Variational encoding of complex dynamics

Carlos X. Hernández, Hannah K. Wayment-Steele, Mohammad M. Sultan, Brooke E. Husic, and Vijay S. Pande

Phys. Rev. E **97**, 062412 — Published 18 June 2018

DOI: 10.1103/PhysRevE.97.062412

Variational Encoding of Complex Dynamics

Carlos X. Hernández ^{*1}**, Hannah K. Wayment-Steele** ^{*2}**, Mohammad M. Sultan** ^{*2}**, Brooke E. Husic**^{2}**, and Vijay S. Pande** $^{\dagger 1,2}$

[1]Biophysics Program, Stanford University, California, USA
[2]Chemistry Department, Stanford University, California, USA

ABSTRACT

Often the analysis of time-dependent chemical and biophysical systems produces high-dimensional time-series data for which it can be difficult to interpret which individual features are most salient. While recent work from our group and others has demonstrated the utility of time-lagged covariate models to study such systems, linearity assumptions can limit the compression of inherently nonlinear dynamics into just a few characteristic components. Recent work in the field of deep learning has led to the development of the variational autoencoder (VAE), which is able to compress complex datasets into simpler manifolds. We present the use of a time-lagged VAE, or variational dynamics encoder (VDE), to reduce complex, nonlinear processes to a single embedding with high fidelity to the underlying dynamics. We demonstrate how the VDE is able to capture nontrivial dynamics in a variety of examples, including Brownian dynamics and atomistic protein folding. Additionally, we demonstrate a method for analyzing the VDE model, inspired by saliency mapping, to determine what features are selected by the VDE model to describe dynamics. The VDE presents an important step in applying techniques from deep learning to more accurately model and interpret complex biophysics.

1 Introduction

Simulations of biomolecules have provided insight into molecular processes with increasing time- and length-scales due to advances in both algorithms[1] and hardware[2]. Such simulations can have thousands of degrees of freedom, making it crucial to have meaningful and statistically robust methods to extract underlying dynamical processes[3].

The dynamics of molecular systems are often represented using the dynamical propagator approach[4]. Given an ensemble of particles at time t distributed in phase space with a given probability distribution $p(x,t)$, we seek to describe a propagator as an operator that can describe the new distribution of the ensemble, $p(x,t+\tau)$, given some lag time τ. When τ is chosen such that these probabilities are independent of the history of the system, the model is said to be Markovian.

The Markovian propagator contains all the information needed to propagate the system forward in time. To make a Markov state model (MSM) from a biomolecular simulation, each frame in the time-series dataset is assigned to a tractable number of discrete states[5]. The transition matrix stores the conditional transition probabilities between all pairs of states at the specified lag time. This transition matrix is also constrained to obey microscopic reversibility and ergodicity. Due to these constraints, the eigenvalues are real with a unique highest Perron eigenvalue of one and all subsequent eigenvalues with absolute values smaller than one. The non-Perron eigenvalues and their eigenfunctions correspond to processes in the time-series, representing their timescales and interstate fluxes, respectively.

In 2013, the derivation of a variational approach to conformational dynamics (VAC)[6] showed that estimates of MSM eigenvalues cannot exceed their true values. Thus, the choice of MSM states can be optimized according to this variational principle. In fact, the VAC applies in a more general case than MSMs: the eigenfunction approximations need not come from the discrete state decomposition that characterizes a MSM, but rather can come from other features; in the case of protein dynamics, these features might represent torsional angles or pairwise distances between amino acids. In this spirit, many methods have been developed to compute approximations to the propagator of a molecular system from simulation data, including time-structure-based independent component analysis (tICA)[7–9] and extensions (kernel tICA[10], sparse tICA[11]), VAMPnets[12], soft-max MSMs[13], and diffusion maps[14]. These methods make different assumptions about the underlying eigenfunctions of the system; for example, approximating the eigenfunctions of the propagator under the constraint of linear combinations of features produces tICA[8], which can be further enhanced as a nonlinear approximation via the kernel trick[10].

Any method for approximating the dynamics of a complex system has two objectives: to adequately represent complexity in the form of model nonlinearity and to be interpretable, that is, to be readily analyzable for feature importance. In Figure

*These authors contributed equally to this work
†Corresponding author: pande@stanford.edu

1, we indicate how several commonly-used methods for dimensionality reduction of dynamical systems compare in terms of achieving these two aims. Complexity and interpretability often come at the expense of each other. For instance, kernel methods such as kernel tICA[10,15] improve the ability to capture nonlinear effects of features in dynamics over linear methods; however, identifying biophysical meaning in coordinates in an implicit kernel space remains a challenge. Conversely, standard tICA and sparse tICA allow for more precisely identifying relevant biophysical features, but the linearity of tICA limits the complexity of dynamics it can represent.

An alternative technique for dimensionality reduction is the autoencoder framework[16,17]. An autoencoder is a deep unsupervised learning algorithm that aims to learn a low-dimensional representation of high-dimensional data[18,19]. An autoencoder has two components: an encoder network and a decoder network. The encoder network reduces the input data to a low-dimensional representation, referred to as the latent space of the autoencoder, and the decoder network reconstructs the latent representation to the original dimensionality. The difference between the original data and the reconstruction is used to update and train the network. variational autoencoder (VAE) adds regularization to the encoder framework by applying Gaussian noise to the latent space[20]. The term "variational" stems from this stochasticity: the autoencoder is an implementation of variational Bayesian inference with a Gaussian prior, which maximizes the lower bound on the log-likelihood of the observed data[16].

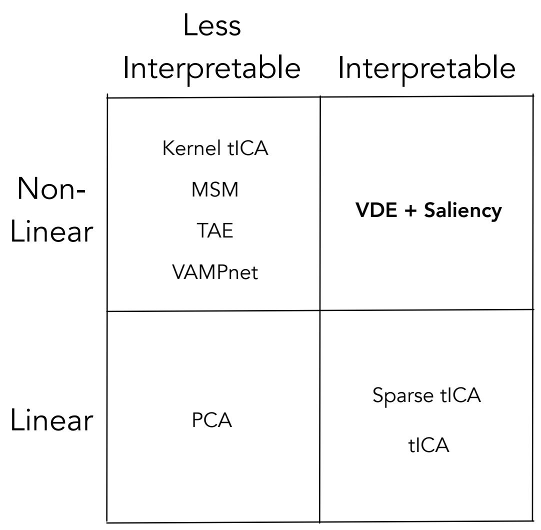

Figure 1. An overview of a subset of the methods used to analyze protein dynamics in terms of model interpretability and ability to capture non-linear motions[7–10,15,21–24]. Here, we define interpretability as the ease with which a scientist can analyze the model for feature importance with respect to dynamics. For example, principal component analysis (PCA), arguably the simplest model mentioned, is typically ill-suited to analyze complex dynamics and, therefore, the resulting principal components are not reliably meaningful. In contrast, the VDE is able to leverage deep learning to model non-linear relationships between time-dependent observables and saliency mapping to understand which observables contribute most to the model. We note that saliency mapping is a general technique for analyzing neural networks and can also be applied to related methods.

Recently, the autoencoder framework has been extended to model time-series data[24–30]. Analysis in these applications typically involves mapping time-series data to latent spaces with the same dimensionality as the length of the initial time-series data and has not focused on approximating a propagator for the time-series data; however, there are a couple of notable exceptions. Doerr and De Fabritis[30] recently compared a simple autoencoder to other methods for dimensionality reduction of biophysical simulation data. Wehmeyer and Noé introduced a time lag into an autoencoder (TAE) framework to describe dynamics[24]. Interestingly, they demonstrate that in the limit of a single linear hidden layer, the tICA solution can be attained.

In this work, we extend the traditional VAE architecture to approximate a propagator for time-series data in an architecture denoted as a variational dynamics encoder (VDE). This represents the first use of a time lag within a variational autoencoder to our knowledge. Additionally, we introduce a novel "autocorrelation" loss function, which is inspired by the VAC[6]. We demonstrate that this approach yields models with more explanatory power than linear dimensionality reduction techniques in both the Müller-Brown potential and the folding landscape of the villin headpiece subdomain. We also explore the generative capability of the VDE as a propagator of dynamics and show that, as implemented, it is unable to reliably capture thermodynamics at differing temperatures. Finally, we demonstrate a novel analysis method, inspired by saliency mapping in neural nets for visual classification[31–33], to lend interpretation to VDE models. This combination of using the VDE with saliency mapping creates a framework that enables nonlinear combinations of features while remaining interpretable.

2 Model: Variational Dynamics Encoder (VDE)

2.1 VDE Architecture

The architecture of the VDE, as seen in Figure 2, closely resembles that of a VAE; however, the training procedure is slightly modified to suit time-series data[24,30]. The most significant modification being that featurized data, x_t, at some timepoint, t, are fed into the network in order to make a prediction of the state of the system, $x'_{t+\tau}$, at a future timepoint, $t+\tau$, where τ is some user-selected lag time such that the dynamics of the system is Markovian. We note that choosing a lag time at which the system is Markovian depends entirely on the system is being modeled[34]. At a long enough lag time for the system to be approximated as a Markov process, intrastate transitions occur much more quickly than interstate transitions. The appropriate lag time depends on the system of study: for protein folding, tens of nanoseconds might be appropriate; for electron dynamics, a suitable lag time might be on the order of femtoseconds. If a system is Markovian at a lag time τ (if the intrastate transitions occur more quickly than τ), then the system will be Markovian at all lag times greater than τ and the timescales of the subprocesses will be constant for all Markovian lag times. Therefore, τ should be large enough to achieve Markovianity, but small enough to make statistically significant observations of the system's dynamics.

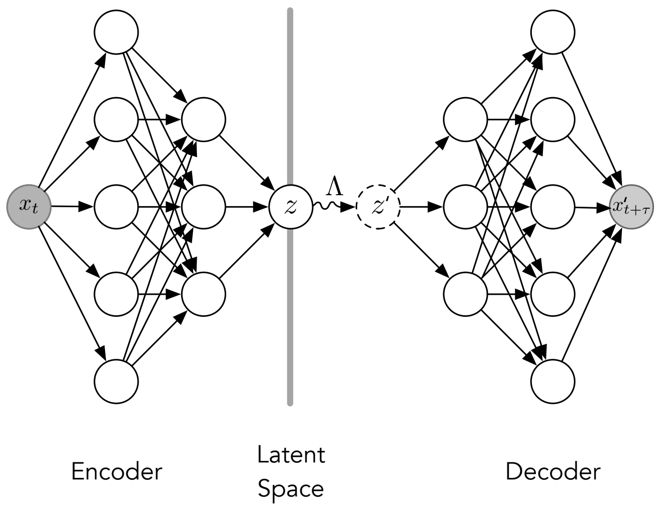

Figure 2. A schematic of the VDE. Features, x_t, at some timepoint, t, are fed into the network in order to make a prediction of the state of the system, $x'_{t+\tau}$, at a future timepoint, $t+\tau$, where τ is some Markovian lag time. As with a traditional VAE, the network can be subdivided into three parts: the encoder network; variational layer, Λ; and the decoder network, as labeled. Our encoder network is a DNN with non-linear activation functions in the hidden layers, which eventually bottlenecks into the one-dimensional latent space, z_t. The latent space is then slightly perturbed with Gaussian noise by the Λ-layer to generate z', as described by Kingma and Welling[16]. Finally, the decoder network, also a DNN, mirrors the encoder network in architecture by using z' to generate $x'_{t+\tau}$, a prediction of how the system will evolve after one lag time of τ.

As with a traditional VAE, the network can be subdivided into three parts: the encoder network; variational layer, Λ; and the decoder network. The encoder network is a deep neural network (DNN) with non-linear activation functions and a user-selected number of hidden layers, which eventually bottlenecks into the one-dimensional latent space, z_t. In this way, the encoder network functions as a non-linear dimensionality reduction of x_t. The latent space is then perturbed by Gaussian noise within the Λ-layer, with mean parameter, μ, variance parameter, σ^2, and arbitrary scaling, α, to generate z', as described by Kingma and Welling[16]. Finally, the decoder network, also a DNN, mirrors the encoder network in architecture by using z' to generate $x'_{t+\tau}$, a prediction of how the system will evolve after a duration of τ.

Once the VDE has been trained, it can be used for both dimensionality reduction and synthetic trajectory generation. During dimensionality reduction, only the encoder network is necessary, which provides a direct mapping of $x \mapsto z$. During trajectory generation, the entire VDE network is needed. An initial set of features, x_0, is fed through the network to generate x'_τ, the predicted state after a duration of τ. This can be done iteratively to generate an arbitrarily long trajectory of features exhibiting dynamics consistent with that of the original system used during training. In order to overcome the model's insensitivity to the Λ-layer used during training, we recommend that the noise factor, α, is increased such that $\alpha_{\text{generation}} \gg \alpha_{\text{train}}$.

2.2 VDE Loss Function

The VDE is quantitatively evaluated by calculating the sum of three loss functions: reconstruction loss ($\mathscr{L}_{\mathrm{R}}$), Kullback–Leibler divergence loss ($\mathscr{L}_{\mathrm{KL}}$), and autocorrelation loss ($\mathscr{L}_{\mathrm{AC}}$):

$$\mathscr{L}_{\mathrm{VDE}} = \mathscr{L}_{\mathrm{R}} + \mathscr{L}_{\mathrm{KL}} + \mathscr{L}_{\mathrm{AC}}. \tag{1}$$

The first of these three, reconstruction loss, attempts to quantify how well the VDE approximates the state of the system at $t+\tau$, given the true state of the system at time t[16,20]. In doing so, we evaluate the ability of the network to approximate the Markovian propagator after a single lag time. This can be done by considering the mean squared error between the predicted propagation, $x'_{t+\tau}$, and the true propagation, $x_{t+\tau}$:

$$\mathscr{L}_{\mathrm{R}} = \mathbb{E}\left[\left\|x'_{t+\tau} - x_{t+\tau}\right\|\right]. \tag{2}$$

The Kullback–Leibler divergence loss allows for variational inference of the latent space and considers the latent space priors that generate z'_t:

$$\mathscr{L}_{\mathrm{KL}} = \mathbb{E}\left[\frac{1 + \log\sigma(z_t)^2 - \mu(z_t)^2 - \sigma(z_t)^2}{2}\right], \tag{3}$$

where μ and σ are separate affine transformations that estimate the mean and standard deviation, respectively, of the Gaussian prior locally applied to the latent space, as seen in Figure 3. Coupled with the reconstruction loss, the Kullback–Leibler divergence enables a trade-off between model complexity and simplicity of the Gaussian prior. Reconstruction loss pushes the model towards having high fidelity to the training data, while the Kullback–Leibler divergence acts as a regularization term to ensure that the latent space behaves as a Gaussian emission[16]. This scheme also has the benefit of allowing for sampling of the latent space, using the same priors, to generate new trajectories as mentioned in Section 2.1.

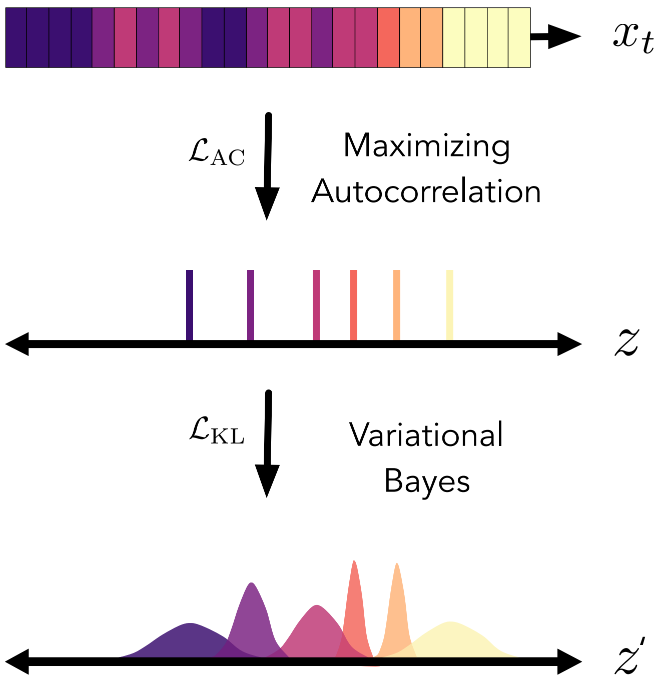
x_t
$\mathcal{L}_{\mathrm{AC}}$
Maximizing Autocorrelation
z
$\mathcal{L}_{\mathrm{KL}}$
Variational Bayes
z'

Figure 3. A diagram representing the effects of the autocorrelation loss and KL-divergence loss functions on the VDE latent space. A trajectory, x, which contains several states represented by different colors, can be mapped onto a latent space, z. Here, we randomly select frames from x from each state to be mapped onto z. Maximization of the autocorrelation of z ensures that the slowest process within a trajectory can be modeled continuously within the latent space. Perturbation of the latent space with learnable, value-specific Gaussian noise enables a variational Bayes approach for propagating values from the original trajectory, x. In doing so, we are also able to infer the posterior probability of values within z and effectively perform sampling within the latent space.

Although minimization of the reconstruction loss has the potential to recover these dynamical processes[23], we find that in some cases, such as in Section 4.3, it alone is not sufficient[35]. In order to improve model convergence, we borrow from the VAC[6], a specific application of the variational principle from quantum mechanics adapted for Markov modeling and a useful tool for parameter selection. The variational principle states that, in the limit of infinite data, no process can be estimated from the data that is slower than the true process. If we interpret the variational principle as the measure of the quality of this approximation, the phase-space decomposition that leads to a linear model with larger leading dynamical eigenvalues is consequently the better phase-space decomposition[11]. In the limit of a single process which determines the dynamics of a system, there is only one eigenvalue to consider, which is equivalent to the autocorrelation of the decomposed trajectories. We propose that maximizing the autocorrelation of z optimizes training towards generating a more complete representation of the long time-scale kinetics observed within time-series. The autocorrelation loss, $\mathscr{L}_{\mathrm{AC}}$, takes the following form:

$$\mathscr{L}_{\mathrm{AC}} = -\rho_{z_t,z_{t+\tau}} = -\frac{\mathbb{E}\left[\left(z_t-\bar{z}_t\right)\left(z_{t+\tau}-\bar{z}_{t+\tau}\right)\right]}{s_{z_t}s_{z_{t+\tau}}}, \tag{4}$$

where $\bar{z}_t$ and s_{z_t} are the sample mean and standard deviation of the latent space for a particular batch of data, respectively. For linear models, this leads only to a first-order approximation of slowest process; however, by incorporating this into the VDE's loss function, we take advantage of the deep encoder as a general approximator to the slowest processes found within in our data[6].

When we consider that z is a rich latent observable of the true dynamics, its autocorrelation also represents a weighted sum of the all the dynamical eigenvalues of the system[36]:

$$\rho_{z_t,z_{t+\tau}} = \sum_{ij}\bar{z}_i P\left(X_t=i\right)\bar{z}_j P\left(X_{t+\tau}=j|X_t=i\right) = \sum_{ij}\bar{z}_i\bar{z}_j\pi_i T_{ij} = \sum_{ijk}\bar{z}_i\bar{z}_j\lambda_k(\phi_k)_i(\phi_k)_j = \sum_k \lambda_k\omega_k^2, \tag{5}$$

where X_t is some latent state of the system at time, t; π is the stationary distribution of X; T is the transition matrix; and λ and ϕ are eigenvalues and eigenvectors of T. ω_k represents the inner product between the observable and the k-th eigenvector. This ω^2-weighted sum is closely related to the generalized matrix Rayleigh quotient (GMRQ), which is calculated as an unweighted sum of leading dynamical eigenvalues and can be used as a scoring metric for cross-validating Markovian models[11]. Through the optimization of $\rho_{z_t,z_{t+\tau}}$, we implicitly maximize the GMRQ for our model.

Algorithm 1 outlines how these two losses are calculated and used for backpropagation in practice. Note that the data is split into many smaller batches during training, with x_t as input variable and $x_{t+\tau}$ as the target variable, to take advantage of stochastic gradient descent methods. We also recommend pre-processing features—either via standardization or median-centering and scaling by interquartile ranges—to prevent the reconstruction loss from overpowering the autocorrelation loss[37].

3 Methods

3.1 Müller-Brown Potential

We first test the VDE as a proof-of-concept in characterizing Brownian dynamics under the Müeller-Brown potential, a well-studied smooth two-dimensional potential energy surface. We generated 10 independent simulations of the 2-D Müller-Brown potential governed by the following equation:

$$\dot{\mathbf{x}} = -\frac{\Delta V\left(\mathbf{x}\right)}{kT} + \sqrt{2D}R\left(t\right),$$

where $k_BT = 1.5\times10^4$ Joules, $D = 10^{-2}$ meters-squared per second, and $R\left(t\right)$ is a delta-correlated Gaussian process with zero mean, and $V\left(\mathbf{x}\right)$ is defined as:

$$V\left(\mathbf{x}\right) = \sum_{j=1}^{4} A_j\cdot\exp\left[a_j\left(x_1-X_j\right)^2 + b_j\left(x_1-X_j\right)\left(x_2-Y_j\right) + c_j\left(x_2-Y_j\right)^2\right],$$

where $a = (-1,-1,-6.5,0.7)$; $b = (0,0,11,0.6)$; $c = (-10,-10,-6.5,0.7)$; $A = (-200,-100,-170,15)$; $X = (1,0,-0.5,-1)$; and $Y = (0,0.5,1.5,1)$ as suggested by Müller and Brown[38]. Using the Euler-Maruyama method for numerical integration and a time step of 0.1, we produced ten unique trajectories with 10^6 time steps, saved every 100 steps. The initial positions were sampled via a uniform distribution over the box: $[-1.5,1.2]\times[-0.2,2.0]$.

VDEs for the Müller-Brown potential were trained with a lag time of 10 time steps; 3 hidden layers with 256 nodes each; the Swish activation function[39]; α-value of 10^{-3}; batch size of 100; dropout rate of 30%; and a learning rate of 1×10^{-4}. We note that these parameters were not optimized using automated hyperparameter selection. Gradient descent was performed with the Adam optimizer[40]. Models were trained for 50 epochs, at which point the losses were observed to be converged. Prior to training, trajectories were preprocessed by subtracting their overall median values and scaling by inter-quartile ranges.

In constructing MSMs for the Müller-Brown potential, the scaled trajectories were then subject to dimensionality reduction using principal component analysis (PCA)[21], time-structure-based independent component analysis (tICA)[7–9], and the pre-trained VDE, in order to generate one-dimensional representations of the system's dynamics. We then partitioned the representations into twelves clusters using the mini-batch k-means algorithm[41,42]. Finally, the clusters were used to construct a maximum-likelihood estimated reversible MSM[43]. A lag time of 10 time steps was chosen for both MSM construction and dimensionality reduction, as the resulting models provided optimal convergence of implied timescales. The MSMs were then evaluated 100 randomly seeded hold-out datasets to generate unbiased GMRQ scores and standard errors. All trajectory generation and analyses were performed with MSMBuilder[37], MDEntropy[44], and MSMExplorer[45].

Finally, in order to generate "fake" trajectories using the VDE, we randomly sampled initial positions via a uniform distribution, as described above, and iteratively propagated these coordinates through the VDE for 1,000 steps, equivalent to 10,000 integrator steps. This was done for five scaling values, α, evenly sampled in logspace between 10^{-2} and 10^{-1} to understand how the Λ-layer affects propagation.

Algorithm 1 Training the VDE

```
1:  procedure TRAIN(model, data)
2:      for batch ∈ data do
3:          $x_t, x_{t+\tau} \leftarrow$ batch
4:          $z_t \leftarrow$ model.encode($x_t$)
5:          $z'_t \leftarrow$ model.lambda($z_t$)
6:          $x'_{t+\tau} \leftarrow$ model.decode($z'_t$)
7:          $z_{t+\tau} \leftarrow$ model.encode($x_{t+\tau}$)
8:
9:          $\mu_t, \sigma_t \leftarrow$ model.lambda.parameters
10:
11:         $\mathscr{L}_{\mathrm{R}} \leftarrow \mathbb{E}\left[\left|\left|x'_{t+\tau} - x_{t+\tau}\right|\right|\right]$
12:         $\mathscr{L}_{\mathrm{KL}} \leftarrow \mathbb{E}\left[\frac{1+\log\sigma_t^2-\mu_t^2-\sigma_t^2}{2}\right]$
13:         $\mathscr{L}_{\mathrm{AC}} \leftarrow -\rho_{z_t, z_{t+\tau}}$
14:
15:         model.loss $\leftarrow \mathscr{L}_{\mathrm{R}} + \mathscr{L}_{\mathrm{KL}} + \mathscr{L}_{\mathrm{AC}}$
16:
17:         model.loss.backward()
18:         model.optimizer.step()
```

3.2 Villin Headpiece Domain

We demonstrate the utility of the VDE method in characterizing the folding landscape of villin headpiece domain (pdb: 2f4k), a widely-studied 35-residue fast-folding protein, referred to henceforth as villin. Simulation data for villin was generated by Lindorff-Larsen et al.[46]. The simulation length was is 125 μs and is strided at 2 ns for analysis. Cα contact distances were used for featurization[37,47]. VDEs for villin were trained with a lag time of 44 ns, selected to be the same as that in the optimal tICA model. Other than expanding the number of hidden layers nodes to 1024, the training procedure was identical to that of Section 3.1. The VDE was compared to an optimized tICA model, with respect to MSM GMRQ scoring, for villin, as featurized with Cα contacts, that was identified via hyperparameter optimization[48]. Husic et al.[48] have indicated that Cα contacts are a useful featurization for representing folding processes[48], hence the selection of this featurization. However, using $\phi - \psi$ backbone dihedral angles for featurization results in a VDE model with a latent space that is highly correlated with the VDE latent space featurized by Cα contacts (Figure S1), indicating the robustness of the VDE dimensionality reduction process. For the optimized tICA model, a tICA lag time of 44 ns, 4 tICA components, and kinetic mapping[49] were selected according to hyperparameter optimization[48]. To construct MSMs on tICA-transformed and VDE-transformed data, analogous steps as for the Müller potential in Section 3.1 were performed. Mini-batch k-means clustering was performed with 125 clusters for both sets of data. This was the optimal cluster number identified for tICA, and hyperparameter searching showed little influence of cluster number on MSMs from VDE-transformed data. MSMs for both tICA- and VDE- transformed models were constructed with 50 ns lag time. To obtain error estimates for MSM equilibrium populations and MSM timescales, 100 rounds of bootstrapping were performed over the original set of trajectories. The resulting ranges of values were used for error bars.

3.3 Protein Saliency Maps

Saliency maps were designed for classification algorithms and thus needed to be modified for our application (Algorithm 2). Briefly, we first generate a faux two-step trajectory starting from a random protein conformation, for instance, a misfolded state, and going to the desired protein conformation, for instance, the folded state. The misfolded state is propagated through the network, and the residual to the folded state (Figure 7a) is propagated back to obtain loadings on individual distances. This back-propagation is just the chained partial derivative of the reconstruction error with respect to the atomic coordinates of the system.

Ideally, we perform this process for a large number of possible misfolded to folded transitions to obtain robust saliency maps. The median values for each feature across all of these maps can then be integrated to obtain residue level statistics or rank ordered to find important features. It is worth noting that our method is different from classical saliency scoring whereby only the desired class label score is propagated backwards.

Algorithm 2 Computing saliency maps

```
1: procedure SALIENCY(model, data)
2:     $x_t, x_{t+\tau} \leftarrow$ data
3:     $x'_{t+\tau} \leftarrow$ model.forward($x_t$)
4:     model.loss $= \|x'_{t+\tau} - x_{t+\tau}\|$
5:     model.loss.backward()
6:     return $\frac{\partial \text{model.loss}}{\partial x_t}$
```

We note that both the VDE's noise parameter and the autocorrelation loss should be set to zero for consistent results and numerical stability. We also recommend computing the saliency scores multiple times across many configurations and averaging out the results. Lastly, we note that the protein saliency maps can be used in a variety of different protein deep learning algorithms, including VAMPnets[12] and TAE[23].

4 Results

4.1 A Non-Linear Encoding for Brownian Dynamics

We first apply the VDE framework to the well-studied 2-D Müller-Brown potential and demonstrate it can adequately describe the dynamics of this simple system. Figure 4 shows results for a) the VDE, b) tICA, and c) PCA. We note that while tICA and PCA both identify the same dominant linear coordinate, representing diffusion from minor to major basin, the VDE generates a non-linear projection that is able to distinguish these basins more clearly, as well as the transition region.

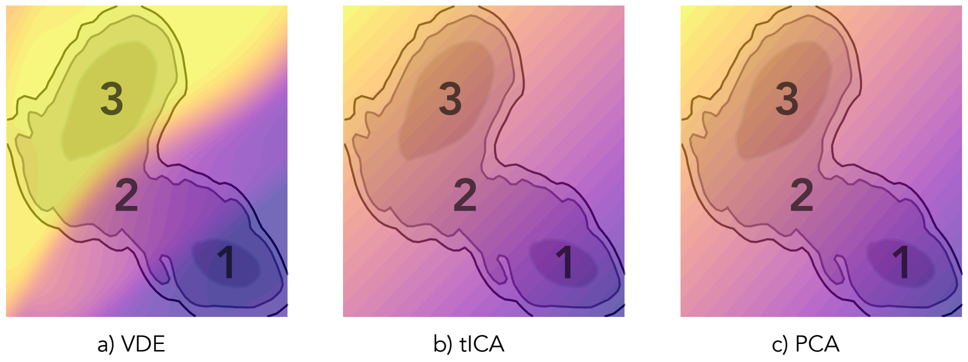

Figure 4. The 2-D Müller-Brown potential (gray-scale contours) overlaid with colormap projections of the one-dimensional a) VDE, b) tICA, and c) PCA coordinates. While tICA and PCA identify a strictly linear mode that approximates the slowest dynamical process (i.e. diffusion from region 1 to 3), the non-linear VDE is better able to map out basins (regions 1 and 3) and intermediate state (region 2). Note that because the region outside of the contours is energetically unfavorable, the color projections in that space are extrapolations of each method, respectively.

To establish an unbiased assessment of the VDE's performance compared to tICA or PCA, we measure its ability to represent the original trajectories, as well as its ability to capture slow-timescale dynamics. In order to measure the former, we employ mutual information to understand exactly how many more bits of information is shared between the VDE latent space and the original features than the principal components of PCA and tICA; as shown in Supplementary Materials Figure S1, we find that it shares more than twice as many bits of information with the original features than the linear methods do. To address the latter, we constructed MSMs using identical hyperparameters and compared GMRQ scores of the slowest process. The VDE achieves a slightly higher mean GMRQ score ($1.8580 \pm 5 \times 10^{-4}$) than tICA ($1.8460 \pm 5 \times 10^{-4}$) or PCA ($1.8472 \pm 5 \times 10^{-4}$) on held-out data. This, along with the mutual information results, suggests that the VDE is better able to represent the dynamics of this system.

4.2 The VDE Does Not Behave as a True Propagator

As VAEs are regarded as a generative model, we consider the relationship between the VDE and the propagator function. When trained on the Müller-Brown potential, with $k_B T = 1.5 \times 10^4$ Joules, the VDE is able to generate "fake" trajectories with some similarities in dynamics to the original simulations, as seen in Figure 5 (pink curve). Furthermore, when we modulate the effect of the Λ-layer by adjusting the scaling parameter α, we are also able to mimic some effects of changing the simulation temperature without having to re-train the VDE. Figure 5b demonstrates that when decreasing (dark blue and purple curves) or increasing (orange and yellow curves) α, the VDE is able to adjust barrier heights in a similar fashion to what is observed in simulation, shown in Figure 5a. However, we find that this fidelity to simulation is lacking in transition regions and previously unobserved regions of phase-space, where the VDE does a poor job of recapitulating true thermodynamics.

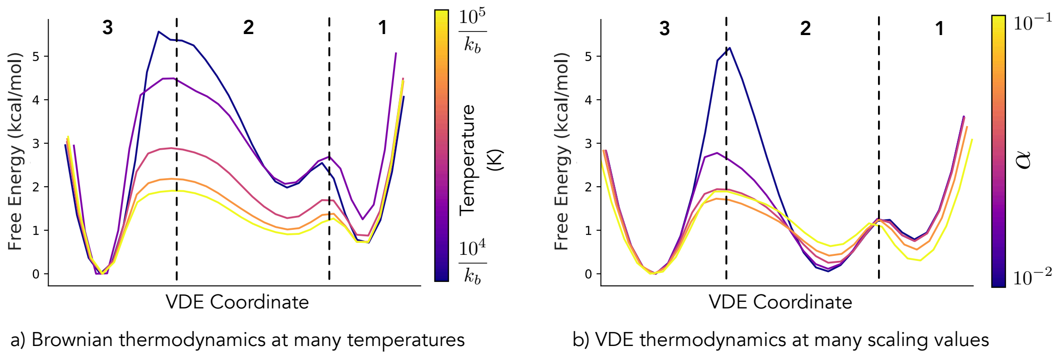

Figure 5. One-dimensional free energy projections generated from the VDE coordinate for a) true Brownian-dynamics simulations at different temperatures and b) fake trajectories generated by the VDE, trained at $k_B T = 1.5 \times 10^4$ Joules, with different scaling values, α, a proxy for temperature, within the Λ-layer. Although not a true one-to-one comparison, we find that free energy barriers (between regions 1, 2, and 3) are lowered, as expected, when temperature is increased within the Müller-Brown potential; however, free energies of transition (region 2) and boundary regions (beyond regions 1 or 3) of phase-space cannot be reproduced reliably. We note that the selected α values have not been rigorously fitted to best match the different values of $k_B T$ shown, but were instead evenly sampled over a fixed interval, in which similar dynamics to simulation are observed.

Also of note is the case of $\alpha = 0$ (not shown in Figure 5b), where the VDE behaves essentially as an indicator function, reporting which basin a given frame will eventually diffuse towards in a low temperature simulation. As α is increased, the VDE must decide which basin to push the now heat-bathed system towards and more realistic dynamics can be observed. Such behavior is analagous to a 'black-box' Langevin equation, whereby the VDE has learned some of the underlying dynamical characteristics of the system; although, there seems to be a strong attraction to certain basins (e.g. region 2) which is not observed in simulation. Because of this attraction, increasing α raisies intermediate basins towards realistic free energies rather than lowering them, as one might expect when raising the temperature of a simulation. We recommend greatly increasing α, as described in Section 2.1, when generating synthetic trajectories due to this trend.

4.3 A Simple Encoding for Villin Headpiece Dynamics

We next apply the VDE to pairwise alpha-carbon (Cα) contacts in order to model the folding process of the villin headpiece subdomain. Here, we aim to assess the quality of the VDE as a dimensionality reduction technique for protein folding by quantifying how well a MSM constructed from VDE-transformed data separates relevant timescales and distinguishes basins within the landscape. With these metrics in mind, the VDE appears to represent the folding landscape well and can even out-perform tICA using similar MSM hyperparameters.

Figure 6a depicts trajectory data projected onto the slowest two tICA processes (tICs) from an optimized tICA model[48] and colored by the projection onto the VDE latent coordinate. In the optimal tICA model, 2 tICs are needed to capture both the folding and a prominent misfolding process. The first tIC is unable to completely separate unfolded and folded state, whereas the second tIC distinguishes the folded and unfolded state but is unable to distinguish the folded and misfolded state. In contrast, the VDE latent coordinate is able to discriminate between all three states: folded, unfolded, and misfolded. By comparing the free energies of the VDE latent space (Figure 6c) and the first tIC (Figure 6d), we observe that the VDE coordinate has a narrower basin of folding than that of the first tIC, indicating the VDE latent coordinate more sharply resolves the folding basin than the first tIC does.

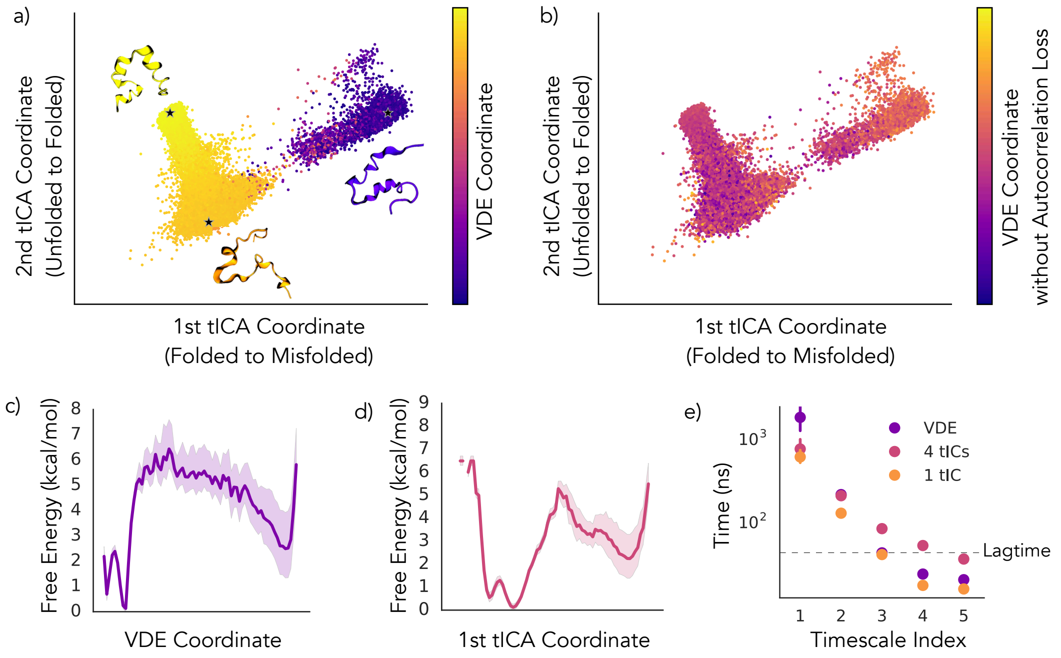

Figure 6. The latent space of the VDE is able to discriminate between three significant states in the folding coordinate of villin: the folded (yellow), unfolded (orange), and a prominent misfolded (purple) states (shown in a). In contrast, an optimized tICA model requires two coordinates to differentiate these states. The autocorrelation loss is crucial for this; without it the VDE is unable to describe the landscape (shown in b). Comparing the free energies of the VDE coordinate (c) and the first tICA coordinate (d) indicates that the VDE is better able to separate the folded and unfolded state from the misfolded state. When comparing the timescales of MSMs constructed from both models (e), the VDE has a slower first process than an optimal tICA model with 4 tICA components and performs significantly better than a tICA model with a single coordinate, indicating a superior model. Error bars represent the range of 100 bootstrapped replicates.

To test the benefit of using the autocorrelation loss discussed in Section 2.2, we trained models of villin using only the reconstruction loss and no autocorrelation loss. The projection of the optimal model with no autocorrelation loss is portrayed in Figure 6b. In this projection, there is minimal differentiation between different parts of the landscape. This highlights the necessity of incorporating an autocorrelation loss into the VDE loss function.

MSMs for the villin landscape were constructed from both the VDE model and the optimized tICA model. Comparing these models indicates that the VDE model identifies a slower timescale than the tICA model. Figure 6e portrays the timescales of the slowest five processes identified by MSMs built from the VDE projection, our optimized tICA model, and a tICA model built with one tICA component. The timescale of the slowest process in the MSM from the VDE projection is 1620 ± 80 nanoseconds, whereas the timescale of the slowest process in the optimal tICA model is 770 ± 40 nanoseconds. According to the variational approach to conformational dynamics, as described in Section 2.2, a model with longer timescales should be closer to modeling the true dynamics of the system.

4.4 Protein Saliency Maps Enable Interpretation of the VDE

As noted in Figure 1, nonlinear methods for time-series analysis tend to sacrifice model interpretability. Linear tICA provides "loadings" on each input feature for each slow mode. Thus, the absolute magnitude of these loadings can be used to understand holistic protein dynamics at the atomic scale[37]. To make VDEs more interpretable, we designed a novel variant to saliency maps (see Section 3.3) to gain insight into how the network operates and propagates protein configurations at a particular lag time. Saliency maps[31–33] were originally proposed for looking at spatial support for varied classification problems. For image data, they find spatial features that a network looks for during classification, i.e., by asking how much does any individual pixel contribute to the final prediction. This is done by back-propagating from the desired class score through the network and into the image pixels. Similar to tICA loadings, the magnitude of the derivative can then be used to gauge feature importance per output class. An alternative closely related method, namely guided back-propagation, only propagates the positive derivatives through the network.This allows the modeler to visualize what pixels a network looks at when it ouputs a certain class label.

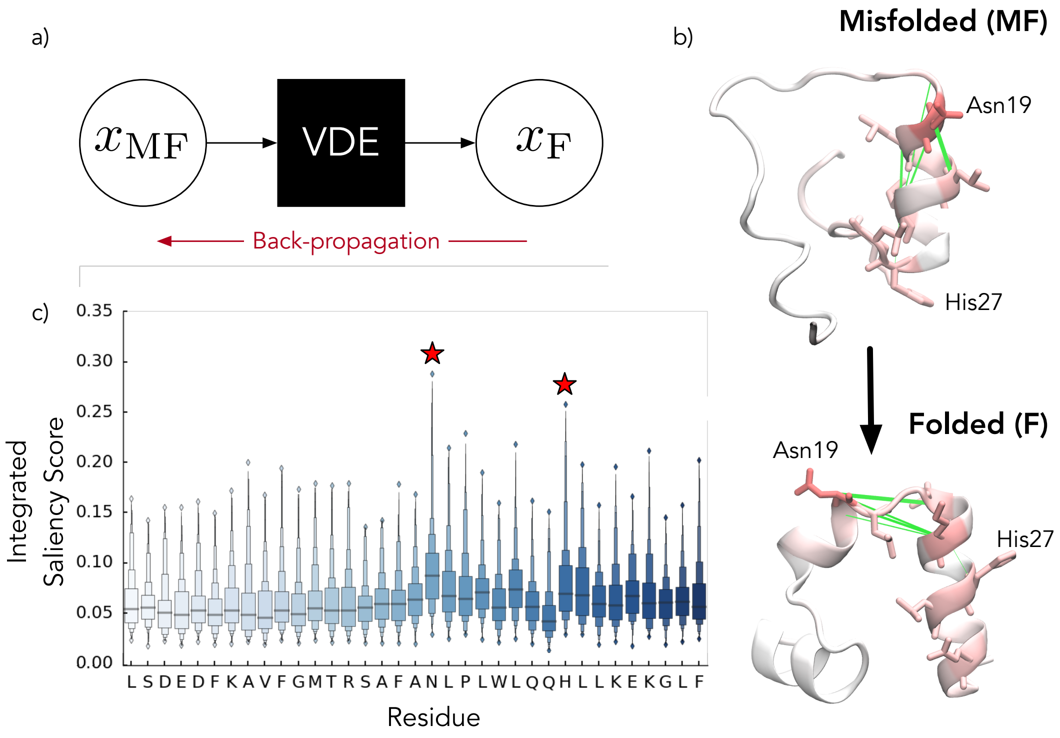

Figure 7. Protein saliency maps can be used to gain insight into the VDE. a) In saliency maps, the distances between the predicted and targeted output (i.e. contact distances in the folded states) are propagated back through the network to the input contact distances in order to gain insight into what the network learns. This is repeated for a large batch of possible configurations. b) For villin, the folded state is characterized by Cα contact distances to the central Asparagine residue. In the misfolded state, this residue is too close to the first helix forming non-native contacts. The green lines denote the 5 contacts with the highest median saliency scores. c) Integrating over the saliency at the atomic level allows us to infer the importances of individual residues in certain state transitions, making them prime candidates (red stars) for further biophysical characterization. The distributions are computed over 200 transitions.

To perform the saliency analysis, we computed a median value for the derivative of the residual between villin's misfolded and folded basin with respect to its input contact features. As shown in Figure 7b, these saliency maps for villin found 5 important Cα contacts. The 5 contacts (indicated with green lines in Figure 7b), all involve contacts to residues around Asn19. Remarkably, we can also integrate the saliency scores for each atomic feature to infer feature importances at the residue scale. The residue importance Figure 7b-c can be used to potentially design new molecular simulations and biophysical experiments. For example, in the case of villin, our model predicts distances to Asn19 as being critical for movement out of the misfolded partially helical state (Figure 7b). Mutating this residue to a proline or glycine could potentially be used to prevent the system from sampling the misfolded state. A potential drawback for this method is it requires sufficient knowledge of the system to identify a relevant path to investigate the corresponding initial and final conformations. This can be accomplished by either some empirical analysis, clustering, or simply sampling conformations at the minima and maxima of the latent space.

5 Discussion

In this work, we have introduced a variational autoencoder to analyze dynamical processes by incorporating a time lag into the traditional autoencoder structure, introducing an autocorrelation loss during training, and leveraging the Gaussian noise introduced into the latent space during training, dimensionality reduction, and synthetic trajectory analysis. Furthermore, we have introduced a saliency mapping approach inspired by advances in deep learning in order to interpret which features contribute to the identified progress coordinate.

We demonstrate that the VDE is able to outperform state-of-the-art methods, such as tICA, in describing slow dynamics in both the 2-D Müller-Brown potential and protein folding. In the more complex case of protein folding, we show the utility of the VDE in understanding the conformational landscape of the villin headpiece domain, which is non-trivial due to the prominent misfolded state observed. The latent space of the VDE captures the transitions among misfolded, unfolded, and folded, and a MSM constructed using the VDE projection exhibits a significantly longer timescale for the slowest process than the optimized one constructed from tICA-transformed data. We also show how biophysical insights into the network's decision-making can be attained via protein saliency mapping. For villin, we identify important Cα contacts that we predict potentially play a role in misfolding-folding transitions. We anticipate that such results will prove useful in experimental design, such as in FRET experiments, to decide how to effectively probe a protein to observe conformational change.

We also examine the generative nature of the VDE, showing that it can generate realistic dynamics when trained on Brownian dynamics trajectories and has potential to extrapolate dynamics at temperatures it has not observed; however, as is, the model is unable to recover proper thermodynamics nor is it able to generate new conformations. We hypothesize that at least some of discrepancies observed in the generated free energies landscapes may be due to the simplicity of the Gaussian prior used. Assuming Langevin dynamics, one might expect the noise term within the reduced coordinate to be a non-trivial function of the original dynamics, rather than a Gaussian process. Another approach to improving trajectory generation may be to train on trajectories sampled using replica exchange methods and condition on an additional temperature variable. We expect that a better understanding of how VDE priors and parameters relate to simulation parameters will lead to using the VDE to efficiently sample thermodynamics across different simulation conditions.

While the VDE shows much promise, there are a few reasons why we cannot recommend it as a complete replacement of previous methods, such as tICA, just yet. First, when training deep autoencoders using a autocorrelation loss (i.e. as inspired by the VAC), there is a noticeable dependence on batch size that arises during training. The autocorrelation, as well as the related variational loss, attempts to calculate global equilibrium statistics, such as the exchange timescale for the slowest process. However, for finite batch sizes, we might only observe a single event in that process within a given batch. This may lead to underestimating the computed statistics since the network has no information about the rest of the dataset. This problem does not arise in tICA or MSMs because timescales and other global statistics are only estimated *after* all the data has already been processed. Another issue with using the autocorrelation loss, as implemented, arises from the reality that many processes can occur with similar timescales. Each of these processes can be assigned highly similar autocorrelations, and thus might lead to volatile training; although, we believe our compound loss function can somewhat attenuate this issue, since the network is designed to keep track of global transition dynamics in addition to fitting the slowest processes.

One area for further study is the effect of the components of the loss function on the latent encoding obtained. Concurrent work in our group has demonstrated that both the incorporation of the autocorrelation loss and the time-lagged reconstruction loss are necessary for obtaining a latent space with maximal autocorrelation[35].

All in all, VDEs and recent related work[23,24] herald exciting opportunities for bridging Markov models and deep learning. We believe the expressive power of neural networks provides a natural solution to the choice-of-basis problem that plagues many Markovian analyses, while the strong theoretical underpinnings behind MSMs allow us to select and potentially even validate cross-validate neural architectures[11,50], ultimately allowing us to address fundamental questions in biophysics.

Availability

Source code for this work is available under the open-source MIT license and is accessible at `https://github.com/msmbuilder/vde`. Complete examples can be found as Jupyter notebooks at `https://github.com/msmbuilder/vde/tree/notebooks/`.

Author Contributions

CXH, HKWS, MMS, and BEH performed analysis and wrote the manuscript. CXH, HKWS, MMS, BEH, and VSP conceived of the methodology and edited the manuscript. VSP supervised the project.

Acknowledgements

We would like to thank J. Chodera, P. Eastman, E. Feinberg, A. Peck, and R. Sharma for insightful discussions. We acknowledge funding from NIH grants U19 AI109662 and 2R01GM062868 for their support of this work. CXH and HKWS acknowledge support from NSF GRFP (DGE-114747). MMS would like to acknowledge support from the National Science Foundation grant NSF-MCB-0954714. This work used the XStream computational resource, supported by the National Science Foundation Major Research Instrumentation program (ACI-1429830), as well as the Sherlock cluster, maintained by the Stanford Research Computing Center.

Disclosures

VSP is a consultant and SAB member of Schrodinger, LLC and Globavir, sits on the Board of Directors of Apeel Inc, Freenome Inc, Omada Health, Patient Ping, Rigetti Computing, and is a General Partner at Andreessen Horowitz.

References

1. Shirts, M. & Pande, V. S. Screen savers of the world unite! *Sci.* **290**, 1903–1904 (2000).
2. Shaw, D. E. *et al.* Anton, a special-purpose machine for molecular dynamics simulation. *Commun. ACM* **51**, 91–97 (2008).
3. Shukla, D., Hernández, C. X., Weber, J. K. & Pande, V. S. Markov state models provide insights into dynamic modulation of protein function. *Acc. Chem. Res.* **48**, 414–422 (2015).
4. Prinz, J.-H. *et al.* Markov models of molecular kinetics: Generation and validation. *J. Chem. Phys.* **134**, 174105 (2011).
5. Husic, B. E. & Pande, V. S. Markov state models: From an art to a science. *J. Am. Chem. Soc.* (2018).
6. Noé, F. & Nüske, F. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Model. Simul.* **11**, 635–655 (2013).
7. Naritomi, Y. & Fuchigami, S. Slow dynamics in protein fluctuations revealed by time-structure based independent component analysis: the case of domain motions. *J. Chem. Phys.* **134**, 065101 (2011).
8. 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.* **139**, 015102 (2013).
9. 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.* **9**, 2000–2009 (2013).
10. Schwantes, C. R. & Pande, V. S. Modeling Molecular Kinetics with tICA and the Kernel Trick. *J. Chem. Theory Comput.* **11**, 600–608 (2015).
11. McGibbon, R. T. & Pande, V. S. Variational cross-validation of slow dynamical modes in molecular kinetics. *J. Chem. Phys.* **142**, 03B621_1 (2015).
12. Wu, H. *et al.* Variational koopman models: Slow collective variables and molecular kinetics from short off-equilibrium simulations. *J. Chem. Phys.* **146**, 154104 (2017).
13. Harrigan, M. P. & Pande, V. S. Towards robust dynamical models of biomolecules. Ph.D. Thesis, Stanford University (2017).
14. Coifman, R. R. & Lafon, S. Diffusion maps. *Appl. Comput. Harmon. Anal.* **21**, 5–30 (2006).
15. Harrigan, M. P. & Pande, V. S. Landmark Kernel tICA For Conformational Dynamics. *bioRxiv* 123752 (2017).
16. Kingma, D. P. & Welling, M. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114* (2013).
17. Rezende, D. J., Mohamed, S. & Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).
18. Hecht-Nielsen, R. Replicator neural networks for universal optimal source coding. *Sci.* 1860–1863 (1995).
19. Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Sci.* **313**, 504–507 (2006).
20. Bengio, Y., Yao, L., Alain, G. & Vincent, P. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, 899–907 (2013).
21. Pearson, K. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, Dublin Philos. Mag. J. Sci.* **2**, 559–572 (1901).

22. McGibbon, R. T., Husic, B. E. & Pande, V. S. Identification of simple reaction coordinates from complex dynamics. *J. Chem. Phys.* **146**, 044109 (2017).

23. Mardt, A., Pasquali, L., Wu, H. & Noé, F. Vampnets for deep learning of molecular kinetics. *Nat. communications* **9**, 5 (2018).

24. Wehmeyer, C. & Noé, F. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The J. Chem. Phys.* **148**, 241703 (2018).

25. Walker, J., Doersch, C., Gupta, A. & Hebert, M. An Uncertain Future: Forecasting from Static Images using Variational Autoencoders. *arXiv preprint arXiv:1606.07873* (2016).

26. Grathwohl, W. & Wilson, A. Disentangling space and time in video with hierarchical variational auto-encoders. *arXiv preprint arXiv:1612.04440* (2016).

27. Fraccaro, M., Kamronn, S., Paquet, U. & Winther, O. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *arXiv preprint arXiv:1710.05741* (2017).

28. Bao, W., Yue, J. & Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one* **12**, e0180944 (2017).

29. Cui, J., Liu, X., Wang, Y. & Liu, H. Deep reconstruction model for dynamic pet images. *PloS one* **12**, e0184667 (2017).

30. Doerr, S., Ariz, I., Harvey, M. J. & De Fabritiis, G. Dimensionality reduction methods for molecular simulations. *arXiv preprint arXiv:1710.10629* (2017).

31. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

32. Springenberg, J. T., Dosovitskiy, A., Brox, T. & Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* (2014).

33. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps (2013).

34. Husic, B. E. & Pande, V. S. Note: MSM lag time cannot be used for variational model selection. *J. Chem. Phys.* **147**, 176101 (2017).

35. Wayment-Steele, H. K. & Pande, V. S. Note: Variational encoding of protein dynamics benefits from maximizing latent autocorrelation. *arXiv preprint arXiv:1803.06449* (2018).

36. Noé, F. *et al.* Dynamical Fingerprints for Probing Individual Relaxation Processes in Biomolecular Dynamics with Simulations and Kinetic Experiments. *Proceedings of the National Academy of Sciences* **108**, 4822–4827 (2011).

37. Harrigan, M. P. *et al.* MSMBuilder: Statistical models for biomolecular dynamics. *Biophys. J.* **112**, 10–15 (2017).

38. Müller, K. & Brown, L. D. Location of saddle points and minimum energy paths by a constrained simplex optimization procedure. *Theor. Chim. Acta* **53**, 75–93 (1979).

39. Ramachandran, P., Zoph, B. & Le, Q. V. Searching for activation functions. *CoRR* **abs/1710.05941** (2017).

40. Kingma, D. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

41. Sculley, D. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web - WWW '10*, 1177 (ACM Press, New York, New York, USA, 2010).

42. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

43. Metzner, P., Noé, F. & Schütte, C. Estimating the sampling error: distribution of transition matrices and functions of transition matrices for given trajectory data. *Phys. Rev. E* **80**, 021106 (2009).

44. Hernández, C. X. & Pande, V. S. MDEntropy: Information-Theoretic Analyses for Molecular Dynamics. *The J. Open Source Softw.* **2**, 427 (2017).

45. Hernández, C. X., Harrigan, M. P., Sultan, M. M. & Pande, V. S. MSMExplorer: Data visualizations for biomolecular dynamics. *J. Open Source Softw.* **2** (2017).

46. Lindorff-Larsen, K., Piana, S., Dror, R. O. & Shaw, D. E. How fast-folding proteins fold. *Sci.* **334**, 517–520 (2011).

47. McGibbon, R. T. *et al.* MDTraj: A modern open library for the analysis of molecular dynamics trajectories. *Biophys. J.* **109**, 1528–1532 (2015).

48. Husic, B. E., McGibbon, R. T., Sultan, M. M. & Pande, V. S. Optimized parameter selection reveals trends in Markov state models for protein folding. *J. Chem. Phys.* **145**, 194103 (2016).

49. Noé, F. & Clementi, C. Kinetic distance and kinetic maps from molecular dynamics simulation. *J. Chem. Theory Comput.* **11**, 5002–5011 (2015).

50. McGibbon, R. T. *et al.* Osprey: Hyperparameter optimization for machine learning. *J. Open Source Softw.* **1**, 00034 (2016).

51. See Supplemental Material at [URL will be inserted by publisher] for the following information: Figure S1 contains a bootstrapped mutual information analysis showing that the VDE captures more information from the original features than previous linear methods; and Figure S2 shows that the VDE is able to produce similar models using internal protein coordinates, unlike previous linear methods.