DISCOVERY OF SLOW VARIABLES IN A CLASS OF MULTISCALE STOCHASTIC SYSTEMS VIA NEURAL NETWORKS

Przemysław Zieliński * **Jan S. Hesthaven**

Institute of Mathematics, EPFL, Switzerland

{przemyslaw.zielinski, jan.hesthaven}@epfl.ch

April 22, 2022

ABSTRACT

Finding a reduction of complex, high-dimensional dynamics to its essential, low-dimensional "heart" remains a challenging yet necessary prerequisite for designing efficient numerical approaches. Machine learning methods have the potential to provide a general framework to automatically discover such representations. In this paper, we consider multiscale stochastic systems with local slow-fast timescale separation and propose a new method to encode in an artificial neural network a map that extracts the slow representation from the system. The architecture of the network consists of an encoder-decoder pair that we train in a supervised manner to learn the appropriate low-dimensional embedding in the bottleneck layer. We test the method on a number of examples that illustrate the ability to discover a correct slow representation. Moreover, we provide an error measure to assess the quality of the embedding and demonstrate that pruning the network can pinpoint essential coordinates of the system to build the slow representation.

Keywords Multiscale dynamics · Slow-fast systems · Dimensionality reduction · Effective dynamics · Neural networks

Mathematics Subject Classification 37xx · 37Mxx · 65Pxx · 68T07

1 Introduction

Extracting effective dynamics from a high-dimensional systems remains one of the most challenging problems in computational modeling [14]. Nowadays, complex models with multiple time scales are found in a variety of domains, including biomechanics [2, 13], material research [27, 17], and climate science [23, 25]. The complexity of processes and scale separation in many such systems make the cost of direct simulation prohibitive. However, the long term evolution of certain aspects of these processes can often be described by much simpler *reduced* dynamics that capture the essential behavior. Thus, knowing the correct aspects allows one to build efficient model reduction techniques that not only decrease the dimension but also accelerate simulations [8, 19, 20, 22].

To take on this challenge, we focus on multiscale stochastic systems with local slow-fast timescale separation and introduce a method to learn a map that extracts the slow representation from the system, which we term

*Corresponding author

the *slow variable*. We build the method on the approximation capabilities of neural networks, owing to their ability to deal, in principle, with very high-dimensional problems and the availability of easy computations of derivatives, which we employ to assess the accuracy of learned maps. Other approaches to this problem were explored based on eigenfunctions of the Koopman operator [14, 15] or on manifold learning [29, 9]. Besides systems with explicit slow-fast time separation, systems that exhibit *metastable* behavior constitute another important class of multiscale models for which the problem of finding an appropriate reduced representation, in this context called *reaction coordinate*, was addressed [24, 4, 3]. Moreover, there exist data-driven techniques, like SINDY autoencoders [6, 5], designed to discover the underlying governing equations in the reduced space.

To set the stage, consider the class of slow-fast stochastic systems which are Markovian models of two coupled equations: a slow and a fast SDE, where the timescale separation is given explicitly by a small parameter ϵ, see equation (6). In such systems, the state variable x decomposes into two sets of coordinates (y, z), with y containing the slow dynamics on the timescale of the order $1/\epsilon$ and z the fast fluctuations on the timescale of the order of ϵ. On the one hand, with the slow variable y fixed, the fast process equilibrates rapidly to a conditional invariant distribution $\mathsf{P}_y(dz)$ supported on the $y = \mathrm{const}$ fiber. On the other hand, the evolution of the slow process can be effectively described on a certain submanifold of the state space $z = z(y)$ parametrized by the slow variable y. These two structures—the y-fibers and the $z(y)$ slow manifold—describe the slow-fast dynamics of the process and can be leveraged to design methods for effective simulation of slow-fast systems [18, 10].

In this study, however, we target a more complicated situation than the prototypical slow-fast systems exhibit: the directions of the slow and fast dynamics do not align with the coordinates in which the system is defined [29, 9, 15]. More precisely, we assume that the variables (y, z) are observed through an unknown nonlinear function f and the state variable reads $x = f(y, z)$. Therefore, the coordinates of the observed process in x have both slow and fast dynamics mixed. Locally around a given x there exist directions of both slow and fast variability, due to the noise in the system, and these directions depend on the current position in some nonlinear fashion, unlike in the slow-fast systems where we can globally split the coordinates into slow and fast. Moreover, since we assume the transformation f is unknown, the slow dynamics of the process in x remains hidden from the observer.

Our main objective is to encode in an artificial neural network a map, which we call the *slow map*, from the observed coordinates x to the slow variables y without explicit knowledge of the nonlinear observation function f. To achieve this, we develop a learning method that uses data from a number of simulations of the observed system. These data capture certain crucial yet computationally accessible features of the observed system. During training, the network explores the hidden noise correlations, contained in the dataset, to reveal the embedded lower-dimensional slow dynamics. This approach has potential to scale to high-dimensional systems and can naturally deal with vector valued slow maps.

To test the approach, we consider a class of stochastic differential equations which arise as a nonlinear transformation of a known slow-fast stochastic system. We term the underlying slow-fast SDE the *hidden system* and the SDE resulting from the nonlinear transformation the *observed system*. Since the observed system arises as a transformation of a slow-fast SDE, it shares certain properties with the hidden system. Crucial for our approach remains the fact that the state space of the observed system foliates into the fast fibers—a family of submanifolds on which the fast dynamics equilibrates rapidly—and contains an embedded slow manifold, transverse to the fibers, on which the long-term evolution happens. For the hidden slow-fast SDEs, the fast fibers always align with the z coordinates, thus the slow and fast variables are clearly distinguishable. However, the geometry of these fibers in the observed state space becomes more complicated, rendering the distinction between slow and fast variables unclear.

The architecture of our networks consists of two stacked, fully connected feed-forward subnetworks: encoder and decoder, connected through the bottleneck layer of width equal to the dimension of the slow variables. Therefore, it is the same as for autoencoders; the difference lies in the way we use this architecture. The network is trained on a dataset consisting of representative points from a sample trajectory of the observed system together with their projections along the fast fibers onto the slow manifold, serving as the targets. In other words, we aim to approximate the projection map and we train the network in the supervised manner.

Since the fully trained network represents the projection along the fast fibers, the encoder part learns a lower-dimensional representation of state space points that is constant along the fibers, i.e. a slow map. We do not explicitly impose this feature on the encoder during the training. Rather, since the decoder learns a parametrization of the slow manifold that is connected to the projection along the fibers, the only continuous representation the encoder can converge to glues the points on the fibers.

We also address the issue of testing the encoder by introducing an error measure based on the orthogonality between the derivative of the encoder and the directions of local fast variability. A lack of orthogonality between the vectors spanning these two subspaces of the state space indicates that the level sets of the encoder deviate from the fast fibers of the observed system. Measuring the amount of this mismatch, we can assess the accuracy of the trained models. Since the derivatives of the encoder with respect to the input can be easily computed via back propagation and the local fast directions via SVD, this yields a viable procedure to test the trained models.

Additionally, we demonstrate that applying a pruning technique during training, we can pinpoint the coordinates of the observed process that are essential for constructing the slow dynamics. It is often the case that in high-dimensional systems only a few of system's coordinates are involved in the hidden slow dynamics. The remaining coordinates add to the "nuisance" dimensions related mostly to the noise in the system. Sifting out these inessential variables can help build efficient methods to simulate the long time dynamics of complex systems.

The paper is organized as follows. We begin in Section 2 by examining the concept of a slow variable as a particular type of the observable of a stochastic process. Next, we introduce in Section 3 a class of multiscale stochastic systems that we use to test our approach, which we describe in Section 4. Sections 5 and 6 contain numerical examples which illustrate and test the method.

The code for all numerical experiments in this paper was written in Python. For training the neural networks we use PyTorch framework [26]. The code developed to train the networks and produce the figures can be found at github.com/przemyslaw-zielinski/sf_nets.

2 Observables and slow variables of stochastic processes

In this section, we discuss in more detail the concept of a slow variable of multiscale processes and offer a rudimentary definition that encompasses the cases we study in this manuscript. Other attempts to elaborate on the definition of slow variables often use the spectral properties of the Koopman or transfer operator, see [14, 15] or [4] in the case of reaction coordinates for metastable systems. Here, we stick to stochastic terminology and anchor our description on the geometric decomposition of the state space and the relaxation properties of the fast dynamics. This reflects the intuition behind our method and the properties of slow-fast systems that are a point of departure for our approach.

Let X_t be a stochastic process on $\mathbb{R}^D$. We call an *observable* any continuous function $\mathcal{F}\colon \mathbb{R}^D \to \mathbb{R}^K$, where $K \leq D$, such that the expectation

$$t \mapsto \mathsf{E}[\mathcal{F}(X_t)]$$

exists for all $t \in [0, T]$, with $T \in (0, +\infty]$ a fixed final time of the simulation. Intuitively, an observable is a *slow variable* if the image process on $\mathbb{R}^K$ through $\mathcal{F}$, given by

$$Y_t = \mathcal{F}(X_t),$$

evolves on a much slower timescale than X_t. To make this notion more precise, we introduce the following assumptions.

We suppose that $\mathcal{F}$ is a C^1 function with full rank Jacobian matrix $J_{\mathcal{F}}$, and for each $y \in \mathbb{R}^K$ we define $\mathbb{L}_y = \{x \in \mathbb{R}^D : \mathcal{F}(x) = y\}$ as the y-level set of $\mathcal{F}$. The family of all level sets $\{\mathbb{L}_y : y \in \mathbb{R}^K\}$ forms a foliation of the state space $\mathbb{R}^D$. Consider the process X_t conditioned to a fixed value y of $\mathcal{F}$

$$X_t \,|\, \mathbb{L}_y := \mathsf{E}[X_t \,|\, \mathcal{F}(X_t) = y], \quad t \in [0, T].$$

We assume that for each y there exists on $\mathbb{L}_y$ an invariant probability measure P_y of the process $X_t \,|\, \mathbb{L}_y$.

Remark 2.1 (On induced invariant measures). If the process X_t has itself an invariant measure $\mathsf{P} = \rho\,\mathrm{d}x$ on $\mathbb{R}^D$, the probability measures P_y do exist [22]. Indeed, for each $y \in \mathbb{R}^K$ we can consider the probability measure P_y such that

$$\mathrm{d}\mathsf{P}_y = \frac{1}{\Gamma(y)}\rho(x)\det\left(J_{\mathcal{F}}(x)^{\mathsf{T}}J_{\mathcal{F}}(x)\right)^{-1/2}\mathrm{d}\mathsf{S}_y, \tag{1}$$

where S_y is the surface measure on $\mathbb{L}_y$ (i.e., the Lebesgue measure on $\mathbb{L}_y$ induced by the Lebesgue measure in the ambient Euclidean space $\mathbb{R}^D$) and $\Gamma(y)$ is the normalization constant. The co-area formula shows that if a random variable X has law P, the law of X conditioned to a fixed value y of $\mathcal{F}$ equals P_y. □

For any $x \in \mathbb{R}^D$, let $X_t^x = \mathsf{E}[X_t \,|\, X_0 = x]$ and set $Y_t^x = \mathcal{F}(X_t^x)$. Intuitively, we call the process X_t multiscale and $\mathcal{F}$ its slow observable, if starting at any $x \in \mathbb{R}^D$ there is a time window on which the process $X_t^x \,|\, \mathbb{L}_y$ already equilibrates to P_y while the image process Y_t^x evolves only slightly from its initial position y. Hence, we postulate an existence of a timescale τ such that the following relations hold approximately

$$\begin{aligned} X_\tau^x \,|\, \mathbb{L}_y &\sim \mathsf{P}_y, \\ Y_\tau^x &\sim \delta_y, \end{aligned} \tag{2}$$

where δ_y is the Dirac mass at $y = \mathcal{F}(x)$. Relations (2) indicate that the slow observable $\mathcal{F}$ collapses the fast fluctuations present in the dynamics of X_t while retaining its slow evolution.

Example 2.2 (Slow and fast chemical reactions). A chemical reaction network with D species and K reactions is described by the state vector $x \in \mathbb{N}_0^D$, containing the number of particles of each species, and instantaneous changes of the state due to transitions of the form

$$x \to x + \nu_k,$$

where ν_k is the stoichiometric vector describing the net change in the number of molecules of each species due to the k-th reaction [12, 30]. The rates for the reaction to take place are quantified by constants $\gamma_k > 0$ and the temporal evolution of the relevant process X_t follows the dynamics in which the probability of the k-th reaction to occur is proportional to γ_k. These dynamics can be simulated directly using the *stochastic simulation algorithm* or by continuous approximation via the *chemical Fokker-Planck equation*.

The reaction network is multiscale if we can decompose all rates γ_k into D^f fast ones, indexed by $\mathbb{K}^f$, and D^s slow ones, indexed by $\mathbb{K}^s$, such that $\gamma_k \gg \gamma_{k'}$ for all $k \in \mathbb{K}^f$ and $k' \in \mathbb{K}^s$. An observable $\mathcal{F}\colon \mathbb{R}^D \to \mathbb{R}^{D^s}$ of the associated multiscale process X_t is slow if it does not change during the fast reactions [11], i.e., if for any x

$$\mathcal{F}(x + \nu_k) = \mathcal{F}(x), \quad \text{for all } k \in \mathbb{K}^f. \tag{3}$$

To obtain a general representation of such observables, it suffices to consider linear mappings satisfying (3). For a linear map $\mathcal{F}(x) = Bx$, where B is a $D^s \times D$ matrix, to fulfill (3), the rows b_i of B must satisfy $b_i \cdot \nu_k = 0$, for all $k \in \mathbb{K}^f$. We can always find such a set $b_1, \ldots, b_{D^s}$ of basis vectors in $\mathbb{R}^D$.

The level sets of linear slow variables are parallel hyperplanes of dimension D^f that do not typically align with the coordinates of the state space, due to the mixing of species during reactions. Thus, multiscale chemical reactions exemplify the generalization of the slow-fast systems we consider in this paper. However, since we focus on nonlinear mixing of slow and fast directions in the process X_t, the linear form of slow variables makes it too simple a case for our considerations. We refer to Section 3.3 for a more pertinent example. □

3 A class of multiscale systems

In this section, we introduce a particular class of models that we consider in this manuscript. The class comprises SDEs that arise from nonlinear transformations of slow-fast stochastic systems. This assumption mimics the situation when we observe the system in terms of the variable $x \in \mathbb{R}^D$ which is an unknown function of a D^s-dimensional slow variable y and a D^f-dimensional fast variable z. In this case, the coordinates of x contain both slow and fast dynamics, and we seek to find an observable $\mathcal{S}$, which we term *the slow map*, such that $\mathcal{S}(x) \approx y = \mathcal{F}(x)$, i.e., the values of $\mathcal{S}$ reproduce the slow variable y.

In Section 3.1, we define the observed and hidden systems that give the template for all test cases in our numerical examples and in Section 3.2 we delineate three features of the observed process that form the

basis for training and testing the networks. Section 3.3 presents a simple two-dimensional example of the observed-hidden system of interest. Finally, in Section 3.4, we return to the notion of the slow variable from Section 2 and use ideas introduced there to give a characterization of the slow map that enables us to assess the accuracy of the trained models.

3.1 Observed and hidden processes

Observed process We assume that the observed process X_t is a time-homogeneous diffusion in $\mathbb{R}^D$, driven by M-dimensional noise. More precisely, X_t satisfies the SDE

$$\mathrm{d}X_t = \mu(X_t)\,\mathrm{d}t + \nu(X_t)\,\mathrm{d}W_t, \tag{4}$$

where $\mu(X_t)$ is the $D \times 1$ drift vector, $\nu(X_t)$ is a $D \times M$ diffusion matrix, and W_t is an $M \times 1$ vector of independent Brownian motions.

The observed process X_t does not have any obvious slow-fast splitting: the slow and fast modes are mixed together. However, we suppose that there is an unknown, non-linear observation function $x = f(y, z)$, where y is a D^s-dimensional slow variable and z a D^f-dimensional fast variable, with $D^s + D^f = D$, and two corresponding processes Y_t and Z_t such that

$$X_t = f(Y_t, Z_t). \tag{5}$$

Hidden process Let us consider a small parameter $\epsilon \ll 1$ that represents the timescale separation. The hidden process (Y_t, Z_t) is governed by the following slow-fast system of SDEs

$$\begin{aligned} \mathrm{d}Y_t &= \mu^s(Y_t, Z_t)\,\mathrm{d}t + \nu^s(Y_t, Z_t)\,\mathrm{d}U_t, \\ \mathrm{d}Z_t &= \frac{\mu^f(Y_t, Z_t)}{\epsilon}\,\mathrm{d}t + \frac{\nu^f(Y_t, Z_t)}{\sqrt{\epsilon}}\,\mathrm{d}V_t. \end{aligned} \tag{6}$$

In (6), the statistics of the fast variable z quickly becomes tied to the value of the slow variable y. We formalize this property by assuming that for fixed y the process $Z_t \mid y$ that solves the equation

$$\mathrm{d}Z_t = \frac{\mu^f(y, Z_t)}{\epsilon}\,\mathrm{d}t + \frac{\nu^f(y, Z_t)}{\sqrt{\epsilon}}\,\mathrm{d}V_t$$

has an invariant measure P_y. The paths of (Y_t, Z_t) fluctuate rapidly along hyper-surfaces $y = \mathrm{const}$ and the process $Z_t \mid y$ equilibrates quickly to P_y. Under certain stability assumptions and in the small noise case, it can be demonstrated that the trajectories of (6) are concentrated in a "layer" surrounding a manifold of the form

$$z(y) = \{\mu^f(y, z) = 0\} + \mathcal{O}(\epsilon)$$

and there is a slow spread in the y-direction [1]. We assume that such stability holds in our case, i.e. we do not consider systems with noise induced transitions, and that the value $z(y)$ is given by the mean of the quasi-invariant distribution P_y. The ratio between the speed of fluctuations in the z-direction and the evolution along the manifold $z(y)$ is given by $1/\epsilon$.

3.2 Multiscale features of the observed system

The multiscale nature of the observed process X_t comes from it being a nonlinear transformation of a hidden slow-fast system (6). Though, in practice, we do not have access to the underlying slow-fast process, there are three main features that arise in the dynamics of X_t due to the presence of the hidden system. These features are "computationally accessible", as long as we know the observed SDE (4) explicitly or are able to freely simulate the trajectories of X_t.

1. **Local slow and fast directions**
   These directions are encoded in the spectral decomposition of the (variance-)covariance matrix of the observed process
   $$\sigma(x) = \nu(x)\nu(x)^{\mathsf{T}}. \tag{7}$$

For two-time-scale slow-fast hidden systems the spectrum of $\sigma(x)$ is divided for each x into two clusters of eigenvalues separated by a large gap. The eigenvectors corresponding to the largest eigenvalues identify directions of large noise variability whereas the ones that correspond to small eigenvalues span the local slow subspace.

2. **Fast fibers**

   The fast fibers form a family $\{\mathbb{F}_y : y \in \mathbb{R}^{D^s}\}$ of D^f-dimensional submanifolds of $\mathbb{R}^D$ along which the fast dynamics occurs. The fast fibers foliate the D-dimensional state space of the observed system and are tangential to the local fast directions. This family arises from the corresponding D^f-dimensional fast hyper-planes of the hidden slow-fast process (6), which are aligned with the direction of the fast variable z. Moreover, by pushing forward the quasi-invariant measures from the fast hyper-planes of (6), we obtain the corresponding family of probability measures P_y on the fibers $\mathbb{F}_y$.

3. **Slow manifold**

   The slow manifold $\mathbb{S}$ is a D^s-dimensional submanifold of $\mathbb{R}^D$ transverse to the fast fibers and such that the slow evolution of process X_t progresses along $\mathbb{S}$. It corresponds to the adiabatic manifold $z(y)$ of the hidden system discussed in Section 3.1. The process X_t does not necessarily evolve on or close to the manifold $\mathbb{S}$, due to random fluctuations in the transverse fast directions, but for any $x \in \mathbb{S}$, the statistics of $X_t^x = \mathsf{E}[X_t \,|\, X_0 = x]$ for a short time windows are fully determined by x.

In the subsequent sections, we discuss some numerical approaches for approximating these features. In particular, owing to the stability assumptions on the hidden slow-fast system from Section 3.1, the points on the slow manifold $\mathbb{S}$ can be approximated by averaging a sample from the quasi-invariant measures P_y. This property provides the basis for computation of the values of the projection onto $\mathbb{S}$ presented in Section 4.2.

3.3 Example: two-dimensional SDE with periodic slow variable

As the first illustrative example, let us examine a set of two coupled equations from [15]. The hidden slow-fast system, dating back to [7], reads

$$\begin{aligned} \mathrm{d}Y_t &= \sin(Z_t)\,\mathrm{d}t + \sqrt{1 + \frac{1}{2}\sin(Z_t)}\,\mathrm{d}U_t, \\ \mathrm{d}Z_t &= \frac{\sin(Y_t) - Z_t}{\epsilon}\,\mathrm{d}t + \frac{1}{\sqrt{\epsilon}}\,\mathrm{d}V_t, \end{aligned} \tag{8}$$

and is defined for $(y, z) \in [0, 2\pi] \times \mathbb{R}$ with periodic boundary conditions on y. When we fix y, the fast variable z follows the Ornstein-Uhlenbeck process with mean $\sin(y)$ and variance $1/2$. The slow variable y itself moves close to the manifold given by the equation $z = \sin(y)$. For all simulations, we set $\epsilon = 10^{-3}$.

In Figure 1, we display a sample path of the slow-fast system (8) with the individual coordinates in the left panel and the system's trajectory in the state space, superimposed by fast fibers and slow manifold, on the right. To generate the path, we use Euler-Maryuama method with time step $\Delta t = \epsilon/8$ and initial condition at $(y_0, z_0) = (2.0, 1.0)$. The slow manifold is given by the relation $z = \sin(y)$ that we obtain by rewriting the fast equation in (8) as $\epsilon\,\mathrm{d}Z_t = [\sin(Y_t) - Z_t]\,\mathrm{d}t + \sqrt{\epsilon}\,\mathrm{d}V_t$ and setting $\epsilon = 0$. The fast fibers align with the z axis due to the separation of slow and fast modes in (8). This separation is also visible in the difference between the variability of the coordinates: the fast variable z fluctuates much more rapidly then the slow y (see the left panel). The correct slow map for (Y_t, Z_t) in (8) is simply the projection onto y.

To obtain the observed system, we consider the hidden system after a change of coordinates

$$f\colon (y, z) \mapsto (x^1, x^2) = (y + \sin(z), z). \tag{9}$$

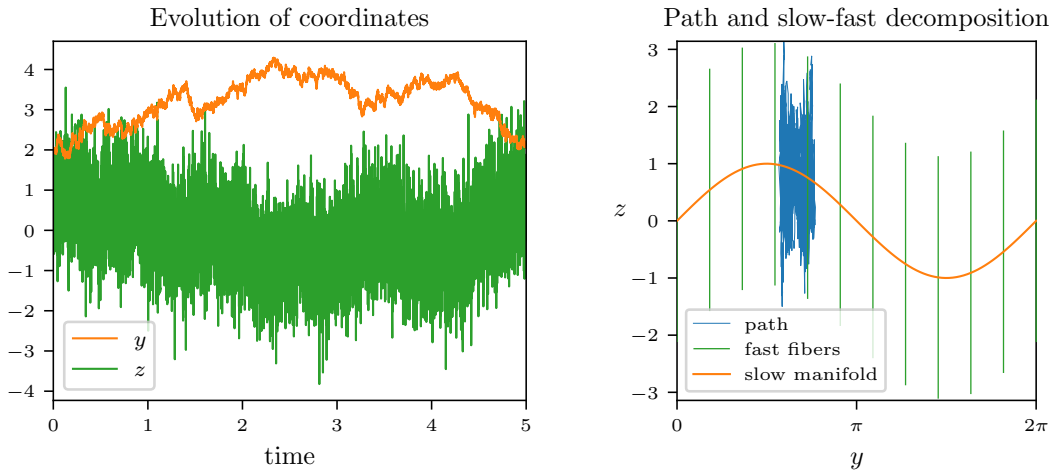

Figure 1: Visualization of a single path of (8). (Left) The evolution of individual coordinates of the path. The difference between variability of z and y on short time intervals illustrates the timescale separation between the two variables. (Right) The initial portion (first 2,000 time points) of the simulated path as seen in the (y, z) state space (blue) with the slow manifold of the system (orange curve) and a few sections of fast fibers (green vertical segments) superimposed. The path fluctuates rapidly along the fast fibers while slowly drifting to the right such that its mean stays on the slow manifold.

The mapping f is indeed a diffeomorphism with an inverse that takes (x^1, x^2) to $(x^1 - \sin(x^2), x^2)$. After applying (9) to (8), the resulting observed process $X_t = (X_t^1, X_t^2)$ follows the SDE

$$\begin{aligned} \mathrm{d}X_t^1 &= \left[\sin(X_t^2) + \frac{\cos(X_t^2)}{\epsilon}\big(\sin(X_t^1 - \sin(X_t^2)\big) - \frac{\sin(X_t^2)}{2\epsilon}\right]\mathrm{d}t \\ &\qquad + \sqrt{1 + \frac{\sin(X_t^2)}{2}}\,\mathrm{d}U_t + \frac{1}{\sqrt{\epsilon}}\cos(X_t^2)\,\mathrm{d}V_t, \\ \mathrm{d}X_t^2 &= \frac{\sin\big(X_t^1 - \sin(X_t^2)\big) - X_t^2}{\epsilon}\,\mathrm{d}t + \frac{1}{\sqrt{\epsilon}}\,\mathrm{d}V_t. \end{aligned} \tag{10}$$

Note that now the terms involving ϵ appear in the formulas for both coordinates. In the left panel of Figure 2, we demonstrate that indeed both coordinates of process (10) fluctuate on the same timescale; there is no separation present in the coordinates of the observed system. Having the change of coordinates (9) explicitly available, we can recover the observed slow manifold and the fast fibers as images through this transformation of respective submanifolds of the state space of the hidden system. In the right panel of Figure 2, we observe essentially the same behavior as in the case of slow-fast SDE (8), albeit more complicated: the process fluctuates rapidly along fast fibers while its mean drifts on the slow manifold. That the fast fibers are no longer aligned with any axis reflects the mixing of slow and fast modes in (10).

Finally, let us look into the information contained in the covariance matrices $\sigma(x)$ of the observed process (10). For this we select 100 points x_n from the simulated path of the observed process and perform an eigen-decomposition of the associated covariance matrices $\sigma(x_n)$. We present the results in Figure 3.

We can detect the timescale separation by looking at the eigenvalues of local noise covariance matrices at each data point. In the left panel of Figure 3, we plot, for each data point x_n, the two eigenvalues of the covariance matrix $\sigma(x_n)$. The spectral gap of size roughly 10^3, equal to the original timescale separation $1/\epsilon$, is clearly visible. In the right panel, we visualize the slow and fast local directions at x_n given by the normalized eigenvectors of $\sigma(x_n)$, which are orthogonal since σ is symmetric. Observe that the fast directions, instead of being aligned with one axis, now follow a spatial sinusoidal pattern resulting from the non-linear transformation (9), and they are tangential to the fast fibers of the observed system. Therefore, even when

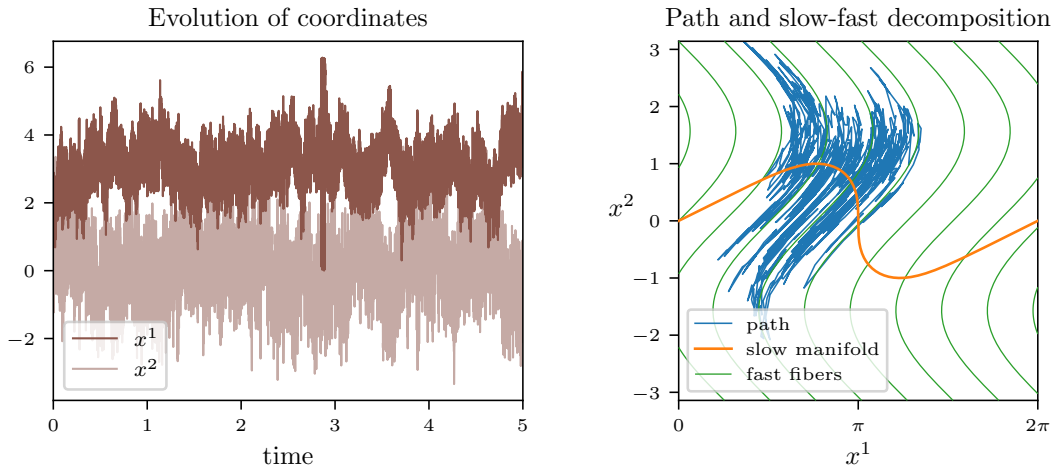

Figure 2: Visualization of a single path of (10). (Left) The evolution of the individual coordinates from the path (dark and light brown). In contrast to Figure 1, there is no clear difference between the variability of the coordinates on any timescale. (Right) The initial portion (first 1,500 time points) of the simulated path (blue) as seen in the (x^1, x^2) state space with the slow manifold of the system (orange curve) and a few sections of fast fibers (green sinusoidal curves) superimposed. We observe the same behavior as in Figure 1, although the geometry of the fast fibers is more complex due to non-linear mixing of slow and fast variables.

we do not know the underlying transformation from the hidden to the observed system, we can recover the underlying structure of fibers from the eigenvalues of local noise covariance matrices. We shall use this insight to assess the trained models in Section 6.2.1.

3.4 Characterizing the slow map

We combine certain features of the observed process with our tentative definition of the slow observable to obtain a characterization of the slow map. This characterization does not require the knowledge of the hidden system nor the underlying transformation from the hidden to the observed process, thus making it a viable approach on which we will build a general computational procedure.

As discussed in Section 3.2, for any observed system (4), we have a foliation $\{\mathbb{F}_y : y \in \mathbb{R}^{D^s}\}$ of $\mathbb{R}^D$ into D^f-dimensional fast fibers $\mathbb{F}_y$. Intuitively, we expect that a slow map, by exploring the correlations of fast local processes, will collapse the fast fibers of the observed process. Following this intuition, we posit that a slow map is any mapping $\mathcal{S}\colon \mathbb{R}^D \to \mathbb{R}^{D^s}$ that satisfies

$$\text{for every } y \in \operatorname{im}\mathcal{S} \text{ there exists a unique fast fiber } \mathbb{F}_{y'} \text{ such that } \mathbb{L}_y \approx \mathbb{F}_{y'},$$

where $\mathbb{L}_y = \{x \in \mathbb{R}^D : \mathcal{S}(x) = y\}$ is the y-level set of $\mathcal{S}$. Therefore, we expect the level sets of a slow map to approximate the fast fibers of the observed system. If this is the case, the local fast directions of the observed process become tangential to the level sets of $\mathcal{S}$, thus they get compressed the most under the action of $\mathcal{S}$. This indicates that $\mathcal{S}$ will satisfy condition (2) whenever X_t is the observed process (4).

Remark 3.1 (Non-uniqueness). Note that the condition given above does not guarantee uniqueness of $\mathcal{S}$. Indeed, whenever $\mathcal{S}$ is a slow map and $\phi\colon \mathbb{R}^{D^s} \to \mathbb{R}^{D^s}$ is one-to-one, the function $\phi \circ \mathcal{S}$ will have level sets that also align with the fast fibers of the observed system. □

The overarching goal of our method is to encode in the artificial neural network a map $\mathcal{S}$ such that the two foliations of the observed state space $\mathbb{R}^D$—the foliation into the fast fibers and the foliation into the level sets of $\mathcal{S}$—align. Because the solution to this problem cannot be unique, the exact values of $\mathcal{S}$ do not matter. Therefore, to assess the accuracy of the models, we need to look at the quality of approximation of the fast fibers by $\mathcal{S}$, see Section 6.2.

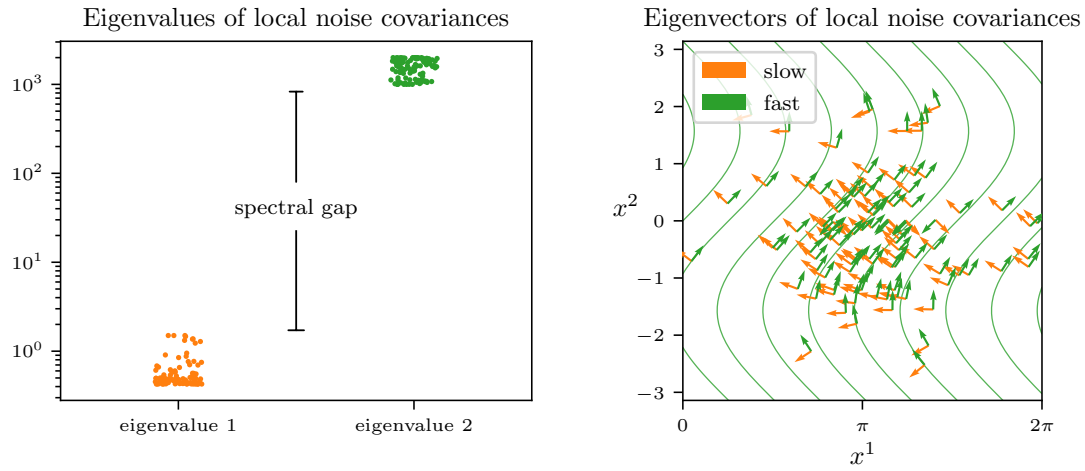

Figure 3: Eigen-decompositions of local noise covariances $\sigma(x)$ of the observed process (10) evaluated on 100 data points from a sample path. (Left) The categorical scatter plot of the eigenvalues of $\sigma(x)$ based on their magnitude. The eigenvalues cluster into two groups and the width of the spectral gap corresponds roughly to the timescale separation equal $1/\epsilon = 10^3$. (Right) The eigenvectors of $\sigma(x)$ colored correspondingly to their associated eigenvalues. The eigenvectors corresponding to the large (fast) eigenvalues are tangential to the fast fibers (green curves).

Remark 3.2 (Stochastic and time-dependent slow maps). In this manuscript, we work with the class of systems that arise as a deterministic transform of slow-fast SDE. This implies that, to extract the slow variables, it suffices to search for deterministic slow maps. In general, the relation between the observed and the underlying slow-fast system may be more complicated and require a time-dependent and stochastic transform to capture the slow variables, see [28]. Memory terms and randomness can be included in neural networks using special architectures, and we leave it for the future work to explore such approaches. □

4 A method for learning a slow map

In this section, we present an approach to approximate the slow map of an observed system (4) by an artificial neural network. The method relies on the insight we discussed in the previous section, i.e., to train the network so that the level sets of the encoder mapping overlap with the fast fibers of the observed process.

One can envisage two ways of computationally accessing the fast fibers of the observed process: directly, by sampling them using the ability to simulate the observed process, or indirectly, by looking at the eigenvectors associated with the fast eigenvalues of the covariance matrices (these eigenvectors, as we mentioned, span the tangent spaces of the fibers). Sampling the fast fibers is a challenging task; these are subsets of the state space with measure zero hence the drift and dispersion in the stochastic process force it to quickly escape the fiber containing the starting point. Computing the eigenvectors of the fast eigenvalues is a viable procedure, but it is not clear how obtaining the fibers as integral curves would generalize for higher-dimensional system.

We pursue a different approach that relies on approximating the projection map $\mathcal{P}$ onto the slow manifold along the fast fibers of the observed process. We use averaging of short bursts of the simulation to produce a reliable approximation of the value of this projection at a given point in state space. By imposing a bottleneck in the network, with size equal to the number of slow variables, we force the network to learn the slow map in this layer. Though the approximation of the slow map clearly falls within the supervised learning techniques, we do not really supervise the representation in the bottleneck. Nevertheless, with a number of test cases presented in Sections 5 and 6, we show that we can obtain correct slow maps in the bottleneck layer.

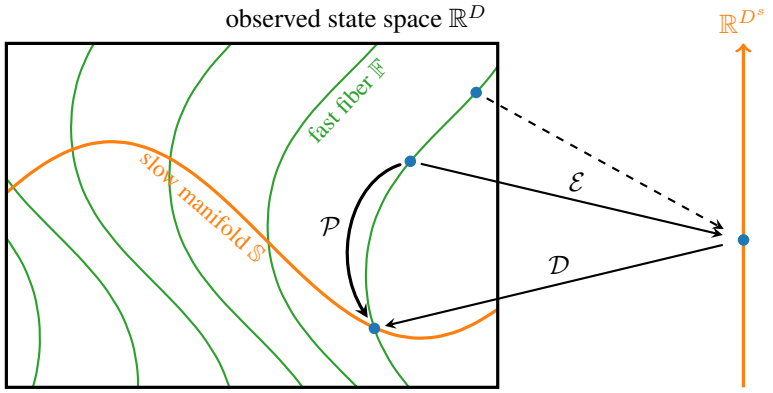

Figure 4: Comparison of the forward action of the projection $\mathcal{P}$ in the observed system with the forward action of the encoder-decoder network $\mathcal{N} = \mathcal{D} \circ \mathcal{E}$. The projection $\mathcal{P}$ moves points along the fast fibers $\mathbb{F}$ of the observed system onto the slow manifold $\mathbb{S}$. The network $\mathcal{N}$ learns to approximate $\mathcal{P}$ while passing by the intermediate D^s-dimensional representation. Since $\mathcal{P}$ glues all points on a fast fiber $\mathbb{F}$, the encoder $\mathcal{E}$ is forced to associate the same representation to these points. That results in the overlap of the encoder level sets with the fast fibers, a salient property of a slow map enunciated in Section 3.4.

4.1 Network architecture

We employ a fully-connected, feed-forward network $\mathcal{N}$ with input and output dimensions both equal to D: the dimension of the state space of process X_t. Additionally, the network contains a bottleneck layer of dimension D^s, equal to the number of slow variables in system (4). In the multiscale systems of interest, D^s will be significantly smaller than D, and it can be estimated by analyzing the spectrum of the covariance matrices $\sigma(x)$.

The architecture of the network is, therefore, the same as for an autoencoder. The main difference between our approach and the standard situation, is that we do not train the network to approximate the identity mapping on $\mathbb{R}^D$. Rather we train it in a supervised manner to approximate the slow projection $\mathcal{P}$. However, owing to this similarity, we will use the same terminology as employed when describing autoencoders: we divide the network into an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$ that are sub-networks before and after the bottleneck, respectively, so that $\mathcal{N} = \mathcal{D} \circ \mathcal{E}$. Owing to the interpretation of the bottleneck, we call the bottleneck layer the slow view. Though we do not employ autoencoders, we provide in Section 5.3 an illustration of applying unsupervised approach to our problem to show what can go wrong when using an autoencoder directly.

The motivation to use encoder-decoder architecture stems from the nature of the approximated map: the projection $\mathcal{P}$. The properties of the slow manifold imply that $\mathcal{P}$ is an endomorphism of $\mathbb{R}^D$ whose image is a D^s-dimensional submanifold of $\mathbb{R}^D$. Consequently, the image of $\mathcal{P}$ can be parametrized, at least locally, by D^s coordinates. After the training, this parametrization is performed by the decoder $\mathcal{D}$ while the encoder $\mathcal{E}$ learns certain D^s-dimensional representations of the points of the state space, see Figure 4.

Since $\mathcal{P}$ collapses the fast fibers, the encoder $\mathcal{E}$ has to assign all points on the fast fiber to the same representation; with such representation the decoder $\mathcal{D}$ becomes unaware of the fast fibers. To demonstrate this fact, let us first note that $\mathcal{E}$ cannot assign the same representation to points on different fast fibers. Otherwise, we could take two points $x, x' \in \mathbb{R}^D$ which have the same representation $\mathcal{E}(x) = \mathcal{E}(x')$ yet different projections $\mathcal{P}(x) \neq \mathcal{P}(x')$. Then, if the network were to approximate $\mathcal{P}$, we would have $\mathcal{N}(x) \neq \mathcal{N}(x')$ which contradicts the fact that $\mathcal{N}(x) = \mathcal{D}(\mathcal{E}(x)) = \mathcal{D}(\mathcal{E}(x')) = \mathcal{N}(x')$. In consequence, the level sets of $\mathcal{E}$ are D^f-dimensional closed submanifolds of the fast fibers $\mathbb{F}$ and, since each $\mathbb{F}$ is connected, the level sets of the encoder must overlap with the fast fibers.

Therefore, the only way $\mathcal{N}$ can learn the approximation of $\mathcal{P}$ is when the encoder $\mathcal{E}$ learns the parametrization of the fast fibers. If this is indeed the case after training the network, for any latent representation $y \in \mathbb{R}^{D^s}$, the

y-level set of the encoder $\{x \in \mathbb{R}^d : \mathcal{E}(x) = y\}$ will overlap with a fast fiber of the observed system. Hence, $\mathcal{E}$ will satisfy the main requirement for being a slow map as discussed in Section 3.4.

4.2 Generating datasets

For training and testing the networks we use three ingredients: 1) a sample of data points from the state space that captures the long-time dynamics of the observed process, 2) the corresponding values of the projection onto the slow manifold, and finally 3) the covariance matrices associated with the data points. Therefore, all datasets consist of M instances in the form of triplets

$$\big(x_m, \mathcal{P}(x_m), \sigma(x_m)\big), \quad m = 1, \dots, M,$$

where x_m are the data points in $\mathbb{R}^D$, $\mathcal{P}(x_m)$ the corresponding projections of data points onto the slow manifold of observed system (4), and $\sigma(x_m)$ the associated local noise covariances. Below, we delineate the exact procedure used to generate our datasets.

Data points The data points come from a single trajectory of the observed process X_t sampled at a uniform time interval Δt over N time steps. To simulate this trajectory, we employ basic Euler-Maruyama method

$$X_{n+1} = X_n + \mu(t_n, X_n)\,\Delta t + \nu(t_n, X_n)\,\Delta W_{n+1}, \quad X_0 = x, \tag{11}$$

where $\Delta W_{n+1} = W_{t_{n+1}} - W_{t_n}$ are Brownian increments and $x \in \mathbb{R}^D$ is a fixed starting point. For stability reasons, the time step Δt needs to be a fraction of the fast timescale ϵ, hence $\Delta t < \epsilon$. However, to ensure the slow variables had enough time to appreciably evolve, we need to have observations spanning time intervals much larger than ϵ, thus $N\Delta t \gg \epsilon$. These two requirements result in a large N, i.e., in a large number of points in the sample trajectory. It is generally unnecessary to use all these points to train the networks since we need only a good sample on the timescale of the slow variables. Therefore, to construct the dataset, we subsample the trajectory by uniformly selecting M points from it.

Values of the projection We obtain the projection $\mathcal{P}(x_m)$ of the data point x_m onto the slow manifold of the observed system by running many parallel short trajectories all starting at x_m and computing the average of their final position. This procedure gives the approximation of the correct values of the projection owing to the connection between the slow manifold and the quasi-invariant measures on the fast fibers $\mathbb{F}_y$, discussed in Section 3.2. To attain good approximation, we need to fix an intermediate timescale τ: sufficiently large to allow the fast variables to sample the invariant distribution on the fast fiber, but short enough to prevent the variables from evolving on the slow manifold (see also Section 2). For observed system (4), the relevant time scales on which fast processes evolve are approximated by $1/|\lambda_k|$, where λ_k are the largest eigenvalues of the covariance matrices σ. In general, τ should be fixed as a small multiple of the average of those inverses and the time step δt, used to sample the short trajectories, should be set accordingly (we usually set δt as a fraction of Δt). With τ chosen correctly, the end points of the short trajectories sample the invariant distribution of fast variables with slow part of x_m frozen and, due to the stability of the slow manifold, their average approximates the projection $\mathcal{P}(x_m)$.

Local noise covariances Finally, to compute $\sigma(x_m)$, we evaluate the dispersion coefficient $\nu(x_m)$ at all data points x_m and obtain the associated covariance via (7). In case (4) is inaccessible directly—because we use another simulation method, e.g., *Gillespie's stochastic simulation algorithm*—we can estimate $\sigma(x)$, for a fixed $x \in \mathbb{R}^D$, by J parallel one-step trajectories all starting at x. The resulting point cloud $\{x_j\}_{1<j<J}$ samples approximately a Gaussian distribution with covariance matrix $\sigma(x)$ whose (d, d')-entry we estimate by

$$\sigma(x)^{d,d'} \approx \frac{1}{\Delta t J} \sum_{j=1}^{J} x_j^d x_j^{d'} - \bar{x}^d \bar{x}^{d'}, \tag{12}$$

where $\bar{x}$ denotes the average value of the point cloud. To obtain good estimation for σ, the size of the time step Δt should be small compared to the parameters of equation (11).

The precise choice of all additional parameters depends on a specific system under study and is a part of the data preprocessing. Therefore, we do not elaborate more on the procedures required to make the right choice and assume we have data of good quality.

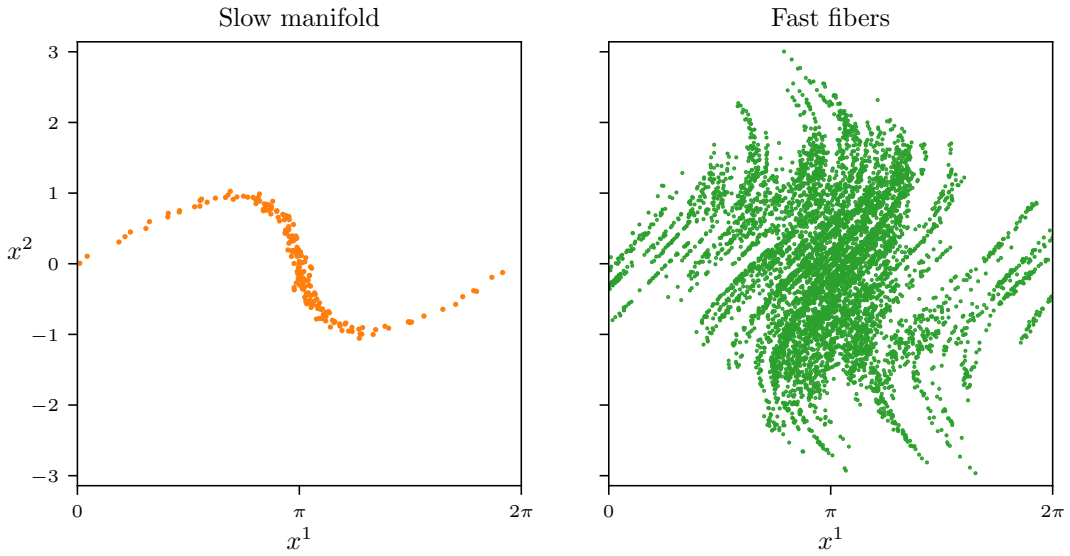

Figure 5: Computing the slow-fast foliation induced by observed process (10). The slow manifold (left) can be approximated by averaging intermediate bursts of path simulations. See discussion in Section 4.2 for more details. The fast fibers (right) emerge when we run a very short path simulations starting from a set of data points.

We present the spectral decomposition of the local noise covariances for system (10) in Figure 3. To give an illustration of the two other sets, we look at the results of sampling the slow manifold and the fast fibers for this system. We plot in Figure 5 the approximation of the slow manifold and the fast fibers of (10). Unlike the corresponding features of the hidden process, these are (computationally) accessible, see Section 3.2, and can be used as additional data attached to the data points sampling the state space.

4.3 Training

We train the networks on randomized mini-batches and divide the dataset into training and validation sets. In this phase, as the data instances, we only use the tuples $\big(x_m, \mathcal{P}(x_m)\big)$, where each x_m serves as the input and $\mathcal{P}(x_m)$ as the corresponding label for supervised learning. We always use the mean square error loss function.

For each training session, we fix a number of maximum epochs and choose the model with the smallest validation loss. We do not employ any stopping criterion, though it is readily available with train-validation splitting. We only monitor the losses during the training to avoid over-fitting and to ensure there is enough epochs for the loss curves to flatten.

4.4 Comparison to some similar methods

Having presented a method for extraction of the slow map, we briefly and non-exhaustively summarize some of the previous work that inspired and stimulated our research. We do not provide an extensive comparison in this expository manuscript. We only aim at indicating some unresolved issues that we try to address with the encoder-decoder architecture.

The authors in [14, 15] achieve a number of objectives. Their method tests for multiscale nature of the investigated system, estimates the timescale separation and distills the reduced slow dynamics. The approach is based on transfer and Koopman operators, and it uses a discretization of Fokker-Planck operator for finding its eigenvalues and eigenfunctions. This discretization requires gridding the domain of the operator and potentially limits the application of the method to low-dimensional systems, like the examples provided. In our approach, by building on the approximation capabilities of neural networks, we attempt to omit such

bottlenecks from the algorithms. On the other hand, the method developed in these papers to compute the reduced dynamics seems applicable to the slow map encoded in the network, thus providing a way to learn additionally the slow equations.

The papers [29, 9] attack the issue of the discovery of the slow dynamics with manifold learning. The authors employ diffusion maps in order to embed the data from the simulation of multiscale system into low dimensional space. However, it remains to be specified how one could extract from this embedding the nature of the slow map as a function of the observed variables. To address this issue, we encode the slow map into the encoder part of our network. The encoder does not reveal the explicit formula for the slow map, yet it provides the access to this mapping giving us more insight into it. For example, in Section 6, we point out how one can learn which variables are essentially involved in building the slow map.

In [5, 6] the authors combine autoencoders with the sparse identification of nonlinear dynamics (SINDy) technique to discover the underlying governing equations in the reduced space of both uniscale and multiscale dynamical systems. In the multiscale case, they focus on nonlinear slow-fast systems with linear coupling. Our choice of encoder-decoder architecture (the same as for autoencoders) makes our approach fell into a similar category, but we target a wider class of observable systems and focus more on investigating the properties of stochastic dynamics. We do not obtain the equations in the latent view, rather trying to encode the slow map. However, incorporating identification technique like SINDy into our approach, both for finding governing equations and the analytical form of the slow map, is feasible and can be pursued in the future work.

5 Visualization with two-dimensional test systems

We begin by considering two systems in $\mathbb{R}^2$, each with one slow and one fast variable. This setting allows us to visually compare the slow map with the trained encoder by plotting the values of the slow map against the values of the encoder over a test dataset. Moreover, we can visualize and compare the level sets of both maps to inspect the overlap.

At the end of this section, we present results of training in an unsupervised manner a usual autoencoder to reconstruct the slow manifold. We point out that by doing that, the fast fibers of the observed system are generally not captured correctly.

5.1 A system with periodic slow variable

For the first test, we return to (10). The slow map, resulting from inverting (9), reads

$$\mathcal{S}:\ x=(x^1,x^2)\mapsto x^1-\sin(x^2). \tag{13}$$

We trained several models with varying architectures from which we present the smallest one with the best accuracy. We use a training dataset of 1876 instances and validate with 804 instances. The chosen network has shape 2 – 4 – 1 – 4 – 2 with *exponential linear unit* (ELU) activations

$$\texttt{ELU}(u)=\begin{cases} x & \text{if } x>0,\\ \alpha\cdot\big(\exp(x)-1\big) & \text{if } x\le 0,\end{cases}$$

with $\alpha=1$, and was trained over 3000 epochs with mini batches of size 16 and using the `Adam` optimizer [21] with learning rate equal to 0.003. We plot some crucial features of the trained model and compare them to the known features of (10) in Figure 6. To make a comparison, we compute the values of the network $\mathcal{N}$ and the encoder $\mathcal{E}$ on the test set of 1321 instances unseen by the network during the training.

In the left panel of Figure 6, we plot in the (x^1,x^2)-state space the values of the network $\mathcal{N}$ obtained on the test data (blue). The network is trained to approximate the projection $\mathcal{P}$ onto the slow manifold $\mathbb{S}$. Therefore, the network should put any point from the state space onto this manifold, which we refer to as *reconstructing* $\mathbb{S}$, and this is confirmed by this plot.

The reconstruction of the slow manifold is only the means to approximating the slow map $\mathcal{S}$ by the encoder $\mathcal{E}$. In the one-dimensional case, we can directly visualize the accuracy of this approximation by plotting the values of $\mathcal{E}$ against the corresponding values of the slow map (13). We display the resulting plot in the right panel. We can see that there is a relation $\mathcal{E}=\phi\circ\mathcal{S}$ between the two, where ϕ is an affine function.

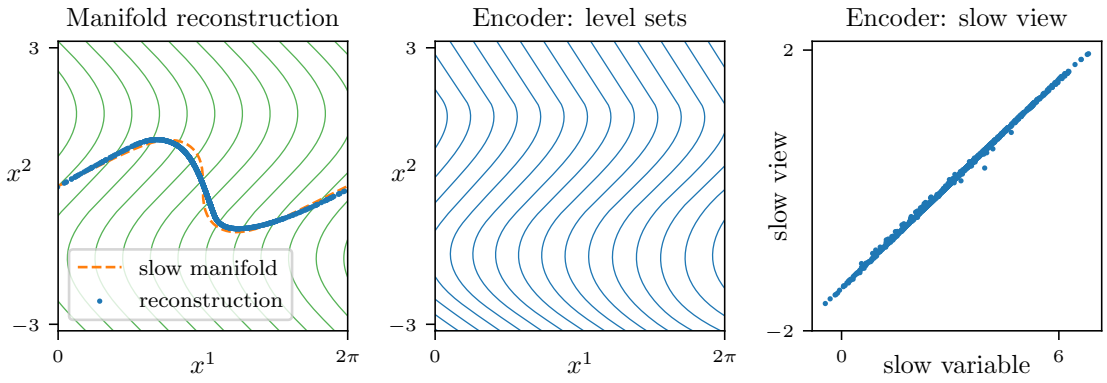

Figure 6: Visualization of the results of the trained model for system (10). (Left) Reconstruction of the values of the projection $\mathcal{P}$ by the network $\mathcal{N}$ (blue) shows a good accuracy with the underlying slow manifold. (Center) The level sets of the encoder $\mathcal{E}$. These also show good agreement with the fast fibers of the system displayed in green on the left. (Right) The plot of the values of the encoder $\mathcal{E}$ against the corresponding values of slow variable (13). The two are clearly related by an affine transformation.

To gain more insight into the performance of the model, we also display in the central panel of Figure 6 the level sets of the encoder $\mathcal{E}$. As discussed in Section 3.4, the more the level sets of $\mathcal{E}$ overlap with the fast fibers of the observed system, equal to the level sets of (13) and plotted in green on the left, the better the approximation. In this case, the model learned the correct sinusoidal shape across the whole range of data.

5.2 Oscillating half-moons

To test the method on a bit more challenging yet still two-dimensional system, we consider the following dynamics from [29], dubbed the "oscillating half-moons". The hidden system of SDEs reads

$$\begin{aligned} \mathrm{d}Y_t &= a_1\,\mathrm{d}t + a_2\,\mathrm{d}U_t \\ \mathrm{d}Z_t &= a_3(1 - Z_t)\,\mathrm{d}t + a_4\,\mathrm{d}V_t, \end{aligned} \tag{14}$$

where a_i are constants which we set as $a_1 = a_2 = 10^{-3}$, $a_3 = a_4 = 2.5 \cdot 10^{-2}$. The processes Y_t and Z_t are decoupled, with Y_t slowly growing on average with rate a_1 and Z_t fluctuating rapidly around 1. The timescale separation, approximately given by $a_4/a_2 = 25$, is moderate but sufficient to observe the required slow-fast behavior (see Figure 7). Moreover, the fast process in (14) with $a_3 = a_4$ has a different scaling than model problem (6), but is within the framework of Section 2.

The nonlinear transformation reads

$$\begin{aligned} x^1 &= z\cos(y + z - 1), \\ x^2 &= z\sin(y + z - 1). \end{aligned} \tag{15}$$

Since Y_t is unstable, this transformation makes the paths of the observed process circle around the center of the observed state space. The fast fluctuations of Z_t are transformed into rapid oscillations along the spirals emanating from the center of the plane. We show an illustrative trajectory in Figure 7.

We do not present the full equations of the observed system – they can be derived (as for all other systems we analyze) from formulas (14) and (15) via Itô's lemma. Let us only display the slow map for this system:

$$y = \arctan(x^2/x^1) + 1 - \sqrt{(x^1)^2 + (x^2)^2}, \tag{16}$$

which we use henceforth to assess the accuracy of the trained models.

We train the networks on the dataset presented in the right panel of Figure 7, comprising 1600 data points. For the first experiment, we sample only a part of the natural domain of the data points (highlighted points in the right panel, comprising 1195 data points) in order to avoid the discontinuity in the slow map (16). Due to

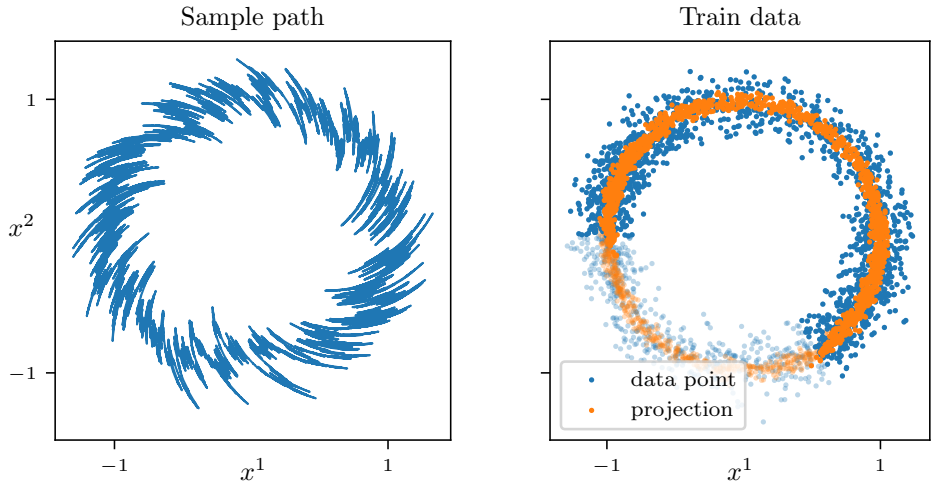

Figure 7: Sample path (left) and training data (right) for the oscillating half-moons system from Section 5.2. The highlighted data is used first to train the plain encoder-decoder network, for reasons described in the text, whilst we use the full dataset with the modified network from Section 5.2.1.

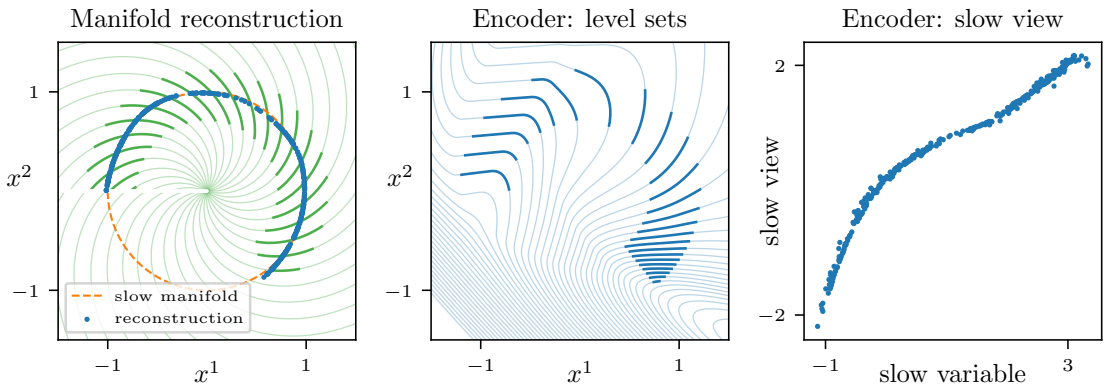

Figure 8: Visualization of the results of trained model for system from Section 5.2. (Left) Reconstruction of the slow manifold. (Center) The level sets of the encoder $\mathcal{E}$. (Right) Values of the encoder $\mathcal{E}$ plotted against the corresponding values of the slow map (16).

this discontinuity, (16) cannot be accurately represented on the whole $\mathbb{R}^2$ by the continuous encoder map. In Section 5.2.1, we modify the network so that it well approximates the slow map for the complete dataset.

First, we train the encoder-decoder of shape $2-4-2-1-2-4-2$ and `ELU` activations over 3000 epochs with mini-batches of size 16 and the `Adam` optimizer with learning rate equal to 0.002. In the leftmost panel of Figure 8, we can see that the reconstruction of the slow manifold is correct. In the rightmost panel, we can observe that the encoder $\mathcal{E}$ learned the slow variable accurately. Notice that the relation between the slow map (16) and $\mathcal{E}$ is not affine in this case but results from a more complicated transformation.

As noted in Remark 3.1, the slow map is never unique. To assess the model, we compare the encoder to the slow map that results from inverting the hidden transformation (15), which may represent an optimal choice in this way. However, we should not expect that the encoder always finds this particular map, as other choices are readily possible, including the affine relation observed in the previous section. Here, we notice only a slight deviation from affinity. This indicates that we found the smallest architecture that can learn the correct slow map, as the more complex networks could stumble upon more complicated, yet correct, encoding.

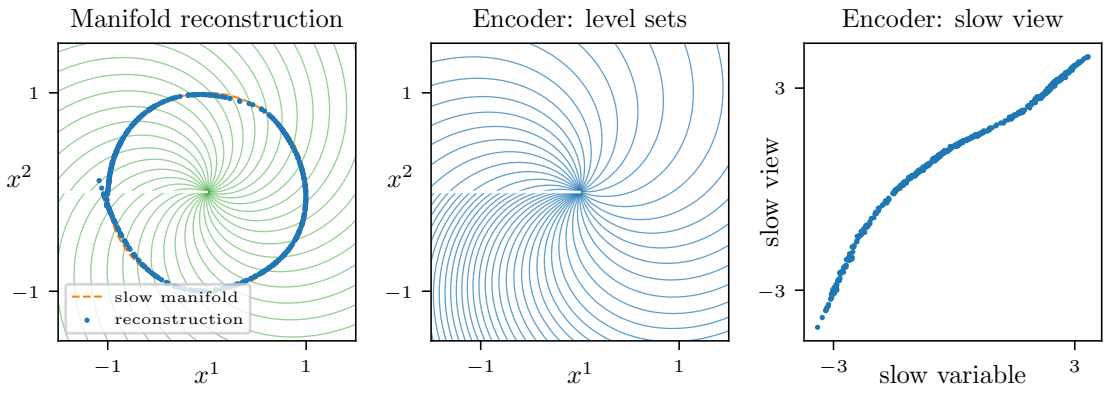

Figure 9: Visualization of the results of trained model for system from Section 5.2 with encoder prepended with cartesian-to-polar layer. (Left) Reconstruction of the slow manifold. (Center) The level sets of the encoder $\mathcal{E}$. (Right) Values of the encoder $\mathcal{E}$ plotted against the corresponding values of the slow map (16).

Looking at the level sets of the encoder in the central panel of Figure 8, we notice that the global patterns are not the same as for the slow map (left panel, in green). However, if we restrict ourselves to the region where the data is sampled (highlighted lines), the pattern resembles the part of level sets of the slow map in the respective area. As always, we can expect to obtain good accuracy of our models only in the part of the state space where the data resides.

5.2.1 Incorporating polar coordinates

To improve the performance of the network and to achieve better accuracy of the encoder, we can make use of additional knowledge about the system. In the case of the oscillating half-moons system this is the periodicity of the observed dynamics. It is an interesting question whether one can build architectures that discover such properties and automatically adjust. Here, we assume that discovering the periodicity of the system can be achieved without complicated learning approaches and thus used to design the models.

We observed that the plain encoder-decoder network has difficulties in learning a good representation in the slow view due to the discontinuity of the slow variable (16). To alleviate this problem and to demonstrate that one can achieve globally accurate encoding in such a test case, we prepend the encoder $\mathcal{E}$ with a layer that has no trainable parameters and computes the polar coordinates associated to the inputs it receives. As a result, the remaining part of the encoder, where the parameters are trained, works on transformed data. This approach can be easily generalized to higher-dimensional systems by making use of (hyper-)spherical coordinates.

We use the network of shape $2-[2]-4-4-1-4-4-2$, where [2] denotes the cartesian-to-polar layer, and the dataset comprising data points sampled from the full path of the system. We train this network over 2000 epochs with mini batches of size 16 and the `Adam` optimizer with learning rate set to 0.002.

In Figure 9, we display the reconstruction of the slow manifold (left), encoder level sets (center) and the accuracy of encoder as compared with (16) (right). As previously seen, the reconstruction of the slow manifold is correct but this time the encoder learned the level sets globally and the accuracy of the encoder is good. Notice that the relation between the slow view and the slow variable resembles the one obtained previously, cf. the right panel of Figure 8. This confirms that the previous model, trained on the restricted dataset, agrees on the corresponding region of the state space with the one found in this section.

5.3 Reconstructing the slow manifold via an autoencoder

Before considering higher dimensional systems, let us examine the use of unsupervised learning to parametrize the slow manifold. The goal is to demonstrate that the unsupervised approach based on the dataset of points from a sample path or their projections on the slow manifold do not produce satisfactory results. The method

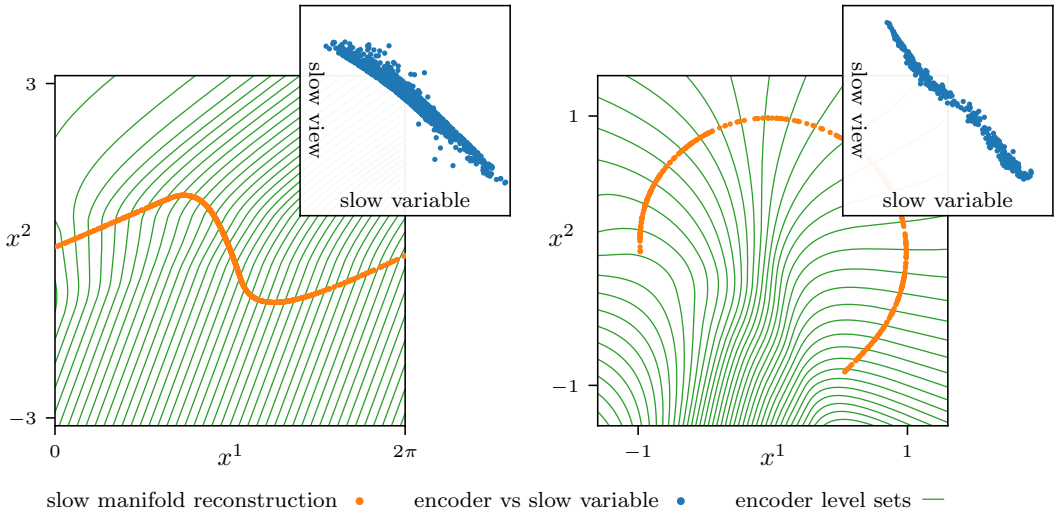

Figure 10: The results of training an autoencoder on the dataset composed solely of the values of the slow projection from Section 5.1 (left) and 5.2 (right). Plotted are the reconstruction of the slow manifold by an autoencoder (orange points), the level sets of its encoder (green curves), and the values of encoder against the slow map (blue points in insets). The reconstruction of the slow manifold is, as expected, accurate, but the level sets of the encoder do not correspond, even locally near the slow manifold, to the fast fibers of the system, cf. the left panels of Figures 6 and 8. This is reflected in poor correlation between the values of the encoder and the actual slow map as seen by comparing the inset plots with the right panels of Figures 6 and 8.

we present in this manuscript combines the two datasets—the points on the sample path as inputs and their respective projections as input labels—to achieve better outcome of learning.

An autoencoder finds a hidden representation of a low-dimensional dataset. The dataset comprising the points generated by the stochastic system fails, in general, to be a low-dimensional subset of the state space: in the stochastic case the trajectories of the process are only statistically bound to the slow manifolds and can undergo large fluctuations. Therefore, the low-dimensional representation of such a dataset is not well-defined, and this ambiguity may hinder the discovery of the correct slow parametrization.

In principle, since we can generate the approximation of the slow manifold by performing short bursting coupled with averaging, it is possible to train an autoencoder on the dataset comprising only of points approximating the slow manifold. This approach allows us to build the reduced representation in the latent space, but the question remains as to the relevance of the encoder to the quest of approximating the slow map. The goal of using an autoencoder for parametrizing the manifold is to have a selective reconstruction (only on a subset of the input space) while making the model insensitive to everything outside the manifold. In the cases considered, though, the geometry of the fast fibers is crucial for the problem and may not overlap with the fibers along which the autoencoder projects arbitrary points onto the manifold.

In Figure 10, we demonstrate that we cannot, in general, expect the autoencoder to find the accurate shape of fast fibers, even locally near the slow manifold. For this, we trained two autoencoders on the datasets form Sections 5.1 and 5.2, and display the results in the left and the right panel, respectively. The prediction of the slow manifold is very good (orange). However, the level sets of the encoder (green lines) in both cases do not correspond to the fast fibers of the corresponding system, cf. Figures 6 and 8. This misalignment, the more pronounced the further from the slow manifold we deviate, indicates that we cannot expect to learn the correct embedding without supervision. This is confirmed in the insets (blue) where we observe that there is poor correlation between the values of the encoder and the values of corresponding slow map on the test sets.

6 Measuring encoder approximation error and learning essential coordinates

In this section, we present an approach to measure an error in the accuracy of the encoder as an approximation of a slow map and employ a pruning strategy to select the essential coordinates of the observed process. To illustrate both of these concepts, we apply the proposed method to higher-dimensional test systems, defined in Section 6.1, for which a simple visualization of the encoder level sets becomes inaccessible.

The error, which we introduce in Section 6.2, measures the lack in orthogonality between the gradients of the encoder components and the eigenvectors pointing towards local fast directions. As noted earlier, in order for the encoder to learn an accurate approximation of a slow map, its level sets should globally overlap with the fast fibers of the observed system. Locally, it means that at every data point the row-space of the encoder Jacobian matrix should be orthogonal to the tangent space of the fast fibers at that point. Since we can compute the former using the backpropagation algorithm on the network and the latter is spanned by the eigenvectors accessible through the eigen-decomposition of local noise covariance matrix (see Figure 3), we obtain a viable error indicator.

We also employ a global pruning of the network as a means to automatically learn the essential coordinates of the observed process. In many high-dimensional systems only a few of the observed variables are involved in the hidden slow dynamics and identifying them constitutes an important task. To address this, we monitor the parameters (weights and biases) of the network during the training and switch off (i.e., fix to zero) the ones with the smallest magnitudes. Such magnitude pruning reduces the complexity of the network and learns the problem-specific architecture along with the parameters. Moreover, and crucial for our application, if the pruning of the parameters of the first layer becomes large enough, it can result in "cutting off" of some input nodes (representing the coordinates of observed process) from the rest of the network. In Section 6.3, we demonstrate that for a test system in which only a few input nodes build the slow transformation, the remaining nodes are consistently cut off from the network by pruning.

6.1 Test system with quadratic observation function

First, let us introduce a model equation we use in this section. It is a slow-fast system that, in addition to the timescale separation ϵ, is parametrized by the dimensions D^s and D^f of slow and fast dynamics, respectively. This allows to choose different test cases with a given dimensionality and the dimensions of the slow and the fast dynamics set in a particular ratio.

Specifically, we consider a generalization of the system used in [9]. The hidden process is given by a set of uncoupled slow-fast equations

$$\begin{aligned} \mathrm{d}Y_t^d &= \mathrm{d}t + \mathrm{d}U_t^d, && d = 1, \dots, D^s, \\ \mathrm{d}Z_t^d &= -\frac{1}{\epsilon} Z_t^d \,\mathrm{d}t + \frac{1}{\sqrt{\epsilon}} \,\mathrm{d}V_t^d, && d = 1, \dots, D^f. \end{aligned} \tag{17}$$

The slow process Y_t is unstable and, as time progresses, it grows on average with constant rate. The fast process Z_t is a stable Ornstein-Uhlenbeck process, so its values center around the origin with normally distributed fluctuations of variance $1/\epsilon$. In all numerical examples, except for Section 6.2.2, we set $\epsilon = 10^{-3}$.

We assume that $D^s \leq D^f$ and consider an observed system obtained from (17) by the map

$$f : \begin{cases} x^d = y^d + (z^d)^2, & d = 1, \dots, D^s, \\ x^{D^s+d} = z^d, & d = 1, \dots, D^f. \end{cases} \tag{18}$$

The observation function (18) mixes the squares of the first D^s fast variables with the corresponding slow ones leaving the remaining $D^f - D^s$ fast variables unchanged. These remaining fast variables do not influence the slow dynamics and can be thought of as additional noise present in the system with no impact on the hidden slow process. Clearly, f is a diffeomorphism and the slow map is given by

$$y^d = x^d - \left(x^{D^s+d}\right)^2, \quad d = 1, \dots, D^s. \tag{19}$$

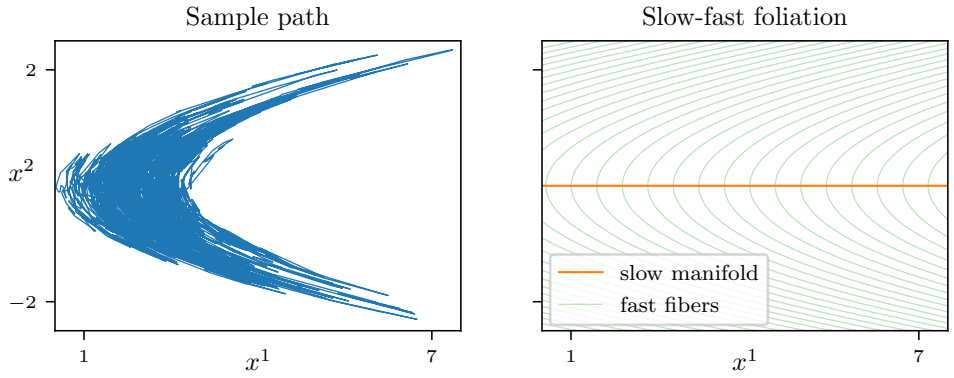

Figure 11: A sample path (left) and the slow manifold with fast fibers (right) of the two-dimensional version of (20). The parabolic shapes of fast fibers also characterize the higher dimensional versions of (20) when plotting x^d against x^{D^s+d} for any $d = 1, \dots, D^s$.

By the Itô formula, the resulting observed system reads

$$\begin{aligned} \mathrm{d}X_t^d &= \frac{1}{\epsilon}\left[1 + \epsilon - 2\left(X_t^{D^s+d}\right)^2\right]\mathrm{d}t \;+\; \mathrm{d}W_t^d + \frac{2}{\sqrt{\epsilon}} X_t^{D^s+d}\,\mathrm{d}W_t^{D^s+d}, \quad d = 1, \dots, D^s, \\ \mathrm{d}X_t^{D^s+d} &= -\frac{1}{\epsilon} X_t^{D^s+d}\,\mathrm{d}t \;+\; \frac{1}{\sqrt{\epsilon}}\,\mathrm{d}W_t^{D^s+d}, \qquad\qquad\qquad\quad d = 1, \dots, D^f. \end{aligned} \tag{20}$$

As in previous examples, notice that the coefficients involving the timescale separation parameter ϵ are mixed in all coordinates of the observed process (20). However, we can still distinguish two groups of coordinates of size D^s and D^f. The first group of D^s coordinates mixes the noise of intensity one with the noise of intensity proportional to $1/\epsilon$. Additionally, the latter noise correlates the coordinates from the first group with the initial D^s coordinates of the second group, as they are driven by the same Brownian motions. This correlation underpins the hidden slow dynamics in (20) and eventually will be collapsed by a network after the training.

A characteristic feature that results from using observation function (18) are the parabolic shapes visible when plotting the (x^d, x^{D^s+d})-projections of the paths of the full observed system, illustrated in Figure 11 for the two-dimensional case. These shapes arise from the intersection of the parabolic fast fibers of the observed process with the (x^d, x^{D^s+d})-plane. Finally, let us note that the D^s-dimensional slow manifold of (20) is contained in the hyper-plane spanned by the first D^s coordinates.

6.2 Testing the slow view

The goal in this section is to introduce an error measure with which we can assess the accuracy of the slow view without explicitly knowing the slow variable of the observed system. To this end, we consider a four-dimensional version of system (20) with a one-dimensional slow variable ($D^s = 1$, $D^f = 3$). This setting allows us to visualize, as in the previous section, the accuracy of the encoder by plotting its values on the test dataset against the corresponding values of the slow map (19). We compare the results of such visualization with the magnitude of the error in Section 6.2.1 to demonstrate that the error indeed captures the inaccuracies in the slow view. For our experiments, we generate training and test datasets of 2010 and 911 instances, respectively, as described in Section 4.2. We use `ELU` activation function on hidden layers, the `Adam` optimizer with learning rate 0.001, and train with batches of size 16. We test three architectures of increasing complexities, see Table 1, for which we inspect the reconstruction of the slow manifold and compare the values of the slow view to the slow variable (19).

Let us first look at the top row of Figure 12, where we plot, in the (x^1, x^2)-view of the state space, the reconstruction of the slow manifold (orange). The quality of this reconstruction is very similar for three models: all find the correct geometry. However, looking at the value of the smallest validation loss in Table 1, we can clearly see that Models 2 and 3 found better minima than Model 1. This discrepancy is due to the

Model	Layer sizes	Max. epochs	Min. validation loss
Model 1	4 - 2 - 1 - 2 - 4	2000	0.1063
Model 2	4 - 3 - 1 - 3 - 4	2000	0.0029
Model 3	4 - 4 - 1 - 4 - 4	2000	0.0029

Table 1: Architectures and training details for models from Section 6.2. Other hyperparameters remain unchanged across the runs: activation `ELU`, the `Adam` optimizer with learning rate 10^{-3}, batch size 16.

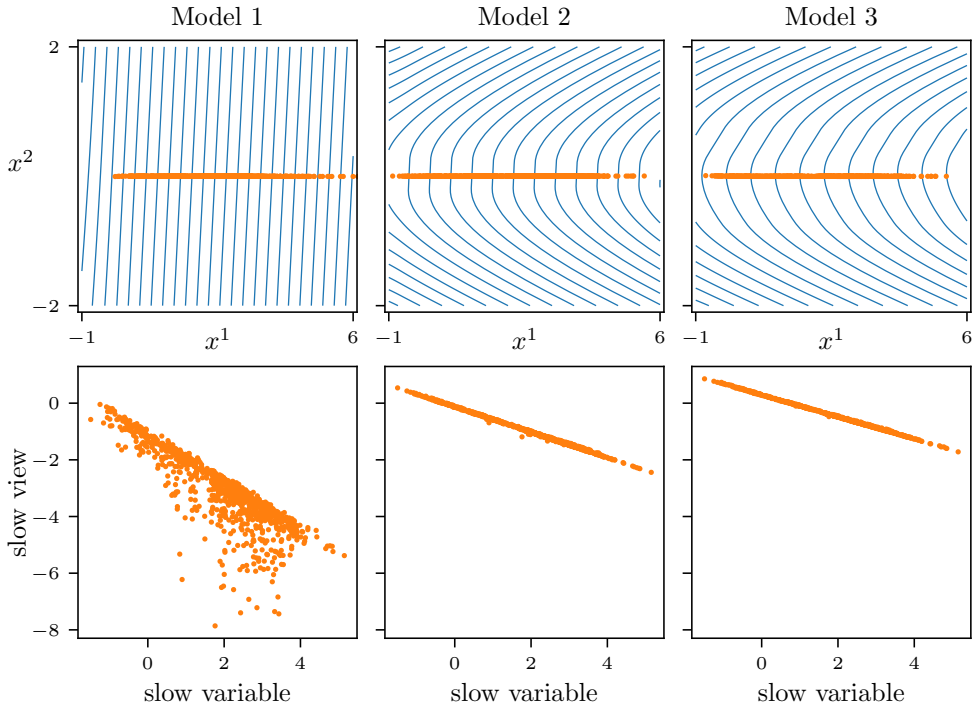

Figure 12: Performance of the models from Section 6.2 on a four-dimensional test dataset for different network architectures (in columns). (Top row) Reconstruction of the slow manifold (orange) and the encoder level sets (blue). (Bottom row) The accuracy of the slow view: values of the encoder versus the corresponding values of the slow map (19). As the architecture complexity of the networks increase from left to right, the models reach lower validation loss (see Table 1) and we get better accuracy of the slow variable (bottom row). Note, however, that the geometry of the slow manifold is well reconstructed by all models (top row, orange), so the difference in accuracy is due mainly to the quality of the approximation of the fast fibers (top row, blue).

simplicity of the geometry of the slow manifold, so that even the smallest architecture, Model 1, can learn the reconstruction correctly, without necessary achieving small loss.

The difference between Model 1 and Models 2 and 3 becomes more pronounced when we look at the approximation of the fast fibers (blue curves) in the top row of Figure 12. Only Models 2 and 3 capture the characteristic parabolic shapes of the fast fibers in the (x^1, x^2)-plane. Model 1 finds a much simpler encoder level sets, which do not correspond to the fast fibers. As a consequence, the whole network $\mathcal{N}$ projects data points onto the slow manifold, but not at the correct place on the manifold as determined by the projection $\mathcal{P}$. In other words, $\mathcal{N}$ captures the correct geometry of the slow manifold $\mathbb{S}$ but the approximation of the associated projection mapping $\mathcal{P}$ remains insufficient. As a result, the validation loss of Model 1 is much larger than for other two models. This results from the fact that the architecture of Model 1, though sufficient to capture a

parametrization of $\mathbb{S}$ by the decoder $\mathcal{D}$, lacks the required complexity in the encoder $\mathcal{E}$ to reproduce the level sets correctly.

Finally, we can confirm that Models 2 and 3 succeeded in learning the slow variable by looking at the plots in the bottom row of Figure 12, which present the scatter plot of the slow view of the encoder against the slow variable (19). We observe that, unlike for Model 1, there is an affine relation between the corresponding slow representations of the data points from the test set.

6.2.1 Measuring local non-orthogonality to the fast directions

We introduce an error measure that is based on the fact that the slow map's level sets align with the fast fibers of the system. Therefore, at each point in the state space, the subspace spanned by the derivative of the slow map is orthogonal to the tangent space of the fast fibers of the system. Since the encoder should approximate the slow map, one way to measure the quality of this approximation is to measure the lack of orthogonality between the subspace spanned by its derivative and the subspace spanned by the local fast directions.

We can obtain the slow and fast directions by computing the eigenvectors of the covariance matrices, see the right plot in Figure 3. The D^f eigenvectors aligning with the local fast directions are the ones corresponding to D^f the largest eigenvalues of the local noise covariance matrices. At every data point x for which we want to evaluate the error, we compute the eigen-decomposition of $\sigma(x)$ and assemble the fast eigenvectors into the columns of a $D \times D^f$-dimensional matrix $V^f(x)$. To obtain the derivative of the encoder with respect to the input x we use backpropagation on $\mathcal{E}$. This yields the transposed Jacobian matrix $J_{\mathcal{E}}(x)^\mathsf{T} \in \mathbb{R}^{D \times D^s}$.

Next, we consider the block matrix

$$U(x) = \left[V^f(x) \,|\, \widehat{J_{\mathcal{E}}(x)}^\mathsf{T}\right] \in \mathbb{R}^{D \times D}$$

combining the fast eigenvectors and the normalized derivative of the encoder at the same data point x. The better the encoder approximates the slow map, the more orthogonal the matrix function U will be across the state space of the system. To measure the lack of orthogonality, we compute the following error

$$E(x) = \|U(x)^\mathsf{T} U(x) - I\|_F / \sqrt{D}, \tag{21}$$

where I is a D-dimensional identity matrix and $\|\cdot\|_F$ the Frobenius norm.

In Figure 13, we present the statistics of the error $E(x)$ for Models 1-3, computed over the test dataset. Considering only the median of the error confirms that Model 1 is worse than Models 2 and 3. Moreover, though the medians of Models 2 and 3 are essentially the same, we notice that the variability of the error across the data stays lower for Model 3. This lower variability results from a slightly better representation of the fast fibers close to the slow manifold achieved by Model 3; as seen in Figure 12, the level sets for Model 2 have flattened tips of the parabolas. Notice also that the difference between Model 2 and 3 is not reflected in the value of the validation loss (Table 1) for these two models.

6.2.2 Error dependence on timescale separation

Finally, let us look at how the orthogonality error behaves as a function of ϵ in equation (17). As the range for the parameter ϵ we discretize the interval $[0.001, \ldots, 1]$ and for each value from the mesh we generate the corresponding dataset. We train the networks with the same architecture as Model 3, the best performing model from the previous section, using the same hyperparameters. After the training, we compute for each model error (21) at all points from the corresponding dataset and look at the mean and standard deviation of these values. Additionally, to see the trend more clearly, we generate a number of independent replicas of models for each ϵ from the mesh and average their means and standard deviations.

We present the obtained results in Figure 14. As expected, the average error decreases when ϵ gets smaller and the timescale gap grows. We can also observe that the variability of model accuracy shrinks in the same limit. Finally, when ϵ gets closer to 1 and the gap closes, there is much more rapid growth of the error than across the remaining part of the interval on which ϵ ranges.

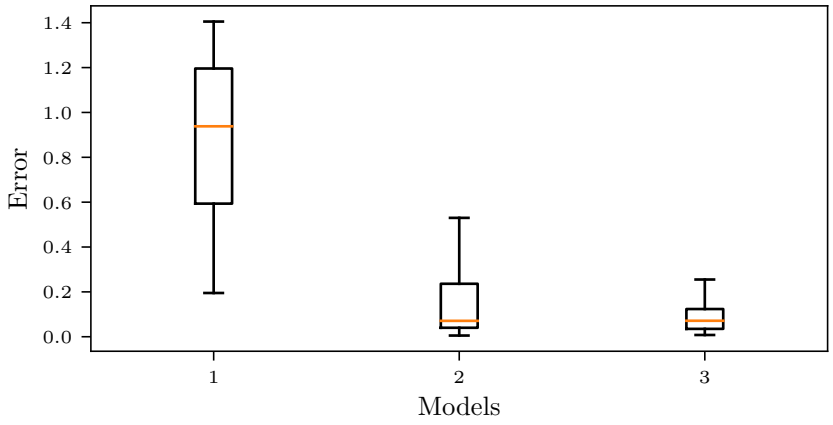

Figure 13: Statistics of the orthogonality error (21) between the encoder gradients and the local fast directions for models from Section 6.2: median (orange horizontal line), lower and upper quartile (box), 1.5 interquartile range (whiskers). The statistics are based on the gradients of the encoder computed at all points from the test dataset. In accordance with Figure 12, the error decreases as the architecture complexity increases from left to right. We can identify the slightly better accuracy of Model 3 over Model 2, revealed in lower error variability of the former model (see also the discussion in the last paragraph of Section 6.2.1.)

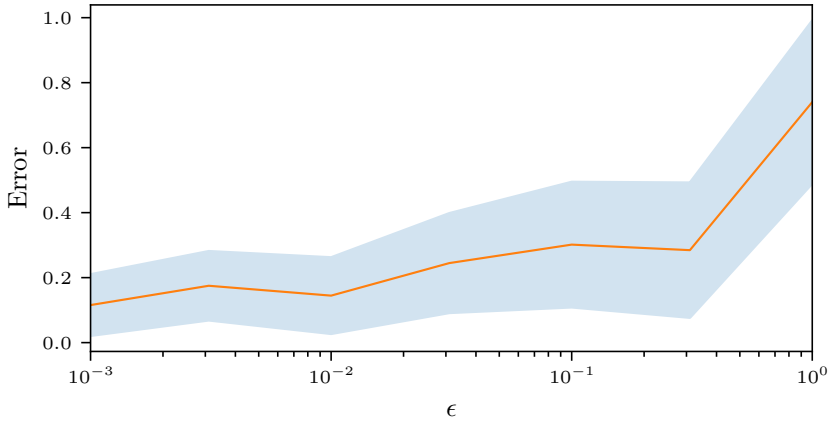

Figure 14: The mean and standard deviation of orthogonality error (21) of Model 3 from Section 6.2 as a function of timescale separation parameter ϵ in equation (17). The results are averaged for each ϵ over 10 model replicas based on independent training sessions. Besides some random fluctuations due to finite number of replicas, the error decreases as the timescale separation $1/\epsilon$ increases. We can also see that the variability in the quality of models also decreases as the separation increases. Moreover, when we approach $\epsilon = 1$ and the timescale gap vanishes, we can observe rapid growth of the error.

Model	Layer sizes	Max. epochs	Min. validation loss
Model 0	10 - 5 - 2 - 5 - 10	3000	0.0044
Model 1	10 - 6 - 4 - 2 - 4 - 6 - 10	3000	0.0029
Model 2	10 - 8 - 4 - 2 - 4 - 8 - 10	3500	0.0025
Model 2p	10 - 8 - 4 - 2 - 4 - 8 - 10	4500	0.0025
Model 3	10 - 8 - 6 - 4 - 2 - 4 - 6 - 8 - 10	3500	0.0025
Model 3p	10 - 8 - 6 - 4 - 2 - 4 - 6 - 8 - 10	5500	0.0024
Model 4	10 - 12 - 8 - 4 - 2 - 4 - 8 - 12 - 10	3000	0.0025
Model 4p	10 - 12 - 8 - 4 - 2 - 4 - 8 - 12 - 10	6000	0.0024

Table 2: Architectures and training details for models trained on a ten-dimensional dataset where we suffix the pruned models with 'p'. Other hyperparameters stay the same across the runs: activation function `ELU`, the `Adam` optimizer with learning rate 10^{-3}, batch size 16.

6.3 Localizing the essential coordinates via pruning

In this section, in addition to learning the slow map by the encoder, we look at the possibility of pinpointing which coordinates of the observed system are involved in the slow variable. It is often the case that in complex, high-dimensional systems only a few of the system's observed coordinates constitute an essential part of the low-dimensional slow dynamics. Finding these coordinates can significantly increase our ability to simulate efficiently the long time dynamics of such system. To accomplish this task, we explore pruning schemes.

Pruning is a method to induce sparsity in the artificial neural network by setting a subset of the model parameters to zero. While numerous techniques for inducing sparsity have been proposed over the last few years [16], in this work we use a simple unstructured iterative pruning that removes small magnitude weights and biases. More specifically, we employ the `l1_unstructured` pruning method from the `torch.nn.utils` module. This method removes a fixed amount of (the currently unpruned) parameters with the lowest L^1-norm. We aggregate the parameters from a number of specified layers, prior to deciding which ones to prune.

We do not prune the parameters of the bottleneck layer and the output layer. For the remaining layers, the amount of parameters we prune at once varies usually between 3-5%. Also, the pruning schedule—i.e., when we start pruning, how often we prune, and when we stop—depends on the model. We usually initialize the pruning strategy after the first 100 epochs and prune every fixed amount of epochs until the desired global sparsity has been reached. After the desired sparsity has been achieved, we continue training for a number of epochs to let the remaining parameters adjust.

We use a ten-dimensional version of (20) with two slow variables as an example test case. According to (19), the slow map of such system will depend only on the first four coordinates $x^1, \dots, x^4$ of the observed system. The remaining six coordinates do not impact the slow dynamics and constitute noise in the observation.

To test the pruning, we train a number of models with increasingly larger architectures, see Table 2, both without and with pruning, on a dataset with 1407 train instances and 603 validation instances taken from a sample path as described in Section 4.2. We use `ELU` activation function on hidden layers, the `Adam` optimizer with learning rate 0.001, and mini-batches of size 16. The first two models are fairly small, and we do not consider the pruned versions. Note also that, besides Model 0, all models reach a similar validation loss. Since we did not observe any over-fitting when training the models, we argue that monitoring solely the validation loss is insufficient to assess the accuracy of the networks trained with our method. This is why other metrics, such as the orthogonality error (21), should be used in early stopping.

In Figure 15 we present the statistics of (21). We can see that Model 1 performs better than Model 0, in accordance with the decrease of the validation loss. However, as we increase the size of the networks the accuracy of non-pruned models becomes worse: they have larger variability and the average value of the error increases for Models 3 and 4. This is a result of over-fitting: the fibers of the encoder become more complicated than necessary which may worsen the orthogonality. As mentioned in the previous paragraph, this over-fitting is not captured by the validation loss alone.

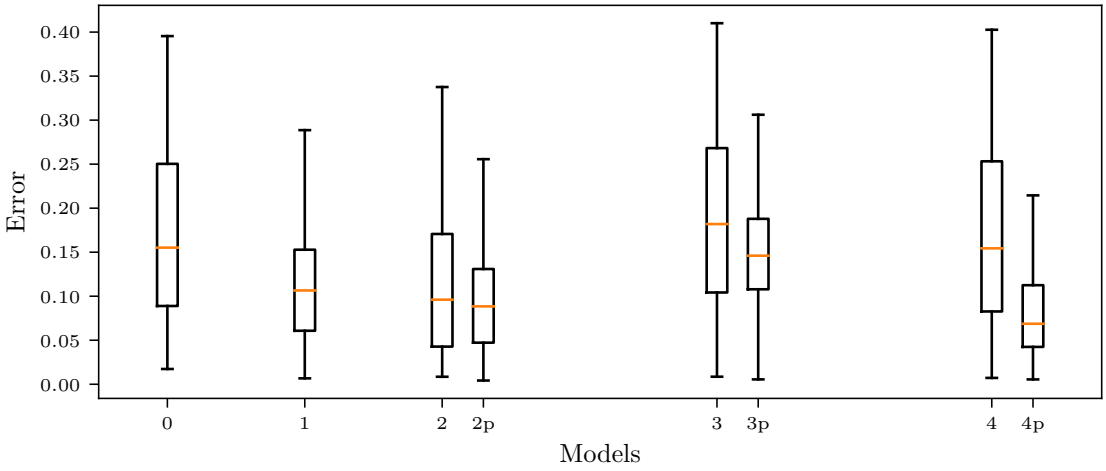

Figure 15: Statistics of the orthogonality of encoder gradients to the local fast directions for different models: median (horizontal line), lower and upper quartile (box), 1.5 interquartile range (whiskers). The plot is based on gradients of the encoder at all points from the test dataset. We suffix the pruned models with 'p'.

Model	Sparsity per layer [%]	Total sparsity [%]
Model 2p	79 - 19 - 0 - 16 - 10 - 0	30
Model 3p	84 - 24 - 32 - 0 - 16 - 43 - 33 - 0	35
Model 4p	87 - 65 - 47 - 0 - 25 - 55 - 82 - 0	55

Table 3: Results of global pruning for each layer of the network. The slow view and output layer were not pruned. Among the remaining layers, the first layer is consistently the most pruned one, indicating that it contains the most redundancy.

Adding the pruning strategy to the training process may improve the accuracy of models with architectures that are too rich. Since we do not know beforehand how large the network should be to accurately model the dynamics, removing nodes may prevent over-fitting. This is most visible for the largest model—Model 4. In fact, Model 4p is the most accurate model we could get among the tested architectures. We were also able to diminish the error variability of Model 2 and Model 3.

In general, the proper tuning of the pruning hyperparameters is challenging and may require substantial testing. We did obtain slight improvements, but this aspect of pruning is not conclusive within the tests we performed. From what we observe by looking at the error, we can also conclude that Model 1 seems to already be a good fit for the test system we use here and not much improvement can be made. To gain a better understanding of the possible improvements in accuracy with pruning strategies a more challenging test case should be used.

We summarize the most important effect of pruning in our case in Table 3 and Figure 16: due to a limited number of observed coordinates involved in the slow dynamics, pruning results in high sparsification of the weight's matrix of the first layer of the networks. In Table 3, we compare the amount of sparsification per layer in the pruned networks. Note that we did not use pruning for the parameters of the bottleneck layer and the output layer. The biggest percentage of pruned parameters appears consistently in the first layer, although we used a global pruning strategy that looks at the parameters closest to zero across all active layers. This clearly indicates that the first layer is the place where most redundancy is present.

Furthermore, the pruning cuts off non-essential input nodes from the rest of the network, indicating which input nodes (representing the coordinates of the observed system) influence the latent view. In Figure 16, we display the activation of weight matrices of the first layers for the pruned models. The orange squares indicate that the weight is different from zero. Notice that rows 5-10 have been completely pruned out: the input nodes

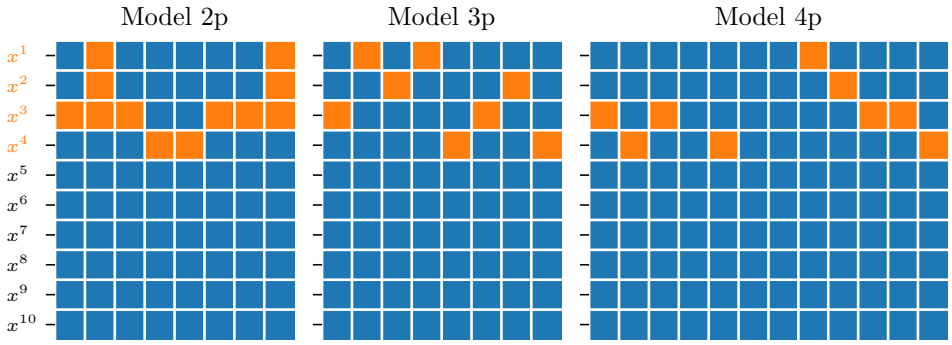

Figure 16: Active (light) and inactive (dark) weights of the first layer of the pruned models. The rows of these matrices represent the weights attached to the coordinates of the observed system. The coordinates with full row inactive are cut off from the rest of the network. For all models, only the first four coordinates, which build up the slow map (19), remain connected. This indicates that the pruning can discover essential coordinates that constitute the slow variable.

corresponding to these rows are no longer connected to the network and thus do not influence the prediction. The rows that still contain active weights correspond to the first four coordinates of the system—exactly those coordinates that constitute the slow dynamics.

7 Conclusions and outlook

The main contribution in this paper is to introduce a learning method based on approximation properties of artificial neural networks that is able to discover the slow variables in stochastic systems with multiple time scales. The networks share the architecture with autoencoders, but we train them in a supervised manner to approximate the projection map along the fast fibers onto the slow manifold of the observed system. We demonstrated, on a set of test examples, that the encoder part of the network can successfully learn a correct slow representation. A number of possible improvements in the architecture and training of the networks exists which we did not explore in this initial study but which may produce better results and become useful in applications to more demanding multiscale systems.

We can visually inspect the accuracy of encoders in approximating the slow map on examples with one-dimensional slow representation. To measure this accuracy we introduced an error based on the lack of orthogonality between the gradients of the encoder and the local fast directions. The error can be easily computed using the backpropagation algorithm with respect to the input on the encoder part of the network. This allows one to not only compute it for trained models, but also using it as a monitoring metric during training. Since, the accuracy is not always reflected in the value of the loss function, i.e., models with the same validation loss still can have noticeably different accuracies, having a metric based on the error allows implementing early stopping and improving the results of training.

The architecture of our network is, as for autoencoders, symmetric. However, while the encoder part of the network is responsible for collapsing the fast fibers that foliate the whole state space, the decoder's task is only to parametrize the lower-dimensional slow manifold. Therefore, it seems reasonable to expect the same or better performance using non-symmetric architecture, in which the decoder part is smaller than encoder. On one hand, this can reduce the size of the network, thus speeding up the training. On the other hand, if such architecture indeed fits the problem better, this can prevent the overfitting. Thorough tests of such architectures on various systems could provide evidence for this hypothesis.

Finally, we used pruning to adjust the architecture to a given problem and to prevent overfitting when starting with too large a network. Due to the simple nature of our test systems, we only achieved modest improvements in these respects and further tests on more challenging systems should follow. However, with pruning we accomplished a more important goal in our case: we were able to locate the essential coordinates of the

observed system that build up the slow map. Knowing such coordinates opens up the possibility to construct more efficient algorithms for the simulation of high-dimensional systems. In this study, we employed the simplest magnitude pruning with aggregation over a number of layers. More intricate techniques are available in the literature, and it remains to be seen whether they can perform even better in certain situations. Since our goal is to remove the whole rows from the weight matrix of the first layer, the techniques that impose block structure on the spare weights, more efficient in parallel processing, can provide an interesting modification to explore in the future work.

Acknowledgments

This work was partially supported by AFOSR under grant FA9550-17-1-9241.

References

[1] Nils Berglund and Barbara Gentz. "Geometric singular perturbation theory for stochastic differential equations". In: *Journal of Differential Equations* 191.1 (2003), pp. 1–54. DOI: 10.1016/S0022-0396(03)00020-2.

[2] Pinaki Bhattacharya and Marco Viceconti. "Multiscale modeling methods in biomechanics". In: *WIREs Systems Biology and Medicine* 9.3 (2017), e1375. DOI: https://doi.org/10.1002/wsbm.1375.

[3] Andreas Bittracher and Christof Schütte. "A weak characterization of slow variables in stochastic dynamical systems". In: *Advances in Dynamics, Optimization and Computation*. Ed. by Oliver Junge et al. Vol. 304. Cham: Springer International Publishing, 2020, pp. 132–150. DOI: 10.1007/978-3-030-51264-4_6.

[4] Andreas Bittracher et al. "Transition manifolds of complex metastable systems: theory and data-driven computation of effective dynamics". In: *Journal of Nonlinear Science* 28.2 (2018), pp. 471–512. DOI: 10.1007/s00332-017-9415-0.

[5] Kathleen P. Champion, Steven L. Brunton, and J. Nathan Kutz. "Discovery of nonlinear multiscale systems: sampling strategies and embeddings". In: *SIAM Journal on Applied Dynamical Systems* 18.1 (2019), pp. 312–333. DOI: 10.1137/18M1188227.

[6] Kathleen P. Champion et al. "Data-driven discovery of coordinates and governing equations". In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22445–22451. DOI: 10.1073/pnas.1906995116.

[7] Daan Crommelin and Eric Vanden-Eijnden. "Reconstruction of diffusions using spectral data from timeseries". In: *Communications in Mathematical Sciences* 4.3 (2006), pp. 651–668. DOI: 10.4310/CMS.2006.v4.n3.a9.

[8] Kristian Debrabant, Giovanni Samaey, and Przemysław Zieliński. "A micro-macro acceleration method for the monte carlo simulation of stochastic differential equations". In: *SIAM Journal on Numerical Analysis* 55.6 (2017), pp. 2745–2786. DOI: 10.1137/16M1066658.

[9] Carmeline J. Dsilva et al. "Data-driven reduction for a class of multiscale fast-slow stochastic dynamical systems". In: *SIAM Journal on Applied Dynamical Systems* 15.3 (2016), pp. 1327–1351. DOI: 10.1137/151004896.

[10] Weinan E, Di Liu, and Eric Vanden-Eijnden. "Analysis of multiscale methods for stochastic differential equations". In: *Communications on Pure and Applied Mathematics* 58.11 (2005), pp. 1544–1585. DOI: 10.1002/cpa.20088.

[11] Weinan E, Di Liu, and Eric Vanden-Eijnden. "Nested stochastic simulation algorithms for chemical kinetic systems with multiple time scales". In: *Journal of Computational Physics* 221.1 (2007), pp. 158–180. DOI: 10.1016/j.jcp.2006.06.019.

[12] Radek Erban and S. Jonathan Chapman. *Stochastic Modelling of Reaction–Diffusion Processes*. 1st ed. Cambridge University Press, 2019. DOI: 10.1017/9781108628389.

[13] Marco Favino et al. *Multiscale modeling, discretization, and algorithms: a survey in biomechanics*. 2018. URL: http://arxiv.org/abs/1609.07719 (visited on 04/19/2021).

[14] Gary Froyland, Georg A. Gottwald, and Andy Hammerlindl. "A computational method to extract macroscopic variables and their dynamics in multiscale systems". In: *SIAM Journal on Applied Dynamical Systems* 13.4 (2014), pp. 1816–1846. DOI: 10.1137/130943637.

[15] Gary Froyland, Georg A. Gottwald, and Andy Hammerlindl. "A trajectory-free framework for analysing multiscale systems". In: *Physica D: Nonlinear Phenomena* 328-329 (2016), pp. 34–43. DOI: 10.1016/j.physd.2016.04.010.

[16] Trevor Gale, Erich Elsen, and Sara Hooker. *The state of sparsity in deep neural networks*. 2019. URL: http://arxiv.org/abs/1902.09574 (visited on 03/29/2021).

[17] Erik van der Giessen et al. "Roadmap on multiscale materials modeling". In: *Modelling and Simulation in Materials Science and Engineering* 28.4 (2020), 043001 (61pp). DOI: 10.1088/1361-651X/ab7150.

[18] Dror Givon, Raz Kupferman, and Andrew Stuart. "Extracting macroscopic dynamics: model problems and algorithms". In: *Nonlinearity* 17.6 (2004), R55–R127. DOI: 10.1088/0951-7715/17/6/R01.

[19] Markos A. Katsoulakis and Pedro Vilanova. "Data-driven, variational model reduction of high-dimensional reaction networks". In: *Journal of Computational Physics* 401 (2020), p. 108997. DOI: 10.1016/j.jcp.2019.108997.

[20] Ioannis G. Kevrekidis and Giovanni Samaey. "Equation-free multiscale computation: algorithms and applications". In: *Annual Review of Physical Chemistry* 60.1 (2009), pp. 321–344. DOI: 10.1146/annurev.physchem.59.032607.093610.

[21] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[22] Frédéric Legoll and Tony Lelièvre. "Effective dynamics using conditional expectations". In: *Nonlinearity* 23.9 (2010), pp. 2131–2163. DOI: 10.1088/0951-7715/23/9/006.

[23] Andrew J. Majda, Ilya Timofeyev, and Eric Vanden Eijnden. "A mathematical framework for stochastic climate models". In: *Communications on Pure and Applied Mathematics* 54.8 (2001), pp. 891–974. DOI: 10.1002/cpa.1014.

[24] Robert T. McGibbon, Brooke E. Husic, and Vijay S. Pande. "Identification of simple reaction coordinates from complex dynamics". In: *The Journal of Chemical Physics* 146.4 (2017), p. 044109. DOI: 10.1063/1.4974306.

[25] T. N. Palmer. "Stochastic weather and climate models". In: *Nature Reviews Physics* 1.7 (2019), pp. 463–471. DOI: 10.1038/s42254-019-0062-2.

[26] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: Vancouver, Canada, 2019, p. 12.

[27] Matej Praprotnik, Luigi Delle Site, and Kurt Kremer. "Multiscale simulation of soft matter: from scale bridging to adaptive resolution". In: *Annual Review of Physical Chemistry* 59.1 (2008), pp. 545–571. DOI: 10.1146/annurev.physchem.59.032607.093707.

[28] Anthony J. Roberts. "Normal form transforms separate slow and fast modes in stochastic dynamical systems". In: *Physica A: Statistical Mechanics and its Applications* 387.1 (2008), pp. 12–38. DOI: 10.1016/j.physa.2007.08.023.

[29] Amit Singer et al. "Detecting intrinsic slow variables in stochastic dynamical systems by anisotropic diffusion maps". In: *Proceedings of the National Academy of Sciences* 106.38 (2009), pp. 16090–16095. DOI: 10.1073/pnas.0905547106.

[30] Stefanie Winkelmann and Christof Schütte. *Stochastic Dynamics in Computational Biology*. Frontiers in Applied Dynamical Systems: Reviews and Tutorials. Cham: Springer International Publishing, 2020. DOI: 10.1007/978-3-030-62387-6.