comparable or superior performance levels.

https://doi.org/10.52202/085713-4684

It is worth noting that flow-based models generally require fewer NFEs compared to diffusion-based approaches, such as GeoTDM. However, this efficiency advantage does not come at the expense of performance quality [27]. Indeed, the relationship between NFEs and performance is not strictly monotonic, as demonstrated in other domains. For instance, Esser et al. [27] achieved optimal image generation results in terms of FID [36] with 25 NFEs, demonstrating that computational efficiency and high performance can be achieved simultaneously with properly designed architectures.

Furthermore, in the case of the Tetrapeptides (4AA) experiments shown in Section 4.5, we employ an adaptive step size solver to achieve the reported performance, which yields better results than an Euler solver. We use Dopri5 as implemented in the `torchdiffeq` package [20].

Table 15: Comparison of the number of function evaluations (NFEs) for LAM-SLIDE and GeoTDM.

	N-Body	MD17	ETH
GeoTDM [35][a]	1000	1000	100
LAM-SLIDE	**10**	**10**	**10**

[a] Results from Han et al. [35].

G.3 Number of Learned Latent Vectors

We conducted experiments to quantify the relationship between model performance and the number of latent vectors L using the MD17 dataset. As shown in Table 16, performance increases with the number of latent vectors L. Of particular significance is the performance at $L = 21$, which corresponds to the maximum number of entities, allowing us to investigate whether this constitutes an upper bound on model capacity. Notably, performance continues to improve at $L = 32$, indicating that model capacity scales favorably even beyond the number of entities. At $L = 16$, representing a compressed latent representation, our model remains competitive with the second-best method, GeoTDM [35].

We further analyze whether the improvement resulting from increasing L is due solely to the encoder-decoder's reconstruction performance. Figure 8 shows the reconstruction error for varying number of latent vectors L. Even with substantially fewer latent vectors than entities, the model achieves good reconstruction performance. This gap suggests that the performance gains from increasing L are not due to improved reconstruction, but rather to the model's ability to leverage the enlarged latent space representation more effectively.

G.4 Identifier Pool Size

We evaluate the impact of identifier pool size using the MD17 dataset. This dataset contains at most 21 atoms, so, in general, an identifier pool of $|\mathcal{I}| = 21$ would be sufficient. However, for the results shown in the main paper, we used $|\mathcal{I}| = 32$. To investigate the impact of a larger identifier pool $\mathcal{I}$, we conduct additional experiments by training multiple first-stage models with varying identifier pool sizes and report the reconstruction error measured by Euclidean distance. Figure 7 shows that the reconstruction error increases with the size of the identifier pool, since a larger pool results in fewer updates to each entity embedding during training.

G.5 Identifier Assignment

To assess the impact of different ID assignments on reconstruction performance, we run our model five times with different random ID assignments and measure the standard deviation across these assignments in terms of reconstruction error in Å. Results are shown in Figures 7 and 8. The low standard deviation across different ID assignments demonstrates the robustness of our model with respect to random ID assignment.

Table 16: Model performance in terms of ADF/FDE with respect to **different number of latent vectors** L.

	Aspirin		Benzene		Ethanol		Malonaldehyde		Naphthalene		Salicylic		Toluene		Uracil	
	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
LAM-SLIDE ($L = 4$)	0.354	0.483	0.213	0.264	0.420	0.541	0.366	0.537	0.210	0.223	0.253	0.286	0.316	0.418	0.243	0.275
LAM-SLIDE ($L = 8$)	0.234	0.315	0.114	0.135	0.242	0.361	0.201	0.300	0.157	0.163	0.179	0.201	0.206	0.250	0.158	0.180
LAM-SLIDE ($L = 16$)	0.099	0.146	0.039	0.049	0.108	0.187	0.095	0.149	0.070	0.089	0.075	0.101	0.073	0.103	0.071	0.093
LAM-SLIDE ($L = 21$)	0.078	0.118	0.031	0.041	0.097	0.175	0.082	0.135	0.054	0.074	0.059	0.085	0.057	0.085	0.059	0.083
LAM-SLIDE ($L = 32$)	**0.059**	**0.098**	**0.021**	**0.032**	**0.087**	**0.167**	**0.073**	**0.124**	**0.037**	**0.058**	**0.047**	**0.074**	**0.045**	**0.075**	**0.050**	**0.074**

G.6 Identifier Latent Utilization

We investigate how identifiers are utilized by our model, addressing three key questions: Does each identifier exhibit a unique query pattern? How do these patterns vary with latent space dimensionality? Do different attention heads learn distinct patterns?

We examined decoder attention weights in overparameterized ($N < L$) and compressed ($N \geq L$) settings using models with $L = 16$ and $L = 32$ latent vectors trained on MD17. Our analysis focuses on Aspirin molecules ($N = 21$ atoms) across different conformations.

The results in Figs. 9 to 12 reveal two main findings. First, identical atom identifiers produce consistent attention patterns across different conformations, with each identifier learning a unique addressing scheme distributed over latent vectors. This suggests that the identifiers function as "retrieval mechanisms" independent of stored content.

Second, attention head utilization adapts to the compression ratio. The overparameterized model ($L = 32$) primarily uses a single head, acting like a hash table-like retrieval. The compressed model ($L = 16$) utilizes both heads with distinct patterns, likely to mitigate identifier collisions in the limited latent space.

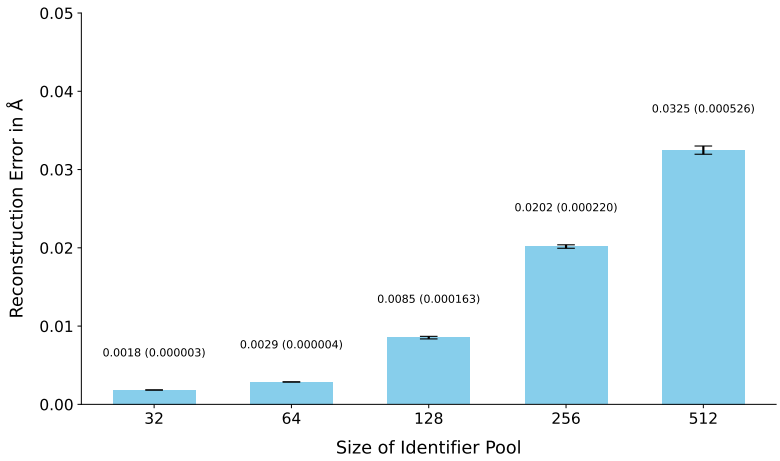

Figure 7: Reconstruction error in Å for the MD17 dataset. We report the reconstruction error for the encoder-decoder model for **different identifier pool sizes**. The error bars show the standard deviation across five runs with different random ID assignments.

https://doi.org/10.52202/085713-4684

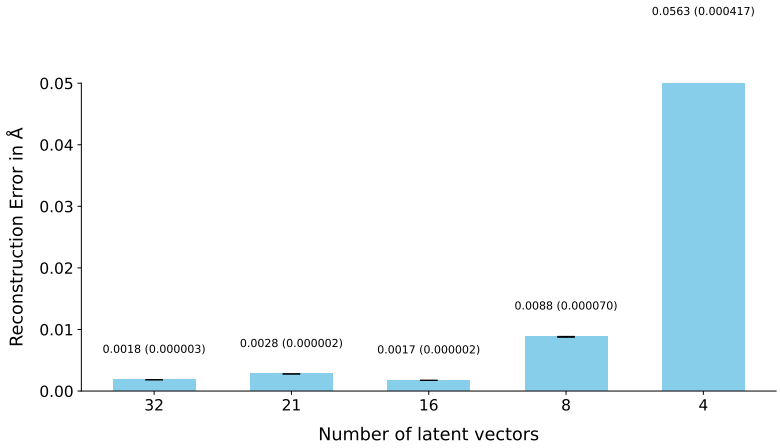

Figure 8: Reconstruction error in Å for the MD17 dataset. We report the reconstruction error for the encoder-decoder model for **different number of latent vectors** L. The error bars show the standard deviation across five runs with different random ID assignments.

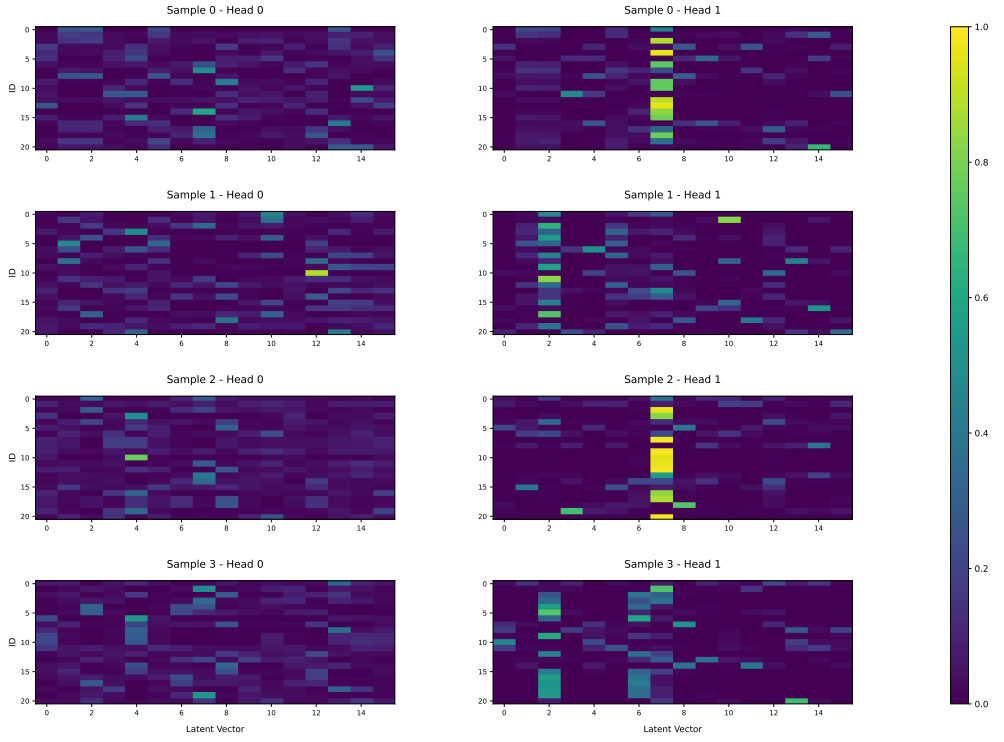

Figure 9: Attention patterns on the MD17 dataset for four random samples of the validation dataset, using a model trained with **16 latent vectors**. Each row corresponds to a single sample and displays the attention patterns across individual attention heads. Each sample employs a **distinct identifier assignment**.

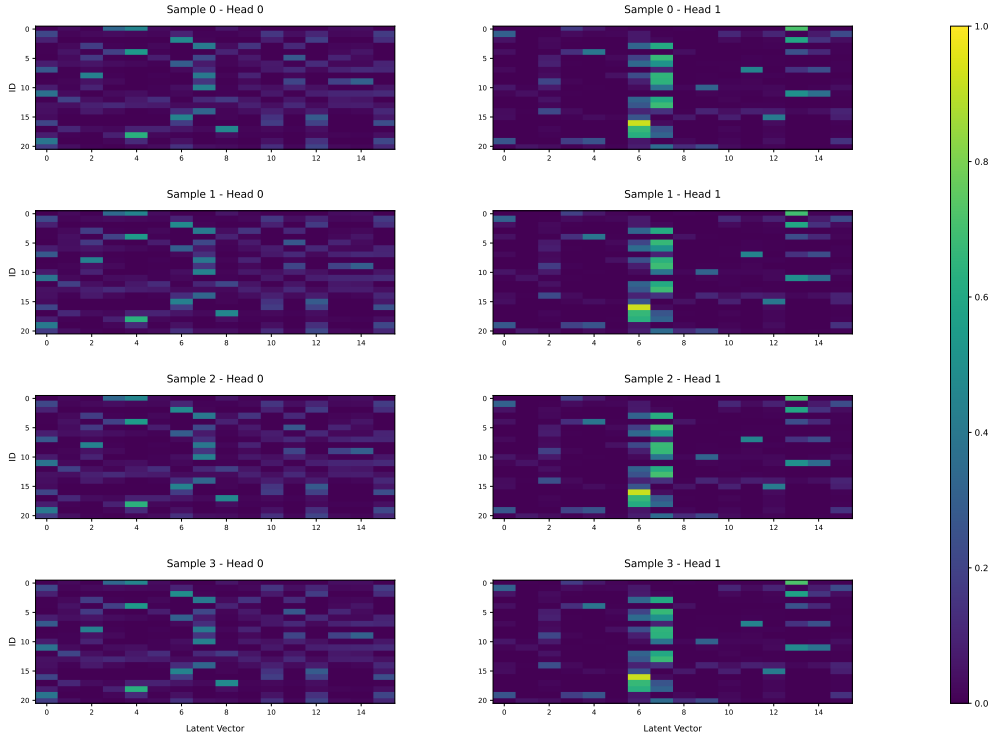

Figure 10: Attention patterns on the MD17 dataset for four random samples of the validation dataset, using a model trained with **16 latent vectors**. Each row corresponds to a single sample and displays the attention patterns across individual attention heads. Each sample employs the **same identifier assignment**.

https://doi.org/10.52202/085713-4684

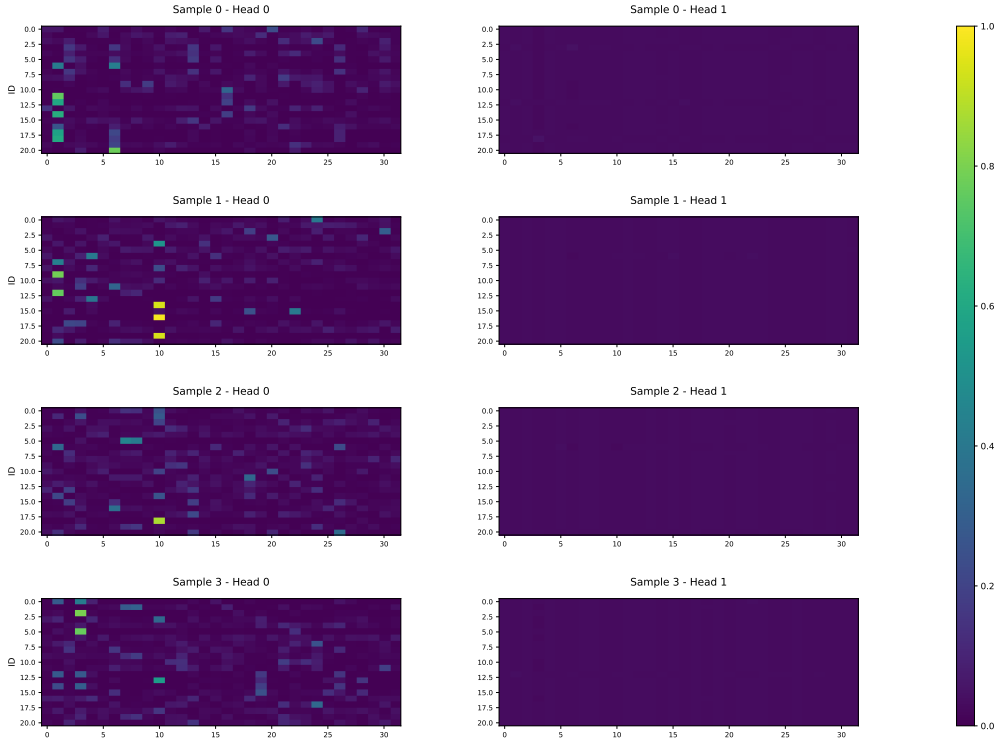

Figure 11: Attention patterns on the MD17 dataset for four random samples of the validation dataset, using a model trained with **32 latent vectors**. Each row corresponds to a single sample and displays the attention patterns across individual attention heads. Each sample employs a **distinct identifier assignment**.

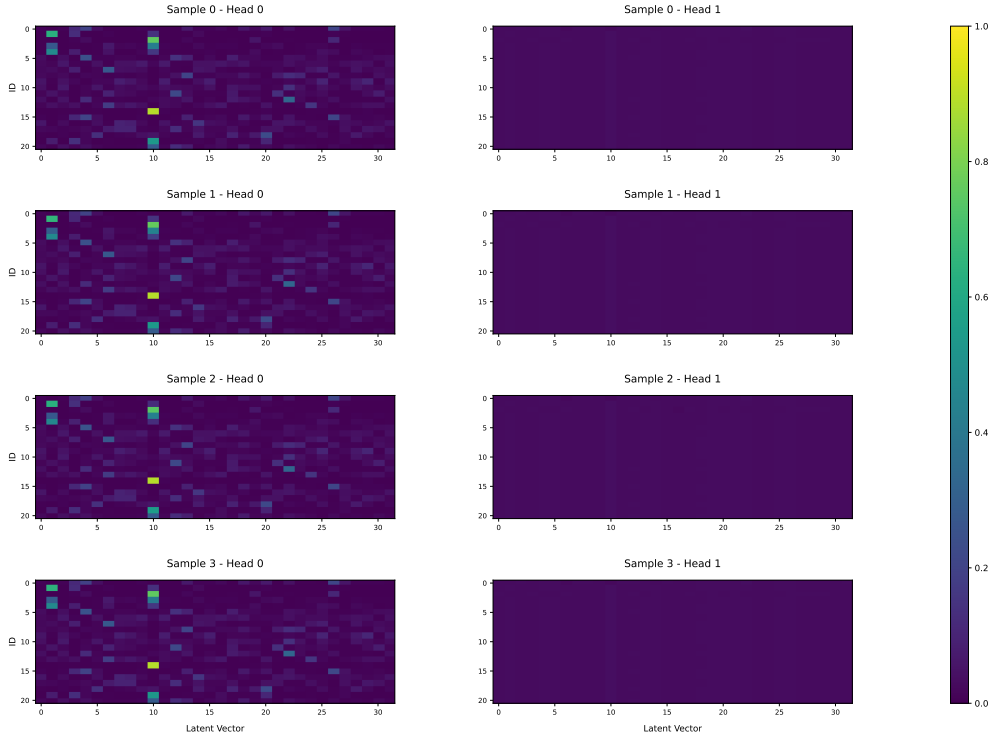

Figure 12: Attention patterns on the MD17 dataset for four random samples of the validation dataset, using a model trained with **16 latent vectors**. Each row corresponds to a single sample and displays the attention patterns across individual attention heads. Each sample employs the **same identifier assignment**.

https://doi.org/10.52202/085713-4684

H Visualizations

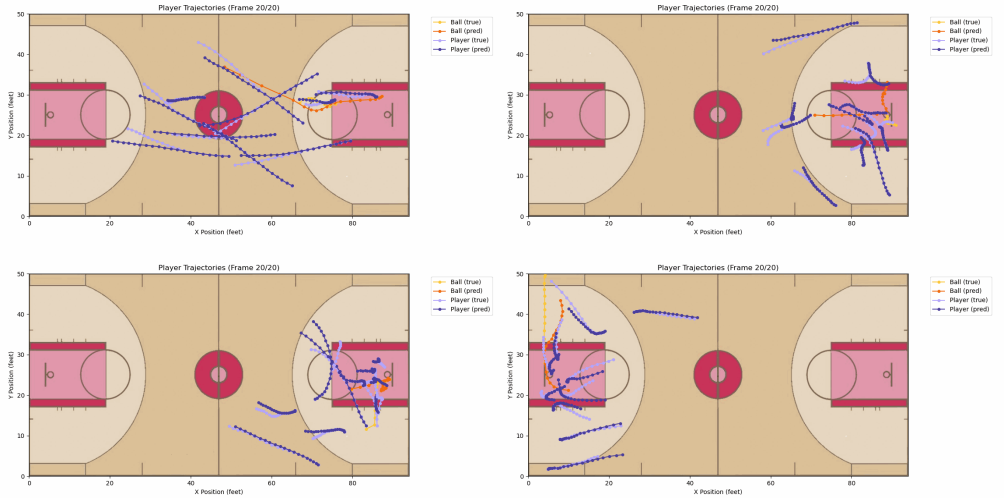

Figure 13: Basketball player trajectories from the NBA dataset. We visualize each player's movement path up to the final frame and include the ball's trajectory in the visualization. **Top row**: Trajectories for scoring scenarios. **Bottom row**: Trajectories for rebound scenarios. (Court image source: `https://github.com/linouk23/NBA-Player-Movements/blob/master/court.png`)

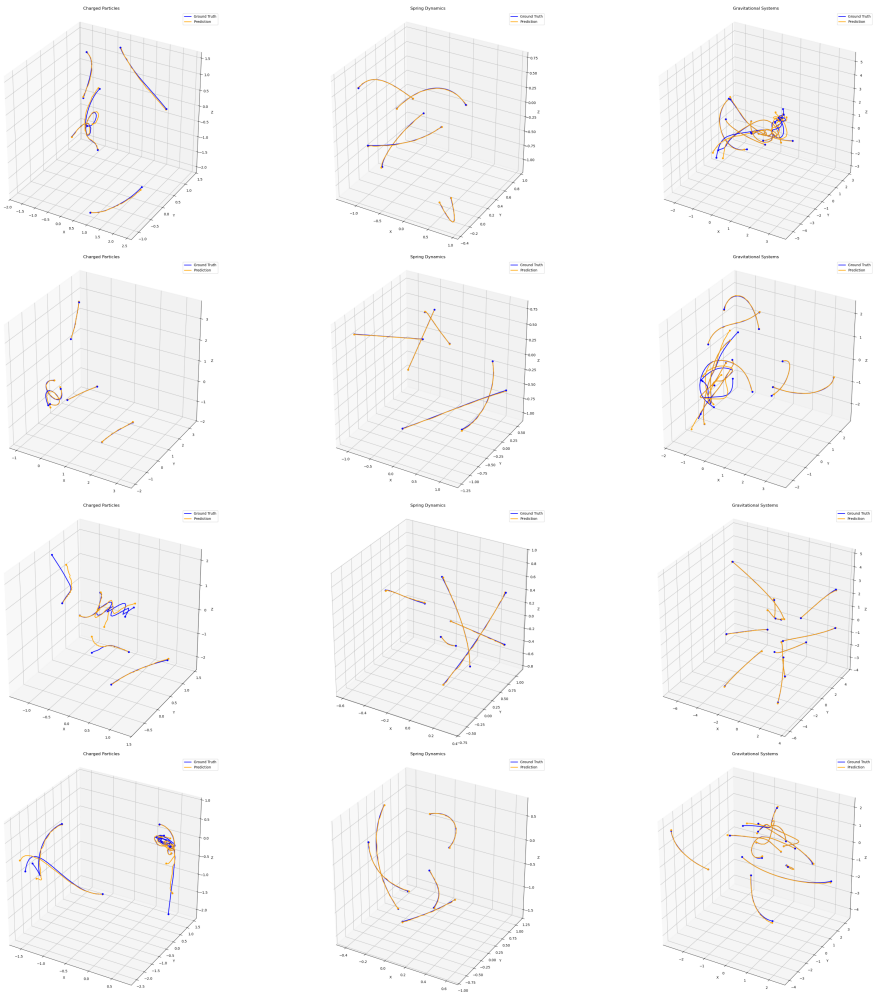

Figure 14: Trajectories from the N-Body dataset, predicted vs ground truth trajectories. **Left**: Charged particles. **Middle**: Spring dynamics. **Right**: Gravitational system.

https://doi.org/10.52202/085713-4684

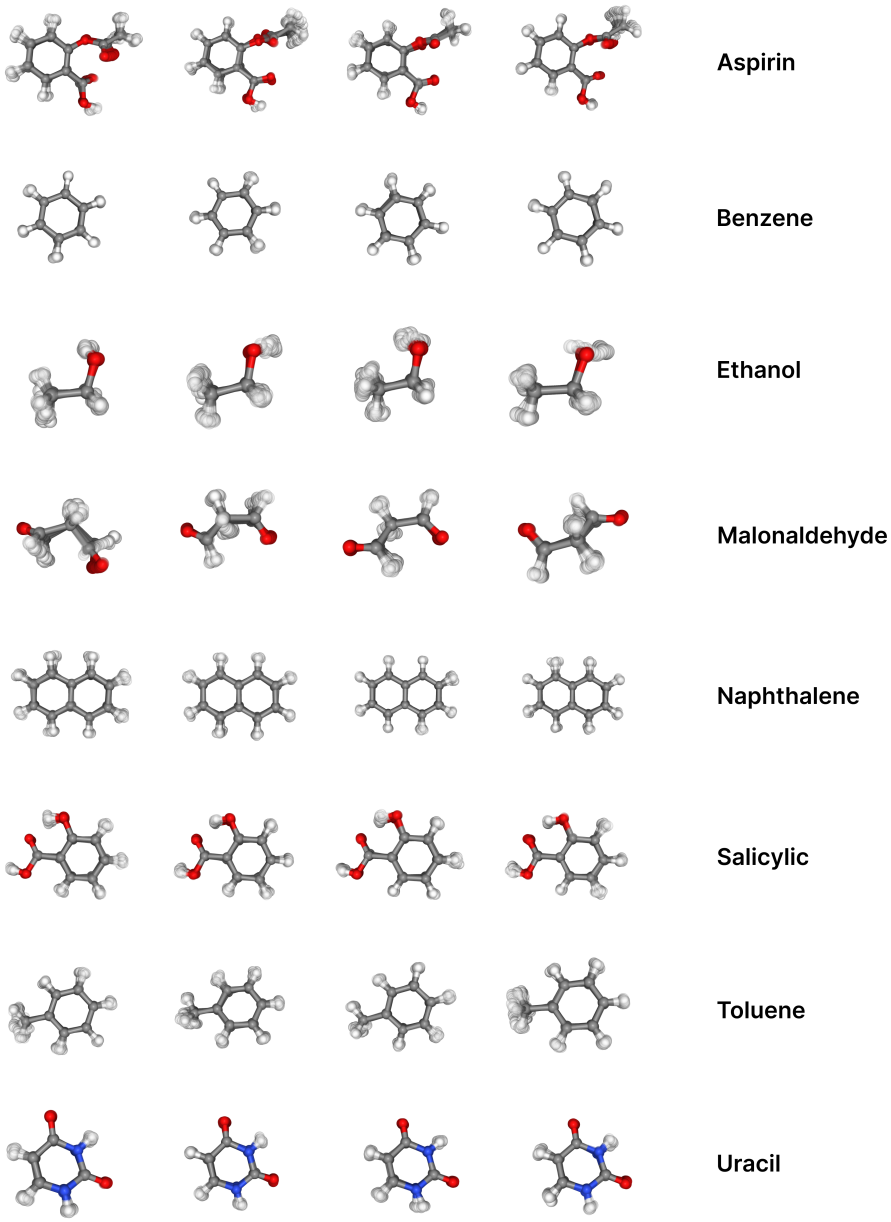

Figure 15: **Molecular dynamics trajectories from the MD17 dataset**, showing time-evolved structural predictions for each molecule. For every compound, we display four distinct trajectory predictions, with each prediction comprising 20 superimposed time frames to illustrate the range of conformational changes.

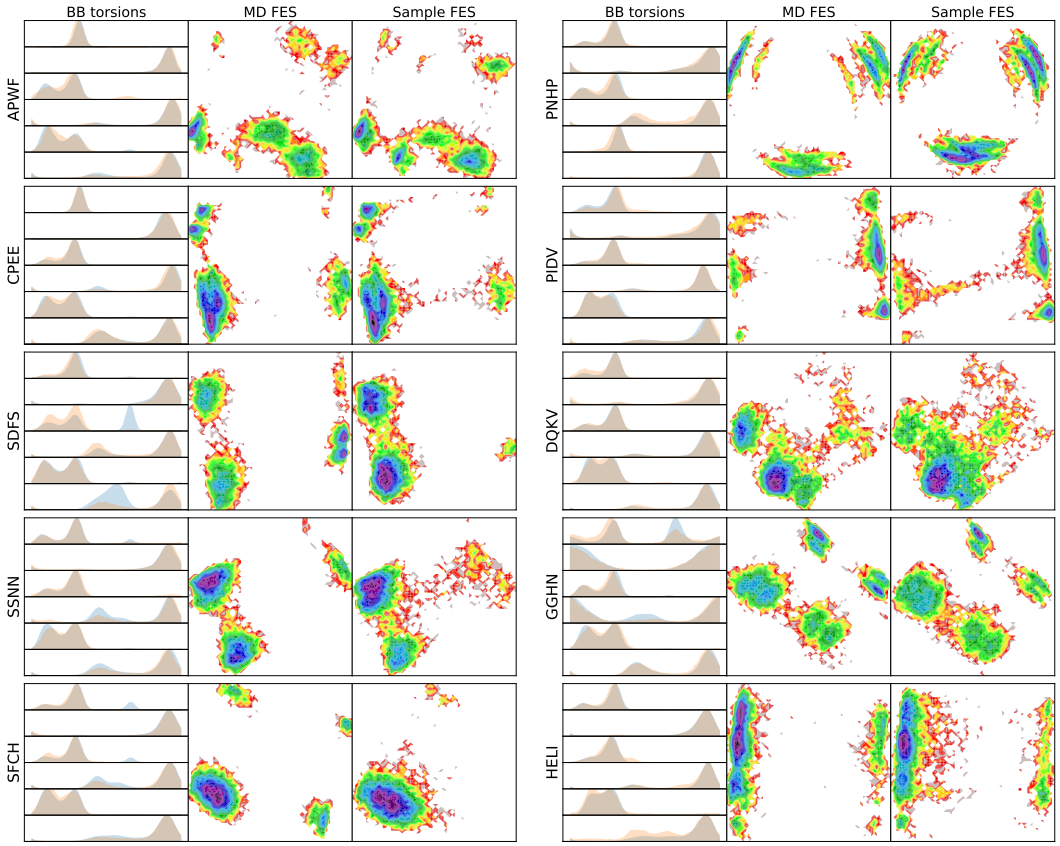

Figure 16: **Torsion angle distributions** of the six backbone torsion angles, comparing molecular dynamics (MD) trajectories (orange) and sampled trajectories (blue); and **Free energy surfaces** projected onto the top two time-lagged independent component analysis (TICA) components, computed from both backbone and sidechain torsion angles.

https://doi.org/10.52202/085713-4684

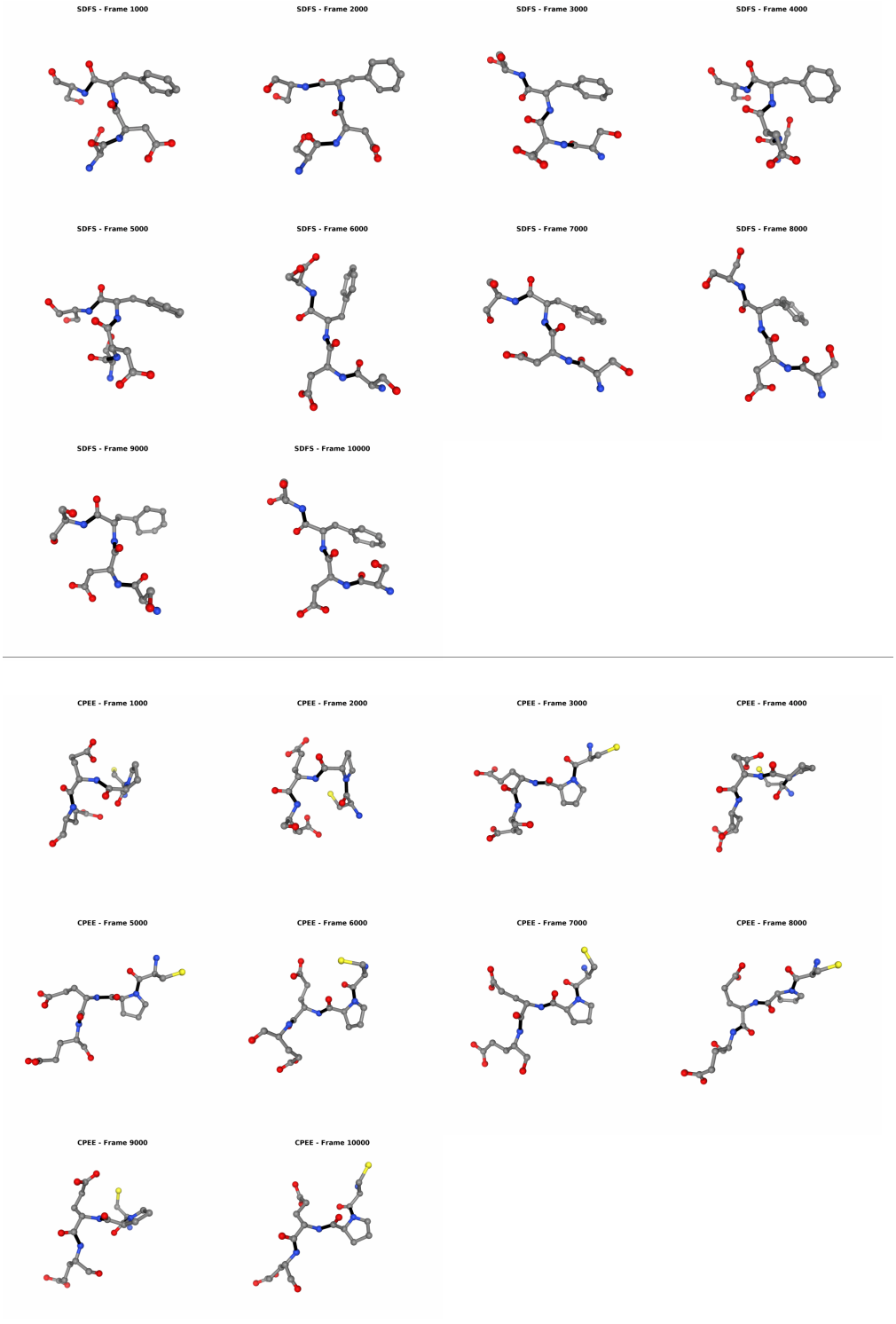

Figure 17: **Molecular Dynamics trajectories from the Tetrapeptides (4AA) dataset**, showing time-evolved structural predictions for ten frames at an interval of 1000 frames. **Top**: SDFS (Serine - Aspartic Acid - Phenylalanine - Serine) peptide. **Bottom**: CPEE peptide (Cysteine - Proline - Glutamic Acid - Glutamic Acid).

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: Our main contributions are declared in the abstract, and we summarize the contributions in the introduction.

   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: We list the limitations in the discussion section and include an additional section in the appendix investigating model performance with respect to parameter size and inference performance.

   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?

https://doi.org/10.52202/085713-4684

Answer: [Yes]

Justification: While our paper's primary contribution is architectural rather than theoretical, we provide a proof establishing the necessary conditions for our identifier assignment.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide pseudocode and hyperparameter tables for our model in the appendix section of our paper.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide full access to the source code to facilitate the production of experimental results. All used datasets have open access.

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 âĂIJNoâĂİ 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: All chosen hyperparameters are discussed in the appendix section Experimental Details, and the splits for the dataset are provided in the provided source code.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: We omit error bars for model parameters due to our extensive experimentation across diverse domains, while reporting metrics averaged across multiple sampling runs for selected experiments.

   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.

https://doi.org/10.52202/085713-4684

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

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: We report aggregate computational resources for all experiments; we do not report the resources for each individual experiment.

   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: The research conforms in every aspect to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [No]

    Justification: We do not anticipate direct negative social impact from our work, though it can serve as a foundational model for dynamical systems research.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We provide the source for all used datasets as URLs in the dataset section of the appendix.

    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.

https://doi.org/10.52202/085713-4684

- 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: The paper does not release new assets.

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

    Guidelines:

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

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

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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: In our publication, LLM usage is not considered as an important part.

    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.

https://doi.org/10.52202/085713-4684