THE JOURNAL OF PHYSICAL CHEMISTRY A
A JOURNAL OF THE AMERICAN CHEMICAL SOCIETY

pubs.acs.org/JPCA

Article

Tutorial on Molecular Latent Space Simulators (LSSs): Spatially and Temporally Continuous Data-Driven Surrogate Dynamical Models of Molecular Systems

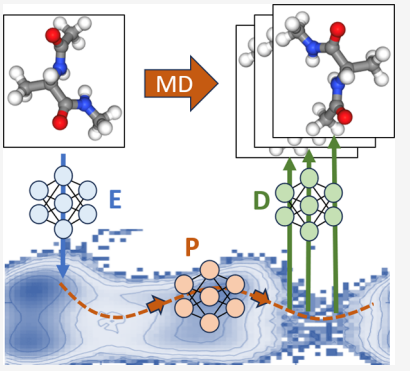
Michael S. Jones, Kirill Shmilovich, and Andrew L. Ferguson*

Cite This: *J. Phys. Chem. A* 2024, 128, 10299−10317

Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: The inherently serial nature and requirement for short integration time steps in the numerical integration of molecular dynamics (MD) calculations place strong limitations on the accessible simulation time scales and statistical uncertainties in sampling slowly relaxing dynamical modes and rare events. Molecular latent space simulators (LSSs) are a data-driven approach to learning a surrogate dynamical model of the molecular system from modest MD training trajectories that can generate synthetic trajectories at a fraction of the computational cost. The training data may comprise single long trajectories or multiple short, discontinuous trajectories collected over, for example, distributed computing resources. Provided the training data provide sufficient sampling of the relevant thermodynamic states and dynamical transitions to robustly learn the underlying microscopic propagator, an LSS furnishes a global model of the dynamics capable of producing temporally and spatially continuous molecular trajectories. Trained LSS models have produced simulation trajectories at up to 6 orders of magnitude lower cost than standard MD to enable dense sampling of molecular phase space and large reduction of the statistical errors in structural, thermodynamic, and kinetic observables. The LSS employs three deep learning architectures to solve three independent learning problems over the training data: (i) an encoding of the high-dimensional MD into a low-dimensional slow latent space using state-free reversible VAMPnets (SRVs), (ii) a propagator of the microscopic dynamics within the low-dimensional latent space using mixture density networks (MDNs), and (iii) a generative decoding of the low-dimensional latent coordinates back to the original high-dimensional molecular configuration space using conditional Wasserstein generative adversarial networks (cWGANs) or denoising diffusion probability models (DDPMs). In this software tutorial, we introduce the mathematical and numerical background and theory of LSS and present example applications of a user-friendly Python package software implementation to alanine dipeptide and a 28-residue beta−beta−alpha (BBA) protein within simple Google Colab notebooks.

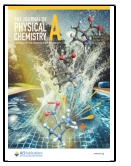


1. BACKGROUND AND THEORY

Classical molecular dynamics (MD) simulations model the microscopic dynamical evolution of atomistic and molecular systems by integrating Newton's equations of motion.[1] Modern high-performance computing hardware and efficient parallel software implementations have expanded the length scales accessible to all-atom simulations to tens of trillions of atoms.[2] However, the inherently serial nature of numerical integration together with the femtosecond time steps required to capture the fastest microscopic motions has limited accessible time scales to milliseconds.[3] This limitation, often termed the "time scale barrier" or "sampling challenge" in molecular simulation, places strong restrictions on the simulation of rare events and slowly relaxing dynamical processes by unbiased MD calculations. The inability to (densely) sample rare but important configurational states and/or dynamical transitions makes simulation trajectories subject to large statistical uncertainties in structural, thermodynamic, and kinetic observables.[4]

A number of strategies have been developed to engage this sampling challenge. *Coarse graining* sacrifices atomistic resolution in service of computational efficiency by developing simplified higher-level models that integrate atomistic degrees of freedom, typically by lumping multiple atoms together into coarse-grained beads.[5−7] A variety of highly successful coarse-grained models have been developed for biological[8] and condensed matter[9] systems and protocols developed to parametrize coarse-grained models from both bottom-up (i.e., fitting to all-atom data) and/or top-down (i.e., fitting to experimental observables) perspectives.[5] Although good strategies exist to ensure thermodynamic

Received: August 9, 2024
Revised: October 12, 2024
Accepted: October 21, 2024
Published: November 14, 2024



https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

consistency between all-atom and coarse-grained models, preservation of dynamical consistency has proven a more difficult challenge such that the kinetics of coarse-grained models may be artificially accelerated in an uncontrolled fashion relative to the all-atom systems.[10] *Enhanced sampling techniques* present an alternative strategy to accelerate sampling via collective variable biasing,[11,12] tempering,[13−15] or path-based techniques.[16−19] Collective variable biasing approaches apply artificial potentials to accelerate barrier crossing and improve sampling. The application of a posteriori analytical corrections can recover unbiased thermodynamic averages, but, except in special cases,[20−24] it is not generally possible to recover unbiased dynamical averages. Tempering approaches combine various thermodynamic ensembles within a single expanded ensemble to achieve improved sampling at the thermodynamic state of interest. In general, this involves exchanges between thermodynamic states that produce short, discontinuous trajectory segments as opposed to a single, long, temporally, and spatially continuous trajectory. Path-based techniques employ a variety of approaches to generate transition pathways between predefined reactant and product states but are generally limited in applicability to single-barrier transitions as opposed to global sampling of the thermally accessible phase space. *Boltzmann generators* represent a relatively new sampling paradigm wherein a normalizing flow is trained to transform a simple easy-to-sample Gaussian prior distribution into a target Boltzmann distribution over molecular states.[25,26] Once trained, the model can be used to efficiently sample from the Boltzmann distribution and compute free energy profiles but is not designed to generate unbiased dynamical trajectories for the estimation of kinetic or path-based observables. *Markov state models* (MSMs) exploit a separation of time scales to model the long-time system dynamics as probabilistic and memoryless jumps between discrete states within which the dynamical relaxations are fast.[27,28] The rate constants for the dynamic jumps between pairs of states are estimated from MD trajectories. The inherently localized nature of these jumps between kinetically linked states imbues MSMs with the very attractive feature that they do not require a single long trajectory against which to fit the transition rates but may be parametrized by a series of short, discontinuous trajectories that sufficiently densely sample the relevant states and transitions and which may be independently generated on distributed computing resources. MSMs can be viewed as a dynamical coarse graining of the configurational phase space into a set of discrete dynamical states amenable to a divide-and-conquer parametrization of interstate transition rates.

Molecular latent space simulators (LSSs) were introduced in 2020 as a means to learn a data-driven surrogate model of the dynamics of a molecular system and generate spatially and temporally continuous unbiased trajectories at a fraction of the cost of standard MD.[29,30] Mathematically, MD may be viewed as an algorithm to propagate the state of a molecular system $\mathbf{x}_t$ at time t to a configuration $\mathbf{x}_{t+\tau}$ at time $(t + \tau)$ via a set of transition density elements $\mathbf{x}_{t+\tau} \sim p_\tau(\mathbf{x}_{t+\tau}|\mathbf{x}_t)$.[31,32] For deterministic dynamics and $\mathbf{x}$ containing the full-dimensional state of the system (i.e., all particle coordinates and momenta), the transition density element $p_\tau(\mathbf{x}_{t+\tau}|\mathbf{x}_t)$ is a Dirac delta function lying on the single deterministic configuration $\mathbf{x}_{t+\tau}$ to which the dynamics evolve at time $(t + \tau)$. If the state vector contains a reduced representation of the system (e.g., only the configurational variables) or the dynamical evolution is stochastic (e.g., the temperature is maintained by a stochastic thermostat), then the transition density elements become distributions. In either case, the transition density elements are computed on-the-fly by accumulating the forces on all of the particles and numerically integrating Newton's equations of motion. The motivation for the LSS approach is to learn an efficient surrogate model for the microscopic transition density elements that can be evaluated at a vastly lower computational cost than in MD.

Attempts to directly learn the transition density elements $p_\tau(\mathbf{x}_{t+\tau}|\mathbf{x}_t)$ have been reported for small systems and have met moderate success,[33] but the curse of dimensionality makes learning of high-dimensional distributions for large systems computationally intractable. The fundamental premise of the LSS approach is to learn a dynamical encoding E: $\mathbf{x}_t \rightarrow \boldsymbol{\psi}_t$ of a molecular system into a latent space spanned by its leading slow modes, learn a surrogate model P: $\boldsymbol{\psi}_t \rightarrow \boldsymbol{\psi}_{t+\tau}$ to propagate the dynamics autonomously within this low-dimensional slow subspace, and learn a decoding D: $\boldsymbol{\psi}_{t+\tau} \rightarrow \mathbf{x}_{t+\tau}$ to generate molecular configurations from the dynamical trajectory generated within the latent space. Crucially, molecular systems generically exhibit a separation of time scales (i.e., a spectral gap) arising from cooperative couplings between the constituent atomic degrees of freedom.[11,34−38] This engenders an emergent low dimensionality of the long-time dynamical evolution within a slow subspace spanned by the leading maximally autocorrelated dynamical modes $\boldsymbol{\psi}$ that are kinetically decoupled from the fast degrees of freedom residing beyond the spectral gap.[11,39−41] If this slow subspace can be discovered and is sufficiently low-dimensional ($\lesssim$10), then learning of the transition density elements $p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$ becomes a computationally tractable low-dimensional learning problem and decoding back to the molecular space $\boldsymbol{\psi}_{t+\tau} \rightarrow \mathbf{x}_{t+\tau}$ becomes a well-posed conditional generation problem that requires learning to sample from the annealed distribution of fast degrees of freedom conditioned on the slow variable state $\mathbf{x}_{t+\tau} \sim p(\mathbf{x}_{t+\tau}|\boldsymbol{\psi}_{t+\tau})$.

Mathematically, the LSS can be expressed as an alternative pathway from $\mathbf{x}_t$ to $\mathbf{x}_{t+\tau}$,[29,31,32]

$$\begin{array}{ccc} \mathbf{x}_t & \underset{E}{\rightarrow} & \boldsymbol{\psi}_t \\ \mathrm{MD}\downarrow & & \downarrow P \\ \mathbf{x}_{t+\tau} & \underset{D}{\leftarrow} & \boldsymbol{\psi}_{t+\tau} \end{array} \tag{1}$$

where the encoder E, propagator P, and decoder D can be learned from training data and, once trained, enable $\mathbf{x}_{t+\tau}$ to be generated at a fraction of the cost of MD. Framed in this manner, E, P, and D can be treated as three separate learning problems that can be trained using the same MD training data. Once E is trained to furnish a dynamically meaningful latent space, both P and D can be trained independently using these latent coordinates as inputs. Furthermore, deep neural network architectures ideally suited to each of these learning problems have been previously developed and can be modularly deployed to serve as the three constituent components of the LSS.[29,30] A pedagogical introduction to the LSS approach and demonstration of a user-friendly Python package implementing the LSS pipeline (https://

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

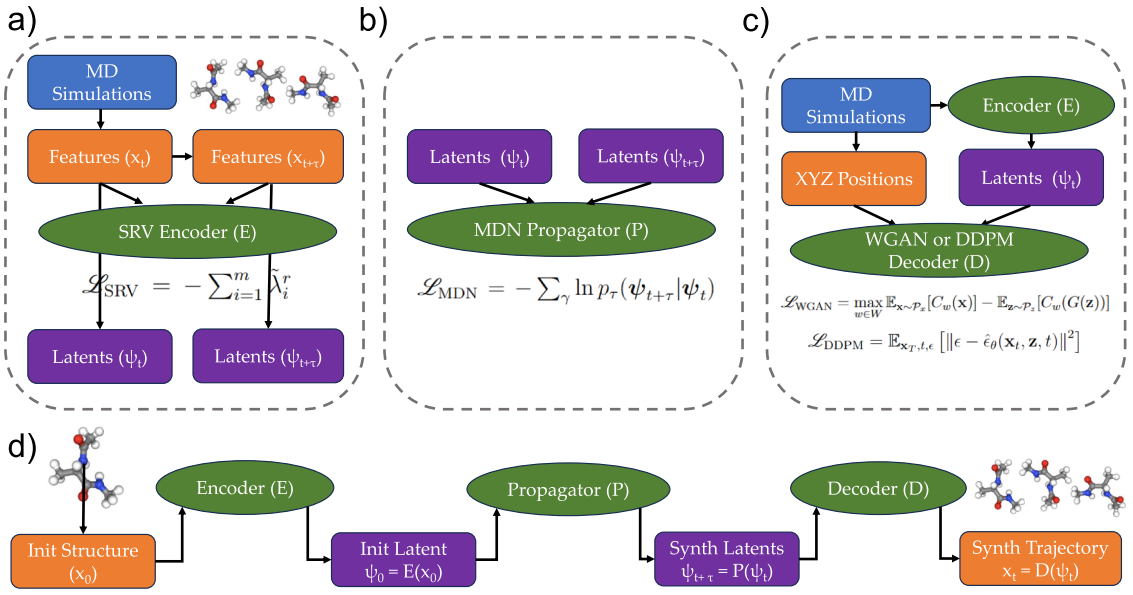


Figure 1. Overview of the molecular LSS approach. (a) SRV encoder E is trained on MD training data to learn a low-dimensional embedding into a latent space spanned by the maximally autocorrelated (i.e., slowest relaxing) dynamical modes $\boldsymbol{\psi} = E(\mathbf{x})$. (b) MDN propagator P is trained on time-lagged snapshots of the MD training data projected into the latent space to learn transition density elements within the latent space that define the latent space propagator $\boldsymbol{\psi}_{t+\tau} = P(\boldsymbol{\psi}_t) \sim p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$. (c) Decoder D is trained to generate a realization of the molecular configuration conditioned on latent space coordinates $\hat{\mathbf{x}}_t = D(\boldsymbol{\psi}_t) \sim p(\hat{\mathbf{x}}_t|\boldsymbol{\psi}_t)$. The decoding operation produces molecular configurations consistent with the slow degrees of freedom encoded by the latent space coordinate with a realization (i.e., in-painting) of the fast degrees of freedom consistent with the learned distribution of molecular configurations at that latent space embedding. We have implemented decoders based on cWGANs and DDPMs. (d) The final LSS model comprising the trained encoder E, propagator P, and decoder D can be deployed to generate novel synthetic molecular simulation trajectories consistent with the learned microscopic dynamics of the slow modes at approximately 6 orders of magnitude (i.e., 1 million fold) faster than standard MD calculations.

github.com/Ferg-Lab/LSS) is the subject of the present tutorial.

1.1. Strengths and Limitations. LSSs share many similarities with MSMs, and it can be instructive to draw a comparison between these approaches to illuminate similarities and differences and strengths and limitations within the context of the more established and familiar MSM formalism. The underlying principle of LSSs is similar to MSMs, but whereas MSMs learn a discrete partitioning of the configurational phase space and the transition rates of a jump process between these states, LSSs learn a slow subspace within which the long-time dynamics evolve and a continuous-time effective dynamical model within this space. In both cases, a separation of time scales motivates and enables parametrization of a dynamical model in a set of slow collective variables governing the long-time dynamical evolution of the system and discarding of the quickly relaxing fast degrees of freedom that are effectively equilibrated to the slow variables. In MSMs, the fast variables rapidly relax within the discrete coarse-grained states that contain dynamically distinct configurations of the slow collective variables. In LSSs, the dynamical coarse-graining procedure is more analogous to the Born−Oppenheimer approximation, wherein the fast electronic degrees of freedom are annealed to the slow nuclear dynamics,[42] and embodies the Mori−Zwanzig projection operator formalism, wherein the effective dynamics of a dynamical system can be written in a subspace of slowly evolving collective variables to which the remaining degrees of freedom couple as noise.[43−46]

Like MSMs, LSSs can also be parametrized by short, discontinuous training trajectories generated by distributed computing, provided that these trajectories sufficiently densely sample the relevant states and transitions in the molecular phase space to enable learning of the underlying microscopic dynamics. Unlike MSMs, LSSs do not induce a discretization of the configurational phase space and can, therefore, furnish spatially and temporally continuous trajectories in the slow collective variables. Moreover, the LSS is generative in the sense that trajectories in the slow subspace can be decoded or backmapped to synthetic MD trajectories using a trained generative model to in-paint the fast degrees of freedom conditioned on the state of the slow collective variables.

Finally, LSSs and MSMs are similar in that they learn the underlying probabilistic and memoryless transition density elements of the microscopic dynamics from the training data. The learned dynamical model can then be used to inexpensively generate temporally continuous—and, in the case of LSSs, spatially continuous in the slow collective variables—synthetic trajectories over the learned phase space. Since these trajectories are generated stochastically from the learned transition density elements, the generated trajectories are not simply carbon copies of the training trajectories but are rather novel trajectories through phase space based on the microscopic transition density elements learned from the data. Synthetic trajectories can be generated at a fraction of the cost of MD simulations and may therefore be used to densely sample states and events that may have only appeared a few times within the training set. Subject to the quality of the LSS model learned from the data, inexpensive synthetic trajectories may be used to greatly reduce statistical uncertainties in structural, thermodynamic, and kinetic observables. By the same token, LSSs are, like MSMs, fundamentally data-driven models. As such, they are only as

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

good as their training data: they cannot be expected to accurately parametrize states or transitions that are not well represented within the training data, and, while some modest extrapolation into new regions of phase space can be anticipated, they are not generally expected to prospectively discover novel states or transitions. Further, the surrogate dynamical model learned from the data pertains only to that particular molecular system under those particular thermodynamic conditions. As such, in the absence of physical inductive biases, the trained model is not necessarily expected to be transferable to other molecules or other thermodynamic conditions. Finally, the trained models are, by construction, subject to the same systematic errors that may be present in the molecular force fields used to furnish the training data.

We note that a few other methods sharing similarities with the LSS formalism have recently been proposed.[47−51] Of these, the LSS approach is perhaps most closely related to learning of effective dynamics (LED),[50] which uses an autoencoding neural network to learn a latent space representation and employs a recurrent structure to add long-term memory to the propagator.

1.2. Components of the LSS. LSSs employ three deep learning architectures to solve three learning problems over the training data: (i) an encoding of the high-dimensional MD into a low-dimensional slow latent space using state-free reversible VAMPnets (SRVs), (ii) a propagator of the microscopic dynamics within the low-dimensional latent space using mixture density networks (MDNs), and (iii) a generative decoding of the low-dimensional latent coordinates back to the original high-dimensional molecular configuration space using conditional Wasserstein generative adversarial networks (cWGANs) or denoising diffusion probability models (DDPMs) (Figure 1). The three learning problems are independent and can be trained over the same MD training set. Full details of these three components have previously been reported elsewhere, and the specific architectures and hyperparameter choices depend on the target molecular system,[29,30] but we provide an overview herein of the mathematical and algorithmic underpinnings.

1.2.1. Encoder, E: State-Free Reversible VAMPnets (SRVs). State-free reversible VAMPnets (SRVs) were introduced in 2019 as a deep neural network approach to perform data-driven nonlinear unsupervised learning of the slowest evolving (i.e., maximally autocorrelated) collective variables in dynamical systems.[39] The name of the approach stems from its algorithmic kinship with variational approach for Markov processes networks (VAMPnets) introduced by Noé and co-workers[52] and can also be considered as a deep variant of time-lagged independent component analysis (Deep-TICA)[53−56] or deep canonical correlation analysis (DCCA).[57] The mathematical basis of SRVs rests upon the transfer operator $\mathcal{T}$ as the mathematical object that propagates the probability distribution over the microstates of a dynamical system as a function of time according to the transition density elements $p_\tau(\mathbf{x}_{t+\tau}|\mathbf{x}_t)$, where τ is the time increment between successive states.[58,59] Molecular systems at equilibrium obey a detailed balance. This induces the transfer operator $\mathcal{T}$ to become self-adjoint with respect to the equilibrium distribution and admit a spectrum of eigenfunctions $\{\boldsymbol{\psi}_i(\mathbf{x})\}$ with real eigenvalues $1 = \lambda_0 > \lambda_1 \geq \lambda_2 \geq \cdots$.[29,31,39,40,58,60] The leading eigenfunction corresponds to the equilibrium distribution over microstates and the higher-order eigenfunctions form a natural basis of increasingly more quickly relaxing deviations from the equilibrium distribution with associated relaxation times of $t_i = -\tau/\ln \lambda_i$.[31,39,61] For sufficiently long lag times τ, the transfer operator becomes effectively low-dimensional as manifested by a gap in the eigenvalue spectrum separating the leading eigenfunctions spanning a slow subspace from the rapidly relaxing higher-order eigenfunctions.[31] This emergent low dimensionality is what makes the entire LSS approach possible since it becomes numerically tractable to learn a low-dimensional surrogate dynamical model from limited MD trajectory data.

The variational approach to conformational dynamics (VAC) provides a route to optimally approximate the transfer operator eigenfunctions within a finite dimensional basis expansion, $\tilde{\boldsymbol{\psi}}_i(\mathbf{x}) = \sum_j s_{ij}\zeta_j(\mathbf{x})$.[39−41] Given a particular choice of basis functions over the microstates $\{\zeta_j(\mathbf{x})\}$, the VAC prescribes the optimal expansion coefficients $\{s_{ij}\}$ via solution of the generalized eigenvalue equation $\mathbf{C}\mathbf{s}_i = \tilde{\lambda}_i\mathbf{Q}\mathbf{s}_i$, where $\mathbf{C}$ is the time-lagged correlation matrix of the basis functions $\{\zeta_j(\mathbf{x})\}$ under a time lag of τ, $\mathbf{Q}$ is the instantaneous correlation matrix of the basis functions $\{\zeta_j(\mathbf{x})\}$, $\mathbf{s}_i$ is the (eigen)vector of linear expansion coefficients for the approximate eigenfunction $\tilde{\boldsymbol{\psi}}_i(\mathbf{x})$, and $\tilde{\lambda}_i$ is the corresponding approximate eigenvalue with an associated implied relaxation time scale of $\tilde{t}_i = -\tau/\ln \tilde{\lambda}_i$.[31,39] In practice, $\mathbf{C}$ and $\mathbf{Q}$ are numerically estimated from training data comprising simulation trajectories of the molecular system, and we solve the generalized eigenvalue problem by standard techniques. It is useful to observe that the mathematical underpinnings of the VAC are isomorphic to the Roothaan−Hall equations in quantum mechanics where one can identify $\mathbf{C}$ as the Fock matrix and $\mathbf{Q}$ as the overlap matrix: instead of finding the lowest energy wave function of the Hamiltonian operator within a defined basis, we are finding the slowest eigenfunction of the transfer operator within a defined basis.[31,41,42] We also observe that an MSM is a special case of the VAC formalism when the basis functions are selected to be indicator functions over a discrete partitioning of configurational states.[31,58]

The VAC guarantees that the approximated eigenfunctions will be no slower than the true eigenfunctions, and, as a corollary, asserts that better choices of basis functions are those that result in slower eigenfunctions.[39−41] The fundamental idea underpinning SRVs is to use deep neural networks as flexible, nonlinear, and data-driven functional approximators to discover basis functions that result in the slowest possible approximations to the leading transfer operator eigenfunctions. As opposed to generic and preselected basis functions (e.g., pairwise distances, atom-centered symmetry functions), the networks learn basis functions tailored to each molecular system that can result in superior approximations to the true leading eigenfunctions of the transfer operator. In this sense, SRVs can simply be viewed as a VAC with a neural network bolted on at the front end whose task is to perform data-driven discovery of bespoke basis functions from MD training data. The networks are trained to learn optimal basis functions with which to construct linear expansions as approximations to the slowest m eigenfunctions by maximizing their implied time scales via the loss function $\mathcal{L}_{\mathrm{SRV}} = -\sum_{i=1}^{m} \tilde{\lambda}_i^r$, where we typically adopt $r = 2$, and an appropriate value of m is selected by resolving a gap in the eigenvalue spectrum

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

(Figure 1a). The learning task can be made end-to-end differentiable through the VAC. It is typically desirable to employ molecular featurizations or symmetrization operations that ensure the basis functions respect the underlying symmetries of the molecular system (e.g., translational invariance, rotational invariance, and permutational invariance).[62] In our applications to date, we have found that simple fully connected feedforward neural networks comprising a handful of layers containing a few hundred neurons per layer have proven adequate for very satisfactory performance.[29,30,39] Conceptually, the success of very simple neural networks can perhaps be understood by the fact that they are tasked only with finding good basis functions and that the mathematical heavy lifting is taken care of by the VAC.

For nonequilibrium systems that do not obey detailed balance, the VAC must be replaced by a more general variational principle termed the variational approach to Markov processes (VAMP).[31,52] This results in a more complex mathematical development that rests upon singular vectors and singular values and underpins the more general state-free nonreversible VAMPnets (SNRV) approach.[24] For multimolecular systems, we must contend with the combinatorial explosion in the state space resulting from the approximately independent nature of the dynamical evolution of each molecule when they are in noninteracting or weakly interacting regions of configurational space.[63,64] This challenge can be engaged by extending the LSS paradigm to contain multiple encoders for each independent and interacting subsystem, multiple propagators to evolve the dynamics of each subsystem and the relative locations of each subsystem, and multiple decoders for each independent and interacting subsystem.[30] In this tutorial, we shall restrict our focus to the equilibrium scenario where the VAC applies and to applications to single molecular systems.

A full description of the mathematical underpinnings of VAC and VAMP[31,52] and the numerical implementation of S(N)RVs[29,30,39] are available in prior publications, and an open-source and user-friendly Python package implementing S(N)RVs is freely available from https://github.com/Ferg-Lab/snrv.

1.2.2. Propagator, P: Mixture Density Networks (MDNs). The SRV furnishes a slow subspace spanned by learned approximations to the m leading transfer operator eigenfunctions $\{\psi_i(x)\}_{i=1}^{m}$ prior to the spectral gap. The separation of time scales delimited by the spectral gap means that at sufficiently long lag times τ, a surrogate model for the dynamical evolution of the system can be constructed within the slow latent subspace. This assumes rapid relaxation of the fast degrees of freedom contained in the eigenvectors beyond the spectral gap on sub-τ time scales, permitting the evolution of the slow variables to be accurately approximated as Markovian (i.e., memoryless) and, at equilibrium, stationary (i.e., time-invariant).[43−46] The effective dynamics within the slow subspace (i.e., approximations to the transition density elements $p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$) can be learned by projection of the MD training data through the trained SRV model. The low dimensionality of the slow subspace $m \ll 3N$ is induced by adopting a sufficiently long lag time, wherein only a small number of collective modes contribute to the long-time system evolution. This allows us to break the curse of dimensionality and define a tractable low-dimensional learning problem for a surrogate model of effective MD. Beyond the low dimensionality, the learning problem is also significantly simplified within the transfer operator eigenfunction basis since, by construction, these eigenfunctions diagonalize the transfer operator and make the dynamical evolution linear within the eigenfunction basis.[31]

Conceptually, the transition density elements in the latent space, $p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$, define the jump probabilities to any other location in the latent space after time τ given that the system currently exists at location $\boldsymbol{\psi}_t$. We recall that the loss of configurational degrees of freedom under projection and neglect of the velocity information mean that these transition density elements are typically distributions over microstates that reflect the stochastic nature of the effective dynamics in the slow subspace. Mixture density networks (MDNs) combine Gaussian mixture models with deep neural networks to efficiently approximate multimodal distributions,[65,66] and the LSS approach uses MDNs to learn the transition density elements $p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$ via linear combinations of C m-dimensional Gaussian kernels $\boldsymbol{\phi}_c$,

$$p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t) = \sum_{c=1}^{C} \alpha_c(\boldsymbol{\psi}_t)\boldsymbol{\phi}_c(\boldsymbol{\psi}_{t+\tau}; \boldsymbol{\mu}_c(\boldsymbol{\psi}_t), \boldsymbol{\sigma}_c(\boldsymbol{\psi}_t)) \tag{2}$$

During training, the MDN is trained to learn the $\boldsymbol{\psi}_t$-dependent means $\boldsymbol{\mu}_c$ and variances $\boldsymbol{\sigma}_c$ of the constituent Gaussians and $\boldsymbol{\psi}_t$-dependent linear mixing coefficients α_c from the projected MD training trajectories to minimize the loss function $\mathcal{L}_{\mathrm{MDN}} = -\sum_\gamma \ln p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$ over time-lagged pairs of training points (Figure 1b). Once trained, the MDN is then pressed into service as the latent space propagator P, by iteratively sampling from the learned $p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$ distributions to drive the dynamics of the system through the slow latent space in time increments of τ.

Importantly, the MDN propagator learns the microscopic transition density elements from the training data as a surrogate model of the effective latent space dynamics and then stochastically samples from them to generate novel trajectories. As such, the trajectories generated by MDN are novel in the sense that they obey the learned microscopic dynamics and are not just carbon copies of the MD training trajectories. On the other hand, the MDN is unlikely to be able to extrapolate far beyond the MD training data and thus is unlikely to spontaneously discover novel states or transitions not present in the training ensemble. Sampling from the MDN is extremely computationally efficient. Numerical benchmarks on the systems we have studied to date indicate that propagating the dynamical evolution of the system through the latent space is approximately 6 orders of magnitude (i.e., 1 million fold) faster than standard MD computations, primarily due to the fact that the learned MDN model does not require the expensive force calculations at each time step that are inherently required by standard MD.[29,30] Importantly, propagation of the system dynamics occurs in a time-invariant, autonomous manner completely within the slow latent subspace. The learned MDN surrogate model for the effective dynamics is expressed exclusively in $\boldsymbol{\psi}$, so there is no requirement to decode the system back up to the molecular space and re-encode back into the slow subspace at each time step in order to propagate the dynamics. This is a valuable attribute of the approach since decoding and encoding during each step of the propagator is computationally slow and can also result in the accumulation of destabilizing errors in the dynamical evolution of the system.[31,67]

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

In practice, we have found that simple MDN networks comprising two hidden layers of 100 neurons and approximately 50 Gaussian kernels have been sufficient for the systems we have studied to date.[29,30] An open-source and user-friendly Python package implementing MDNs is freely available from https://github.com/Ferg-Lab/mdn_propagator.

1.2.3. Decoder, D: Conditional Wasserstein GANs (cWGANs) and Denoising Diffusion Probabilistic Models (DDPMs). The trained MDN is used to efficiently generate synthetic trajectories within the slow latent space by sampling from the learned microscopic transition density elements at a fraction of the cost of standard MD. The final component of the LSS is a generative model to decode these latent space trajectories back into the molecular configuration space. Importantly, this decoding is passive in the sense that there is no requirement that it be done contemporaneously with the MDN trajectory generation or that every frame of the latent space trajectory must be decoded. By construction, the latent space trajectories describe the time evolution of the leading slow variables. The task of the decoder can be conceived of as mapping the slow variables back to molecular configurational space by, in a sense, inverting the operation of the SRV encoder. To do so requires restoration of the fast degrees of freedom omitted in the slow latent space which were assumed to be in quasi-equilibrium with the slow variables at each time step. Accordingly, we should expect the decoder to be able to generate a smooth, temporally continuous molecular trajectory in the slow collective variables since there is a bijective mapping between the molecular configurational state in these slow variables and each point in the latent space. Conversely, there are multiple realizations of the fast degrees of freedom associated with each point in the latent space, so the decoding operation is one-to-many. The trained decoder should therefore generate molecular configurations drawn from a distribution over the annealed, quasi-equilibrium distribution of fast degrees of freedom that is consistent with the state of the slow collective variables. In the limit of large training data volumes and good training of the decoder over a molecular system at equilibrium, the ensemble of fast degrees of freedom produced by the decoder is expected to approach the Boltzmann distribution.

Mathematically, the decoder is a conditional generative model tasked with learning to sample molecular configurations consistent with each possible state of the slow collective variables $\mathbf{x}_{t+\tau} \sim p(\mathbf{x}_{t+\tau}|\boldsymbol{\psi}_{t+\tau})$. Computationally, we have developed two alternative implementations to accomplish this generative decoding: cWGANs and DDPMs. While we have observed that both models achieve excellent reconstruction, we find DDPMs to be typically more stable to train and less sensitive to hyperparameters, whereas cWGANs tend to be more computationally efficient during inference. Given the overall better performance in adherence to conditioning variables and reconstruction quality, we recommend employing the DDPM unless decoding speed is of the highest priority.

The cWGAN comprises two components: a generator $G(\mathbf{z})$ that is tasked to output realizations of molecular configurations $\mathbf{x}$ from inputs $\mathbf{z}$ and a critic $C(\mathbf{x})$ that is tasked with evaluating the quality of a molecular configuration $\mathbf{x}$. The generator and critic are cotrained in an adversarial manner to minimize the Wasserstein (i.e., earth mover's) distance,

$$\mathcal{L}_{\mathrm{WGAN}} = \max_{w \in W} \mathbb{E}_{\mathbf{x}\sim\mathcal{P}_x}[C_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim\mathcal{P}_z}[C_w(G(\mathbf{z}))] \tag{3}$$

where $\mathcal{P}_x(\mathbf{x})$ is the distribution over molecular configurations sampled in the MD training data, $\mathcal{P}_z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1}) \in \mathbb{R}^d$ is d-dimensional Gaussian noise, and $\{C_w\}_{w \in W}$ is a family of K-Lipschitz functions enforced through a gradient penalty[68,69] (Figure 1c). Conditioning is introduced by additionally passing the latent space location $\boldsymbol{\psi}$ we wish to decode into a molecular configuration to both the generator and critic.[70] As such, the generator is driven by both white noise $\mathbf{z}$, which induces diversity into the generated ensemble of fast degrees of freedom, and a conditioning variable $\boldsymbol{\psi}$, which informs and restrains the state of the slow degrees of freedom encoded within the latent space. Once trained, the cWGAN critic is discarded and the generator serves as the LSS decoder. To date, we have typically been concerned with biomolecules in isotropic environments and have trained our cWGAN implementations over rotationally and translationally aligned configurations $\mathbf{x}$ from the MD training data. Alternatively, one could conceive of training the cWGAN to operate on internal molecular coordinates. For nonisotropic applications (e.g., interactions of molecules with surfaces), it may be desirable to preserve the center-of-mass translation and rotation within the decoding. For weakly coupled two-molecule systems, we trained cWGANs on each system independently oriented to their respective frames of reference.[30] So far, we have restricted our decoder to operate on biomolecular solutes and have not tasked it to also decode the coordinates of solvent molecules. To do so would require engaging the permutational invariance of the solvent molecules using brute force data augmentation or, as a more scalable and elegant solution, permutationally invariant descriptors of the solvent environment.[62,71−80]

We have also developed an alternative decoder that leverages a DDPM to conditionally denoise samples from an isotropic Gaussian distribution. The model is based on the pioneering work of Ho et al.[81] and the code is adapted from the work of Wang et al.[82] which used a DDPM to predict distributions of backbone dihedrals conditioned on temperature. Training consists of a forward noising process in which the Euclidean positions of MD configurations $\mathbf{x}_T$ are gradually converted into an isotropic Gaussian $\mathbf{x}_0$. During inference, this process is reversed and realistic samples are generated over T steps by gradually denoising intermediate samples $\mathbf{x}_t$ via predictions from a neural network which is exposed to the latent space coordinates $\mathbf{z}$. The forward diffusion process represents the conditional probability between subsequent steps $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and has a Gaussian form represented by $\mathcal{N}(\sqrt{1-\beta_t}\,\mathbf{x}_{i-1}, \beta_t\mathbf{I})$, where β_t is defined by a variance schedule and gradually ascends as a function of diffusion time t. An intermediate sample $\mathbf{x}_t$ can be computed directly from $\mathbf{x}_T$ and t by $\mathbf{x}_t = \sqrt{\alpha_0}\mathbf{x}_T + \sqrt{1-\alpha_0}\epsilon$, where $\alpha_t = \prod_{j=t}^{j=1}(1-\beta_j)$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. This direct sampling trick is crucial, as it allows an arbitrary intermediate configuration to be retrieved during training. We use a neural network with a one-dimensional U-net architecture to make a prediction of the noise $\hat{\epsilon}$ as a function of the intermediate configuration, diffusion time step, and latent space coordinates and regress this prediction against the actual noise that is deposited (Figure 1c),

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

$$\mathcal{L}_{\mathrm{DDPM}} = \mathbb{E}_{\mathbf{x}_T,t,\epsilon}[\left|\left|\epsilon - \hat{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)\right|\right|^2] \quad (4)$$

The reverse diffusion process estimates $q_\theta(\mathbf{x}_{t+1}|\mathbf{x}_t)$ by drawing from another Gaussian distribution parametrized by $\mathcal{N}(\mu_\theta(\mathbf{x}_t, \mathbf{z}, t), \sigma_t^2\mathbf{I})$. The reverse time diffusion process is therefore conditioned on the time step with a learnable mean function μ_θ and an untrained time-dependent variance $\sigma_t^2\mathbf{I} = \beta_t\mathbf{I}$.[81] Our learned model is essentially tasked with learning the mean function μ_θ of this reverse time denoising diffusion process.[81] Inference proceeds from a given latent space coordinate $\mathbf{z}$ by sampling random noise $\mathbf{x}_0$ and making a prediction of the noise using the trained U-net to remove $\hat{\epsilon}_\theta(\mathbf{x}_0, \mathbf{z}, t = 0)$,

$$\mu_\theta(\mathbf{x}_t, \mathbf{z}, t) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}_\theta(\mathbf{x}_t, \mathbf{z}, t)) \quad (5)$$

$$\sigma_t^2\mathbf{I} = \beta_t\mathbf{I} \quad (6)$$

$$\mathbf{x}_{t+1} \sim \mathcal{N}(\mu_\theta(\mathbf{x}_t, \mathbf{z}, t), \sigma_t^2\mathbf{I}) \quad (7)$$

This process is repeated until $t = T$, at which point no additional noise is added to the mean prediction. By default, the DDPM decoder uses 1000 diffusion steps during training and inference; however, we have demonstrated successful results with a related model[83] using as few as 50 diffusion steps to realize a 20× acceleration in inference time. The number of diffusion steps is a hyperparameter that can be tuned based on the size and complexity of the encoded molecular system.

In practice, we have found that even for the largest systems studied to date, relatively simple networks were sufficient to achieve high generative performance: cWGANs comprising generators and discriminator networks comprising two hidden layers of 200 neurons each and DDPMs employing a 1D U-net architecture with 32 channels and three up/down sampling layers. An open-source and user-friendly Python package implementing the cWGAN and DDPM decoders is freely available from https://github.com/Ferg-Lab/molgen.

1.2.4. Deployment. Once all three components of the LSS—the SRV encoder, MDN propagator, and cWGAN/DDPM decoder—have been trained, the LSS model can then be deployed for computationally efficient synthetic trajectory generation in three successive steps (Figure 1d).

1. An initial molecular configuration $\mathbf{x}_{t=0}$ is deterministically encoded into the latent space via the trained SRV encoder to define the initial coordinates $\boldsymbol{\psi}_{t=0} = E(\mathbf{x}_{t=0})$ of the latent space trajectory.
2. The MDN propagator then samples from the learned distribution of jump probabilities to define the next state in the latent space trajectory $\boldsymbol{\psi}_{t=\tau} = P(\boldsymbol{\psi}_{t=0}) \sim p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$, and sampling is iteratively repeated to construct the succession of states $[\boldsymbol{\psi}_{t=0}, \boldsymbol{\psi}_{t=\tau}, \boldsymbol{\psi}_{t=2\tau}, \cdots]$ defining the latent space trajectory in increments of τ. Since the MDN stochastically samples from the learned jump probability distributions, the latent space trajectories are not simply copies of the training data but generate novel trajectories that obey the statistics of the learned microscopic dynamics projected into the latent space and encoded within the trained MDN.
3. The latent space trajectory can be passed to the trained cWGAN or DDPM decoder to generate corresponding molecular configurations consistent with each frame of the latent space trajectory $\hat{\mathbf{x}}_t = D(\boldsymbol{\psi}_t) \sim p(\hat{\mathbf{x}}_t|\boldsymbol{\psi}_t)$. Since the MDN trajectory generation proceeds autonomously within the latent space (i.e., successive frames depend only on the latent space coordinates of the prior frame), decoding is typically conducted after MDN trajectory generation is completed. There is no requirement to decode every frame, but since the decoding operations are independent, decoding is an embarrassingly parallel computational task. The decoding operation can be conceived of as in-painting a realization of the equilibrated fast degrees of freedom consistent with the state of the slow degrees of freedom encoded into the latent space coordinate $\boldsymbol{\psi}_t$. As such, the decoded trajectory $[\hat{\mathbf{x}}_{t=0}, \hat{\mathbf{x}}_{t=\tau}, \hat{\mathbf{x}}_{t=2\tau}, \cdots]$ is expected to be temporally and structurally continuous in the slow degrees of freedom encoded in the latent space but may be discontinuous in the fast degrees of freedom drawn from the learned distribution of molecular configurations consistent with each latent space coordinate $p(\hat{\mathbf{x}}_t|\boldsymbol{\psi}_t)$. Accordingly, it may be desirable to generate multiple decoded realizations associated with each step of the latent space trajectory to produce an ensemble of K molecular configurations at each time step $[\{\hat{\mathbf{x}}_{t=0}^{(i)}\}_{i=1\cdots K}, \{\hat{\mathbf{x}}_{t=\tau}^{(i)}\}_{i=1\cdots K}$, and $\{\hat{\mathbf{x}}_{t=2\tau}^{(i)}\}_{i=1\cdots K}, \cdots]$. It is also possible to select from the ensemble of configurations at each time step that is most structurally consistent with that produced from the prior time step in order to generate a synthetic molecular trajectory that is temporally coherent in both the slow and fast degrees of freedom.[84]

1.2.5. Uncertainty Quantification, Adaptive Retraining, and Model Transferability. The LSS is a fundamentally data-driven model that is only as good as the training data it is provided. As such, the trained LSS model will contain biases associated with systematic errors contained within the training trajectories due to approximations and biases inherent to the force field and the finite nature of the training data.[4] It can be instructive to quantify the epistemic uncertainties in the model by training an ensemble of LSS models over different partitions of the training data. One useful way of doing so is to train the SRV encoder and cWGAN/DDPM decoder over the full training data but an ensemble of MDN propagators over temporally contiguous stratifications of the training data. Analyzing the variability in the learned transition density elements across the ensemble of MDN propagators within a consistent latent space embedding can help expose regions of the latent space that are undersampled in the training data. This can also naturally inform an adaptive retraining paradigm, wherein additional MD simulations are initialized in the vicinity of undersampled states and transitions within the latent space to improve the predictions of the model in the regions where it carries the most uncertainty.[27] Iterative cycles of model training and adaptive sampling can be conducted to optimally deploy collection of MD training data and efficiently converge the LSS model.

The trained LSS model is also constrained to learn a latent space embedding and latent space transition probabilities consistent with the MD training data for the molecular system and thermodynamic conditions under which it was collected. The LSS model, as presented, contains no physical

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

model or inductive biases that ensure its transferability to other thermodynamic state points (e.g., temperatures, pressures, salt concentrations, and solvent conditions) or molecules (e.g., protein mutants). We have previously demonstrated transferability of the encoder and decoder across temperatures and salt conditions for DNA oligomer hybridization/dehybridization, requiring only retraining of the propagator.[30] Very recent work by Dobers et al. employing a variant of an LSS using a SE(3)-invariant encoder-decoder has demonstrated limited transferability across an ensemble of dipeptides.[85] Realizing generic and scalable transferability to larger molecules will likely require more substantial innovations, such as the inclusion of inductive biases and/or conditioning within the LSS paradigm.

2. PREREQUISITES AND INSTALLATION

The LSS software is made freely available as open-source Python packages implementing the SRV encoder, MDN propagator, and cWGAN or DDPM decoders via https://github.com/Ferg-Lab/LSS. The three packages themselves require a small number of common publicly available Python packages to run, which can be easily installed via conda (https://anaconda.org/) or pip (https://www.python.org/). For the purposes of this tutorial, we have created demonstration exercises in Google Colab notebooks (https://colab.research.google.com/) where prerequisite packages are most straightforwardly installed via pip.

Within a Google Colab notebook, we first install the necessary prerequisite packages.

```
# install prerequisite packages
%pip install numpy scipy pandas scikit-learn jupyter ipywidgets==7.7.2
   widgetsnbextension jupyter_contrib_nbextensions matplotlib MDTraj tqdm
   pytest pyemma deeptime einops torch torchvision pytorch-lightning nglview
```

Next, we enable the widgets required to view molecular structures and trajectories by using nglview.

```
# enable Jupyter widgets for nglview
!jupyter nbextension enable --py --sys-prefix widgetsnbextension
!jupyter nbextension enable nglview --py --sys-prefix
!nglview enable
```

Finally, we install the Python packages implementing the SRV encoder, MDN propagator, and cWGAN and DDPM decoders.

```
# install package for SRV encoder
%pip install git+https://github.com/andrewlferguson/snrv.git
```

```
# install package for MDN propagator
%pip install git+https://github.com/Ferg-Lab/mdn_propagator.git
```

```
# install package for cWGAN and DDPM decoders
%pip install git+https://github.com/Ferg-Lab/molgen.git
```

The installation of all prerequisites should require no more than a few minutes for download and installation under a typical high-speed network connection.

3. EXERCISES

3.1. Alanine Dipeptide. As a first exercise, we demonstrate the training and deployment of an LSS for the "hydrogen atom of protein folding", alanine dipeptide. Full instructions and materials necessary to run this tutorial are available at https://github.com/Ferg-Lab/IMSI_LSS. This tutorial was developed for and presented at the workshop "Learning Collective Variables and Coarse Grained Models" held April 22−26, 2024 at the Institute for Mathematical and Statistical Innovation (IMSI) at the University of Chicago.

3.1.1. Preparing Environment. We first load the various required components from the prerequisite packages installed above. To improve model training and inference speeds, GPU runtime can be optionally enabled within the Colab notebook environment.

```
# loading required components
from mdn_propagator.propagator import Propagator
from molgen.models import DDPM
from snrv import Snrv
from snrv.utils import set_random_seed
import mdtraj as md
from pathlib import Path
import torch
import matplotlib.pyplot as plt
import numpy as np
import nglview as nv
from google.colab import output
output.enable_custom_widget_manager()
```

3.1.2. Loading and Processing Training Data. Next, we upload the alanine dipeptide training trajectory provided within the GitHub. The 250 ns trajectory contains 250,000 frames saved at 1 ps intervals.

```
# opening file upload dialog
# N.B. If file upload fails, try using alternate upload means by clicking on
    file icon in left menu and directly uploading to colab session storage
   or by uploading to and mounting Google Drive
from google.colab import files
files.upload()
```

We then proceed to process the trajectory data for LSS training. We use mdtraj to center the trajectory to the origin for visualization convenience and then proceed to extract the pairwise distances between all atoms in the molecule as a translationally and rotationally invariant featurization of the molecular configuration that we pass to the SRV and from which it learns basis functions $\{\zeta_j(\mathbf{x})\}$ and constructs approximations to the eigenfunctions of the transfer operator $\tilde{\psi}_i(\mathbf{x}) = \sum_j s_{ij}\zeta_j(\mathbf{x})$. We also visualize the trajectory, a snapshot of which is presented in Figure 2.

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

```
# processing trajectory
trj_fnames = sorted([str(i) for i in Path('./').glob('alanine-dipeptide
    -*-250ns-nowater.xtc')])
top_fname = 'alanine-dipeptide-nowater.pdb'
trjs = [md.load(t, top=top_fname).center_coordinates() for t in trj_fnames]
trjs
# visualizing
v = nv.show_mdtraj(trjs[0])
v
# extracting atomic pairwise distances
# N.B. Used commented lines to instead extract pairwise distances between
    only backbone atoms
coords_torch = list()
for trj in trjs:
    #t_backbone = trj.atom_slice(trj.top.select('backbone')).
        center_coordinates()
    #pdists = [torch.pdist(p)[None] for p in torch.tensor(t_backbone.xyz)]
    pdists = [torch.pdist(p)[None] for p in torch.tensor(trj.xyz)]
    coords_torch.append(torch.cat(pdists))
len(coords_torch), coords_torch[0].shape
```

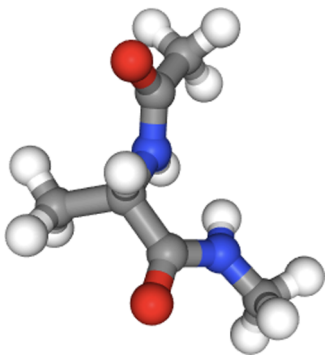


Figure 2. Snapshot of the nglview visualization of the alanine dipeptide training trajectory. Carbon atoms are colored in gray, hydrogen in white, oxygen in red, and nitrogen in blue.

3.1.3. Training the SRV Encoder. We now proceed to train the SRV encoder to learn the leading slow modes of the alanine dipeptide conformational dynamics. Training should require no more than a few minutes with GPU runtime enabled. We elect to learn three slow modes. The first of these corresponds to the equilibrium distribution with a formally infinite implied time scale, which we discard. The next two correspond to the two slowest relaxing collective modes defining the SRV approximations to the dynamical processes with the longest autocorrelation times.

Optionally, we can create visualizations of the training and validation loss curves and a bar plot of the learned leading implied time scales.

```
# setting random seed for reproducibility
set_random_seed(42)
# defining SRV training parameters
input_size = coords_torch[0].size()[1]
output_size = 3
hidden_depth = 2
hidden_size = 100
batch_norm = True
dropout_rate = 0.0
lr = 1E-2
weight_decay = 0.0
val_frac = 0.05
n_epochs = 30
batch_size = 25000
VAMPdegree = 2
is_reversible = True
num_workers = 0

model_snrv = Snrv(input_size, output_size, hidden_depth=hidden_depth,
    hidden_size=hidden_size,
            batch_norm=batch_norm, dropout_rate=dropout_rate, lr=lr,
                weight_decay=weight_decay,
            val_frac=val_frac, n_epochs=n_epochs, batch_size=batch_size,
            VAMPdegree=VAMPdegree,is_reversible=is_reversible, num_workers=
                num_workers,
            activation=torch.nn.GELU(), device=device)
model_snrv = model_snrv.to(device)
# defining lag time
lag_n = 10
# fitting model
model_snrv.fit(coords_torch, lag=lag_n, scheduler=0.9)
# flipping trained model from training to evaluation mode
model_snrv.eval()
```

```
# plotting loss curves
fig, ax = plt.subplots()
ax.plot(np.arange(len(model_snrv.training_losses)), model_snrv.
    training_losses)
ax.plot(np.arange(len(model_snrv.validation_losses)), model_snrv.
    validation_losses)
ax.set_xlabel('epoch')
ax.set_ylabel('loss')
ax.legend(['training','validation'])
fig.tight_layout()
# plotting implied time scales
save_freq = 1 # ps
evals = model_snrv.evals.cpu().detach().numpy()
plt.bar(range(1,evals.size), -lag_n*save_freq/np.log(evals[1:]))
plt.ylabel('Implied timescale (ps)')
plt.xticks(range(1,evals.size))
plt.xlabel('Timescale index')
plt.axhline(lag_n*save_freq, color='k', linestyle=':')
```

It is well-known that the conformational dynamics of alanine dipeptide are well represented within a Ramachandran plot of the backbone ϕ and ψ dihedral angles. An intuition for the learned SRV slow modes can be gained by constructing a Ramachandran plot of the training data and generating a heat map of the slow modes.[39] We note that alanine dipeptide is a particularly simple and well-understood

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

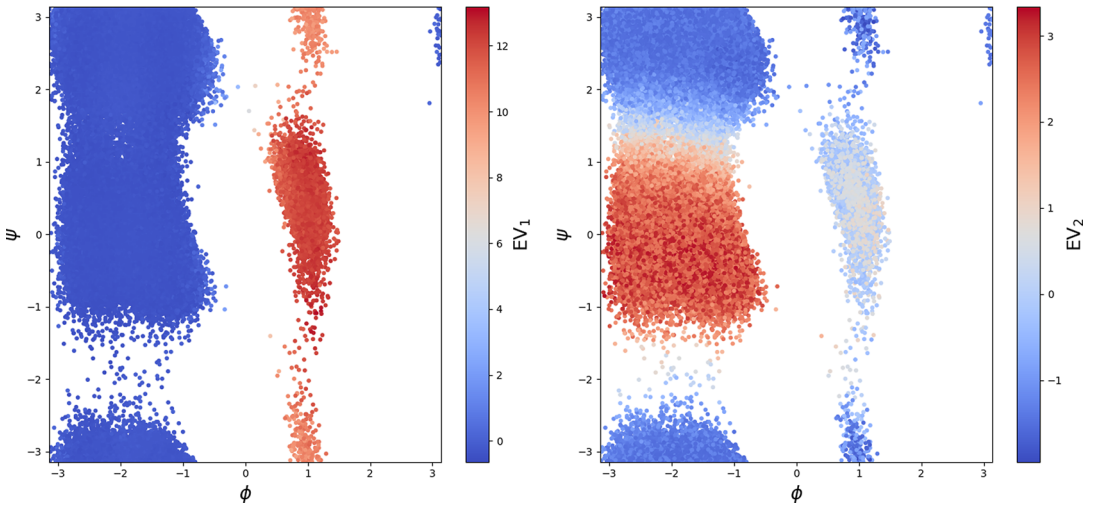


Figure 3. Ramachandran plots of the 250,000 frames constituting the alanine dipeptide training data embedded according to the backbone ϕ and ψ dihedral angles and colored by the value of the SRV approximations to the first (left) and second (right) eigenfunctions of the transfer operator corresponding to the two leading most slowly relaxing dynamical modes of the molecular system. As expected, the leading mode is highly correlated with ϕ and describes transitions between the triplet well at $-\pi < \phi < 0$ comprising the β, P_{II}, and α states, from the doublet well at $\phi \approx 1$ containing the α_L and γ states, while the second leading mode is strongly correlated with ψ in the half space $-\pi < \phi < 0$ and subdivides the triplet well to characterize transitions between the (β, P_{II}) and α states.[39]

system, and physical interpretability can be more challenging to develop for systems for which good structural order parameters are not known a priori. The leading slow modes of the SRV spanning the LSS latent space are themselves, by construction, good order parameters for tracking the long-time dynamical evolution of the system but can be challenging to physically interpret as they are typically complicated nonlinear functions of the molecular featurization. We can optionally construct this visualization directly within the notebook, which we present in Figure 3.

```
# computing latent space coordinates of each training data frame
evecs = model_snrv.transform(torch.cat(coords_torch)).cpu().detach().numpy()
# plotting Ramachandran heatmaps
trj_cat = md.join(trjs)
phi = md.compute_phi(trj_cat)[1].flatten()
psi = md.compute_psi(trj_cat)[1].flatten()
fig, axes = plt.subplots(1, 2, figsize = (15, 7))
axes = axes.flatten()

for e in range(1, evecs.shape[1]):
    evec = evecs[:, e]
    ax = axes[e-1]

    im = ax.scatter(phi, psi, c=evec, s=10, cmap='coolwarm')
    ax.set_xlabel('$\phi$', fontsize=18)
    ax.set_ylabel('$\psi$', fontsize=18)
    ax.set_xlim(-np.pi, np.pi)
    ax.set_ylim(-np.pi, np.pi)
    cbar = plt.colorbar(im, ax=ax)
    cbar.set_label(f'EV$_{e}$', size=18)

plt.tight_layout()
```

Finally, we extract the embeddings of the training data into the latent space coordinates $\boldsymbol{\psi} = (\psi_1, \psi_2) = E(\mathbf{x})$ in preparation for training the MDN propagator and DDPM decoder.

```
# extracting projection of training data into latent space
CVs = [model_snrv.transform(x).cpu().detach()[:, 1:] for x in coords_torch]
```

3.1.4. Training and Deploying the MDN Propagator. We now train the MDN propagator to learn the latent space transition density elements $p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$ from the projection of the training trajectories into the latent space, where we adopt a lag time of $\tau = 10$ frames = 10 ps. Training takes only a few minutes with GPU runtime enabled.

```
# training MDN propagator
model_mdn = Propagator(dim = CVs[0].size(1))
model_mdn.fit(CVs, lag = 10, max_epochs=10)
```

Once trained, the MDN can then be deployed to generate a synthetic trajectory in the latent space by defining an initial state within the latent space corresponding, arbitrarily, to the embedding of the first frame of the training data $\boldsymbol{\psi}_{t=0} = E(\mathbf{x}_{t=0})$, then repeatedly sampling from the learned jump distributions $\boldsymbol{\psi}_{t=\tau} = P(\boldsymbol{\psi}_{t=0}) \sim p_\tau(\boldsymbol{\psi}_{t+\tau}|\boldsymbol{\psi}_t)$ to generate a succession of 100 states within the latent space at $\tau = 10$ ps intervals that defines the synthetic latent space simulation trajectory.

```
# deploying trained MDN propagator
n_steps = int(1E2)
x = CVs[0][0][None]
synthetic_traj_CVs = model_mdn.gen_synthetic_traj(x, n_steps)
```

3.1.5. Training and Deploying the DDPM Decoder. Finally, we train a DDPM decoder to learn to generatively decode molecular configurations from latent space coordinates by learning the relationship $p(\hat{\mathbf{x}}_i|\boldsymbol{\psi}_i)$. To do so, we train over $(\mathbf{x}_i, \boldsymbol{\psi}_i = E(\mathbf{x}_i))$ pairs corresponding to translationally and rotationally aligned molecular configurations from the training trajectories and their projected images in the latent space. We note that the training trajectory is already translationally and rotationally aligned, so we do not need to conduct this operation here, but it is easy to do so using the functionality within the mdtraj package. Additionally,

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

DDPM training can be quite computationally burdensome, so for the purposes of this tutorial, we reduce the molecular representation to only the backbone atoms to reduce the training time. Having done so, the DDPM trains in just a few minutes with GPU runtime enabled.

```
# preparing training data
xyz = list()
for trj in trjs:
    t_backbone = trj.atom_slice(trj.top.select('backbone')).
        center_coordinates()
    n = trj.xyz.shape[0]
    xyz.append(torch.tensor(t_backbone.xyz.reshape(n, -1)).float())
# training DDPM decoder
model_ddpm = DDPM(xyz[0].shape[1], CVs[0].shape[1])
model_ddpm.fit(xyz, CVs, max_epochs=3)
```

Once trained, the DDPM can then be deployed to decode the synthetic latent space trajectory generated by the trained MDN by performing the generative decoding operation $\hat{\mathbf{x}}_t = D(\boldsymbol{\psi}_t) \sim p(\hat{\mathbf{x}}_t|\boldsymbol{\psi}_t)$. This requires only a couple of minutes with GPU runtime enabled. We then visualize the resulting trajectory using nglview and save it to a file. A snapshot from the synthetic trajectory is presented in Figure 4. We note that the quality of the reconstruction can be somewhat improved by training the DDPM for additional epochs, but we choose to train for only three epochs in this tutorial to limit the required training time.

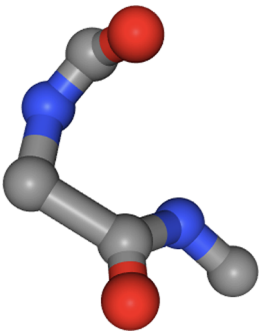


Figure 4. Snapshot of nglview visualization of backbone-only reconstruction of alanine dipeptide from the synthetic trajectory produced by the trained LSS. Carbon atoms are colored in gray, oxygen in red, and nitrogen in blue.

```
# deploying trained DDPM decoder
xyz_gen = model_ddpm.generate(synthetic_traj_CVs)
# visualizing decoded synthetic latent space trajectory
xyz_gen = xyz_gen.reshape(xyz_gen.size(0), -1 , 3).numpy()
fake_trj = md.Trajectory(xyz = xyz_gen, topology=t_backbone.top)
v = nv.show_mdtraj(fake_trj)
v
# saving synthetic molecular trajectory to file
fake_trj.save_pdb('ADP_backbone_synthetic_traj.pdb')
```

3.2. BBA. As a second, more sophisticated exercise, we illustrate the training and deployment of an LSS for the de novo designed beta−beta−alpha (BBA) protein (PDB: 1FME). BBA is a 28-residue fast-folding protein that was designed based on a zinc finger template (PDB: 1AAY) to stabilize a native structure containing two short β-sheets and one α-helix.[86] A long simulation of this protein was performed by Lindorff-Larsen et al.[3] at D.E. Shaw Research (DESRES), which uncovered numerous (un)folding events that traverse through an ensemble of metastable states. We use 200 μs of these simulation data saved every 200 ps (resulting in 1,000,000 frames) to train our LSS. Given the larger size of BBA compared to alanine dipeptide, we reduce our representation of the system to the Cα atoms only and use the set of pairwise distances between each of these atoms as our feature set. A more aggressively parsimonious featurization may retain only every n^{th} Cα atom. The training process is very similar to that of alanine dipeptide, with the main difference being the scale and composition of features and the number of learned SRV coordinates. Based on the substantially larger size of the training set and increased training time, we provide a walkthrough that loads pretrained models for the SRV, MDN, and DDPM decoder and show some quantitative results below. Full instructions and materials necessary to run this tutorial are available at https://github.com/Ferg-Lab/LSS_BBA including an additional code that can be used to train the models from scratch.

3.2.1. Loading and Processing Training Data. We have provided the reduced representation of DESRES trajectories in the collab_files directory, with frames strided at 10 ns (compared to 200 ps during training), reducing the number

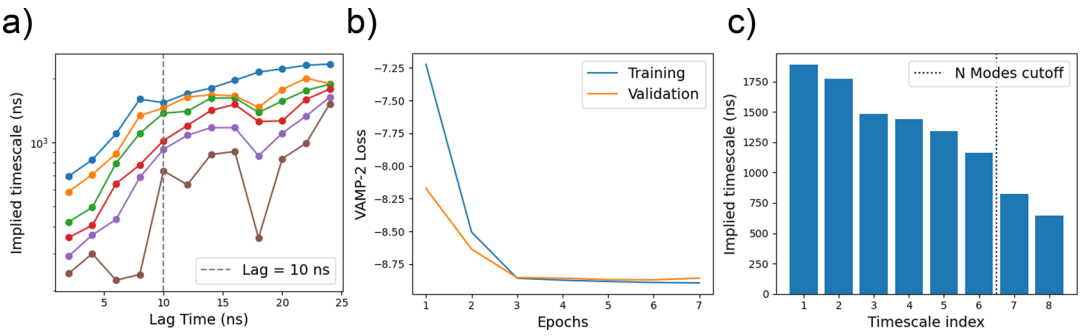


Figure 5. Hyperparameter selection for training the SRV encoder applied to BBA trajectories. (a) Implied time scale of leading models based on SRV models trained at various lag times. Each curve corresponds to a different mode, where the blue curve shows the slowest process, and brown shows the fastest process. A lag time of 10 ns (dashed line) was selected based on approximate convergence of these modes. (b) Training and validation loss calculated by the sum of leading eigenvalues squared (VAMP-2 loss). Training time of three epochs was selected based on convergence of validation loss. (c) Implied time scales corresponding to the leading eight nontrivial modes. A spectral gap is observed after the sixth mode, and we retrain our model and perform subsequent analysis using only the six leading modes.

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

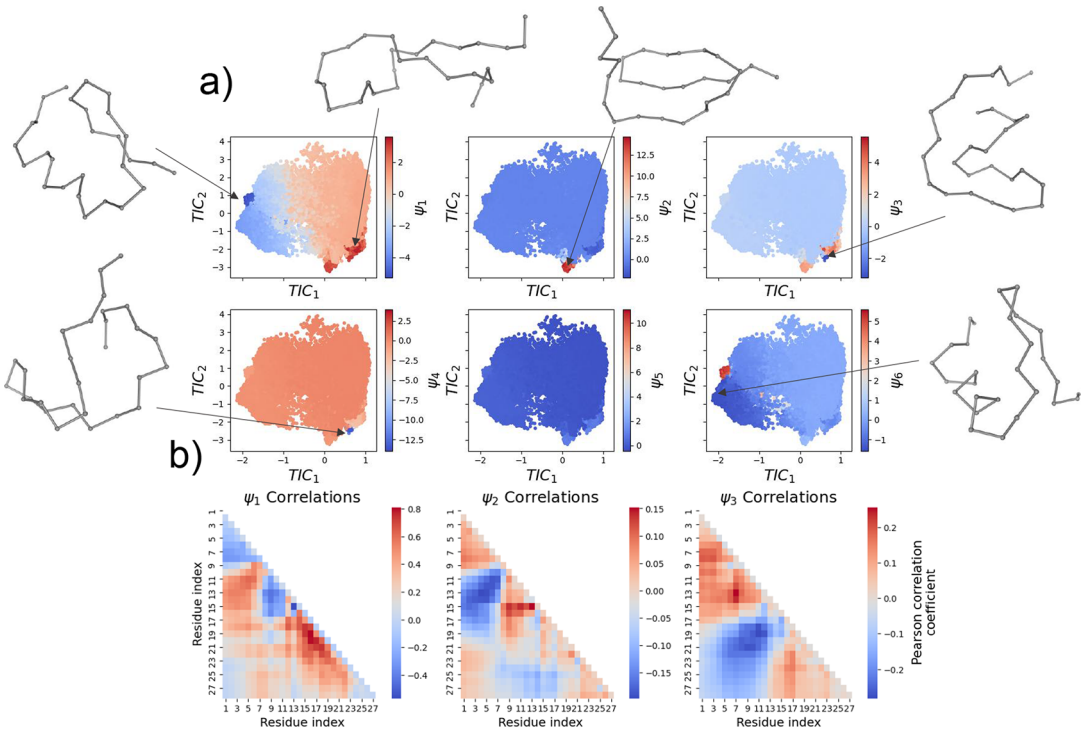


Figure 6. Projection of the 20,000 frames of the BBA training data into the leading two TICA modes. (a) Heat maps coloring the points by the values of the six leading SRV modes $\{\psi_i\}_{i=1}^{6}$ onto which we have also projected representative molecular configurations. Changes in the values of the SRV modes expose interconversions between different configurational populations. ψ_1, for example, is strongly correlated with TIC_1 and corresponds to global (un)folding. Noting the striking, but purely serendipitous, similarity of our TICA embedding to a map of Australia, ψ_2 corresponds to transitions between the red partially unfolded β-sheet state located in Melbourne to the rest of the island, whereas ψ_3 corresponds to transitions between the blue "folded-in-half" state in Canberra to the more populated red unfolded states in Melbourne and Sydney. (b) Pearson correlation coefficients between each of the 378 pairwise distance features and three leading SRV modes. Analysis of these linear correlations can help expose the interpretability of the learned SRV modes.

of frames from 1,000,000 to 20,000. These files can be downloaded locally and then loaded into the notebook. Alternatively, users can pass in their custom trajectories and topologies by setting trj_fnames and top_fname to their corresponding paths. Trajectory data are contained in 10 .dcd files, and the topology is specified by a single .pdb file. To simplify visualization, after data are loaded, we select the first frame of the first trajectory as our reference configuration and superpose each subsequent frame with respect to this configuration.

```
# Path to local or uploaded trajectories
load_path = '/PATH/TO/DES'
trj_fnames = sorted(glob.glob(f'{load_path}/1FME-0-c-alpha-00*.dcd'))

# Path to local or uploaded topology
top_fname = f'{load_path}/1FME-0-c-alpha.pdb'
save_freq = 10 # ns

trjs = [md.load(t, top=top_fname).center_coordinates() for t in trj_fnames]
ref_frame = trjs[0][0]
trjs = [t.superpose(ref_frame, 0) for t in trjs]
```

We then use the torch.pdist function to calculate all pairwise distances between the 28 Cα atoms, resulting in a $(28 \times 27)/2 = 378$-dimensional feature set.

```
coords_torch = list()
for trj in trjs:
    pdists = [torch.pdist(p)[None] for p in torch.tensor(trj.xyz)]
    coords_torch.append(torch.cat(pdists))
len(coords_torch), coords_torch[0].shape
```

3.2.2. Loading and Deploying the SRV Encoder. We load our pretrained SRV, which is stored in the collab_files directory. We select a lag of 50 steps (10 ns) based on the approximate convergence of leading time scales (Figure 5a). We train our SRV for three epochs based on convergence of the validation loss (Figure 5b) and project into the six leading nontrivial eigenvectors based on a spectral gap after the sixth mode (Figure 5c). We recall that we discard the trivial leading eigenvector ψ_0 corresponding to the equilibrium distribution and possessing a formally infinite implied time scale. In addition to loading the model weights, we load the expansion coefficients and specify that the model has already been fitted. The code for retraining the SRV from scratch is also available in the notebook.

```
model_save_path = './model_snrv-3.pth'
ckp = torch.load(model_save_path)
model_snrv.load_state_dict(ckp['model_state_dict'])
model_snrv.evals = ckp['evals']
model_snrv.expansion_coefficients = ckp['expansion_coefficients']
model_snrv.is_fitted = True
```


https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

We then encode our molecular features into SRV eigenvectors using the snrv.transform method. These eigenvectors define the latent space coordinates, which will be used later to train both the propagator and decoder. We save another copy of CVs_cat which concatenates latent space coordinates across all trajectories and converts the data to an np.ndarray in order to perform additional analysis.

```
CVs = [model_snrv.transform(x).cpu().detach()[:, 1:] for x in coords_torch]
CVs_cat = torch.cat(CVs).numpy()
```

Without access to good a priori collective variables, such as phi/psi angles for alanine dipeptide, it can be more challenging to interpret our learned SRV coordinates. As an initial check, we first plot comparisons against the leading modes of time-lagged independent component analysis (TICA).[55] TICA solves a linear time-lagged eigenvalue problem (without using a neural network to optimize the basis) and can serve as a reasonable linear approximation for the slowest modes. TICA modes also tend to possess higher variance than SRV modes, which can make them better suited to interpretable visualizations. We obtain the leading TICA coordinates using the deeptime Python package[87] as shown below. We specify a lag of one time step, corresponding to a lag time of 10 ns, since we loaded the trajectories with a 10 ns stride.

```
import deeptime as dt
tica = dt.decomposition.TICA(lagtime=1, dim=2)
TICs = tica.fit_transform([a.numpy() for a in coords_torch]).reshape(-1, 2)
```

In Figure 6a, we show our leading SRV modes overlaid on the first two TIC coordinates. We see a strong correlation between ψ_1 and TIC_1, corresponding to the global (un)-folding process. Higher-order eigenvectors differentiate smaller regions of TIC space from the greater ensemble, corresponding to transitions in and out of metastable states. The code for generating the SRV heatmaps projected into the leading TIC coordinates is provided below, and in Figure 6a, we manually annotate these with representative molecular configurations.

```
fig, axes = plt.subplots(2, 3, figsize = (12, 6), sharey=True, sharex=True)
axes = axes.flatten()
for e in range(1, evecs.shape[1]):
    evec = evecs[:, e]
    ax = axes[e-1]
    im = ax.scatter(TICs[::stride, 0], TICs[::stride, 1], c=evec, s=10, cmap
        ='coolwarm')
    ax.set_xlabel('$TIC_1$', fontsize=18)
    ax.set_ylabel('$TIC_2$', fontsize=18)
    cbar = plt.colorbar(im, ax=ax)
    cbar.set_label(f'$\psi_{e}$', size=18)
plt.tight_layout()
```

Physical interpretation of the SRV coordinates can be aided by examining their linear correlations with candidate input features. In Figure 6b, we illustrate the Pearson correlation coefficients between the leading three modes and each of the 378 pairwise distance features used to train the encoder. We observe strong positive correlations between ψ_1 and pairwise distances within the group of residues 15−23, which correspond to interactions within the α-helix. These correlations appear to strongly emerge since the α-helix tends to be the last secondary structural element that forms during the folding process. We also observe significant negative correlations between ψ_2 and pairwise distances from residues (1−7) to residues (10−19) which correspond to changes in distance between the two β-sheets. Indeed, we find that ψ_2 distinguishes partially unfolded states that contain a β-sheet structure from those that do not. The ψ_3 coordinate correlates with transitions into a more rare metastable state in which the protein is effectively folded in half and the termini are in close proximity. The strong negative correlations of ψ_3 with pairwise distances from residues (1−12) ascending to residues (17−28) descending correlate with the relative proximity of these residue pairs extending from the termini (residues 1 and 28) down to the hinge at residues 14−16. The code to generate these correlation plots is provided below.

```
fig, axes = plt.subplots(1, 3, figsize=(14, 4))
all_feats = np.concatenate(coords_torch)
num_features = all_feats.shape[1]
nres = trjs[0].n_residues

for evec_idx, ax in enumerate(axes):
    heatmap = np.zeros((nres, nres))
    correlations = np.zeros(num_features)
    cnt = 0
    for i in range(num_features):
        correlations[i], _ = pearsonr(all_feats[:, i], evecs[:, evec_idx +
            1])
    for i in range(len(heatmap)):
        for j in range(i+1, nres):
            heatmap[i, j] = correlations[cnt]
            heatmap[j, i] = correlations[cnt]
            cnt += 1
    mask = np.triu(np.ones_like(heatmap, dtype=bool))
    sns.heatmap(heatmap, mask=mask, cmap='coolwarm', ax=ax)
    ax.set_title(f'$\psi_{evec_idx+1}$ Correlations', fontsize=18)
    ax.set_xlabel('Residue index', fontsize=14)
    ax.set_xticklabels(1 + 2*np.arange(14))
    ax.set_yticklabels(1 + 2*np.arange(14))
axes[0].set_ylabel('Residue index', fontsize=14)
```

3.2.3. Loading and Deploying the MDN Propagator. Next, we deploy our MDN propagator to generate synthetic trajectories in the learned latent space. We load our pretrained model which was run for 100 epochs and predicts conditional probabilities in the six-dimensional latent space. We trained the MDN using the same 10 ns lag time as that of the SRV encoder. It is not a requirement that the same lag time be used for training the SRV and the MDN; however, using a shorter MDN lag time risks breaking the Markovian (i.e., memoryless) assumption, and longer lag times unnecessarily reduce the temporal resolution of the trajectory generated by the propagator.

```
model_mdn = Propagator(dim = CVs[0].size(1))
model_save_path = f'{load_path}/model_mdn-100.pth'
model_mdn.load_state_dict(torch.load(model_save_path))
model_mdn.is_fit = True
```

We use our pretrained model to generate a synthetic trajectory that is 10× longer than the training data, corresponding to an effective simulation time of 2 ms. This

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

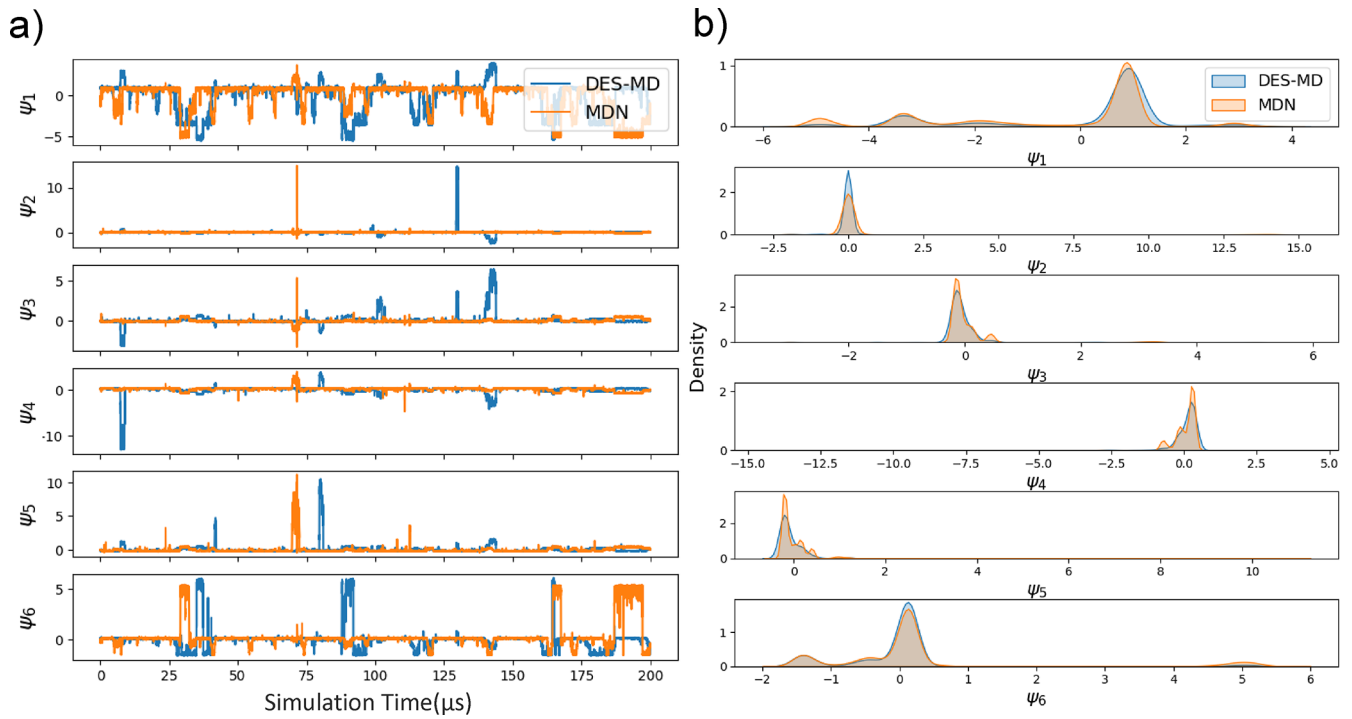


Figure 7. Comparison of synthetic LSS MDN trajectory dynamics and thermodynamics with ground-truth D.E. Shaw molecular dynamics (DES-MD) training data for BBA. (a) DES-MD trajectory data (blue) projected into the leading six SRV modes as a function of time. The LSS MDN trajectory (orange) initialized from the same starting coordinates as the MD data and propagated for an equivalent time scale, where each step is separated by a lag of 10 ns corresponding to a total simulation time of 200 μs. (b) Probability distributions of each leading mode for the MD trajectory (blue) and the synthetic LSS MDN trajectory (orange) constructed by kernel density estimation.

time scale represents an extremely large computational expense at both all-atom and coarse-grained resolution, but the MDN generates this trajectory in less than 2 min on a single GPU. In order to ensure consistency with our MD trajectory, we calculate the number of integration steps by dividing by the MDN lag time. Our initial configuration x is set to the latent coordinates of the first frame of our SRV embedding and each subsequent frame is sampled from the predicted MDN distribution.

```
n_steps = 10 * len(CVs_cat) / lag_n
x = CVs[0][0][None]
synthetic_traj_CVs = model_mdn.gen_synthetic_traj(x, n_steps)
```

To inspect the consistency of our model with the training data, we plot in Figure 7a the first 10% of our synthetic trajectory as a function of time (orange) and compare it with the SRV embedding of MD data (blue). We show each of the six leading modes on a separate axis and observe similar dynamical trends between the corresponding modes of the two trajectories. As described above, ψ_1 corresponds to the global (un)folding process, and we observe several transitions occurring at distinct time points for each trajectory. Higher-order modes mainly correspond to rare excursions through metastable states, and we see the abruptness of transitions and the relatively short lifetime of these states recapitulated in the synthetic trajectory. Because synthetic trajectories are substantially cheaper to generate than standard MD, we can sample many more transitions than are available in the training data and use these to greatly reduce the statistical uncertainty in structural, thermodynamic, and kinetic observables. The code to generate these trajectory comparisons is provided below.

```
fig, axes = plt.subplots(6, figsize=(8, 8), sharex=True)
x = 0.01*np.arange(len(evecs)//lag_n)  # convert to ms
for evec_idx, ax in enumerate(axes):
    ax.plot(x, CVs_cat[:, evec_idx], label='DES')
    ax.plot(x, synthetic_traj_CVs[:len(x), evec_idx], label='MDN')
    ax.set_ylabel(f'$\psi_{evec_idx+1}$', fontsize=16)
axes[-1].legend(fontsize=14)
axes[-1].set_xlabel('Simulation Time (ms)', fontsize=16)
```

Additionally, we may be interested in a thermodynamic comparison between the MD trajectories and the synthetic LSS trajectories generated by the MDN. In Figure 7b, we plot a kernel density estimate of the probability distributions along the six leading SRV modes from each trajectory. In this plot, we include data from the complete synthetic trajectory, which effectively represents a 10× longer time scale than the MD data. As such, we expect strong similarities between the synthetic and MD distributions but also expect differences that reflect both the stochasticity in the generated trajectory and the disparate data volumes. Indeed, there is a close correspondence overall, but, for example, we observe that the MDN predicts a slightly higher population in the $\psi_1 < -4$ region and a slightly lower population near $\psi_2 = 0$ than is seen in the MD data. The code to plot these distributions is provided below.

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

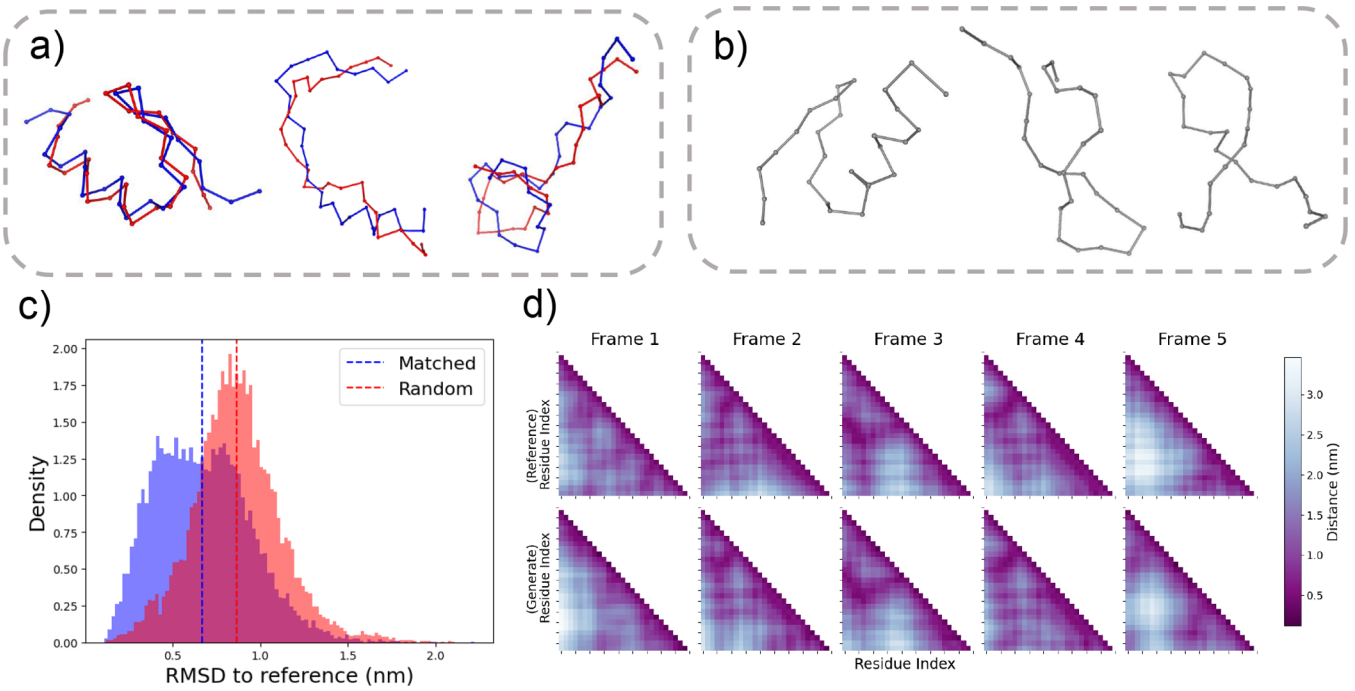


Figure 8. Performance of the DDPM decoder applied to BBA. (a) Synthetic configurations (red) conditionally generated from the SRV latent space coordinates superposed on the corresponding reference configurations (blue). (b) Synthetic configurations generated from randomly selected frames of an MDN trajectory. No corresponding reference configurations exist, but configurations are physically plausible with respect to the MD ensemble. (c) Distribution of RMSDs of all conditionally generated structures with respect to their corresponding (matched) reference structures (blue) and with respect to randomly selected structures (red). The matched structures have on average significantly lower RMSDs, indicating that the decoder is paying attention to the conditioning provided by the latent space coordinates. (d) All pairwise distances between Cα carbons for five evenly spaced frames within the test trajectory of reference (top row) and generated (bottom row) configurations. Given the correct adherence to SRV conditioning, pairwise distance maps of reference and generated structures are expected to approximately match.

```
fig, axes = plt.subplots(6, figsize=(8, 8), sharex=False)
evecs = np.array(evecs)
synthetic_traj_CVs = np.array(synthetic_traj_CVs)
evecs[np.isinf(evecs)] = np.nan
synthetic_traj_CVs[np.isinf(synthetic_traj_CVs)] = np.nan

for evec_idx, ax in enumerate(axes):
    sns.kdeplot(evecs[::lag_n, evec_idx+1], ax=ax, label='DES', fill=True)
    sns.kdeplot(synthetic_traj_CVs[:, evec_idx], ax=ax, label='MDN', fill=
        True)
    ax.set_ylabel('')
    ax.set_xlabel(f'$\psi_{evec_idx+1}$', fontsize=16)

axes[0].legend(fontsize=14)
axes[3].set_ylabel(' '*20 + 'Density', fontsize=16)
plt.tight_layout()
```

3.2.4. Loading and Deploying the DDPM Decoder. Finally, we load our pretrained DDPM decoder in order to restore the Cα resolution of the original MD training data. This model was trained for 40 epochs and used 100 diffusion steps. The first frame was selected as a reference configuration to align all subsequent frames. We tested several reference configurations and did not find a difference in DDPM performance in this case, but in general, we recommend selecting a frame close to the native state or an otherwise highly populated state. To ensure that the model was not overfit, we performed a 90/10 split on segments of each of our 10 training trajectories. All subsequent analyses will be shown on the 10% hold-out test data.

```
test_perc = 0.1
n_test = int(trjs[0].xyz.shape[0]*test_perc)
n_train = (trjs[0].xyz.shape[0] - n_test)

xyz_train = [torch.tensor(trj.xyz[n_test:].reshape(n_train, -1)).float() for
    trj in trjs]
CVs_train = [cv[n_test:] for cv in CVs]
xyz_test = [torch.tensor(trj.xyz[:n_test].reshape(n_test, -1)).float() for
    trj in trjs]
CVs_test = [cv[:n_test] for cv in CVs]

# Training a cWGAN offers faster decoding but lower structural quality
# model = WGANGP(xyz[0].shape[1], CVs_train[0].shape[1])

n_timesteps = 100
model_ddpm = DDPM(xyz_train[0].shape[1], CVs_train[0].shape[1], timesteps=
    n_timesteps)

model_save_path = f'{load_path}/model_ddpm-40-eps-{n_timesteps}-ts.pth'
model_ddpm.load_state_dict(torch.load(model_save_path))
model_ddpm.is_fit = True
```

To evaluate the quality of our decoder, we can first apply the model to SRV embeddings of our MD data and compare against the corresponding reference structure. In the following code, we aggregate the trajectory frames corre-

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

sponding to our test set and generate synthetic trajectories conditioned on each SRV embedding.

```
trj_ref = md.join([trj[:n_test]for trj in trjs])
CVs_ref = torch.Tensor(np.concatenate([cv.numpy() for cv in CVs_test]))

xyz_gen = model_ddpm.generate(torch.Tensor(CVs_ref))
xyz_gen = xyz_gen.reshape(xyz_gen.size(0), -1 , 3).numpy()
fake_trj = md.Trajectory(xyz = xyz_gen, topology=trjs[0].top)
fake_trj = md.join([ft.superpose(rf) for ft, rf in zip(fake_trj, trj_ref)])
```

To compare generated structures against their corresponding reference configurations, in Figure 8a, we visualize three structures from distinct regions of SRV latent space to compare the generated structure (red) superposed on the reference structure (blue). We do not expect these structures to be identical given that information about fast degrees of freedom is lost during the SRV encoding and the model is based on a denoising diffusion process that will stochastically produce distinct samples for identical conditionings, but as expected, we observe structural similarities associated with the slow dynamics governing the (un)folding of the protein. In Figure 8b, we illustrate synthetic configurations decoded from randomly selected frames of an MDN trajectory. In this case, no corresponding reference configurations exist, but configurations are physically plausible with respect to the MD ensemble.

To verify that the model is paying attention to the latent space coordinate conditioning, we compared the root-mean-square deviation (RMSD) of rototranslationally aligned generated samples with respect to their corresponding matched reference structures versus their alignment to randomly selected frames. We show a comparison of these two distributions in Figure 8c, which, as expected, reveals significantly lower mean RMSDs for the matched samples than for the random ones. The code to perform this analysis is provided below.

```
rmsds_aligned = []
for f_gen, f_ref in zip(fake_trj, trj_ref):
    rmsds_aligned.append(md.rmsd(f_gen, f_ref)[0])

rmsds_random = []
rand_list = list(range(len(trj_ref)))
random.shuffle(rand_list)
for f_gen, f_ref in zip(fake_trj, [trj_ref[i] for i in rand_list]):
    rmsds_random.append(md.rmsd(f_gen, f_ref)[0])

x = np.arange(len(rmsds_aligned))
plt.hist(rmsds_aligned, bins=100, alpha=0.5, color='blue', density=True)
plt.hist(rmsds_random, bins=100, alpha=0.5, color='red', density=True)

plt.axvline(np.mean(rmsds_aligned), linestyle='dashed', c='blue', label='
    Matched')
plt.axvline(np.mean(rmsds_random), linestyle='dashed', c='red', label='
    Random')

plt.legend(fontsize=16)
plt.xlabel('RMSD to reference (nm)', fontsize=18)
plt.ylabel('Density', fontsize=18)
```

As an additional comparison, we visualize all Cα carbon pairwise distances between the reference and generated configurations. Although not identical, we would expect overall structural trends to remain consistent at samples corresponding to the same regions of SRV space. In Figure 8d, we observe strong similarities between reference (top row) and generated (bottom row) structures for five evenly spaced frames in the test trajectory. The code to generate these plots is provided below.

```
pdists_ref = [torch.pdist(p)[None] for p in torch.tensor(trj_ref.xyz
    [::2000])]
pdists_gen = [torch.pdist(p)[None] for p in torch.tensor(fake_trj.xyz
    [::2000])]

fig, axes = plt.subplots(2, 5, figsize=(15, 6))
cmap = 'BuPu_r'

global_min = np.inf
global_max = -np.inf

for p_ref, p_gen in zip(pdists_ref, pdists_gen):
    global_min = min(global_min, p_ref.min(), p_gen.min())
    global_max = max(global_max, p_ref.max(), p_gen.max())
# Create heatmaps
for f, (p_ref, p_gen, ax_row) in enumerate(zip(pdists_ref, pdists_gen, axes.
    T)):
    heatmap_ref = np.zeros((nres, nres))
    heatmap_gen = np.zeros((nres, nres))
    cnt = 0
    for i in range(len(heatmap_ref)):
        for j in range(i + 1, nres):
            heatmap_ref[j, i] = p_ref[0, cnt]
            heatmap_gen[j, i] = p_gen[0, cnt]
            cnt += 1
    mask = np.triu(np.ones_like(heatmap_ref, dtype=bool))
    sns.heatmap(heatmap_ref, mask=mask, cmap=cmap, ax=ax_row[0], cbar=False,
        vmin=global_min, vmax=global_max)
    sns.heatmap(heatmap_gen, mask=mask, cmap=cmap, ax=ax_row[1], cbar=False,
        vmin=global_min, vmax=global_max)
    ax_row[0].set_title(f'Frame {f+1}', fontsize=22)
```

After verifying that our DDPM is correctly adhering to the conditioning and predicting a physically plausible structure, we employ the model to decode our synthetic MDN trajectory to high-resolution molecular structures. As visualized in Figure 8b, these structures are physically robust and retain characteristics of the MD data. The transformation from latent space trajectories to Cα-resolution synthetic structures is performed by the code below.

```
xyz_syn = model_ddpm.generate(torch.Tensor(synthetic_traj_CVs[:1000]))
xyz_syn = xyz_syn.reshape(xyz_syn.size(0), -1 , 3).numpy()
trj_syn = md.Trajectory(xyz = xyz_syn, topology=trjs[0].top)
```

4. CONCLUSIONS

The LSS approach offers a powerful tool to expand the effective time scales of MD simulations. The model leverages deep learning architectures to construct a surrogate dynamical model from modest training data and enable the generation of synthetic trajectories at a drastically reduced computational cost at approximately 6 orders of magnitude lower than that of standard MD. This approach facilitates dense sampling of

https://doi.org/10.1021/acs.jpca.4c05389
J. Phys. Chem. A 2024, 128, 10299−10317

molecular phase space and enables a large reduction in the statistical errors in structural, thermodynamic, and kinetic observables. The LSS framework employs three distinct deep learning architectures to encode high-dimensional dynamics into a low-dimensional latent space, propagate dynamics within this space, and decode these dynamics back to the original molecular configuration. This software tutorial introduces the theoretical underpinnings and practical applications of LSS, with examples demonstrated on alanine dipeptide and the 28-residue BBA protein within accessible Google Colab notebooks.

The LSS framework has been extended to multimolecular and highly distributed systems,[30] and going forward, we are interested in developing conditioning on thermodynamic parameters such as temperature as well as chemical features like protein sequence. We also acknowledge current limitations in terms of the flexibility of the encoding and decoding frameworks. For example, molecular featurizations are currently selected by the user but could instead be learned by a structure-agnostic graph encoding similar to Schnet.[88] Additionally, the decoder currently requires alignment to a reference configuration, which may be ambiguous for larger and more complex systems and could be replaced by an equivariant decoding mechanism. Indeed, the field is developing rapidly, and an advantage of the LSS framework is that novel architectures can be modularly deployed to enhance or replace any of the three components. Going forward, we hope that the molecular modeling community will find utility in the LSS framework, collaboratively build upon this approach by experimenting with novel learning methods and network architectures, and apply these models to increasingly complex and challenging systems.

■ ASSOCIATED CONTENT

Data Availability Statement

The LSS Python package is available free and open source from https://github.com/Ferg-Lab/LSS and the alanine dipeptide and BBA exercises are hosted, respectively, at https://github.com/Ferg-Lab/IMSI_LSS and https://github.com/Ferg-Lab/LSS_BBA.

■ AUTHOR INFORMATION

Corresponding Author

Andrew L. Ferguson − *Pritzker School of Molecular Engineering, The University of Chicago, Chicago, Illinois 60637, United States;* orcid.org/0000-0002-8829-9726; Email: andrewferguson@uchicago.edu

Authors

Michael S. Jones − *Pritzker School of Molecular Engineering, The University of Chicago, Chicago, Illinois 60637, United States*

Kirill Shmilovich − *Pritzker School of Molecular Engineering, The University of Chicago, Chicago, Illinois 60637, United States;* Present Address: Prescient Design, Genentech Research and Early Development (gRED) Computational Science, Genentech, South San Francisco, California 94080, United States

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jpca.4c05389

Notes

The authors declare the following competing financial interest(s): A.L.F. is a co-founder and consultant of Evozyne, Inc. and a co-author of US Patent Applications 16/887,710 and 17/642,582, US Provisional Patent Applications 62/853,919, 62/900,420, 63/314,898, 63/479,378, 63/521,617, and 63/669,836, and International Patent Applications PCT/US2020/035206, PCT/US2020/050466, and PCT/US24/10805.

■ ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grant No. CHE-2152521. This work was completed in part with resources provided by the University of Chicago Research Computing Center. We gratefully acknowledge computing time on the University of Chicago's high-performance GPU-based cyberinfrastructure supported by the National Science Foundation under Grant No. DMR-1828629. Part of this research was performed while the author was visiting the Institute for Mathematical and Statistical Innovation (IMSI), which is supported by the National Science Foundation (Grant No. DMS-1929348).

■ REFERENCES

(1) Frenkel, D.; Smit, B. *Understanding Molecular Simulation: From algorithms to applications*; Elsevier, 2023.

(2) Tchipev, N.; Seckler, S.; Heinen, M.; Vrabec, J.; Gratl, F.; Horsch, M.; Bernreuther, M.; Glass, C. W.; Niethammer, C.; Hammer, N.; et al. TweTriS: Twenty trillion-atom simulation. *International Journal of High Performance Computing Applications* **2019**, *33*, 838−854.

(3) Lindorff-Larsen, K.; Piana, S.; Dror, R. O.; Shaw, D. E. How fast-folding proteins fold. *Science* **2011**, *334*, 517−520.

(4) Karplus, M.; Petsko, G. A. Molecular dynamics simulations in biology. *Nature* **1990**, *347*, 631−639.

(5) Noid, W. G. Perspective: Coarse-grained models for biomolecular systems. *J. Chem. Phys.* **2013**, *139*, No. 090901.

(6) Noid, W. Perspective: Advances, challenges, and insight for predictive coarse-grained models. *J. Phys. Chem. B* **2023**, *127*, 4174−4207.

(7) Jin, J.; Pak, A. J.; Durumeric, A. E.; Loose, T. D.; Voth, G. A. Bottom-up coarse-graining: Principles and perspectives. *J. Chem. Theory Comput.* **2022**, *18*, 5759−5791.

(8) Saunders, M. G.; Voth, G. A. Coarse-graining methods for computational biology. *Annual Review of Biophysics* **2013**, *42*, 73−93.

(9) Padding, J.; Briels, W. J. Systematic coarse-graining of the dynamics of entangled polymer melts: the road from chemistry to rheology. *J. Phys.: Condens. Matter* **2011**, *23*, No. 233101.

(10) Rudzinski, J. F. Recent progress towards chemically-specific coarse-grained simulation models with consistent dynamical properties. *Computation* **2019**, *7*, 42.

(11) Sidky, H.; Chen, W.; Ferguson, A. L. Machine learning for collective variable discovery and enhanced sampling in biomolecular simulation. *Mol. Phys.* **2020**, *118*, No. e1737742.

(12) Rohrdanz, M. A.; Zheng, W.; Clementi, C. Discovering mountain passes via torchlight: Methods for the definition of reaction coordinates and pathways in complex macromolecular reactions. *Annu. Rev. Phys. Chem.* **2013**, *64*, 295−316.

(13) Abrams, C.; Bussi, G. Enhanced sampling in molecular dynamics using metadynamics, replica-exchange, and temperature-acceleration. *Entropy* **2014**, *16*, 163−199.

(14) Yang, Y. I.; Shao, Q.; Zhang, J.; Yang, L.; Gao, Y. Q. Enhanced sampling in molecular dynamics. *J. Chem. Phys.* **2019**, *151*, No. 070902.

(15) Lyubartsev, A.; Martsinovski, A.; Shevkunov, S.; Vorontsov-Velyaminov, P. New approach to Monte Carlo calculation of the

free energy: Method of expanded ensembles. *J. Chem. Phys.* **1992**, *96*, 1776−1783.
(16) Bolhuis, P. G.; Chandler, D.; Dellago, C.; Geissler, P. L. Transition path sampling: Throwing ropes over rough mountain passes, in the dark. *Annu. Rev. Phys. Chem.* **2002**, *53*, 291−318.
(17) Chong, L. T.; Saglam, A. S.; Zuckerman, D. M. Path-sampling strategies for simulating rare events in biomolecular systems. *Curr. Opin. Struct. Biol.* **2017**, *43*, 88−94.
(18) Escobedo, F. A.; Borrero, E. E.; Araque, J. C. Transition path sampling and forward flux sampling. Applications to biological systems. *J. Phys.: Condens. Matter* **2009**, *21*, No. 333101.
(19) Bolhuis, P.; Dellago, C. Practical and conceptual path sampling issues. *European Physical Journal Special Topics* **2015**, *224*, 2409−2427.
(20) Tiwary, P.; Parrinello, M. From metadynamics to dynamics. *Phys. Rev. Lett.* **2013**, *111*, No. 230602.
(21) Piccini, G.; McCarty, J. J.; Valsson, O.; Parrinello, M. Variational flooding study of a SN2 reaction. *J. Phys. Chem. Lett.* **2017**, *8*, 580−583.
(22) Ray, D.; Ansari, N.; Rizzi, V.; Invernizzi, M.; Parrinello, M. Rare event kinetics from adaptive bias enhanced sampling. *J. Chem. Theory Comput.* **2022**, *18*, 6500−6509.
(23) Schäfer, J.-L.; Keller, B. G. Implementation of Girsanov Reweighting in OpenMM and Deeptime. *J. Phys. Chem. B* **2024**, *128*, 6014−6027.
(24) Shmilovich, K.; Ferguson, A. L. Girsanov Reweighting Enhanced Sampling Technique (GREST): On-the-Fly Data-Driven Discovery of and Enhanced Sampling in Slow Collective Variables. *J. Phys. Chem. A* **2023**, *127*, 3497−3517.
(25) Noé, F.; Olsson, S.; Köhler, J.; Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science* **2019**, *365*, No. eaaw1147.
(26) Coretti, A.; Falkner, S.; Weinreich, J.; Dellago, C.; von Lilienfeld, O. A. Boltzmann Generators and the New Frontier of Computational Sampling in Many-Body Systems. *arXiv preprint arXiv:2404.16566*, 2024, DOI: .
(27) Pande, V. S.; Beauchamp, K.; Bowman, G. R. Everything you wanted to know about Markov state models but were afraid to ask. *Methods* **2010**, *52*, 99−105.
(28) Wehmeyer, C.; Scherer, M. K.; Hempel, T.; Husic, B. E.; Olsson, S.; Noé, F. Introduction to Markov State Modeling with the PyEMMA Software. *Living J. Comput. Mol. Sci.* **2019**, *1*, 10607.
(29) Sidky, H.; Chen, W.; Ferguson, A. L. Molecular latent space simulators. *Chemical Science* **2020**, *11*, 9459−9467.
(30) Jones, M. S.; McDargh, Z. A.; Wiewiora, R. P.; Izaguirre, J. A.; Xu, H.; Ferguson, A. L. Molecular latent space simulators for distributed and multimolecular trajectories. *J. Phys. Chem. A* **2023**, *127*, 5470−5490.
(31) Noé, F. Machine learning for molecular dynamics on long timescales. *Lecture Notes in Physics* **2020**, *968*, 331−372.
(32) Fernández, A. Deep learning unravels a dynamic hierarchy while empowering molecular dynamics simulations. *Ann. Phys.* **2020**, *532*, No. 1900526.
(33) Klein, L.; Foong, A.; Fjelde, T.; Mlodozeniec, B.; Brockschmidt, M.; Nowozin, S.; Noe, F.; Tomioka, R. Timewarp: Transferable Acceleration of Molecular Dynamics by Learning Time-Coarsened Dynamics. In *Advances in Neural Information Processing Systems*; 2023; pp. 52863−52883. https://proceedings.neurips.cc/paper-files/paper/2023/file/a598c367280f9054434fdcc227ce4d38-Paper-Conference.pdf (Accessed 11 Oct 2024).
(34) García, A. E. Large-amplitude nonlinear motions in proteins. *Phys. Rev. Lett.* **1992**, *68*, 2696.
(35) Amadei, A.; Linssen, A. B.; Berendsen, H. J. Essential dynamics of proteins. *Proteins: Struct., Funct., Bioinf.* **1993**, *17*, 412−425.
(36) Hegger, R.; Altis, A.; Nguyen, P. H.; Stock, G. How complex is the dynamics of peptide folding? *Phys. Rev. Lett.* **2007**, *98*, No. 028102.
(37) Das, P.; Moll, M.; Stamati, H.; Kavraki, L. E.; Clementi, C. Low-dimensional, free-energy landscapes of protein-folding reactions by nonlinear dimensionality reduction. *Proc. Natl. Acad. Sci. U.S.A.* **2006**, *103*, 9885−9890.
(38) Ferguson, A. L.; Panagiotopoulos, A. Z.; Debenedetti, P. G.; Kevrekidis, I. G. Systematic determination of order parameters for chain dynamics using diffusion maps. *Proc. Natl. Acad. Sci. U.S.A.* **2010**, *107*, 13597−13602.
(39) Chen, W.; Sidky, H.; Ferguson, A. L. Nonlinear discovery of slow molecular modes using state-free reversible VAMPnets. *J. Chem. Phys.* **2019**, *150*, 214114.
(40) Noé, F.; Nüske, F. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Modeling and Simulation* **2013**, *11*, 635−655.
(41) Nuske, F.; Keller, B. G.; Pérez-Hernández, G.; Mey, A. S.; Noé, F. Variational approach to molecular kinetics. *J. Chem. Theory Comput.* **2014**, *10*, 1739−1752.
(42) Szabo, A.; Ostlund, N. S. *Modern Quantum Chemistry: Introduction to advanced electronic structure theory*; Courier Corporation, 2012.
(43) Mori, H. Transport, collective motion, and Brownian motion. *Prog. Theor. Phys.* **1965**, *33*, 423−455.
(44) Zwanzig, R. Nonlinear generalized Langevin equations. *J. Stat. Phys.* **1973**, *9*, 215−220.
(45) Zwanzig, R. *Nonequilibrium Statistical Mechanics*; Oxford University Press: Oxford, 2001.
(46) Risken, H.; Frank, T. *The Fokker-Planck Equation: Methods of Solution and Applications*, 2nd ed.; Springer Verlag: Berlin Heidelberg New York, 2012.
(47) Fu, X.; Xie, T.; Rebello, N. J.; Olsen, B. D.; Jaakkola, T. Simulate time-integrated coarse-grained molecular dynamics with geometric machine learning. *arXiv preprint arXiv:2204.10348*, 2022. DOI: .
(48) Lin, K.; Peng, J.; Xu, C.; Gu, F. L.; Lan, Z. Realization of the trajectory propagation in the MM-SQC dynamics by using machine learning. *arXiv preprint arXiv:2207.05556*, 2022. DOI: .
(49) Tsai, S.-T.; Kuo, E.-J.; Tiwary, P. Learning molecular dynamics with simple language model built upon long short-term memory neural network. *Nat. Commun.* **2020**, *11*, 5115.
(50) Vlachas, P. R.; Zavadlav, J.; Praprotnik, M.; Koumoutsakos, P. Accelerated simulations of molecular systems through learning of effective dynamics. *J. Chem. Theory Comput.* **2022**, *18*, 538−549.
(51) Wu, H.; Noé, F. Reaction coordinate flows for model reduction of molecular kinetics. *J. Chem. Phys.* **2024**, *160*, No. 044109.
(52) Mardt, A.; Pasquali, L.; Wu, H.; Noé, F. VAMPnets for deep learning of molecular kinetics. *Nat. Commun.* **2018**, *9*, 5.
(53) Bonati, L.; Piccini, G.; Parrinello, M. Deep learning the slow modes for rare events sampling. *Proc. Natl. Acad. Sci. U.S.A.* **2021**, *118*, No. e2113533118.
(54) Pérez-Hernández, G.; Paul, F.; Giorgino, T.; De Fabritiis, G.; Noé, F. Identification of slow molecular order parameters for Markov model construction. *J. Chem. Phys.* **2013**, *139*, No. 015102.
(55) Schwantes, C. R.; Pande, V. S. Improvements in Markov state model construction reveal many non-native interactions in the folding of NTL9. *J. Chem. Theory Comput.* **2013**, *9*, 2000−2009.
(56) Molgedey, L.; Schuster, H. G. Separation of a mixture of independent signals using time delayed correlations. *Phys. Rev. Lett.* **1994**, *72*, 3634−3637.
(57) Andrew, G.; Arora, R.; Bilmes, J.; Livescu, K. Deep canonical correlation analysis, In *Proceedings of the 30th International Conference on Machine Learning*; Dasgupta, S.; McAllester, D., Eds.; 2013; Vol. *28*; pp. 1247−1255. https://proceedings.mlr.press/v28/andrew13.html (Accessed 11 Oct 2024).
(58) Klus, S.; Nüske, F.; Koltai, P.; Wu, H.; Kevrekidis, I.; Schütte, C.; Noé, F. Data-driven model reduction and transfer operator approximation. *J. Nonlinear Sci.* **2018**, *28*, 985−1010.

(59) Koltai, P.; Wu, H.; Noé, F.; Schütte, C. Optimal data-driven estimation of generalized Markov state models for non-equilibrium dynamics. *Computation* **2018**, *6*, 22.
(60) Wu, H.; Noé, F. Variational ppproach for learning Markov processes from time series data. *J. Nonlinear Sci.* **2020**, *30*, 23−66.
(61) McCandlish, D. M. Visualizing fitness landscapes. *Evolution* **2011**, *65*, 1544−1558.
(62) Herringer, N. S.; Dasetty, S.; Gandhi, D.; Lee, J.; Ferguson, A. L. Permutationally Invariant Networks for Enhanced Sampling (PINES): Discovery of multimolecular and solvent-inclusive collective variables. *J. Chem. Theory Comput.* **2024**, *20*, 178−198.
(63) Hempel, T.; Olsson, S.; Noé, F. Markov field models: Scaling molecular kinetics approaches to large molecular machines. *Curr. Opin. Struct. Biol.* **2022**, *77*, No. 102458.
(64) Mardt, A.; Hempel, T.; Clementi, C.; Noé, F. Deep learning to decompose macromolecules into independent Markovian domains. *Nat. Commun.* **2022**, *13*, 7101.
(65) Bishop, C. M. Mixture density networks. *Neural Computing Research Group Report*, **1994**, pp. 171−198.
(66) Bishop, C. M. *Pattern Recognition and Machine Learning*; Springer-Verlag: New York, Inc.: Secaucus, NJ, USA, 2006.
(67) Pathak, J.; Hunt, B.; Girvan, M.; Lu, Z.; Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.* **2018**, *120*, 24102.
(68) Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017; pp. 214−223. https://proceedings.mlr.press/v70/arjovsky17a.html (Accessed 11 Oct 2024).
(69) Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. C. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, 2017, https://proceedings.neurips.cc/paper-files/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf (Accessed 11 Oct 2024).
(70) Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. DOI: .
(71) Musil, F.; Grisafi, A.; Bartók, A. P.; Ortner, C.; Csányi, G.; Ceriotti, M. Physics-inspired structural representations for molecules and materials. *Chem. Rev.* **2021**, *121*, 9759−9815.
(72) Braams, B. J.; Bowman, J. M. Permutationally invariant potential energy surfaces in high dimensionality. *Int. Rev. Phys. Chem.* **2009**, *28*, 577−606.
(73) Xie, Z.; Bowman, J. M. Permutationally invariant polynomial basis for molecular energy surface fitting via monomial symmetrization. *J. Chem. Theory Comput.* **2010**, *6*, 26−34.
(74) Behler, J.; Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* **2007**, *98*, No. 146401.
(75) Behler, J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *J. Chem. Phys.* **2011**, *134*, No. 074106.
(76) Huo, H.; Rupp, M. Unified representation of molecules and crystals for machine learning. *Machine Learning: Science and Technology* **2022**, *3*, No. 045017.
(77) Gu, C.; Chang, H.-W.; Maibaum, L.; Pande, V. S.; Carlsson, G. E.; Guibas, L. J. Building Markov state models with solvent dynamics. *BMC Bioinformatics* **2013**, *14*, S8.
(78) Harrigan, M. P.; Shukla, D.; Pande, V. S. Conserve water: A method for the analysis of solvent in molecular dynamics. *J. Chem. Theory Comput.* **2015**, *11*, 1094−1101.
(79) Raddi, R.; Voelz, V. A Markov state Model of solvent features reveals water dynamics in protein-peptide binding. *ChemRxiv preprint 10.26434/chemrxiv-2023-r6njk*, 2023, DOI: .
(80) Bartók, A. P.; Kondor, R.; Csányi, G. On representing chemical environments. *Phys. Rev. B* **2013**, *87*, No. 184115.
(81) Ho, J.; Jain, A.; Abbeel, P. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, 2020; pp. 6840−6851. https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html (Accessed 11 Oct 2024).
(82) Wang, Y.; Herron, L.; Tiwary, P. From data to noise to data for mixing physics across temperatures with generative artificial intelligence. *Proc. Natl. Acad. Sci. U. S. A.* **2022**, *119*, No. e2203656119.
(83) Jones, M. S.; Shmilovich, K.; Ferguson, A. L. DiAMoNDBack: Diffusion-Denoising Autoregressive Model for Non-Deterministic Backmapping of Cα protein traces. *J. Chem. Theory Comput.* **2023**, *19*, 7908−7923.
(84) Shmilovich, K.; Stieffenhofer, M.; Charron, N. E.; Hoffmann, M. Temporally coherent backmapping of molecular trajectories from coarse-grained to atomistic resolution. *J. Phys. Chem. A* **2022**, *126*, 9124−9139.
(85) Dobers, S.; Stark, H.; Fu, X.; Beaini, D.; Günnemann, S. Latent Space Simulator for Unveiling Molecular Free Energy Landscapes and Predicting Transition Dynamics. In *NeurIPS 2023 AI for Science Workshop*, 2023. https://openreview.net/forum?id=XIxcglPy9c (Accessed 11 Oct 2024).
(86) Sarisky, C. A.; Mayo, S. L. The *ββα* fold: explorations in sequence space. *J. Mol. Biol.* **2001**, *307*, 1411−1418.
(87) Hoffmann, M.; Scherer, M.; Hempel, T.; Mardt, A.; de Silva, B.; Husic, B. E.; Klus, S.; Wu, H.; Kutz, N.; Brunton, S. L.; Noé, F. Deeptime: a Python library for machine learning dynamical models from time series data. *Machine Learning: Science and Technology* **2022**, 3, No. 015009.
(88) Schütt, K. T.; Sauceda, H. E.; Kindermans, P.-J.; Tkatchenko, A.; Müller, K.-R. Schnet−a deep learning architecture for molecules and materials. *J. Chem. Phys.* **2018**, *148*, 241722.