

A Multi-Fidelity approach to Deep Kernel Learning of dynamical systems from high-dimensional data sources

Tesi di Laurea Magistrale in Mathematical Engineering - Statistical Learning

Author: Federico Lancellotti

Student ID: 10685832

Advisor: Prof. Andrea Manzoni

Co-advisor: Dr. Nicolò Botteghi

Academic Year: 2023-2024



**POLITECNICO
MILANO 1863**

Table of contents

1. Introduction
2. Preliminaries
3. Methods
4. Numerical results
5. Conclusions

Introduction

Introduction

When modelling dynamical systems, data is often **high-dimensional**.

Dynamical systems can be described by a **small number of state variables**.

High-fidelity data is usually well-correlated with their **low-fidelity** counterpart.

Introduction

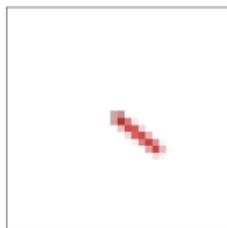
We present a Multi-Fidelity generalization of the framework presented by Botteghi et al., considering:

- both **high** and **low fidelity** measurements;
- a larger variety of test cases.

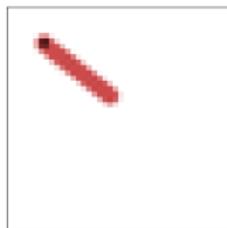
The goal of our model is to **reduce the dimensionality** of the input data, **learn the latent state and dynamics** of the system, and **quantify the uncertainty**.

Introduction

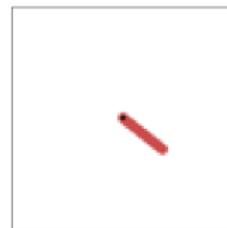
In particular, we will consider **RGB videos** of the dynamical systems.



(a) Fidelity level 0



(b) Fidelity level 1



(c) Fidelity level 2

Preliminaries

Gaussian Process Regression

Definition

Gaussian Process Regression (GPR) is a non-parametric regression technique that models the prior knowledge about a computational model $G(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$, with a Gaussian process $Z(\mathbf{x})$, with mean function $m(\mathbf{x})$ and kernel function $k(\mathbf{x}, \mathbf{x}')$.

Consider the training data inputs $\mathcal{X} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)})$, $\mathbf{x}^{(i)} \in \mathbb{R}^n$, and the corresponding training targets $\mathcal{Y} = (G(\mathbf{x}^{(1)}), G(\mathbf{x}^{(2)}), \dots, G(\mathbf{x}^{(N)}))$.

The predictive distribution evaluated at N^* test data points \mathcal{X}^* is

$$[Z(\mathcal{X}^*) | Z(\mathcal{X}) = \mathcal{Y}, \sigma^2, \theta] \sim \mathcal{N}(\mathbb{E}[Z(\mathcal{X}^*)], C(Z(\mathcal{X}^*)))$$

Drawback

Fitting the model is expensive for high-dimensional and large datasets.

Deep Kernel Learning

Embed a deep NN into the kernel function:

$$k(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) \rightarrow k(g(\mathbf{x}, \mathbf{w}), g(\mathbf{x}', \mathbf{w}); \boldsymbol{\theta}, \mathbf{w}) ,$$

where $g(\mathbf{x}, \mathbf{w})$ is a **non-linear mapping** given by a deep architecture, parametrised by weights \mathbf{w} .

Stochastic Variational Deep Kernel Learning

We approximate the posterior distribution with the best fitting Gaussian to a set of inducing data points sampled from the posterior.

Multi-Fidelity methods

The core idea is to leverage

- a large amount of **cheap low-fidelity** (LF) data,
- a limited number of **high-fidelity** (HF) samples, from detailed simulations or accurate sensors,

and **exploit their correlation.**

Linear Model of Coregionalisation

Each level of solution $Z_l(\mathbf{x})$ can be written as a **linear combination of L independent Gaussian processes** $u_i \sim \mathcal{GP}(0, k_i(\cdot, \cdot))$:

$$Z_l(\mathbf{x}) = \sum_{i=0}^{L-1} a_{l,i} u_i(\mathbf{x}), \quad l = 0, 1, \dots, L-1,$$

Levina-Bickel algorithm

Consider a MLE of the dimension ID from i.i.d. observations $\mathbf{z}_1, \dots, \mathbf{z}_n$ in \mathbb{R}^p . The observations represent an embedding of a lower-dimensional sample, i.e. $\mathbf{z}_i = g(\mathbf{s}_i)$, where \mathbf{s}_i are sampled from an unknown smooth density f on \mathbb{R}^{ID} .

The **global ID estimator** is calculated as

$$ID_{L-B} = \frac{1}{N} \sum_{i=1}^N \frac{1}{k-2} \sum_{j=1}^{k-1} \log \frac{T_k(\mathbf{z}_i)}{T_j(\mathbf{z}_i)} \quad ,$$

where $T_k(\mathbf{z})$ is the Euclidean distance between \mathbf{z} and its k -th nearest neighbor in $\mathbf{z}_1, \dots, \mathbf{z}_n$.

Methods

Consider the following nonlinear dynamical system:

$$\frac{d}{dt}\mathbf{s}(t) = \mathcal{F}(\mathbf{s}(t)), \quad \mathbf{s}(t_0) = \mathbf{s}_0, \quad t \in [t_0, t_f]$$

- $\mathbf{s}(t) \in \mathcal{S} \subset \mathbb{R}^d$ is the state vector at time t ,
- $\mathcal{F} : \mathcal{S} \rightarrow \mathcal{S}$ is a nonlinear function determining the evolution of the system given the current state $\mathbf{s}(t)$

Goal

Identify a meaningful representation of the hidden states and a surrogate model for \mathcal{F} , **exploiting all data sources, of any fidelity.**

Framework

Consider L sets of N_l d_l -dimensional measurements of such system

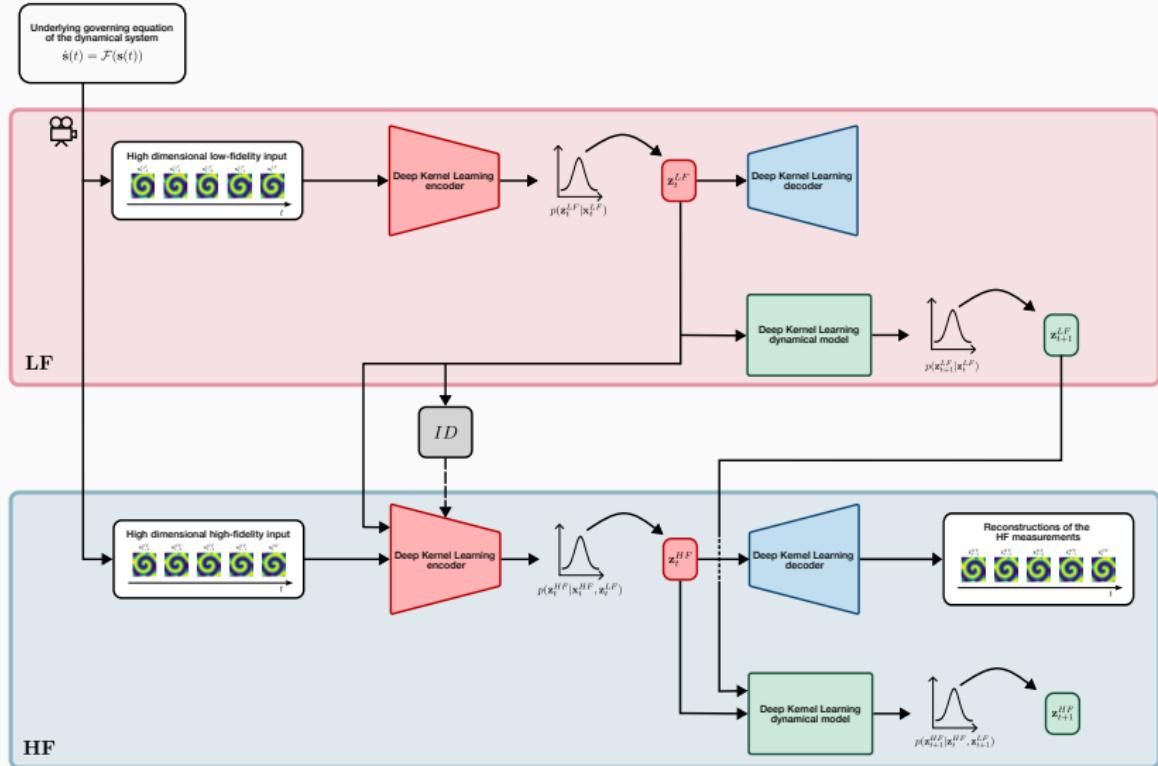
$\mathbf{X}_l = (\mathbf{x}_l^1, \dots, \mathbf{x}_l^{N_l}) \in \mathcal{X}_l$, $\mathbf{x}_l^i \in \mathbb{R}^{d_l}$, with $d_l \gg 1$ and l being the fidelity level,
 $0 \leq l \leq L$.

We train a sub-model Z_l on the set of measurements \mathbf{X}_l and leverage the knowledge on the system acquired by the previous less accurate models Z_i , $i < l$.

Each sub-model is composed of:

- a **SVDKL Autoencoder**;
- a **SVDKL dynamical model**.

Scheme of the model



Learning the hidden state

We introduce a **Stochastic Variational DKL encoder** $E_I : \mathcal{X}_I \rightarrow \mathcal{L}_I$.

A latent state sample can be obtained as:

$$z_{i,I}^{(t)} = Z_i^{E_I}(\mathbf{x}_I^{(t)}) + \varepsilon_{E_I}, \quad \varepsilon_{E_I} \sim \mathcal{N}(0, \sigma_{E_I}^2)$$

$$Z_{I,i}^{E_I}(\mathbf{x}_I^{(t)}) \sim \mathcal{GP}(\mu(g_{E_I}(\mathbf{x}_I^{(t)}; \mathbf{w}_{E_I})), k(g_{E_I}(\mathbf{x}_I^{(t)}; \mathbf{w}_{E_I}), g_{E_I}(\mathbf{x}_I^{(t')}; \mathbf{w}_{E_I}); \boldsymbol{\theta}_{E_I, i})) ,$$
$$1 \leq i \leq |\mathbf{z}_I|$$

The parameters $(\mathbf{w}_{E_I}, \boldsymbol{\theta}_{E_I}, \sigma_{E_I}^2)$ are optimised with a **decoder** $D_I : \mathcal{L}_I \times \mathcal{L}_{I-1} \rightarrow \mathcal{X}_I$ that reconstructs the measurements given

$$\tilde{\mathbf{z}}_I = \mathbf{z}_I + \rho \mathbf{z}_{I-1} .$$

Learning the latent dynamics

We introduce a **SVDKL surrogate model** $F_l : \mathcal{L}_l \times \mathcal{L}_{l-1} \rightarrow \mathcal{L}_l$.

The next latent states, at fidelity level l , $\mathbf{z}_l^{(t+1)}$ can be sampled with F_l :

$$\mathbf{z}_{i,l}^{(t+1)} = Z_i^{F_l}(\mathbf{z}_l^{(t)}, \mathbf{z}_{l-1}^{(t+1)}) + \varepsilon_{F_l}, \quad \varepsilon_{F_l} \sim \mathcal{N}(0, \sigma_{F_l}^2)$$

$$Z_i^{F_l}(\mathbf{z}_l^{(t)}, \mathbf{z}_{l-1}^{(t+1)}) \sim$$

$$\mathcal{GP}(\mu(g_{F_l}(\mathbf{z}_l^{(t)}, \mathbf{z}_{l-1}^{(t+1)}; \mathbf{w}_{F_l})), k(g_{F_l}(\mathbf{z}_l^{(t)}, \mathbf{z}_{l-1}^{(t+1)}; \mathbf{w}_{F_l}), g_{F_l}(\mathbf{z}_l^{(t')}, \mathbf{z}_{l-1}^{(t'+1)}; \mathbf{w}_{F_l}); \theta_{F_l, i})) , \\ 1 \leq i \leq |\mathbf{z}_l|$$

where $g_{F_l}(\mathbf{z}_l^{(t)}, \mathbf{z}_{l-1}^{(t+1)}; \mathbf{w}^{F_l})$ is the feature vector output of the NN of F_l .

Training the model

The two components E_I and F_I are jointly trained.

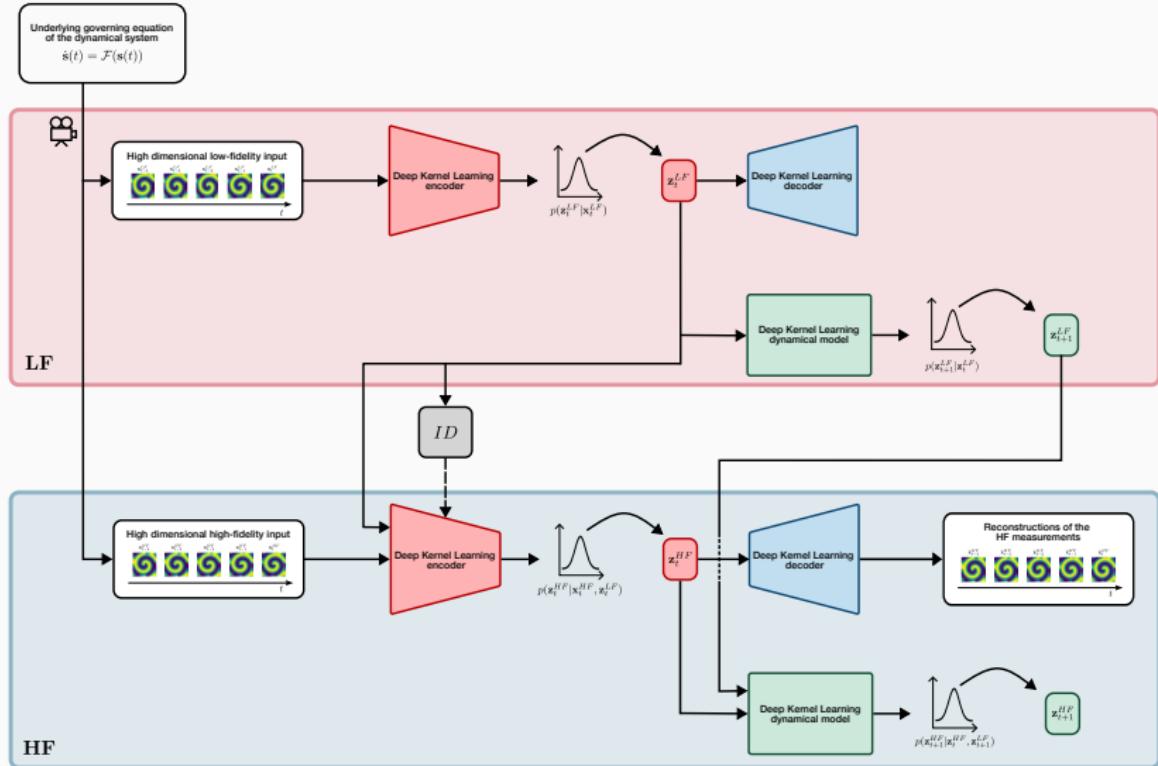
- E_I is trained by minimising the error between the reconstruction $\hat{\mathbf{x}}_I^{(t)}$ and the measurement $\mathbf{x}_I^{(t)}$;
- F_I is trained by minimising the **Kullback-Leibler divergence** between the distributions $p(\mathbf{z}_I^{(t+1)}|\mathbf{x}_I^{(t+1)})$ and $p(\mathbf{z}_I^{(t+1)}|\mathbf{z}_I^{(t)}, \mathbf{z}_{I-1}^{(t+1)})$.

The loss function is given by:

$$\begin{aligned}\text{loss}_I(\mathbf{w}_{E_I}, \theta_{E_I}, \sigma_{E_I}^2, \theta_{D_I}, \mathbf{w}_{F_I}, \theta_{F_I}, \sigma_{F_I}^2) = \\ \mathbb{E}_{\mathbf{x}_I^{(t)}, \mathbf{x}_I^{(t+1)} \sim \mathbf{x}_I} [-\log p(\hat{\mathbf{x}}_I^{(t)}|\mathbf{z}_I^{(t)}, \mathbf{z}_{I-1}^{(t)}) + \beta \text{KL}[p(\mathbf{z}_I^{(t+1)}|\mathbf{x}_I^{(t+1)}) || p(\mathbf{z}_I^{(t+1)}|\mathbf{z}_I^{(t)}, \mathbf{z}_{I-1}^{(t+1)})] + \\ -\log p(\hat{\mathbf{x}}_I^{(t+1)}|\mathbf{z}_I^{(t+1)}, \mathbf{z}_{I-1}^{(t+1)})]\end{aligned}$$

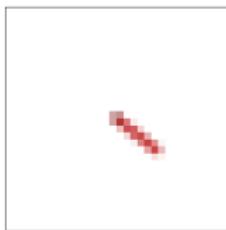
in which β is used to scale the contribution of the two loss terms.

Scheme of the model

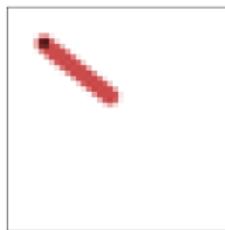


Numerical results

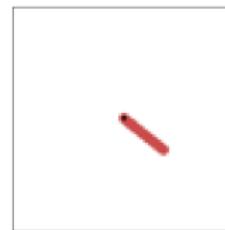
Pendulum: input data



(a) Fidelity level 0



(b) Fidelity level 1



(c) Fidelity level 2

Figure 2: Pendulum case.

Pendulum: ID and latent variables

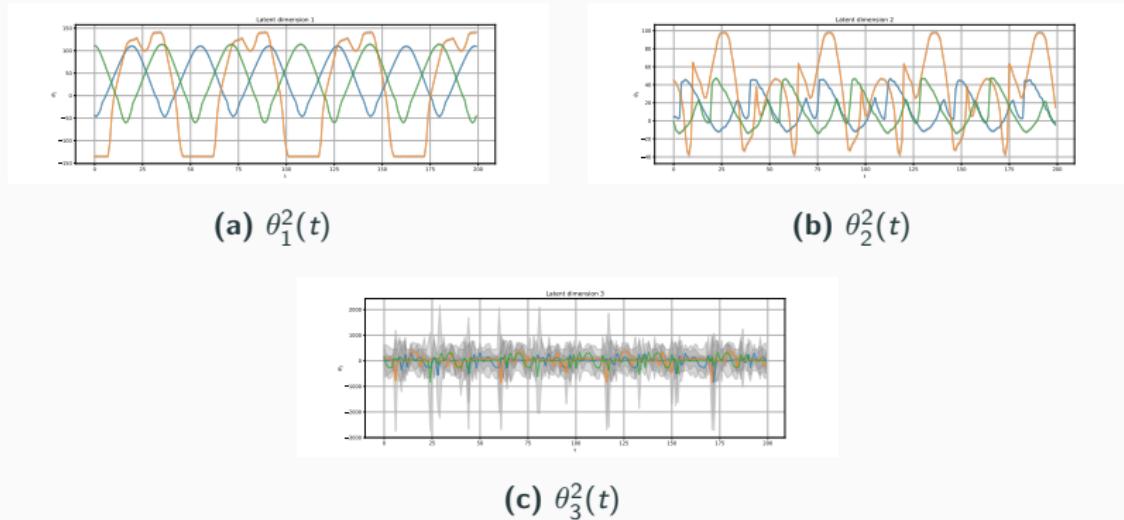


Figure 3: Pendulum case.

Pendulum: reconstruction and one-step forward prediction

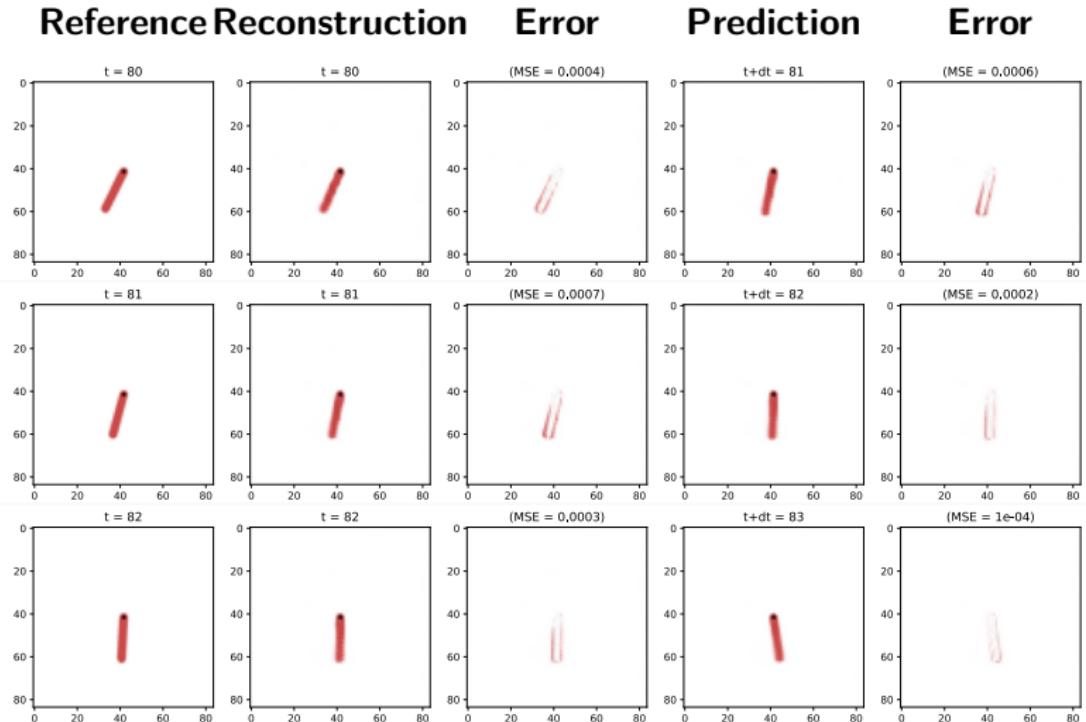


Figure 4: Pendulum case.

Pendulum: extrapolation in time

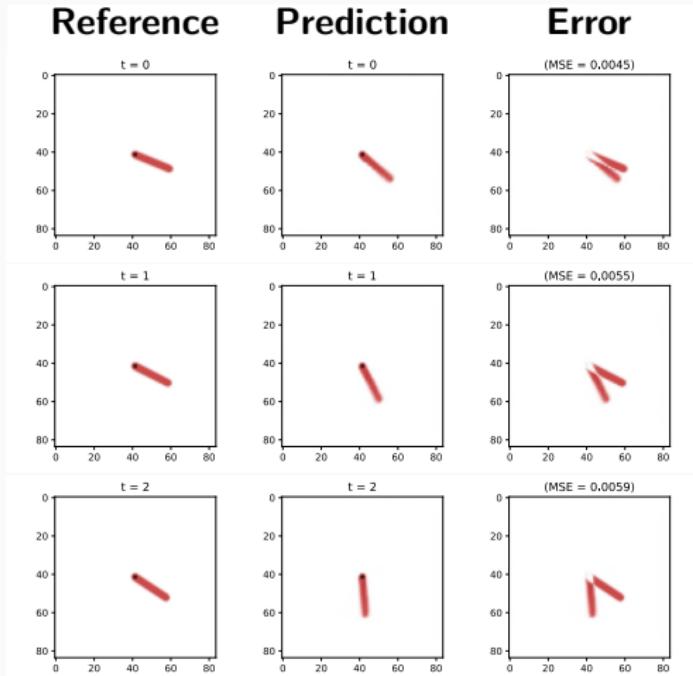
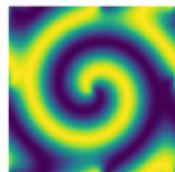
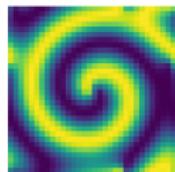


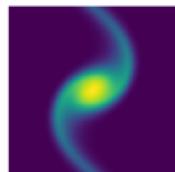
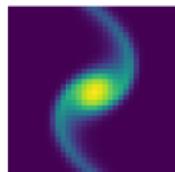
Figure 5: Pendulum case.

PDE: input data



(a) Low fidelity sample (b) High fidelity sample

Figure 6: Samples from the reaction-diffusion dataset.



(a) Low fidelity sample (b) High fidelity sample

Figure 7: Samples from the diffusion-advection dataset.

PDE: ID and latent variables

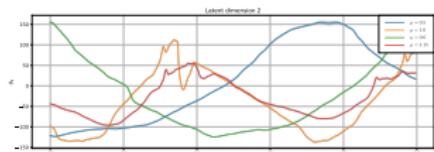
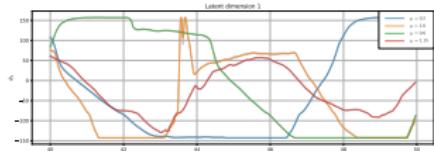


Figure 8: Reaction-diffusion case.

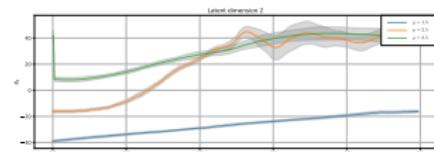
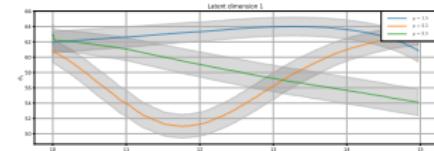


Figure 9: Diffusion-advection case.

PDE: reconstruction and one-step forward prediction

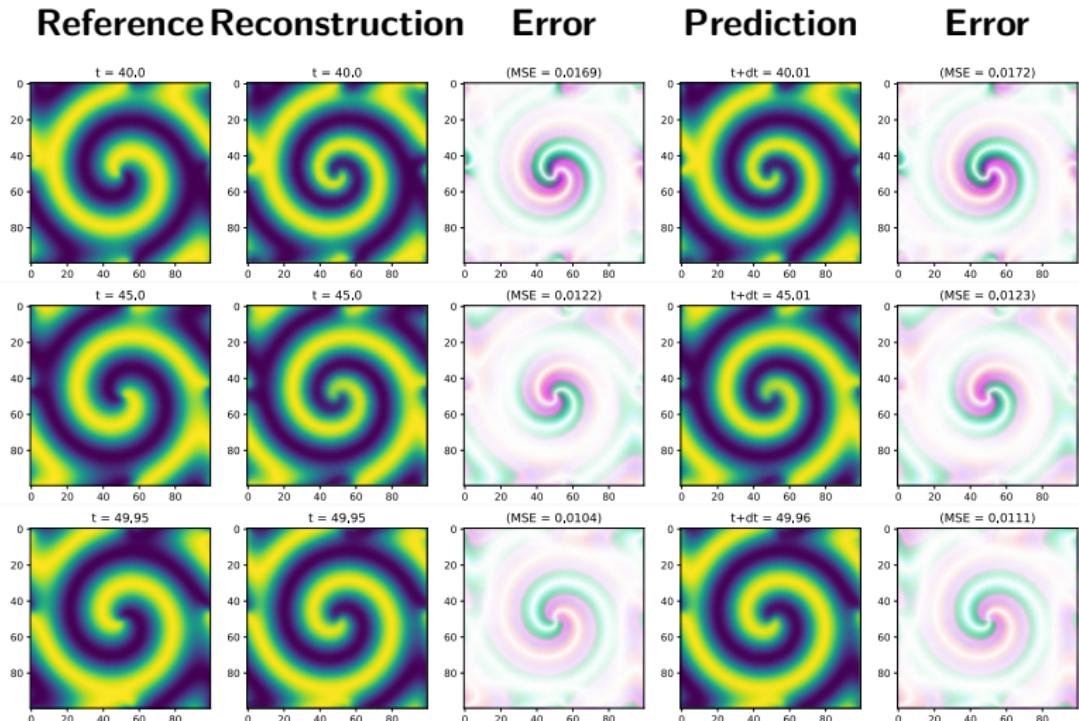


Figure 10: Reaction diffusion case.

PDE: reconstruction and one-step forward prediction

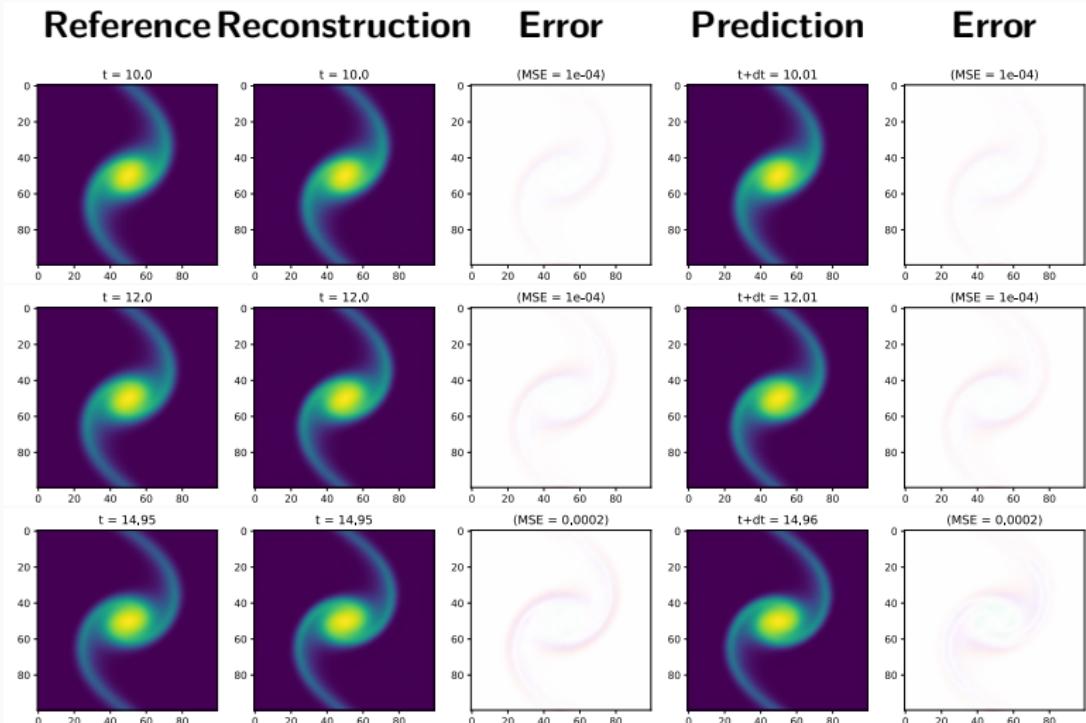


Figure 11: Diffusion-advection case.

PDE: extrapolation in time

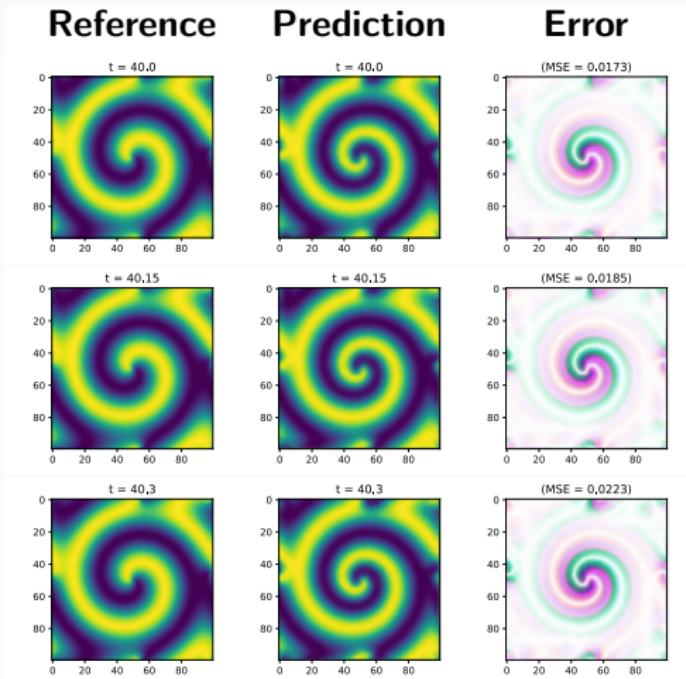


Figure 12: Reaction-diffusion case.

PDE: extrapolation in time

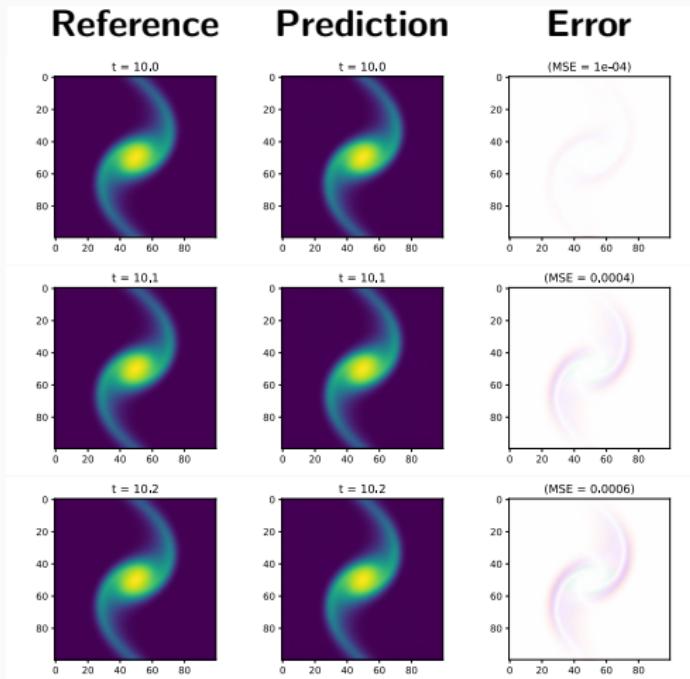


Figure 13: Diffusion-advection case.

Conclusions

Summary

Conclusions

- ✓ The model **generalises** the work from Botteghi et al. **to the MF context**, considering new test cases.
- ✓ The framework allows accurate data-driven unsupervised learning **from ordinary RGB video**.
- ✓ The model proves **effective at learning unknown system dynamics**, producing an efficient latent representation.

Further developments

- Leverage both low-dimensional and high-dimensional measurements.
- Consider different time steps of discretisation among fidelity levels.
- Fully identify the dynamical system state space.

Thank you for the attention!

Questions?

Appendix: Preliminaries

Gaussian process

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Let $\mathbf{x} \in \mathbb{R}^n$, we define the mean function $m(\mathbf{x})$ and the kernel function $k(\mathbf{x}, \mathbf{x}')$ of a real process $Z(\mathbf{x})$ as

$$\begin{aligned} m(\mathbf{x}) &= E[Z(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(Z(\mathbf{x}) - m(\mathbf{x}))(Z(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned} \tag{1}$$

We write the Gaussian process as

$$Z(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) . \tag{2}$$

Kernel function

The kernel function can be parametrised as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_k^2 r(\mathbf{x}, \mathbf{x}'; \theta) \quad (3)$$

where σ^2 and θ have to be estimated.

To build up a GP model, the class of the correlation kernel $r(\mathbf{x}, \mathbf{x}'; \theta)$ needs to be set. A popular choice is the automatic relevance determination (ARD) squared exponential (SE) kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \exp\left(-\frac{1}{2} \sum_{j=1}^d \frac{(x_j - x'_j)^2}{l_j^2}\right) \quad (4)$$

where $\{l_j\}_{(1 \leq j \leq d)}$ are the length-scales along each individual input direction.

Predictive distribution

Assuming additive Gaussian noise $G(\mathbf{x})|Z(\mathbf{x}) \sim \mathcal{N}(0, \sigma_\epsilon^2)$, the predictive distribution of the GP evaluated at N^* test data points \mathcal{X}^* is

$$[Z(\mathcal{X}^*)|Z(\mathcal{X}) = \mathcal{Y}, \sigma^2, \boldsymbol{\theta}] \sim \mathcal{N}(\mathbb{E}[Z(\mathcal{X}^*)], C(Z(\mathcal{X}^*))) \quad (5)$$

where

$$\begin{aligned}\mathbb{E}[Z(\mathcal{X}^*)] &= \boldsymbol{\mu}_{\mathcal{X}^*} + K_{\mathcal{X}^*, \mathcal{X}}(K_{\mathcal{X}, \mathcal{X}} + \sigma I)^{-1}\mathcal{Y} \\ C(Z(\mathcal{X}^*)) &= K_{\mathcal{X}^*, \mathcal{X}^*} + K_{\mathcal{X}^*, \mathcal{X}}(K_{\mathcal{X}, \mathcal{X}} + \sigma I)^{-1}K_{\mathcal{X}, \mathcal{X}}^*\end{aligned} \quad (6)$$

$K_{\mathcal{X}^*, \mathcal{X}}$ is an $N^* \times N$ matrix of covariances between the GP evaluated at \mathcal{X}^* and \mathcal{X} . $\boldsymbol{\mu}_{\mathcal{X}^*}$ is the $N^* \times 1$ mean vector, and $K_{\mathcal{X}, \mathcal{X}}$ is the $N \times N$ covariance matrix evaluated at training inputs \mathcal{X} . All covariance matrices implicitly depend on the kernel hyperparameters $\boldsymbol{\theta}$.

Parameter optimisation

By maximising the marginal likelihood given the training targets \mathcal{Y} , the optimal values of $(\theta, \sigma_\epsilon)$ can be estimated:

$$\begin{aligned}(\hat{\theta}, \hat{\sigma}_\epsilon) &= \arg \max_{\theta, \sigma_\epsilon} \log p(\mathcal{Y} | \mathcal{X}; \theta) = \\&= \arg \max_{\theta, \sigma_\epsilon} \left\{ -\mathcal{Y}^\top (K_{\mathcal{X}, \mathcal{X} | \theta} + \sigma_\epsilon^2 I)^{-1} \mathcal{Y} - \log |K_{\mathcal{X}, \mathcal{X} | \theta} + \sigma_\epsilon^2 I| \right\}\end{aligned}\tag{7}$$

Levina-Bickel algorithm 1

The Levina-Bickel algorithm for ID estimation provides a maximum likelihood estimator (MLE) of the dimension m from i.i.d. observations $\mathbf{z}_1, \dots, \mathbf{z}_n$ in \mathbb{R}^p .

The observations $\mathbf{z}_i = g(\mathbf{s}_i)$, where \mathbf{s}_i are sampled from an unknown smooth density f on \mathbb{R}^{ID} , with unknown $ID \leq p$, and g is a continuous and sufficiently smooth mapping.

The key idea is to fix a point \mathbf{z} , assume $f(\mathbf{z}) \approx \text{const}$ in a small sphere $S_{\mathbf{z}}(R)$ of radius R around \mathbf{z} , and treat the observations as a homogeneous Poisson process in $S_{\mathbf{z}}(R)$.

Levina-Bickel algorithm 2

The Levina–Bickel algorithm derives the local ID estimator near \mathbf{z} as

$$ID_{L-B}(\mathbf{z}) = \frac{1}{k-2} \sum_{j=1}^{k-1} \log \frac{T_k(\mathbf{z})}{T_j(\mathbf{z})} , \quad (8)$$

where we are fixing the number of neighbors k rather than the radius of the sphere R , and $T_k(\mathbf{z})$ is the Euclidean distance between \mathbf{z} and its k -th nearest neighbor in $\mathbf{z}_1, \dots, \mathbf{z}_n$.

By dividing by $k - 1$, rather than $k - 2$, we obtain an asymptotically unbiased estimator.

Appendix: Methods

Variational inference

Since the two SVDKL components exploit variational inference to approximate the respective posterior distributions, we add two extra terms to the loss function, of the form

$$\text{loss}_{\text{var}}(\mathbf{w}, \theta) = \text{KL}[p(\mathbf{v}) || q(\mathbf{v})], \quad (9)$$

where $p(\mathbf{v})$ is the posterior to be approximated over the inducing points \mathbf{v} , and $q(\mathbf{v})$ represents an approximating candidate distribution. The inducing points \mathbf{v} are placed on a grid, following the intuition in the original SVDKL work.

Kullback–Leibler divergence

Definition

The Kullback–Leibler (KL) divergence is a measure of how one probability distribution P diverges from a second, reference probability distribution Q .

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} \quad (10)$$

Properties:

- KL divergence is non-negative: $D_{\text{KL}}(P \parallel Q) \geq 0$.
- $D_{\text{KL}}(P \parallel Q) = 0$ if and only if $P = Q$ almost everywhere.
- It is not symmetric: $D_{\text{KL}}(P \parallel Q) \neq D_{\text{KL}}(Q \parallel P)$.
- Often interpreted as the "information loss" when Q is used to approximate P .

Algorithm i

```
1: for level = LF, HF do
2:   if level == LF then
3:     Initialise  $\mathbf{z}_{LF}, \mathbf{z}_{LF}^{next}, \mathbf{z}_{LF}^{fwd} = \mathbf{0}$ ;
4:     Set latentDimension to an arbitrary value;
5:   else
6:     Initialise  $\mathbf{z}_{LF} = \mathbf{z}, \mathbf{z}_{LF}^{next} = \mathbf{z}^{next}, \mathbf{z}_{LF}^{fwd} = \mathbf{z}^{fwd}$ ;
7:     Set latentDimension = ID;
8:   end if
9:
10:  for epoch = 1, 2, ... do
11:    Encode  $\mathbf{x} \rightarrow \mathbf{z}, \mathbf{x}^{next} \rightarrow \mathbf{z}^{next}$ ;
12:    Decode  $\hat{\mathbf{x}} \leftarrow \mathbf{z} + \rho \mathbf{z}_{LF}, \hat{\mathbf{x}}^{next} \leftarrow \mathbf{z}^{next} + \rho \mathbf{z}_{LF}^{next}$ ;
13:    Estimate  $\mathbf{z} + \rho \mathbf{z}_{LF}^{fwd} \rightarrow \mathbf{z}^{fwd}$ ;
14:    Decode  $\hat{\mathbf{x}}^{fwd} \leftarrow \mathbf{z}^{fwd}$ ;
15:    Compute the  $loss(\mathbf{x}, \hat{\mathbf{x}}, \hat{\mathbf{x}}^{next}, \hat{\mathbf{x}}^{fwd}, \mathbf{z}^{next}, \mathbf{z}^{fwd})$ ;
```

Algorithm ii

```
16:     Backward propagate the gradients;  
17: end for  
18:  
19: if level == LF then  
20:     Compute ID =  $\lceil \frac{1}{3}LB(\mathbf{z}) + \frac{1}{3}LB(\mathbf{z}^{next}) + \frac{1}{3}LB(\mathbf{z}^{fwd}) \rceil$ ;  
21: end if  
22: end for
```

Appendix: Results

Pendulum

We consider the simple pendulum, described by the following equation:

$$\ddot{\phi}(t) = -\frac{g}{l} \sin \phi(t), \quad (11)$$

where ϕ is the angle of the pendulum, $\ddot{\phi}$ is the angular acceleration, l is the length, g is the gravitational acceleration. The starting state is a random angle and a random angular velocity, in particular

$$\phi(0) \sim \mathcal{U}([-\pi, \pi]), \quad (12a)$$

$$\dot{\phi}(0) \sim \mathcal{U}([-1, 1]). \quad (12b)$$

Our goal is to approximate the time-dependent state of the pendulum $x_1 = \phi, x_2 = \dot{\phi}$, considering a test set of new episodes, unseen during training.

Pendulum: dataset structure

We adopt 3 fidelity levels that differ in the resolution of the frame:

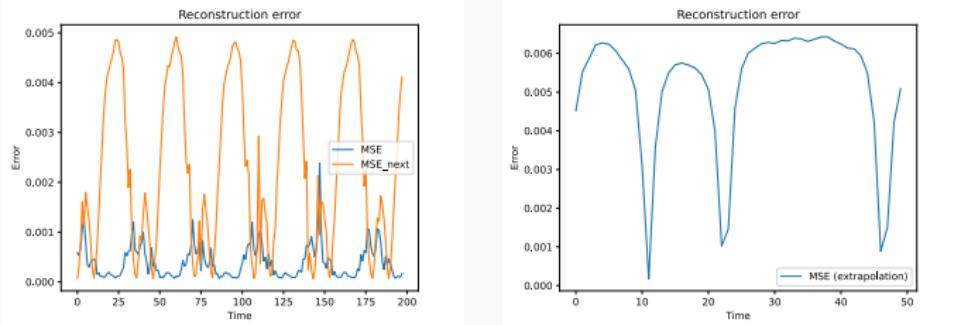
- **level 0**, (full) RGB images of size $l_0 \times l_0 \times 3$;
- **level 1**, cropped RGB images of size $l_1 \times l_1 \times 3$;
- **level 2**, full RGB images of high resolution size $l_2 \times l_2 \times 3$, $l_2 > l_0$.

The initial conditions are randomly initialised for each episode.

Parameters	Level 0	Level 1	Level 2
N	150	80	50
l	32	42	84
$portion$	1	0.6	1
N_{test}	3	3	3

Table 1: List of configuration parameters for the pendulum dataset.

Pendulum: errors



(a) MSEs of the reconstruction and (b) MSE of the extrapolation in forward prediction

Figure 14: MSEs of the pendulum case.

Pendulum: MF vs. SF

Episode	Reconstruction $\overline{\text{MSE}}(t)$		Forward prediction $\overline{\text{MSE}}(t)$	
	MF	SF	MF	SF
Episode 1	$4.2048 \cdot 10^{-4}$	$9.9681 \cdot 10^{-4}$	$2.2964 \cdot 10^{-3}$	$3.2726 \cdot 10^{-3}$
Episode 2	$8.9107 \cdot 10^{-4}$	$1.0004 \cdot 10^{-3}$	$4.3134 \cdot 10^{-3}$	$4.8076 \cdot 10^{-3}$
Episode 3	$3.9761 \cdot 10^{-4}$	$9.8603 \cdot 10^{-4}$	$2.4977 \cdot 10^{-3}$	$3.3353 \cdot 10^{-3}$

Table 2: Comparison of the mean in time of the MSEs, in the pendulum case, for both reconstruction and forward prediction, between the MF model and a SF model that considers only LF data.

Reaction-diffusion

We consider a lambda-omega reaction-diffusion system defined by the following equations

$$\dot{u} = (1 - (u^2 + v^2)) u + \mu (u^2 + v^2) v + d(u_{xx} + u_{yy}), \quad (13a)$$

$$\dot{v} = -\mu (u^2 + v^2) u + (1 - (u^2 + v^2)) v + d(v_{xx} + v_{yy}) \quad (13b)$$

defined over a spatial domain $(x, y) \in [-L, L]^2$ and a time span $t \in [0, T]$, where μ and d are the reaction and diffusion parameters, respectively.

The initial condition is defined as

$$u(x, y, 0) = v(x, y, 0) = \tanh \left(\sqrt{x^2 + y^2} \cos \left((x + iy) - \sqrt{x^2 + y^2} \right) \right). \quad (14)$$

We are interested in approximating the solution components u and v as functions of the varying reaction parameter $\mu \in \mathcal{P} = [0.5, 1.5]$, with a fixed diffusion coefficient d .

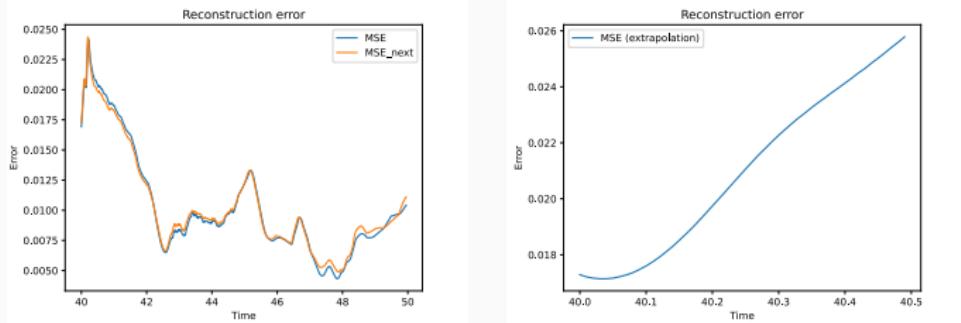
Reaction-diffusion: dataset structure

LF data are associated with a coarser grid of discretisation ($n^{LF} < n^{HF}$), but they are observed for a longer time window, $T_{train}^{LF} > T_{train}^{HF}$.

The model is tested on new measurements for $t \in [T_{train}^{HF}, T_{train}^{HF} + T_{test}]$, extrapolating over time and \mathcal{P} .

Parameters	LF	HF
n	32	100
d	0.05	0.05
μ_{train}	{0.5, 0.75, 1, 0.6, 1.15, 1.375}	{0.5, 0.75, 1}
T_{train}	80	40
L		10
dt		0.01
μ_{test}		{0.5, 1, 0.6, 1.15}
T_{test}		10

Reaction-diffusion: errors



(a) MSEs of the reconstruction and (b) MSE of the extrapolation in forward prediction

Figure 15: MSEs of the reaction-diffusion case.

Diffusion-advection

We consider a diffusion-advection problem describing a fluid motion in the shallow water limit given by

$$\frac{\partial \omega}{\partial t} + \mu \left(\frac{\partial \psi}{\partial x} \frac{\partial \omega}{\partial y} - \frac{\partial \psi}{\partial y} \frac{\partial \omega}{\partial x} \right) = d \nabla^2 \omega, \quad (15a)$$

$$\nabla^2 \psi = \omega \quad (15b)$$

defined over a spatial domain $(x, y) \in [-L, L]^2$ and a time span $t \in [0, T]$. Here $\omega(x, y, t)$ and $\psi(x, y, t)$ represent the vorticity and stream function, respectively, $\nabla^2 = \partial x^2 + \partial y^2$ is the two-dimensional Laplacian, and d is the diffusion coefficient. We consider the following initial condition of vorticity:

$$\omega(x, y, 0) = \exp \left(-2x^2 - \frac{y^2}{20} \right), \quad (x, y) \in [-L, L]^2. \quad (16)$$

Our goal is to approximate the time-dependent vorticity field ω as the parameter μ varies over $\mathcal{P} = [1, 5]$.

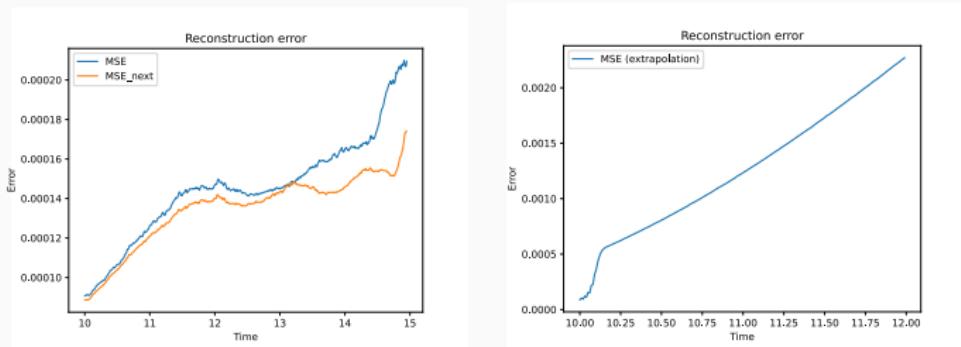
Diffusion-advection: dataset structure

LF data are associated with a coarser grid of discretisation ($n^{LF} < n^{HF}$), but they are observed for a longer time window, $T_{train}^{LF} > T_{train}^{HF}$.

The model is tested on new measurements for $t \in [T_{train}^{HF}, T_{train}^{HF} + T_{test}]$, extrapolating over time and \mathcal{P} .

Parameters	LF	HF
n	32	100
d	0.001	0.001
μ_{train}	{1.5, 2.5, 3.5, 4.5}	{1.5, 2.5}
T_{train}	30	10
L	5	
dt	0.01	
μ_{test}	{1.5, 2.5, 3.5}	
T_{test}	5	

Diffusion-advection: errors



(a) MSEs of the reconstruction and forward prediction

(b) MSE of the extrapolation in time

Figure 16: MSEs of the diffusion-advection case.

Reaction-diffusion: MF vs. SF

μ value	Reconstruction $\overline{\text{MSE}}(t)$		Forward prediction $\overline{\text{MSE}}(t)$	
	MF	SF	MF	SF
$\mu = 0.5$	$1.0436 \cdot 10^{-2}$	$1.8990 \cdot 10^{-2}$	$1.0396 \cdot 10^{-2}$	$1.8747 \cdot 10^{-2}$
$\mu = 1.0$	$2.1082 \cdot 10^{-2}$	$3.2734 \cdot 10^{-2}$	$2.1275 \cdot 10^{-2}$	$3.2628 \cdot 10^{-2}$
$\mu = 0.6$	$0.9110 \cdot 10^{-2}$	$1.9871 \cdot 10^{-2}$	$0.8927 \cdot 10^{-2}$	$1.9587 \cdot 10^{-2}$
$\mu = 1.15$	$5.6527 \cdot 10^{-2}$	$5.3440 \cdot 10^{-2}$	$5.6545 \cdot 10^{-2}$	$5.3560 \cdot 10^{-2}$

Table 5: Comparison of the mean in time of the MSEs, in the reaction-diffusion case, for both reconstruction and forward prediction, between the MF model and a SF model that considers only LF data.

References i

-  Nicolò Botteghi, Mengwu Guo, and Christoph Brune.
Deep Kernel Learning of Dynamical Models from High-Dimensional Noisy Data.
Scientific Reports, 12(1):21530, December 2022.
arXiv:2208.12975 [cs, stat].
-  Boyuan Chen, Kuang Huang, Sunand Raghupathi, Ishaan Chandratreya, Qiang Du, and Hod Lipson.
Discovering State Variables Hidden in Experimental Data, December 2021.
arXiv:2112.10755 [physics].

References ii

-  Paolo Conti, Mengwu Guo, Andrea Manzoni, Attilio Frangi, Steven L. Brunton, and J. Nathan Kutz.
Multi-fidelity reduced-order surrogate modeling, September 2023.
arXiv:2309.00325 [cs, math].
-  Mengwu Guo, Andrea Manzoni, Maurice Amendt, Paolo Conti, and Jan S. Hesthaven.
Multi-fidelity regression using artificial neural networks: efficient approximation of parameter-dependent output quantities.
Computer Methods in Applied Mechanics and Engineering, 389:114378, February 2022.
arXiv:2102.13403 [cs, math].

References iii

-  Elizaveta Levina and Peter Bickel.
Maximum Likelihood Estimation of Intrinsic Dimension.
In *Advances in Neural Information Processing Systems*, volume 17.
MIT Press, 2004.
-  GPyTorch Developers.
GPyTorch Documentation, 2024.
Accessed: 2024-06-19.
-  PyTorch Developers.
PyTorch Documentation, 2024.
Accessed: 2024-06-19.
-  Gymnasium Developers.
Gymnasium Documentation, 2024.
Accessed: 2024-06-24.