Dissertation

Deep learning of the dynamics of complex systems with its applications to biochemical molecules

eingereicht von

Andreas Mardt

Berlin, 2022

zur Erlangung des Doktorgrades der Naturwissenschaften am Fachbereich Mathematik und Informatik
- Dr. rer. nat. -

Freie Universität Berlin

Erstgutachter:	Prof. Dr. Frank Noé
Zweitgutachterin:	Prof. Dr. Bettina G. Keller
Tag der Disputation:	28. April 2023

Selbstständigkeitserklärung

Name: Mardt
Vorname: Andreas

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde weder in gleicher noch ähnlicher Form in einem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Datum: Unterschrift:

Abstract

Recent advancements in deep learning have revolutionized method development in several scientific fields and beyond. One central application is the extraction of equilibrium structures and long- timescale kinetics from molecular dynamics simulations, i.e. the well-known sampling problem. Previous state-of-the art methods employed a multi-step handcrafted data processing pipeline resulting in Markov state models (MSM), which can be understood as an approximation of the underlying Koopman operator. However, this approach demands choosing a set of features characterizing the molecular structure, methods and their parameters for dimension reduction to collective variables and clustering, and estimation strategies for MSMs throughout the processing pipeline. As this requires specific expertise, the approach is ultimately inaccessible to a broader community.
In this thesis we apply deep learning techniques to approximate the Koopman operator in an end-to-end learning framework by employing the variational approach for Markov processes (VAMP). Thereby, the framework bypasses the multi-step process and automates the pipeline while yielding a model similar to a coarse-grained MSM. We further transfer advanced techniques from the MSM field to the deep learning framework, making it possible to (i) include experimental evidence into the model estimation, (ii) enforce reversibility, and (iii) perform coarse-graining. At this stage, post-analysis tools from MSMs can be borrowed to estimate rates of relevant rare events. Finally, we extend this approach to decompose a system into its (almost) independent subsystems and simultaneously estimate dynamical models for each of them, making it much more data efficient and enabling applications to larger proteins.
Although our results solely focus on protein dynamics, the application to climate, weather, and ocean currents data is an intriguing possibility with potential to yield new insights and improve predictive power in these fields.

Keywords: Koopman operator, deep learning, molecular dynamics, neural networks, generative models, physical constraints, decomposition

Contents

1. Introduction **1**

2. Theory **5**

2.1. Koopman operator 5

2.1.1. Finite rank approximation 6

2.1.2. Variational approach for Markov processes 7

2.1.3. Estimation algorithms 8

2.1.4. Validation 10

2.1.5. Connection of transition density and VAMP-E score 10

2.1.6. Reversible Koopman operators 12

2.1.7. Connection Markov state model and Koopman model 13

2.1.8. Decompostion into independent subsystems 14

2.2. Artificial neural networks 15

2.2.1. Feed forward neural network 17

2.2.2. Invariances and transferable architectures 17

2.2.3. NN as transformers between probability distributions 18

3. Methods **19**

3.1. VAMPnets for deep learning of molecular kinetics 20

3.2. Deep Markov state model 21

3.2.1. Making the deep MSM generative 25

3.3. Deep reversible Koopman model 27

3.3.1. Incorporate experimental measurements into model estimation 31

3.4. Coarse-graining with VAMP 33

3.5. iVAMPnets for decomposing macromolecules into independent Markovian domains 35

3.6. Attention mechanism 38

3.6.1. Attention mechanism for proteins 39

3.6.2. iVAMPnets with attention . . . 40
3.7. Validation . . . 41
3.7.1. Chapman-Kolmogorov equation . . . 41
3.7.2. Implied timescales . . . 42
3.7.3. Eigenfunctions of the Markov state model . . . 42

4. Vampnets: Deep learning of molecular kinetics 45
4.1. Results . . . 46
4.1.1. Asymmetric double well potential . . . 46
4.1.2. Protein folding model . . . 46
4.1.3. Alanine dipeptide . . . 49
4.1.4. Choice of lag time, network depth and number of output states 51
4.1.5. VAMPnets learn to transform Cartesian to torsion coordinates 53
4.1.6. NTL9 Protein folding dynamics . . . 55
4.2. Methods . . . 55
4.2.1. Neural network structure . . . 55
4.2.2. Neural network hyperparameters . . . 56
4.2.3. VAMPnet training and validation . . . 57
4.2.4. Brownian dynamics simulations . . . 57
4.3. Discussion . . . 58

5. Deep Generative Markov State Models 61
5.1. Results . . . 62
5.1.1. Diffusion in Prinz potential . . . 62
5.1.2. Alanine dipeptide . . . 64
5.2. Methods . . . 67
5.3. Discussion . . . 69

6. Deep reversible Koopman model 71
6.1. Results . . . 72
6.1.1. Reversible VAMPnets and reversible deep MSMs obtain transition matrices with real eigenvalues and nonnegative entries . 73
6.1.2. Reversible VAMPnets converge to unbiased timescales and state probabilities for biased training data . . . 74
6.1.3. Approximation of the true eigenfunctions . . . 76
6.1.4. Performance on a larger system . . . 77

6.2. Methods 77
6.3. Discussion 78

7. Coarse-graining and experimental constraints 79
7.1. Results 80
7.1.1. Obtaining real eigenvalues and positive entries in the transition matrix via a RevDMSM 80
7.1.2. Building an MSM and testing its validity for a VAMPnet and a RevDMSM 81
7.1.3. Building an hierarchical model with an interpretable attention mechanism 81
7.1.4. Approximation of the leading eigenfunctions via a RevDMSM 85
7.1.5. Estimation of folding rates 86
7.1.6. Estimating deep MSMs with experimental restraints 86
7.2. Methods 89
7.2.1. Matching two dependent time correlations 89
7.2.2. Connection between timescales and folding/unfolding rates . . 90
7.2.3. Pretraining the VAMPnet 90
7.2.4. Data manipulation routine for the observable model 91
7.3. Discussion 92

8. iVAMPnets 95
8.1. Results 96
8.1.1. Toy model with 2 independent subsystems 97
8.1.2. 10D hypercube toy model 99
8.1.3. Synaptotagmin-C2A 100
8.1.4. Counterexample and post-training independence assessment . 104
8.1.5. Testing statistical independence of the learned dynamical subsystems 105
8.2. Methods 105
8.3. Discussion 107

9. Summary and Perspective 111

A. Zusammenfassung 131

Nomenclature

Latin Letters

Variable	Meaning
$a(\mathbf{x})$	microscopic observable
$\mathbf{A}$	Attention network
$\mathbf{C}_{00}$	correlation matrix of $\boldsymbol{\chi}_0$
$\mathbf{C}_{11}$	correlation matrix of $\boldsymbol{\chi}_1$
$\mathbf{C}_{01}$	cross correlation matrix of $\boldsymbol{\chi}_0$ and $\boldsymbol{\chi}_1$
D	energy distance
$\mathbf{e}$	one-hot encoded vector
$\mathbf{f}$	approximation of the left singular functions of the Koopman operator
$\mathbf{g}$	approximation of the right singular functions of the Koopman operator
$\mathbf{G}$	Generative network
$\mathbf{K}$	diagonal matrix with the approximated singular values of the Koopman operator
$\mathcal{K}$	true Koopman operator
$\hat{\mathcal{K}}$	the low rank Koopman operator
$\bar{\mathbf{K}}$	Koopman matrix for normalized feature functions
$\mathbf{K}_\chi$	Koopman matrix
L	loss function
LL	log-likelihood
$\mathbf{M}$	coarse-graining matrix
$p_\tau(\mathbf{y}\|\mathbf{x})$	true transition density
$\hat{p}_\tau(\mathbf{y}\|\mathbf{x})$	approximated transition density
$\mathbf{q}$	$q_i(\mathbf{y};\tau)$ probability to jump to configuration $\mathbf{y}$ given the system was in state i a time window τ before

Variable	Meaning
$\mathbf{R}$	right eigenvector matrix
$\mathcal{R}_E$	VAMP-E score
$\mathcal{R}_r$	VAMP-r score
$\mathbf{S}$	estimates the alignment of a propagated $\boldsymbol{\chi}_0$ with $\boldsymbol{\chi}_1$
t_i	timescale of process i
$\mathbf{T}$	transition matrix
$\mathbf{u}$	reweighting vector for the state space
$\mathbf{U}$	matrix to map $\boldsymbol{\chi}_0$ on the left singular functions
$\mathbf{V}$	matrix to map $\boldsymbol{\chi}_1$ on the right singular functions
$\mathbf{x}_t$	configuration at time t
$\mathbf{y}$	configuration at time $t + \tau$

Greek Letters

Variable	Meaning
$\boldsymbol{\gamma}$	neural network to construct $\boldsymbol{q}$
$\bar{\boldsymbol{\gamma}}$	expectation value of $\boldsymbol{\gamma}$
$\boldsymbol{\epsilon}$	random noise vector
$\boldsymbol{\eta}$	neural network to construct $\boldsymbol{\chi}$
$\boldsymbol{\lambda}$	eigenvalues of the transition matrix
μ	$\mu(\mathbf{y})$ the stationary probability of configuration $\mathbf{y}$
ξ	regularization strength in the loss function
$\boldsymbol{\pi}$	the stationary vector in state space
ρ_0	empirical distribution of configurations at time t
ρ_1	empirical distribution of configurations at time $t + \tau$
$\boldsymbol{\sigma}$	singular values of the true Koopman operator
$\boldsymbol{\Sigma}$	correlation matrix in equilibrium
τ	lag time
$\boldsymbol{\phi}$	right singular functions of the true Koopman operator
$\boldsymbol{\varphi}$	eigenfunctions of the reversible Koopman operator
$\boldsymbol{\chi}(\mathbf{x})$	mapping from the configuration $\mathbf{x}$ to the state space
$\boldsymbol{\chi}_0$	feature functions of the Koopman model at time t
$\boldsymbol{\chi}_1$	feature functions of the Koopman model at time $t + \tau$

Variable	Meaning
ψ	left singular functions of the true Koopman operator

Abbreviation

Abbreviation	Meaning
CCA	canonical correlation analysis
CK	Chapman-Kolmogorov
DMD	dynamic mode decomposition
ED	energy distance
ELU	exponential linear unit
F	folded state
GNN	graph neural network
IMD	independent Markov decomposition
M	misfolded state
MD	Molecular Dynamic
ML	maximum likelihood
MSM	Markov state model
NN	artificial neural network
PF	partially folded state
ReLU	rectified linear unit
RevDMSM	reversible deep Markov state model
RevVAMPnet	reversible VAMPnet
SVD	singular value decomposition
SymVAMPnet	symmettrized VAMPnet
TICA	time-lagged independent component analysis
TPT	transition path theory
U	unfolded state
VAMP	variational approach for Markov processes

1. Introduction

Research fields such as climate prediction, fluid dynamics, molecular dynamics, and drug discovery are confronted with the problem of understanding the thermodynamics and kinetics of complex systems. Rapid advances in computing power and simulation techniques allow the generation of extensive simulation data. Therefore, it is of great interest to develop methods capable to automatically extract the statistically relevant information.

The difficulties in analyzing such problems arise from the fact that they are typically observed in a high dimensional space, where the dynamics are nonlinear. Consequently, a wide range of dynamical models have been developed, where the complex nonlinear dynamics in the observed space are lifted by non-linear functions into a space where the dynamics become linear. The approach can be summarized by the following equation:

$$\mathbb{E}[\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})] = \mathbf{K}_\chi^T \mathbb{E}[\boldsymbol{\chi}_0(\mathbf{x}_t)], \tag{1.1}$$

where $\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})$ and $\boldsymbol{\chi}_0(\mathbf{x}_t)$ transform the observed state variable $\mathbf{x}$ into a feature space, where the dynamics are described linearly by the matrix $\mathbf{K}_\chi$. The expectation value $\mathbb{E}$ is evaluated over time and accounts for stochasticity. By special choices for the feature transformation the model can be linked to specific methods. Choosing $\boldsymbol{\chi}_1 = \boldsymbol{\chi}_0 = \mathbf{x}$ the model describes dynamic mode decomposition (DMD) [1–3]. Restricting the feature functions to indicator functions, thereby clustering the observable state into states, Eq. 1.1 describes the propagation of a Markov State Model (MSM) or equivalently of Ulam's Galerkin method [4–7]. There, the propagation matrix $\mathbf{K}_\chi$ encodes the probability to transition between the defined states, hence called transition matrix.

In general, the equation describes a low rank approximation of the Koopman operator [8, 9]. In practice, the optimal choices for feature transformation are sought, which are given by the singular functions of the Koopman operator [9–13]. Therefore, several methods to approximate these functions in a variational [7, 14–18] approach were developed, which include dictionary [19, 20], kernel [21–23], and tensor [24, 25]

based methods.
Another class of function approximator has recently roused a great deal of excitement: artificial neural networks (NN), which deep learning is based on. Although invented over 70 years ago [26], its great success started only two decades ago [27–29]. While the universal approximation theorem [30] was a great theoretical result, further developments in hardware and software, namely the architecture and training routines, were necessary to pave the path to its broad application today.
In this thesis we explore the capability of NNs to approximate the singular functions of the Koopman operator and thereby finding optimal choices for the feature functions for Eq. 1.1. On one hand, the universal approximation theorem promises that given enough data the true singular functions can be arbitrarily closely approximated. On the other hand, the flexibility of the networks architecture allows to impose further constraints, e.g. reversibility.
We apply the proposed methods to molecular dynamic (MD) simulations of biomolecules, which allow to study the thermodynamics and kinetics at an atomistic scale [31–35]. As these simulations are often performed in thermal equilibrium without the influence of external forces, the second law of thermodynamics states that no work can be extracted from such a system. As a consequence, the absolute probability of observing a transition from a configuration $\mathbf{x}$ at time t to point $\mathbf{y}$ at time $t + \tau$ must be equal to the reverse transition, a condition called detailed balance. Hence, no closed loops of probability fluxes can be exploited to extract work; such systems are said to be statistically reversible. Any model approximating a reversible system should therefore comply with the detailed balance constraint.
Taking these thermodynamic constraints into account, we develop network architectures which not only allow to enforce reversibility but also facilitate to constrain the propagation matrix $\mathbf{K}_\chi$ to be a transition matrix. This empowers the application of post analysis tools from the MSM field, e.g. finding the most probable folding paths and folding rates via transition path theory [36–38].
Further developments in the field of MSMs can be transferred to the deep learning framework. On the one hand, coarse-graining approaches using spectral clustering allow to study the system on different degrees of detail [39–41]. By constructing a specific layer, we realize coarse-graining with NNs by still optimizing for the singular functions of the Koopman operator.
On the other hand, MD simulations can suffer from biases in the underlying classical force field, where it was shown that incorporating existing experimental information

into the model estimation can counteract these biases [42–49]. By changing the objective of the NN to not exclusively approximate the singular functions of the Koopman operator but also fulfill the experimental constraints, we adapt the deep learning framework to the aforementioned methods.
Finally, as the interest shifts to larger proteins, a fundamental problem arises when approximating the kinetics of the protein from a global perspective. Larger systems might display localized conformational changes [50] or consist of weakly coupled or independent subsystems. A global description of the system consists then of the combination of the description of all subsystems, which implies that the number of possible global states grows exponentially with the number of subsystems [51, 52]. Hence, any approach describing the global state explicitly is fundamentally unscalable. The solution to this problem is twofold: first, divide the protein into Markovian subsystems, each described by their nearly independent Koopman operators, and second, learn the coupling between them. In this thesis, we present a framework focusing on the former. Extracting the coupling between the weakly coupled subsystems might be solved by future work in a consecutive step borrowing ideas from Olsson & Noé [51]. They treated the global system as an Ising model, where the states or "spins" of the subsystems are connected via a dynamic graphical model representing the coupling.

2. Theory

The theoretical background of this thesis is split into two major sections: the Koopman operator and neural networks. The former covers the development of the variational principle - a summary of the work of Wu *et al.* [18] - and it also covers the properties of the Koopman operator in the case of independent systems. The latter is a compact introduction into the concept of neural networks, how to train and validate them, how problem specific architectures are developed, and how they can be used to sample from a probability distribution implicitly given by data.

2.1. Koopman operator

A Markov process of configurations $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, which live in a potentially high-dimensional space, is fully described by its transition density:

$$\mathbb{P}(\mathbf{x}_{t+\tau} = \mathbf{y} | \mathbf{x}_t = \mathbf{x}) = p_\tau(\mathbf{y}|\mathbf{x}), \tag{2.1}$$

which is the probability of transitioning to configuration $\mathbf{y}$ at time $t + \tau$ given that the system was at state $\mathbf{x}$ at time t. Based on the transition density the time evolution of general observable functions $\chi(\mathbf{x})$ can be characterized as:

$$\mathbb{E}[\chi(\mathbf{x}_{t+\tau})|\mathbf{x}_t = \mathbf{x}] = (\mathcal{K}_\tau \chi)(\mathbf{x}) \triangleq \int p_\tau(\mathbf{y}|\mathbf{x})\chi(\mathbf{y})\,\mathrm{d}\mathbf{y}. \tag{2.2}$$

The integral operator $\mathcal{K}_\tau$ is called the *Koopman operator* and is able to fully describe the Markovian dynamics [9].
Here, we consider $\mathcal{K}_\tau$ to be a Hilbert-Schmidt operator from $\mathcal{L}^2_{\rho_1} = \{g | \langle g, g \rangle_{\rho_1} < \infty\}$ to $\mathcal{L}^2_{\rho_0} = \{f | \langle f, f \rangle_{\rho_0} < \infty\}$, where ρ_0 and ρ_1 are the empirical distributions of $\mathbf{x}_t$ and $\mathbf{x}_{t+\tau}$ of all transition pairs $\{(\mathbf{x}_t, \mathbf{x}_{t+\tau})\}$ observed in a time series. The weighted inner product is defined as $\langle f, g \rangle_{\rho_i} = \int f(\mathbf{x})g(\mathbf{x})\rho_i(\mathbf{x})\,\mathrm{d}\mathbf{x}$.
The Koopman operator is linear but infinite-dimensional and since it is Hilbert-

Schmidt the following SVD exists:

$$\mathcal{K}_\tau \chi = \sum_{i=1}^{\infty} \sigma_i \langle \chi, \phi_i \rangle_{\rho_1} \psi_i, \tag{2.3}$$

with the singular values σ_i and the corresponding left and right singular functions ψ_i, ϕ_i. Due to the orthonormality of the right singular functions it directly follows:

$$\mathbb{E}[\phi_i(\mathbf{x}_{t+\tau})] = \sigma_i \mathbb{E}[\psi_i(\mathbf{x}_t)], \tag{2.4}$$

which shows that the time evolution of a general observable $\chi \in \mathcal{L}^2_{\rho_1}$, which can be written as a linear combination of right singular functions $\chi = \sum_{i=1}^{\infty} c_i \phi_i$, is given by:

$$\begin{aligned} \mathbb{E}[\chi(\mathbf{x}_{t+\tau})] &= \sum_{i=1}^{\infty} c_i \mathbb{E}[\phi_i(\mathbf{x}_{t+\tau})] \\ &= \sum_{i=1}^{\infty} c_i \sigma_i \mathbb{E}[\psi_i(\mathbf{x}_t)]. \end{aligned}$$

2.1.1. Finite rank approximation

For analysis, it is favorable to consider a finite rank approximation of the true Koopman operator, which is often justified by choosing a large enough lagtime τ, where the essential part of the dynamics becomes finite dimensional.
The optimal approximation of the Koopman operator with the smallest modeling error in Hilbert-Schmidt norm with dimension k is given by

$$\hat{\mathcal{K}}_\tau \chi = \sum_{i=1}^{k} \sigma_i \langle \chi, \phi_i \rangle_{\rho_1} \psi_i, \tag{2.5}$$

where σ_i are the k-largest singular values. The first singular component is always given by $(\sigma_1, \phi_1, \psi_1) = (1, \mathbf{1}, \mathbf{1})$ [13].
With the convention of $\boldsymbol{\psi} = (\psi_1, ..., \psi_k)^T$, $\boldsymbol{\phi} = (\phi_1, ..., \phi_k)^T$, and $\mathbf{K} = \mathrm{diag}(\sigma_1, ..., \sigma_k)$ Eq.2.4 can be written in matrix form:

$$\mathbb{E}[\boldsymbol{\phi}(\mathbf{x}_{t+\tau})] = \mathbf{K}^T \mathbb{E}[\boldsymbol{\psi}(\mathbf{x}_t)]. \tag{2.6}$$

Due to the finite dimension, the space of observables that can be propagated by the projected operator is restricted to functions that can be written as linear combina-

tions of the right singular functions:

$$\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau}) = (\mathbf{V}^{-1})^T \boldsymbol{\phi}(\mathbf{x}_{t+\tau}), \tag{2.7}$$

where we assume that the functions $\boldsymbol{\chi}_1$ are linearly independent and therefore $\mathbf{V} \in \mathbb{R}^{k\times k}$ invertible.
If the result of the propagation should not be written in terms of $\boldsymbol{\psi}$ but in a linear combination

$$\boldsymbol{\chi}_0(\mathbf{x}_t) = (\mathbf{U}^{-1})^T \boldsymbol{\psi}(\mathbf{x}_t), \tag{2.8}$$

the propagation reads:

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})] &= (\mathbf{V}^{-1})^T \mathbb{E}[\boldsymbol{\phi}(\mathbf{x}_{t+\tau})] && (2.9)\\
&= (\mathbf{V}^{-1})^T \mathbf{K}^T \mathbb{E}[\boldsymbol{\psi}(\mathbf{x}_t)] && (2.10)\\
&= (\mathbf{V}^{-1})^T \mathbf{K}^T \mathbf{U}^T \mathbb{E}[\boldsymbol{\chi}_0(\mathbf{x}_t)] && (2.11)\\
&= (\mathbf{U}\mathbf{K}\mathbf{V}^{-1})^T \mathbb{E}[\boldsymbol{\chi}_0(\mathbf{x}_t)] && (2.12)\\
&= \mathbf{K}_\chi^T \mathbb{E}[\boldsymbol{\chi}_0(\mathbf{x}_t)]. && (2.13)
\end{aligned}$$

The results let us interpret Eq.1.1 as a finite-rank approximation of the Koopman operator and therefore $\mathbf{K}_\chi$ is often referred to as Koopman matrix [9, 53].

2.1.2. Variational approach for Markov processes

The previous subsection implies that, if the feature functions $\boldsymbol{\chi}_0, \boldsymbol{\chi}_1$ can span the left and right singular functions $\boldsymbol{\psi}(\mathbf{x}_t) = \mathbf{U}^T \boldsymbol{\chi}_0(\mathbf{x}_t)$, they represent an optimal finite-rank approximation of the Koopman operator. However, there is no recipe how to construct these feature functions. Eq. 1.1 cannot help us solve this problem, since taking a regression error would lead to an uninformative model with constant feature functions.
Subsequently, Wu and Noé proposed a variational approach for Markov processes (VAMP), where optimal feature functions can be constructed by maximizing the generalized Rayleigh quotient [13]. Given test functions $\mathbf{f}$ and $\mathbf{g}$ the k dominant singular components of a Koopman operator are the solution of the following max-

imization problem:

$$\sum_{i=1}^{k} \sigma_i^r = \max_{\mathbf{f},\mathbf{g}} \mathcal{R}_r[\mathbf{f},\mathbf{g}], \tag{2.14}$$

$$\text{s.t. } \langle f_i, f_j \rangle_{\rho_0} = 1_{i=j}, \tag{2.15}$$

$$\langle g_i, g_j \rangle_{\rho_1} = 1_{i=j}, \tag{2.16}$$

where $r \geq 1$ and the maximal value of $\mathcal{R}_r$, which is called VAMP-r score, is achieved by the true singular functions $f_i = \psi_i$ and $g_i = \phi_i$ defined as:

$$\max_{\mathbf{f},\mathbf{g}} \mathcal{R}_r[\mathbf{f},\mathbf{g}] = \max_{\mathbf{f},\mathbf{g}} \sum_{i=1}^{k} \langle f_i, \mathcal{K}_\tau g_i \rangle_{\rho_0}^r \tag{2.17}$$

$$= \sum_{i=1}^{k} \langle \psi_i, \mathcal{K}_\tau \phi_i \rangle_{\rho_0}^r \tag{2.18}$$

$$= \sum_{i=1}^{k} \sigma_i^r \tag{2.19}$$

Furthermore, they proved that the approximation error of the low-rank Koopman operator can be decomposed into an unknown constant part dependent on the true Koopman operator $\mathcal{K}_\tau$ and a model dependent part $\mathcal{R}_E$, which is called VAMP-E score:

$$||\hat{\mathcal{K}}_\tau - \mathcal{K}_\tau||_{\text{HS}}^2 = -\mathcal{R}_E[\mathbf{K},\mathbf{f},\mathbf{g}] + ||\mathcal{K}_\tau||_{\text{HS}}^2 \tag{2.20}$$

$$\text{with } \mathcal{R}_E[\mathbf{K},\mathbf{f},\mathbf{g}] = 2\sum_{i}^{k} K_{ii} \langle f_i, \mathcal{K}_\tau g_i \rangle_{\rho_0} - \sum_{i,j}^{k} K_{ii} K_{jj} \langle f_i, f_j \rangle_{\rho_0} \langle g_i, g_j \rangle_{\rho_1}. \tag{2.21}$$

2.1.3. Estimation algorithms

Given T many observations $\{\mathbf{x}_t, \mathbf{x}_{t+\tau}\}_{t=1,..,T}$ and feature functions $\boldsymbol{\chi}_0(\mathbf{x}_t)$ and $\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})$ we seek to find optimal singular functions $\mathbf{f}(\mathbf{x}_t)$ and $\mathbf{g}(\mathbf{x}_{t+\tau})$ which fulfill the orthonormality constraint. We make the linear ansatz:

$$\mathbf{f}(\mathbf{x}_t) = \mathbf{U}^T \boldsymbol{\chi}_0(\mathbf{x}_t) \tag{2.22}$$

$$\mathbf{g}(\mathbf{x}_{t+\tau}) = \mathbf{V}^T \boldsymbol{\chi}_1(\mathbf{x}_{t+\tau}). \tag{2.23}$$

Therefore, the maximization problem Eq. 2.14 translates to:

$$\max_{\mathbf{U},\mathbf{V}} \sum_{i=1}^{k} (\mathbf{u}_i^T \mathbf{C}_{01} \mathbf{v}_i)^r \tag{2.24}$$

$$\text{s.t. } \mathbf{U}^T \mathbf{C}_{00} \mathbf{U} = \mathbf{I}, \tag{2.25}$$

$$\mathbf{V}^T \mathbf{C}_{11} \mathbf{V} = \mathbf{I}, \tag{2.26}$$

where the covariance matrices are introduced as:

$$\mathbf{C}_{00} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\chi}_0(\mathbf{x}_t) \boldsymbol{\chi}_0^T(\mathbf{x}_t) \tag{2.27}$$

$$\mathbf{C}_{11} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\chi}_1(\mathbf{x}_{t+\tau}) \boldsymbol{\chi}_1^T(\mathbf{x}_{t+\tau}) \tag{2.28}$$

$$\mathbf{C}_{01} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\chi}_0(\mathbf{x}_t) \boldsymbol{\chi}_1^T(\mathbf{x}_{t+\tau}). \tag{2.29}$$

An existing algorithm which extracts the optimal choices for $\mathbf{U}$ and $\mathbf{V}$ is the canonical correlation analysis (CCA), where the SVD of the Koopman matrix $\bar{\mathbf{K}}$ for the normalized feature functions $\mathbf{C}_{00}^{-1/2} \boldsymbol{\chi}_0(\mathbf{x}_t)$ and $\mathbf{C}_{11}^{-1/2} \boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})$ is applied:

$$\bar{\mathbf{K}} = \mathbf{C}_{00}^{-\frac{1}{2}} \mathbf{C}_{01} \mathbf{C}_{11}^{-\frac{1}{2}} = \bar{\mathbf{U}} \mathbf{K} \bar{\mathbf{V}}^T. \tag{2.30}$$

The matrix $\mathbf{K} = \text{diag}(K_{11}, ..., K_{kk})$ is then the approximation of the true singular values $\sigma_1, ..., \sigma_k$. The coefficients to construct the corresponding singular functions are given by:

$$\mathbf{U} = \mathbf{C}_{00}^{-\frac{1}{2}} \bar{\mathbf{U}}, \tag{2.31}$$

$$\mathbf{V} = \mathbf{C}_{11}^{-\frac{1}{2}} \bar{\mathbf{V}}. \tag{2.32}$$

Following, the Koopman matrix for the feature space can be estimated when plugging the result into Eq.2.13:

$$\begin{aligned}\mathbf{K}_\chi &= \mathbf{U}\mathbf{K}\mathbf{V}^{-1} && (2.33)\\ &= \mathbf{C}_{00}^{-\frac{1}{2}}\bar{\mathbf{U}}\mathbf{K}\bar{\mathbf{V}}^T\mathbf{C}_{11}^{+\frac{1}{2}} && (2.34)\\ &= \mathbf{C}_{00}^{-\frac{1}{2}}\bar{\mathbf{K}}\mathbf{C}_{11}^{+\frac{1}{2}} && (2.35)\\ &= \mathbf{C}_{00}^{-\frac{1}{2}}\mathbf{C}_{00}^{-\frac{1}{2}}\mathbf{C}_{01}\mathbf{C}_{11}^{-\frac{1}{2}}\mathbf{C}_{11}^{+\frac{1}{2}} && (2.36)\\ &= \mathbf{C}_{00}^{-1}\mathbf{C}_{01}. && (2.37)\end{aligned}$$

2.1.4. Validation

To evaluate the performance of a trained model, we test its capabilities of predicting the dynamics of hold-out data to find the right balance between statistical bias and variance. Therefore, the available data is usually split into train and test set and the model estimated on the training data is scored on the latter.
Wu et Noé transferred the idea of McGibbon et Pande to utilize the subspace variational score [54]. There, the consistency between the singular subspaces of training and test set are measured without the constraint of orthonormality and the score reads:

$$\mathcal{R}_r^{\text{subspace}} = ||(\mathbf{U}^T\mathbf{C}_{00}^{\text{test}}\mathbf{U})^{-\frac{1}{2}}(\mathbf{U}^T\mathbf{C}_{01}^{\text{test}}\mathbf{V})(\mathbf{V}^T\mathbf{C}_{11}^{\text{test}}\mathbf{V})^{-\frac{1}{2}}||_r^r, \tag{2.38}$$

where $\mathbf{U}$ and $\mathbf{V}$ are estimated on the training set and the correlation matrices on the test set.
Since the inversion might cause numerical instabilities they propose to utilize the VAMP-E score instead:

$$\mathcal{R}_E^{\text{subspace}} = \text{tr}[2\mathbf{K}\mathbf{U}^T\mathbf{C}_{01}^{\text{test}}\mathbf{V} - \mathbf{K}\mathbf{U}^T\mathbf{C}_{00}^{\text{test}}\mathbf{U}\mathbf{K}\mathbf{V}^T\mathbf{C}_{11}^{\text{test}}\mathbf{V}]. \tag{2.39}$$

2.1.5. Connection of transition density and VAMP-E score

The transition density can be estimated from the real Koopman operator as

$$\begin{aligned}p_\tau(\mathbf{y}|\mathbf{x}) &= \mathcal{K}_\tau\delta_{\mathbf{y}}(\mathbf{x}) && (2.40)\\ &= \sum_{i=1}^{\infty}\sigma_i\psi_i(\mathbf{x})\phi_i(\mathbf{y})\rho_1(\mathbf{y}) && (2.41)\end{aligned}$$

An approximation of the transition density $\hat{p}_\tau(\mathbf{y}|\mathbf{x})$ is given via the finite dimensional Koopman operator, where we can insert our approximations of the singular functions:

$$\hat{p}_\tau(\mathbf{y}|\mathbf{x}) = \boldsymbol{\chi}_0^T(\mathbf{x})\mathbf{U}\mathbf{K}\mathbf{V}^T\boldsymbol{\chi}_1(\mathbf{y})\rho_1(\mathbf{y}) \tag{2.42}$$

$$= \boldsymbol{\chi}_0^T(\mathbf{x})\mathbf{S}\boldsymbol{\chi}_1(\mathbf{y})\rho_1(\mathbf{y}), \tag{2.43}$$

where we defined the matrix $\mathbf{S} = \mathbf{U}\mathbf{K}\mathbf{V}^T$. For interpretation reasons it is beneficial to further write out the definition:

$$\mathbf{S} = \mathbf{U}\mathbf{K}\mathbf{V}^T \tag{2.44}$$

$$= \mathbf{C}_{00}^{-\frac{1}{2}}\bar{\mathbf{U}}\mathbf{K}\bar{\mathbf{V}}^T\mathbf{C}_{11}^{-\frac{1}{2}} \tag{2.45}$$

$$= \mathbf{C}_{00}^{-\frac{1}{2}}\mathbf{C}_{00}^{-\frac{1}{2}}\mathbf{C}_{01}\mathbf{C}_{11}^{-\frac{1}{2}}\mathbf{C}_{11}^{-\frac{1}{2}} \tag{2.46}$$

$$= \mathbf{C}_{00}^{-1}\mathbf{C}_{01}\mathbf{C}_{11}^{-1} \tag{2.47}$$

$$= \mathbf{K}_\chi\mathbf{C}_{11}^{-1}. \tag{2.48}$$

The equation shows that the matrix $\mathbf{S}$ is a combination of the Koopman matrix $\mathbf{K}_\chi$ making the transition from $\boldsymbol{\chi}_0(\mathbf{x}_t)$ to $\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})$ during a time step τ and the covariance matrix $\mathbf{C}_{11}$ which normalizes the transition density $\int \hat{p}_\tau(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{y} = 1$. Therefore, the transition density Eq. 2.42 can be interpreted as mapping to state space $\boldsymbol{\chi}_0(\mathbf{x})$, a time transition to the feature space of $\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau}) = \mathbf{K}_\chi^T\boldsymbol{\chi}_0(\mathbf{x})$, and a back mapping to configuration $\mathbf{y}$ depending on the similarity of the feature vector $\boldsymbol{\chi}_1(\mathbf{y})$ and the propagated one $\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})$.
The Koopman matrix $\mathbf{K}_\chi$ is a mapping from feature space $\boldsymbol{\chi}_0$ to $\boldsymbol{\chi}_1$ during a time step τ. However, sometimes a propagation or transition matrix within the feature space of $\boldsymbol{\chi}_0$ during a time step τ is favorable:

$$\mathbb{E}[\boldsymbol{\chi}_0(\mathbf{x}_{t+\tau})] = \mathbf{K}_{\chi_1\chi_0}^T\mathbb{E}[\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})] \tag{2.49}$$

$$= \mathbf{K}_{\chi_1\chi_0}^T\mathbf{K}_\chi^T\mathbb{E}[\boldsymbol{\chi}_0(\mathbf{x}_t)] \tag{2.50}$$

$$= \mathbf{T}^T\mathbb{E}[\boldsymbol{\chi}_0(\mathbf{x}_t)], \tag{2.51}$$

where with the definition of $\mathbf{\Sigma} = \mathbb{E}[\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau})\boldsymbol{\chi}_0(\mathbf{x}_{t+\tau})]$ the transition matrix can be evaluated as:

$$\begin{aligned}\mathbf{T} &= \mathbf{K}_\chi \mathbf{K}_{\chi_1\chi_0} && (2.52)\\ &= \mathbf{K}_\chi \mathbf{C}_{11}^{-1}\mathbf{\Sigma} && (2.53)\\ &= \mathbf{S}\mathbf{\Sigma}. && (2.54)\end{aligned}$$

Finally, the VAMP-E score can be expressed via the matrix $\mathbf{S} = \mathbf{U}\mathbf{K}\mathbf{V}^T$ by plugging it into Eq. 2.39:

$$\mathcal{R}_E = \mathrm{tr}[2\mathbf{S}^T\mathbf{C}_{01} - \mathbf{S}^T\mathbf{C}_{00}\mathbf{S}\mathbf{C}_{11}]. \tag{2.55}$$

2.1.6. Reversible Koopman operators

In practice, the system of interest is often studied in thermal equilibrium, i.e. for a simulation no external forces are present. As a consequence the system is purely driven by thermal energy allowing no work extraction. Therefore, the existence of a cycle of probability flux is forbidden. With the equilibrium distribution $\mu(\mathbf{x})$, this manifests in the the following equation also called detailed balance:

$$\mu(\mathbf{x})p_\tau(\mathbf{y}|\mathbf{x}) = \mu(\mathbf{y})p_\tau(\mathbf{x}|\mathbf{y}). \tag{2.56}$$

The probability to observe the transition from configuration $\mathbf{x}$ to $\mathbf{y}$ is equal to observe the reverse transition. Here, the probability is given by the product of the transition density and the equilibrium distribution.
In this case, the Koopman operator becomes self-adjoint, and the singular value decomposition transfers to an eigendecomposition:

$$\hat{p}_\tau(\mathbf{y}|\mathbf{x}) = \boldsymbol{\varphi}^T(\mathbf{x})\mathbf{\Lambda}\boldsymbol{\varphi}(\mathbf{y})\mu(\mathbf{y}), \tag{2.57}$$

with $\varphi_i(\mathbf{x}) = \psi_i(\mathbf{x}) = \phi_i(\mathbf{x})$ and the diagonal matrix $\Lambda_{ii} = \sigma_i$ if the data present is in equilibrium [55].
However, usually the sampling is not sufficient to achieve equilibrium data. Therefore, a reweighting scheme can be employed to reweight the empirical to the stationary distribution $\mu(\mathbf{y}) = w(\mathbf{y})\rho_1(\mathbf{y})$.
This strategy can be adopted to the choice of the feature functions. If we choose

$\boldsymbol{\chi}_0(\mathbf{x}) = \boldsymbol{\chi}(\mathbf{x})$ and $\boldsymbol{\chi}_1(\mathbf{y}) = \boldsymbol{\chi}(\mathbf{y})w(\mathbf{y})$, Eq.2.42 translates to:

$$\hat{p}_\tau(\mathbf{y}|\mathbf{x}) = \boldsymbol{\chi}^T(\mathbf{x})\mathbf{U}\mathbf{K}\mathbf{V}^T\boldsymbol{\chi}(\mathbf{y})w(\mathbf{y})\rho_1(\mathbf{y}) \tag{2.58}$$

$$= \boldsymbol{\chi}^T(\mathbf{x})\mathbf{R}\boldsymbol{\Lambda}\mathbf{R}^T\boldsymbol{\chi}(\mathbf{y})\mu(\mathbf{y}), \tag{2.59}$$

where $\mathbf{R} = \mathbf{U} = \mathbf{V}$ and the eigenfunctions are $\boldsymbol{\varphi}(\mathbf{x}) = \mathbf{R}^T\boldsymbol{\chi}(\mathbf{x})$. Thereby, the matrix $\mathbf{S}$:

$$\mathbf{S} = \mathbf{R}\boldsymbol{\Lambda}\mathbf{R}^T = \mathbf{S}^T \tag{2.60}$$

becomes symmetric and since $\boldsymbol{\Sigma}$ is likewise symmetric - in fact it is the covariance matrix in equilibrium - it follows for the transition matrix:

$$\mathbf{T} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\bar{\mathbf{S}}\boldsymbol{\Sigma}^{\frac{1}{2}} \tag{2.61}$$

$$= \boldsymbol{\Sigma}^{-\frac{1}{2}}(\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{S}\boldsymbol{\Sigma}^{\frac{1}{2}})\boldsymbol{\Sigma}^{\frac{1}{2}}, \tag{2.62}$$

that it is similar to the Hermitian matrix $\bar{\mathbf{S}}$ and has therefore real eigenvalues.
Plugging the expressions for the stationary distribution and transition density into Eq. 2.56 confirms the reversibility:

$$\mu(\mathbf{x})\hat{p}_\tau(\mathbf{y}|\mathbf{x}) = \mu(\mathbf{x})\boldsymbol{\chi}^T(\mathbf{x})\mathbf{S}\boldsymbol{\chi}(\mathbf{y})\mu(\mathbf{y}) \tag{2.63}$$

$$= \mu(\mathbf{x})\boldsymbol{\chi}^T(\mathbf{y})\mathbf{S}^T\boldsymbol{\chi}(\mathbf{x})\mu(\mathbf{y}) \tag{2.64}$$

$$= \mu(\mathbf{y})\boldsymbol{\chi}^T(\mathbf{y})\mathbf{S}\boldsymbol{\chi}(\mathbf{x})\mu(\mathbf{x}) \tag{2.65}$$

$$= \mu(\mathbf{y})\hat{p}_\tau(\mathbf{x}|\mathbf{y}). \tag{2.66}$$

2.1.7. Connection Markov state model and Koopman model

For the specific choice of indicator functions as feature functions, the Koopman model becomes a traditional Markov state model. These indicator functions partition the configuration space into disjoint subspaces A_i:

$$\chi_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathrm{A_i} \\ 0 & \text{otherwise.} \end{cases} \tag{2.67}$$

In that particular case the covariance matrices $\mathbf{C}_{00}$, $\mathbf{C}_{11}$ become diagonal count matrices how often each state was visited and $\mathbf{C}_{01}$ counts the transitions between

the states for all data pairs. As a consequence the Koopman matrix $\mathbf{K}_\chi$ has non-negative entries and the rows sum up to one $\mathbf{K}_\chi \mathbf{1} = \mathbf{1}$ and is therefore a transition matrix.
In the special case of a reversible Koopman model with indicator functions, the matrix $\mathbf{S} = \mathbf{T}\mathbf{\Sigma}^{-1}$, where the equilibrium covariance matrix $\mathbf{\Sigma} = \mathbf{\Pi} = \mathrm{diag}(\boldsymbol{\pi})$ is a diagonal matrix with the stationary distribution vector on the diagonal. If we choose the weights to be $w(\mathbf{x}) = \boldsymbol{\pi}^T \mathbf{\Pi}_\rho^{-1} \boldsymbol{\chi}(\mathbf{x})$ with $\mathbf{\Pi}_\rho = \mathrm{diag}(\boldsymbol{\pi}_\rho)$ being the empirical distribution vector on the diagonal, the stationary distribution and transition density are given by their expected expressions:

$$\mu(\mathbf{x}) = \boldsymbol{\pi}^T \mathbf{\Pi}_\rho^{-1} \boldsymbol{\chi}(\mathbf{x}) \rho(\mathbf{x}) \tag{2.68}$$

$$\hat{p}_\tau(\mathbf{y}|\mathbf{x}) = \boldsymbol{\chi}(\mathbf{x})^T \mathbf{T} \mathbf{\Pi}^{-1} \boldsymbol{\chi}(\mathbf{y}) \rho(\mathbf{y}) \boldsymbol{\chi}^T(\mathbf{y}) \mathbf{\Pi}_\rho^{-1} \boldsymbol{\pi} \tag{2.69}$$

$$= \boldsymbol{\chi}(\mathbf{x})^T \mathbf{T} \mathbf{\Pi}_\rho^{-1} \boldsymbol{\chi}(\mathbf{y}) \rho(\mathbf{y}) \boldsymbol{\chi}^T(\mathbf{y}) \mathbf{\Pi}^{-1} \boldsymbol{\pi} \tag{2.70}$$

$$= \boldsymbol{\chi}(\mathbf{x})^T \mathbf{T} \mathbf{\Pi}_\rho^{-1} \boldsymbol{\chi}(\mathbf{y}) \rho(\mathbf{y}). \tag{2.71}$$

Since $\mathbf{S}$ is symmetric it follows that:

$$\mathbf{S} = \mathbf{T}\mathbf{\Pi}^{-1} = \mathbf{\Pi}^{-1}\mathbf{T}^T \tag{2.72}$$

$$\iff \mathbf{\Pi}\mathbf{T} = \mathbf{T}^T\mathbf{\Pi} = (\mathbf{\Pi}\mathbf{T})^T, \tag{2.73}$$

which is the common condition for a reversible Markov state model.

2.1.8. Decompostion into independent subsystems

When studying larger molecules with Koopman theory a fundamental problem arises. The larger system might be constituted of weakly coupled or independent subsystems [50, 51]. To describe the full system each combination of processes in the subsystems has to be described and for model estimation observed. However, with the number of subsystems the number of combinations increases exponentially leading to an unfeasible demand of data. Therefore, it is of great interest to construct the global model as a combination of independent subsystems.
Given two independent subsystems constituting a global system described by their transition densities $p_\tau^1(\mathbf{y}^1|\mathbf{x}^1)$ and $p_\tau^2(\mathbf{y}^2|\mathbf{x}^2)$, the global transition density is:

$$p_\tau^G(\mathbf{y}^1, \mathbf{y}^2|\mathbf{x}^1, \mathbf{x}^2) = p_\tau^1(\mathbf{y}^1|\mathbf{x}^1) \cdot p_\tau^2(\mathbf{y}^2|\mathbf{x}^2). \tag{2.74}$$

Each subsystem can be described by a Koopman operator propagating observables $\chi^1(\mathbf{x}^1)$ and $\chi^2(\mathbf{x}^2)$. The global Koopman operator which describes the propagation of the product of the individual observables $\chi^G(\mathbf{x}^1, \mathbf{x}^2) = \chi^1(\mathbf{x}^1)\chi^2(\mathbf{x}^2)$ decomposes into the individual operators:

$$\mathcal{K}_\tau^G \chi^G(\mathbf{x}^1, \mathbf{x}^2) = \iint p_\tau^G(\mathbf{y}^1, \mathbf{y}^2 | \mathbf{x}^1, \mathbf{x}^2) \chi^G(\mathbf{x}^1, \mathbf{x}^2) \mathrm{d}\mathbf{y}^1 \mathrm{d}\mathbf{y}^2 \tag{2.75}$$

$$= \int p_\tau^1(\mathbf{y}^1 | \mathbf{x}^1) \chi^1(\mathbf{x}^1) \mathrm{d}\mathbf{y}^1 \int p_\tau^2(\mathbf{y}^2 | \mathbf{x}^2) \chi^2(\mathbf{x}^2) \mathrm{d}\mathbf{y}^2 \tag{2.76}$$

$$= \mathcal{K}_\tau^1 \chi^1(\mathbf{x}^1)\, \mathcal{K}_\tau^2 \chi^2(\mathbf{x}^2) \tag{2.77}$$

If we have a finite rank approximation of each operator with k^1, k^2 singular functions the constructed global operator reads:

$$\hat{\mathcal{K}}_\tau^G \chi^G(\mathbf{x}^1, \mathbf{x}^2) = \hat{\mathcal{K}}_\tau^1 \chi(\mathbf{x}^1)\, \hat{\mathcal{K}}_\tau^2 \chi(\mathbf{x}^2) \tag{2.78}$$

$$= \sum_{i=1}^{k^1} \sigma_i^1 \langle \chi^1, \phi_i^1 \rangle_{\rho_1^1} \psi_i^1(\mathbf{x}^1) \sum_{j=1}^{k^2} \sigma_j^2 \langle \chi^2, \phi_j^2 \rangle_{\rho_1^2} \psi_j^2(\mathbf{x}^2) \tag{2.79}$$

$$= \sum_{i=1}^{k^1} \sum_{j=1}^{k^2} \sigma_i^1 \sigma_j^2 \langle \chi^1 \chi^2, \phi_i^1 \phi_j^2 \rangle_{\rho_1^G} \psi_i^1(\mathbf{x}^1) \psi_j^2(\mathbf{x}^2) \tag{2.80}$$

$$= \sum_{l=1}^{k^1 k^2} \sigma_l^G \langle \chi^G, \phi_l^G \rangle_{\rho_1^G} \psi_l^G(\mathbf{x}^1, \mathbf{x}^2). \tag{2.81}$$

This shows that the global singular functions and values are simply the product of the individual subsystems $\psi_l^G = \psi_i^1 \psi_j^2$, $\sigma_l^G = \sigma_i^1 \sigma_j^2$, and $\phi_l^G = \phi_i^1 \phi_j^2$.
This result can be expanded to any number of subsystems, where the global model can be always constructed as the product of all subsystems.

2.2. Artificial neural networks

This section is meant to introduce the most important concepts in training Artificial neural networks (NN) - also called deep learning -, which are useful to understand the later application of approximating the Koopman operator. For a thorough introduction to deep learning I refer the reader to [27].
NNs are a collection of nodes connected by edges which loosely speaking resemble the neurons of biological brains. Usually, a node estimates a weighted sum of all the real valued inputs coming from the nodes connected via the edges. It further

applies some non-linear function to the output, before sending the messages to the other nodes. However, the intermediate steps and the edges depend on the chosen architecture. The architecture includes trainable parameters, e.g. the weights in the sum, which will be adopted during the so called training routine.
The architecture allows to feed in an input vector, e.g. the input coordinates of the atoms of a molecule, and outputs a transformed vector of possible different size. Therefore, the NN can be understood as a highly non-linear function.
In order to train the NN to approximate a sought function, e.g. mapping the coordinates at a given time to a specific metastable state, an objective function has to be defined. The goal of training the NN is to update the parameters of the architecture to optimize the objective function. It is usually framed as a loss function which should be minimized.
If the NN represents a differentiable function, the parameters can be updated by the stochastic gradient descent algorithm. There, a subset of training inputs, called batch, is chosen and transformed by the NN. The gradient of the objective function with respect to the parameters are estimated via the backpropagation algorithm [56]. These are used to update the parameters with a so called optimizer in order to increase the quality of the function. By only using a subset of the training points for an update, noise is introduced compared to the gradient of the whole dataset. This is assumed to provide two advantages. Firstly, by only passing a smaller amount of data through the network the hardware requirements decrease making it possible to work on very large data sets.
Secondly, it avoids the model being trapped in local minima during the training. A standard gradient descent algorithm guarantees solely the convergence to a local minimum. By estimating the gradients on different subsets of the training data, it might be that the actual position in the parameter space is a local minimum for the whole dataset but not for the current subset. Thereby, this particular update allows the NN to escape the local minimum.
However, the second advantage is still actively discussed by the community, e.g. others argue that these local minima simply don't exist, because of the vast parameter space [57, 58].
Although the first formulation of NN dates back several decades, the recent successes are mainly fueled by four different developments: greater computational power, new non-linear functions, which exhibit better training behaviour [59, 60], more advanced architectures, which usually exploit some structure in the data known beforehand

[28, 61–63], and more advanced optimizers [64, 65], which converge faster to the sought function.

2.2.1. Feed forward neural network

The former mentioned backpropagation algorithm imposes conditions on the architecture of the NN. Namely, it must be possible to unroll the architecture to a state where the information flows from the input always forward to the output, called feed forward NN.
The most simple architecture is the so called fully connected feed forward NN, where a collection of nodes, called layer, is connected to all the nodes of the next layer. The layers between the input and output layer are called hidden layers.
The simple architecture allowed the formulation of the universal approximation theorem, stating that in principle a one hidden layer NN with arbitrary width and the *sigmoid* activation function is sufficient to approximate any non-linear continuous function to arbitrary precision [30].

2.2.2. Invariances and transferable architectures

The inclusion of structure from the underlying data into the model architecture can significantly improve the performance of the NN when confronted with limited data. In the case of a NN classifying protein frames into metastable states, we can exploit symmetries. Since the classification should be invariant under the rotation and translation of the whole protein, the output of the NN should be unaffected under these transformations. A simple solution is to not feed the coordinates of all atoms into the NN but instead let it act solely on internal coordinates, e.g. distances or angles. In order to counteract the increase in input features, a cutoff distance can be introduced or all atoms of a residue can be summarized by a common feature.
Furthermore, information of the same atom or residue type can be fed into the architecture by sharing the same feature representation.
A nowadays common approach is to represent the molecule or protein as a graph, where the information flows along the edges between atoms or residues [62]. The NN represents the function which collects the messages coming from the other edges and updates the node accordingly. The architecture of graph NN allows to share parameters between the participating nodes, i.e. using the same NN for all nodes. This does not only reduce the amount of necessary parameters, when applying it

to larger molecules. It also enables the transfer of knowledge within the chemical space when applying it to unseen molecules [66–69].

2.2.3. NN as transformers between probability distributions

Neural networks are not restricted to map given data to some target value. Instead, they can be trained to map from easily sampled distributions, e.g. Gaussian distributions, to the complex distributions of some data. Thereby, the networks can be employed to generate new unobserved data points. These generative neural networks were developed in different flavors [70–73]. They can be even advanced to sampling from the Boltzmann distribution without the need for training samples [74]. These models differ by their architecture or how they define the loss function. One way to design a loss function is to define a metric between probability distributions.
The standard Energy Distance (ED) is a metric between two distributions of random vectors $\mathbf{X}$ and $\mathbf{Y}$, defined as in [75]:

$$D_E(\mathbb{P}(\mathbf{X}), \mathbb{P}(\mathbf{Y})) = \mathbb{E}[2||\mathbf{x} - \mathbf{y}|| - ||\mathbf{x} - \mathbf{x}'|| - ||\mathbf{y} - \mathbf{y}'||]. \tag{2.82}$$

The outcomes $\mathbf{x}, \mathbf{x}'$ and $\mathbf{y}, \mathbf{y}'$ are independently distributed according to the distributions of $\mathbf{X}$ and $\mathbf{Y}$, respectively. This means, that the ED is only zero, if the two distributions match. Thus, given a generative network $\mathbf{G}$, which maps from Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ to test samples $\mathbf{y} = \mathbf{G}(\boldsymbol{\epsilon})$, it can be trained to minimize the ED to some data distribution $\mathbf{X}$. If the ED approaches zero, the samples drawn by the network $\mathbf{G}$ should be close to samples drawn from the data distribution $\mathbf{X}$ w.r.t. the norm $||\cdot||$.
The metric can be expanded to conditional probability distributions [76]:

$$D \triangleq \mathbb{E}[D_E(\mathbb{P}(\mathbf{X}_{t+\tau}|\mathbf{x}_t), \mathbb{P}(\hat{\mathbf{X}}_{t+\tau}|\mathbf{x}_t)|\mathbf{x}_t] \tag{2.83}$$

$$= \mathbb{E}[2||\hat{\mathbf{x}}_{t+\tau} - \mathbf{x}_{t+\tau}|| - ||\hat{x}_{t+\tau} - \hat{\mathbf{x}}'_{t+\tau}|| - ||\mathbf{x}_{t+\tau} - \mathbf{x}'_{t+\tau}||], \tag{2.84}$$

where the outcomes $\mathbf{x}_{t+\tau}$ and $\mathbf{x}'_{t+\tau}$ are distributed according to the transition density given a starting configuration $\mathbf{x}_t$. The outcomes $\hat{\mathbf{x}}_{t+\tau}, \hat{\mathbf{x}}'_{t+\tau}$ are independent predictions of the generative model, which are conditioned on the same configuration $\mathbf{x}_t$.

3. Methods

This chapter contains the methods, which combine the theory of Koopman operators with deep learning. In general, all methods will include a mapping from configurations $\mathbf{x}_t$ to a state assignment, which we will call $\boldsymbol{\chi}(\mathbf{x}_t)$. This mapping is either a composition of several transformations including a neural network or simply given by one called $\boldsymbol{\eta}$.

The chapter starts with the first developed method called VAMPnets, which estimates the most general Koopman model without any further restrains on reversibility or the transition matrix [77]. To enforce a stochastic transition matrix and enable the back mapping from state to configuration space, we advanced VAMPnets by introducing a second NN and call it deep MSM. The work further included the possibility to generate new configurations by a generative NN [76]. Since the reversibility was still unaddressed, our next project focused on deep reversible Koopman models, where additional trainable parameters model the stationary distribution and transition matrix explicitly. Thereby, two constraints can be formulated which can ensure reversibility or a stochastic transition matrix. In principle, by individually turning on/off the constraints the method allows to build four different classes of models. However, since two of them were already addressed by VAMPnets and deep MSMs, we focused on reversible VAMPnets and reversible deep MSMs [78].

To further close the gap between ordinary MSMs and their deep counterpart, we developed extensions to include experimental evidence in the model estimation by updating the loss function and to coarse-grain the state space by a specialized network layer [79].

The final method I have worked on is iVAMPnets, where we try to address the unscalability of global MSMs to larger proteins due to the exponential growth of states. The model combines a separation of the input features into local sub-segments and subsequent parallel VAMPnets, which estimate independent kinetics on these. The work is as up to today in the revision process.

3.1. VAMPnets for deep learning of molecular kinetics

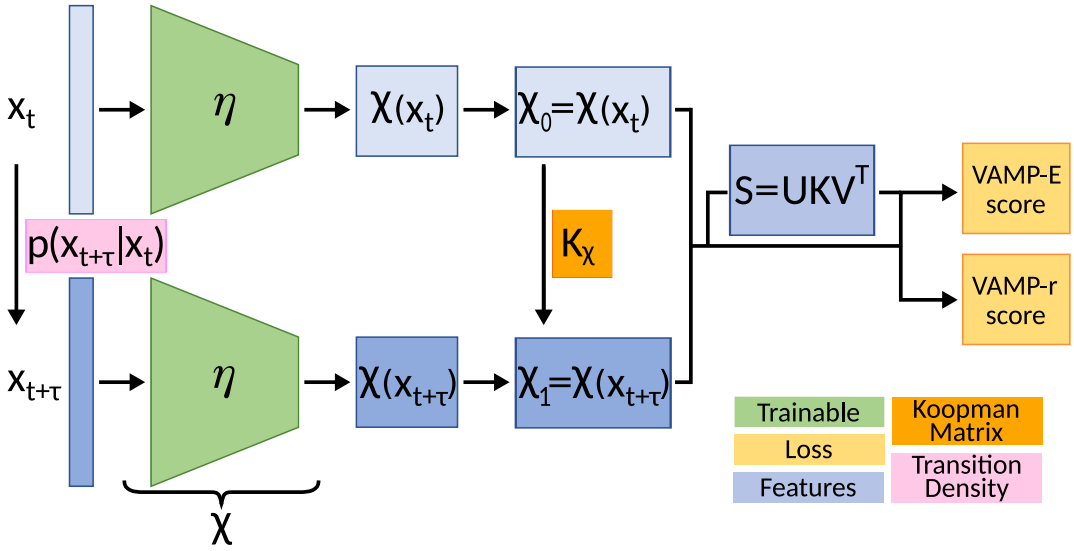

Figure 3.1.: Scheme of the VAMPnet learning architecture. The input pair $(\mathbf{x}_t, \mathbf{x}_{t+\tau})$ is transformed by the same NN $\boldsymbol{\chi}(\mathbf{x}_t) = \boldsymbol{\eta}(\mathbf{x}_t)$ and $\boldsymbol{\chi}(\mathbf{x}_{t+\tau}) = \boldsymbol{\eta}(\mathbf{x}_{t+\tau})$. The Koopman matrix $\mathbf{K}_\chi$ from Eq. 2.13 propagates the state vector $\boldsymbol{\chi}_0(\mathbf{x}_t) = \boldsymbol{\chi}(\mathbf{x}_t)$ in time to approximate $\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau}) = \boldsymbol{\chi}(\mathbf{x}_{t+\tau})$. The NN can be trained by maximizing either the VAMP-r or VAMP-E score Eq. 2.14, 2.39. The later makes use of the definition of the matrix $\mathbf{S}$, which in the case of the VAMPnet is simply estimated from data as shown.

VAMPnets are a deep learning framework which aims to build Koopman models of molecular simulation data [77]. We represent the feature functions by a NN $\boldsymbol{\eta}$ mapping from the configuration space $\mathbf{x}_t \in \mathbb{R}^d$ into the feature space $\boldsymbol{\chi}(\mathbf{x}_t) = \boldsymbol{\eta}(\mathbf{x}_t) \in \mathbb{R}^k$. Since the VAMPnet is expected to find metastable states in biomolecules, we make two important choices:

1. The feature functions use the *softmax* function as an output activation. Thereby, each configuration $\mathbf{x}_t$ will be classified in which metastable state the system is in.

2. The feature functions are identical $\boldsymbol{\chi}_0(\mathbf{x}) = \boldsymbol{\chi}_1(\mathbf{x}) = \boldsymbol{\chi}(\mathbf{x})$ $\forall \mathbf{x}$ assuming that the system is visiting the same metastable states for data at time t and $t + \tau$.

VAMPnets is not restricted to model molecular kinetics. However, these choices make it especially fitting for molecular simulations, where similar metastable states

are expected in the empirically distributions ρ_0 and ρ_1. The parameters of the NN can be trained by maximizing either the VAMP-r or VAMP-E score.
In summary, a VAMPnets makes the specific choices for the general Koopman model (cf. Fig. 3.1):

1. $\boldsymbol{\chi}_0(\mathbf{x}_t) = \boldsymbol{\chi}(\mathbf{x}_t)$.
2. $\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau}) = \boldsymbol{\chi}(\mathbf{x}_{t+\tau})$.
3. $\mathbf{S} = \mathbf{U}\mathbf{K}\mathbf{V}^T$.

The method has shortcomings which will be addressed by methods introduced below. These include:

1. The Koopman matrix $\mathbf{K}_\chi$, which propagates the metastable states in time is not a stochastic transition matrix. Although the rows are normalized due to the normalized state assignments $\mathbf{K}_\chi \mathbf{1} = \mathbf{1}$, individual entries might be negative. Therefore, they cannot be interpreted as probabilities and further analysis tools such as transition path theory (TPT) cannot be applied.
2. Once mapped on the feature space $\boldsymbol{\chi}(\mathbf{x})$, it is not possible to map a state back to a configuration $\mathbf{x}$.
3. The model is not guaranteed to be reversible.

3.2. Deep Markov state model

With the deep Markov state model (deep MSM) we model the transition density explicitly in a two step process:

$$\hat{p}_\tau(\mathbf{y}|\mathbf{x}) = \boldsymbol{\chi}(\mathbf{x})^T \mathbf{q}(\mathbf{y};\tau) = \sum_{i=1}^{k} \chi_i(\mathbf{x}) q_i(\mathbf{y};\tau). \tag{3.1}$$

Like in VAMPnets, the function $\chi_i(\mathbf{x})$ models the probability how likely a configuration belongs to the state i. Additionally, the function $q_i(\mathbf{y};\tau)$ models the probability how likely the system jumps to configuration $\mathbf{y}$ given that it was in state i a time interval τ before.
The transition matrix propagating $\boldsymbol{\chi}(\mathbf{x})$ τ in time can be estimated via:

$$T_{ij}(\tau) = \int_{\mathbf{y}} q_i(\mathbf{y};\tau)\chi_j(\mathbf{y})\mathrm{d}\mathbf{y}, \tag{3.2}$$

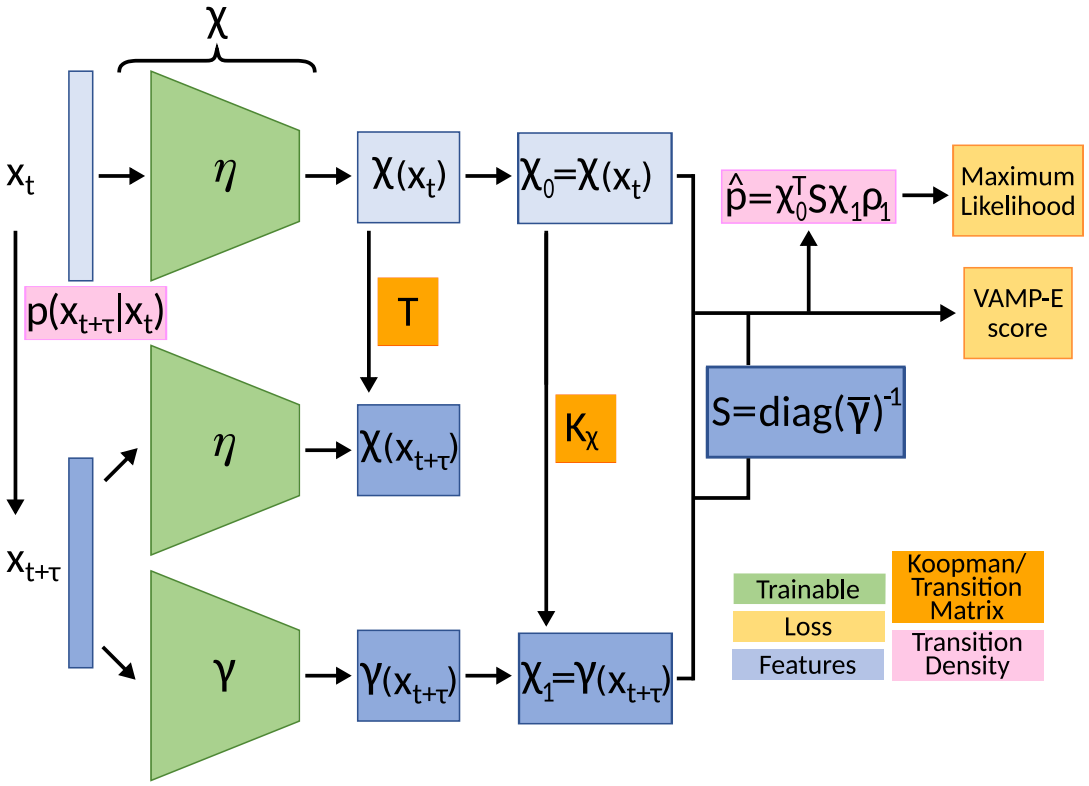

Figure 3.2.: Scheme of the deep MSM learning architecture. The data pair $(\mathbf{x}_t, \mathbf{x}_{t+\tau})$ is transformed by two different NN $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$. The Koopman model is then defined between $\boldsymbol{\chi}_0(\mathbf{x}_t) = \boldsymbol{\chi}(\mathbf{x}_t) = \boldsymbol{\eta}(\mathbf{x}_t)$ and $\boldsymbol{\chi}_1(\mathbf{x}_{t+\tau}) = \boldsymbol{\gamma}(\mathbf{x}_{t+\tau})$. Consecutively, the Koopman matrix $\mathbf{K}_\chi$ maps between them and the model can be trained with the VAMP-E score. The transition matrix $\mathbf{T}$, which propagates the state vector $\boldsymbol{\chi}(\mathbf{x}_t)$ in time, can be estimated by Eq. 3.2. The matrix $\mathbf{S}$ is chosen to normalize the probability $\mathbf{q}(\mathbf{x}_{t+\tau}) = \mathbf{S}\boldsymbol{\gamma}(\mathbf{x}_{t+\tau})\rho_1(\mathbf{x}_{t+\tau})$. $q_i(\mathbf{x}_{t+\tau})$ approximates the probability of how likely the system ends up in the configuration $\mathbf{x}_{t+\tau}$ at time $t+\tau$, given that it was at state i at time t. Given $\mathbf{q}(\mathbf{x}_{t+\tau})$ and $\boldsymbol{\chi}(\mathbf{x}_t)$ an estimate for the transition density $\hat{p}_\tau(\mathbf{x}_{t+\tau}|\mathbf{x}_t)$ can be formulated. Thereby, the model can be alternatively trained by the maximum likelihood that the observed data pairs are generated by the model. Thereby, it will approximate the true transition density $p_\tau(\mathbf{x}_{t+\tau}|\mathbf{x}_t)$, which generated the data during the simulation.

which estimates the expectation value over all possible configurations $\mathbf{y}$ of ending in state j given the system started in state i. Since $T_{ij} > 0\ \forall i, j$ and $\sum_j T_{ij} = 1\ \forall i$, the matrix is a real transition matrix, hence the name deep MSM. Therefore, it exists a stationary vector with $\boldsymbol{\pi} = \mathbf{T}^T\boldsymbol{\pi}$, which remains unchanged when propagated. This

vector can be used to estimate the equilibrium distribution $\mu(\mathbf{y})$ as:

$$\mu(\mathbf{y}) = \boldsymbol{\pi}^T \mathbf{q}(\mathbf{y}). \tag{3.3}$$

This will allow us to address two shortcomings of the VAMPnets model: The matrix $\mathbf{T}$ will be a real transition matrix and the function $\mathbf{q}(\mathbf{y};\tau)$ allows to sample configurations given that the system is in a specific state.
We approximate the function $\mathbf{q}(\mathbf{y})$ with a NN $\boldsymbol{\gamma}(\mathbf{y})$ via:

$$\mathbf{q}(\mathbf{y}) = \mathrm{diag}(\bar{\boldsymbol{\gamma}})^{-1}\boldsymbol{\gamma}(\mathbf{y})\rho_1(\mathbf{y}), \tag{3.4}$$

where $\rho_1(\mathbf{y})$ is the empirical distribution of all configurations at $t+\tau$ and $\bar{\gamma}_i = \mathbb{E}_{\mathbf{y}\sim\rho_1}[\gamma_i(\mathbf{y})]$ the expectation value of γ_i. The latter normalizes the probability distribution, i.e. $\int_{\mathbf{y}} q_i(\mathbf{y})\mathrm{d}\mathbf{y} = 1$. In order to ensure strictly non-negative probabilities the output of the NN $\boldsymbol{\gamma}$ must be non-negative, which we ensure by a proper choice of the activation function, e.g. *ReLU* [59] or shifted *ELU* [60].
The model can be either trained by maximizing the likelihood of generating the observed T data pairs $(\mathbf{x}_t, \mathbf{x}_{t+\tau})$. The log-likelihood is given by:

$$\mathrm{LL} = \sum_{t=1}^{T} \ln(\boldsymbol{\chi}(\mathbf{x}_t)^T \mathbf{q}(\mathbf{x}_{t+\tau})). \tag{3.5}$$

Alternatively, we can interpret the deep MSM as a Koopman model. When plugging Eq. 3.4 into Eq. 3.1:

$$\hat{p}_\tau(\mathbf{y}|\mathbf{x}) = \boldsymbol{\chi}(\mathbf{x})^T \mathrm{diag}(\bar{\boldsymbol{\gamma}})^{-1}\boldsymbol{\gamma}(\mathbf{y})\rho_1(\mathbf{y}), \tag{3.6}$$

we can by comparison with Eq. 2.43 identify that (cf. Fig. 3.2):

1. $\boldsymbol{\chi}_0(\mathbf{x}) = \boldsymbol{\chi}(\mathbf{x})$.
2. $\boldsymbol{\chi}_1(\mathbf{y}) = \boldsymbol{\gamma}(\mathbf{y})$.
3. $\mathbf{S} = \mathrm{diag}(\bar{\boldsymbol{\gamma}})^{-1}$.

With the above definitions the model can be trained using the VAMP-E score given in Eq. 2.55. Furthermore, the formula for the transition matrix in Eq. 2.54 agrees with the definition given in Eq. 3.2.
Despite the advantages over VAMPnets, the model still suffers from some drawbacks which are mainly connected to the effect of introducing a second NN:

1. The training process is more challenging because the two networks depend on each other.

2. Evaluating the model for a different lagtime τ necessitates to retrain the NN $\boldsymbol{\gamma}$.

3. The model does not guarantee reversibility.

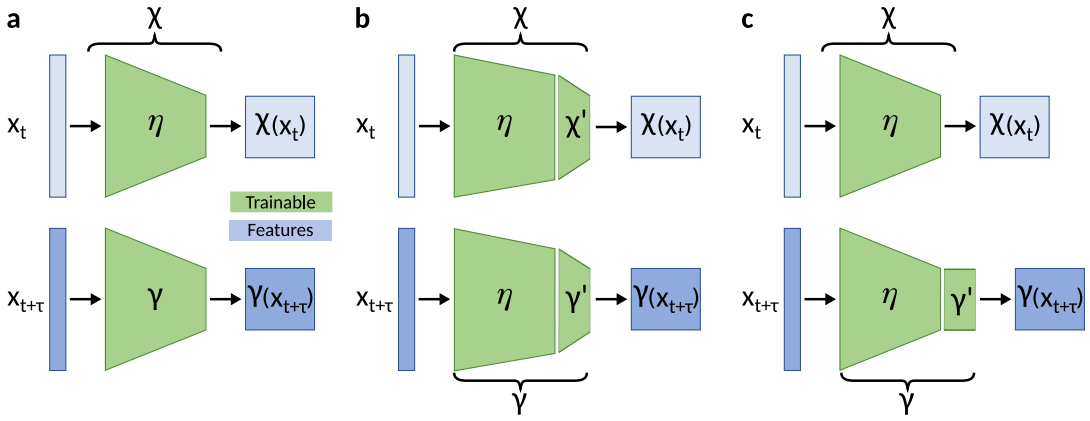

Figure 3.3.: Three different learning architectures for the deep MSM model, where the two mappings $\boldsymbol{\chi}$ and $\boldsymbol{\gamma}$ share a different amount of parameters. (**a**) The most general architecture, where $\boldsymbol{\chi}$ and $\boldsymbol{\gamma}$ are completely independent parameter wise. (**b**) The two NN share the network $\boldsymbol{\eta}$ and differ only by their last layer $\boldsymbol{\chi} = \boldsymbol{\chi}' \circ \boldsymbol{\eta}$, $\boldsymbol{\gamma} = \boldsymbol{\gamma}' \circ \boldsymbol{\eta}$. (**c**) The second network $\boldsymbol{\gamma}$ builds on the output of $\boldsymbol{\chi}$ assuming that the system will most likely jump to a configuration from the same state $\boldsymbol{\gamma} = \boldsymbol{\gamma}' \circ \boldsymbol{\chi}$.

However, the problems arising from the second network can be diminished by the following observation. The state assignment task of a NN can be seen as a two step process. Firstly, it has to extract the important features from the given input $\mathbf{x}$ until the last hidden layer, based on which the last layer assigns the configuration to a specific state. For the task of the second NN $\boldsymbol{\gamma}$ the feature extraction will similarly be an important step. Therefore, we propose to share the feature extraction between the two networks, which means that effectively only one last layer will be added by introducing the NN $\boldsymbol{\gamma}$ (cf. Fig. 3.3 b). It is even possible to go one step further. For estimating the probability $q_i(\mathbf{y})$, an important feature might be the probability to which state the configuration $\mathbf{y}$ itself belongs. Following this argument, the independent layer of $\boldsymbol{\gamma}$ could be downstream to the state assignment network $\boldsymbol{\eta}$ (cf. Fig. 3.3 c).

Furthermore, since the state assignment task is very similar to VAMPnets we implemented a stable training routine:

1. Train the state assignment $\boldsymbol{\chi}$ as a VAMPnet.
2. Train only the $\boldsymbol{\gamma}$ specific part for a deep MSM.
3. Train both networks at the same time.

3.2.1. Making the deep MSM generative

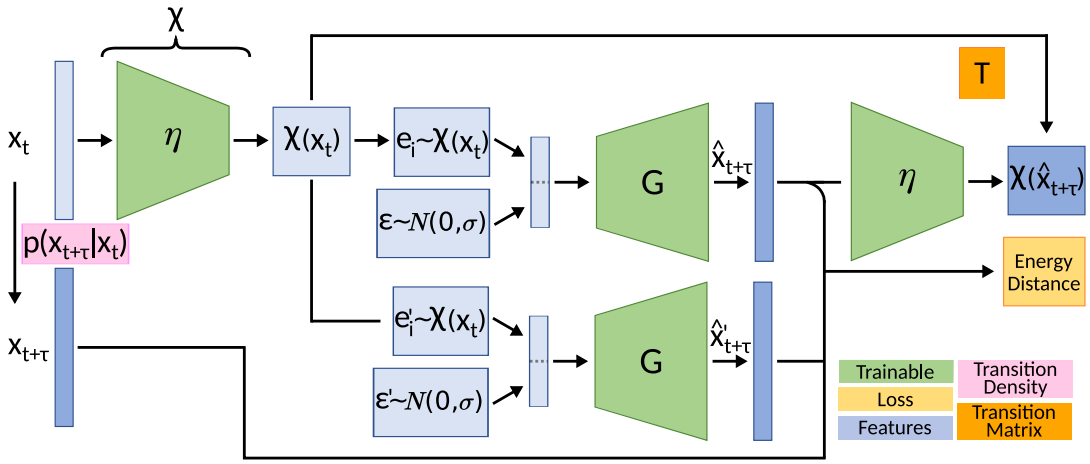

Figure 3.4.: Architecture scheme for the deep generative MSM. The configuration $\mathbf{x}_t$ is again mapped to a state space by $\boldsymbol{\chi}(\mathbf{x}_t) = \boldsymbol{\eta}(\mathbf{x}_t)$. The second network $\mathbf{G}$ targets to sample configurations $\hat{\boldsymbol{x}}_{t+\tau}$ given the system is at state i (represented by the unit vector e_i) at time t and some Gaussian noise vector $\boldsymbol{\epsilon}$. This process has to be done twice in order to train the model by the energy distance Eq. 2.84. By sampling the starting state according to $\boldsymbol{\chi}(\mathbf{x}_t)$, jointly the two NNs aim to sample from the transition density $p_\tau(\mathbf{x}_{t+\tau}|\mathbf{x}_t)$, which generated the data during the simulation. The energy distance can be used to train both networks or only $\mathbf{G}$, where $\boldsymbol{\chi}$ is then pretrained as a VAMPnet or deep MSM. The transition matrix propagating between the state vector $\boldsymbol{\chi}(\mathbf{x}_t)$ and the generated configurations $\boldsymbol{\chi}(\hat{\mathbf{x}}_{t+\tau})$ can be computed by Eq. 3.12.

Up to this point the deep MSM is able to map back to the configurations present in the data. However, it is not capable to sample new unobserved configurations. Instead of training a network to approximate the probability distribution $q_i(\mathbf{y})$, we can train a network to generate samples according to the distribution:

$$\mathbf{y} = \mathbf{G}(\mathbf{e}_i, \boldsymbol{\epsilon}), \tag{3.7}$$

where the vector $\mathbf{e}_i \in \mathbb{R}^k$ is a one-hot encoding of the metastable state and $\boldsymbol{\epsilon}$ is a i.i.d. random vector with each component sampled from a Gaussian normal distribution (cf. Fig. 3.4). We call the resulting model deep generative MSM.
Given a trained state assignment $\boldsymbol{\chi}$, the NN $\mathbf{G}$ can be optimized minimizing the Energy Distance Eq. 2.84. During training we approximate the expectation value as the mean value over a batch. Noticing that $\mathbb{E}[||\mathbf{x}_{t+\tau} - \mathbf{x}'_{t+\tau}||]$ is constant with respect to the model weights and splitting $\mathbb{E}[2||\hat{\mathbf{x}}_{t+\tau} - \mathbf{x}_{t+\tau}||] = \mathbb{E}[||\hat{\mathbf{x}}_{t+\tau} - \mathbf{x}_{t+\tau}|| + ||\hat{\mathbf{x}}'_{t+\tau} - \mathbf{x}_{t+\tau}||]$, the loss can be written as:

$$
\begin{aligned}
D &= \mathbb{E}[||\hat{\mathbf{x}}_{t+\tau} - \mathbf{x}_{t+\tau}|| + ||\hat{\mathbf{x}}'_{t+\tau} - \mathbf{x}_{t+\tau}|| - ||\hat{\mathbf{x}}_{t+\tau} - \hat{\mathbf{x}}'_{t+\tau}||] + \text{const} && (3.8)\\
&= \mathbb{E}[d_t] + \text{const} && (3.9)\\
&\approx \frac{1}{T}\sum_t d_t + \text{const}, && (3.10)
\end{aligned}
$$

where T is the number of transition pairs $(\mathbf{x}_t, \mathbf{x}_{t+\tau})$ coming from the data and

$$
d_t = ||\mathbf{G}(\mathbf{e}_{I_t}, \boldsymbol{\epsilon}) - \mathbf{x}_{t+\tau}|| + ||\mathbf{G}(\mathbf{e}_{I'_t}, \boldsymbol{\epsilon}') - \mathbf{x}_{t+\tau}|| - ||\mathbf{G}(\mathbf{e}_{I_t}, \boldsymbol{\epsilon}) - \mathbf{G}(\mathbf{e}_{I'_t}, \boldsymbol{\epsilon}')||. \quad (3.11)
$$

The probability distribution of the discrete random variables I_t, I'_t are defined as $\mathbb{P}(I_t = i) = \mathbb{P}(I'_t) = \chi_i(\mathbf{x}_t)$.
The generative network $\mathbf{G}(\mathbf{e}_i, \boldsymbol{\epsilon})$ samples configuration $\mathbf{y}$ at time $t+\tau$ given that the system was in a specific state i at time t. Since the states are discrete, the model $\mathbf{G}$ effectively needs to learn only as many functions as metastable states to generate configurations $\mathbf{y}$. This appears to be the less challenging task compared to a model which would be directly conditioned on the starting configuration $\mathbf{x}_t$.
In order to estimate transition probabilities T_{ij}, we simply compute the expectation value of the resulting states of independently sampled configurations $\mathbf{y}$ given the system starts in state i:

$$
T_{ij} = \mathbb{E}_{\boldsymbol{\epsilon}}[\chi_j(\mathbf{G}(\mathbf{e}_i, \boldsymbol{\epsilon}))]. \quad (3.12)
$$

In principle, both networks $\mathbf{G}$ and $\boldsymbol{\eta}$ can be simultaneously trained for the energy distance. However, since I_t, I'_t are discrete valued backpropagating the gradients with respect to the weights of the network $\boldsymbol{\eta}$ is not straight forward. We solved this problem by inspecting the expected loss summing over all possible instances of I_t

and I'_t:

$$\mathbb{E}[d_t|\mathbf{x}_t, \mathbf{x}_{t+\tau}] = \sum_{i,j} \mathbb{P}(I_t = i)\mathbb{P}(I'_t = j)\mathbb{E}_{\boldsymbol{\epsilon}}[d_t] \tag{3.13}$$

$$= \sum_{i,j} \chi_i(\mathbf{x}_t)\chi_j(\mathbf{x}_t)\mathbb{E}_{\boldsymbol{\epsilon}}[d_t]. \tag{3.14}$$

Therefore the gradients of the expected loss with respect to the input to the *softmax* functions $\chi_i = \eta_i = \frac{\exp o_i}{\sum_l \exp o_l}$ of the NN $\boldsymbol{\eta}$ can be estimated as:

$$\frac{\partial}{\partial o_k}\mathbb{E}[d_t|\mathbf{x}_t, \mathbf{x}_{t+\tau}] = \sum_{i,j} \chi_i(\mathbf{x}_t)\chi_j(\mathbf{x}_t)(1_{i=k} + 1_{j=k} - 2\chi_k(\mathbf{x}_t)) \cdot \mathbb{E}_{\boldsymbol{\epsilon}}[d_t] \tag{3.15}$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}, I_t, I'_t}[(1_{I_t=k} + 1_{I'_t=k} - 2\chi_k(\mathbf{x}_t)) \cdot d_t]. \tag{3.16}$$

The loss can be further backpropagated with the normal chain rule to all parameters of $\boldsymbol{\eta}$. The expectation values will be approximated during training by drawing one instance of $I_t, I'_t, \boldsymbol{\epsilon}_t, \boldsymbol{\epsilon}'_t$ for each data pair $(\mathbf{x}_t, \mathbf{x}_{t+\tau})$ in the mini batch and averaging.

3.3. Deep reversible Koopman model

As we have seen for the deep MSM, we can learn a Koopman model by not only training for $\boldsymbol{\chi}_0$ and $\boldsymbol{\chi}_1$, but also making the matrix $\mathbf{S}$ trainable. As demonstrated in Sec. 2.1.6, the matrix $\mathbf{S}$ must be symmetric for a reversible model. Furthermore, the stationary distribution has to be learned on the fly by reweighting the observed empirical distribution appropriately.
Our proposed reversible model makes the following choices (cf. Fig. 3.5) [78]:

1. $\boldsymbol{\chi}_0(\mathbf{x}) = \boldsymbol{\chi}(\mathbf{x})$.

2. $\boldsymbol{\chi}_1(\mathbf{y}) = \boldsymbol{\chi}(\mathbf{y})\boldsymbol{\chi}^T(\mathbf{y})\mathbf{u}$.

3. $\mathbf{S} = \mathbf{S}^T$.

The function $\boldsymbol{\chi}(\mathbf{x})$ maps again a configuration $\mathbf{x}$ through the *softmax* function to a state space. The stationary distribution is modeled by $\mu(\mathbf{y}) = \boldsymbol{\chi}^T(\mathbf{y})\mathbf{u}\rho_1(\mathbf{y})$, where the trainable vector $\mathbf{u}$ reweights the states defined by $\boldsymbol{\chi}$ to the stationary distribution. In order to make the model reversible, the matrix $\mathbf{S}$ is trainable and

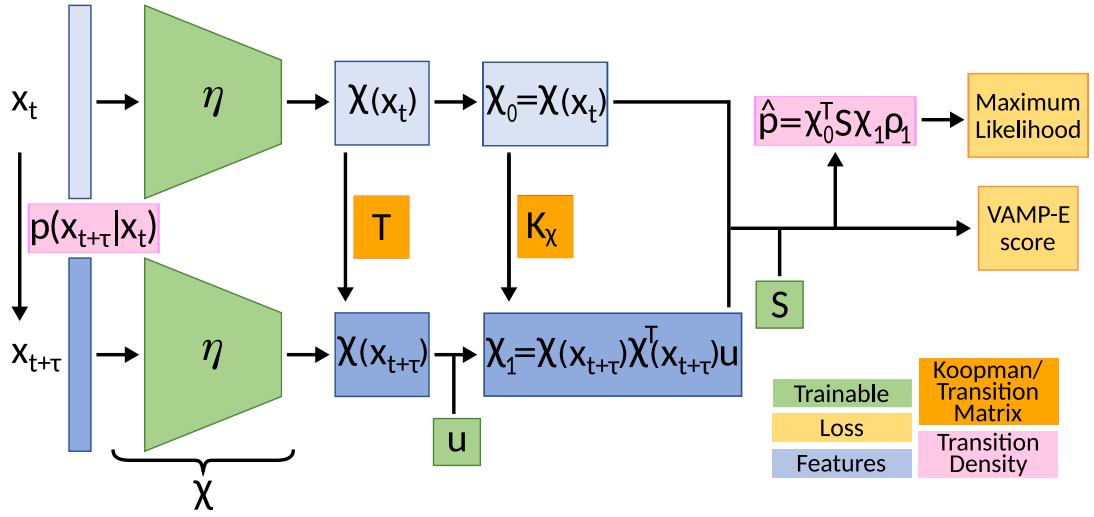

Figure 3.5.: Architecture scheme for the deep reversible Koopman model. The data pair $(\mathbf{x}_t, \mathbf{x}_{t+\tau})$ is individually transformed to the state space $\boldsymbol{\chi}(\mathbf{x}_t)$. By updating $\boldsymbol{\chi}(\mathbf{x}_{t+\tau})$ with the weights $w(\mathbf{x}_{t+\tau}) = \boldsymbol{\chi}^T(\mathbf{x}_{t+\tau})\mathbf{u}$ defined by the trainable instance $\mathbf{u}$, a Koopman model is constructed. With the additional trainable matrix $\mathbf{S}$ an approximation of the transition density can be constructed. Appropriate constraints on $\mathbf{u}$ and $\mathbf{S}$ make the model reversible with respect to the learned stationary distribution $\mu(\mathbf{x}_{t+\tau}) = \boldsymbol{\chi}^T(\mathbf{x}_{t+\tau})\mathbf{u}\rho_1(\mathbf{x}_{t+\tau})$. Similarly to the deep MSM, the model can be trained either with the VAMP-E or maximum likelihood score. The transition matrix $\mathbf{T}$ in the state space is given by Eq. 2.54.

symmetric. The final transition density reads then:

$$p_\tau(\mathbf{y}|\mathbf{x}) = \boldsymbol{\chi}(\mathbf{x})^T \mathbf{S} \boldsymbol{\chi}(\mathbf{y}) \mu(\mathbf{y}). \tag{3.17}$$

Two further constraints have to be fulfilled to guarantee a normalized equilibrium and transition density:

1. **Normalization of equilibrium density**: $\int \mu(\mathbf{y})\mathrm{d}\mathbf{y} = \int \boldsymbol{\chi}^T(\mathbf{y})\rho_1(\mathbf{y})\mathrm{d}\mathbf{y}\ \mathbf{u} = \bar{\boldsymbol{\chi}}^T\mathbf{u} = 1$, where $\bar{\boldsymbol{\chi}}$ is the empirical state probability over $\rho_1(\mathbf{y})$.

2. **Normalization of transition density**:
   $\int \hat{p}_\tau(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{y} = \boldsymbol{\chi}^T(\mathbf{x})\mathbf{S}\int \boldsymbol{\chi}(\mathbf{y})\boldsymbol{\chi}^T(\mathbf{y})\rho_1(\mathbf{y})\mathrm{d}\mathbf{y}\ \mathbf{u} = \boldsymbol{\chi}^T(\mathbf{x})\mathbf{S}\mathbf{C}_{\tau\tau}\mathbf{u} = 1$, if $\mathbf{S}\mathbf{C}_{\tau\tau}\mathbf{u} = \mathbf{1}$ with $\mathbf{C}_{\tau\tau} = \int \boldsymbol{\chi}(\mathbf{y})\boldsymbol{\chi}^T(\mathbf{y})\rho_1(\mathbf{y})\mathrm{d}\mathbf{y}$. As a result the transition matrix preserves probability mass $\mathbf{T}\mathbf{1} = \mathbf{S}\boldsymbol{\Sigma}\mathbf{1} = \mathbf{S}\mathbf{C}_{\tau\tau}\mathbf{u} = \mathbf{1}$.

In order to make it a Markov state model, we additionally need that all elements in the transition matrix are non-negative, which is fulfilled if:

$$\text{All elements of } \mathbf{u}, \mathbf{S} \text{ are non-negative.} \tag{3.18}$$

This last constraint can be switched on or off depending on what kind of model is favored by the practitioner, a reversible VAMPnet or a reversible deep MSM. In fact, by ceasing the symmetry constraint of $\mathbf{S}$, the model can be either seen as an alternative approach of a VAMPnet or a deep MSM, where an additional reweighting and the Koopman matrix are learned and not estimated from the data. We could prove that the constructed reversible model is an universal approximator for reversible Markov processes [78].
As for the deep MSM the model can be trained for two different objectives: The VAMP-E score or the maximum likelihood.
However, we need to enforce the constraints during training. We will achieve that by making $\mathbf{u}$ and $\mathbf{S}$ functions of trainable parameters $\mathbf{w_u}$ and $\mathbf{W_S}$, respectively. In the case of $\mathbf{u}$ the normalization of the equilibrium density is easily fulfilled by $\mathbf{w}_1 = f(\mathbf{w_u})$:

$$\mathbf{u} = \frac{\mathbf{w}_1}{\tilde{\boldsymbol{\chi}}^T \mathbf{w}_1}, \tag{3.19}$$

where $f(\mathbf{w_u})$ needs to be an element wise function mapping to non-negative values in case of a deep reversible MSM, e.g. the $ReLU$ activation function.
If applicable, the symmetry and the non-negativity of $\mathbf{S}$ can be similarly enforced:

$$\mathbf{W}_1 = f(\mathbf{W_S}) + f(\mathbf{W_S}^T), \tag{3.20}$$

where f has the same properties as above. Furthermore, the normalization of the transition density needs to be fulfilled $\mathbf{S}\mathbf{C}_{\tau\tau}\mathbf{u} = \mathbf{S}\mathbf{v} = \mathbf{1}$. In order to keep the symmetry untouched, we can still update the diagonal of the matrix $\mathbf{W}_1$ by values $\mathrm{diag}(\mathbf{w}_2)$ and normalize the matrix by a factor z:

$$\mathbf{S} = \mathbf{W}_1/z + \mathrm{diag}(\mathbf{w}_2). \tag{3.21}$$

$$(\mathbf{S}\mathbf{v})_i = (\mathbf{W}_1\mathbf{v})_i/z + \mathbf{w}_i\mathbf{v}_i. \tag{3.22}$$

$$\mathbf{w}_i = \frac{1 - (\mathbf{W}_1\mathbf{v})_i/z}{\mathbf{v}_i}. \tag{3.23}$$

With these choices we normalized the transition density. However, it could be that elements of $\mathbf{S}$ are now negative. To avoid that, we have to choose z appropriately:

$$0 \leq (\mathbf{W}_1)_{ii}/z + (\mathbf{w}_2)_i \; \forall_i \tag{3.24}$$

$$z \geq (\mathbf{W}_1\mathbf{v})_i - (\mathbf{W}_1)_{ii}\mathbf{v}_i \; \forall_i. \tag{3.25}$$

An optimal choice would be $z = ||\mathbf{W}_1\mathbf{v} - \mathrm{diag}(\mathbf{W}) \odot \mathbf{v}||_{\mathrm{inf}}$, where $\odot$ means an element wise multiplication. Any function which returns a larger value than the maximum norm is applicable, although it should be differentiable for gradient based optimization methods. Any p-norm fulfills these requirements. The higher the order the closer the value will be to the maximum norm. This is preferable, since with larger values of z the matrix $\mathbf{S}$ will be dominated by its diagonal values and thus harder to train.
Despite its flexible structure of incorporating contraints for reversibility and a stochastic transition matrix and thereby training for different models, the method has some drawbacks:

1. The training process is challenging since the trainable parts $\boldsymbol{\chi}, \mathbf{S}, \mathbf{u}$ are interdependent. Therefore, if one of them is stuck in a suboptimal solution, the others will only be optimal with respect to that. NNs have shown great success in avoiding local minima. However, $\mathbf{u}$ and $\mathbf{S}$ are a parametrized vector or matrix and not a NN parametrized function and can thus suffer from that.

2. Depending on the choice of activation functions for $\mathbf{u}$ and $\mathbf{S}$, the trainable parts will be updated differently fast.

3. Evaluating the model for a different lagtime τ necessitates to retrain $\mathbf{S}$ and $\mathbf{u}$.

The first problem can be mitigated by a smart initialization. Given a state assignment $\boldsymbol{\chi}$ trained with a VAMPnet, an optimal $\mathbf{u}$ can be estimated from the stationary vector $\boldsymbol{\pi}^T = \boldsymbol{\pi}^T\mathbf{K}_{\boldsymbol{\chi}}$, where $\mathbf{K}_{\boldsymbol{\chi}}$ is the Koopman matrix of the VAMPnet:

$$\boldsymbol{\pi} = \int \boldsymbol{\chi}(\mathbf{x})\mu(\mathbf{x})\mathrm{d}\mathbf{x} \tag{3.26}$$

$$= \int \boldsymbol{\chi}(\mathbf{x})\boldsymbol{\chi}^T(\mathbf{x})\rho_1(\mathbf{x})\mathrm{d}\mathbf{x}\; \mathbf{u} \tag{3.27}$$

$$= \mathbf{C}_{11}\mathbf{u}. \tag{3.28}$$

$$\Rightarrow \mathbf{u} = \mathbf{C}_{11}^{-1}\boldsymbol{\pi}, \tag{3.29}$$

where the normalization of the stationary distribution must be still fulfilled by Eq. 3.19. Our preferred training routine can be described as follows:

1. Train the state assignment $\boldsymbol{\chi}$ as a VAMPnet.
2. Set $\mathbf{u}$ to its optimal value according to the VAMPnet.
3. Train $\mathbf{S}$ to its optimal value.
4. Train all three instances at the same time.

In order to tackle problem 2 from above the fourth training step can be updated to a more elaborate training routine: Trying to keep $\mathbf{u}$ and $\mathbf{S}$ close to its optimal values by alternating between training for $(\boldsymbol{\chi}, \mathbf{u}, \mathbf{S})$ and only training for $(\mathbf{u}, \mathbf{S})$.

3.3.1. Incorporate experimental measurements into model estimation

Further advancements have been proposed for standard MSM estimations. One important branch is concerned with incorporating existing experimental information into the model estimation [42–49]. Thereby, these methods aim to overcome systematic errors in the force field of the simulation, which can improve the estimation of observables not measured by experiment. This approach can be transferred to the deep model estimation.
We assume that experiments, such as fluorescence, chemical shift in NMR, and IR spectroscopy, measure the ensemble average of a scalar observable a. Furthermore, we make the assumption that we can map each conformation $\mathbf{x}$ to an observable value $a(\mathbf{x})$. According to the ergodic hypothesis, the ensemble average of the observable equals the time average over a long simulation sampling the equilibrium distribution. Therefore, the time average value can be estimated by the reversible deep MSM via:

$$\mathbb{E}[a] \approx \sum_{t=1}^{T} \mu(\mathbf{x}_t) a(\mathbf{x}_t). \tag{3.30}$$

Due to the bias in the simulation and finite sampling the estimated value might differ from the experiment. However, if experimental information is available, they can be incorporated into model estimation and possibly remove the bias. Therefore, we propose to extent the loss function to additionally match the observables measured

by experiments:

$$L_{\text{total}} = L_{\text{MSM}} + \sum_i \xi_i ||O_i - \mathbb{E}[a_i]||^2, \tag{3.31}$$

where O_i is the ensemble average measured by experiment for the ith observable and L_{MSM} is the score used to train the reversible deep MSM. Each observable can be weighted by ξ_i, which encodes uncertainty about the observation [79].
Since L_{MSM} is system and hyperparameter dependent (e.g. number of states, lag time τ) and the observables can origin from different domains with different units, a general rule how to choose ξ cannot be defined. However, each experimental measurement carries an error estimate. The regularization parameters ξ_i should therefore be chosen as small as possible that still sufficient many model estimated observables fall into the standard deviation of the experiment. Thereby, the kinetics estimation of the VAMP score is disturbed as little as possible.
The same procedure can be applied for available kinetic information through time-correlation experiments. The expectation value of these time-correlations can be expressed via the model as:

$$\mathbb{E}[a(t)a(t+n\tau)] \approx \sum_{t_1=1}^{T} \mu(\mathbf{x}_{t_1}) a(\mathbf{x}_{t_1}) \sum_{t_2=1}^{T} p_\tau(\mathbf{x}_{t_2}|\mathbf{x}_{t_1}) a(\mathbf{x}_{t_2}) \tag{3.32}$$

$$= \mathbf{a}^T \mathbf{X}(n\tau)\mathbf{a}, \tag{3.33}$$

where $\mathbf{a}$ is the average value of the observable within each state and $\mathbf{X}(n\tau)$ is the unconditional probability to jump between the states [42]. We express the quantities via the model parameters as:

$$\mathbf{a}_i = \sum_t a(\mathbf{x}_t) \frac{\boldsymbol{\chi}_i(\mathbf{x}_t)\mu(\mathbf{x}_t)}{\sum_{t'} \boldsymbol{\chi}_i(\mathbf{x}_{t'})\mu(\mathbf{x}_{t'})}, \tag{3.34}$$

$$\mathbf{X}(n\tau) = \boldsymbol{\Sigma}\mathbf{T}^n. \tag{3.35}$$

A third possible observable for an experiment are relaxation timescales t_i, which can be connected to folding rates [79]. These are directly related to the eigenvalues λ_i of the transition matrix $t_i = -\log(\lambda_i)/\tau$. Since the optimization of the eigenvalues would require the estimation of their gradients, which is numerically unstable for the non-Hermitian transition matrix and not supported by the common deep learning frameworks, we make use of Eq. 2.61, which shows that $\mathbf{T}$ is similar to - and has

therefore the same eigenvalues as - the Hermitian matrix $\bar{\mathbf{S}}$:

$$\bar{\mathbf{S}} = \mathbf{\Sigma}^{\frac{1}{2}}\mathbf{S}\mathbf{\Sigma}^{\frac{1}{2}} = \mathbf{\Sigma}^{\frac{1}{2}}\mathbf{T}\mathbf{\Sigma}^{-\frac{1}{2}}. \tag{3.36}$$

Both these observables can be included into the loss function to match experimental values as shown in Eq. 3.31.

3.4. Coarse-graining with VAMP

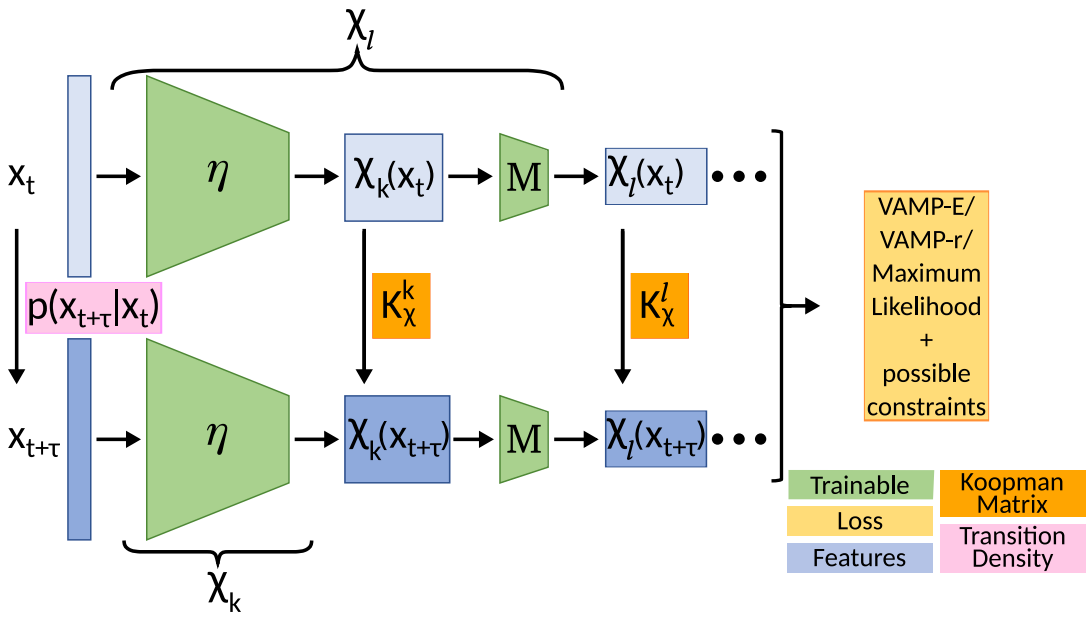

Figure 3.6.: Coarse-graining architecture. The state definition of $\boldsymbol{\chi}_k(\mathbf{x}_t) = \boldsymbol{\eta}(\mathbf{x}_t)$ can be mapped to a coarser representation by a matrix $\mathbf{M}$ which maps each state with some probability to a new state $\boldsymbol{\chi}_l(\mathbf{x}_t) = \mathbf{M}^T\boldsymbol{\chi}_k(\mathbf{x}_t)$. $\mathbf{M}$ is constructed by trainable parameters fulfilling probability constraints and can be trained for the same losses introduced above. However, in the case of the deep MSM or the reversible Koopman model consistency between the two representations regarding the transition density or stationary distribution should be taken into account. This can be either achieved by estimating the additional parameters according to these constraints (Eq. 3.45-3.46), or by retraining them with an updated loss function (Eq. 3.47). The model is not restricted to one coarse-graining matrix $\mathbf{M}$, but can be expanded by applying several in succession.

When studying highly complex systems which might exhibit many metastable states, it can be beneficial to study the system on different levels of resolution, where

a set of metastable states is grouped into a new metastable state. The assignment from k metastable states $\boldsymbol{\chi}_k(\mathbf{x})$ to l new metastable states $\boldsymbol{\chi}_l(\mathbf{x})$ with $l < k$ is optimal in the variational approach, if the feature functions can still span the l leading left and right singular functions. Therefore, we can still optimize for any VAMP score as long as the assignment $\mathbf{M} \in \mathbb{R}^{k \times l}$ is differential and fulfills the following constraints (cf. Fig. 3.6):

$$\boldsymbol{\chi}_l^T(\mathbf{x}_t) = \boldsymbol{\chi}_k^T(\mathbf{x}_t)\mathbf{M}. \tag{3.37}$$

$$M_{ij} > 0 \; \forall \; i, j. \tag{3.38}$$

$$\sum_j M_{ij} = 1 \; \forall \; i. \tag{3.39}$$

The coarse-graining matrix $\mathbf{M}$ can be restraint by choosing trainable parameters m_{ij} so that:

$$M_{ij} = \frac{\exp(m_{ij})}{\sum_\nu \exp(m_{i\nu})}. \tag{3.40}$$

In case of the deep MSM and reversible deep MSM it is not sufficient to simply map the state assignments $\boldsymbol{\chi}_k$ to the finer representation, but new optimal values for $\mathbf{q}_l$ or $\mathbf{S}_l$ and $\mathbf{u}_l$ have to be found. However, given a coarse-graining matrix $\mathbf{M}$, the optimal values can be estimated without the need to retrain them. This is possible by establishing consistency between the models: The transition probability for the finer model $p_{k,\tau}(\mathbf{y}|\mathbf{x}) \stackrel{!}{=} p_{l,\tau}(\mathbf{y}|\mathbf{x})$ should be as close as possible to the coarser one. Additionally, for the reversible model the same is true for the stationary distribution $\mu_k(\mathbf{y}) \stackrel{!}{=} \mu_l(\mathbf{y})$.
These conditions result in the following equations:

$$\mathbf{q}_k \stackrel{!}{=} \mathbf{M}\mathbf{q}_l. \tag{3.41}$$

$$\mathbf{u}_k \stackrel{!}{=} \mathbf{M}\mathbf{u}_l. \tag{3.42}$$

$$\mathbf{S}_k \stackrel{!}{=} \mathbf{M}\mathbf{S}_l\mathbf{M}^T. \tag{3.43}$$

With the singular value decomposition $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, the least square solution to these problems can be found via the pseudoinverse $\mathbf{M}^* = \mathbf{V}\mathbf{D}^{*T}\mathbf{U}^T$, where $\mathbf{D}^*$ has

the reciprocal value of every non-zero diagonal element of $\mathbf{D}$:

$$\mathbf{q}_l = \mathbf{M}^* \mathbf{q}_k. \tag{3.44}$$

$$\mathbf{u}_l = \mathbf{M}^* \mathbf{u}_k. \tag{3.45}$$

$$\mathbf{S}_l = \mathbf{M}^* \mathbf{S}_k \mathbf{M}^{*T}. \tag{3.46}$$

However, since the pseudoinverse represents the least square solution, the resulting probability distribution might not be normalized. Therefore, a final normalization has to be executed (cf. Eq. 3.19 for $\mathbf{u}_l$, Eq. 3.21 for $\mathbf{S}_l$, and $\bar{\boldsymbol{\gamma}}_l = \mathbb{E}[\mathbf{M}^* \boldsymbol{\gamma}_k]$ for $\mathbf{q}_l$). Alternatively, the parameters for the coarse representation can be retrained together with the coarse-graining matrix $\mathbf{M}$. But we propose to add a penalty to the loss if the consistency is violated, e.g. for the deep MSM:

$$L = L_{\text{MSM}} + ||p_{k,\tau}(\mathbf{y}|\mathbf{x}) - p_{l,\tau}(\mathbf{y}|\mathbf{x})||^2. \tag{3.47}$$

Independent of which method is used, we want to find the optimal solutions of the common parameters for both representations k, l. Therefore, we propose to train for the sum of the two VAMP-scores. Furthermore, the approach is not restricted to one coarse-grained layer. Instead, several instances can be applied in succession [79].

3.5. iVAMPnets for decomposing macromolecules into independent Markovian domains

We aim to describe the system by independent Koopman operators, which will allow us to construct a global operator, which is unfeasible to estimate directly.
As shown in Sec. 2.1.8 the global Koopman operator decomposes in its independent subsystem operators if the global feature functions are written as the product of the individual feature functions $\boldsymbol{\chi}^i(\mathbf{x}_t)$ of subsystem i. Therefore, we construct the global feature functions as the Kronecker product, which estimates the product of all possible combinations:

$$\boldsymbol{\chi}^G(\mathbf{x}_t) = \bigotimes_i \boldsymbol{\chi}^i(\mathbf{x}_t). \tag{3.48}$$

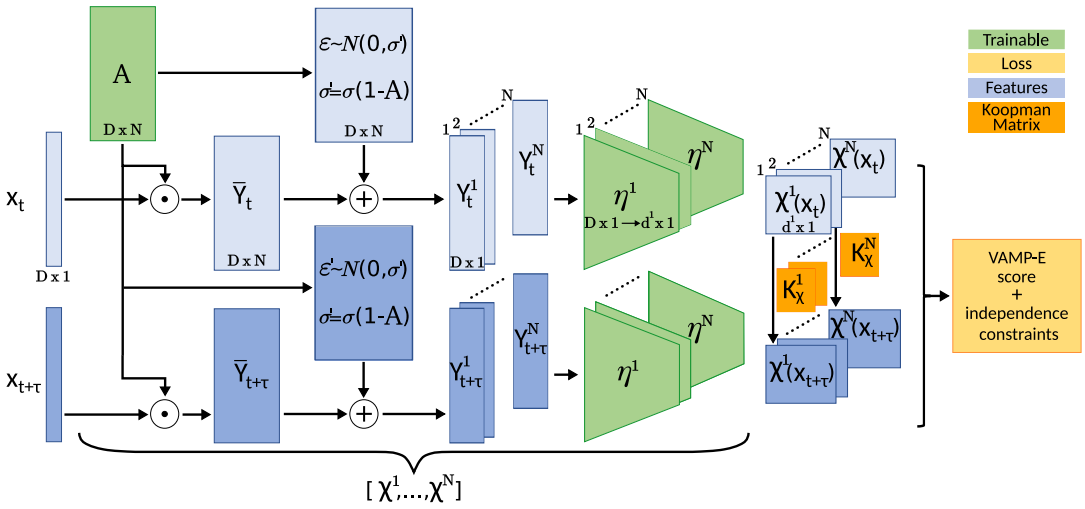

Figure 3.7.: Scheme of the iVAMPnets architecture for N subsystems with the attention mechanism (cf. Sec. 3.6.2). Two lobes are given for configuration pairs $\mathbf{x}_t$ and $\mathbf{x}_{t+\tau}$, where the weights are shared. Firstly, the input features are element wise weighted $\bar{\mathbf{Y}}_t = \mathbf{A} \odot \mathbf{x}_t$ with a mask $\mathbf{A} \in \mathbb{R}^{D \times N}$, where each subsystem learns its individual weighting. The mask values can be interpreted as probabilities to which subsystem the input feature belongs. In order to prevent the subsequent neural network to reverse the effects of the mask, we draw for each input feature i and subsystem j an independent, normally distributed random variable $\epsilon_{ij} \sim \mathcal{N}(0, \sigma(1 - A_{ij}))$. This noise is added to the weighted features $\mathbf{Y}_t = \bar{\mathbf{Y}}_t + \boldsymbol{\epsilon}$. Thereby, the attention weight linearly interpolates between input feature and Gaussian noise, i.e., if the attention weight $A_{ij} = 1$, Y_{ij} carries exclusively the input feature x_i, if $A_{ij} = 0$, Y_{ij} is simple Gaussian noise. Afterwards, the transformed feature vector is split for each individual subsystem $\mathbf{Y}_t = [\mathbf{Y}_t^1, ..., \mathbf{Y}_t^N]$ and passed through the subsystem specific neural network $\boldsymbol{\eta}^i$. We call the whole transformation for a subsystem i $\boldsymbol{\chi}^i(\mathbf{x}_t) = \boldsymbol{\eta}^i(\mathbf{Y}_t^i(\mathbf{x}_t))$. The whole model can be trained for the updated VAMP-E score Eq. 3.56 and the additional independence constraints Eq. 3.57.

Furthermore, the same construction is applied for the singular functions:

$$\hat{\mathbf{U}}^G = \bigotimes_i \mathbf{U}^i \tag{3.49}$$

$$\hat{\mathbf{V}}^G = \bigotimes_i \mathbf{U}^i \tag{3.50}$$

$$\hat{\mathbf{K}}^G = \bigotimes_i \mathbf{K}^i. \tag{3.51}$$

If the participating subsystems are independent, the constructed singular functions are indeed the singular functions of the global system and therefore the following statements are true:

$$\mathcal{R}^G = \prod_i \mathcal{R}^i \tag{3.52}$$

$$(\hat{\mathbf{U}}^G)^T \mathbf{C}_{00}^G \hat{\mathbf{U}}^G = \mathbf{1} \tag{3.53}$$

$$(\hat{\mathbf{V}}^G)^T \mathbf{C}_{11}^G \hat{\mathbf{V}}^G = \mathbf{1} \tag{3.54}$$

$$(\hat{\mathbf{U}}^G)^T \mathbf{C}_{01}^G \hat{\mathbf{V}}^G = \hat{\mathbf{K}}^G. \tag{3.55}$$

$\mathbf{C}^G$ represents the correlation matrix in the global feature space and $\mathcal{R}$ is a VAMP score. By these equations it is possible to test if the independence is fulfilled.
The global model can be scored via the validation VAMP-E score 2.39, where we test how well the constructed operator can predict the dynamics in the global feature space:

$$\mathcal{R}_E^G = \mathrm{tr}[2\hat{\mathbf{K}}^G(\hat{\mathbf{U}}^G)^T \mathbf{C}_{01}^G \hat{\mathbf{V}}^G - \hat{\mathbf{K}}^G(\hat{\mathbf{U}}^G)^T \mathbf{C}_{00}^G \hat{\mathbf{U}}^G \hat{\mathbf{K}}^G (\hat{\mathbf{V}}^G)^T \mathbf{C}_{11}^G \hat{\mathbf{V}}^G] \tag{3.56}$$

This procedure would still require to estimate covariance matrices in the global space which suffers from the exponential growth. However, as an approximation it is sufficient to estimate the score and the independence equations only pairwise. Thereby, it is validated that all global singular functions are indeed independent σ_{ind}^G which are constructed by at most two singular functions with singular values smaller than 1: $\sigma_{\mathrm{ind}}^G \in \{\sigma^i \sigma^j \prod_{l \notin \{i,j\}} \sigma_l | \sigma_l = 1\}$. As a consequence, we can only safely propagate observables in the global feature space, which can be written as the product of two subsystems. However, since the neglected global singular functions are the product of at least three singular values smaller than 1 their contribution in Hilbert-Schmidt norm is expected to be minor. Furthermore, it is in principle possible to validate after the training whether these specific singular functions fulfill the independence constraints.
Our final idea is to define N state assignments $\boldsymbol{\chi}^i$ and train them simultaneously for Eq. 3.56, which is evaluated pairwise, and at the same time enforce Eq. 3.52:

$$L_{\mathrm{iVAMP}} = -\sum_{i<j} \mathcal{R}_E^{ij} + \sum_{i<j} \frac{||\mathcal{R}_E^{ij} - \mathcal{R}_E^i \mathcal{R}_E^j||}{\mathcal{R}_E^{ij}}. \tag{3.57}$$

We use the remaining Eqs. 3.53 3.54 3.55 for independence to validate the final trained model (pairwise if needed):

$$M_U = ||(\hat{\mathbf{U}}^G)^T \mathbf{C}_{00}^G \hat{\mathbf{U}}^G - \mathbf{1}|| \tag{3.58}$$

$$M_V = ||(\hat{\mathbf{V}}^G)^T \mathbf{C}_{\tau\tau}^G \hat{\mathbf{V}}^G - \mathbf{1}|| \tag{3.59}$$

$$M_{UV} = ||(\hat{\mathbf{U}}^G)^T \mathbf{C}_{0\tau}^G \hat{\mathbf{V}}^G - \hat{\mathbf{K}}^G|| \tag{3.60}$$

$$M_R = \frac{|\mathcal{R}_E^G - \prod_i \mathcal{R}_E^i|}{\mathcal{R}_E^G}. \tag{3.61}$$

3.6. Attention mechanism

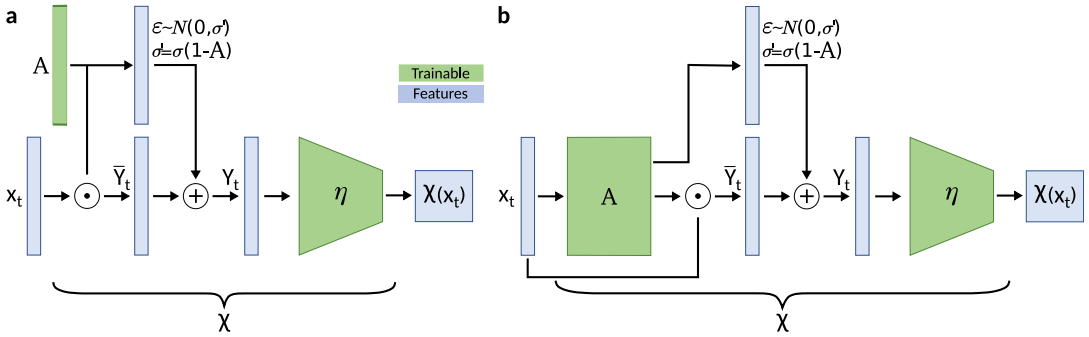

Figure 3.8.: Scheme of the possible attention mechanism with noise constraints. (**a**) A time independent attention mechanism, where the attention weights are a vector $\mathbf{A}$ of the same size as the input features $\mathbf{x}_t$. (**b**) The attention weights are estimated by an additional NN $\mathbf{A}$, whose input is given by $\mathbf{x}_t$ making it time dependent. For both cases, the resulting weights, which are normalized along the input dimension, are multiplied element wise with the input features $\bar{\mathbf{Y}}_t = \mathbf{A} \odot \mathbf{x}_t$. In order to prevent the subsequent neural network to reverse the effects of the mask, we draw for each input feature i an independent, normally distributed random variable $\epsilon_i \sim \mathcal{N}(0, \sigma(1 - A_i))$. This noise is added to the weighted features $\mathbf{Y}_t = \bar{\mathbf{Y}}_t + \boldsymbol{\epsilon}$. Finally, the whole transformation is given by $\boldsymbol{\chi}(\mathbf{x}_t) = \boldsymbol{\eta}(\boldsymbol{Y}_t(\mathbf{x}_t))$. Thereby, inputs which were scaled down by the attention weights will not be distinguishable from the noise.

In applications it is of great interest to understand how the NN, which is often seen as a black box, assigns states. For the application on proteins this translates to the questions which residues are important for the assignment to metastable states. There exists two major approaches to extract these information: the post hoc analysis of the network [80–83], or a simultaneously trained attention mechanism

[84, 85]. Here, we focus on the latter.
By adding a second trainable part $\mathbf{A}$, acting as a mask, we scale the input features to the network $\boldsymbol{\eta}$ by an element wise multiplication $\boldsymbol{\chi}(\mathbf{x}_t) = \boldsymbol{\eta}(\mathbf{A} \odot \mathbf{x}_t)$. Due to a *softmax* output function of the attention mechanism, its output is an indication how important the individual input features are.
It is possible to make $\mathbf{A}(\mathbf{x}_t)$ explicitly dependent on the configuration $\mathbf{x}_t$. Thereby, it can be studied which input features are important for that particular configuration. However, this task requires $\mathbf{A}$ to be a more complex function, e.g. represented by a neural network. If $\mathbf{A}$ is not configuration dependent, a trainable vector of the same size as $\mathbf{x}_t$ would be sufficient.
In principle, the consecutive neural network $\boldsymbol{\eta}$ could revert the effects of the attention mask by rescaling the values back to its former magnitude. Therefore, we propose to add a weight dependent noise term to the scaled input features $\boldsymbol{\eta}(\mathbf{A} \odot \mathbf{x}_t + \boldsymbol{\epsilon})$, where $\epsilon_i \sim \mathcal{N}(0, \sigma(1 - A_i))$ is sampled from an independent, normally distributed random variable with zero mean and standard deviation $\sigma_i' = \sigma(1 - A_i)$. Thereby, the attention weight linearly interpolates between input feature and Gaussian noise, i.e., if the attention weight for the input feature i is large $A_i \approx 1$, the added noise is small $\epsilon_i \approx 0$ and the network $\boldsymbol{\eta}$ receives a clear signal from x_i. However, if $A_i \approx 0$, the ith input to $\boldsymbol{\eta}$ is dominated by noise. A large value of the tuneable σ will cause a harder assignment of the importance weights to overcome the noise level.
This attention mechanism can be in principle used with any method presented here.

3.6.1. Attention mechanism for proteins

In order to make the prediction for proteins invariant under rotation and translation, internal coordinates are often used as input features, e.g. residue distances d_{ij}. For interpretation purposes it is preferable to have importance weights for each residue. Therefore, we propose to define $\mathbf{A}$ in such a way, that it predicts weights A_i for each residue i, which are normalized along the residue axis. The input to the network $\boldsymbol{\eta}$ is scaled as $x_{ij} = f(d_{ij})A_iA_j$, where f can be an arbitrary function.
In order to smoothen the attention weights along the residue chain, $\mathbf{A}$ does not directly predict the attention weights for each residue but instead predicts values $\bar{\mathbf{a}}$ for a window. These are normalized by a *softmax* function. Furthermore, they are shared between residues within a window along the chain and the weight for a residue is constituted by the product of all window weights it falls into $\bar{A}_i = \prod_{\{j|\mathrm{Res}_i \text{ in window } j\}} \bar{\mathbf{a}}_j$. The size of the windows B and the distance s between them

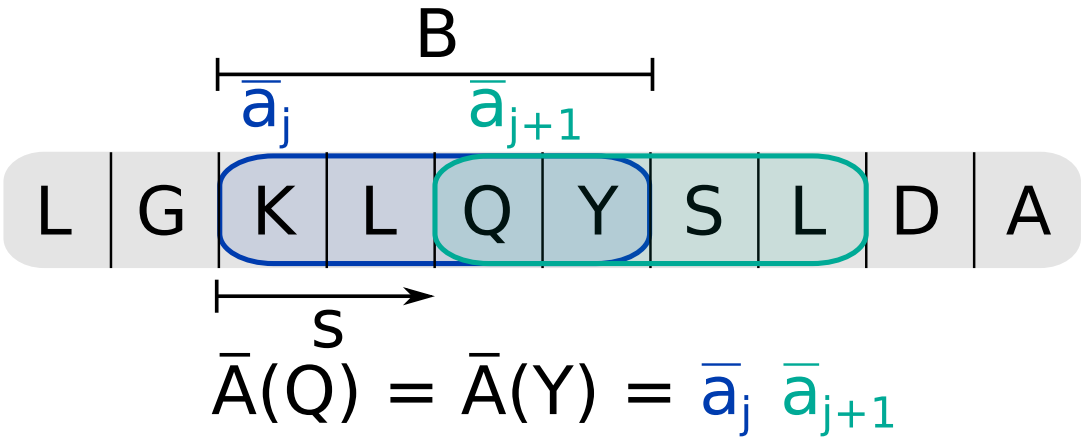

Figure 3.9.: Scheme for assigning attention weights along a residue chain of a protein. Windows of size B are placed along the chain with a step size of s resulting into W many windows. Each window is assigned with a normalized attention weight $\bar{\mathbf{a}}$. Here a window size of $B = 4$ and a step size of $s = 2$ is chosen. As a consequence the weight of the amino acid glutamine (Q) is given as the product of the two windows it is part of $\bar{A}(Q) = \bar{a}_j \bar{a}_{j+1}$. The choice of the step size determines how many neighboring amino acids have the exact same weight, which applies here for the tyrosine (Y). Together with the window size it is regulated how many residues share parts of their weights. Hence, the serine (S) shares the weight $\bar{a}_{j+1}$ with the previous two amino acids $\bar{A}(S) = \bar{a}_{j+1}\bar{a}_{j+2}$, which has a smoothing effect on the attention mechanism along the chain.

are hyperparameters (cf. Fig. 3.9).

In order to make the attention weights of the residues sparse, a threshold θ is introduced $\hat{A}_i = \mathrm{ReLU}(\bar{A}_i - \theta)$ and the resulting weights are normalized across all residues: $A_i = \frac{\hat{A}_i}{\sum_j \hat{A}_j}$.

3.6.2. iVAMPnets with attention

In the case of iVAMPnets the question to answer by the attention mechanism shifts to which residues belong to which subsystem. Therefore, slightly changes are necessary. If N is the number of subsystems and D the number of residues, the attention matrix has the size $\mathbf{A} \in \mathbb{R}^{D \times N}$. Firstly, the attention weights are estimated for each residue and subsystem as before $\mathbf{a}_1 = [\hat{\mathbf{A}}^1, ..., \hat{\mathbf{A}}^N]$.

Since residues could be negligible for all subsystems, a dummy system is added which has a constant attention weight $\mathbf{c} \in \mathbb{R}^{D \times 1}$ for all residues $\mathbf{a}_2 = [\mathbf{a}_1, \mathbf{c}]$.

In order to let the subsystem compete for the residues, i.e. the subsystems should fo-

cus on disjoint subareas, they are normalized along the subsystem axis $\mathbf{a}_3 = \frac{\mathbf{a}_2}{\sum_n^N \mathbf{a}_{2n}}$. The final mask can be retrieved by removing the dummy system in a final step $\mathbf{a}_3 = [\mathbf{A}, \bar{\mathbf{c}}]$.

3.7. Validation

After training a model, it is important to test its capability to perform on unseen data. As long as the new data is drawn from the same probability distribution as the training data, there should not be a significant drop in performance. However, since the available data is limited the model might overfit to the data seen during training, hindering its performance on the hold out data. Therefore, a common approach is to split the available data threefold: training, validation, and test set. The training data is used to optimize for the parameters of the model. The validation set might be used to optimize for hyperparameters, e.g. architecture or as a stopping criteria to prevent the model to overfit. The test set solely serves as a final performance check of the model [86, 87].
For validation and testing the VAMP-E or any VAMP-r validation score might be used.

3.7.1. Chapman-Kolmogorov equation

Instead of testing the model on the variational principle, two other tests have been established for Markov state models. Both of them origin from the Chapman-Kolmogorov (CK) equation, which can be exploited to test the Markovian assumption. It states, that if the Markovian property holds:

$$\mathbf{T}(n\tau) = \mathbf{T}^n(\tau), \tag{3.62}$$

for any $n > 1$, where $\mathbf{T}(\tau')$ indicates the transition matrix estimated at lag time τ'. The equation formalizes the fact, that estimating the probability to jump between states i, j at a lag time $n\tau$ should result in the the same probability as estimating the transition probabilities at lag time τ and calculating the probability to end in state j within n jumps starting in state i while being ignorant which intermediate states are visited.
Since we have limited data and any Markovian model of MD can be only approximate [12, 88], Eq. 3.62 can only be fulfilled within statistical uncertainty.

The final procedure involves defining a base time lag τ_{msm}, where the model is expected to be approximately Markovian, and computing the transition matrix $\mathbf{T}(\tau_{\text{msm}})$. During the estimation of the transition matrix $\mathbf{T}(n\tau_{\text{msm}})$ for different n the definition of states $\boldsymbol{\chi}(\mathbf{x}_t)$ remains fixed. However, $\mathbf{q}$ for the deep MSM and $\mathbf{u}$ and $\mathbf{S}$ for the reversible methods are retrained for each time lag.
By using different data splits in a cross-validation setting an statistical error can be estimated to test the CK-equation [89].

3.7.2. Implied timescales

In order to find a good candidate for τ_{msm} the second test was established. By plugging the eigenvalue decomposition into the CK-Eq. 3.62, a lag time dependence of the eigenvalues λ_i of the transition matrix can be established:

$$|\lambda_i(\tau)| = \exp\left(\frac{-\tau}{t_i}\right) \tag{3.63}$$

The eigenvalues decay with a characteristic timescale t_i [90].
Since the timescales should be independent of the chosen lag time, we can choose τ_{msm}, where the timescales are approximately constant with respect to the lag time. Afterwards, we can proceed with the CK-test.

3.7.3. Eigenfunctions of the Markov state model

Since $\mathbf{T}$ is a transition matrix as known from an ordinary Markov state model, we can study the eigenfunctions of it to learn about its metastable sets and the processes connecting them [12]. The relaxation times are given by the timescales defined above. The corresponding eigenfunctions $\boldsymbol{\varphi}(\mathbf{x})$ can be computed as:

$$\boldsymbol{\varphi}(\mathbf{x}) = \mathbf{R}\boldsymbol{\chi}(\mathbf{x}), \tag{3.64}$$

where $\mathbf{R}$ are the right eigenvectors of $\mathbf{T}$. These eigenfunctions and eigenvalues are only guaranteed to be real for the reversible deep MSM.
In the case of a VAMPnet, the state vector is propagated by the Koopman matrix $\mathbf{K}_\chi$ and the same analysis can be conducted for it.
The Koopman matrix $\mathbf{K}_\chi$ is given by:

$$\mathbf{K}_\chi = \mathbf{C}_{00}^{-1}\mathbf{C}_{01} = \mathbf{U}\mathbf{K}\mathbf{V}^{-1}. \tag{3.65}$$

If the left and right singular functions are equal $\mathbf{U} = \mathbf{V}$, they coincide with the eigenfunctions and the singular values equal the eigenvalues. Banisch and Koltai [91] connect the singular functions to coherent sets. Meaning that, if the singular functions are equal, the coherent set remains unchanged within a time step, i.e. it is a metastable set.
This is in particular the case if the underlying process is reversible and the data present in equilibrium. So if the system is simulated in thermal equilibrium and sampled close to the equilibrium distribution, real eigenfunctions and eigenvalues are expected independent of the chosen model.
The presented results in the next chapters have been published as part of the mentioned papers. Here, they are recollected to bring them into a common context and outline the differences and applicability of the proposed methods.

4. Vampnets: Deep learning of molecular kinetics

The results for the VAMPnet method have been published in:

> Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. "Vampnets: Deep learning of molecular kinetics". *Nature Communications* , 9:5, 2018. DOI: 10.1038/s41467-017-02388-1.

Part of the text and illustrations have been adopted unchanged or only slightly changed to fit into this context. Reprinted from the stated paper with the permission of Springer Nature.
Andreas Mardt and Luca Pasquali contributed equally to this work. In particular the contributions of the authors were as follows: Hao Wu and Frank Noé conceived the project and developed the theory. Andreas Mardt implemented the model. Andreas Mardt and Luca Pasquali performed research. They both trained the models, and analyzed them by visualizing the data. All contributors wrote the paper.

4.1. Results

Overview We use VAMPnets to learn molecular kinetics from simulation data of a range of model systems. We will first apply it to benchmark systems where reference solutions are well established. However, to show the outperforming or competitive performance on a larger system we conducted a comparison of our method with standard MSM analysis on the NTL9 dataset [92].

4.1.1. Asymmetric double well potential

We first model the kinetics of a bistable one-dimensional process, simulated by Brownian dynamics (cf. Sec. 4.2.4) in an asymmetric double-well potential (Fig. 4.1 a). A trajectory of $50,000$ time steps is generated. Three-layer VAMPnets are set up with 1-5-10-5 nodes representing $\boldsymbol{\chi}(x_t) = \boldsymbol{\eta}(x_t)$. The single input node of $\boldsymbol{\eta}$ is given by 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. 4.1 b), which is known to be important for the accuracy of a Markov state model [12, 88]. This placement minimizes the Koopman approximation error, as seen by comparing the dominant Koopman eigenfunction (Eq. 3.64) 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 [12] for details. The implied timescale and Chapman-Kolmogorov tests (Eqs. 3.63 and 3.62) confirm that the kinetic model learned by the VAMPnet successfully predicts the long-time kinetics (Fig. 4.1 c, d).

4.1.2. 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. 4.2) using the simple protein folding model defined by the potential

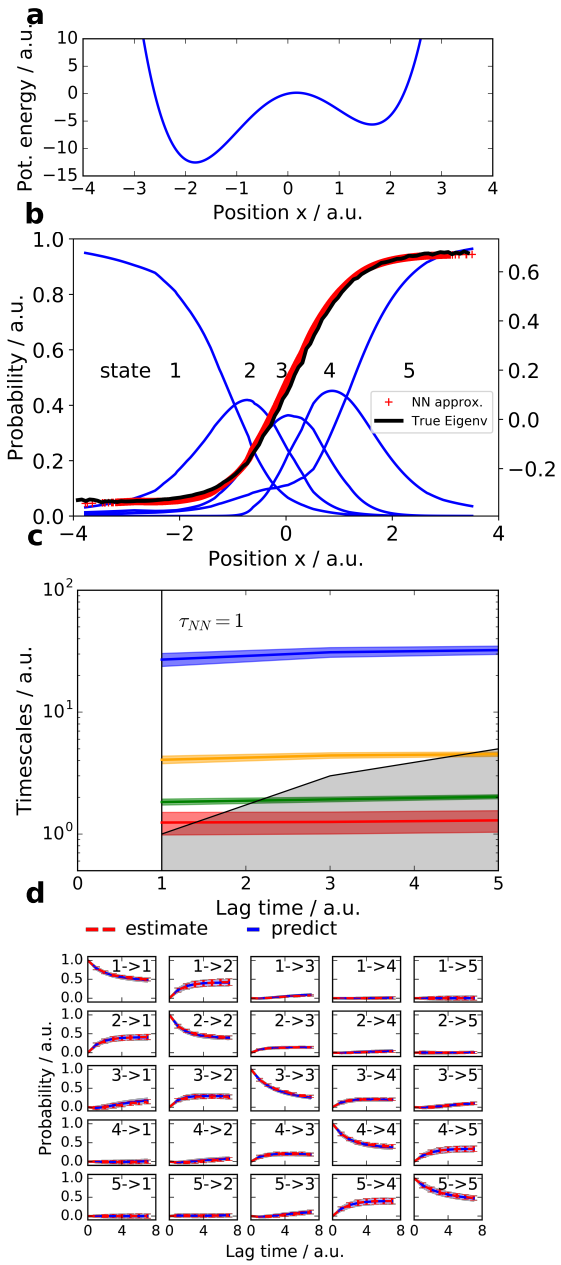

Figure 4.1.: Approximation of the slow transition in a bistable potential. (**a**) Potential energy function $U(x) = x^4 - 6x^2 + 2x$. (**b**) Eigenvector of the slowest process calculated by direct numerical approximation (black) and approximated by a VAMPnet with five output nodes (red). Activation 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. Panels (c) and (d) report 95% confidence interval error bars over 100 training runs.

energy function (Fig. 4.2 a):

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

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

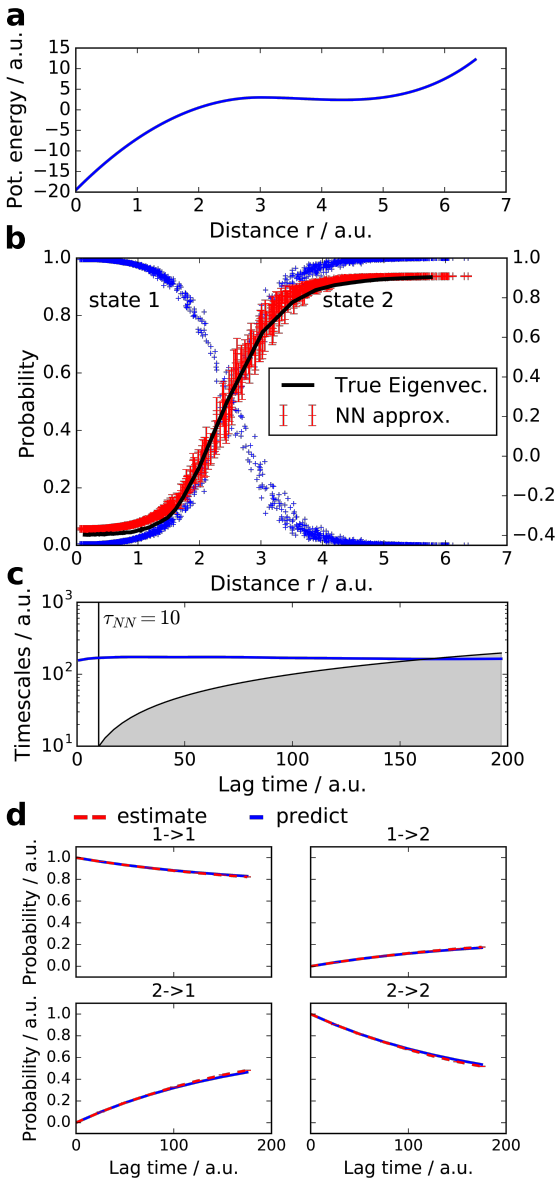

Figure 4.2.: VAMPnet model of a simple protein folding model. (**a**) Potential energy as a function of the radial position r. (**b**) Eigenvector of the slowest process calculated by direct numerical approximation on the radial position r (black) and approximated by the neural network build on the 5D-coordinates with two output nodes with $\tau = 10$ (red). Reported here is the mean and one standard deviation for the neural network over 100 runs. Activation of the two *softmax* output nodes define the state membership probabilities (blue). (**c**) Relaxation timescales computed from the Koopman model using the neural network transformation. (**d**) Chapman-Kolmogorov test comparing long-time predictions of the Koopman model estimated at $\tau = 10$ and estimates at longer lag times.

configurations available on a five-dimensional hypersphere grows dramatically with r. Thus, the dynamics are bistable along the reaction coordinate r. Four-layer NN with 5-32-16-8-2 nodes was 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 intersecting transition region (Fig. 4.2 b). 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. (3.64) with the eigenvector computed from a high-resolution MSM

built on the r coordinate (Fig. 4.2 b). This demonstrates that the network can learn the nonlinear reaction coordinate mapping $r = |\mathbf{x}|$ based only on maximizing the variational score Eq. 2.14. Furthermore, the implied timescales and the CK-test indicate that the network model predicts the long-time kinetics almost perfectly (Fig. 4.2 c, d).

4.1.3. 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 nanosecond MD trajectory generated in Ref. [93] (MD setup described there) serves as a dataset. The solute coordinates were stored every picosecond, resulting in $250,000$ configurations that are all aligned on the first frame using minimal RMSD fit to remove global translation and rotation. The NN uses the three-dimensional coordinates of the 10 heavy atoms as input, $(x_1, y_1, z_1, ..., x_{10}, y_{10}, z_{10})$, and the network is trained using time lag $\tau = 40\,\mathrm{ps}$. Different numbers of output states and layer depths are considered.

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. 4.3 b). The implied timescales indicate that given the coordinate transformation found by the network, the two slowest timescales are converged at lag time $\tau = 50\,\mathrm{ps}$ or larger (Fig. 4.3 c). Thus we estimated a Koopman model at $\tau = 50\,\mathrm{ps}$, whose Markov transition probability matrix is depicted in Fig. 4.3 d. Note that transition probabilities between state pairs $1 \leftrightarrow 4$ and $2 \leftrightarrow 3$ are important for the correct kinetics at $\tau = 50\,\mathrm{ps}$, but the actual trajectories typically pass via the directly adjacent intermediate states. The model performs excellently in the CK-Test (Fig. 4.3 e).

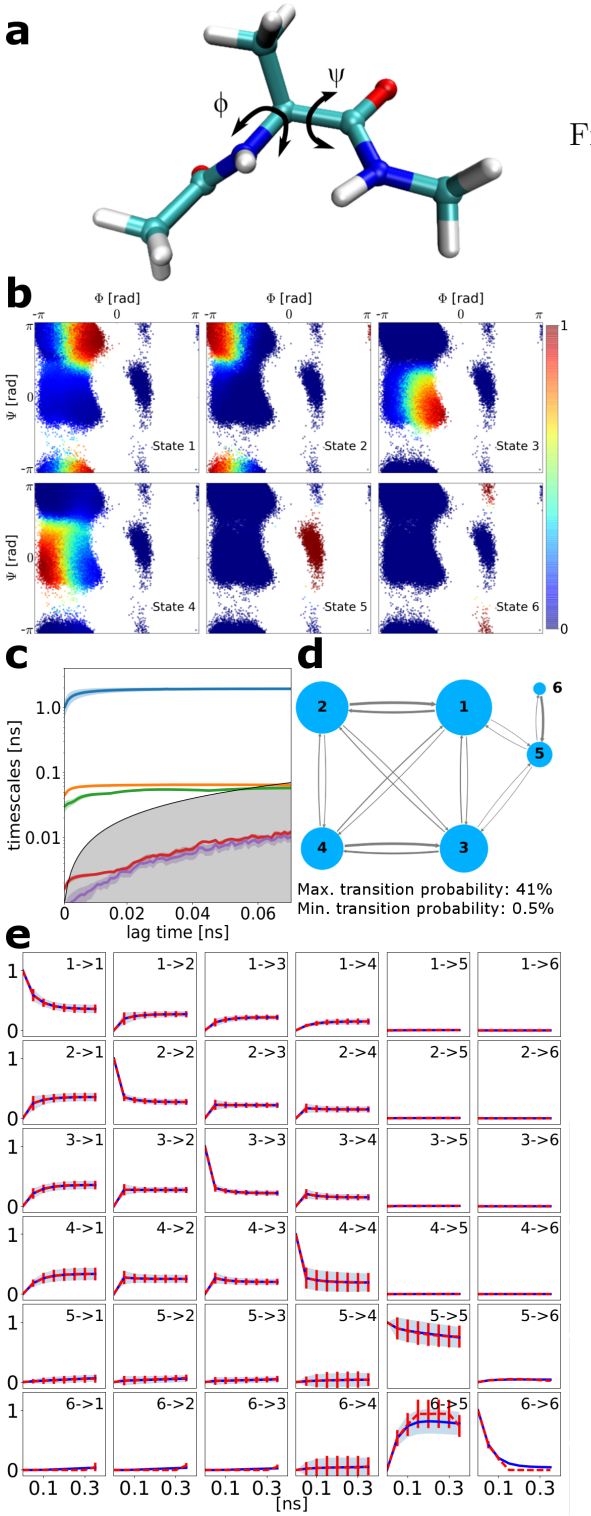

Figure 4.3.: VAMPnet kinetic model of alanine dipeptide.
(**a**) Structure of alanine dipeptide. The main coordinates describing the slow transitions are the backbone torsion angles ϕ and ψ, however the neural network inputs are only the Cartesian coordinates of heavy atoms.
(**b**) Assignment of all 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. Panels (c) and (e) report 95% confidence interval error bars over 100 training runs excluding failed runs (see text).

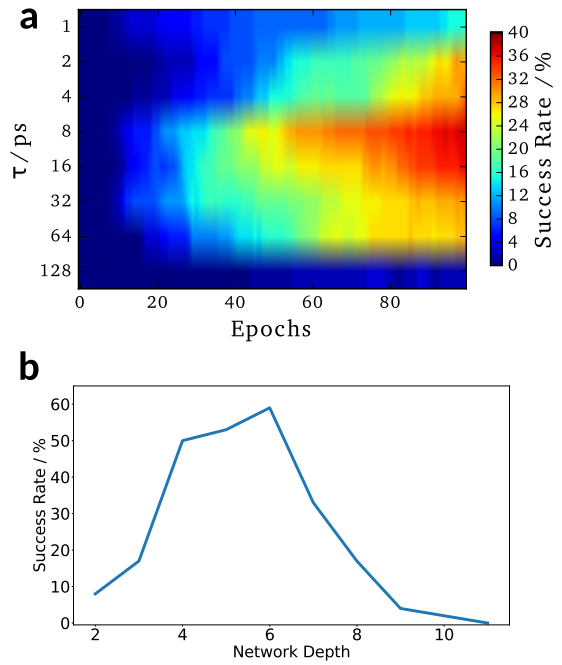

Figure 4.4.: Training success rate in alanine dipeptide VAMPnets with six output states as a function of hyperparameters. Training success rate is defined as the fraction of optimizations of the network to find all three slow processes (see text for details).
(**a**) Training success rate in a five-layer network as a function of the lag time τ and the number of optimization epochs.
(**b**) Training success rate at lag time $\tau = 8$ ps and after 100 epochs as a function of the network depth.

4.1.4. 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. 4.3 are reported for successful runs in this definition. There is a range of τ values from 4 to 32 picoseconds where the training succeeds with a significant probability (Fig. 4.4 a). 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. 4.3 b, state 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 propagators [12]. 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 (cf. Sec. 4.2). On the other hand, deeper networks are more difficult to train. As seen in Fig. 4.4 b, a high success rate is found for four to seven layers.

Next, we studied the dependency of the network-based discretization as a function

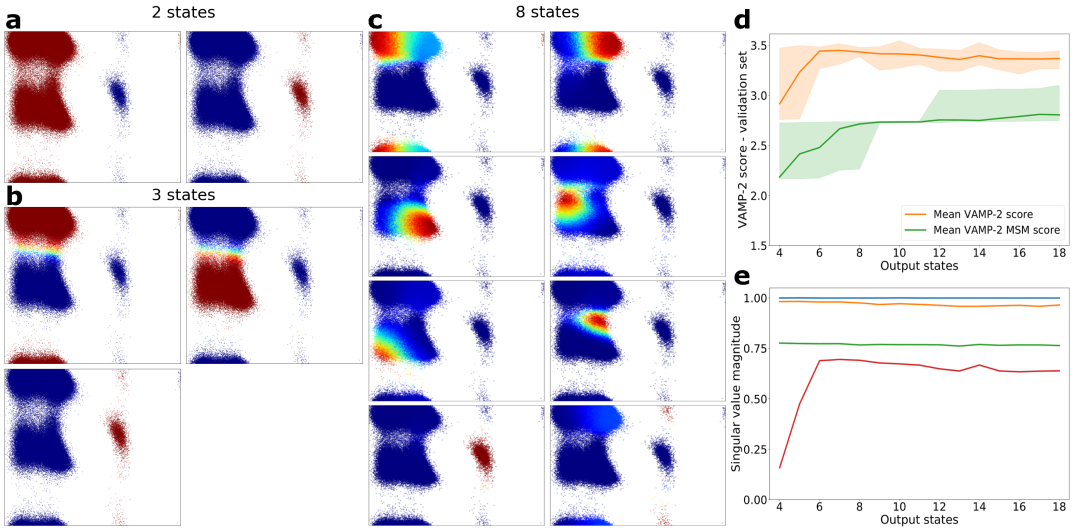

Figure 4.5.: Kinetic model of alanine dipeptide as a function of the number of output states. (**a-c**) Assignment of input coordinates, plotted as a function of ϕ and ψ, to two, three, and eight output states. Color corresponds to activation of the respective output neuron, indicating the membership probability to this state (cf. Fig. 4.3 b). (**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 panel (d).

of the number of output nodes (Fig. 4.5 a-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. 4.5 a). 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. 4.5 b). For 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. 4.5 c). We chose the lag time depending on the number of output nodes of the network, using $\tau = 200\,\mathrm{ps}$ for two output nodes, $\tau = 60\,\mathrm{ps}$ for three output nodes, and $\tau = 1\,\mathrm{ps}$ 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 (3 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. 4.5 d-e).

For comparison, we investigated how a standard MSM would perform as a function of the number of states (Fig. 4.5 d). For a fair 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 variance [94], 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 less than 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.

4.1.5. 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 30-10-3-3-2-6 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, Fig. 4.6 a, 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 (Fig. 4.6 c), and then show where these samples end up in the space of the bottleneck node activations (Fig. 4.6 d). It is apparent that the

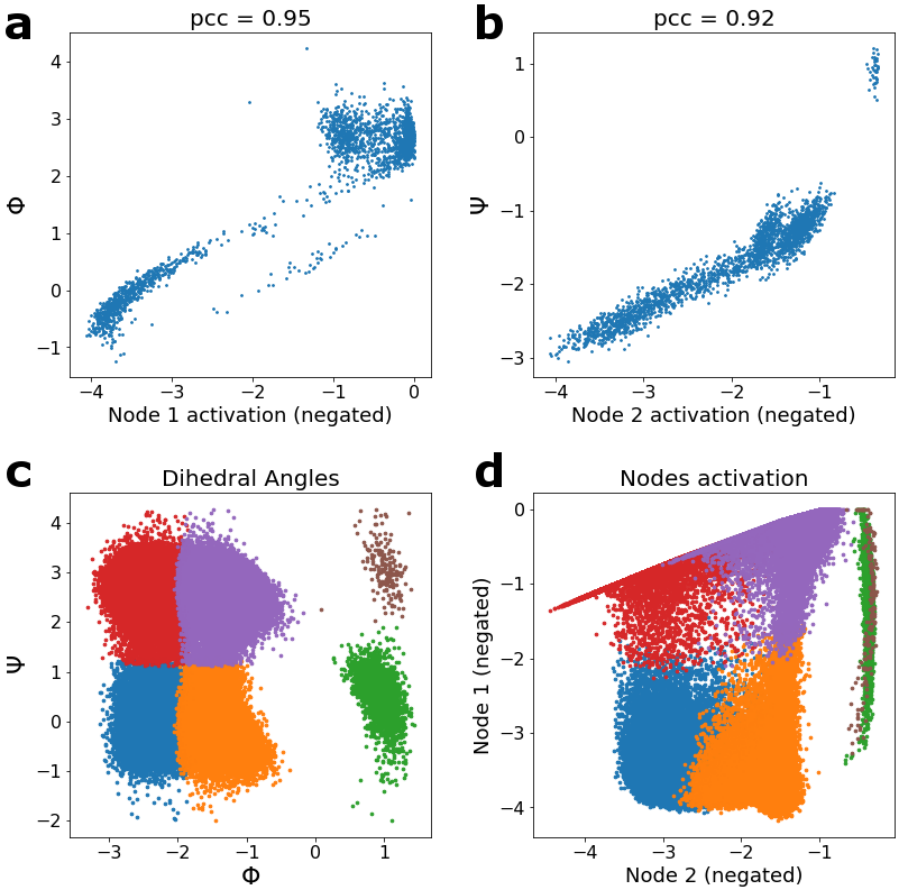

Figure 4.6.: Network with a bottleneck of two nodes implicitly learns the transformation from heavy atom positions to ϕ and ψ torsion angles. (**a, b**) Correlation of torsion angles and the activations of the bottleneck nodes. Pearson correlation coefficient (pcc) is shown above. (**c**) Samples plotted as a function of ϕ and ψ angles, colored according to six clusters found by a optimized network with six output states and no bottleneck. (**d**) Samples with same colors, plotted on the activations of the bottleneck nodes.

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.

4.1.6. 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 supercomputer [92]. A five-layer VAMPnet was trained at lag time $\tau = 10\,\mathrm{ns}$ using $111,000$ time steps, uniformly sampled from a $1.11\,\mathrm{ms}$ 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. Fig. 4.7 a 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. 4.7 a, 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. 4.7 b-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.

4.2. Methods

4.2.1. Neural network structure

Each network lobe in Fig. 3.1 has a number of input nodes given by the data dimension. According to the VAMP variational principle, 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 Eq. 2.19. 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:

$$\frac{n_i}{n_{i+1}} = \left(\frac{n_{\text{in}}}{n_{\text{out}}}\right)^{1/d} \tag{4.1}$$

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, Fig. 4.4 b 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 400000, 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 training [61], and select hyper-parameters using the VAMP-2 validation score.

4.2.2. Neural network hyperparameters

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 higher than 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 16000. 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 k and the hidden layers are selected, which determines the network structure as described above. Then, a lag time is chosen in which the largest k singular values (corresponding to the $n-1$ slowest processes) can be trained consistently.

4.2.3. 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. In order to ensure robustness of the results, we performed 100 network optimization runs for each problem. In each run, the dataset 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. Hyperparameter optimization was done using the validation score averaged over the remaining runs. Figures report training or validation mean and 95% confidence intervals.

4.2.4. 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:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t - \Delta t \frac{\nabla U(\mathbf{x})}{kT} + \sqrt{2\Delta t D}\mathbf{w}_t, \tag{4.2}$$

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.

Code availability TICA, k-means and MSM analyses were conducted with PyEMMA version 2.4, freely available at pyemma.org. VAMPnets are implemented using the freely available packages *keras [95]* with *tensorflow-gpu [96]* as a backend. The code can be obtained at `https://github.com/markovmodel/deeptime`. A newer code version is implemented with *PyTorch* [97] in the deeptime package [98].

Data availability Data for NTL9 can be requested from the autors of [92]. Data for all other examples is available at `https://github.com/markovmodel/deeptime`.

4.3. 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 hand-crafted 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 variational approach for Markov processes which 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 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*, *PyTorch*, or *keras*. Despite their benefits, VAMPnets still miss many of the benefits that come with extensions developed for 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 [99–104], Augmented Markov models that combine simulation data with experimental observation data [43], and statistical error estimators developed for MSMs [105–107]. 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 towards 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 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 filters [108] or different types of transfer functions. Suitably chosen convolutions, as in [109] may also lead to learned feature transformations that are transferable within a given class of molecules.

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

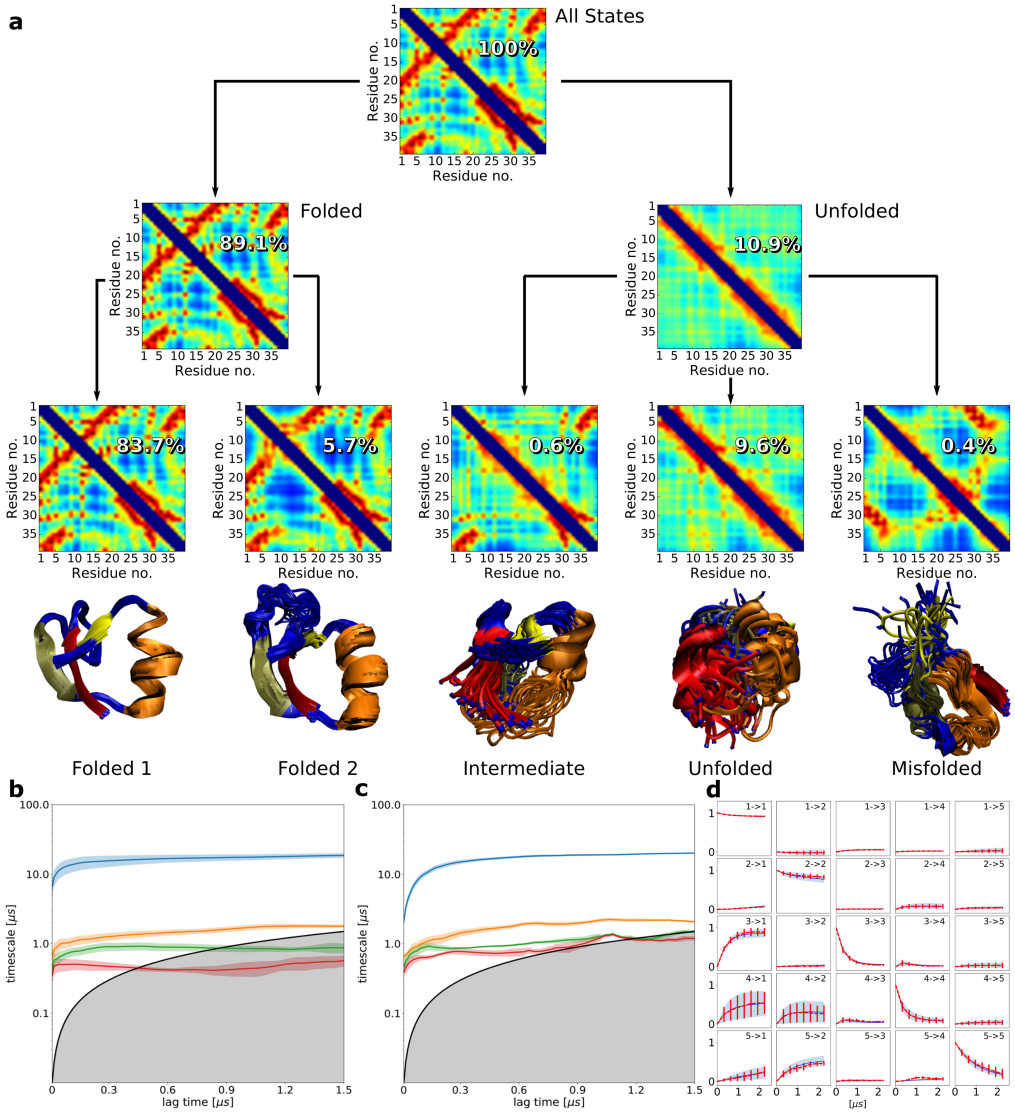

Figure 4.7.: VAMPnet results of NTL9 folding kinetics. (**a**) Hierarchical decomposition of the NTL9 protein state space by a network with two and five output nodes. Mean contact maps are shown for all MD samples grouped by the network, along with the fraction of samples in that group. 3D structures 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. Panels (b), (c) and (d) report 95% confidence interval error bars over 100 training runs.

5. Deep Generative Markov State Models

The results for the deep MSM method have been published in the following proceedings of the Neurips 2018 conference:

> Hao Wu, Andreas Mardt, Luca Pasquali, and Frank Noé. "Deep Generative Markov State Models". *Advances in Neural Information Processing Systems* , 31, 2018. URL: https://papers.nips.cc/.

Part of the text and illustrations have been adopted unchanged or only slightly changed to fit into this context. Reprinted from the stated paper with the permission of Curran Associates, Inc..
Hao Wu, Andreas Mardt, and Luca Pasquali contributed equally to this work. In particular the contributions of the authors were as follows: Hao Wu and Frank Noé conceived the project and developed the theory. Andreas Mardt and Luca Pasquali conducted the research, where Andreas Mardt focused on the Prinz and alanine example with the whole dataset and Luca Pasquali on the alanine example removing data and try to recover the distributions. All contributors wrote the paper.

5.1. Results

Overview Below we establish our framework by applying it to two well-defined benchmark systems that exhibit metastable stochastic dynamics. We validate the stationary distribution and kinetics by computing $\boldsymbol{\chi}(\mathbf{x})$, $\mathbf{q}(\mathbf{y})$, the stationary distribution $\mu(\mathbf{y})$ and the relaxation times $t_i(\tau)$ and comparing them with reference solutions. We will also test the abilities of a deep generative MSM to generate physically valid molecular configurations.

5.1.1. Diffusion in Prinz potential

We first apply our framework to the time-discretized diffusion process $x_{t+\Delta t} = -\Delta t\, \nabla V(x_t) + \sqrt{2\Delta t}\dot{\omega}_t$ with $\Delta t = 0.01$ in the Prinz potential $V(x_t)$ introduced in [12] (Fig. 5.1 a). For this system we know exact results for benchmarking: the stationary distribution and relaxation timescales (black lines in Fig. 5.1 b, c) and the transition density (Fig. 5.1 d). We simulate trajectories of lengths $250,000$ and $125,000$ time steps for training and validation, respectively. For all methods, we repeat the data generation and model estimation process 10 times and compute mean and standard deviations for all quantities of interest, which thus represent the mean and variance of the estimators.

The functions $\boldsymbol{\chi}$, $\boldsymbol{\gamma}$ and $\mathbf{G}$ are represented with densely connected neural networks. The details of the architecture and the training procedure can be found in the methods section.

We compare the deep MSMs and deep generative MSMs with standard MSMs using four or ten states obtained with k-means clustering. Note that standard MSMs do not directly operate on configuration space. When using an MSM, the transition density (Eq. 3.1) is thus simulated by:

$$\mathbf{x}_t \xrightarrow{\boldsymbol{\chi}(\mathbf{x}_t)} i \xrightarrow{\sim \mathbf{T}_{i,*}} j \xrightarrow{\sim \rho_j(\mathbf{y})} \mathbf{x}_{t+\tau}, \tag{5.1}$$

i.e., we find the cluster i associated with a configuration $\mathbf{x}_t$, which is deterministic for regular MSMs, then sample the cluster j at the next time-step, and sample from the conditional distribution of configurations in cluster j to generate $\mathbf{x}_{t+\tau}$.

Both deep MSMs trained with the maximum likelihood (ML) method and standard MSMs can reproduce the stationary distribution within statistical uncertainty (Fig. 5.1 b). For long lag times τ, all methods converge from below to the correct

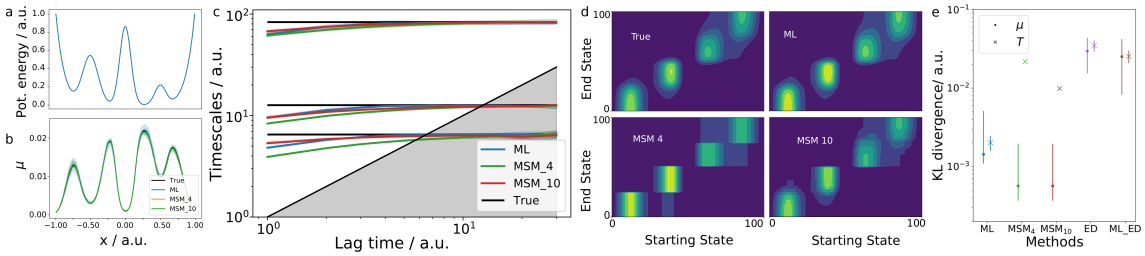

Figure 5.1.: Performance of deep versus standard MSMs for diffusion in the Prinz Potential. (**a**) Potential energy as a function of position $\mathbf{x}$. (**b**) Stationary distribution estimates of all methods with the exact distribution (black). (**c**) Implied timescales of the Prinz potential compared to the real ones (black line). (**d**) True transition density and approximations using maximum likelihood (ML) deep MSM, four and ten state MSMs. (**e**) KL-divergence of the stationary and transition distributions with respect to the true ones for all presented methods (also deep generative MSM).

relaxation timescales (Fig. 5.1 c), as expected from theory [12, 110]. When using equally many states (here: four), the deep MSM has a much lower bias in the relaxation timescales than the standard MSM. This is expected from approximation theory, as the deep MSMs represents the four metastable states with a meaningful, smooth membership functions $\boldsymbol{\chi}(\mathbf{x}_t)$, while the four-state MSM cuts the memberships hard at boundaries with low sample density. When increasing the number of metastable states, the bias of all estimators will reduce. An MSM with ten states is needed to perform approximately equal to a four-state deep MSM (Fig. 5.1 c). All subsequent analyses use a lag time of $\tau = 5$.

The deep MSM generates a transition density that is very similar to the exact density, while the MSM transition densities are coarse-grained by virtue of the fact that $\boldsymbol{\chi}(\mathbf{x}_t)$ performs a hard clustering in an MSM (Fig. 5.1 d). This impression is confirmed when computing the Kullback-Leibler divergence of the distributions (Fig. 5.1 e).

Encouraged by the accurate results of deep MSMs, we now train deep generative MSMs, either by training both the $\boldsymbol{\chi} = \boldsymbol{\eta}$ and $\mathbf{G}$ networks by minimizing the energy distance (ED), or by taking $\boldsymbol{\chi}$ from a ML-trained deep MSM and only training the $\mathbf{G}$ network by minimizing the energy distance (ML-ED). The stationary densities, relaxation timescales and transition densities can still be approximated in these settings, although the deep generative MSMs exhibit larger statistical fluctuations than the deep MSMs (Fig. 5.2). ML-ED appears to perform slightly better than ED

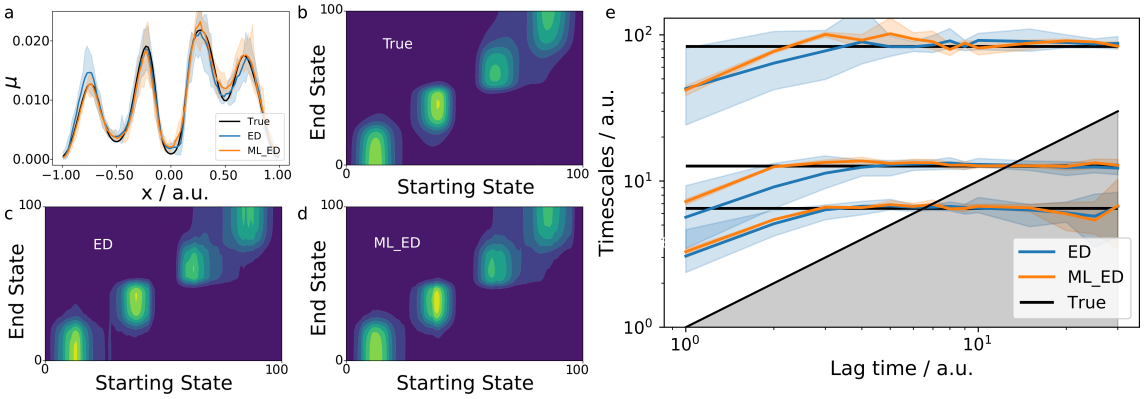

Figure 5.2.: Performance of deep generative MSMs for diffusion in the Prinz Potential. Comparison between exact reference (black), deep generative MSMs estimated using only energy distance (ED) or combined ML-ED training. (**a**) Stationary distribution. (**b-d**) Transition densities. (**e**) Relaxation timescales.

alone, likely because reusing $\boldsymbol{\chi}$ from the ML training makes the problem of training the generator easier.

For a one-dimensional example like the Prinz potential, learning a generative model does not provide any added value, as the distributions can be well approximated by the empirical distributions. The fact that we can still get approximately correct results for stationary, kinetics and dynamical properties encourages us to use deep generative MSMs for a higher-dimensional example, where the generation of configurations is a hard problem.

5.1.2. Alanine dipeptide

We use explicit-solvent MD simulations of alanine dipeptide as a second example. Our aim is to learn stationary and kinetic properties, but especially to learn a generative model that generates genuinely novel but physically meaningful configurations. One 250 ns trajectory with a storage interval of 1 ps is used and split 80%/20% for training and validation – see [77] for details of the simulation setup. We characterize all structures by the three-dimensional Cartesian coordinates of the heavy atoms, resulting in a 30 dimensional configuration space. While we do not have exact results for alanine dipeptide, the system is small enough and well enough sampled, such that high-quality estimates of stationary and kinetic properties can be obtained from a very fine MSM [12]. We therefore define an MSM built on 400 equally sized grid areas in the (ϕ, ψ)-plane as a reference at a lag time of $\tau = 25\,\mathrm{ps}$ that has been

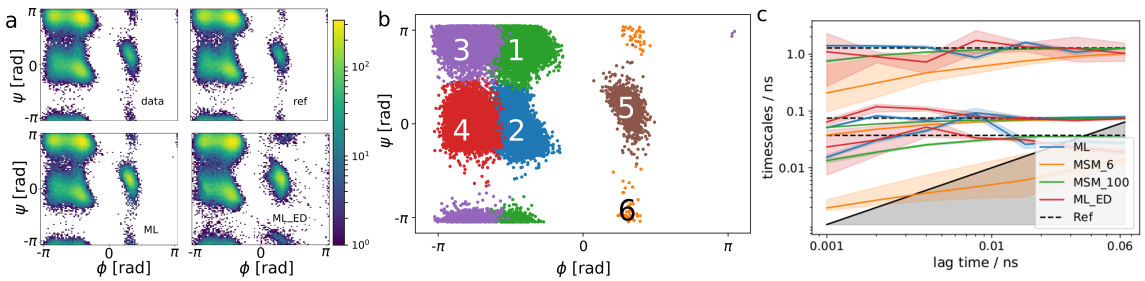

Figure 5.3.: Performance of deep MSMs and deep generative MSMs versus standard MSMs on the alanine dipeptide simulation trajectory. (**a**) Data distribution and stationary distributions from reference MSM, deep MSM, and deep generative MSM. (**b**) State classification by deep MSM (**c**) Relaxation timescales.

validated by established methods [12].

Neural network and training details are again found at the git repository and in the methods section.

For comparison with deep MSMs, we build two standard MSMs following a state of the art protocol: we transform input configurations with a kinetic map preserving 95% of the cumulative kinetic variance [94], followed by k-means clustering, where $k = 6$ and $k = 100$ are used.

Deep MSMs trained with ML method approximate the stationary distribution very well (Fig. 5.3 a). The reference MSM assigns a slightly lower weight to the lowest-populated state 6, but otherwise the data, reference distribution and deep MSM distribution are visually indistinguishable. The relaxation timescales estimated by a six-state deep MSM are significantly better than with six-state standard MSMs. MSMs with 100 states have a similar performance as the deep MSMs but this comes at the cost of a model with a much larger latent space.

Finally, we test deep generative MSMs for alanine dipeptide where $\boldsymbol{\chi}$ is trained with the ML method and the generator is then trained using ED (ML-ED). The stationary distribution generated by simulating the model recursively results in a stationary distribution which is very similar to the reference distribution in states 1-4 with small ϕ values (Fig. 5.3 a). States number 5 and 6 with large ϕ values are captured, but their shapes and weights are somewhat distorted (Fig. 5.3 a). The one-step transition densities predicted by the generator are high quality for all states (Fig. 5.4), thus the differences observed for the stationary distribution must come from small errors made in the transitions between metastable states that are very rarely observed for states 5 and 6. These rare events result in poor training data for

the generator. However, the deep generative MSM approximates the kinetics well within the uncertainty that is mostly due to estimator variance (Fig. 5.3 c).

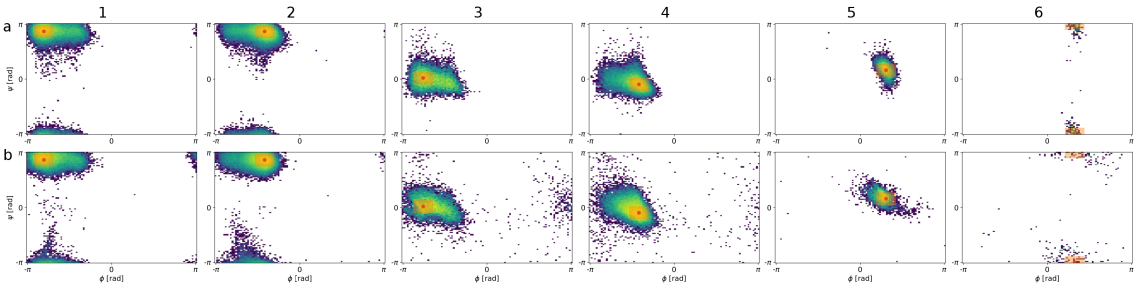

Figure 5.4.: Conditional transition distributions for alanine dipeptide starting from different metastable states. The starting distribution are sampled from the empirical distribution in the yellow region around the red point. (**a**) Distribution sampled from the MD simulation. (**b**) Distribution generated by the deep generative MSM.

Now we ask whether deep generative MSMs can sample valid structures in the 30-dimensional configuration space, i.e., if the placement of atoms is physically meaningful. As we generate configurations in Cartesian space, we first check if the internal coordinates are physically viable by comparing all bond lengths and angles between real MD data and generated trajectories (Fig. 5.5). The true bond lengths and angles are almost perfectly Gaussian distributed, and we thus normalize them by shifting each distribution to a mean of 0 and scaling it to have standard deviation 1, which results in all reference distributions to collapse to a normal distribution (Fig. 5.5 a, c). We normalize the generated distribution with the mean and standard distribution of the true data. Although there are clear differences (Fig. 5.5 b, d), these distributions are very encouraging. Bonds and angles are very stiff degrees of freedom, and the fact that most differences in mean and standard deviation are small when compared to the true fluctuation width means that the generated structures are close to physically accurate and could be refined by little additional MD simulation effort.

Finally, we perform an experiment to test whether the deep generative MSM is able to generate genuinely new configurations that do exist for alanine dipeptide but have not been seen in the training data. In other words, can the generator "extrapolate" in a meaningful way? This is a fundamental question, because simulating MD is exorbitantly expensive, with each simulation time step being computationally

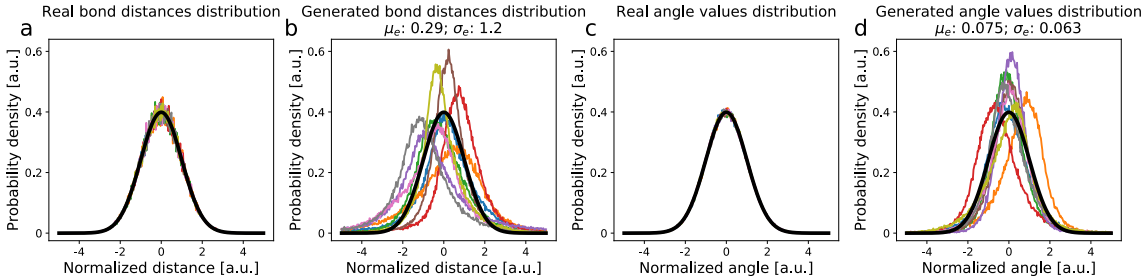

Figure 5.5.: (**a**, **b**) Normalized bond and (**c**, **d**) angle distributions of alanine dipeptide compared to Gaussian normal distribution (black). (**a**, **c**) True MD data. (**b**, **d**) Data from trajectories generated by deep generative MSMs.

expensive but progressing time only of the order of 10^{-15} seconds, while often total simulation timescales of 10^{-3} seconds or longer are needed. A deep generative MSM that makes leaps of length τ – orders of magnitude larger than the MD simulation time-step – and has even a small chance of generating new and meaningful structures would be extremely valuable to discover new states and thereby accelerate MD sampling.

To test this ability, we conduct six experiments, in each of which we remove all data belonging to one of the six metastable states of alanine dipeptide (Fig. 5.6 a). We train a deep generative MSM with each of these datasets separately, and simulate it to predict the stationary distribution (Fig. 5.6 b). While the generated stationary distributions are skewed and the shape of the distribution in the (ϕ, ψ) range with missing-data are not quantitatively predicted, the deep generative MSMs do indeed predict configurations where no training data was present (Fig. 5.6 b). Surprisingly, the quality of most of these configurations is high (Fig. 5.6 c). While the structures of the two low-populated states 5-6 do not look realistic, each of the metastable states 1-4 are generated with high quality, as shown by the overlap of a real MD structure and the 100 most similar generated structures (Fig. 5.6 c).

5.2. Methods

Network architecture and training procedure All neural networks representing the functions $\boldsymbol{\chi} = \boldsymbol{\eta}$, $\boldsymbol{\gamma}$ and $\mathbf{G}$ for the Prinz potential are using 64 nodes in all 4 hidden layers and batch normalization after each layer [63]. Rectified linear activation functions (ReLUs) are used, except for the output layer of $\boldsymbol{\eta}$ which uses

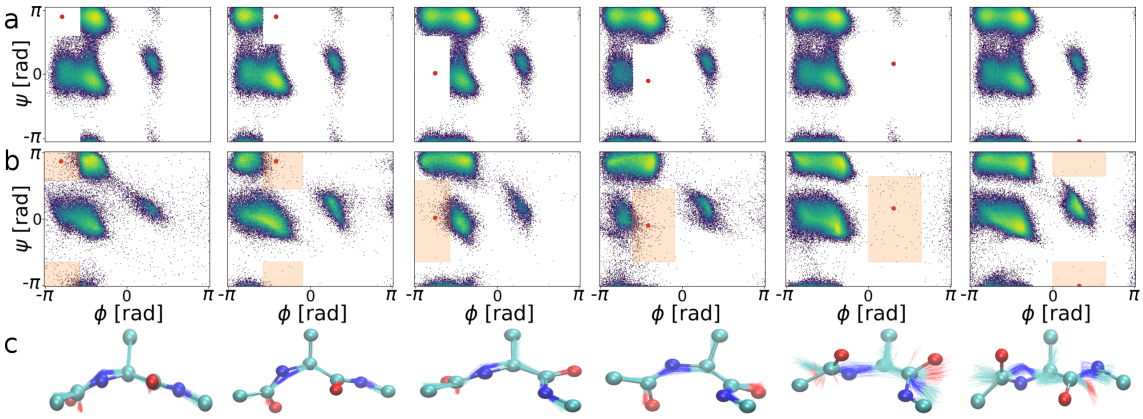

Figure 5.6.: Deep generative MSMs can generate physically realistic structures in areas that were not included in the training data. (**a**) Distribution of training data. (**b**) Generated stationary distribution. (**c**) Representative "real" molecular configuration (from MD simulation) in each of the metastable states (sticks and balls), and the 100 closest configurations generated by the deep generative MSM (lines).

softmax and the output layer of $\mathbf{G}$ which has a linear activation function. Both $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ have 4 output nodes, and $\mathbf{G}$ receives a four-dimensional 1-hot-encoding of the metastable state plus a four-dimensional noise vector as inputs. Optimization is done using Adam [111], with early stopping checking if the validation score is not increasing over 5 epochs. The learning rate for the training of $\boldsymbol{\eta}$, $\boldsymbol{\gamma}$ is $\lambda = 10^{-3}$, and for $\mathbf{G}$ $\lambda = 10^{-5}$ with a batchsize of 100. We are using a time-lag of $\tau = 5$ frames.

For alanine dipeptide, $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ consist both of 3 residual blocks [112] built of 3 layers all having 100 nodes, with exponential linear units (ELUs) [60], and batch normalization for each layer. The output layer has 6 output nodes, where $\boldsymbol{\eta}$ uses a *softmax* activation function and $\boldsymbol{\gamma}$ a RELU [65], respectively. In order to find all slow processes, it was necessary to pre-train $\boldsymbol{\eta}$ with the VAMPnet method [77]. The generator $\mathbf{G}$ uses 6 noise inputs and a six-dimensional 1-hot-encoding of the metastable state and the ML-ED scheme. Networks are trained with Adam until the validation score converges with a learning rate of $\lambda = 10^{-5}$ for $\boldsymbol{\eta}$, $\boldsymbol{\gamma}$ using 8000 as batchsize and $\lambda = 10^{-4}$ for $\mathbf{G}$ using 1500 frames for a batch. All subsequent analyses that use a fixed lag time employ $\tau = 1\,\mathrm{ps}$.

Code availability The networks were implemented using *PyTorch* [97] and *tensorflow* [96]. For the full code and all details about the neural network architecture, hyper-parameters and training algorithm, please refer to

https://github.com/markovmodel/deep_gen_msm.

5.3. Discussion

In conclusion, deep MSMs provide high-quality models of the stationary and kinetic properties for stochastic dynamical systems such as MD simulations. In contrast to other high-quality models such as VAMPnets, the resulting model is truly probabilistic and can thus be physically interpreted and be used in a Bayesian framework. For the first time, it was shown that generating dynamical trajectories in a 30-dimensional molecular configuration space results in sampling of physically realistic molecular structures. While alanine dipeptide is a small system compared to proteins and other macromolecules that are of biological interest, our results demonstrate that efficient sampling of new molecular structures is possible with generative dynamic models, and improved methods can be built upon this. Future methods will especially need to address the difficulties of generating valid configurations in low-probability regimes, and it is likely that the energy distance used here for generator training needs to be revisited to achieve this goal.

Acknowledgements This work was funded by the European Research Commission (ERC CoG "ScaleCell"), Deutsche Forschungsgemeinschaft (CRC 1114/A04, Transregio 186/A12, NO 825/4–1, Dynlon P8), and the "1000-Talent Program of Young Scientists in China".

6. Deep reversible Koopman model

The results for the deep reversible Koopman models have been published in the Proceedings of Machine Learning Research as part of the Mathematical Scientific Machine Learning conference 2020:

> Andreas Mardt, Luca Pasquali, Frank Noé, and Hao Wu. "Deep learning Markov and Koopman models with physical constraints". *Mathematical and Scientific Machine Learning* , 107:451-475, PMLR, 2020. URL: http://proceedings.mlr.press/v107/mardt20a.html.

Part of the text and illustrations have been adopted unchanged or only slightly changed to fit into this context. Reprinted from the stated paper with the permission of PMLR.
Andreas Mardt is the single first author. In particular the contributions of the authors were as follows: Hao Wu and Frank Noé conceived the project and developed the theory. Andreas Mardt conducted the main research, where Luca Pasquali helped running experiments. All contributors wrote the paper.

6.1. Results

Symmetrized VAMPnet (SymVAMPnet)

We compare our reversible models to a previously proposed model [113, 114], which estimates a VAMPnet but additionally enforces reversibility by symmetrizing the correlation matrices entering the VAMP score as follows:

$$\begin{aligned}
\mathbf{C}_{00} &= \frac{1}{2}(\mathbb{E}[\boldsymbol{\chi}(\mathbf{x}_t)\boldsymbol{\chi}(\mathbf{x}_t)^T] + \mathbb{E}[\boldsymbol{\chi}(\mathbf{x}_{t+\tau})\boldsymbol{\chi}(\mathbf{x}_{t+\tau})^T]),\\
\mathbf{C}_{01} &= \frac{1}{2}(\mathbb{E}[\boldsymbol{\chi}(\mathbf{x}_t)\boldsymbol{\chi}(\mathbf{x}_{t+\tau})^T] + \mathbb{E}[\boldsymbol{\chi}(\mathbf{x}_{t+\tau})\boldsymbol{\chi}(\mathbf{x}_t)^T]),\\
\mathbf{C}_{11} &= \mathbf{C}_{00}.
\end{aligned}$$

In the limit of a long equilibrium simulation, this model is asymptotically unbiased, but it can be subject to a strong bias in the case of short simulations starting from a non-equilibrium distribution, which is the main application scenario of Markov modeling (cf. [114])

Overview Below we demonstrate our model by applying it to a time-discretized one-dimensional diffusion process $x_{t+\Delta t} = -\Delta t \nabla V(x_t) + \sqrt{2\Delta t}\omega_t$ in the Prinz potential $V(x)$ [12] (Fig. 6.1 a and same as in Sec. 5.1.1) with time step $\Delta t = 0.001$ and ω_t being standard normal random variables. When generating training data we save the state x every five timesteps. The neural network $\boldsymbol{\eta}$ representing $\boldsymbol{\chi}$ has one input node, receiving the current value of the x coordinate. We validate that when enforcing the nonnegativity (Eq. 3.18) and reversibility (Eq. 2.60) constraints, our models will result in a valid transition matrix and real eigenvalues respectively, even in the case of poorly sampled data. Furthermore, we show that our reversible models give unbiased results for implied timescales and equilibrium probabilities even when using non-equilibrium data for training, while a simple symmetrization of the correlation matrices (SymVAMPnet) does not. Finally, we study the ability of the proposed methods to approximate the exact eigenfunctions of the test system. We focus on the 1-D toy model system to demonstrate the performance of the estimator for a system where exact solutions are available. However, to show the outperforming or competitive performance on a larger system we conducted a comparison of our method with standard MSM analysis on the NTL9 dataset [92] (cf. Sec. 6.1.4).

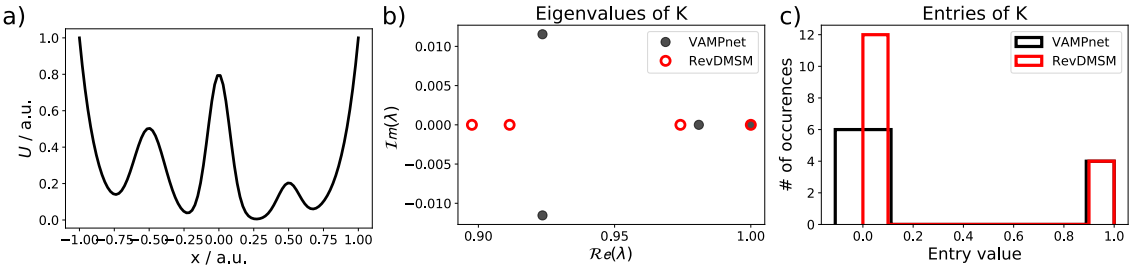

Figure 6.1.: Demonstration of incorporation of physical constraints: reversibility and non-negativity. The eigenvalues and the distribution of elements of the transition matrix $\mathbf{T}$ for the RevDMSM and of $\mathbf{K}_\chi$ for an unconstrained VAMPnet are shown trained on poorly sampled training data. (**a**) Potential energy profile of the Prinz model (**b**) Imaginary and real part of the eigenvalues of a model estimated with non-reversible VAMPnets and a RevDMSM. (**c**) Entries of the matrix $\mathbf{T}$ of these two models.

6.1.1. Reversible VAMPnets and reversible deep MSMs obtain transition matrices with real eigenvalues and nonnegative entries

To simulate an insufficiently sampled example, we created 1000 trajectories of 1 time step with the starting distribution as stated in Sec. 6.2 for training, validation, and test set, respectively. We train a regular VAMPnet and a reversible deep MSM (RevDMSM) on the training data with an early stopping given by the performance on the validation set and estimate the resulting Koopman matrix for the VAMPnet and the transition matrix for the RevDMSM on the test set with a fixed number of output nodes $k = 4$. Fig. 6.1 b) shows the resulting eigenvalues of these matrices and Fig. 6.1 c) the distribution of the entries. Using the non-reversible VAMPnet, the poor sampling leads to complex eigenvalues and negative entries for the Koopman matrix. Thus, we not only obtain a non-reversible model, but the Koopman matrix also does not correspond to a valid transition matrix. The RevDMSM model does not suffer from these shortcomings, nevertheless the constraints imposed on the model result in slightly lower eigenvalues, which can be expected since the constraints hinder the ability to approximate the eigenfunctions of the Koopman operator.

6.1.2. Reversible VAMPnets converge to unbiased timescales and state probabilities for biased training data

In furtherance of showing the necessity of new methods compared to the already introduced SymVAMPnets, we demonstrate the shortcomings of it in the case of systematically biased training data in Fig. 6.2. We restrict ourself to only compare the performance with reversible VAMPnets (RevVAMPnets), since both methods result in a reversible Koopman model. Thereby, a special focus lies on how well the two methods approximate the stationary probability of the four main states ($[(-\infty, -.5]$, $[-0.5, 0.]$, $[0., .5]$, $[.5, \infty)]$) and the estimated values for the timescales of the dynamical system. We use as benchmark the Prinz potential, using as training data a varying number of trajectories with fixed length of 11 frames, and a single trajectory with a varying number of frames; we chose a fixed length of 11 frames in order to estimate the timescales at $\tau = 10$. We test the convergence over an increasing number of trajectories of the two models using $10^2, 10^3, 10^4$ trajectories, respectively. This mimics the case of systematically biased training data, since the simulations are started from a non-equilibrium distribution, which results in training data sampled from a distribution different from the equilibrium distribution even in the case of infinitely many simulations. We also vary the trajectory length between $2{\cdot}10^3, 10^4, 5{\cdot}10^4$ frames, respectively. The true values of the timescales are numerical approximations by a transition matrix computed for a direct uniform 1000-state discretization of the x-axis for $2 \cdot 10^7$ frames [12], while the true state probabilities were calculated directly from the analytical expression of the potential.

The test of convergence in trajectory length shows how both methods converge to the true values of the system's timescales and state probabilities, as it is expected when the training data distribution converges to the stationary distribution (Fig. 6.2 a-d). The test of convergence in trajectory numbers shows how the RevVAMPnets method is able to approximate the real state probabilities and timescale values already with a small number of short trajectories within statistical uncertainty, and converges to a value consistently close to the real one when the number of trajectories used as training data increases; we did not observe this behavior for the SymVAMPnets, as this method is unable to recover the true dynamics and equilibrium distribution of the system when working with a heavily biased sampling (Fig. 6.2 e-h), which results in a first timescale nearly a factor 3 too low.

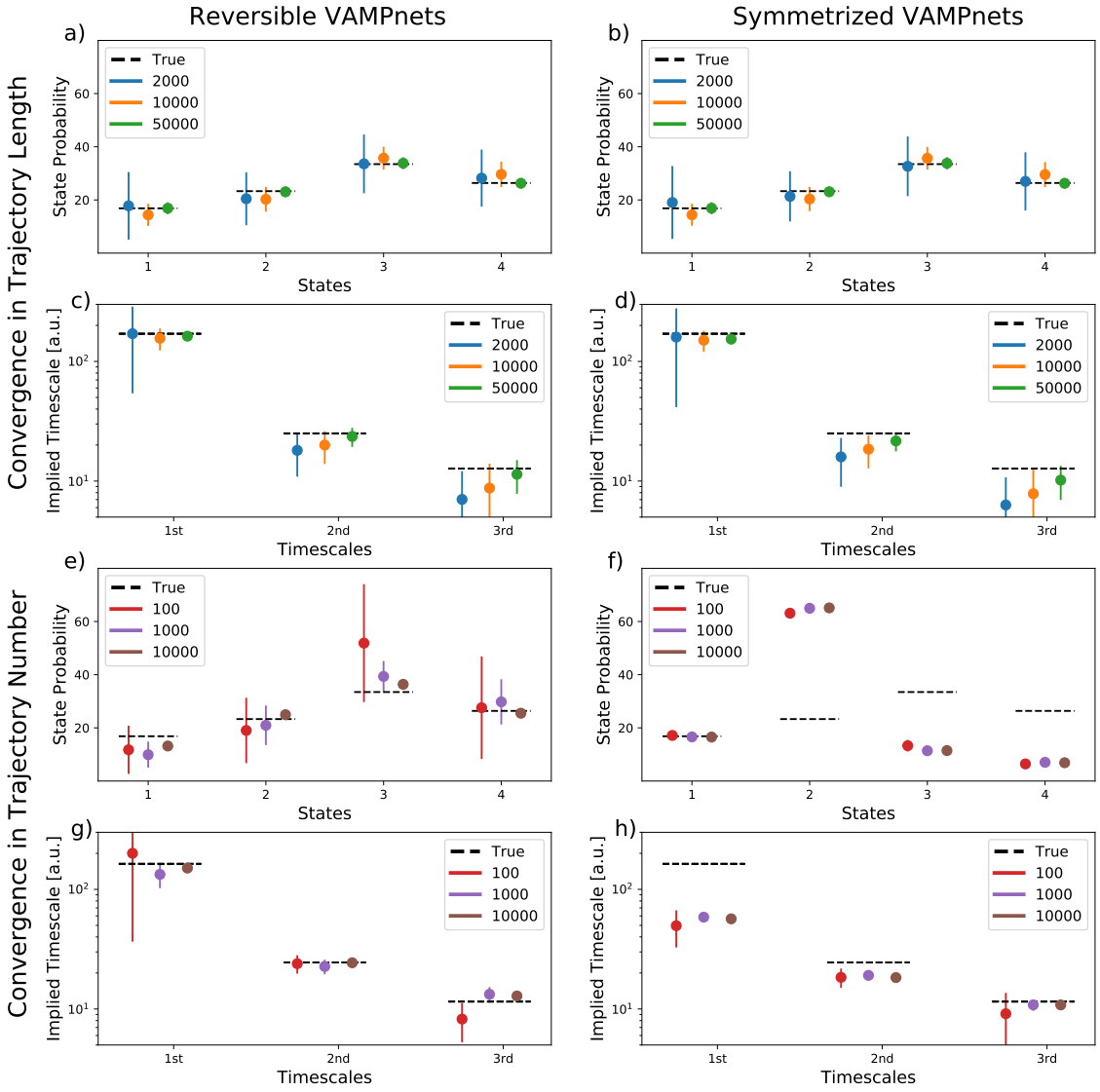

Figure 6.2.: Reversible VAMPnets converge to unbiased equilibrium probabilities from biased data. Comparison of building a Koopman model with a RevVAMPnet (left column) and SymVAMPnets (right column) on the Prinz potential dataset with a varying number of trajectories of 11 frames each starting from an off-equilibrium distribution (**a-d**), and varying length of a single trajectory (**e-h**). Depicted is the state probability to be in the four intervals ($[-1, -.5], [-.5, 0], [0, .5], [.5, 1]$) and the three slowest timescales as the mean over the lag times $[6, 8, 10]$, where the horizontal black line marks the true value (bottom). Errors are estimated over 5 runs as two sigma intervals.

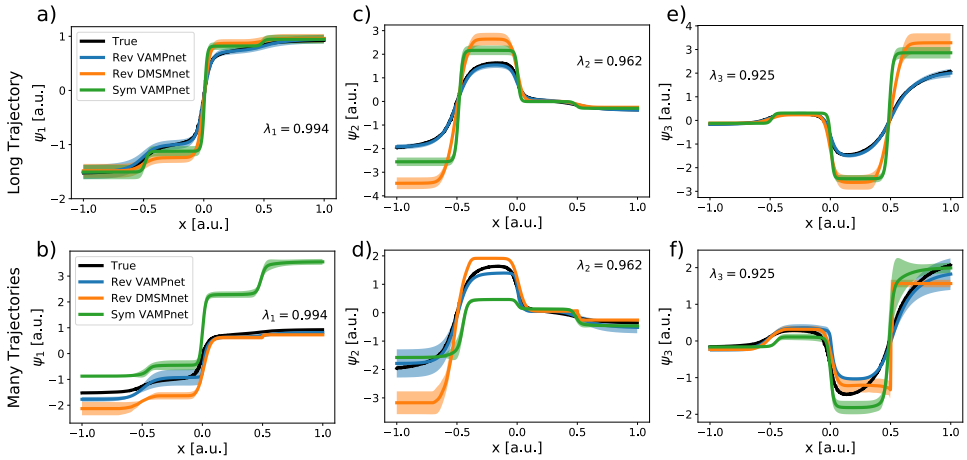

Figure 6.3.: Estimating the three slowest eigenfunction on the Prinz dataset with a RevVAMPnet, a SymVAMPnet, and a RevDMSM. **a**, **c**, **e**) Comparison of the eigenfunction estimated on one long trajectory. **b**, **d**, **f**) Comparison for many short trajectories having an off-equilibrium starting point distribution. Errors are estimated over 5 runs as two sigma intervals.

6.1.3. Approximation of the true eigenfunctions

Finally, we compare the approximation quality of the three slowest non-trivial eigenfunctions for the Prinz potential for the RevVAMPnet, the RevDMSM, and the SymVAMPnet (Fig. 6.3). The data from the analysis before is reused of a single trajectory of length 50,000 frames, and of 10,000 trajectories 11 frames each. We compare the eigenfunctions against a numerical approximation of the true eigenfunctions by a transition matrix computed for a direct uniform 1,000-state discretization of the x-axis for $2 \cdot 10^7$ frames as before [12].

The RevVAMPnet is approximating accurately the true eigenfunctions for all settings. In particular, it is able to recover the eigenfunctions remarkably even in the case of the biased data. The SymVAMPnet results are consistent with previous observations: the approximation of the first two eigenfunctions of the non-equilibrium data (Fig. 6.3 b, d) are strongly biased, resulting in the underestimation of the implied timescales. The constraints in the case of the RevDMSM lead to less smoothly changing eigenfunctions and therefore less accurate approximations. In the case of the long trajectory, both the SymVAMPnet and the RevDMSM exhibit a stepwise behavior of the eigenfunctions, as they tend to result in a harder assignment of states $\boldsymbol{\chi}$. Note that for SymVAMPnet this can be alleviated by avoiding a *softmax* clustering in the last layer and rather directly mapping onto the eigenfunctions [113].

	MSM 5	MSM 100	RevVAMPnet	RevDMSM
VAMP-E	4.2 ± 0.3	4.86 ± 0.01	4.92 ± 0.01	4.93 ± 0.01
Timescale 1	0.14 ± 0.09	0.73 ± 0.09	0.405 ± 0.004	0.424 ± 0.005
Timescale 2	0.3 ± 0.1	0.83 ± 0.05	0.56 ± 0.07	0.50 ± 0.01
Timescale 3	0.6 ± 0.4	1.6 ± 0.3	1.4 ± 0.4	1.2 ± 0.2
Timescale 4	10 ± 4	16 ± 2	12 ± 3	13 ± 1

Table 6.1.: Comparison of the proposed models against ordinary MSM estimation on the NTL9 dataset. Reported is the VAMP-E score at a lag time of $\tau =$ 10 ns and the timescales in μs as the mean and the standard deviation over 5 runs. Our methods are outperforming the MSM estimation for the same number of states and exhibit a competitive performance to the 100 state MSM while keeping the model easily interpretable.

6.1.4. Performance on a larger system

In order to show that the proposed methods outperform or equally perform compared to the standard pipeline of MSM analysis of protein simulations (time-lagged independent component analysis (TICA) as dimension reduction followed by kmeans clustering and a reversible MSM estimation [115, 116]) we applied our methods with 5 output nodes and the MSM estimation with 5 and 100 cluster centers on the NTL9 dataset [92], where the minimal residue distances acted as input features (as in [77]). We compare the VAMP-E score and the estimated 4 highest timescales. The results show that our methods outperform the 5 state MSM and exhibit a competitive performance to the 100 state MSM. However, our methods have the advantages of yielding an easily interpretable model (cf. Tab. 6.1).

6.2. Methods

Code availability The methods were implemented using *Keras* [95] with *tensorflow* [96] as a backend. For the full code and details about the neural network architecture, hyper-parameters and training routine, please refer to
https://github.com/markovmodel/deep_rev_msm.

Unless otherwise noted, we used the adam optimizer [111], a batch-size of 5000, and a six-layer-deep neural network with a constant width of 100 nodes for $\boldsymbol{\eta}$.

Data availability As training data we use either a single simulation trajectory of variable length, or a varying number of short trajectories with fixed length (see

above). Non-equilibrium data are sampled from a starting distribution with probabilities $[15\%, 70\%, 9\%, 6\%]$ to start at the points $[-0.75+x_1, -.25+x_2, .25+x_3, .75+x_4]$ where x_i are independent random variables sampled from a Gaussian distribution with zero mean and standard deviation 0.15.

6.3. Discussion

We have introduced an end-to-end deep learning framework for molecular kinetics that allows us to learn high-quality Markov models with physical constraints such as reversibility and non-negativity of the learned transition matrix. The proposed method is generally applicable for reversible/non-reversible Markov State and Koopman models depending on which constraints are enforced, thus it can be seen as an extension and generalization of previous models such as VAMPnets and deep MSMs. Additionally, the optimization for the state classification and the reversible transition matrix are not explicitly separate processing steps compared to [113]. The proposed method is able to estimate dynamical and stationary properties even from highly biased data and gives state of the art results when studying the slow processes and stationary characteristics of a small toy model.

Despite these advantages, a remaining concern is the optimization procedure, which requires a good balance when fitting the three trainable units at the same time. However, we are confident that the used protocol of first fixing $\boldsymbol{\chi}$ and resetting $\mathbf{u}$ to optimal values according to a non-reversible Koopman model during the training process establishes a reproduceable procedure.

Furthermore, we expect that the maximum likelihood formulation of the proposed method allows us to develop deep learning variants of multi-ensemble MSMs ([99–104]) that alleviate rare event sampling, and augmented MSMs [117] that incorporate experimental data into the model estimation.

Acknowledgements This work was funded by the European Research Commission (ERC CoG "ScaleCell"), Deutsche Forschungsgemeinschaft (CRC 1114/A04, Transregio 186/A12, NO 825/4– 1, Dynlon P8), MATH+ excellence cluster (Project EF1-2), and the "1000-Talent Program of Young Scientists in China". Part of this research was performed while the author was visiting the Institute for Pure and Applied Mathematics (IPAM), which is supported by the National Science Foundation (Grant No. DMS-1440415).

7. Coarse-graining and experimental constraints

The results for the coarse-graining and experimental constraints have been published in the following paper:

> Andreas Mardt and Frank Noé. "Progress in deep Markov State Modeling: Coarse graining and experimental data restraints". *The Journal of Chemical Physics* , 155:214106, 2021. DOI: 10.1063/5.0064668.

Part of the text and illustrations have been adopted unchanged or only slightly changed to fit into this context. Reproduced from the stated paper, with the permission of AIP Publishing.
Andreas Mardt is the single first author. In particular the contributions of the authors were as follows: Andreas Mardt and Frank Noé conceived the project and developed the theory. Andreas Mardt conducted the research. Both contributors wrote the paper.

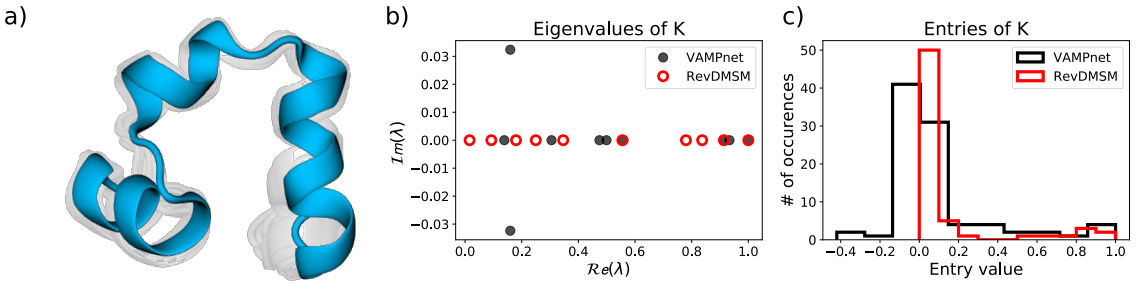

Figure 7.1.: Incorporating constraints on the matrix $\mathbf{T}$ to enforce a reversible model in the regime of poorly sampled systems. (**a**) Folded structure of villin (**b**) Imaginary and real part of the spectrum and (**c**) entries of the transition matrix $\mathbf{T}$ of the RevDMSM and for the Koopman matrix $\mathbf{K}_\chi$ of the non-reversible VAMPnet with a skip of 100 frames and 10 states (1 frame $= \tau = 20$ ns).

7.1. Results

Overview Below we demonstrate how a reversible deep MSM (RevDMSM) can be applied and expanded by our proposed methods to the villin dataset provided by [92] (folded structure Fig. 7.1 a). The input for $\boldsymbol{\eta}$ representing $\boldsymbol{\chi}$ is $\exp(-\mathbf{d})$, where $\mathbf{d}$ is the minimal heavy atom distance between all residues [77]. Firstly, we will study the ability of the method to guarantee a reversible MSM and compare the result against a VAMPnet in a low data regime. Afterwards, we will build an hierarchical model and applying our proposed attention mechanism to it. Based on this, we study folding and unfolding rates via transition path theory. Finally, we will look at the effects of including ground truth observables into the training.

7.1.1. Obtaining real eigenvalues and positive entries in the transition matrix via a RevDMSM

To simulate an insufficient sampled example, we used a skip of 100 frames (1 frame = 20 ns) and 10 output nodes. We train a regular VAMPnet and a RevDMSM on the training data with an early stopping given by the performance on the validation set and estimate the resulting transition matrix on the test set at a time-lag of $\tau = 20$ ns. The eigenvalues of the transition matrix are always real for the RevDMSM, whereas in the case of the Koopman matrix for a VAMPnet pairwise complex eigenvalues may occur (7.1 b). Furthermore, the distribution of the entries demonstrate how the RevDMSM in contrast to the VAMPnet guarantees values be-

tween 0 and 1, which can be interpreted as probabilities (7.1 c). The insufficient sampling leads in the case of the VAMPnet to a non-reversible model, where the Koopman matrix is not a stochastic matrix. The RevDMSM model does not suffer from these shortcomings, nevertheless the constraints imposed on the model result in slightly lower eigenvalues, which can be expected since the constraints impair the ability to approximate the eigenfunctions of the underlying operator.

7.1.2. Building an MSM and testing its validity for a VAMPnet and a RevDMSM

Based on the same data as above but with a skip of 25 frames (1 frame = τ = 5 ns), we built an MSM with 4 output nodes using a RevDMSM and a VAMPnet, respectively. By inspecting the state network connected by their transition probabilities we observe again negative transition probabilities for the VAMPnet (red arrows Fig. 7.2 b). However, the implied timescale and the CK test confirm the ability of both models to predict the long-time kinetics. Furthermore, the models agree upon the timescales within the 70th percentile estimated over 10 runs and discover similar metastable states, where 10 aligned representative structures are depicted next to each state (Fig. 7.2 a, b). Although each model uses different trained state assignments $\boldsymbol{\chi}$ they both identify a folded state (F), an unfolded state (U), a partially folded state (PF), and a misfolded state (M) characterized by a helix including the amino acids 20LEU and 21PRO which form a coil in the folded state.

7.1.3. Building an hierarchical model with an interpretable attention mechanism

In order to demonstrate the application of an hierarchical model we coarse-grain a 4-state RevDMSM to a 3-state and consecutively to a 2-state model. Additionally, we incorporate an attention mechanism into the architecture, where the attention weights are time dependent (cf. Fig. 3.8). After training the 4-state model at τ = 50 ns we train the coarse-graining matrices with the VAMP-E score consecutively with the pseudoinverse method. Finally, we simultaneously update $\mathbf{u}$ and $\mathbf{S}$ from the 4-state model and the two coarse-graining matrices to maximize the sum of the VAMP-E scores of all three models.

For the estimation of the implied timescales of all models it is sufficient to exclusively retrain $\mathbf{u}$ and $\mathbf{S}$ from the 4-state model to optimize the sum of the individual

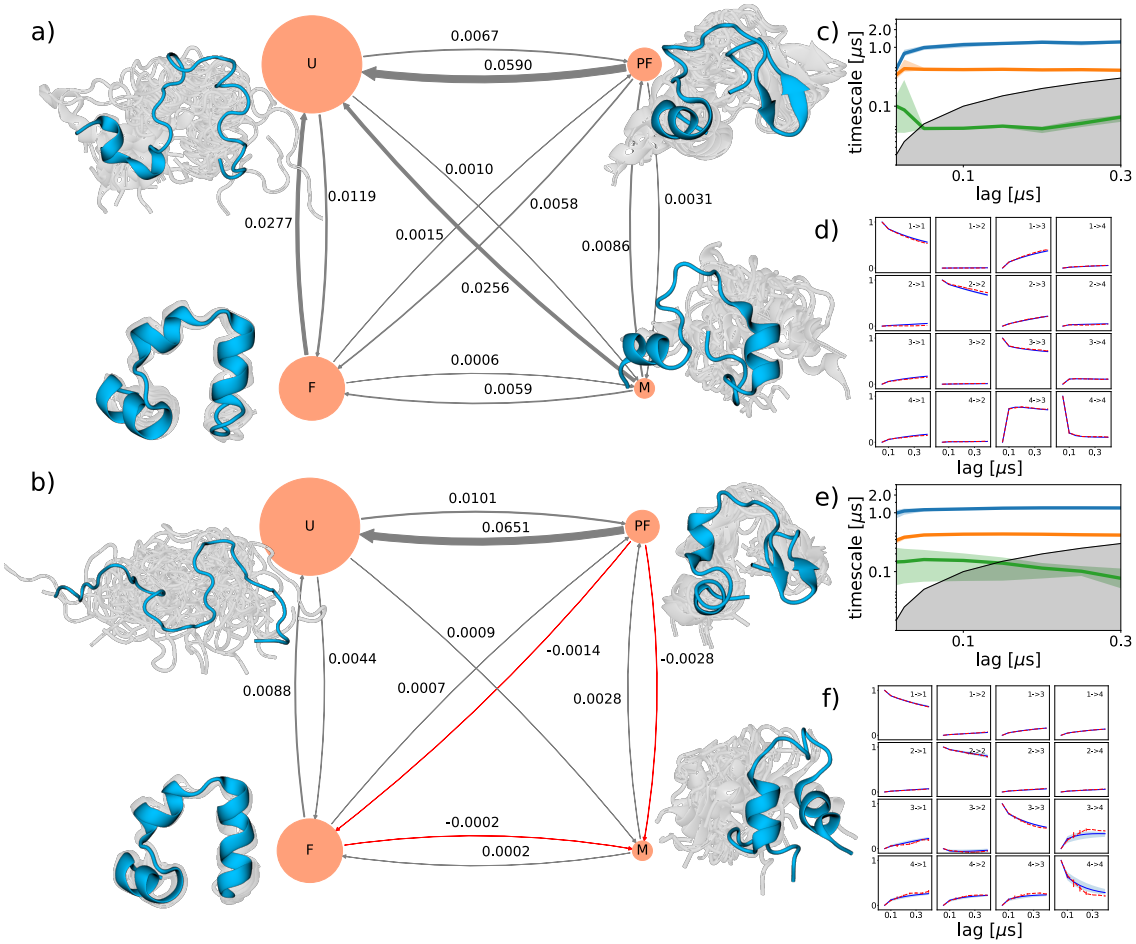

Figure 7.2.: Comparison of building an MSM with non-reversible VAMPnets and RevDMSM with a skip of 25 frames (1 frame = 5 ns) and 4 states. State network of the RevDMSM model (**a**) and VAMPnet (**b**), the size of a state corresponds to the stationary distribution, the arrows are the transition rates of the $\mathbf{T}$ or $\mathbf{K}_\chi$ matrix, respectively (red arrows indicate negative entries). Additionally, 10 representative structures aligned according to their secondary structure are shown next to the states. Although trained independently both feature functions identify a folded state (F), and unfolded state (U), a partially folded state (PF), and a misfolded state (M). Model validation through the implied timescales of the (**c**) RevDMSM and (**e**) VAMPnets model and the CK-test (**d**) and (**f**) at a base model estimated at $\tau_{\mathrm{MSM}} = 50$ ns. Errors are estimated over 10 runs.

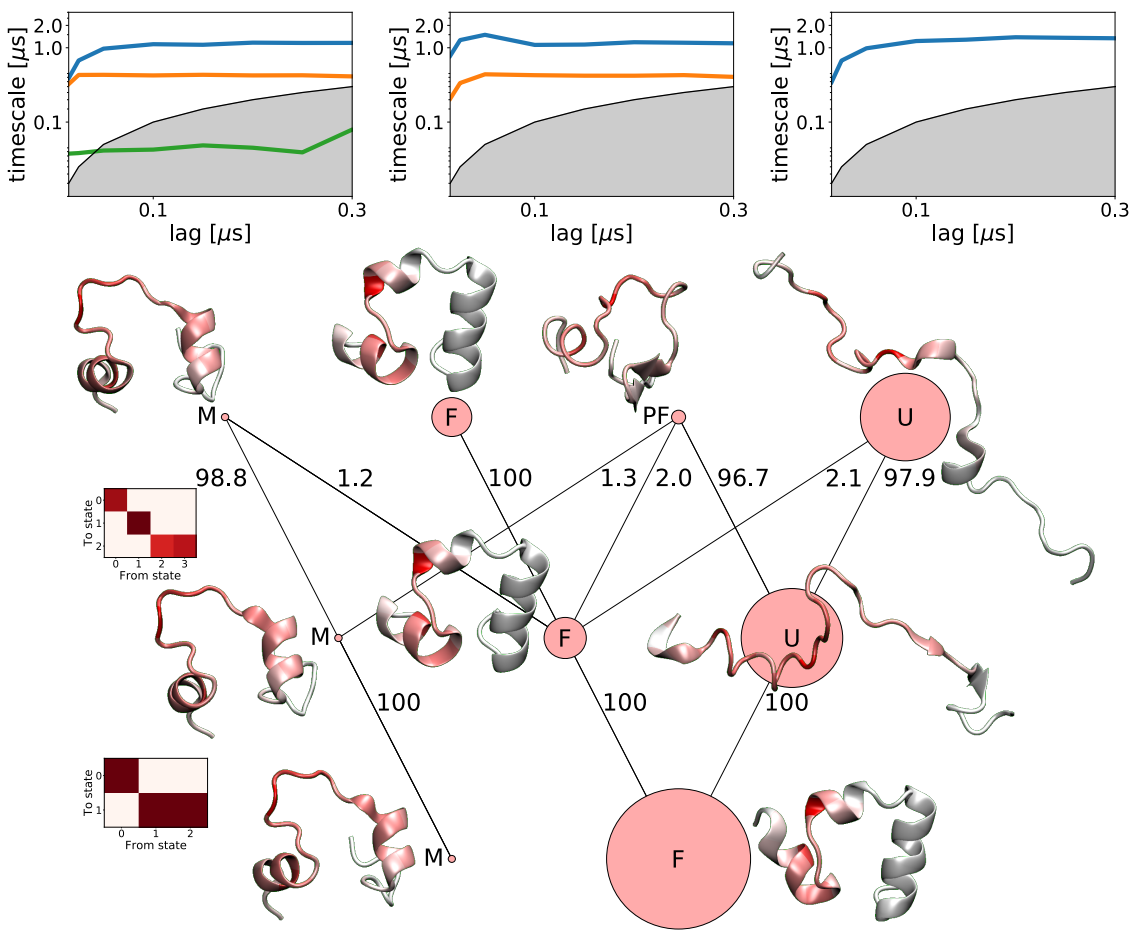

Figure 7.3.: Hierarchical model from 4 to 3 to 2 states with additional attention analysis. The top row shows the implied timescales for each model where the slowest timescales are conserved when coarse-grained. Below, the network graph is depicted where each node represents a state: folded (F), unfolded (U), partially folded (PF), and misfolded (M). The connections between the nodes represent how the states contribute to the coarser representation, where the probability of belonging to the coarse-grained state is attached to it (probabilities $< 1\%$ are omitted). Additionally, a graphical representation of the matrices $\mathbf{M}$ are depicted beside it (white-low, red-high probability). For each state the most likely configuration is shown, where the color indicates the attention weight learned during training (white-low red-high attention).

scores. The 3- and 2-state model conserve both the highest timescales as expected (Fig. 7.3 top row). Thereby, the 2-state model, where the unfolded state is missing, indicates that the slowest timescale is not connected to the folding process. Instead, the state of the unfolded structure appears in the 3-state model emphasizing the nature of folding of the second timescale.

Below the timescales we depict a graphical representation of the hierarchical model, where each node represents a state, where the same nomenclature is used as above. The connections between the nodes represent the coarse-graining matrix, where the numbers encode the probability that the state belongs to the coarse-grained state (connections with a probability less than 1% are not displayed). For reasons of clarity we added additionally visual representations of the matrices beside it, where the color encodes the probability (white-low, red-high). There, it becomes evident that the coarse-grain matrices have a very sparse structure. The large values seem to be in agreement with the observation in [78] that RevDMSMs tend to rather hard assignments of states. However, the hard assignment supports the interpretation of an hierarchical model.

Next to the nodes we added visualization of the structures of villin with the highest probability for that particular state. There, the hierarchical splitting is visible: structure elements of the higher model hierarchy are preserved in the lower level.

Furthermore, we depict the attention weight of each residue in the color scheme of the structures (red-high, white-low). It is worth mentioning, that the attention mechanism needs to highlight areas which are important to distinguish all four metastable states. This implies that the absence of a specific secondary structure could be important and therefore highlighted.

Remarkably, our attention mechanism detects 13ARG as important for the folded structure (F), which we found aligns well with the folding process (Sec. 7.1.6). The misfolded state (M) shows high attention at the amino acids 20LEU and 21PRO being part of a helix which form a coil in the folded state. The last residues of the chain seem to be bad descriptors for the dynamics which seems reasonable due to their more flexible nature.

In general, the network assigns high attention mainly to regions where states have themselves secondary structure or where they lack the structure other states have, e.g. the middle helix of the folded state.

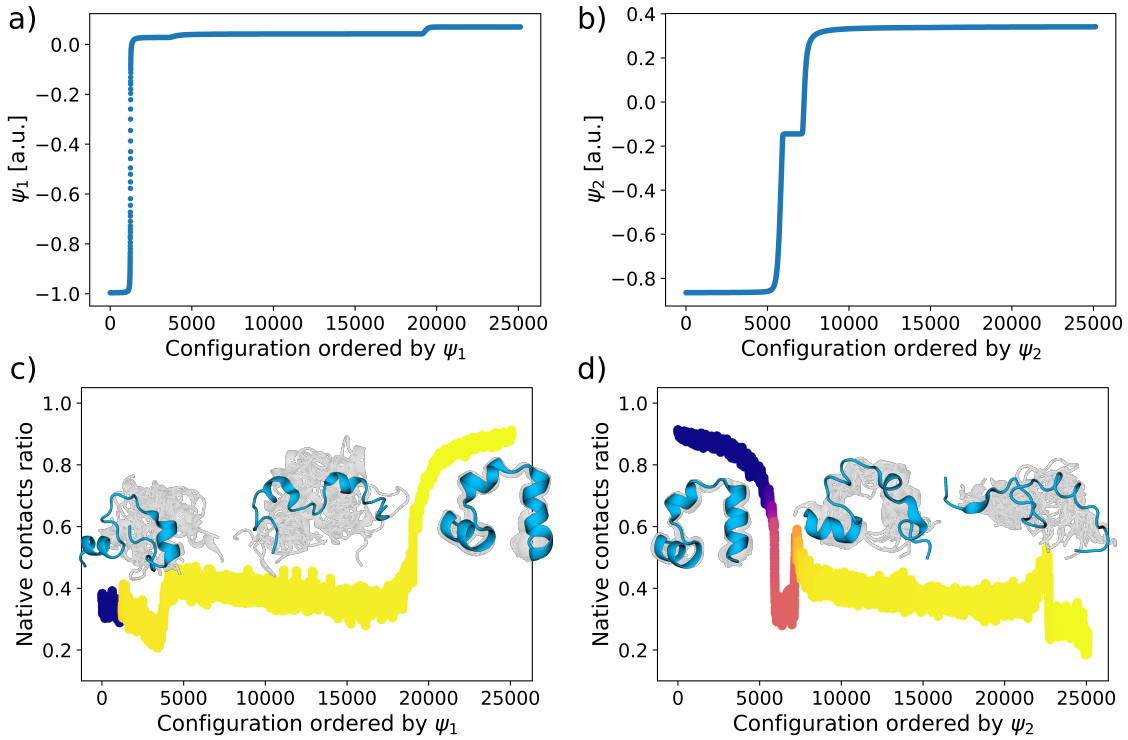

Figure 7.4.: Estimating the two slowest eigenfunctions with a RevDMSM on the villin dataset with a skip of 25 frames ($\tau = 50$ ns) and 4 states. (**a**), (**b**) Eigenfunctions against frames ordered by their eigenfunction value (fake trajectory along the process). (**c**), (**d**) The ratio of formed native contacts with a mean window over 30 frames is plotted against the same x-axis, where the color encodes the value of the eigenfunction. Representative structures for three regions are shown.

7.1.4. Approximation of the leading eigenfunctions via a RevDMSM

Driven by the observation that the slowest process might not be the folding process, we test the ability of the above RevDMSM to approximate the two slowest eigenfunctions of the 4 state model of villin employing Eq. 3.64. In order to visualize the process, the frames are ordered by their eigenfunction value (Fig. 7.4 a, b) and plotted against the ratio of how many native contacts are formed (Fig. 7.4 c, d). We define residues being in contact, which are at least two amino acids apart in the chain and closer than 0.45 nm with respect to their closest heavy atoms. In favor of a smoother result, we apply a mean window over 30 frames. For both eigenfunctions 10 aligned representative structures for three regions along the process are shown.

The results confirm that the slowest eigenfunction is not the folding process [118]. Instead, the analysis reveals a process from the misfolded via the unfolded to the folded structure. However, the second slowest process represents the expected folding process from folded to unfolded.

7.1.5. Estimation of folding rates

We can study via transition-path-theory (TPT) the folding and unfolding rates of the system. We built a three state RevDMSM, where $\boldsymbol{\chi}$ was trained at a time-lag of $\tau = 5$ ns and $\mathbf{u}$ and $\mathbf{S}$ retrained at a lag-time of $\tau_{\mathrm{MSM}} = 100$ ns, where the timescales were converged. The model has to be constituted of at least three states because the folding process is the second slowest timescale in the villin trajectory. Given the model we estimate from the transition matrix $\mathbf{T}$ the mean first passage time with the *PyEMMA* package [115] for the folding and unfolding process:

$$\tau_{\mathrm{folding}} = (3.0 \pm 0.3)\ \mu\mathrm{s}, \tag{7.1}$$

$$\tau_{\mathrm{unfolding}} = (1.0 \pm 0.1)\ \mu\mathrm{s}, \tag{7.2}$$

where the error is given via the standard deviations over 10 runs. Lindorff-Larsson *et al.*[92] report a folding rate of $(2.8 \pm 0.5)\ \mu$s and unfolding rate of $(0.9 \pm 0.2)\ \mu$s which is in perfect agreement with our findings, although they utilize a handcrafted definition of states and simply measure the average lifetime of the folded and unfolded state and the transition time as the average of all events in the trajectory [92].

7.1.6. Estimating deep MSMs with experimental restraints

In order to mimic the situation of having ground truth values preferably from an experiment but biased simulation data for model estimation, we treat expectation values from the whole long simulation as ground truth. However, we imitate the situation of a simulation which overestimates transition energy barriers compared to the ground truth by removing three fourth of folding and unfolding events from the full trajectory. We detect these events by inspecting the sign changes of the second slowest eigenfunction (Sec. 7.2.4).

Since we perturbed by our data modification the folding/unfolding process, we focus on observables related to it. Some of them serve as additional information for

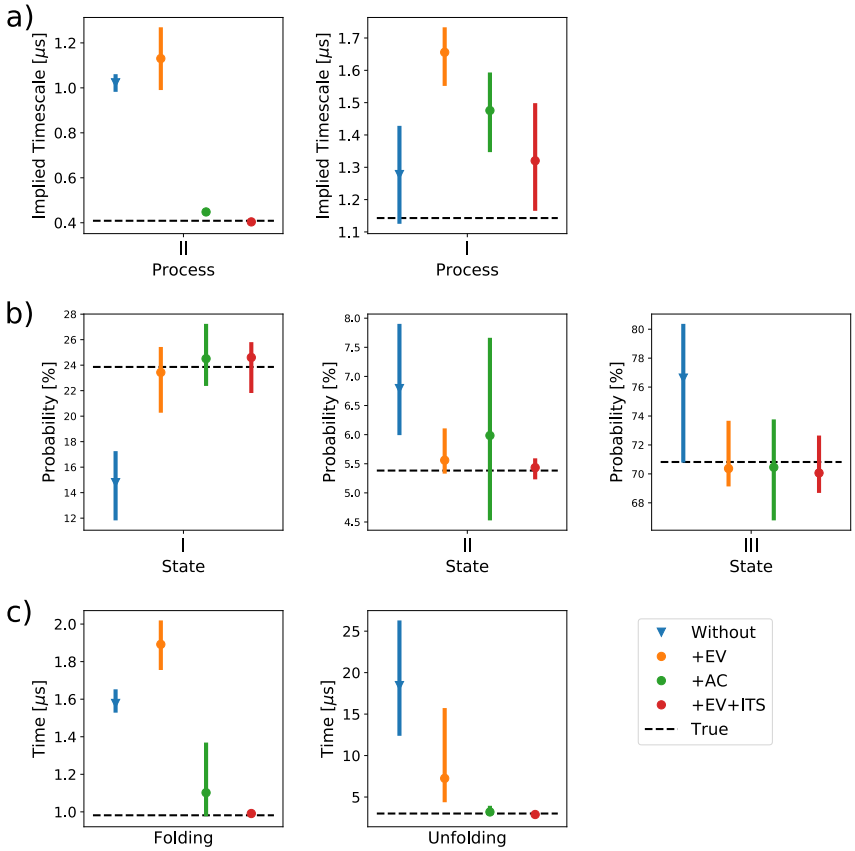

Figure 7.5.: Estimating observables with biased data and comparing estimates between models incorporating some of them into the training routine (orange, green, red) and a model incorporating none (blue). The data is biased by removing folding/unfolding events from the trajectory to simulate a probable higher transition barrier originating from possible erroneous force fields. The true values (dashed black line), ideally from an experiment, were estimated from a model trained on the whole unbiased data set. (**a**) two slowest timescales, where process I transitions between misfolded and folded (Fig. 7.4) and process II is the perturbed folding process, which was used for the red model as training observable. Therefore, the better approximation of that process is expected. (**b**) state probabilities (state I (folded), II (misfolded), III (unfolded), compare Fig. 7.3 for representations) and therefore a stationary observable, which was not used during training. However, using the expectation value of the contact 1LEU-13ARG being formed improves the estimation of the stationary properties (orange, red). (**c**) folding and unfolding rates estimated via the transition path theory not used for training, where incorporating the timescale or the autocorrelations of the contact 1LEU-13ARG staying formed or unformed improves the estimation significantly (red and green). The error bars indicate the 70th percentile.

training (cf. Sec. 3.3.1), the others as a validation set to compare the performance on them:

1. The kinetics via the implied timescales (ITS) of the two slowest processes.
2. The stationary distribution of three predefined states.
3. The folding and unfolding rates via TPT analysis.
4. The expectation value (EV) of contact 1LEU-13ARG being formed.
5. The autocorrelation (AC) of the contact 1LEU-13ARG staying formed or unformed.

The contact 1LEU-13ARG is chosen because it correlates well with the folding eigenfunction and could be possibly experimentally observed by attaching fluorescence labels and conducting an fluorescence correlation experiment. Although identifying contacts accessible to experiments needs the expertise of an experimentalist, we only made the present choices to illustrate the effect of incorporating such constraints on the estimates of the mentioned observables. A contact is said to be formed if the minimal residue distance is shorter than 0.45 nm:

$$a_t = \begin{cases} 1 & \text{if } d(t)_{\text{1-13}} < 0.45 \text{ nm} \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

The three predefined states origin from the classification of the model from Sec. 7.1.5 (State I (folded), II (misfolded), III (unfolded)). The observation of the contact not being formed is simply $b_t = 1 - a_t$.

Both models, with (further called observable model) or without (ordinary model) additional observable, take the same data splitting and the same pretrained VAMPnet as a starting point. Afterwards both are trained as described above except for the modified loss function with the same time-lag values as above. The factor in front of the observable loss is always $\xi = 10$ for the results presented here. We rotate the observable used for training and report the results in Fig. 7.5. Error bars indicate the 70th percentile over 10 training runs. It can be seen that the second timescale is confidentially overestimated by the ordinary model (blue) as intended, which has a direct effect on the folding and unfolding rates. Furthermore, the stationary distribution of the three states is affected, which implies that estimates of expectation values cannot be trusted.

Building an observable model (orange) including the expectation of the contact being formed improves the estimation of the stationary distribution overall (Fig. 7.5 b). However, there is no positive effect on the estimation of the kinetics.

Taking both autocorrelations of the contact staying formed and unformed as observables, the observations change (green). It improves the estimates on both stationary and kinetic properties, respectively. Whereas a good performance on the kinetics might be expected, the reason for the positive impact on the stationary distribution becomes only obvious by studying the properties of the autocorrelations. The difference between the two unnormalized autocorrelations is given by the expectation value of the contact (cf. Sec. 7.2.1). Therefore, if both autocorrelations are matched, the expectation value should be matched, which has a positive effect on the state probabilities as seen above.

Motivated by these findings we tried to match the second implied timescale and the expectation value along training (red). Here, all observables are matched the best.

7.2. Methods

7.2.1. Matching two dependent time correlations

Given a microscopic observable a_1 and defining $a_2 = 1 - a_1$, the expectation value of a_2 is:

$$\mathbb{E}[a_2] = \mathbb{E}[1 - a_1] = 1 - \mathbb{E}[a_1]. \tag{7.4}$$

Therefore, it is of no use to match both expectation values $\mathbb{E}[a_1], \mathbb{E}[a_2]$. However, if we inspect the time correlation:

$$\begin{aligned} \mathbb{E}[a_2(t)a_2(t+\tau)] &= \mathbb{E}[(1 - a_1(t))(1 - a_1(t+\tau))] && (7.5)\\ &= 1 - \mathbb{E}[a_1(t)] - \mathbb{E}[a_1(t+\tau)] && (7.6)\\ &\quad + \mathbb{E}[a_1(t)a_1(t+\tau)], \end{aligned}$$

it is obvious that matching both time correlations is equivalent to matching one time correlation and the corresponding expectation value.

7.2.2. Connection between timescales and folding/unfolding rates

Given experimental rates (time interval per event) for folding r_{on} and unfolding r_{off} it can be described as a two state (folded and unfolded) Markov process with a transition matrix $\mathbf{T}$ or equivalently by the rate matrix $\mathbf{K}$. Due to the conservation of probability the eigenvalues of the rate matrix are given by $\boldsymbol{\lambda}(\mathbf{K}) = [0, -r_{\text{on}} - r_{\text{off}}]$. The transition matrix and rate matrix are connected by [119]:

$$\mathbf{T}(\tau) = \exp(\mathbf{K}\tau). \tag{7.7}$$

The eigenvalues of $\mathbf{T}$ are $[1, \lambda]$, where λ is the eigenvalue corresponding to the folding process. Via the trace the connection between eigenvalue λ and the rates is recognizable:

$$\begin{aligned} &\text{tr}(\mathbf{T}) = 1 + \lambda = \text{tr}(\exp(\mathbf{K}\tau)) = 1 + \exp(-\tau(r_{\text{on}} + r_{\text{off}})) \\ &\Rightarrow \lambda = \exp(-\tau(r_{\text{on}} + r_{\text{off}})), \end{aligned} \tag{7.8}$$

which can be extended to the corresponding timescale $t = \frac{1}{r_{\text{on}} + r_{\text{off}}}$.

7.2.3. Pretraining the VAMPnet

Mardt *et al.[77]* propose a pretraining of the VAMPnet with a symmetrized VAMP-loss in order to achieve a more crisp assignment of the states. Since there is no clear motivation about this particular procedure, we modified the pretraining by adding to the VAMP-2 score the term $\text{tr}(\mathbf{C}_{00})$ which will maximize the eigenvalues of the matrix and therefore favors harder state assignments. The updated loss changes to the following:

$$L = -\text{VAMP-2} - \xi \text{tr}(\mathbf{C}_{00}), \tag{7.9}$$

where ξ balances the two terms and can be set to zero during the following unperturbed training phase. Thereby, we give a clear motivation that the additional term pushes the network into a more favorable region during training.

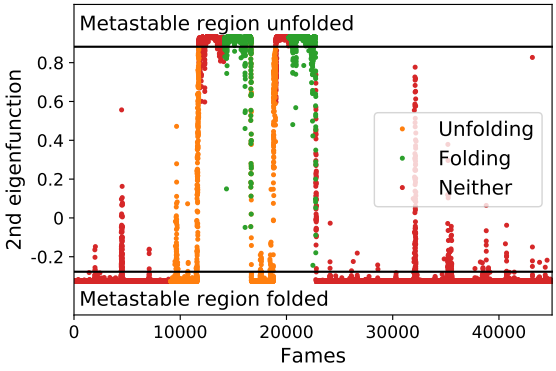

Figure 7.6.: Example of the classification based on the 2nd eigenfunction of the trajectory into the three possible classes: folding (orange), unfolding (green) and staying within a metastable set (red). As we showed in Sec. 7.1.4, the 2nd eigenfunction can be connected to the folding process. Based on the classification we removed 75 % of the detected transitions of both directions.

7.2.4. Data manipulation routine for the observable model

We seek to mimic the situation where in a simulation the transition barrier is overestimated by the force field compared to a ground truth situation such as a real world experiment. Therefore, we take a long simulation as ground truth and perturb the biased simulation by only removing a fraction of observed transitions.

In order to define a transition we make use of the estimated eigenfunction connected to folding on the long unperturbed trajectory. We define the two metastable sets (folded and unfolded) as configurations which lie within $\approx$ 10 % of the maximum and minimum of the eigenfunction (Fig. 7.6 black lines). A transition is then defined as a sign change of the eigenfunction starting from one metastable state to the other. Thereby, we found 80 folding (orange) and unfolding (green) events which are connected by simulation parts where the system remains in one metastable state (red). The perturbed training data are all the trajectory parts which remain within a metastable set and 25 % of the detected transitions.

Code availability The methods were implemented using *PyTorch* [97]. For the full code and details about the neural network architecture, hyperparameters and training routine, please refer to https://github.com/markovmodel/deepmsm. In general, we used the Adam optimizer [111], a batchsize of 10000, and a 6 layer deep neural network with a constant width of 100 nodes with the ELU activation function for

$\boldsymbol{\eta}$. When using an attention network it has the same architecture as $\boldsymbol{\eta}$ except of the output size and a window size of 4 motivated by the fact that the shortest beta strands and a helix turn is about four residues long.

The whole simulation data is randomly split threefold into 70% training, 20% validation and 10% test set data. The validation data is used to tune hyperparameters and enact an early stopping mechanism. The results are reported for the test set.

Data availability The data that support the findings of this study are available from Lindorff-Larsen *et al.*[92]. Restrictions apply to the availability of these data, which were used under license for this study. Data are available from the authors upon reasonable request and with the permission of Lindorff-Larsen *et al.*[92].

7.3. Discussion

Here we extend the previously proposed reversible deep MSMs[78] by adding features well established for traditional MSMs: the coarse-graining of Markov states to a fewer-state MSM, and the inclusion of experimental restrains into the MSM estimation process. We apply these methods to the study various aspects of the villin headpiece miniprotein kinetics. We exploit the fact that RevDMSMs are faithful probability models and apply transition path theory to study mean first passage times of the folding and unfolding event, where our result coincides with the previously published results [92]. Furthermore, we established an approach how experimental data can be incorporated into the model estimation and how it can possibly compensate for biases in the underlying force fields. In addition, the coarse-graining method proved valuable in constructing hierarchical models, which give rise to easily interpretable states and allow to study the system on different levels of detail. Finally, we demonstrated how an attention mechanism can draw the attention to residues being important for the classification of the dynamics. Thereby, it could be a valuable tool for practitioners to find targets for mutations to be studied.

Despite these benefits, it remains an open challenge to develop specialized network architectures for protein dynamics analysis, especially the attention network could profit from an architecture where parameters are shared among residues. Furthermore, the inclusion of real experimental observables remains a task for future studies, where the method would need to prove its capabilities to counteract biases of the simulation due to the underlying force field.

Acknowledgements This work was funded by the European Research Commission (ERC CoG "ScaleCell"), Deutsche Forschungsgemeinschaft (CRC 1114/A04, CRC 958/A04), the Berlin mathematics research center MATH+ (Projects AA1-6 and EF1-2), and the German ministry for research and education (BIFOLD).

8. iVAMPnets

The results for the iVAMPnets project have still to be reviewed:

> Andreas Mardt, Tim Hempel, Cecilia Clementi, and Frank Noé. "Deep learning to decompose macromolecules into independent Markovian domains". *bioRxiv*, DOI: 10.1101/2022.03.30.486366.

Part of the text and illustrations have been adopted unchanged or only slightly changed to fit into this context.
Andreas Mardt and Tim Hempel contributed equally to this work. In particular the contributions of the authors were as follows: Andreas Mardt, Tim Hempel, Cecilia Clementi, and Frank Noé conceived the project. Andreas Mardt developed the theory. Tim Hempel developed test systems. Andreas Mardt and Tim Hempel performed research. All contributors wrote the paper.

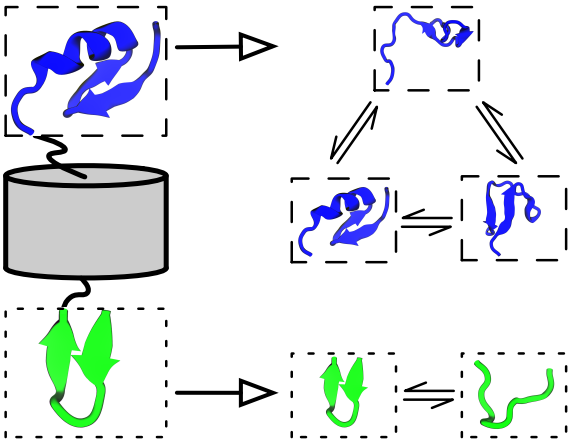

Figure 8.1.: The iVAMP concept as visualized by modeling dynamics of a protein that has two independent, flexible regions separated by a rigid barrel. iVAMPnets learn an assignment of the C- (blue/top) and N-termini (green/bottom) into independent subsystems from molecular dynamics trajectories (left panel). Subsequently, the folding/unfolding dynamics of either terminus is modeled with a VAMPnet, disregarding the state of the other terminus (right panel).

8.1. Results

Overview A new fundamental problem arises from the increasing interest for large protein systems: Growing numbers of possibly metastable states necessitate ever larger amounts of MD sampling to gather sufficient statistics, a vicious cycle considering that large systems are also harder to sample. However, large systems might display localized conformational changes [50] or consist of weakly-coupled or independent subsystems (cf. Fig. 8.1), a notion which can be exploited to reduce the necessary sampling. Since a global state is a combination of the individual states of all subsystems, the number of global states grows exponentially with the number of subsystems [51, 52]. Therefore, a method aiming at describing the global states explicitly is fundamentally unscalable. A solution to this problem thus needs to address two separate issues: (a) dividing the protein system into Markovian subsystems and (b) learning the coupling between them. Olsson & Noé [51] addressed this challenge, in particular (b), by a dynamic graphical model treating the global system as an Ising model, where the states or "spins" of the subsystems are coupled. In comparison, Hempel et al. [52] addressed (a) by approximating the global system dynamics as a set of independent (uncoupled) Markov models (termed Independent Markov decomposition, IMD). They furthermore propose a pairwise independence

score of features, which allows to detect nearly uncoupled regions where independent Markov state models can be estimated subsequently.

In this manuscript, we present a joint IMD and VAMP approach (termed independent VAMPnets, or shorthand iVAMPnets) that aims at solving issue (a) by generalizing IMD to neural network basis functions. iVAMPnets are an integrated end-to-end learning approach that decomposes the global dynamics into weakly coupled subsystems with a subsystem-membership matrix, and estimates a VAMPnet for each of these subsystems for a simplified, subsequent analysis. Please compare Fig. 8.1 for a conceptual overview. In comparison to previous implementations of IMD, our approach autonomously decouples independent subsystems and accounts for dynamics that is hidden in non-linear functions of the input features.

8.1.1. Toy model with 2 independent subsystems

We first demonstrate that iVAMPnets are capable of decomposing a dynamical system into its independent Markovian subsystems based on observed trajectory data using an exactly decomposable benchmark model.

Akin to the protein illustrated in Fig. 8.1, we define a system that consists of two independent subsystems with two and three states, respectively. It is modeled by two transition matrices with the corresponding number of states. We subsequently sample a discrete trajectory with each matrix (100k steps) [98]. The global state is defined as a combination of these discrete states. The discrete subsystem states are now interpreted as the hidden states of hidden Markov models [120] that emit to separate, subsystem-specific dimensions of a 2D space. The output of each subsystem is modeled with Gaussian noise $N(\mu_i, \tilde{\sigma}) \in \mathbb{R}$ that is specific to the state that the system is in, specified by the mean μ_i, and a constant $\tilde{\sigma}$. The two state subsystem therefore describes a jump process between Gaussian basins along the x-axis and the three state along the y-axis, respectively (Fig. 8.2 a). These variables compare to collective variables of the green (x) and blue (y) system depicted in Fig. 8.1. Please note that while in this toy system the relevant slow collective variables are known, iVAMPnets are generally capable of finding them (cf. Sec. 10D hypercube toy model and Synaptotagmin-C2A).

We now explain how we apply iVAMPnets. Since the generative toy model consists of perfectly independent subsystems and the pair already describes the global system, our method can simply be optimized for the global VAMP-E score (Eq. 3.56) without the need for any further constraints. We train a model with a two

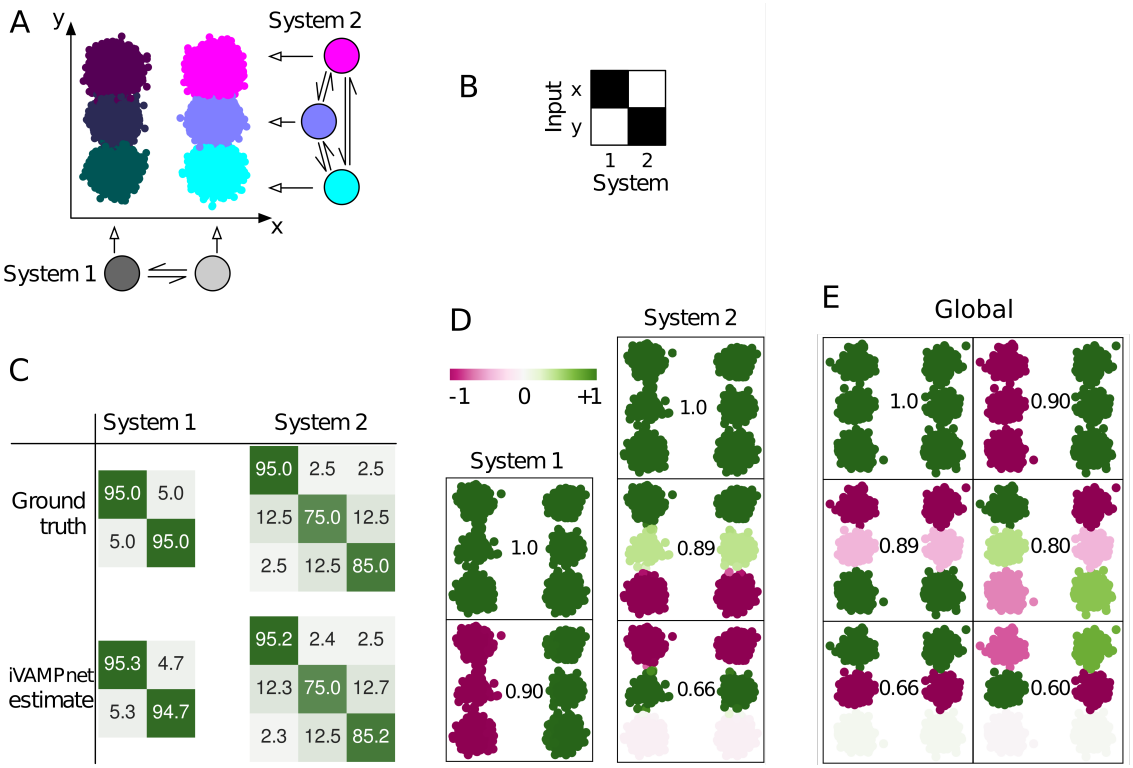

Figure 8.2.: Hidden Markov state model as a toy example for independent subsystems: (**a**) 2 subsystems with 2 and 3 states emit independently to an x and y axis, respectively. The corresponding 2D space embeds all 6 global states. (**b**) The learned mask shows that each subsystem focuses on one input dimension. (**c**) The estimated subsystem transition matrices are compared with the ground truth. (**d**) Subsystem eigenfunctions and corresponding eigenvalues as found by iVAMPnets. Independent processes are recovered from the 2D data. Here, it is not necessary to enforce any independence constraint. (**e**) The 6 global eigenfunctions supplied with their eigenvalues revealing the 4 independent processes and the 2 resulting mixed product processes. Eigenvalues of the latter are computed from the product of independent process eigenvalues.

and three state subsystem at a lag time $\tau = 1$ frame (cf. Fig. 3.7).

Once trained, iVAMPnets yield a model of the dynamics in each of the identified subsystems. As expected, we find that the estimated transition matrices for both subsystems closely agree with the ground truth (Fig. 8.2 c). To additionally assess the slow subsystem dynamics in more detail, we borrow concepts from MSM analysis and conduct an eigenvalue decomposition of the iVAMPnet models (cf. Sec. 3.7.3).

The analysis of the eigenfunctions demonstrates that, by construction, the system exhibits one independent process along the x-axis ($\lambda_1 = 0.9$) and two along the y-axis ($\lambda_2 = 0.887$ and $\lambda_4 = 0.663$) (Fig. 8.2 d).

In contrast, we note that in the picture of *global* states, two additional processes would appear as a result of mixing the independent processes (cf. Fig. 8.2 e), which makes the combined dynamical model more challenging to analyze, whereas the iVAMPnet analysis remains straightforward and simple. Besides the dynamical models, our iVAMPnet yields assignments between input features and subsystems. We find that the method correctly identifies the two state system as the x-axis and the three states as the y-axis feature, respectively (Fig. 8.2 b).

8.1.2. 10D hypercube toy model

In a next step we test the iVAMPnet approach with ten 2-state subsystems, which corresponds to 1024 global states (Fig. 8.3 a, b). As before, the dynamics is generated by ten independent Hidden Markov state models with unique timescales. The system is split into five pairs of subsystems, and the two coordinates governing the transition dynamics of each pair are rotated in order to make it more difficult to separate them (Fig. 8.3 a). Additionally, we make the learning problem harder by adding ten noise dimensions such that the global system lives on a 10-dimensional hypercube embedded in a 20 dimensional space.

Although the subsystems are perfectly independent, we will estimate iVAMPnets with the VAMP-E score in a pairwise fashion, thereby avoiding to estimate expensively large correlation matrices in $\mathbf{C} \in \mathbb{R}^{1024\times1024}$. As this is only justified if all systems are independent, we additionally enforce Eq. 3.52 during training by minimizing Eq. 3.57 and thereby rule out that any two subsystems approximate the same process.

The iVAMPnet estimation yields subsystem models which, as common in MSM analysis, can be validated by testing whether their implied relaxation timescales are converged in the model lag time τ. We find that the implied timescales learned by

the iVAMPnet are indeed converged and accurately reproduce the ground truth (Fig. 8.3 d). We note that in addition to the timescales of the individual subsystems that are identified by the iVAMPnet, a global model would also contain all timescales that result from products of eigenvalues, resulting in a total of 1024 timescales (1023 when excluding the stationary timescale). Thus, the iVAMPnet analysis provides a much simpler and more concise model than a global MSM or VAMPnet would. Furthermore, the subsystem assignment mask indicates that the method correctly assigns high importance weight to two input features for each model (Fig. 8.3 c). Therefore, the method proves its capability of decomposing a noisy, high dimensional global system into its independent sub-processes in a data efficient way.

8.1.3. Synaptotagmin-C2A

Finally, we test iVAMPnets on an all-atom protein system, where we use the attention mechanism described in Sec. 3.6.2 with inter residue distances as input features. In comparison to our toy examples, we expect the underlying global dynamics to be only approximately decomposable into independent subsystems. Our test system is the C2A domain of synaptotagmin that was described by our group previously [123]; it plays a crucial role in the regulation of neurotransmitter release [124]. It was shown to consist of approximately uncoupled subsystems containing the calcium binding region (CBR) and the C78 loop, respectively [52].

First, we attempted to model the protein with a global model, i.e., with a single (regular) VAMPnet. However, this approach failed because there were not enough simulation statistics to estimate a reversibly connected transition model between all global metastable states, resulting in diverging implied timescales (cf. Fig. 8.4). This is exactly the scenario where iVAMPnets should provide an advantage, by only relying on locally rather than globally converged transition statistics.

Next, we train iVAMPnets to seek two subsystems of 8 states each at a lag time of $\tau = 10$ ns where we enforce constraint 3.52 to find uncoupled subsystems. By taking $x_{ij} = w_i w_j \exp(-d_{ij})$ as input features where d_{ij} is the minimal heavy atom distance between residue i, j we ensure that our results are translationally and rotationally invariant (cf. Sec. 3.6.1).

The trained iVAMPnet identifies one subsystem comprising all three CBR loops (CBR-1, CBR-2, CBR-3; Fig. 8.5 a). The second subsystem consists not only of the aforementioned C78 loop but also of the loop connecting beta sheets 3 and 4 [125] (termed C34 henceforth). When mapping the residue positions on the protein

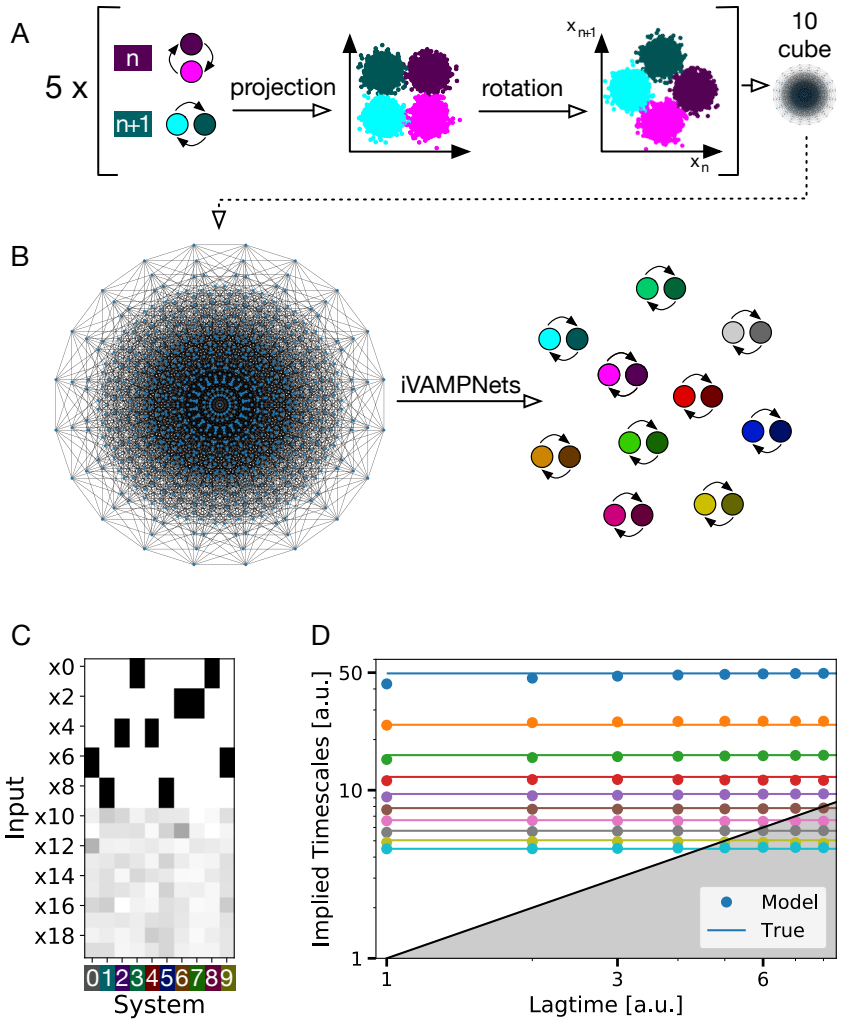

Figure 8.3.: Hidden Markov State model with 1024 global states building a 10D hypercube embedded in a 20D space.(**a**) The hypercube is composed of ten independent 2 state subsystems. A pair of 2 subsystems always lives in a common rotated 2D-manifold. Therefore, two subsystems need the same input features to be well approximated. Here, the pairwise VAMP-E score is maximized and the independence constraint Eq. 3.52 is enforced. (**b**) 2D depiction of the hypercube in an orthographic projection [121, 122], where the global system can jump freely between all 1024 vertices, and the ten 2-state models retrieved from it by the iVAMPnet. (**c**) Learned mask shows that for each subsystem, the network assigns 2 highly important input features which are shared with exactly one other subsystem, mirroring the rotated input space. Noise dimensions (x10-x19) are assigned low importance values. (**d**) Implied timescales of all 10 subsystems learned by our method (dots) approximate the underlying true timescales (lines).

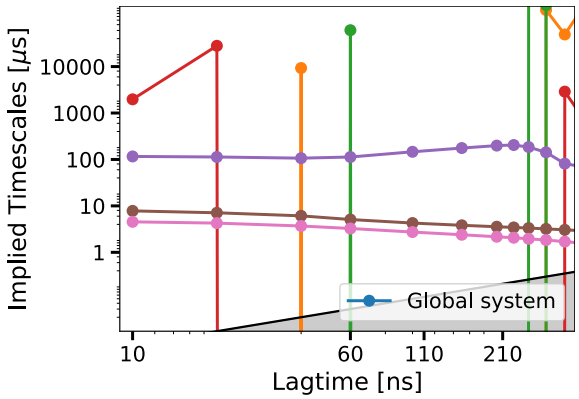

Figure 8.4.: Failed global implied timescales test with a classical VAMPNet of synaptotagmin with 8 output nodes. Since the model resolves processes which are not connected, the eigenvalues are ≈ 1, making the implied timescales estimation numerical unstable. It indicates that the amount of data is insufficient to build a global model. The model has no attention mechanism, but utilizes otherwise the same hyperparameters as the ones used in iVAMPNets.

structure it becomes obvious that the two subsystems are physically well separated (Fig. 8.5 a), supporting the conclusion that both regions are only weakly coupled [52].

The implied timescales of both systems are approximately constant in the model lag time τ. Most timescales are in the range of $1 - 10$ μs, with the exception of one much slower process with a 100 μs relaxation time found in the first subsystem (Fig. 8.5 b), which has not been found previously. Analysis of the structural changes governing this process reveals that it involves an orchestrated transition of all CBR loops (Fig. 8.5 c). Such a process could however not be resolved by the previous study [123] as the CBR was modeled as individual loops. The process of the second system involves a simultaneous movement of the C78 and C34 loops (Fig. 8.5 c).

We note that when analyzing the root mean squared fluctuation (RMSF) of the two subsystems, we find when considering only frames residing in the metastable state of the first subsystem comparably low fluctuations for the residues modeled by it, but higher RMSF in the other subsystem residues (Fig. 8.6). In other words, knowing the structure of one iVAMPnet subsystem does not allow one to predict the structure of the other subsystem, supporting the Markov independence assumption employed here.

Although estimating a global VAMPnet model for synaptotagmin was not feasible given the sparse data sample, iVAMPnets use the same data efficiently and estimate

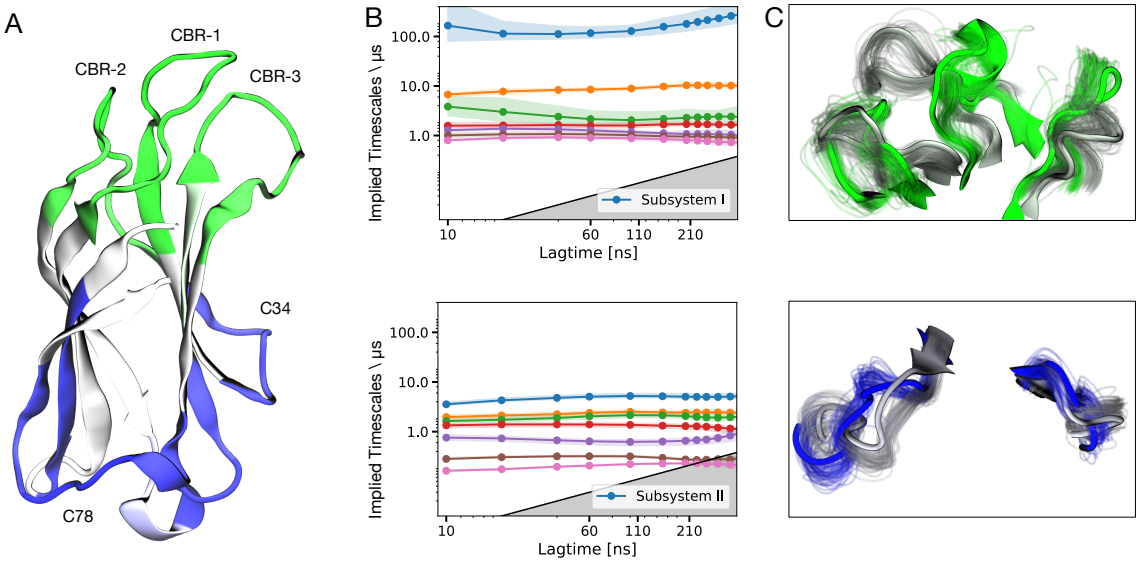

Figure 8.5.: iVAMPnets of synaptotagmin-C2A with 2 subsystems and 8 states each. **a)** Importance values of the trainable mask depicted as color-coded protein secondary structure, indicating assignment to subsystem I (II) in green (blue). **b)** Implied timescales of the 2 subsystems with a 90% percentile over 20 runs. **c)** Superposed representative structures of both extrema of the slowest resolved eigenfunctions of each subsystem (residues not assigned a high importance value or not showing significant movement are omitted for clarity). The slowest process of subsystem 1 changes between green and gray structures showing an orchestrated movement of full Calcium Binding Region (CBR1, CBR2, and CBR3). The slowest process of the second subsystem occurs between the blue and gray structures and describes a combined movement of C78 and C34.

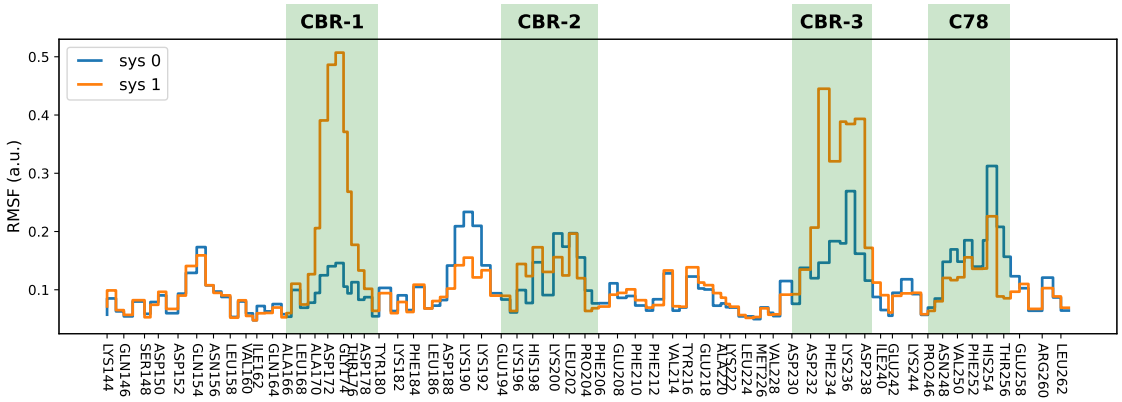

Figure 8.6.: Comparison of root mean square fluctuation (RMSF) in the two subsystems. We draw representative structures of minimum and maximum of the first singular functions of both sub-systems and compute the RMSF for each sub-system. We find that as expected, the RMSF is lower in areas that was modeled by the respective sub-model, i.e. CBR-i for subsystem 0 and C78 and C34 for subsystem 1.

a statistically valid dynamical model. This result is especially striking because the iVAMPnet approach also simplifies the subsequent task of interpreting models by separating dynamically independent protein domains.

8.1.4. Counterexample and post-training independence assessment

Finally, we conducted an experiment on the villin dataset [92] as a negative example. Small proteins such as villin are typically cooperative, i.e., the slowest processes related to folding involve all residues. Thus, these processes cannot be resolved when decomposing the system into several subsystems. If studied by a VAMPnet, the folding is recovered as the second slowest process. The slowest process describes a transition between a mis-folded and the folded state [118]. However, the majority of residues is involved in both of these processes making them ideal candidates for global processes. Thus, these processes cannot be resolved when decomposing the system into several subsystems.

For the analysis the same hyperparameters are used as for synaptotagmin but we choose only 2 states per subsystem. The resolved processes resemble a localized folding of either the left or right helix of the folded structure in each subsystem (Fig. 8.7). However, the implied timescales are not converged expressing a non-

Markovian behavior. The results imply that the network makes a compromise of learning nearly independent processes and approximating slow processes. Since the independence is strongly enforced the processes are badly approximated resulting in poor implied timescales. In order to interpret the iVAMPNets of villin, we have correlated their eigenfunctions with the ones of a standard, global VAMPNet (Fig. 8.8). We find that the process found by subsystem 1 has the highest Pearson correlation ($r = -0.79$) with the 2nd global process, which corresponds to peptide folding. However, the second subsystem cannot be clearly assigned to a global process. These results are not surprising since the poor implied timescales and the other independent scores (Tab. 8.1) reveal that the independence approximation does not hold in this example, i.e., the system expresses dynamics on the global level that are not or only poorly approximated by the described iVAMPNets.

8.1.5. Testing statistical independence of the learned dynamical subsystems

To assess the validity of an estimated subsystem assignment, we evaluate the constraints that were not enforced during training (Eq. 3.58-3.60) as post-training independence score. Low values for M_U and M_V imply that the constructed left and right singular functions $\hat{\mathbf{U}}^G$, $\hat{\mathbf{V}}^G$ are indeed valid candidates for singular functions in the global state space. A small value for M_{UV} indicates that the kinetics in the global state space is well predicted by the Kronecker product of subsystem models. They are computed for all test systems and presented in Tab. 8.1. Out of the tested systems only villin cannot be split into independent parts (all scores > 0.1). In comparison, the toy models and synaptotagmin can be decomposed into statistically uncoupled subsystems (all scores < 0.01). The slightly increased values for synaptotagmin suggest that its subsystems might be weakly coupled.

8.2. Methods

Code availability The iVAMPnets architecture, which is implemented using *PyTorch* [97], is depicted in Fig. 3.7. The details for the training routine, choice of hyper-parameters, and network architecture can be found in our GitHub repository. In general, we employ the *Adam* [111] optimizer with a batch size of 10000. We choose fully connected feed forward neural networks with 3 or 5 hidden layers with

	M_U	M_V	M_{UV}	M_R
Toy 2	0.0058(3)	0.0059(4)	0.0055(3)	0.0002(1)
Toy 10	0.0039(2)	0.0039(3)	0.0046(3)	0.00045(5)
Syt	0.0067(5)	0.0067(5)	0.0070(5)	0.0015(9)
Villin	0.135(3)	0.136(4)	0.149(3)	0.002(1)

Table 8.1.: Post-training independence validation. The scores in columns 1-3 (M_U, M_V, M_{UV}, cf. Eq. 3.58- 3.60) are computed from independence constraints that were not enforced during the training. The score in the last column (M_R) is used during the training and shown for reference. The three post-training validation scores M_U, M_V, and M_{UV} indicate that the final subsystems of both toy examples and synaptotagmin are indeed independent, whereas the scores for villin strictly oppose this conjunction. The indicated errors are standard deviations over 10 different runs.

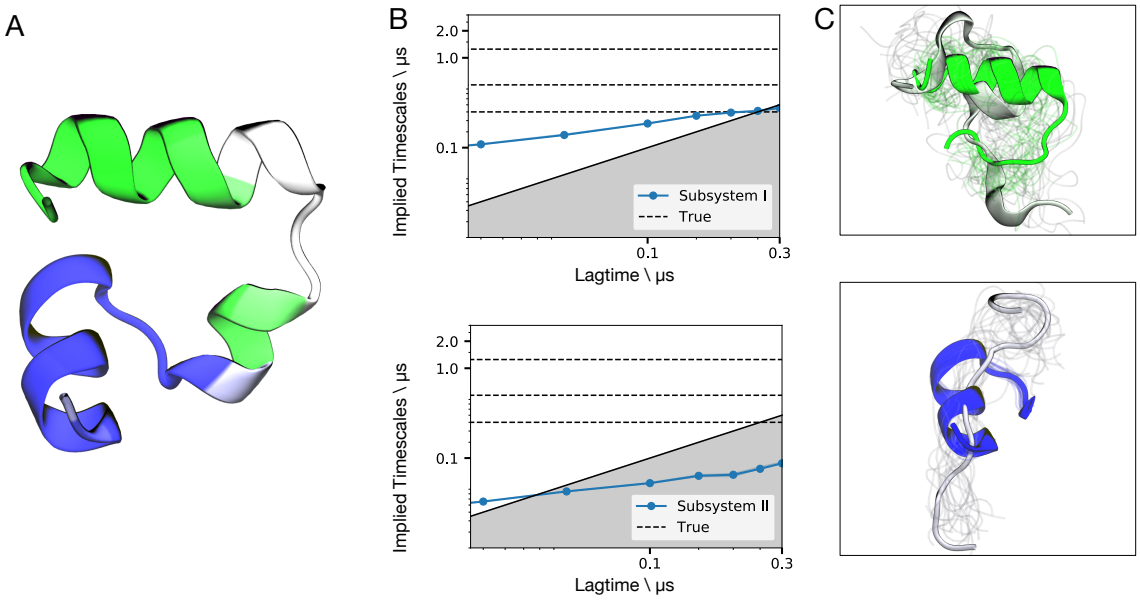

Figure 8.7.: Counter example to iVAMPNets using villin, trained with the independence constraint Eq. 3.52. (**a**) Subsystem assignment, i.e., masked importance values, are shown as color code on the folded structure. (**b**) Implied timescales of the 2 subsystems, the black dotted lines are reference timescales of a global model trained with a standard VAMPnet. (**c**) 20 representative structures of both extrema of the slowest resolved eigenfunctions for both subsystems. The processes tend to approximate formation of the N- and C-terminal helices, respectively.

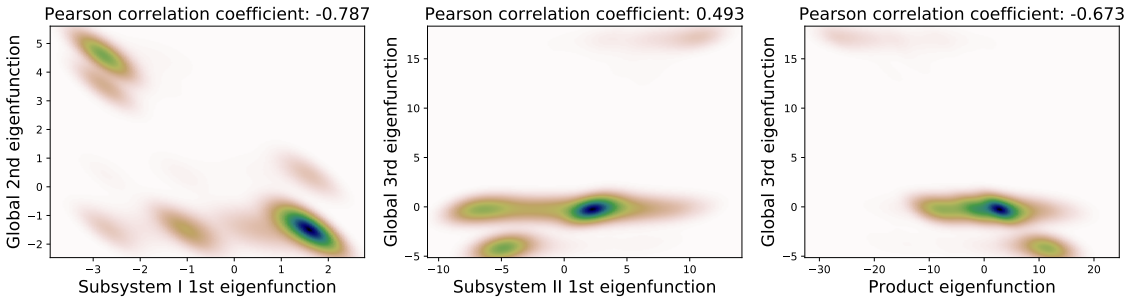

Figure 8.8.: Interpreting the processes found by the iVAMPnets for villin by correlating them with the global processes found with a standard VAMPnet. Shown are the processes with the highest correlation plotted against each other. The first subsystem correlates with the second global process, which relates to peptide folding. The second subsystem cannot clearly assigned to a global process. However, the product of the two eigenfunctions exhibits significant correlation with the third global eigenfunction.

a width of 30 or 100 nodes and the *ELU* [60] activation function for the toy models and the proteins, respectively.

Data availability The code that implements the presented models and reproduces the presented results can be found in our GitHub repository. The molecular dynamics data set of synaptotagmin C2A is available upon request. Restrictions apply to the availability of the villin data set, which were used under license for this study. Data are available from the authors upon reasonable request and with the permission of Lindorff-Larsen *et al.* [92].

8.3. Discussion

We have proposed a deep learning framework, iVAMPnet, that learns from trajectory data to decompose a complex dynamical system into subsystems which behave as approximately independent Markov models. Thereby, iVAMPnet is an end-to-end learning framework that points a way out of the exponentially growing demand for simulation data that is required to sample increasingly large biomolecular complexes.

Specifically, we have developed and demonstrated iVAMPnets for molecular dynamics, but the approach is, in principle, also applicable to different application areas, such as fluid dynamics. Of course the specific implementation, such as the representation of the input vectors $\mathbf{x}_t$ and the neural network architecture of the

$\boldsymbol{\chi}$-functions, depend on the application and will have to be tailored to suit new applications.

We now have a hierarchy of increasingly powerful models ranging from MSMs over VAMPnets to iVAMPnets: MSMs always consist of (1) a state space decomposition and (2) a Markovian transition matrix governing the dynamics between these states. VAMPnets provide a deep learning framework for MSMs, and additionally (3) learn the collective coordinates in which the state space discretization (1) is best made. iVAMPnets additionally learn (4) a physical separation of the molecular system into subsystems, each of which has its own slow coordinates, Markov states, and transition matrix.

We have demonstrated that iVAMPNets are a powerful multiscale learning method that succeeds in finding and modeling molecular subsystems when these subsystems indeed evolve statistically independently. Additionally, iVAMPnets are capable of learning from high dimensional MD data. To prove that point, we have demonstrated that the synaptotagmin C2A domain is decomposable into two almost independent Markov state models. Importantly, we have shown that this dynamical decomposition of synaptotagmin C2A succeeds while an attempt to model the system with a global VAMPnet fails due to poor sampling. This is a direct demonstration that iVAMPnets are statistically more efficient than VAMPnets, MSMs or other global-state models and may indeed scale to much larger systems.

We note, however, that iVAMPnets do not learn how the subsystems are coupled, and are therefore, in their current form, only applicable to molecular systems that consist of uncoupled or weakly coupled subsystems. Following up on Ref. [51] and introducing coupling parameters that describe how the learned MSMs are coupled, is subject to ongoing research.

Besides the usual hyperparameter choices in deep learning approaches, iVAMPnets require the specification of the number of sought subsystems. This choice can be guided by training iVAMPnets for different numbers of subsystems and then interrogating the independence scores (Eq. 3.58-3.61) to choose a decomposition where statistical independence is optimal. Furthermore, this choice can be guided by the number of structural domains or by using the network-based approach presented in Ref. [52].

iVAMPnets can be improved and further developed in multiple ways, e.g. by employing more advanced network architectures, e.g. graph neural networks, where parameters could be shared across subsystems. This might result in higher quality

models and a greater robustness against the hyperparameter choice.

In summary, iVAMPnets pave a possible path for modeling the kinetics of large biological systems in a data-efficient and interpretable manner.

Acknowledgements We acknowledge financial support from Deutsche Forschungsgemeinschaft DFG (SFB/TRR 186, Project A12; SFB 1114, Projects C03 and A04; SFB 1078, Project C7, and RTG 2433), the European Commission (ERC CoG 772230 "ScaleCell"), the Berlin Mathematics center MATH+ (AA1-6 and AA1-10), the BMBF (Research center BIFOLD), the National Science Foundation (CHE-1738990, CHE-1900374, and PHY-2019745), the Welch Foundation (C-1570), and the Einstein Foundation Berlin. We further thank Manuel Dibak and Moritz Hoffmann (FU Berlin) for fruitful discussions.

9. Summary and Perspective

This thesis shows how Koopman theory and deep learning can be combined to produce a robust data-driven framework to approximate the dynamics of complex systems, in specific the long-time dynamics of molecular processes. The proposed methods are based on the variational approach for Markov Processes [18] and allow the approximation of the Koopman operator while incorporating different levels of physical constraints. The methods are validated by applying them to toy models, and they are further tested into real world molecular systems.

Summary The most general method presented is the VAMPnet, where the dynamics are approximated by feature state assignments in conjunction with the Koopman matrix. We applied the method to systems ranging from small toymodels to protein folding, where the results are competitive or even outperform state-of-the art Markov modeling approaches. It can further be used to approximate both non-reversible and non-stationary processes. Despite being a powerful method, it has several shortcomings: there exists no back-mapping from the state space to the configuration space since it does not explicitly model the transition density; the Koopman matrix cannot be interpreted as a real transition matrix, restricting the post-analysis tools; and it does not necessarily yield a reversible model even if the underlying data generating process is reversible. However, even with these shortcomings, VAMPnets has proven crucial as a pretraining step for more advanced models.
The first two shortcomings are alleviated by constructing an additional transition matrix for the state space and by modeling the transition density explicitly, we called this method deep MSM. It can be further enhanced by adding a generative model producing new unseen configurations. The method is suitable mainly for non-reversible systems, where subsequent analysis tools demand true transition probabilities, e.g. transition path theory. Our applications to a small molecule shows that deep MSMs manage to model the kinetics with similar accuracy but with far

less states compared to an ordinary Markov modeling approach, thus staying easily interpretable. Furthermore, the generated molecule structures exhibit physically accurate bond and angle distributions.
The last shortcoming is addressed by parameterizing the stationary distribution and the transition matrix, which allows reversibility to be enforced. The constraints in the parametrization can be adapted to not only mimic the previous methods, but also build a reversible VAMPnet or reversible deep MSM. By applying it to a small toy model, we showed that, even in the case of highly biased data, our method is able to recover the ground truth. When applied to protein-folding and compared to ordinary MSMs, our model achieves comparable results with far less states.
The architecture of deep reversible MSM further allows to include experimental evidence as constraints. By updating the loss function to incorporate experimental measurements, we showed for a large peptide that the method is able to counter an artificial bias stemming from the simulation. Moreover, we transferred the idea of coarse-graining from MSMs to the deep learning of Koopman models. This is achieved by incorporating an additional coarse-graining layer, which is applicable to all of the previous methods. Applied to the same large peptide, we demonstrated how the coarse-graining facilitates the analysis at different degrees of resolution. The interpretation of the model can be further simplified by the proposed attention mechanism, which marks the dynamical important residues of each particular state.
The final method presented paves the way to solve the fundamental scaling problem in MD by building independent VAMPnets. Instead of modeling the global states directly, the method constructs the global states and operator as the product of its independent subsystem states and operators. At the same time it partitions the input features via a trainable mask into independent regions that are easily interpretable. The applications ranged from toy systems composed by many independent subsystems up to a protein, namely the C2A domain of synaptotagmin, where we were able to identify two independent regions. We showed that the proposed scores are able to indicate independence and that the method is more data efficient than a standard global VAMPnet.

Critic and current problems Despite the success of the presented methods, they still have to prove advantageous when faced with much larger systems. For instance, in the example of synaptotagmin, the available data is too sparse to build a global

model; only by employing iVAMPnets can we approximate the long-time dynamics. However, if the system of interest cannot be decomposed into independent subsystems, the application of iVAMPnets is also ruled out. Moreover, if the system is composed of coupled subsystems, the method can only be a part of a final solution, since it misses the possibility to model coupling. This will be the task of future work and might include trainable coupling parameters as presented in Olsson and Noé [51].

Another issue with iVAMPnets current implementation is the lack of capability to include constraints on the transition matrix or reversibility. This is not a complex task because one only needs to modify the independence constraints. However, these have to be implemented individually for each method; a task left for future work.

Another concern regards the deep generative MSM, which approximates the sampling from the transition density. The main issue with this approach is the need for simulation data at the training stage, which might render the trained generative model obsolete. However, it might turn out beneficial if it predicts samples that are likely according to the Boltzmann distribution but stem from an unobserved subspace of the configuration space. This would allow to start new simulations from these regions and accelerate the exploration of the phase space.

However, this requires that the model has some physical intuition where these regions might be. This knowledge would need to be acquired and transferred from the same system with different simulation conditions or even from other systems. Alternative models, which try to generate unseen data, comprise the Boltzmann generators [74] or coarse-graining MD approaches [126]. The first manages in principle to approximate the Boltzmann distribution without training data but neglect the dynamics. The second can accelerate the sampling of the dynamics but needs sufficient training data. However, it is more straightforward how to make it transferable and therefore predictive.

Finally, although we showed that the inclusion of artificial experimental evidence can improve the accuracy of predicted observables, we did not show the benefits when confronted with real experimental values and true simulation biases stemming from the underlying force field. There, the usefulness of this approach remains to be proven.

Possible improvements The application of more advanced architectures for the common neural network $\boldsymbol{\eta}$ has the greatest potential to improve the model estimation for all methods. The recent developments in neural network architecture such as new activation functions (ELU [60]) and structures (residual blocks [112]) has increased the success rates of VAMPnets to nearly 100 % from the stated 40 % in the paper. Furthermore, smart initialization strategies can help significantly. In the case of deep MSMs, it turned out crucial to initialize the biases of the last layer of $\boldsymbol{\gamma}$ with a positive value, so that the mean value of each entry is invertible (necessary for the computation of the training score).
Another recent discovery of the so called double descent [127] questions the use of early stopping in the presented methods. They observed that a model might first overfit the training data - letting the validation loss increase and the early stopping therefore end the training - before it descents a second time and starts to generalize much better on the validation set. An extended optimizer [128] has shown to accelerate the second descent most rapidly compared to other regularization schemes [129]. Therefore, a possible improvement would be to remove the early stopping and apply this optimizer.
When applying the methods to proteins a natural choice of advanced architecture are graph neural networks (GNN), which resemble the graph structure of molecules [62, 66, 130]. Instead of modeling each atom individually, a node could represent a whole amino acid and the attached features might be the amino type and the position in the residue chain. In this case, the features would be assembled to messages passing between the nodes, which might additionally depend on the distance between the two involved amino acids or the orientation of the side chains. Finally, these incoming messages would be collected to update the node features. The message creation and node updating is usually performed by neural networks, where the parameters are shared between the different nodes or even between molecules. Thereby, the whole architecture is very parameter efficient and possibly transferable across chemical space [66]. When applying these to VAMPnets, the GNN would output features for each amino acid based on which a final system specific layer could assign states. The GNN could be shared across a set of proteins hoping to find universal features for the description of the dynamics of amino acids. Training a VAMPnet for a new protein would then only involve the optimization of a last layer to assign states based on the features extracted for each amino acid.
While this transferable approach needs to be validated by future work, the applica-

tion of GNNs will be definitely beneficial in the context of larger proteins. In these cases the number of parameters of a fully connected feed forward network (used in all the presented work here) would scale quadratically with the number of residues making it unscalable (assuming amino acid distances as inputs). Instead, since the parameters are shared across all residues for GNNs, the number of parameters remains constant with increasing residue numbers.

Finally, another possible extension for the reversible Koopman model could include the multi-ensemble Markov model approach, where kinetic models are built taking into account the information of simulations with different bias potentials or at different temperatures, allowing a faster exploration of the configuration space [131]. The state assignment $\boldsymbol{\chi}(\mathbf{x}_t)$ would be shared across the ensemble, but the reweighting for the stationary distribution of the ensemble would need to follow the updated Boltzmann distribution, which is modified by temperature changes or biasing potentials. The transition matrix similarly to the ordinary MSM approach would need to be trained individually. Thereby, possible processes happening on timescales inaccessible by simulations at normal conditions become feasible while all the collected data can be exploited for model estimation.

The applications in this thesis are solely in protein dynamics, although the methods are not restricted to them. Possible applications to other complex dynamical systems, such as climate, weather, or ocean currents systems, are intriguing and can yield new insights and improve predictive power in these fields. However, I see especially great potential within the protein dynamics domain, since it enables the automatic building of high quality models within a high-throughput screening framework of drug candidates with MD. This might ultimately turn out to be a crucial step in accelerating the process of drug discovery and in making it less costly.

Bibliography

[1] Clarence W. Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

[2] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[3] Jonathan H. Tu. *Dynamic mode decomposition: Theory and applications*. PhD thesis, Princeton University, 2013.

[4] John D. Chodera, Nina Singhal, Vijay S. Pande, Ken A. Dill, and William C. Swope. Automatic discovery of metastable states for the construction of Markov models of macromolecular conformational dynamics. *The Journal of chemical physics*, 126(15):04B616, 2007.

[5] Gregory R. Bowman, Kyle A. Beauchamp, George Boxer, and Vijay S. Pande. Progress and challenges in the automated construction of Markov state models for full protein systems. *The Journal of chemical physics*, 131(12):124101, 2009.

[6] Gary Froyland and Kathrin Padberg-Gehle. Almost-invariant and finite-time coherent sets: directionality, duration, and diffusion. In *Ergodic Theory, Open Dynamics, and Coherent Structures*, pages 171–216. Springer, 2014.

[7] Lorenzo Boninsegna, Gianpaolo Gobbo, Frank Noé, and Cecilia Clementi. Investigating molecular kinetics by variationally optimized diffusion maps. *Journal of chemical theory and computation*, 11(12):5947–5960, 2015.

[8] Bernard O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the national academy of sciences of the united states of america*, 17(5):315, 1931.

[9] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynam.*, 41:309–325, 2005.

[10] Ch Schütte, Alexander Fischer, Wilhelm Huisinga, and Peter Deuflhard. A direct approach to conformational dynamics based on hybrid Monte Carlo. *Journal of Computational Physics*, 151(1):146–168, 1999.

[11] Michael Dellnitz, Gary Froyland, and Oliver Junge. The algorithms behind gaio-set oriented numerical methods for dynamical systems. In *Ergodic theory, analysis, and efficient simulation of dynamical systems*, pages 145–174. Springer, 2001.

[12] Jan-Hendrik Prinz, Hao Wu, Marco Sarich, Bettina Keller, Martin Senne, Martin Held, John D. Chodera, Christof Schütte, and Frank Noé. Markov models of molecular kinetics: Generation and validation. *The Journal of chemical physics*, 134(17):174105, 2011.

[13] Hao Wu and Frank Noé. Variational approach for learning Markov processes from time series data. *Journal of Nonlinear Science*, Aug 2019.

[14] Feliks Nüske, Bettina G. Keller, Guillermo Pérez-Hernández, Antonia S. J. S. Mey, and Frank Noé. Variational approach to molecular kinetics. *Journal of chemical theory and computation*, 10(4):1739–1752, 2014.

[15] Frank Noé and Feliks Nüske. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Modeling & Simulation*, 11(2):635–655, 2013.

[16] Hao Wu, Feliks Nüske, Fabian Paul, Stefan Klus, Péter Koltai, and Frank Noé. Variational Koopman models: slow collective variables and molecular kinetics from short off-equilibrium simulations. *The Journal of chemical physics*, 146(15):154104, 2017.

[17] Frank Noé and Cecilia Clementi. Kinetic distance and kinetic maps from molecular dynamics simulation. *Journal of Chemical Theory and Computation*, 11(10):5002–5011, 2015.

[18] Hao Wu and Frank Noé. Variational approach for learning Markov processes from time series data. *Journal of Nonlinear Science*, 30(1):23–66, 2020.

[19] Guillermo Pérez-Hernández, Fabian Paul, Toni Giorgino, Gianni De Fabritiis, and Frank Noé. Identification of slow molecular order parameters for Markov model construction. *The Journal of chemical physics*, 139(1):07B604_1, 2013.

[20] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[21] Christian R. Schwantes and Vijay S. Pande. Modeling molecular kinetics with tICA and the kernel trick. *Journal of chemical theory and computation*, 11(2):600–608, 2015.

[22] Stefan Klus, Brooke E Husic, Mattes Mollenhauer, and Frank Noé. Kernel methods for detecting coherent structures in dynamical data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12):123112, 2019.

[23] Le Song, Kenji Fukumizu, and Arthur Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111, 2013.

[24] Feliks Nüske, Reinhold Schneider, Francesca Vitalini, and Frank Noé. Variational tensor approach for approximating the rare-event kinetics of macromolecular systems. *The Journal of chemical physics*, 144(5):054105, 2016.

[25] Stefan Klus, Patrick Gelß, Sebastian Peitz, and Christof Schütte. Tensor-based dynamic mode decomposition. *Nonlinearity*, 31(7):3359, 2018.

[26] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[27] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[28] Krizhevsky Alex, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional networks. In *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems*. Curran Associates, Inc., 2012.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[30] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[31] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kale, and Klaus Schulten. Scalable molecular dynamics with namd. *Journal of computational chemistry*, 26(16):1781–1802, 2005.

[32] David E. Shaw, Paul Maragakis, Kresten Lindorff-Larsen, Stefano Piana, Ron O. Dror, Michael P. Eastwood, Joseph A. Bank, John M. Jumper, John K. Salmon, Yibing Shan, et al. Atomic-level characterization of the structural dynamics of proteins. *Science*, 330(6002):341–346, 2010.

[33] Peter Eastman, Mark S. Friedrichs, John D. Chodera, Randall J. Radmer, Christopher M. Bruns, Joy P. Ku, Kyle A. Beauchamp, Thomas J. Lane, Lee-Ping Wang, Diwakar Shukla, et al. OpenMM 4: a reusable, extensible, hardware independent library for high performance molecular simulation. *Journal of chemical theory and computation*, 9(1):461–469, 2013.

[34] Romelia Salomon-Ferrer, Andreas W. Gotz, Duncan Poole, Scott Le Grand, and Ross C Walker. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 2. explicit solvent particle mesh ewald. *Journal of chemical theory and computation*, 9(9):3878–3888, 2013.

[35] Sander Pronk, Szilárd Páll, Roland Schulz, Per Larsson, Pär Bjelkmar, Rossen Apostolov, Michael R. Shirts, Jeremy C. Smith, Peter M. Kasson, David Van Der Spoel, et al. Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 29(7):845–854, 2013.

[36] Weinan E. and Eric Vanden-Eijnden. Towards a theory of transition paths. *Journal of statistical physics*, 123(3):503–523, 2006.

[37] Philipp Metzner, Christof Schütte, and Eric Vanden-Eijnden. Transition path theory for Markov jump processes. *Multiscale Modeling & Simulation*, 7(3):1192–1219, 2009.

[38] Frank Noé, Christof Schütte, Eric Vanden-Eijnden, Lothar Reich, and Thomas R Weikl. Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations. *Proceedings of the National Academy of Sciences*, 106(45):19011–19016, 2009.

[39] Peter Deuflhard, Wilhelm Huisinga, Alexander Fischer, and Ch Schütte. Identification of almost invariant aggregates in reversible nearly uncoupled Markov chains. *Linear Algebra and its Applications*, 315(1-3):39–59, 2000.

[40] Peter Deuflhard and Marcus Weber. Robust Perron cluster analysis in conformation dynamics. *Linear algebra and its applications*, 398:161–184, 2005.

[41] Konstantin Fackeldey and Marcus Weber. Genpcca-Markov state models for non-equilibrium steady states. *WIAS Rep*, 29:70–80, 2017.

[42] Jan-Hendrik Prinz, Bettina Keller, and Frank Noé. Probing molecular kinetics with Markov models: metastable states, transition pathways and spectroscopic observables. *Physical Chemistry Chemical Physics*, 13(38):16912–16927, 2011.

[43] Simon Olsson, Hao Wu, Fabian Paul, Cecilia Clementi, and Frank Noé. Combining experimental and simulation data of molecular processes via augmented Markov models. *Proceedings of the National Academy of Sciences*, 114(31):8265–8270, 2017.

[44] Gerhard Hummer and Jürgen Köfinger. Bayesian ensemble refinement by replica simulations and reweighting. *The Journal of chemical physics*, 143(24):12B634_1, 2015.

[45] Jed W. Pitera and John D. Chodera. On the use of experimental observations to bias simulated ensembles. *Journal of chemical theory and computation*, 8(10):3445–3451, 2012.

[46] Wouter Boomsma, Jesper Ferkinghoff-Borg, and Kresten Lindorff-Larsen. Combining experiments and simulations using the maximum entropy principle. *PLoS computational biology*, 10(2):e1003406, 2014.

[47] Kyle A. Beauchamp, Vijay S. Pande, and Rhiju Das. Bayesian energy landscape tilting: Towards concordant models of molecular ensembles. *Biophysical journal*, 106(6):1381–1390, 2014.

[48] Andrea Cavalli, Carlo Camilloni, and Michele Vendruscolo. Molecular dynamics simulations with replica-averaged structural restraints generate structural ensembles according to the maximum entropy principle. *The Journal of chemical physics*, 138(9):03B603, 2013.

[49] Hoi Tik Alvin Leung, Olivier Bignucolo, Regula Aregger, Sonja A. Dames, Adam Mazur, Simon Berneche, and Stephan Grzesiek. A rigorous and efficient method to reweight very large conformational ensembles using average experimental data and to determine their relative information content. *Journal of chemical theory and computation*, 12(1):383–394, 2016.

[50] Kirill A. Konovalov, Ilona Christy Unarta, Siqin Cao, Eshani C. Goonetilleke, and Xuhui Huang. Markov state models to study the functional dynamics of proteins in the wake of machine learning. *JACS Au*, 2021.

[51] Simon Olsson and Frank Noé. Dynamic graphical models of molecular kinetics. *Proceedings of the National Academy of Sciences*, 116(30):15001–15006, 2019.

[52] Tim Hempel, Mauricio J. del Razo, Christopher T. Lee, Bryn C. Taylor, Rommie E. Amaro, and Frank Noé. Independent Markov decomposition: Towards modeling kinetics of biomolecular complexes. *bioRxiv*, 2021.

[53] Bernard O. Koopman. Hamiltonian systems and transformations in Hilbert space. *Proc. Natl. Acad. Sci. USA*, 17:315–318, 1931.

[54] Robert T. McGibbon and Vijay S. Pande. Variational cross-validation of slow dynamical modes in molecular kinetics. *J. Chem. Phys.*, 142:124105, 2015.

[55] Stefan Klus, Feliks Nüske, Sebastian Peitz, Jan-Hendrik Niemann, Cecilia Clementi, and Christof Schütte. Data-driven approximation of the Koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena*, 406:132416, 2020.

[56] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[57] Kenji Kawaguchi and Yoshua Bengio. Depth with nonlinearity creates no bad local minima in ResNets. *Neural Networks*, 118:167–174, 2019.

[58] Kenji Kawaguchi and Leslie Kaelbling. Elimination of all bad local minima in deep learning. In *International Conference on Artificial Intelligence and Statistics*, pages 853–863. PMLR, 2020.

[59] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Icml*, 2010.

[60] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[61] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[62] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[63] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[64] Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2. IEEE, 2018.

[65] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.

[66] Kristof T. Schütt, Huziel E. Sauceda, P.-J. Kindermans, Alexandre Tkatchenko, and K.-R. Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[67] Kristof T. Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. *arXiv preprint arXiv:2102.03150*, 2021.

[68] Brandon Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *arXiv preprint arXiv:1906.04015*, 2019.

[69] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. Se (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *arXiv preprint arXiv:2101.03164*, 2021.

[70] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[71] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[72] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[73] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

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

[75] Gábor J. Székely, Maria L. Rizzo, et al. Testing for equal distributions in high dimension. *InterStat*, 5(16.10):1249–1272, 2004.

[76] Hao Wu, Andreas Mardt, Luca Pasquali, and Frank Noé. Deep generative Markov state models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[77] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. Vampnets: Deep learning of molecular kinetics. *Nat. Commun.*, 9:5, 2018.

[78] Andreas Mardt, Luca Pasquali, Frank Noé, and Hao Wu. Deep learning Markov and Koopman models with physical constraints. In *Mathematical and Scientific Machine Learning*, pages 451–475. PMLR, 2020.

[79] Andreas Mardt and Frank Noé. Progress in deep Markov state modeling: Coarse graining and experimental data restraints. *The Journal of Chemical Physics*, 155(21):214106, 2021.

[80] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

[81] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[82] Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision*, pages 3429–3437, 2017.

[83] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[84] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[85] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[86] Anders Krogh, Jesper Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7:231–238, 1995.

[87] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. *Encyclopedia of database systems*, 5:532–538, 2009.

[88] Marco Sarich, Frank Noé, and Christof Schütte. On the approximation quality of Markov state models. *Multiscale Modeling & Simulation*, 8(4):1154–1177, 2010.

[89] David M. Allen. The relationship between variable selection and data agumentation and a method for prediction. *technometrics*, 16(1):125–127, 1974.

[90] William C. Swope, Jed W. Pitera, and Frank Suits. Describing protein folding kinetics by molecular dynamics simulations. 1. theory. *The Journal of Physical Chemistry B*, 108(21):6571–6581, 2004.

[91] Ralf Banisch and Péter Koltai. Understanding the geometry of transport: Diffusion maps for Lagrangian trajectory data unravel coherent sets. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(3):035804, 2017.

[92] Kresten Lindorff-Larsen, Stefano Piana, Ron O. Dror, and David E. Shaw. How fast-folding proteins fold. *Science*, 334:517–520, 2011.

[93] Feliks Nüske, Hao Wu, Jan-Hendrik Prinz, Christoph Wehmeyer, Cecilia Clementi, and Frank Noé. Markov state models from short non-equilibrium simulations-analysis and correction of estimation bias. *The Journal of Chemical Physics*, 146(9):094104, 2017.

[94] Frank Noé and Cecilia Clementi. Kinetic distance and kinetic maps from molecular dynamics simulation. *J. Chem. Theory Comput.*, 11:5002–5011, 2015.

[95] François Chollet et al. Keras: The python deep learning library. *Astrophysics Source Code Library*, pages ascl–1806, 2018.

[96] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. *arXiv.org:1603.04467*, 2015.

[97] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[98] Moritz Hoffmann, Martin Scherer, Tim Hempel, Andreas Mardt, Brian de Silva, Brooke E. Husic, Stefan Klus, et al. Deeptime: a python library for machine learning dynamical models from time series data. *Machine Learning: Science and Technology*, 3(1):015009, 2021.

[99] Hao Wu, Fabian Paul, Christof Wehmeyer, and Frank Noé. Multiensemble Markov models of molecular thermodynamics and kinetics. *Proc. Natl. Acad. Sci. USA*, 113(23):E3221–E3230, 2016.

[100] Hao Wu, Antonia S. J. S. Mey, Edina Rosta, and Frank Noé. Statistically optimal analysis of state-discretized trajectory data from multiple thermodynamic states. *J. Chem. Phys.*, 141:214106, 2014.

[101] John D. Chodera, William C. Swope, Frank Noé, Jan-Hendrik Prinz, and Vijay S. Pande. Dynamical reweighting: Improved estimates of dynamical properties from simulations at multiple temperatures. *J. Phys. Chem.*, 134:244107, 2011.

[102] Jan-Hendrik Prinz, John D. Chodera, Vijay S. Pande, William C. Swope, J. C. Smith, and Frank Noé. 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.

[103] Edina Rosta and Gerhard Hummer. Free energies from dynamic weighted histogram analysis using unbiased Markov state model. *J. Chem. Theory Comput.*, 11:276–285, 2015.

[104] Antonia S. J. S. Mey, Hao Wu, and Frank Noé. xTRAM: Estimating equilibrium expectations from time-correlated simulation data at multiple thermodynamic states. *Phys. Rev. X*, 4:041018, 2014.

[105] Nina S. Hinrichs and Vijay S. Pande. Calculation of the distribution of eigenvalues and eigenvectors in Markovian state models for molecular dynamics. *J. Chem. Phys.*, 126:244101, 2007.

[106] Frank Noé. Probability Distributions of Molecular Observables computed from Markov Models. *J. Chem. Phys.*, 128:244103, 2008.

[107] John D. Chodera and Frank Noé. Probability distributions of molecular observables computed from Markov models. ii: Uncertainties in observables and their time-evolution. *J. Chem. Phys.*, 133:105102, 2010.

[108] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324, 1998.

[109] Kristoff T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller. Moleculenet: A continuous-filter convolutional neural network for modeling quantum interactions. *arXiv:1706.08566*, 2017.

[110] Frank Noé and Feliks Nüske. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Model. Simul.*, 11:635–655, 2013.

[111] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[112] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[113] Wei Chen, Hythem Sidky, and Andrew L. Ferguson. Nonlinear discovery of slow molecular modes using state-free reversible vampnets. *The Journal of Chemical Physics*, 150(21):214114, Jun 2019.

[114] Hao Wu, Frank Nüske, Fabian Paul, Stefan Klus, Peter Koltai, and Frank Noé. Variational Koopman models: slow collective variables and molecular kinetics from short off-equilibrium simulations. *J. Chem. Phys.*, 146:154104, 2017.

[115] Martin K Scherer, Benjamin Trendelkamp-Schroer, Fabian Paul, Guillermo Pérez-Hernández, Moritz Hoffmann, Nuria Plattner, Christoph Wehmeyer, Jan-Hendrik Prinz, and Frank Noé. PyEMMA 2: A software package for estimation, validation and analysis of Markov models. *J. Chem. Theory Comput.*, 11:5525–5542, 2015.

[116] Guillermo Pérez-Hernández, Fabian Paul, Toni Giorgino, Gianni De Fabritiis, and Frank Noé. Identification of slow molecular order parameters for Markov model construction. *J. Chem. Phys.*, 139:015102, 2013.

[117] Simon Olsson, Hao Wu, Fabian Paul, Cecilia Clementi, and Frank Noé. Combining experimental and simulation data of molecular processes via augmented Markov models. *Proc. Natl. Acad. Sci. USA*, 114:8265–8270, 2017.

[118] Brooke E. Husic and Frank Noé. Deflation reveals dynamical structure in nondominant reaction coordinates. *The Journal of Chemical Physics*, 151(5):054103, 2019.

[119] Jan-Hendrik Prinz, John D Chodera, Vijay S Pande, William C Swope, Jeremy C Smith, and Frank Noé. Optimal use of data in parallel tempering simulations for the construction of discrete-state Markov models of biomolecular dynamics. *The Journal of chemical physics*, 134(24):06B613, 2011.

[120] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:257–286, 1989.

[121] Inc. Wolfram Research. Mathematica, Version 11.2.0, 2017.

[122] Aric Hagberg, Pieter Swart, and Daniel S. Chult. Exploring network structure, dynamics, and function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11–15, Pasadena, CA USA, 2008.

[123] Tim Hempel, Nuria Plattner, and Frank Noé. Coupling of conformational switches in calcium sensor unraveled with local Markov models and transfer entropy. *Journal of chemical theory and computation*, 16(4):2584–2593, 2020.

[124] Thomas C. Südhof. Neurotransmitter release: The last millisecond in the life of a synaptic vesicle. *Neuron*, 80:675–690, 2013.

[125] José L. Jiménez, Graham R. Smith, Bruno Contreras-Moreira, John G. Sgouros, Frederic A. Meunier, Paul A. Bates, and Giampietro Schiavo. Functional Recycling of C2 Domains Throughout Evolution: A Comparative Study of Synaptotagmin, Protein Kinase C and Phospholipase C by Sequence, Structural and Modelling Approaches. *J. Mol. Biol.*, 333(3):621–639, October 2003.

[126] Brooke E. Husic, Nicholas E. Charron, Dominik Lemm, Jiang Wang, Adrià Pérez, Maciej Majewski, Andreas Krämer, Yaoyi Chen, Simon Olsson, Gianni de Fabritiis, et al. Coarse graining molecular dynamics with graph neural networks. *The Journal of Chemical Physics*, 153(19):194101, 2020.

[127] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[128] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[129] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[130] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[131] Hao Wu, Fabian Paul, Christoph Wehmeyer, and Frank Noé. Multiensemble Markov models of molecular thermodynamics and kinetics. *Proceedings of the National Academy of Sciences*, 113(23):E3221–E3230, 2016.

A. Zusammenfassung

In meiner Dissertation habe ich gezeigt wie Techniken des Deep Learnings benutzt werden können um die Kinetik von biomolekularen Prozessen zu modelieren. Hierbei nutzen wir den variationellen Ansatz für Markov Prozesse (VAMP) um mit Hilfe von neuronalen Netzen die Singulärfunktionen des Koopman-Operators zu approximieren. Dies erlaubt uns den schrittweisen Prozess zur Schätzung eines Markov State Models (MSM) durch einen automatischen Ansatz zu ersetzen. Darauf aufbauend haben wir fortgeschrittene Techniken des MSM Bauens in die Welt der neuronalen Netzwerke transferiert. Wir haben demonstriert, (i) wie experimentelle Evidenz in die Modellschätzung einfliessen kann und systematischen Bias der Simulation ausbalanzieren kann, (ii) wie die Architektur des Neuronalen Netzwerkes angepasst werden kann, um einen reversiblen Operator zu schätzen, (iii) wie ein hierarchisches Model durch das Einführen von Coarse-Graining Layern möglich wird, (iv) wie ein Model trainiert werden kann um unabhängige Subsysteme zu finden und deren Dynamik zu schätzen, was im Falle der (beinahen) Unabhängigkeit zu einer deutlichen Dateneffizienz führen kann.
Zum Schluss meiner Arbeit spreche ich ausstehende Probleme an. Diese umfassen die Anwendung von spezialisierten Netzwerkarchitekturen, die mit deutlich weniger Parametern qualitativ hochwertige Modelle versprechen und Transferierbarkeit ermöglichen. Des Weiteren steht es aus, den reversiblen Ansatz auf die Zerlegung in unabhängige Systeme auszuweiten. Ferner fehlt die Möglichkeit schwache Kopplung zwischen den Systemen explizit zu modelieren. Zuletzt empfehle ich den Multiensemble Ansatz für MSMs auf die reversiblen Koopman Modelle anzuwenden, wodurch es möglich wird Daten aus Simulation mit verschiedenen Bias-Potentialen oder unterschiedlichen Temperaturen zusammenzuführen.
Ich bin überzeugt, dass die aufgezeigten Methoden helfen können automatisch kinetische Modelle zu schätzen, wodurch sie in einem Screening-Prozess für die Medikamentenentwicklung einsetzbar sind. Ferner bin ich überzeugt, dass sie sich auch für die Anwendung auf klimatische, ozeanische und athmosphärische Daten eignen und neue Erkenntnisse hervorbringen können.

Academic Education

Nov 2016 - Now	PhD student, **Freie Universität Berlin** Advisor: Frank Noé Topics of interest: Deep learning, chemical physics, molecular kinetics
Oct 2014 - Sep 2016	Master of Science in Physics, **Freie Universität Berlin** Specialisation: Computational Biophysics Thesis: "Uracil-Guanine Mispairs in DNA" Graduation: September 2016 Grade: average A+ (1.2 in German system)
Oct 2011 - Sep 2014	Bachelor of Science in Physics, **Georg August Universität** Thesis: “The Energetics in Energy Conversion in F_1-ATPase” Grade: average A (1.5 in German system)

Work & Professional Experience

Sep 2019 - Dec 2019	Participant, **IPAM Long Program**, UCLA "Machine Learning for Physics and the Physics of Learning" Workshops, Seminars, and Poster presentations
May 2016 - Oct 2016	Intern, **Fraunhofer IPK**, Berlin Institute for Production facilities and Construction Technology Software developement in the Area of Machine Controlling
May 2016 - Sep 2016	Scientific Assistant, **Freie Universität Berlin**, Berlin Institut for Computational Biophysics
Oct 2015 - Mar 2016	Research Stay, **College of Technology**, New York within the Research Phase of the Master’s Thesis Conception and Coding of Analysis Tools for Simulations

Awards

Jan 2016 - Mar 2016	PROMOS-Scholarship for Research Stay, New York
Oct 2015 - Dec 2015	PROMOS-Scholarship for Research Stay, New York

Publications

* shared first author

Mardt*, A., Hempel, T., Clementi, C. & Noé, F. (2021).
Deep learning to decompose macromolecules into independent Markovian domains.
bioRxiv.

Mardt, A. & Noé, F. (2021).
Progress in deep Markov State Modeling: Coarse graining and experimental data restraints.
The Journal of Chemical Physics.

Mardt, A., Pasquali, L., Noé, F.& Wu, H. (2020).
Deep learning Markov and Koopman models with physical constraints.
Mathematical and Scientific Machine Learning.

Wu, H., Mardt*, A., Pasquali, L., & Noé, F. (2018).
Deep Generative Markov State Models.
Advances in Neural Information Processing Systems., 3975-3984.

Mardt*, A., Pasquali, L., Wu, H., & Noé, F. (2018).
VAMPnets for deep learning of molecular kinetics.
Nature communications, 9(1), 5.

Invited Talks

CECAM Workshop, Paris (2019)
"Learning the Collective Variables of Biomolecular Processes"

Lorentz Center Workshop, Leiden (2018)
"Machine Learning and Reverse Engineering for Soft Materials"

Posters

Neural Information Processing Systems, Montreal (2018)
"Deep Generative Markov State Models"
CECAM Workshop, Berlin (2018)
"Molecular Coarse Graining"
Machine Learning & Molecules Conference, Copenhagen (2017)
"VAMPnets for deep learning of molecular kinetics"

Acknowledgment

At this point, I would like to thank everyone who supported and motivated me during the PhD thesis, which includes all the people of the working group and of course my family, and in particular my wife.

In particular, I would like to wholeheartedly thank three people. First of all, I would like to thank Prof. Noé for supervising me, giving me impulses, and for his constructive criticism guiding me for the last common years.
Secondly, my thanks go to my dear friend Luca Pasquali, who made it always fun working on these presented projects.
Finally, I would like to mention Mauricio del Razo, who apart from being a wonderful friend was immensely supportive when writing this thesis with his suggestions and constructive criticism.

Andreas Mardt