

ARTICLE

DOI: 10.1038/s41467-017-02388-1

OPEN

VAMPnets for deep learning of molecular kinetics

Andreas Mardt¹, Luca Pasquali¹, Hao Wu¹ & Frank Noé¹

There is an increasing demand for computing the relevant structures, equilibria, and long-timescale kinetics of biomolecular processes, such as protein-drug binding, from high-throughput molecular dynamics simulations. Current methods employ transformation of simulated coordinates into structural features, dimension reduction, clustering the dimension-reduced data, and estimation of a Markov state model or related model of the interconversion rates between molecular structures. This handcrafted approach demands a substantial amount of modeling expertise, as poor decisions at any step will lead to large modeling errors. Here we employ the variational approach for Markov processes (VAMP) to develop a deep learning framework for molecular kinetics using neural networks, dubbed VAMPnets. A VAMPnet encodes the entire mapping from molecular coordinates to Markov states, thus combining the whole data processing pipeline in a single end-to-end framework. Our method performs equally or better than state-of-the-art Markov modeling methods and provides easily interpretable few-state kinetic models.

¹Department of Mathematics and Computer Science, Freie Universität Berlin, Arnimallee 6, 14195 Berlin, Germany. Andreas Mardt and Luca Pasquali contributed equally to this work. Correspondence and requests for materials should be addressed to F.N. (email: frank.noe@fu-berlin.de)

The rapid advances in computing power and simulation technologies for molecular dynamics (MD) of biomolecules and fluids^{1–4}, and ab initio MD of small molecules and materials^{5,6}, allow the generation of extensive simulation data of complex molecular systems. Thus, it is of high interest to automatically extract statistically relevant information, including stationary, kinetic, and mechanistic properties.

The Markov modeling approach^{7–12} has been a driving force in the development of kinetic modeling techniques from MD mass data, chiefly as it facilitates a divide-and-conquer approach to integrate short, distributed MD simulations into a model of the long-timescale behavior. State-of-the-art analysis approaches and software packages^{4,13,14} operate by a sequence, or pipeline, of multiple processing steps, that has been engineered by practitioners over the last decade. The first step of a typical processing pipeline is featurization, where the MD coordinates are either aligned (removing translation and rotation of the molecule of interest) or transformed into internal coordinates such as residue distances, contact maps, or torsion angles^{4,13,15,16}. This is followed by a dimension reduction, in which the dimension is reduced to much fewer (typically 2–100) slow collective variables, often based on the variational approach or conformation dynamics^{17,18}, time-lagged independent component analysis (TICA)^{19,20}, blind source separation^{21–23}, or dynamic mode decomposition^{24–28}—see refs^{29,30} for an overview. The resulting coordinates may be scaled, in order to embed them in a metric space whose distances correspond to some form of dynamical distance^{31,32}. The resulting metric space is discretized by clustering the projected data using hard or fuzzy data-based clustering methods^{11,13,33–36,37}, typically resulting in 100–1000 discrete states. A transition matrix or rate matrix describing the transition probabilities or rate between the discrete states at some lag time τ is then estimated^{8,12,38,39} (alternatively, a Koopman model can be built after the dimension reduction^{27,28}). The final step toward an easily interpretable kinetic model is coarse-graining of the estimated Markov state model (MSM) down to a few states^{40–46}.

This sequence of analysis steps has been developed by combining physico-chemical intuition and technical experience gathered in the last ~10 years. Although each of the steps in the above pipeline appears meaningful, there is no fundamental reason why this or any other given analysis pipeline should be optimal. More dramatically, the success of kinetic modeling currently relies on substantial technical expertise of the modeler, as suboptimal decisions in each step may deteriorate the result. As an example, failure to select suitable features in step 1 will almost certainly lead to large modeling errors.

An important step toward selecting optimal models (parameters) and modeling procedures (hyper-parameters) has been the development of the variational approach for conformation dynamics (VAC)^{17,18}, which offers a way to define scores that measure the optimality of a given kinetic model compared to the (unknown) MD operator that governs the true kinetics underlying the data. The VAC has recently been generalized to the variational approach for Markov processes (VAMP), which allows to optimize models of arbitrary Markov processes, including nonreversible and non-stationary dynamics⁴⁷. The VAC has been employed using cross-validation in order to make optimal hyper-parameter choices within the analysis pipeline described above while avoiding overfitting^{34,48}. However, a variational score is not only useful to optimize the steps of a given analysis pipeline, but in fact allows us to replace the entire pipeline with a more general learning structure.

Here we develop a deep learning structure that is in principle able to replace the entire analysis pipeline above. Deep learning has been very successful in a broad range of data analysis and

learning problems^{49–51}. A feedforward deep neural network is a structure that can learn a complex, nonlinear function $y=F(x)$. In order to train the network, a scoring or loss function is needed that is maximized or minimized, respectively. Here we develop VAMPnets, a neural network architecture that can be trained by maximizing a VAMP variational score. VAMPnets contain two network lobes that transform the molecular configurations found at a time delay τ along the simulation trajectories. Compared to previous attempts to include “depth” or “hierarchy” into the analysis method^{52,53}, VAMPnets combine the tasks of featurization, dimension reduction, discretization, and coarse-grained kinetic modeling into a single end-to-end learning framework. We demonstrate the performance of our networks using a variety of stochastic models and data sets, including a protein-folding data set. The results are competitive with and sometimes surpass the state-of-the-art handcrafted analysis pipeline. Given the rapid improvements of training efficiency and accuracy of deep neural networks seen in a broad range of disciplines, it is likely that follow-up works can lead to superior kinetic models.

Results

Variational principle for Markov processes. Molecular dynamics can be theoretically described as a Markov process $\{x_t\}$ in the full state space Ω . For a given potential energy function, the simulation setup (e.g., periodic boundaries) and the time-step integrator used, the dynamics are fully characterized by a transition density $p_\tau(x, y)$, i.e., the probability density that a MD trajectory will be found at configuration y given that it was at configuration x a time lag τ before. Markovianity implies that the y can be sampled by knowing x alone, without the knowledge of previous time steps. While the dynamics might be highly non-linear in the variables x , Koopman theory^{24,54} tells us that there is a transformation of the original variables into some features or latent variables that, on average, evolve according to a linear transformation. In mathematical terms, there exist transformations to features or latent variables, $\chi_0(x)=(\chi_{01}(x), \dots, \chi_{0m}(x))^\top$ and $\chi_1(x)=(\chi_{11}(x), \dots, \chi_{1m}(x))^\top$, such that the dynamics in these variables are approximately governed by the matrix K :

$$\mathbb{E}[\chi_1(x_{t+\tau})] \approx K^\top \mathbb{E}[\chi_0(x_t)]. \quad (1)$$

This approximation becomes exact in the limit of an infinitely large set of features ($m \rightarrow \infty$) χ_0 and χ_1 , but for a sufficiently large lag time τ the approximation can be excellent with low-dimensional feature transformations, as we will demonstrate below. The expectation values \mathbb{E} account for stochasticity in the dynamics, such as in MD, but they can be omitted for deterministic dynamical systems^{24,26,27}.

To illustrate the meaning of Eq. (1), consider the example of $\{x_t\}$ being a discrete-state Markov chain. If we choose the feature transformation to be indicator functions ($\chi_{0i}=1$ when $x_t=i$ and 0 otherwise, and correspondingly with χ_{1i} and $x_{t+\tau}$), their expectation values are equal to the probabilities of the chain to be in any given state, p_t and $p_{t+\tau}$, and $K=P(\tau)$ is equal to the matrix of transition probabilities, i.e., $p_{t+\tau}=P^\top(\tau)p_t$. Previous papers on MD kinetics have usually employed a propagator or transfer operator formulation instead of (1)^{7,8}. However, the above formulation is more powerful as it also applies to nonreversible and non-stationary dynamics, as found for MD of molecules subject to external force, such as voltage, flow, or radiation^{55,56}.

A central result of the VAMP theory is that the best finite-dimensional linear model, i.e., the best approximation in Eq. (1), is found when the subspaces spanned by χ_0 and χ_1 are identical to those spanned by the top m left and right singular functions, respectively, of the so-called Koopman operator⁴⁷.

For an introduction to the Koopman operator, please refer to refs. 24,30,54.

How do we choose χ_0 , χ_1 , and K from data? First, suppose we are given some feature transformation χ_0 , χ_1 and define the following covariance matrices:

$$\mathbf{C}_{00} = \mathbb{E}_t [\chi_0(\mathbf{x}_t) \chi_0(\mathbf{x}_t)^\top] \quad (2)$$

$$\mathbf{C}_{01} = \mathbb{E}_t [\chi_0(\mathbf{x}_t) \chi_1(\mathbf{x}_{t+\tau})^\top] \quad (3)$$

$$\mathbf{C}_{11} = \mathbb{E}_{t+\tau} [\chi_1(\mathbf{x}_{t+\tau}) \chi_1(\mathbf{x}_{t+\tau})^\top], \quad (4)$$

where $\mathbb{E}_t[\cdot]$ and $\mathbb{E}_{t+\tau}[\cdot]$ denote the averages that extend over time points and lagged time points within trajectories, respectively, and across trajectories. Then the optimal K that minimizes the least square error $\mathbb{E}_t [\|\chi_1(\mathbf{x}_{t+\tau}) - K^\top \chi_0(\mathbf{x}_t)\|^2]$ is refs 27,57,47:

$$\mathbf{K} = \mathbf{C}_{00}^{-1} \mathbf{C}_{01}. \quad (5)$$

Now the remaining problem is how to find suitable transformations χ_0 , χ_1 . This problem cannot be solved by minimizing the least square error above, as is illustrated by the following example: suppose we define $\chi_0(\mathbf{x}) = \chi_1(\mathbf{x}) = 1(\mathbf{x})$, i.e., we just map the state space to the constant 1—in this case the least square error is 0 for $\mathbf{K} = [1]$, but the model is completely uninformative as all dynamical information is lost.

Instead, in order to seek χ_0 and χ_1 based on available simulation data, we employ the VAMP theorem introduced in ref. 47, that can be equivalently formulated as the following subspace version.

VAMP variational principle. For any two sets of linearly independent functions $\chi_0(\mathbf{x})$ and $\chi_1(\mathbf{x})$, let us call

$$\hat{R}_2[\chi_0, \chi_1] = \left\| \mathbf{C}_{00}^{-\frac{1}{2}} \mathbf{C}_{01} \mathbf{C}_{11}^{-\frac{1}{2}} \right\|_F^2$$

their VAMP-2 score, where \mathbf{C}_{00} , \mathbf{C}_{01} , \mathbf{C}_{11} are defined by Eqs. (2)–(4) and $\|\mathbf{A}\|_F^2 = n^{-1} \sum_{i,j} A_{ij}^2$ is the Frobenius norm of $n \times n$ matrix \mathbf{A} . The maximum value of VAMP-2 score is achieved when the top m left and right Koopman singular functions belong to span($\chi_0_1, \dots, \chi_0_m$) and span($\chi_1_1, \dots, \chi_1_m$), respectively.

This variational theorem shows that the VAMP-2 score measures the consistency between subspaces of basis functions and those of dominant singular functions, and we can therefore optimize χ_0 and χ_1 via maximizing the VAMP-2 score. In the special case where the dynamics are reversible with respect to equilibrium distribution the theorem above specializes to variational principle for reversible Markov processes^{17,18}.

Learning the feature transformation using VAMPnets. Here we employ neural networks to find an optimal set of basis functions, $\chi_0(\mathbf{x})$ and $\chi_1(\mathbf{x})$. Neural networks with at least one hidden layer are universal function approximators⁵⁸, and deep networks can express strongly nonlinear functions with a fairly few neurons per layer⁵⁹. Our networks use VAMP as a guiding principle and are hence called VAMPnets. VAMPnets consist of two parallel lobes, each receiving the coordinates of time-lagged MD configurations \mathbf{x}_t and $\mathbf{x}_{t+\tau}$ as input (Fig. 1). The lobes have m output nodes and are trained to learn the transformations $\chi_0(\mathbf{x}_t)$ and $\chi_1(\mathbf{x}_{t+\tau})$, respectively. For a given set of transformations, χ_0 and χ_1 , we pass a batch of training data through the network and compute the training VAMP score of our choice. VAMPnets bear similarities with auto-encoders^{60,61} using a time-delay embedding and are closely related to deep canonical covariance analysis (CCA)⁶².

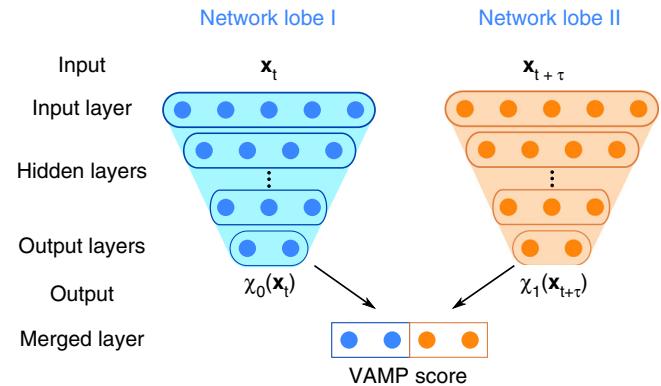


Fig. 1 Scheme of the neural network architecture used. For each time step t of the simulation trajectory, the coordinates \mathbf{x}_t and $\mathbf{x}_{t+\tau}$ are inputs to two deep networks that conduct a nonlinear dimension reduction. In the present implementation, the output layer consists of a Softmax classifier. The outputs are then merged to compute the variational score that is maximized to optimize the networks. In all present applications, the two network lobes are identical clones, but they can also be trained independently

VAMPnets are identical to deep CCA with time-delay embedding when using the VAMP-1 score discussed in ref. 47, however the VAMP-2 score has easier-to-handle gradients and is more suitable for time series data, due to its direct relation to the Koopman approximation error⁴⁷.

The first left and right singular functions of the Koopman operator are always equal to the constant function $1(\mathbf{x}) \equiv 1$ ⁴⁷. We can thus add 1 to basis functions and train the network by maximizing

$$\hat{R}_2 \left[\begin{pmatrix} 1 \\ \chi_0 \end{pmatrix}, \begin{pmatrix} 1 \\ \chi_1 \end{pmatrix} \right] = \left\| \overline{\mathbf{C}}_{00}^{-\frac{1}{2}} \overline{\mathbf{C}}_{01} \overline{\mathbf{C}}_{11}^{-\frac{1}{2}} \right\|_F^2 + 1, \quad (6)$$

where $\overline{\mathbf{C}}_{00}$, $\overline{\mathbf{C}}_{01}$, $\overline{\mathbf{C}}_{11}$ are mean-free covariances of the feature-transformed coordinates:

$$\overline{\mathbf{C}}_{00} = (T-1)^{-1} \overline{\mathbf{X}} \overline{\mathbf{X}}^\top \quad (7)$$

$$\overline{\mathbf{C}}_{01} = (T-1)^{-1} \overline{\mathbf{X}} \overline{\mathbf{Y}}^\top \quad (8)$$

$$\overline{\mathbf{C}}_{11} = (T-1)^{-1} \overline{\mathbf{Y}} \overline{\mathbf{Y}}^\top. \quad (9)$$

Here we have defined the matrices $\mathbf{X} = [X_{ij}] = \chi_0_i(\mathbf{x}_j) \in \mathbb{R}^{m \times T}$ and $\mathbf{Y} = [Y_{ij}] = \chi_1_i(\mathbf{x}_{j+\tau}) \in \mathbb{R}^{m \times T}$ with $\{(\mathbf{x}_j, \mathbf{x}_{j+\tau})\}_{j=1}^T$ representing all available transition pairs, and their mean-free versions $\overline{\mathbf{X}} = \mathbf{X} - T^{-1} \mathbf{X} \mathbf{1}$, $\overline{\mathbf{Y}} = \mathbf{Y} - T^{-1} \mathbf{Y} \mathbf{1}$. The gradients of \hat{R}_2 are given by:

$$\nabla_{\mathbf{X}} \hat{R}_2 = \frac{2}{T-1} \overline{\mathbf{C}}_{00}^{-1} \overline{\mathbf{C}}_{01} \overline{\mathbf{C}}_{11}^{-1} (\overline{\mathbf{Y}} - \overline{\mathbf{C}}_{01}^T \overline{\mathbf{C}}_{00}^{-1} \overline{\mathbf{X}}) \quad (10)$$

$$\nabla_{\mathbf{Y}} \hat{R}_2 = \frac{2}{T-1} \overline{\mathbf{C}}_{11}^{-1} \overline{\mathbf{C}}_{01}^T \overline{\mathbf{C}}_{00}^{-1} (\overline{\mathbf{X}} - \overline{\mathbf{C}}_{01} \overline{\mathbf{C}}_{11}^{-1} \overline{\mathbf{Y}}) \quad (11)$$

and are back-propagated to train the two network lobes. See Supplementary Note 1 for derivations of Eqs. (6), (10), and (11).

For simplicity of interpretation, we may just use a unique basis set $\chi = \chi_0 = \chi_1$. Even when using two different basis sets would be meaningful, we can unify them by simply defining $\chi = (\chi_0, \chi_1)^\top$. In this case, we clone the lobes of the network and train them using the total gradient $\nabla \hat{R}_2 = \nabla_{\mathbf{X}} \hat{R}_2 + \nabla_{\mathbf{Y}} \hat{R}_2$.

After training, we assess the quality of the learned features and select hyper-parameters (e.g., network size) while avoiding overfitting using the VAMP-2 validation score

$$\hat{R}_2^{\text{val}} = \left\| \left(\bar{\mathbf{C}}_{00}^{\text{val}} \right)^{-\frac{1}{2}} \bar{\mathbf{C}}_{01}^{\text{val}} \left(\bar{\mathbf{C}}_{11}^{\text{val}} \right)^{-\frac{1}{2}} \right\|_F^2 + 1, \quad (12)$$

where $\bar{\mathbf{C}}_{00}^{\text{val}}$, $\bar{\mathbf{C}}_{01}^{\text{val}}$, $\bar{\mathbf{C}}_{11}^{\text{val}}$ are mean-free covariance matrices computed from a validation data set not used during the training.

Dynamical model and validation. The direct estimate of the time-lagged covariance matrix \mathbf{C}_{01} is generally nonsymmetric. Hence the Koopman model or MSM \mathbf{K} given by Eq. (5) is typically not time-reversible²⁸. In MD, it is often desirable to obtain a time-reversible kinetic model—see³⁹ for a detailed discussion. To enforce reversibility, \mathbf{K} can be reweighted as described in²⁸ and implemented in PyEMMA¹³. The present results do not depend on enforcing reversibility, as classical analyses such as PCCA+⁶³ are avoided as the VAMPnet structure automatically performs coarse graining.

Since \mathbf{K} is a Markovian model, it is expected to fulfill the Chapman–Kolmogorov (CK) equation:

$$\mathbf{K}(n\tau) = \mathbf{K}^n(\tau), \quad (13)$$

for any value of $n \geq 1$, where $\mathbf{K}(\tau)$ and $\mathbf{K}(n\tau)$ indicate the models estimated at a lag time of τ and $n\tau$, respectively. However, since any Markovian model of MD can be only approximate^{8,64}, Eq. (13) can only be fulfilled approximately, and the relevant test is whether it holds within statistical uncertainty. We construct two tests based on Eq. (13): in order to select a suitable dynamical model, we proceed as for Markov state models by conducting an eigenvalue decomposition for every estimated Koopman matrix, $\mathbf{K}(\tau)\mathbf{r}_i = \mathbf{r}_i\lambda_i(\tau)$, and computing the implied timescales⁹ as a function of lag time:

$$t_i(\tau) = -\frac{\tau}{\ln|\lambda_i(\tau)|}, \quad (14)$$

We chose a value τ , where $t_i(\tau)$ are approximately constant in τ . After having chosen τ , we test whether Eq. (13) holds within statistical uncertainty⁶⁵. For both the implied timescales and the CK test, we proceed as follows: train the neural network at a fixed lag time τ^* , thus obtaining the network transformation χ , and then compute Eq. (13) or Eq. (14) for different values of τ with a fixed transformation χ . Finally, the approximation of the i th eigenfunction is given by

$$\psi_i^c(\mathbf{x}) = \sum_j r_{ij}\chi_j(\mathbf{x}). \quad (15)$$

If dynamics are reversible, the singular value decomposition and eigenvalue decomposition are identical, i.e., $\sigma_i = \lambda_i$ and $\psi_i = \psi_i^c$.

Network architecture and training. We use VAMPnets to learn molecular kinetics from simulation data of a range of model systems. While any neural network architecture can be employed inside the VAMPnet lobes, we chose the following setup for our applications: the two network lobes are identical clones, i.e., $\chi_0 \equiv \chi_1$, and consist of fully connected networks. In most cases, the networks have less output than input nodes, i.e., the network conducts a dimension reduction. In order to divide the work equally between network layers, we reduce the number of nodes from each layer to the next by a constant factor. Thus, the network architecture is defined by two parameters: the depth d and

the number of output nodes n_{out} . All hidden layers employ rectified linear units (ReLU)^{66,67}.

Here, we build the output layer with Softmax output nodes, i.e., $\chi_i(\mathbf{x}) \geq 0$ for all i and $\sum_i \chi_i(\mathbf{x}) = 1$. Therefore, the activation of an output node can be interpreted as a probability to be in state i . As a result, the network effectively performs featurization, dimension reduction, and finally a fuzzy clustering to metastable states, and the $\mathbf{K}(\tau)$ matrix computed from the network-transformed data is the transition matrix of a fuzzy MSM^{36,37}. Consequently, Eq. (1) propagates probability distributions in time.

The networks were trained with pairs of MD configurations ($\mathbf{x}_t, \mathbf{x}_{t+\tau}$) using the Adam stochastic gradient descent method⁶⁸. For each result, we repeated 100 training runs, each of which with a randomly chosen 90%/10% division of the data into training and validation data. See Methods section for details on network architecture, training, and choice of hyper-parameters.

Asymmetric double-well potential. We first model the kinetics of a bistable one-dimensional process, simulated by Brownian dynamics (Methods) in an asymmetric double-well potential (Fig. 2a). A trajectory of 50,000 time steps is generated. Three-layer VAMPnets are set up with 1-5-10-5 nodes in each lobe. The single input node of each lobe is given the current and time-lagged mean-free x coordinate of the system, i.e., $x_t - \mu_1$ and $x_{t+\tau} - \mu_2$, where μ_1 and μ_2 are the respective means, and $\tau=1$ is used. The network maps to five Softmax output nodes that we will refer to as states, as the network performs a fuzzy discretization by mapping the input configurations to the output activations. The network is trained by using the VAMP-2 score with the four largest singular values.

The network learns to place the output states in a way to resolve the transition region best (Fig. 2b), which is known to be important for the accuracy of a Markov state model^{8,64}. This placement minimizes the Koopman approximation error, as seen by comparing the dominant Koopman eigenfunction (Eq. (15)) with a direct numerical approximation of the true eigenfunction obtained by a transition matrix computed for a direct uniform 200-state discretization of the x axis—see ref.⁸ for details. The implied timescale and CK tests (Eqs. (13) and (14)) confirm that the kinetic model learned by the VAMPnet successfully predicts the long-time kinetics (Fig. 2c, d).

Protein-folding model. While the first example was one-dimensional, we now test if VAMPnets are able to learn reaction coordinates that are nonlinear functions of a multi-dimensional configuration space. For this, we simulate a 100,000 time step Brownian dynamics trajectory (Eq. (17)) using the simple protein-folding model defined by the potential energy function (Supplementary Fig. 1a):

$$U(r) = \begin{cases} -2.5(r-3)^2 & r < 3 \\ 0.5(r-3)^3 - (r-3)^2 & r \geq 3 \end{cases}$$

The system has a five-dimensional configuration space, $\mathbf{x} \in \mathbb{R}^5$, however the energy only depends on the norm of the vector $r = |\mathbf{x}|$. While small values of r are energetically favorable, large values of r are entropically favorable as the number of configurations available on a five-dimensional hypersphere grows dramatically with r . Thus, the dynamics are bistable along the reaction coordinate r . Four-layer network lobes with 5-32-16-8-2 nodes each were employed and trained to maximize the VAMP-2 score involving the largest nontrivial singular value.

The two output nodes successfully identify the folded and the unfolded states, and use intermediate memberships for the

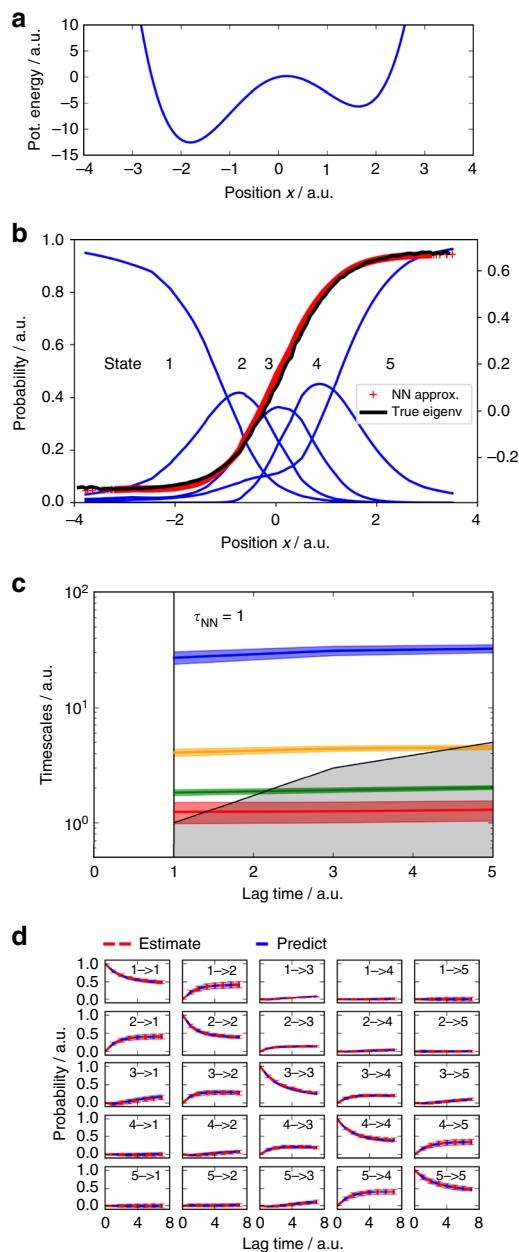


Fig. 2 Approximation of the slow transition in a bistable potential. **a** Potential energy function $U(x) = x^4 - 6x^2 + 2x$. **b** Eigenvector of the slowest process calculated by direct numerical approximation (black) and approximated by a VAMPNet with five output nodes (red). Activation of the five Softmax output nodes define the state membership probabilities (blue). **c** Relaxation timescales computed from the Koopman model using the VAMPNet transformation. **d** Chapman-Kolmogorov test comparing long-time predictions of the Koopman model estimated at $\tau=1$ and estimates at longer lag times. **c, d** report 95% confidence interval error bars over 100 training runs

intersecting transition region (Supplementary Fig. 1b). The network excellently approximates the Koopman eigenfunction of the folding process, as apparent from the comparison of the values of the network eigenfunction computed by Eq. (15) with the eigenvector computed from a high-resolution MSM built on the r coordinate (Supplementary Fig. 1b). This demonstrates that the network can learn the nonlinear reaction coordinate mapping $r = |\mathbf{x}|$ based only on maximizing the variational score (Eq. 6). Furthermore, the implied timescales and the CK test indicate that

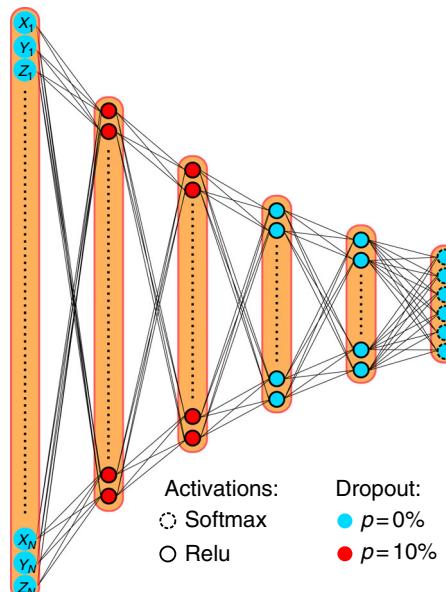


Fig. 3 Representative structure of one lobe of the VAMPNet used for alanine dipeptide. Here, the five-layer network with six output states used for the results shown in Fig. 4 is shown. Layers are fully connected, have 30-22-16-12-9-6 nodes, and use dropout in the first two hidden layers. All hidden neurons use ReLU activation functions, while the output layer uses Softmax activation function in order to achieve a fuzzy discretization of state space

the network model predicts the long-time kinetics almost perfectly (Supplementary Fig. 1c, d).

Alanine dipeptide. As a next level, VAMPnets are used to learn the kinetics of alanine dipeptide from simulation data. It is known that the ϕ and ψ backbone torsion angles are the most important reaction coordinates that separate the metastable states of alanine dipeptide, however, our networks only receive Cartesian coordinates as an input, and are thus forced to learn both the nonlinear transformation to the torsion angle space and an optimal cluster discretization within this space, in order to obtain an accurate kinetic model.

A 250 ns MD trajectory generated in ref.⁶⁹ (MD setup described there) serves as a data set. The solute coordinates were stored every ps, resulting in 250,000 configurations that are all aligned on the first frame using minimal root mean square deviation fit to remove global translation and rotation. Each network lobe uses the three-dimensional coordinates of the 10 heavy atoms as input, $(x_1, y_1, z_1, \dots, x_{10}, y_{10}, z_{10})$, and the network is trained using time lag $\tau=40$ ps. Different numbers of output states and layer depths are considered, employing the layer sizing scheme described in the Methods section (see Fig. 3 for an example).

A VAMPnet with six output states learns a discretization in six metastable sets corresponding to the free energy minima of the ϕ/ψ space (Fig. 4b). The implied timescales indicate that given the coordinate transformation found by the network, the two slowest timescales are converged at lag time $\tau=50$ ps or larger (Fig. 4c). Thus, we estimated a Koopman model at $\tau=50$ ps, whose Markov transition probability matrix is depicted in Fig. 4d. Note that transition probabilities between state pairs 1 \leftrightarrow 4 and 2 \leftrightarrow 3 are important for the correct kinetics at $\tau=50$ ps, but the actual trajectories typically pass via the directly adjacent intermediate states. The model performs excellently in the CK test (Fig. 4e).

Choice of lag time, network depth, and number of output states. We studied the success probability of optimizing a VAMPnet with six output states as a function of the lag time τ by conducting 200 optimization runs. Success was defined as resolving the three slowest processes by finding three slowest timescale higher than 0.2, 0.4, and 1 ns, respectively. Note that the results

shown in Fig. 4 are reported for successful runs in this definition. There is a range of τ values from 4 to 32 ps where the training succeeds with a significant probability (Supplementary Fig. 2a). However, even in this range the success rate is still below 40%, which is mainly due to the fact that many runs fail to find the rarely occurring third-slowest process that corresponds to the ψ transition of the positive ϕ range (Fig. 4b, states 5 and 6).

The breakdown of optimization success for small and large lag times can be most easily explained by the eigenvalue decomposition of Markov propagators⁸. When the lag time exceeds the timescale of a process, the amplitude of this process becomes negligible, making it hard to fit given noisy data. At short lag times, many processes have large eigenvalues, which increases the search space of the neural network and appears to increase the probability of getting stuck in suboptimal maxima of the training score.

We have also studied the success probability, as defined above, as a function of network depth. Deeper networks can represent more complex functions. Also, since the networks defined here reduce the input dimension to the output dimension by a constant factor per layer, deeper networks perform a less radical dimension reduction per layer. On the other hand, deeper networks are more difficult to train. As seen in Supplementary Fig. 2b, a high success rate is found for four to seven layers.

Next, we studied the dependency of the network-based discretization as a function of the number of output nodes (Fig. 5a–c). With two output states, the network separates the state space at the slowest transition between negative and positive values of the ϕ angle (Fig. 5a). The result with three output nodes keeps the same separation and additionally distinguishes between the α and β regions of the Ramachandran plot, i.e., small and large values of the ψ angle (Fig. 5b). For a higher number of output states, finer discretizations and smaller interconversion timescales are found, until the network starts discretizing the transition regions, such as the two transition states between the α and β regions along the ψ angle (Fig. 5c). We chose the lag time depending on the number of output nodes of the network, using $\tau = 200$ ps for two output nodes, $\tau = 60$ ps for three output nodes, and $\tau = 1$ ps for eight output nodes.

A network output with k Softmax neurons describes a $(k - 1)$ -dimensional feature space as the Softmax normalization removes one degree of freedom. Thus, to resolve $k - 1$ relaxation timescales, at least k output nodes or metastable states are required. However, the network quality can improve when given more degrees of freedom in order to approximate the dominant singular functions accurately. Indeed, the best scores using $k = 4$ singular values (three nontrivial singular values) are achieved when using at least six output states that separate each of the six metastable states in the Ramachandran plane (Fig. 5d, e).

For comparison, we investigated how a standard MSM would perform as a function of the number of states (Fig. 5d). For a fair

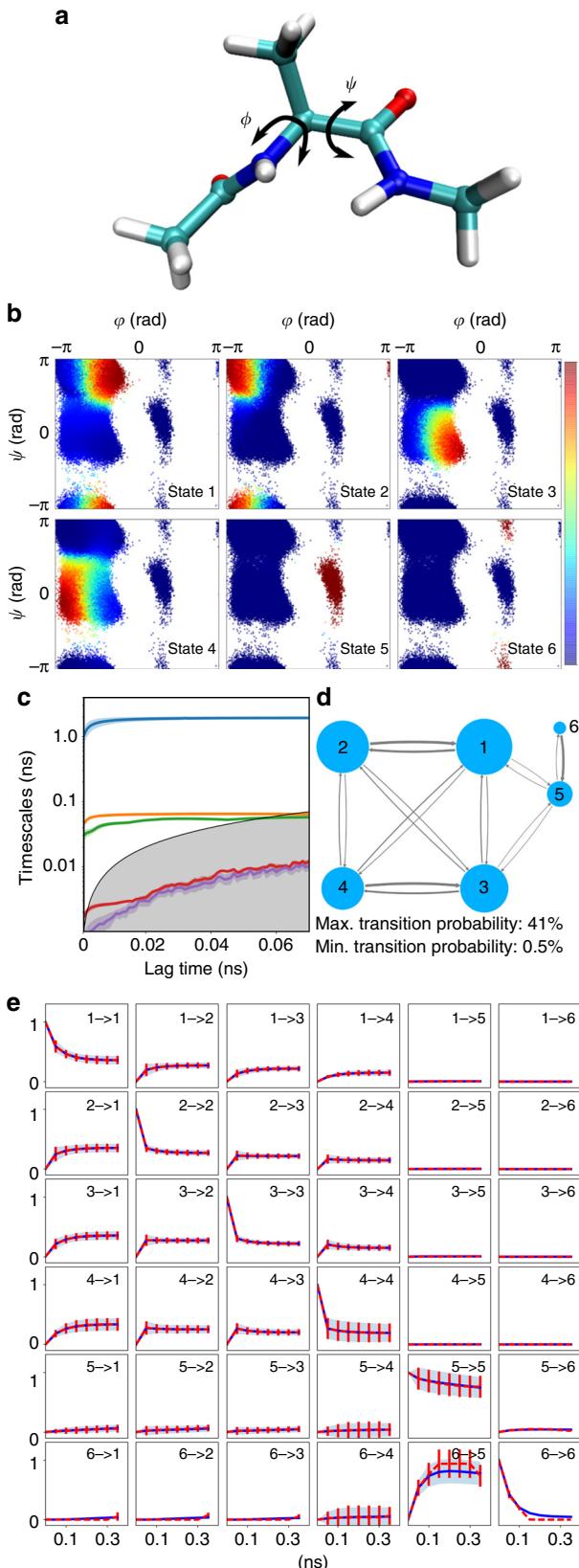


Fig. 4 VAMPnet kinetic model of alanine dipeptide. **a** Structure of alanine dipeptide. The main coordinates describing the slow transitions are the backbone torsion angles ϕ and ψ , however the neural network inputs are only the Cartesian coordinates of heavy atoms. **b** Assignment of all simulated molecular coordinates, plotted as a function of ϕ and ψ , to the six Softmax output states. Color corresponds to activation of the respective output neuron, indicating the membership probability to the associated metastable state. **c** Relaxation timescales computed from the Koopman model using the neural network transformation. **d** Representation of the transition probabilities matrix of the Koopman model; transitions with a probability lower than 0.5% have been omitted. **e** Chapman-Kolmogorov test comparing long-time predictions of the Koopman model estimated at $\tau = 50$ ps and estimates at longer lag times. **c, e** report 95% confidence interval error bars over 100 training runs excluding failed runs (see text).

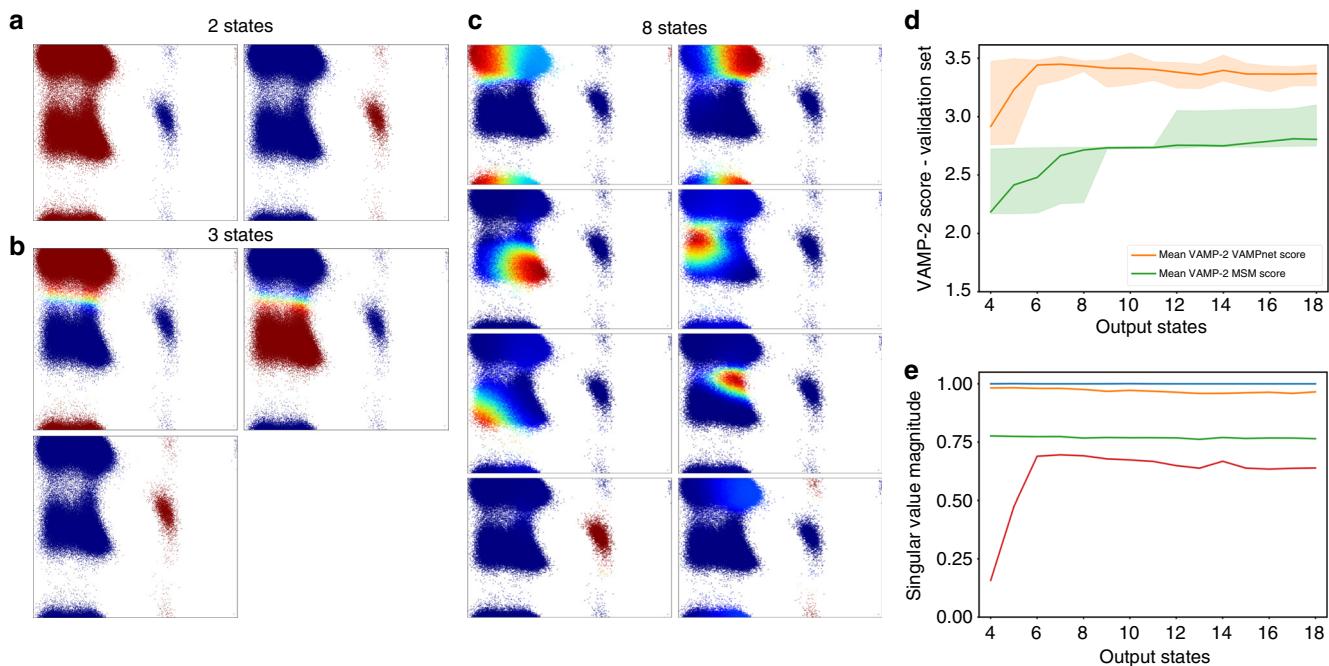


Fig. 5 Kinetic model of alanine dipeptide as a function of the number of output states. **a–c** Assignment of input coordinates, plotted as a function of ϕ and ψ , to two, three, and eight output states. Color corresponds to activation of the respective output neuron, indicating the membership probability to this state (Fig. 4b). **d** Comparison of VAMPnet and MSM performance as a function of the number of output states/MSM states. Mean VAMP-2 score and 95% confidence interval from 100 runs are shown. **e** Mean squared values of the four largest singular values that make up the VAMPnets score plotted in **d**

comparison, the MSMs also used Cartesian coordinates as an input, but then employed a state-of-the-art procedure using a kinetic map transformation that preserves 95% of the cumulative kinetic variance³¹, followed by k -means clustering, where the parameter k is varied. It is seen that the MSM VAMP-2 scores obtained by this procedure is significantly worse than by VAMPnets when <20 states are employed. Clearly, MSMs will succeed when sufficiently many states are used, but in order to obtain an interpretable model, those states must again be coarse-grained onto a fewer-state model, while VAMPnets directly produce an accurate model with few states.

VAMPnets learn to transform Cartesian to torsion coordinates. The results above indicate that the VAMPnet has implicitly learned the feature transformation from Cartesian coordinates to backbone torsions. In order to probe this ability more explicitly, we trained a network with 30-10-3-3-2-5 layers, i.e., including a bottleneck of two nodes before the output layer. We find that the activation of the two bottleneck nodes correlates excellently with the ϕ and ψ torsion angles that were not presented to the network (Pearson correlation coefficients of 0.95 and 0.92, respectively, Supplementary Fig. 3a, b). To visualize the internal representation that the network learns, we color data samples depending on the free energy minima in the ϕ/ψ space they belong to (Supplementary Fig. 3c), and then show where these samples end up in the space of the bottleneck node activations (Supplementary Fig. 3d). It is apparent that the network learns a representation of the Ramachandran plot—the four free energy minima at small ϕ values (α_R and β areas) are represented as contiguous clusters with the correct connectivity, and are well separated from states with large ϕ values (α_L area). The network fails to separate the two substates in the large ϕ value range well, which explains the frequent failure to find the corresponding transition process and the third-largest relaxation timescale.

NTL9 protein-folding dynamics. In order to proceed to a higher-dimensional problem, we analyze the kinetics of an all-atom protein-folding simulation of the NTL9 protein generated by the Anton supercomputer¹. A five-layer VAMPnet was trained at lag time $\tau = 10$ ns using 111,000 time steps, uniformly sampled from a 1.11 ms trajectory. Since NTL9 is folding and unfolding, there is no unique reference structure to align Cartesian coordinates to—hence we use internal coordinates as a network input. We computed the nearest-neighbor heavy-atom distance, d_{ij} for all non-redundant pairs of residues i and j and transformed them into contact maps using the definition $c_{ij} = \exp(-d_{ij})$, resulting in 666 input nodes.

Again, the network performs a hierarchical decomposition of the molecular configuration space when increasing the number of output nodes. Figure 6a shows the decomposition of state space for two and five output nodes, and the corresponding mean contact maps and state probabilities. With two output nodes, the network finds the folded and unfolded state that are separated by the slowest transition process (Fig. 6a, middle row). With five output states, the folded state is decomposed into a stable and well-defined fully folded substate and a less stable, more flexible substate that is missing some of the tertiary contacts compared to the fully folded substate. The unfolded substate decomposes into three substates, one of them largely unstructured, a second one with residual structure, thus forming a folding intermediate, and a mis-folded state with an entirely different fold including a non-native β -sheet.

The relaxation timescales found by a five-state VAMPnet model are en par with those found by a 40-state MSM using state-of-the-art estimation methods (Fig. 6b, c). However, the fact that only five states are required in the VAMPnet model makes it easier to interpret and analyze. Additionally, the CK test indicates excellent agreement between long-time predictions and direct estimates.

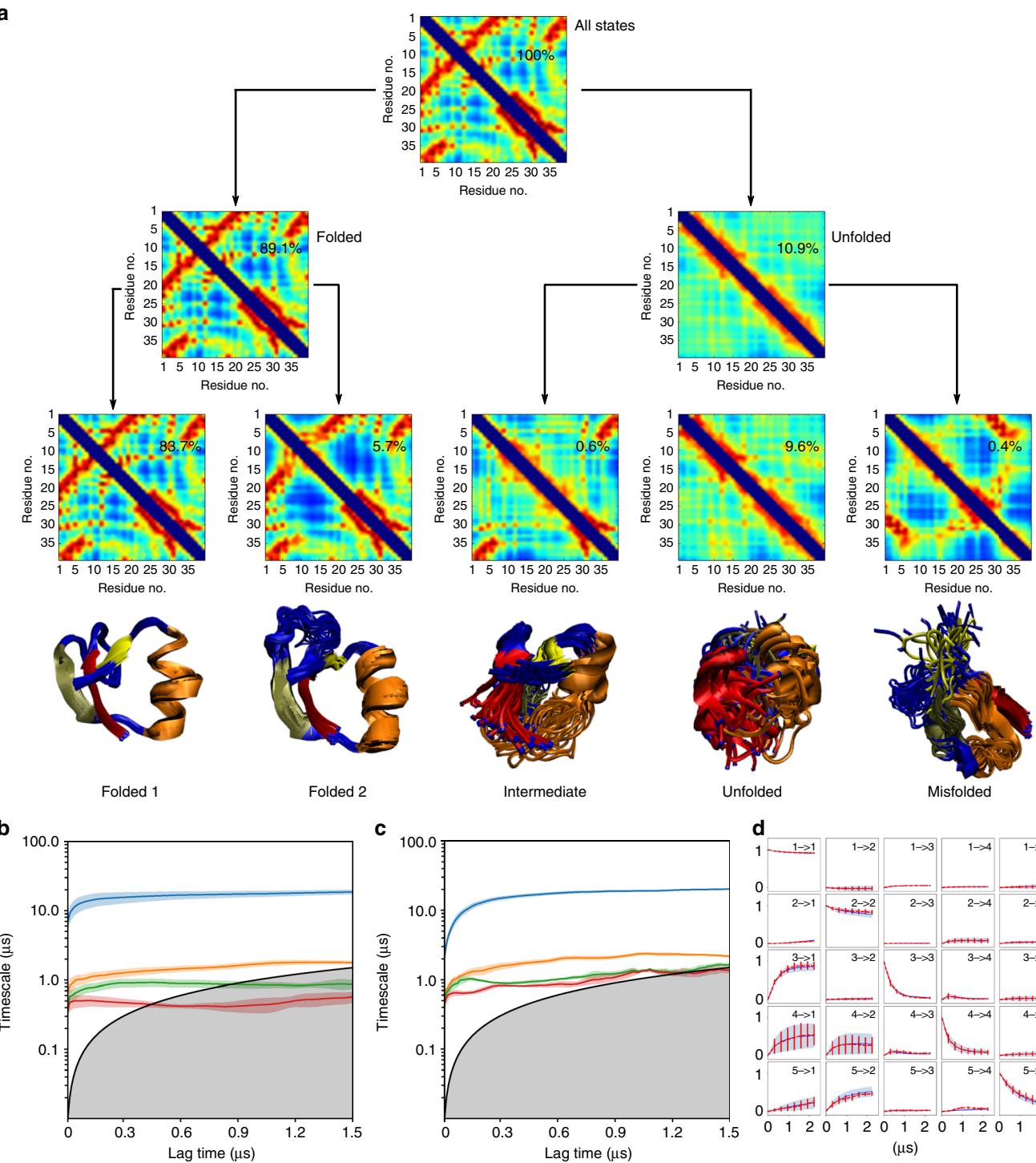


Fig. 6 VAMPnet results of NTL9-folding kinetics. **a** Hierarchical decomposition of the NTL9 protein state space by a network with two and five output nodes. Mean contact maps are shown for all MD samples grouped by the network, along with the fraction of samples in that group. 3D structures are shown for the five-state decomposition, residues involved in α -helices or β -sheets in the folded state are colored identically across the different states. **b** Relaxation timescales computed from the Koopman model approximated using the transformation applied by a neural network with five output nodes. **c** Relaxation timescales from a Markov state model computed from a TICA transformation of the contact maps, followed by k-means clustering with $k = 40$. **d** Chapman-Kolmogorov test comparing long-time predictions of the Koopman model estimated at $\tau = 320$ ns and estimates at longer lag times. **b-d** report 95% confidence interval error bars over 100 training runs

Discussion

We have introduced a deep learning framework for molecular kinetics, called VAMPnet. Data-driven learning of molecular kinetics is usually done by shallow learning structures, such as TICA and MSMs. However, the processing pipeline, typically

consisting of featurization, dimension reduction, MSM estimation, and MSM coarse-graining is, in principle, a handcrafted deep learning structure. Here we propose to replace the entire pipeline by a deep neural network that learns optimal feature transformations, dimension reduction and, if desired, maps the

MD time steps to a fuzzy clustering. The key to optimize the network is the VAMP variational approach that defines scores by which learning structures can be optimized to learn models of both equilibrium and non-equilibrium MD.

Although MSM-based kinetic modeling has been refined over more than a decade, VAMPnets perform competitively or superior in our examples. In particular, they perform extremely well in the Chapman–Kolmogorov test that validates the long-time prediction of the model. VAMPnets have a number of advantages over models based on MSM pipelines: (i) they may be overall more optimal, because featurization, dimension reduction, and clustering are not explicitly separate processing steps. (ii) When using Softmax output nodes, the VAMPNet performs a fuzzy clustering of the MD structures fed into the network and constructs a fuzzy MSM, which is readily interpretable in terms of transition probabilities between metastable states. In contrast to other MSM coarse-graining techniques, it is thus not necessary to accept reduction in model quality in order to obtain a few-state MSM, but such a coarse-grained model is seamlessly learned within the same learning structure. (iii) VAMPnets require less user expertise to train than an MSM-based processing pipelines, and the formulation of the molecular kinetics as a neural network learning problem enables us to exploit an arsenal of highly developed and optimized tools in learning softwares such as tensorflow, theano, or keras.

Despite these advantages, VAMPnets still miss many of the benefits that come with extensions developed for the MSM approach. This includes multi-ensemble Markov models that are superior to single conventional MSMs in terms of sampling rare events by combining data from multiple ensembles^{70–75}, augmented Markov models that combine simulation data with experimental observation data⁷⁶, and statistical error estimators developed for MSMs^{77–79}. Since these methods explicitly use the MSM likelihood, it is currently unclear, how they could be implemented in a deep learning structure such as a VAMPNet. Extending VAMPnets toward these special capabilities is a challenge for future studies.

Finally, a remaining concern is that the optimization of VAMPnets can get stuck in suboptimal local maxima. In other applications of network-based learning, a working knowledge has been established to find which type of network implementation and learning algorithm are most suitable for robust and reproducible learning. For example, it is conceivable that the VAMPNet lobes may benefit from convolutional filters⁸⁰ or different types of transfer functions. Suitably chosen convolutions, as in ref. ⁸¹ may also lead to learned feature transformations that are transferable within a given class of molecules.

Methods

Neural network structure. Each network lobe in Fig. 1 has a number of input nodes given by the data dimension. According to the VAMP variational principle (Sec. A), the output dimension must be at least equal to the number of Koopman singular functions that we want to approximate, i.e., equal to k used in the score function \hat{R}_2 . In most applications, the number of input nodes exceeds the number of output nodes, i.e., the network conducts a dimension reduction. Here, we keep the dimension reduction from layer i with n_i nodes to layer $i+1$ with n_{i+1} nodes constant:

$$\frac{n_i}{n_{i+1}} = \left(\frac{n_{\text{in}}}{n_{\text{out}}} \right)^{1/d}, \quad (16)$$

where d is the network depth, i.e., the number of layers excluding the input layer. Thus, the network structure is fixed by n_{out} and d . We tested different values for d ranging from 2 to 11; for alanine dipeptide, Supplementary Fig. 2b reports the results in terms of the training success rate described in the Results section. Networks have a number of parameters that ranges between 100 and 400,000, most of which are between the first and second layer due to the rapid dimension reduction of the network. To avoid overfitting, we use dropout during training⁸², and select hyper-parameters using the VAMP-2 validation score.

Neural network hyper-parameters. Hyper-parameters include the regularization factors for the weights of the fully connected and the Softmax layer, the dropout probabilities for each layer, the batch size, and the learning rate for the Adam algorithm. Since a grid search in the joint parameter space would have been too computationally expensive, each hyper-parameter was optimized using the VAMP-2 validation score while keeping the other hyper-parameters constant. We started with the regularization factors due to their large effect on the training performance, and observed optimal performance for a factor of 10^{-7} for the fully connected hidden layers and 10^{-8} for the output layer; regularization factors $>10^{-4}$ frequently led to training failure. Subsequently, we tested the dropout probabilities with values ranging from 0 to 50% and found 10% dropout in the first two hidden layers and no dropout otherwise to perform well. The results did not strongly depend on the training batch size, however, more training iterations are necessary for large batches, while small batches exhibit stronger fluctuations in the training score. We found a batch size of 4000 to be a good compromise, with tested values ranging between 100 and 16,000. The optimal learning rate strongly depends on the network topology (e.g., the number of hidden layers and the number of output nodes). In order to adapt the learning rate, we started from an arbitrary rate of 0.05. If no improvement on the validation VAMP-2 score was observed over 10 training iterations, the learning rate was reduced by a factor of 10. This scheme led to better convergence of the training and validation scores and better kinetic model validation compared to using a high learning rate throughout.

The time lag between the input pairs of configurations was selected depending on the number of output nodes of the network: larger lag times are better at isolating the slowest processes, and thus are more suitable with a small number of output nodes. The procedure of choosing network structure and lag time is thus as follows: First, the number of output nodes n and the hidden layers are selected, which determines the network structure as described above. Then, a lag time is chosen in which the largest n singular values (corresponding to the $n-1$ slowest processes) can be trained consistently.

VAMPnet training and validation. We pre-trained the network by minimizing the negative VAMP-1 score during the first third of the total number of epochs, and subsequently optimize the network with VAMP-2 optimization (Sec. B). In order to ensure robustness of the results, we performed 100 network optimization runs for each problem. In each run, the data set was shuffled and randomly split into 90%/10% for training and validation, respectively. To exclude outliers, we then discarded the best 5% and the worst 5% of results. Hyper-parameter optimization was done using the validation score averaged over the remaining runs. Figures report training or validation mean and 95% confidence intervals.

Brownian dynamics simulations. The asymmetric double well and the protein-folding toy model are simulated by over-damped Langevin dynamics in a potential energy function $U(\mathbf{x})$, also known as Brownian dynamics, using an forward Euler integration scheme. The position \mathbf{x}_t is propagated by time step Δt via:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t - \Delta t \frac{\nabla U(\mathbf{x})}{kT} + \sqrt{2\Delta t D} \mathbf{w}_t, \quad (17)$$

where D is the diffusion constant and kT is the Boltzmann constant and temperature. Here, dimensionless units are used and $D=1$, $kT=1$. The elements of the random vector \mathbf{w}_t are sampled from a normal distribution with zero mean and unit variance.

Hardware used and training times. VAMPnets were trained on a single NVIDIA GeForce GTX 1080 GPU, requiring between 20 seconds (for the double-well problem) and 180 seconds for NTL9 for each run.

Code availability. TICA, k -means and MSM analyses were conducted with PyEMMA version 2.4, freely available at <http://www.pyemma.org>. VAMPnets are implemented using the freely available packages keras⁸³ with tensorflow-gpu⁸⁴ as a backend. The code can be obtained at <https://github.com/markovmodel/deeptime>.

Data availability. Data for NTL9 can be requested from the authors of ref. ¹. Data for all other examples is available at <https://github.com/markovmodel/deeptime>.

Received: 14 July 2017 Accepted: 22 November 2017

Published online: 02 January 2018

References

- Lindorff-Larsen, K., Piana, S., Dror, R. O. & Shaw, D. E. How fast-folding proteins fold. *Science* **334**, 517–520 (2011).
- Plattner, N., Doerr, S., Fabritiis, G. D. & Noé, F. Complete protein-protein association kinetics in atomic detail revealed by molecular dynamics simulations and Markov modelling. *Nat. Chem.* **9**, 1005–1011 (2017).

3. Kohlhoff, K. J. et al. Cloud-based simulations on google exacycle reveal ligand modulation of gpcr activation pathways. *Nat. Chem.* **6**, 15–21 (2014).
4. Doerr, S., Harvey, M. J., Noé, F. & De Fabritiis, G. HTMD: high-throughput molecular dynamics for molecular discovery. *J. Chem. Theory Comput.* **12**, 1845–1852 (2016).
5. Ufimtsev, I. S. & Martinez, T. J. Graphical processing units for quantum chemistry. *Comp. Sci. Eng.* **10**, 26–34 (2008).
6. Marx, D. & Hutter, J. in *Modern Methods and Algorithms of Quantum Chemistry of NIC Series Vol. 1*, 301–449 Ab initio molecular dynamics: theory and implementation (ed Grotendorst, J.) (John von Neumann Institute for Computing, Jülich, 2000).
7. Schütte, C., Fischer, A., Huisings, W. & Deuflhard, P. A direct approach to conformational dynamics based on hybrid monte carlo. *J. Comput. Phys.* **151**, 146–168 (1999).
8. Prinz, J.-H. et al. Markov models of molecular kinetics: generation and validation. *J. Chem. Phys.* **134**, 174105 (2011).
9. Swope, W. C., Pitera, J. W. & Suits, F. Describing protein folding kinetics by molecular dynamics simulations: 1. *Theory J. Phys. Chem. B* **108**, 6571–6581 (2004).
10. Noé, F., Horenko, I., Schütte, C. & Smith, J. C. Hierarchical analysis of conformational dynamics in biomolecules: transition networks of metastable states. *J. Chem. Phys.* **126**, 155102 (2007).
11. Chodera, J. D. et al. Automatic discovery of metastable states for the construction of Markov models of macromolecular conformational dynamics. *J. Chem. Phys.* **126**, 155101 (2007).
12. Buchete, N. V. & Hummer, G. Coarse master equations for peptide folding dynamics. *J. Phys. Chem. B* **112**, 6057–6069 (2008).
13. Scherer, M. K. et al. PyEMMA 2: a software package for estimation, validation and analysis of Markov models. *J. Chem. Theory Comput.* **11**, 5525–5542 (2015).
14. Harrigan, M. P. et al. Msmbuilder: statistical models for biomolecular dynamics. *Biophys. J.* **112**, 10–15 (2017).
15. Humphrey, W., Dalke, A. & Schulten, K. Vmd—visual molecular dynamics. *J. Mol. Graph.* **14**, 33–38 (1996).
16. McGibbon, R. T. et al. Mdtraj: a modern open library for the analysis of molecular dynamics trajectories. *Biophys. J.* **109**, 1528–1532 (2015).
17. Noé, F. & Nüske, F. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Model. Simul.* **11**, 635–655 (2013).
18. Nüske, F., Keller, B. G., Pérez-Hernández, G., Mey, A. S. J. S. & Noé, F. Variational approach to molecular kinetics. *J. Chem. Theory Comput.* **10**, 1739–1752 (2014).
19. Pérez-Hernandez, G., Paul, F., Giorgino, T., D Fabritiis, G. & Noé, F. Identification of slow molecular order parameters for Markov model construction. *J. Chem. Phys.* **139**, 015102 (2013).
20. 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).
21. Molgedey, L. & Schuster, H. G. Separation of a mixture of independent signals using time delayed correlations. *Phys. Rev. Lett.* **72**, 3634–3637 (1994).
22. Ziehe, A. & Müller, K.-R. in *ICANN Vol. 98*, 675–680 TDSEP—an efficient algorithm for blind separation using time structure (Springer, London, 1998).
23. Harmeling, S., Ziehe, A., Kawanabe, M. & Müller, K.-R. Kernel-based nonlinear blind source separation. *Neural Comput.* **15**, 1089–1124 (2003).
24. Mezić, I. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynam.* **41**, 309–325 (2005).
25. Schmid, P. J. & Sesterhenn, J. Dynamic mode decomposition of numerical and experimental data. In *61st Annual Meeting of the APS Division of Fluid Dynamics* (American Physical Society, 2008).
26. Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L. & Kutz, J. N. On dynamic mode decomposition: theory and applications. *J. Comput. Dyn.* **1**, 391–421 (2014).
27. Williams, M. O., Kevrekidis, I. G. & Rowley, C. W. A data-driven approximation of the koopman operator: extending dynamic mode decomposition. *J. Nonlinear Sci.* **25**, 1307–1346 (2015).
28. Wu, H. et al. Variational koopman models: slow collective variables and molecular kinetics from short off-equilibrium simulations. *J. Chem. Phys.* **146**, 154104 (2017).
29. Noé, F. & Clementi, C. Collective variables for the study of long-time kinetics from molecular trajectories: theory and methods. *Curr. Opin. Struc. Biol.* **43**, 141–147 (2017).
30. Klus, S. et al. Data-driven model reduction and transfer operator approximation. Preprint at <http://arXiv:1703.10112> (2017).
31. Noé, F. & Clementi, C. Kinetic distance and kinetic maps from molecular dynamics simulation. *J. Chem. Theory Comput.* **11**, 5002–5011 (2015).
32. Noé, F., Banisch, R. & Clementi, C. Commute maps: separating slowly-mixing molecular configurations for kinetic modeling. *J. Chem. Theory Comput.* **12**, 5620–5630 (2016).
33. Bowman, G. R., Pande, V. S. & Noé, F. in *Advances in Experimental Medicine and Biology* Vol. 797 An introduction to Markov state models and their application to long timescale molecular simulation (Springer, Heidelberg, 2014).
34. Husic, B. E. & Pande, V. S. Ward clustering improves cross-validated Markov state models of protein folding. *J. Chem. Theory Comput.* **13**, 963–967 (2017).
35. Sheong, F. K., Silva, D.-A., Meng, L., Zhao, Y. & Huang, X. Automatic state partitioning for multibody systems (APM): an efficient algorithm for constructing Markov state models to elucidate conformational dynamics of multibody systems. *J. Chem. Theory Comput.* **11**, 17–27 (2015).
36. Wu, H. & Noé, F. Gaussian Markov transition models of molecular kinetics. *J. Chem. Phys.* **142**, 084104 (2015).
37. Weber, M., Fackeldey, K. & Schütte, C. Set-free Markov state model building. *J. Chem. Phys.* **146**, 124133 (2017).
38. Bowman, G. R., Beauchamp, K. A., Boxer, G. & Pande, V. S. Progress and challenges in the automated construction of Markov state models for full protein systems. *J. Chem. Phys.* **131**, 124101 (2009).
39. Trendelkamp-Schroer, B., Wu, H., Paul, F. & Noé, F. Estimation and uncertainty of reversible Markov models. *J. Chem. Phys.* **143**, 174101 (2015).
40. Kube, S. & Weber, M. A coarse graining method for the identification of transition rates between molecular conformations. *J. Chem. Phys.* **126**, 024103 (2007).
41. Yao, Y. et al. Hierarchical nyström methods for constructing Markov state models for conformational dynamics. *J. Chem. Phys.* **138**, 174106 (2013).
42. Fackeldey, K. & Weber, M. Genpca—Markov state models for non-equilibrium steady states. *WIAS Rep.* **29**, 70–80 (2017).
43. Gerber, S. & Horenko, I. Toward a direct and scalable identification of reduced models for categorical processes. *Proc. Natl Acad. Sci. USA* **114**, 4863–4868 (2017).
44. Hummer, G. & Szabo, A. Optimal dimensionality reduction of multistate kinetic and Markov-state models. *J. Phys. Chem. B* **119**, 9029–9037 (2015).
45. Orioli, S. & Faccioli, P. Dimensional reduction of Markov state models from renormalization group theory. *J. Chem. Phys.* **145**, 124120 (2016).
46. Noé, F., Wu, H., Prinz, J.-H. & Plattner, N. Projected and hidden Markov models for calculating kinetics and metastable states of complex molecules. *J. Chem. Phys.* **139**, 184114 (2013).
47. Wu, H. & Noé, F. Variational approach for learning Markov processes from time series data. Preprint at <http://arXiv:1707.04659> (2017).
48. McGibbon, R. T. & Pande, V. S. Variational cross-validation of slow dynamical modes in molecular kinetics. *J. Chem. Phys.* **142**, 124105 (2015).
49. LeCun, Y., Bengio, Y. & Hinton, G. E. Deep learning. *Nature* **521**, 436–444 (2015).
50. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems* Vol. 1, 1097–1105 (Curran Associates, Inc., 2012).
51. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
52. Perez-Hernandez, G. & Noé, F. Hierarchical time-lagged independent component analysis: computing slow modes and reaction coordinates for large molecular systems. *J. Chem. Theory Comput.* **12**, 6118–6129 (2016).
53. Nüske, F., Schneider, R., Vitalini, F. & Noé, F. Variational tensor approach for approximating the rare-event kinetics of macromolecular systems. *J. Chem. Phys.* **144**, 054105 (2016).
54. Koopman, B. Hamiltonian systems and transformations in hilbert space. *Proc. Natl Acad. Sci. USA* **17**, 315–318 (1931).
55. Knoch, F. & Speck, T. Cycle representatives for the coarse-graining of systems driven into non-equilibrium steady state. *New J. Phys.* **17**, 115004 (2015).
56. Wang, H. & Schütte, C. Building Markov state models for periodically driven non-equilibrium systems. *J. Chem. Theory Comput.* **11**, 1819–1831 (2015).
57. Horenko, I., Hartmann, C., Schütte, C., Noé, F. Data-based parameter estimation of generalized multidimensional Langevin processes. *Phys. Rev. E* **76**, 016706 (2007).
58. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals* **2**, 303–314 (1989).
59. Eigen, D., Rolfe, J., Fergus, R. & LeCun, Y. Understanding deep architectures using a recursive convolutional network. Preprint at <http://arXiv:1312.1847> (2014).
60. Ranzato, M., Poultney, C., Chopra, S. & LeCun, Y. in *Advances in Neural Information Processing Systems 18* (ed Platt, J. et al.) Efficient learning of sparse representations with an energy-based model (MIT Press, 2006).
61. Bengio, Y., Lamblin, P., Popovici, D. & Larochelle, H. in *Advances in Neural Information Processing Systems 19*, Vol. 19, 153 Greedy layer-wise training of deep networks (MIT Press, 2007).
62. Galen, A., Arora, R., Bilmes, J. & Livescu, K. Deep canonical correlation analysis. In *ICML'13 Proceedings of the 30th International Conference on International Conference on Machine Learning* Vol. 28, III-1247–III-1255 (2013).
63. Röblitz, S. & Weber, M. Fuzzy spectral clustering by PCCA+: application to Markov state models and data classification. *Adv. Data Anal. Classif.* **7**, 147–179 (2013).
64. Sarich, M., Noé, F. & Schütte, C. On the approximation quality of Markov state models. *Multiscale Model. Simul.* **8**, 1154–1177 (2010).

- 65 Noé, F., Schütte, C., Vanden-Eijnden, E., Reich, L. & Weikl, T. R. Constructing the full ensemble of folding pathways from short off-equilibrium simulations. *Proc. Natl Acad. Sci. USA* **106**, 19011–19016 (2009).
- 66 Hahnloser, R. L. T. On the piecewise analysis of networks of linear threshold neurons. *Neural Netw.* **11**, 691–697 (1998).
- 67 Nair, V. & Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* Vol. 27, 807–814 (ACM, New York, 2010).
- 68 Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. Preprint at <http://arXiv.org/1412.6980> (2014).
- 69 Nüske, F., Wu, H., Wehmeyer, C., Clementi, C. & Noé, F. Markov state models from short non-equilibrium simulations—analysis and correction of estimation bias. Preprint at <http://arXiv:1701.01665> (2017).
- 70 Wu, H., Paul, F., Wehmeyer, C. & Noé, F. Multiensemble Markov models of molecular thermodynamics and kinetics. *Proc. Natl Acad. Sci. USA* **113**, E3221–E3230 (2016).
- 71 Wu, H., Mey, A. S. J. S., Rosta, E. & Noé, F. Statistically optimal analysis of state-discretized trajectory data from multiple thermodynamic states. *J. Chem. Phys.* **141**, 214106 (2014).
- 72 Chodera, J. D., Swope, W. C., Noé, F., Prinz, J.-H. & Pande, V. S. Dynamical reweighting: improved estimates of dynamical properties from simulations at multiple temperatures. *J. Phys. Chem.* **134**, 244107 (2011).
- 73 Prinz, J.-H. et al. Optimal use of data in parallel tempering simulations for the construction of discrete-state Markov models of biomolecular dynamics. *J. Chem. Phys.* **134**, 244108 (2011).
- 74 Rosta, E. & Hummer, G. Free energies from dynamic weighted histogram analysis using unbiased Markov state model. *J. Chem. Theory Comput.* **11**, 276–285 (2015).
- 75 Mey, A. S. J. S., Wu, H. & Noé, F. xTRAM: estimating equilibrium expectations from time-correlated simulation data at multiple thermodynamic states. *Phys. Rev. X* **4**, 041018 (2014).
- 76 Olsson, S., Wu, H., Paul, F., Clementi, C. & Noé, F. Combining experimental and simulation data of molecular processes via augmented Markov models. *Proc. Natl Acad. Sci. USA* **114**, 8265–8270 (2017).
- 77 Hinrichs, N. S. & Pande, V. S. Calculation of the distribution of eigenvalues and eigenvectors in Markovian state models for molecular dynamics. *J. Chem. Phys.* **126**, 244101 (2007).
- 78 Noé, F. Probability distributions of molecular observables computed from Markov models. *J. Chem. Phys.* **128**, 244103 (2008).
- 79 Chodera, J. D. & Noé, F. Probability distributions of molecular observables computed from Markov models. ii: Uncertainties in observables and their time-evolution. *J. Chem. Phys.* **133**, 105102 (2010).
- 80 LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
- 81 Schütt, K. T. et al. Moleculenet: a continuous-filter convolutional neural network for modeling quantum interactions. Preprint at <http://arXiv:1706.08566> (2017).
- 82 Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
- 83 Chollet, F. et al. Keras. <https://github.com/fchollet/keras> (2015).
- 84 Abadi, M. et al. TensorFlow: large-scale machine learning on heterogeneous systems. Preprint at <http://arXiv.org:1603.04467> (2015).

Acknowledgements

We are grateful to Cecilia Clementi, Robert T. McGibbon, and Max Welling for valuable discussions. This work was funded by Deutsche Forschungsgemeinschaft (SFB958/A04, Transregio 186/A12, SFB 1114/A4, NO 825/4–1 as part of research group 2518) and European Research Commission (ERC StG 307494 “pcCell”).

Author contributions

A.M. and L.P. conducted research and developed software. H.W. and F.N. designed research and developed theory. All authors wrote the paper.

Additional information

Supplementary Information accompanies this paper at <https://doi.org/10.1038/s41467-017-02388-1>.

Competing interests: The authors declare no competing financial interests.

Reprints and permission information is available online at <http://npg.nature.com/reprintsandpermissions/>

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2017