A study on quantifying effective training of DLDMD

Joseph A.G. Diaz

Master of Science in Applied Mathematics
with a Concentration in Dynamical Systems,
San Diego State University

July 30, 2022

Introduction

In the study of dynamical systems a central problem is how to derive models from measured data to facilitate the prediction of future states. The data-driven method Dynamic Mode Decomposition and it's extensions offer a compelling avenue in the problem of prediction from time-series data.
The marriage of these methods with Machine Learning and Neural Networks allows for the leveraging the power of these tools in the space.
If standard metrics of model training are unavailable,

Introduction - Prediction from time-series

As it says on the tin, we seek a predictive model for a time series $\{\boldsymbol{y}_j\}_{j=1}^{N^T+1}$, which are the measurements of a uniformly sampled unknown dynamical system of the form

$$\frac{d\boldsymbol{y}}{dt} = f\left(\boldsymbol{y}(t)\right), \quad \boldsymbol{y}(0) = \boldsymbol{x} \in \mathcal{M} \subseteq \mathbb{R}^{N_s} \tag{1}$$

One method that can be leveraged for this is via the Koopman Operator $\mathcal{K}$. If we denote $\varphi(t; \boldsymbol{x}) = \boldsymbol{y}(t)$ to be the flow map affiliated with the initial condition $\boldsymbol{x}$ and denote $g : \mathcal{M} \to \mathbb{C}$ to be a square integrable observable of the system then, according to [3], there exists a linear representation of the flow map given by $\mathcal{K}$;

$$\mathcal{K}^t g(\boldsymbol{x}) = g(\varphi(t; \boldsymbol{x})), \tag{2}$$

Introduction - Infinite Dimensional Linear Operator $\mathcal{K}$

In order to actually use this new representation, it suffices to find the eigenvalues $\{\lambda_\ell\}$, and their affiliated eigenfunctions $\{\phi_\ell\}$, of the Koopman Operator such that

$$\mathcal{K}^t \phi_\ell = \exp(t\lambda_\ell)\phi_\ell \implies g(\boldsymbol{x}) = \sum_{\ell \in \mathbb{N}} a_\ell \phi_\ell(\boldsymbol{x}), \tag{3}$$

where we can essentially construct a modal decomposition of g. From here advancing the dynamics to time t is equivalent to writing

$$\mathcal{K}^t g(\boldsymbol{x}) = \sum_{\ell \in \mathbb{N}} a_\ell \exp(t\lambda_\ell)\phi_\ell(\boldsymbol{x}) \tag{4}$$

$$g(\boldsymbol{x}) = \sum_{\ell=1}^{N_O} a_\ell \psi_\ell(\boldsymbol{x}) \tag{5}$$

which is to say that the observable g exists in a N_O-dimensional subspace of $L^2(\mathcal{M})$ with basis functions $\{\psi_\ell\}_{\ell=1}^{N_O}$, and applying the Koopman operator implies that

$$\mathcal{K}^{\delta t} g(\boldsymbol{x}) = \sum_{\ell=1}^{N_O} a_\ell \exp(\delta t \lambda_\ell) \psi_\ell(\boldsymbol{x}) \tag{6}$$

$$= \sum_{\ell=1}^{N_O} \psi_\ell(\boldsymbol{x}) (\boldsymbol{K}_O^T \boldsymbol{a})_\ell + r(\boldsymbol{x}; \boldsymbol{K}_O) \tag{7}$$

Introduction - Time advancement

for discrete time step δt. $\boldsymbol{K}_O$ is the $N_O \times N_O$ matrix that

$$\boldsymbol{K}_O = \underset{K}{\operatorname{argmin}} ||\boldsymbol{\Psi}_+ - K\boldsymbol{\Psi}_-||_F^2 \tag{8}$$

and $r(\boldsymbol{x}, \boldsymbol{K}_O)$ is a residual that represents the total error due to DMD. If the ansatz that g lives in a finite dimensional space holds, then r is identically 0. We define $\boldsymbol{\Psi}_\pm$ to be

$$\boldsymbol{\Psi}_- = (\Psi_1\ \Psi_2\ \cdots \Psi_{N_T}), \quad \boldsymbol{\Psi}_+ = (\Psi_2\ \Psi_3\ \cdots \Psi_{N_T+1}) \tag{9}$$

where $\{\Psi_j\}$ is an observable of the time series of interest $\{\boldsymbol{y}_j\}$. What the expression for $\boldsymbol{K}_O$ tells us is that, we are trying to find a one-step mapping from each data point to the next. Practically speaking, $\boldsymbol{K}_O$ is found using a singular value decomposition (SVD), with which we can write

$$\boldsymbol{\Psi}_- = \mathbf{U}\boldsymbol{\Sigma}\mathbf{W}^\dagger \implies \boldsymbol{K}_O = \boldsymbol{\Psi}_+\mathbf{W}\boldsymbol{\Sigma}^{-P}\mathbf{U}^\dagger \tag{10}$$

Introduction - Flow reconstruction

Finding the eigenvalues, eigenfunctions, and Koopman modes comes down to an eigen-decomposition of $\boldsymbol{K}_O$,

$$\boldsymbol{K}_O = \mathbf{V}\mathbf{T}\mathbf{V}^{-1}, \tag{11}$$

with $\lambda_\ell = \ln((\mathbf{T})_{\ell\ell})/\delta t$, from which the dynamics can be approximated as

$$y(t;\boldsymbol{x}) \approx \boldsymbol{K}_m \exp(t\boldsymbol{\Lambda})\mathbf{V}^{-1}\boldsymbol{\Psi}(\boldsymbol{x}) \tag{12}$$

where $\boldsymbol{\Psi}$ is the representation of the initial condition in terms of the observables, $\boldsymbol{K}_m$ is the $N_S \times N_O$ matrix whose columns are the Koopman modes $\{\mathbf{k}_\ell\}_{\ell=1}^{N_O}$ which solve the initial value problem

$$\boldsymbol{x} = \sum_{\ell=1}^{N_O} \mathbf{k}_\ell \phi(\boldsymbol{x}), \tag{13}$$

and $\boldsymbol{\Lambda}$ is the diagonal matrix whose elements are $\lambda_\ell = (\boldsymbol{\Lambda})_{\ell\ell}$.

Introduction - Neural Networks

Before we move on, a brief digression on Neural Networks (NNs) is appropriate. Ever since scientists discovered the relatively simple interaction between neurons and axons in the human brain, they have been enamored with the ability to create a learning computer with the similar ability to become more adept at a task with training and practice. In the same way that art imitates life, the most straight-forward attempts have been to construct artificial neural networks that pantomime the biological ones that we carry around in our heads; and while these pale imitations have not been developed to rival the human brain, they have led to some major achievements in automation. In much the same way that a person is "trained" to perform a task by repeating it with feedback and adjusting their performance as they go, an artificial NN uses data, a loss function, and an optimization algorithm to change the state of the NN to one which can better accomplish the task. The loss function and optimizer are to the feedback and behavioral adjustment what the data is to the experience.

Introduction - Function approximation and regression

Say we have a set of training data $d_{tr} = \{(x_j, y_j)\}_{j=1}^{N_{tr}}$ and we believe that there is some function $f(x)$ such that

$$y_j = f(x_j) + \varepsilon_j, \tag{14}$$

where ε_j is assumed to be a sample drawn from $\mathcal{N}(0, \bar{\sigma}^2)$. [2, 4] Generally f is not known; so we must, using the training data, build an estimator $f_e(x; d_{tr})$ and measure the performance of the estimator by examining the quantity

$$\mathbb{E}_{d_{tr}}\left[\left(f(x) + \varepsilon - f_e(x; d_{tr})\right)^2\right] = \text{bias}^2_{d_{tr}} + \mathbb{V}(f_e) + \bar{\sigma}^2 \tag{15}$$

where

$$\text{bias}_{d_{tr}} = \mathbb{E}_{d_{tr}}\left[f_e\right] - f(x) \tag{16}$$

and

$$\mathbb{V}(f_e) = \mathbb{E}_{d_{tr}}\left[\left(\mathbb{E}_{d_{tr}}\left[f_e\right] - f_e(x; d_{tr})\right)^2\right] \tag{17}$$

For our purposes, we will instead consider how metrics from information theory can be used to assess the change in information as data is fed through a NN and a model is trained

Introduction - DLDMD

The key innovation of [1] is to use a neural network to come up with the collection of observables on $\{\boldsymbol{y}_j\}$ that allow for the best prediction of future system states, we call this method Deep Learning Enhanced DMD (DLDMD). This is implemented by defining an encoder $\mathcal{E} : \mathbb{R}^{N_S} \to \mathbb{R}^{N_O}$ and decoder $\mathcal{D} : \mathbb{R}^{N_O} \to \mathbb{R}^{N_S}$ composed of dense layers such that

$$(\mathcal{D} \circ \mathcal{E})(\boldsymbol{x}) = \boldsymbol{x} \tag{18}$$

We choose $N_O \geq N_S$ and an appropriate loss function so that $\mathcal{E}$ and $\mathcal{D}$ give a richer space of observables, called the latent space, for EDMD to use when advancing the dynamics. The implementation of NNs for this purpose requires a method of tuning to allow $\mathcal{E}$ and $\mathcal{D}$ to learn the best representations possible.

Introduction - The loss function

A natural choice considering these constraints is given by

$$\mathcal{L} = \alpha_1 \mathcal{L}_{\text{recon}} + \alpha_2 \mathcal{L}_{\text{dmd}} + \alpha_3 \mathcal{L}_{\text{pred}} + \alpha_4 \left|\left| \boldsymbol{W}_g \right|\right|_2 \tag{19}$$

where

$$\mathcal{L}_{\text{recon}} = \frac{1}{N_T + 1} \sum_{j=1}^{N_T+1} \left|\left| \boldsymbol{y}_j - (\mathcal{D} \circ \mathcal{E})(\boldsymbol{y}_j) \right|\right|_2, \tag{20}$$

$$\mathcal{L}_{\text{dmd}} = E_r(\boldsymbol{K}_O), \tag{21}$$

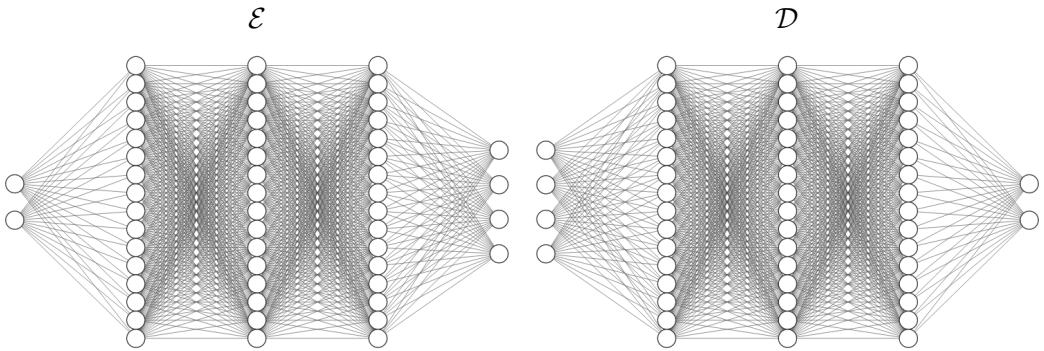
$$\mathcal{L}_{\text{pred}} = \frac{1}{N_T} \sum_{j=1}^{N_T} \left|\left| \boldsymbol{y}_{j+1} - \mathcal{D}\left(V T^j V^{-1} \mathcal{E}(\boldsymbol{x})\right) \right|\right|_2, \tag{22}$$

Recall that N_T is the number of measurements in our time-series data after the first.

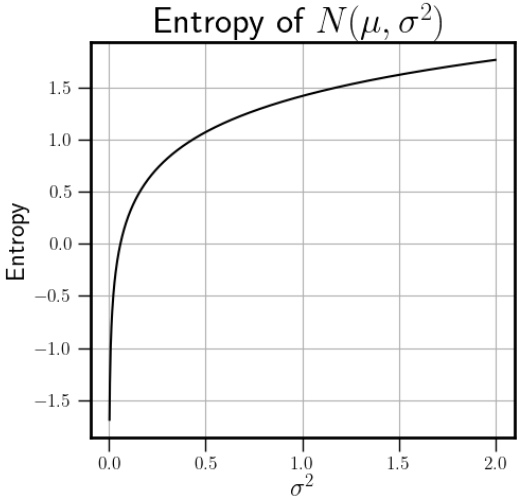
Introduction -

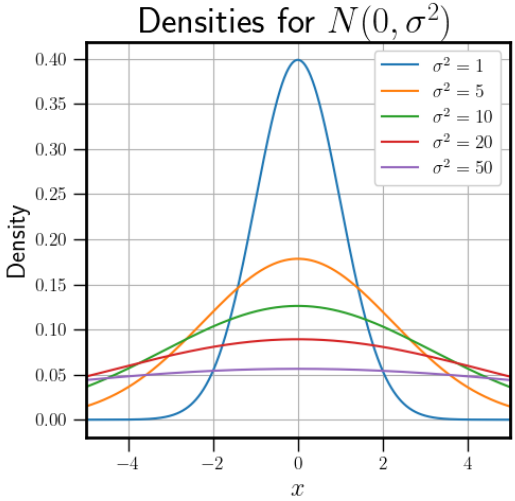
Introduction - The Network

Example of DLDMD newtork with $N_S = 2$, $N_O = 4$, and $N_L = 3$ where every hidden layer has 16 neurons. The layers in the Figure are labeled, sequentially, left to right: Enc in, Enc 0, Enc 1, Enc 2, Enc out, Dec in, Dec 0, Dec 1, Dec 2, Dec out.

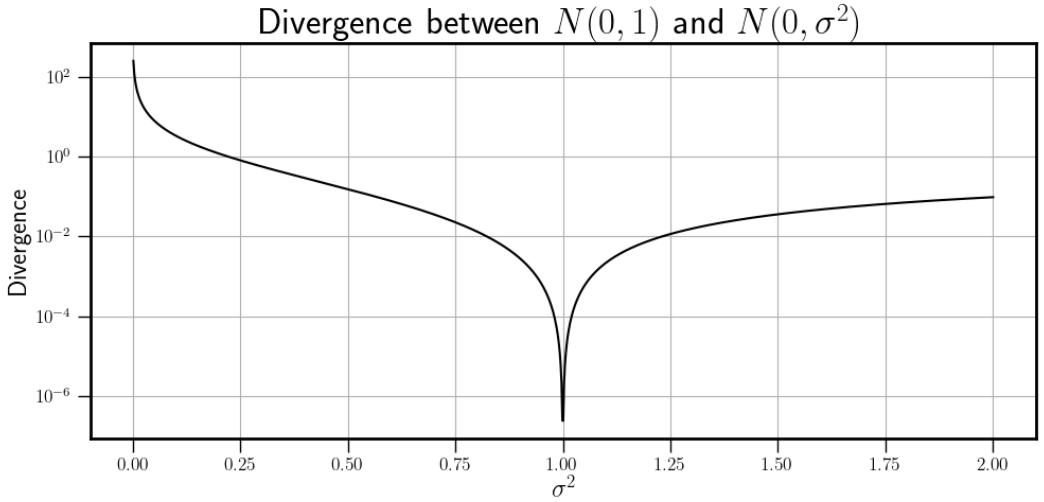


Kullback-Leibler Divergence - Entropy

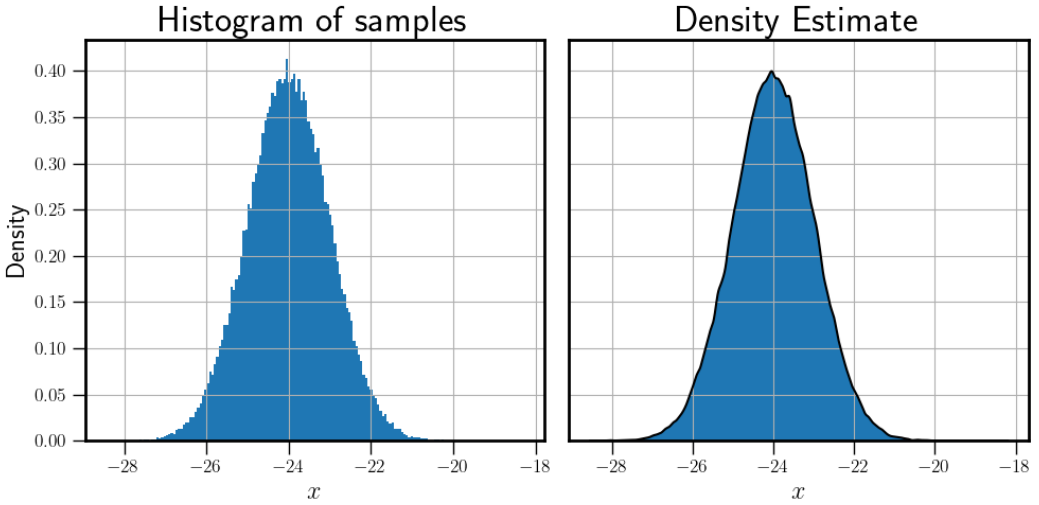




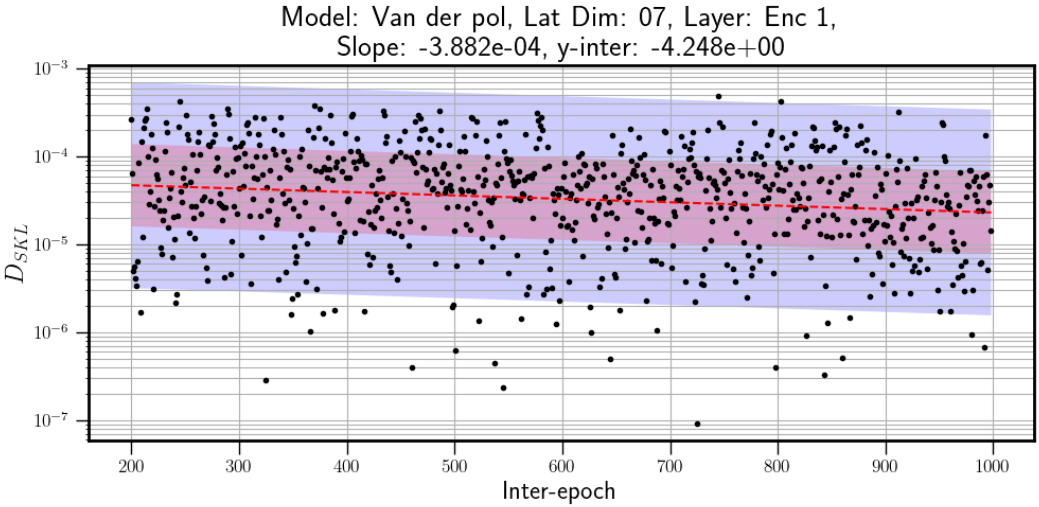
Kullback-Leibler Divergence - normal distribution example



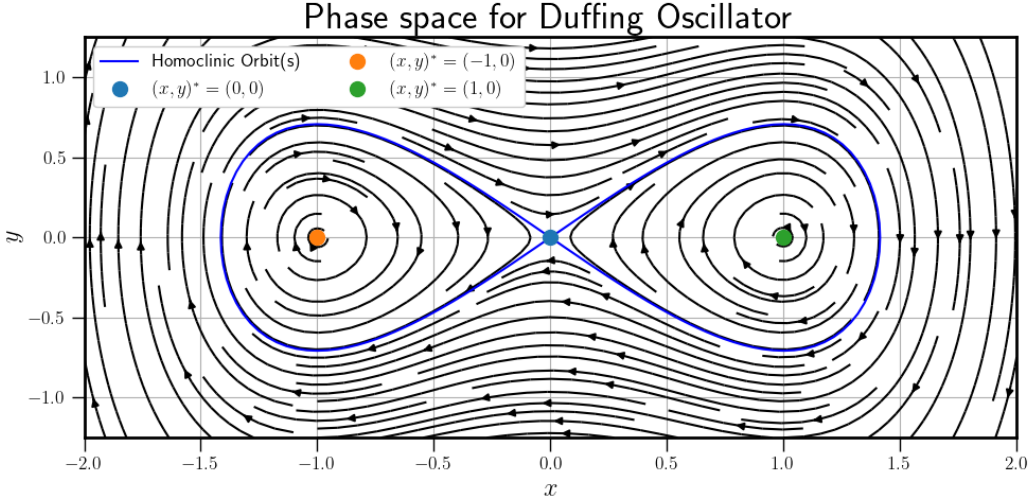
Kullback-Leibler Divergence - KDE example



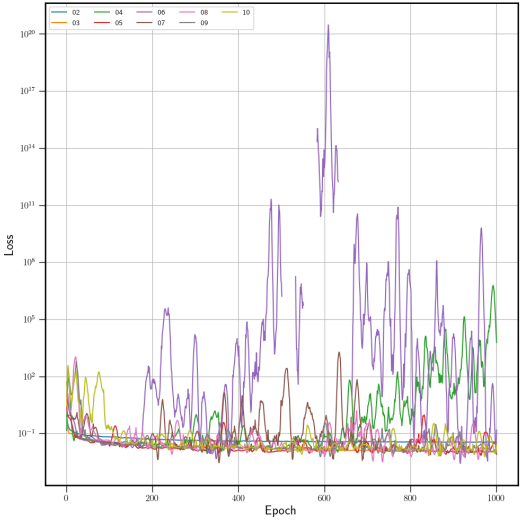
Kullback-Leibler Divergence - Example fitting

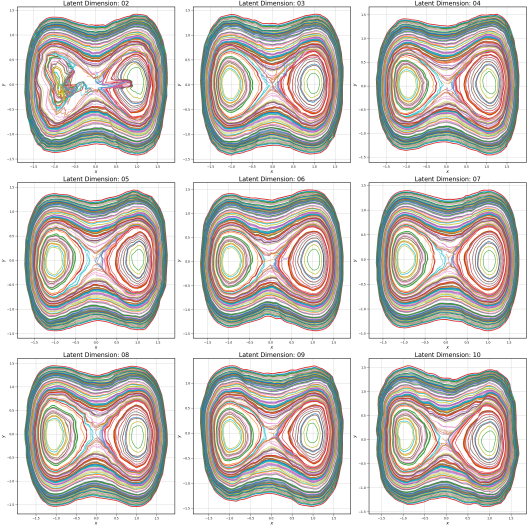


Results - Duffing

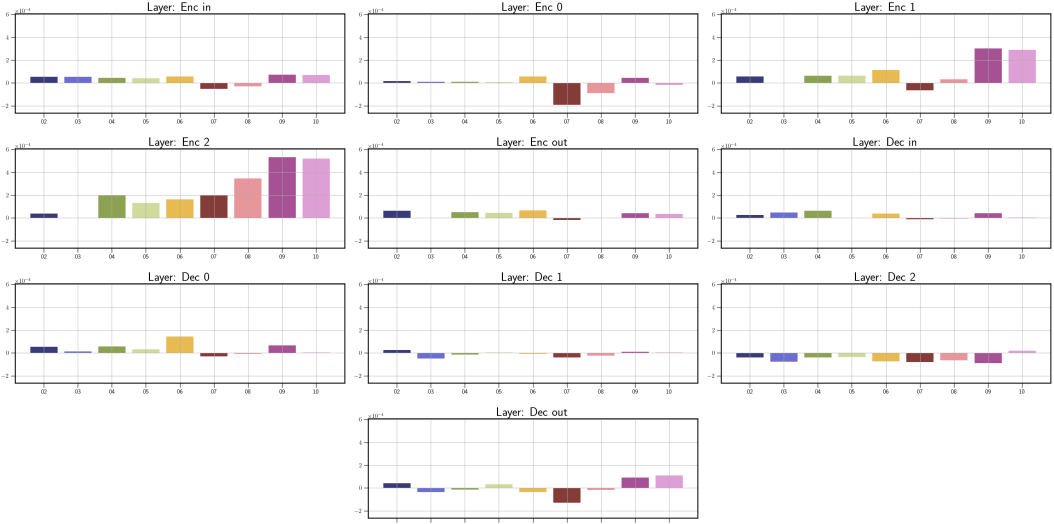


Results - Duffing loss curves and phase space

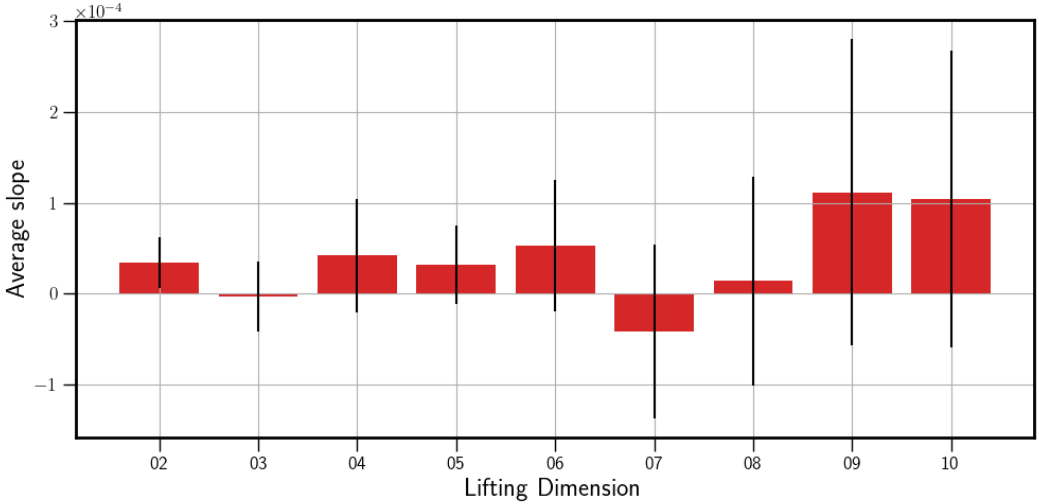




Results - Duffing linear fit slopes

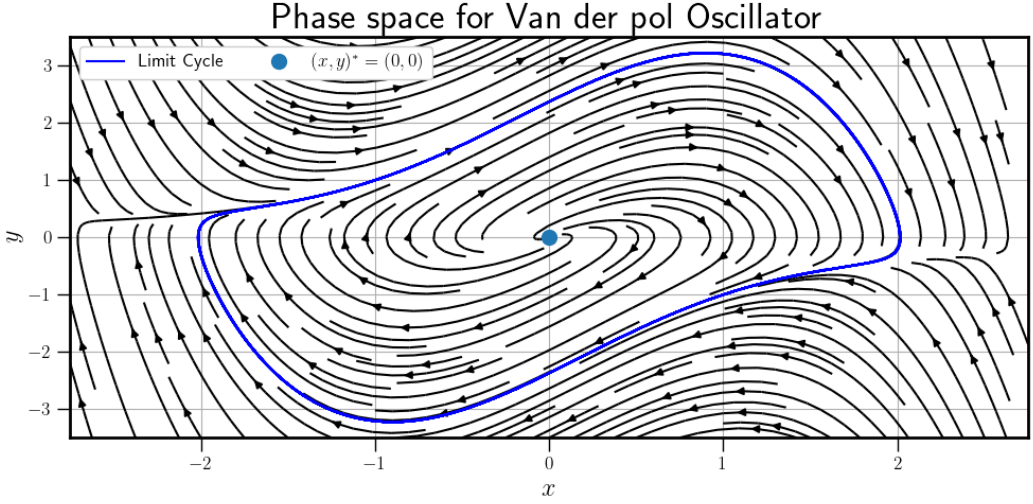


Results - Duffing linear fit slope averages and variances

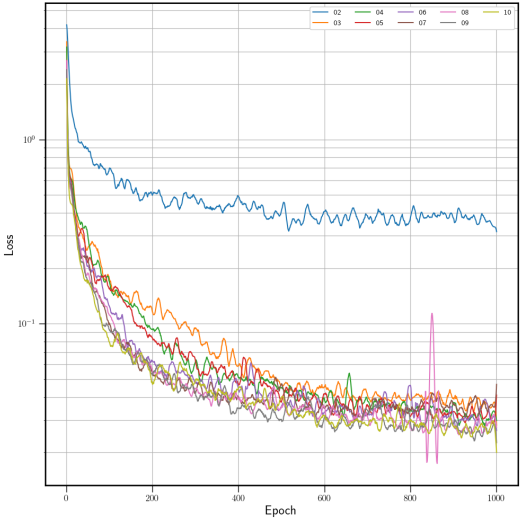


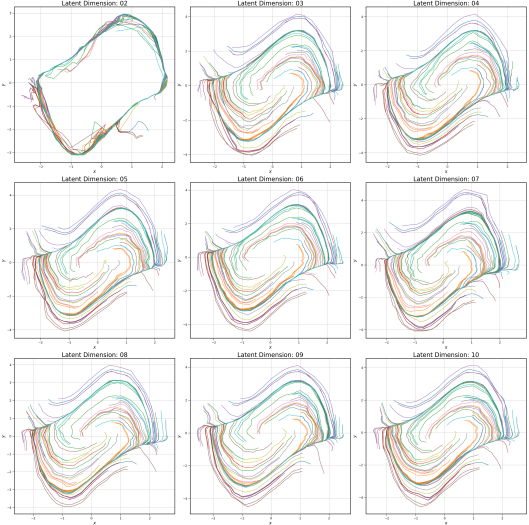
Results - Duffing notes

Results - Van der Pol

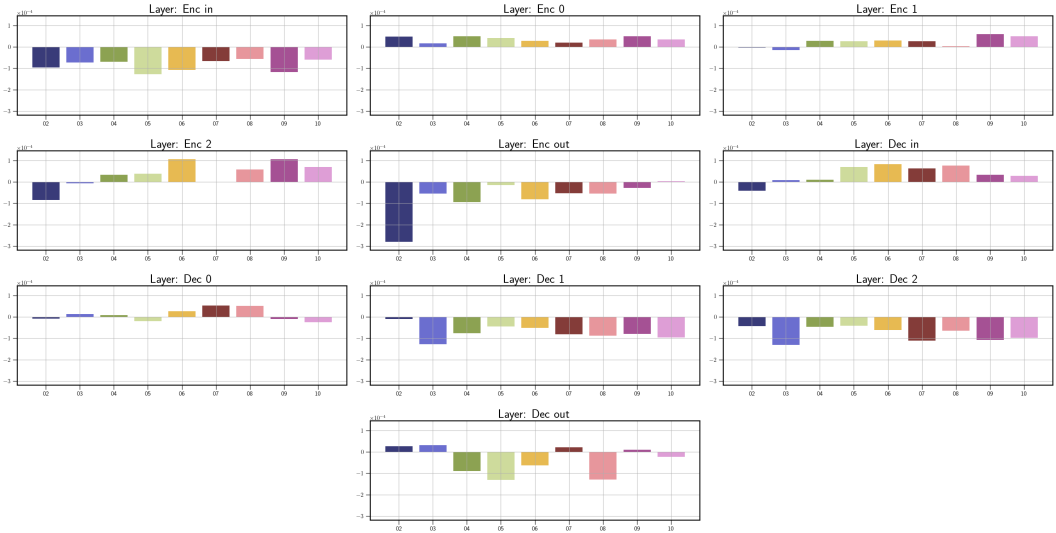


Results - Van der Pol loss curves and phase space

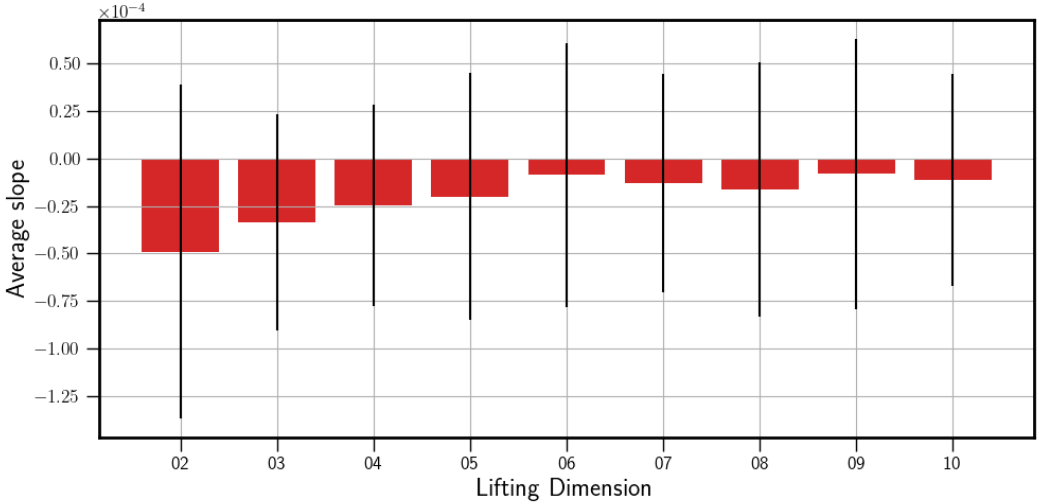




Results - Van der Pol linear fit slopes



Results - Van der Pol linear fit slope averages and variances



Results - Van der Pol notes

Discussion

References

D.J. Alford-Lago, C. W. Curtis, A. T. Ihler, and O. Issan.
Deep Learning Enhanced Dynamic Mode Decomposition.
2022.

Aurelien Geron.
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.
O'Reilly Media, Inc., 2nd edition, 2019.

B.O. Koopman.
Hamiltonian systems and transformations in Hilbert space.
Proc. Nat. Acad. Sci., 17:315–318, 1931.

Sergios Theodoridis.
Machine Learning: A Bayesian and Optimization Perspective.
Academic Press, Inc., USA, 1st edition, 2015.

The End!

Thank you for your time!
Questions?