A study on quantifying effective training of DLDMD

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Applied Mathematics

with a Concentration in

Dynamical Systems

---

by

Joseph Aaron Gene Diaz

July 1, 2022

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Joseph Aaron Gene Diaz:

A study on quantifying effective training of DLDMD

Christopher Curtis, Chair
Department of Mathematics and Statistics

Jérôme Gilles
Department of Mathematics and Statistics

Saeed Manshadi
Department of Electrical Engineering

Approval Date

DEDICATION

Everyone who's helped me get this far: You know who you are.

Kurt Gödel showed that math was not complete many years before David Hilbert died; so why did Hilbert have "We must know, we shall know." put on his grave stone when it was already shown to be impossible?

– Joseph Diaz

ABSTRACT OF THE THESIS

A study on quantifying effective training of DLDMD
by
Joseph Aaron Gene Diaz
Master of Science in Applied Mathematics with a Concentration in Dynamical Systems
San Diego State University, 2022

Write this once everything else is written.

TABLE OF CONTENTS

PAGE

ABSTRACT vi

LIST OF TABLES viii

LIST OF FIGURES ix

GLOSSARY x

ACKNOWLEDGMENTS xi

CHAPTER

1 INTRODUCTION 1

1.1 What is E-DMD? 1

1.2 What are Neural Networks? 3

1.3 DLDMD and it's limitations 4

1.4 Statement of Dilemma 6

2 KULLBACK-LEIBLER DIVERGENCE 8

2.1 Basic Definitions 8

2.2 How to build distributions (Non-parametric statistics) 9

2.3 Implementation Details 9

3 RESULTS 11

3.1 Duffing 11

3.2 Van der Pol 13

3.3 Lorenz (Maybe) 15

4 DISCUSSION 16

BIBLIOGRAPHY 17

LIST OF TABLES

PAGE

LIST OF FIGURES

PAGE

GLOSSARY

ACKNOWLEDGMENTS

I would like to thank Dr. Gauss for allowing me to work on this thesis even though he has been dead for so many years. I would also like to thank Dr. Bolzano for not having any comments on this thesis, and I would like to thank Dr. Knuth for being the only living person on my thesis committee, and for writing the wonderful TEX.

This thesis is partially protected against the evil forces of the Montezuma Publishing thesis reviewers by the magic of the Department of Mathematics and Statistics Master's Thesis LATEX Template.

CHAPTER 1
INTRODUCTION

In the study dynamical systems a central problem is how to derive models from measured data to facilitate the prediction of future states. Many approaches and techniques exist in the literature, from deriving sets of governing equations via application of the simple physical principles to the statistically meaningful Principal Component Analysis and other modal decompositions. The main goal of many of these methods is to come up with a generalized framework so that the dynamics can be extracted from the data for the sake of control and prediction. While there are genuine advantages to the more concrete and scientifically-sound methods of deriving models from first principles, it is often very difficult if not outright untenable.

As such, data-driven methods have rapidly become very useful approaches for coming up with rough and ready models.

Say more once you've thought of it.

1.1 What is E-DMD?

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

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

As established in the introduction, there is an old and venerable literature dedicated to deriving system using classical methods. What this literature will admit is that such a process is not trivial and even difficult or impossible to do in practice; particularly for nonlinear phenomena worth investigating. We want something that is more easily generalized and algorithmic.

One method that can be leveraged for this is via the Koopman Operator $\mathcal{K}$. If we denote $\boldsymbol{\varphi}(t;\boldsymbol{x}) = \boldsymbol{y}(t)$ 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 [6], there exists a linear representation of the flow map given by $\mathcal{K}$;

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

which means that $\mathcal{K}$ is a linear time-advancement operator of the dynamics. This might seem like it trivializes the problem, given that we now have a linear system, but it does

not. The Koopman operator is an *infinite* dimensional operator, so we've traded a potentially nonlinear problem for an infinite dimensional linear one.

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

$$\mathcal{K}^t\boldsymbol{\phi}_\ell = \exp(t\lambda_\ell)\boldsymbol{\phi}_\ell \implies g(\boldsymbol{x}) = \sum_{\ell\in\mathbb{N}} c_\ell\boldsymbol{\phi}_\ell(\boldsymbol{x}),$$

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

$$\mathcal{K}^t g(\boldsymbol{x}) = \sum_{\ell\in\mathbb{N}} a_\ell\boldsymbol{\phi}_\ell(\boldsymbol{\varphi}(t,\boldsymbol{x}))$$

The most useful property of this framing is that we have a global linearization of the flow, with a major caveat: Generally, finding these eigenvalues and eigenfunctions is impossible. This led to the development of the already mentioned and much much lauded Dynamic Mode Decomposition (DMD) and it's extensions, which seeks to numerically approximate a finite number of these modes. We'll focus on the particular extension that [1] implements: Extended DMD (E-DMD) [8]. With E-DMD, we suppose that

$$g(\boldsymbol{x}) = \sum_{\ell=1}^{N_O} a_\ell\boldsymbol{\psi}_\ell(\boldsymbol{x})$$

which is to say that the observable g exists in a finite dimensional subspace of $L^2(\mathcal{M})$, applying the Koopman operator implies that

$$\begin{aligned}\mathcal{K}^{\delta t} g(\boldsymbol{x}) &= \sum_{\ell=1}^{N_O} a_\ell\boldsymbol{\phi}_\ell(\boldsymbol{\varphi}(\delta t,\boldsymbol{x})) \\ &= \sum_{\ell=1}^{N_O} \boldsymbol{\phi}_\ell(\boldsymbol{x})(\boldsymbol{K}_O^T\boldsymbol{a})_\ell + r(\boldsymbol{x};\boldsymbol{K}_O)\end{aligned}$$

for discrete time step δt. K_O is the $N_O \times N_O$ matrix that minimizes

$$\boldsymbol{K}_O = \underset{K}{\operatorname{argmin}}\, ||\Psi_+ - K\Psi_-||_F^2$$

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 space holds, then r is identically 0. We define $\Psi_\pm$ to be

$$\Psi_- = (\Psi_1\ \Psi_2\ \cdots \Psi_{N_T}), \quad \Psi_+ = (\Psi_2\ \Psi_3\ \cdots \Psi_{N_T+1})$$

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 an SVD, with which we can write

$$\Psi_- = U\Sigma W^\dagger \implies \boldsymbol{K}_O = \Psi_+ W \Sigma^{-P} U^\dagger$$

where $-P$ denotes the Moore-Penrose pseudo-inverse and $\dagger$ denotes the conjugate transpose. This gives us an expression for r in terms of the observables Ψ:

$$E_r(\boldsymbol{K}_O) = \left|\left|\Psi_+(I - WW^\dagger)\right|\right|_F$$

Finding the eigenvalues, eigenfunctions, and Koopman modes comes down to an eigen-decomposition, from which the dynamics can be approximated as

$$y(t;\boldsymbol{x}) \approx V \exp(t\Lambda) V^{-1} \boldsymbol{\Psi}(\boldsymbol{x})$$

where $\boldsymbol{K}_O = VTV^{-1}$, $\Lambda_{\ell\ell} = \ln(T_{\ell\ell})/\delta t$ is a diagonal matrix and $\boldsymbol{\Psi}$ is the representation of the initial condition in terms of the observables.

Expand on everything else above for a more complete picture of E-DMD.

1.2 What are 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.

The oldest known example of NNs were the perceptrons of the 1950s, which could perform binary classifications of linearly separable data. Multilayer versions of this simple architecture allowed for classification of larger set of classes, but the researcher of the time arrived at the limits of their computing environments rather

quickly and a brief dark age in the study and application of neural networks ensued. Interest was renewed in the 1980s when many of the hardware limitations of previous decades were ameliorated and new algorithms for optimizing had been pioneered.

Expand once you've got more to say.

In modern mathematical study and application, the question of function representation is an ever-present one; specifically with respect to approximating quantities that can be given via series. While truncations and use of the likes of fourier series are useful and effective in some narrow applications, one means of representation that has shown much promise in recent years due to the greater availability of computing power is through the use of Neural Networks (NNs). In 1989, [5] showed that, with some appropriate parameters, a NN can be used to approximate any continuous function from one vector space to another to any arbitrary precision required.

Expand once you've got more to say.

1.3 DLDMD and it's limitations

Incomplete picture, add more detail later. Consider re-skimming Jay's paper.

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. This is implemented by defining an encoder $\mathcal{E} : N_S \to N_O$ and decoder $\mathcal{D} : N_O \to N_S$ composed of Dense layers such that

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

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. As such, a loss function that correctly identifies and prioritizes the desired properties is a necessary condition for the DLDMD to function as needed. A natural choice considering these constraints is given by

$$\mathcal{L} = \alpha_1 \mathcal{L}_{\text{recon}} + \alpha_2 \mathcal{L}_{\text{dmd}} + \alpha_3 \mathcal{L}_{\text{pred}} + \alpha_4 \left|\left|\boldsymbol{W}_g\right|\right|_2^2$$

where

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

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

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

Each component guides the machine to a particular outcome:

1. $\mathcal{L}_{\text{recon}}$ is the Mean Squared Error (MSE) of each time step with respect to the reconstruction from the composition of $\mathcal{E}$ and $\mathcal{D}$. This component ensures that, under training, the network effectively acts as a near identity transformation for data that is fed into it. This quality allows the DMD advanced trajectories to be recovered from the higher dimensional latent space back to the original dimension of the data.
2. $\mathcal{L}_{\text{DMD}}$ is the error associated with DMD. Consequently, this component is the one that is most responsible for finding the optimal set of observables for DMD. $\mathcal{E}$ immerses the data into a higher dimensional latent space, in effect acting as our set of observable; minimizing this gives us greater flexibility in the latent space.
3. $\mathcal{L}_{\text{pred}}$ is the (MSE) for each forward time prediction due to DMD and immersion/submersion due to $\mathcal{E}/\mathcal{D}$. In addition to balancing the last conditions, this condition ensures that the DMD step in the latent dimension is consistent with the next time-step from the time series after encoding, advancing, and decoding.
4. $\boldsymbol{W}_g$ is the vectorized quantity that represents all weights in both $\mathcal{E}$ and $\mathcal{D}$. This is really only a regularization condition to keep the coefficients of the weight matrices from blowing up in value as the model trains, which can be a concern for ML models.
5. $\alpha_1,\ \alpha_2,\ \alpha_3$, and α_4 are 4 positive constants that allow us to assign a weighting to each component of the loss. This allows the loss function to be dynamically weighted to prioritize some conditions over others. In [1], $\alpha_1 = \alpha_2 = \alpha_3 = 1$ and $\alpha_4 < 10^{-10}$.

The marriage of E-DMD with Dense NNs and some meaningful choices of loss function as an innovation was made by [1, 3, 7], and has proved to be quite a robust method for learning dynamics from data. In researching this thesis, many attempts were made to test the limits of DLDMD. [1] was primarily interested in recreating the phase space behavior of a handful of well known nonlinear oscillators such as the nonlinear harmonic oscillator, the Duffing oscillator, and the Van der Pol oscillator. Additionally, chaotic systems like Lorenz 63 and the Rössler system were examined but chaos proved to be quite a challenge for the algorithm to overcome. A study into the

addition of gaussian noise led to the discovery of the machine being quite robust to data due to the characteristics of it's loss function and additional noise mitigation was accomplished by implementing convolutional layers instead of traditional dense layers.

1.4 Statement of Dilemma

Having estabilshed DLDMD and having discussed testing it's limitations via training new models and *visually* qualifying each run as a success or failure, a question is begged: Can we, in an empirical and quantitative fashion, determine whether good training is taking place or otherwise deduce whether our model is well-posed given the hyperparameters? A machine learning model that must learn how to accomplish even a simple task can take quite a long time and plenty of computing resources to complete it's training and only then can the results be checked to determine whether productive training was taking place as opposed to the optimizer becoming stuck in a local minimum or there being no global minimum at all.

How can this be quantified? A useful starting point would be to examine the weights of the matrices that make up the layers of a model. For Dense layers, passing a vector of data $\boldsymbol{x} \in \mathbb{R}^d$ through a layer L can be written as

$$L(\boldsymbol{x}; A, \sigma, b) = \sigma(A\boldsymbol{x} + b)$$

for some matrix $A \in \mathbb{R}^{D\times d}$, vector $b \in \mathbb{R}^D$ and (typically nonlinear) activation function $\sigma : \mathbb{R}^D \to \mathbb{R}^D$. For a given layer, σ is a fixed hyperparameter of the machine, but both A and b have entries that are tuned by the optimizer as training takes places. Training takes place on a per epoch basis for most models, so considering the configuration of these matrices epoch to epoch can help us determine whether they are converging to a set of elements that accomplish the desired task or not. The next state for each layer is dependent on the previous, so a reasonable framing would be as one of a discrete dynamical system of the form

$$Q_{n+1} = P(Q_n) \quad \text{(Come up with better symbols later)}$$

where P is the training procedure and Q_n represents the configuration of the network's weights at each epoch n. Typically, the initial state Q_0 is "random" in the sense that the weights are selected from a probability distribution. In the interest of examining convergences, one might consider the following limit

$$\lim ||Q_{n+1} - Q_n||_2$$

for some L^2 norm on the space that Q_n inhabits. While this "one-step" Cauchy convergence will tell us whether the machine is approaching a particular configuration

point-wise, this might not actually tell us much of anything else. The changes between epochs is, in some sense, stochastic due to not knowing how the optimizer will update Q between epochs. As such, we propose a more statistical approach wherein we examine how the information content is changing from epoch to epoch.

More details forthcoming, get more details on why euclidean difference may be unhelpful.

Not helpful because it tells us nothing about information content or expressivity of neural network architecture. All we get is convergence; useful but not terribly so.

CHAPTER 2

KULLBACK-LEIBLER DIVERGENCE

In the field of information theory, it is often useful to frame information as being represented by a random variable and the values it can take on. For example, consider a normally distributed random variable $X \sim N(\mu, \sigma^2)$ where μ is the mean or expected values $\mathbb{E}[X]$ of X and σ^2 is the variance $\mathrm{Var}[X]$ of X. This is one of the most well understood probability distributions due to countless years of study into properties, as such we know that if we were to draw a sample for X that we're much more likely to get something close to μ than we to get something far away from μ.

Examine Goofy gaussian variance change, same mean.

Bias vs Variance.

De-trending. KL div of differences.

2.1 Basic Definitions

The KL Divergence (also called relative-entropy) is a statistical distance between a pair of probability distributions which measures how different a distribution P is from a reference distribution Q. "A simple interpretation of the divergence of P from Q is the expected excess surprise from using Q as a model when the actual distribution is P." If P and Q are discrete distributions defined on the probability space $\mathcal{X}$, then the KL Divergence of P with respect to Q is given by

$$D_{KL}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(P(x)/Q(x) \right)$$

In other words, it is the expectation of the logarithmic difference between the probabilities P and Q, where the expectation is taken using the probabilities P. As a "distance", the KL Divergence is non-negative and 0 when $P = Q$ almost everywhere. Unlike more standard metrics, it is not symmetric and does not satisfy the triangle equality. In order to deal with something that is, at least, symmetric we'll consider using the symmetric KL Divergence which is defined as

$$D_{SKL}(P \parallel Q) = \frac{1}{2} \left(D_{KL}(P \parallel Q) + D_{KL}(Q \parallel P) \right)$$

Which is the average of the KL Divergences for P with respect to Q and for Q with respect to P.

2.2 How to build distributions (Non-parametric statistics)

The question remains about how we obtain the probability distribution of the random variable that represents the evolution of each layer of an ML model from measured data. Many techniques exist, but we'll be using Kernel Density Estimation (KDE). The most basic idea behind KDE is using a different bin centering method and a smoothing factor, called a kernel, to approximate a probability density f from measured data in the form of a histogram. This is accomplished with the *kernel density estimator* for f given by:

$$\hat{f}(x; K, h) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where K is the chosen kernel, h is what's called the bandwidth parameter, and $\{x_i\}_{i=1}^{n}$ is the data that generates our histogram. Per [reference], the choice of kernel does not provide much statistical significance; the choice of the bandwidth parameter h is very crucial for finding a density estimate that approximates the underlying density function appropriately. Choosing an optimal h depends on the given data and the common method, Silverman's rule of thumb, works on the assumption that the density function of the data is unimodal and close to normal. For general data, on which no assumptions of normality are made, a more general method can be found using the Improved Sheather Jones algorithm.

2.3 Implementation Details

Ok, so what the hell did we actually do?

1. Ran a series of DLDMD models on two very well understood dynamical systems:
   - Van der Pol oscillator and Multiple nonlinear centers Duffing oscillator
   - Used latent/lifting dimensions 2 - 15
   - Trained for 1000 epochs
   - Saved configuration of DLDMD network weights at each epoch
   - Used different data for each training, but each data set was drawn from the same system's dynamics. Working on reproducing with same data/determining if that's necessary.
2. Detrended weight data via differencing and used Kernel Density Estimation to create empirical PDFs from inter-epoch weights.
   - For sake of ensuring statistical significance, kernel and bias weights are combined.

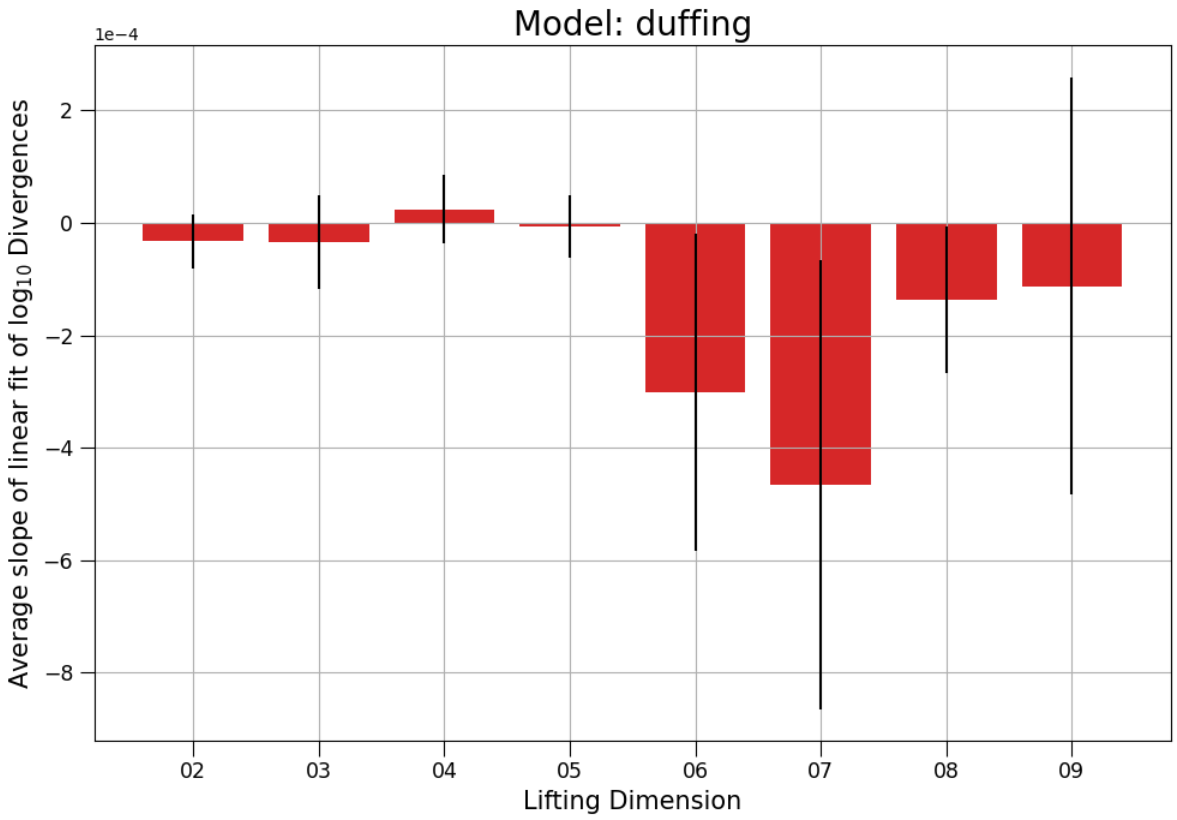
- Used Epanechnikov kernel (optimal in a mean square error sense) which is $K(x) = 3(1 - x^2)/4$ for $|x| < 1$ else $K(x) = 0$ and has finite support [4].
- Kernel choice not stastically significant [4].
- Bandwidth computed using Improved Sheather-Jones method [2]

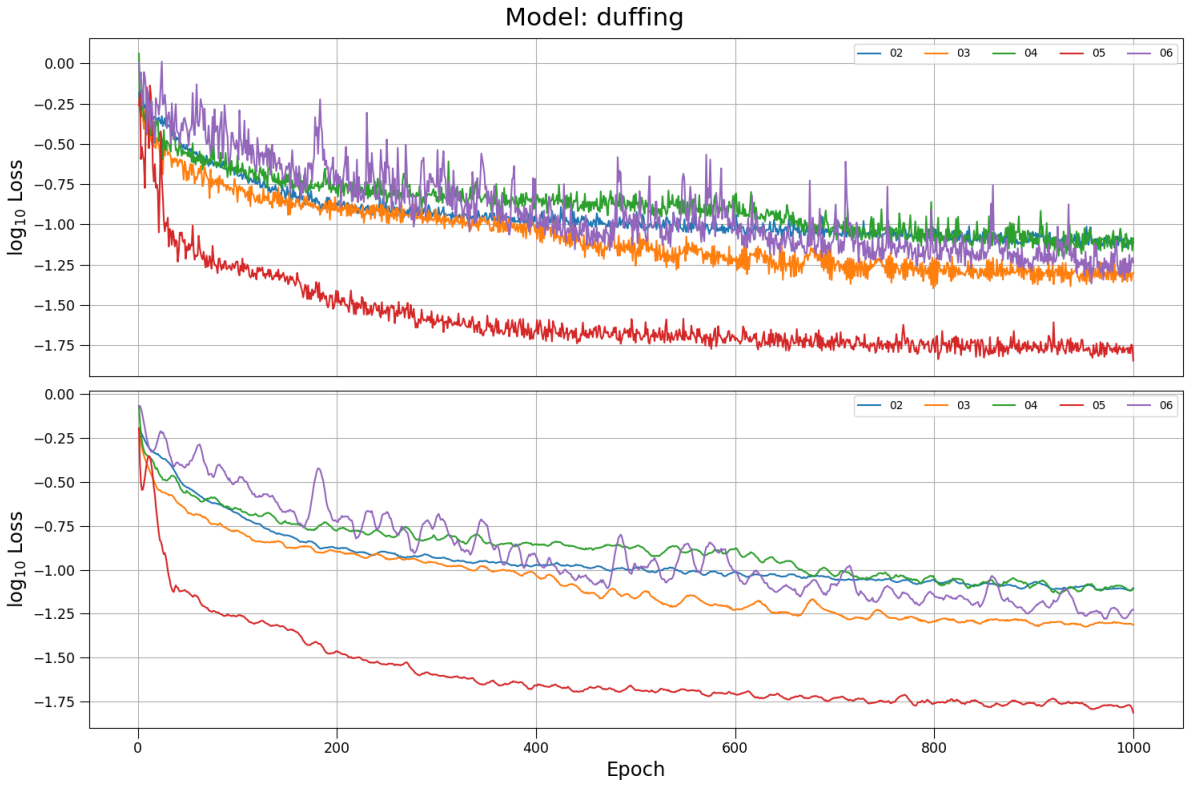
3. Used symmetric KL Divergence to find "distance" between empirical PDFs from data.
   - Both PDFs are assumed to have same support.
   - Using Simpson's Rule with 10,000 points for numerical integration, considering approximating fourth derivative for error bound on integration.
   -

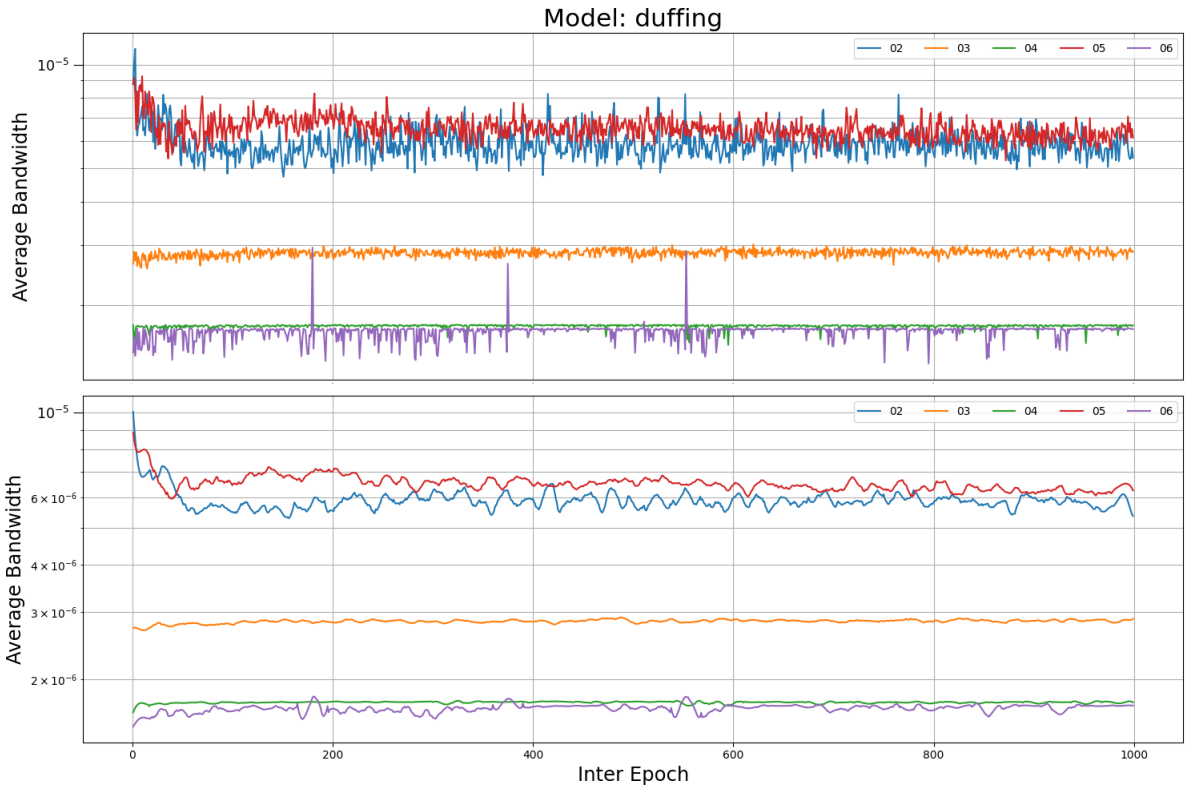
Integration via Simpson's Rule (?) Ask Chris about this. Since we are using density estimation, we do not have access to the actual probability densities for computing the divergences between consecutive distributions, we only have the data driven approximations from KDE. As such, we turn to numerical integration. Many strategies for this exist, we default to Simpson's Rule due to it's... accuracy.

CHAPTER 3
RESULTS

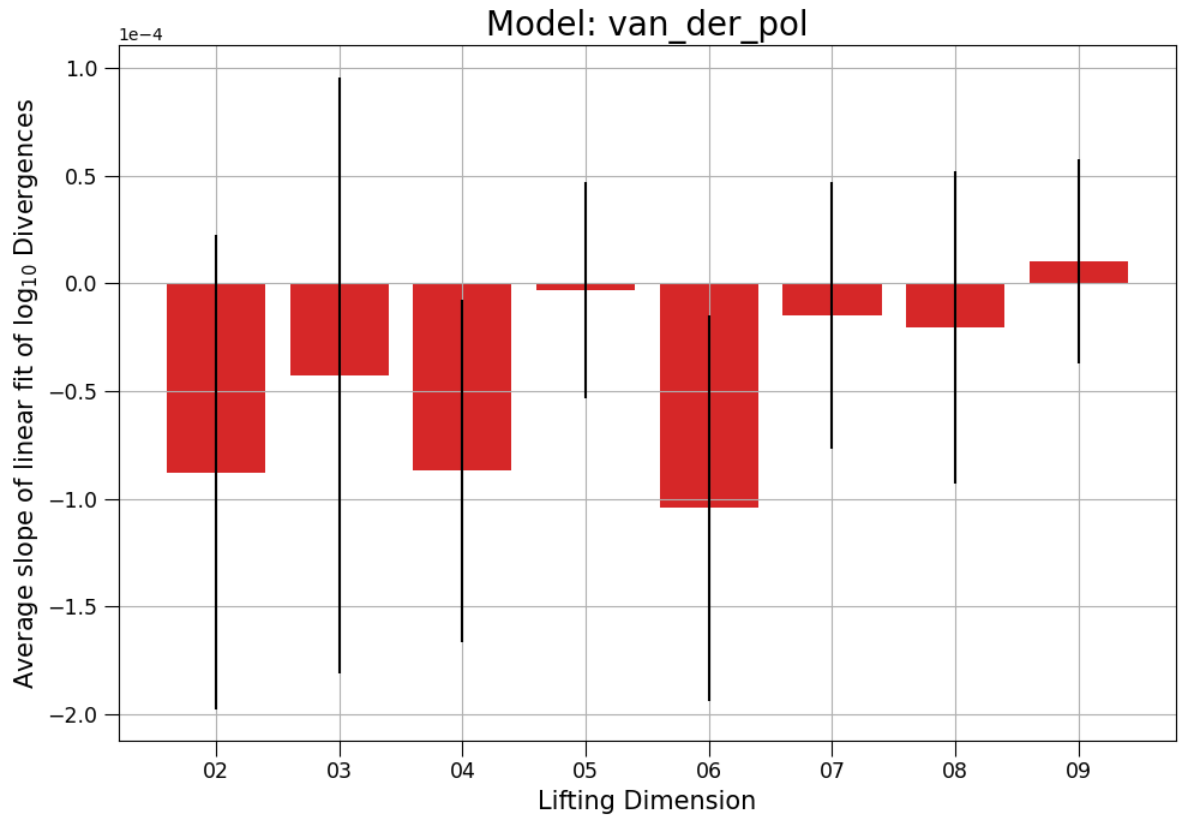
3.1 Duffing

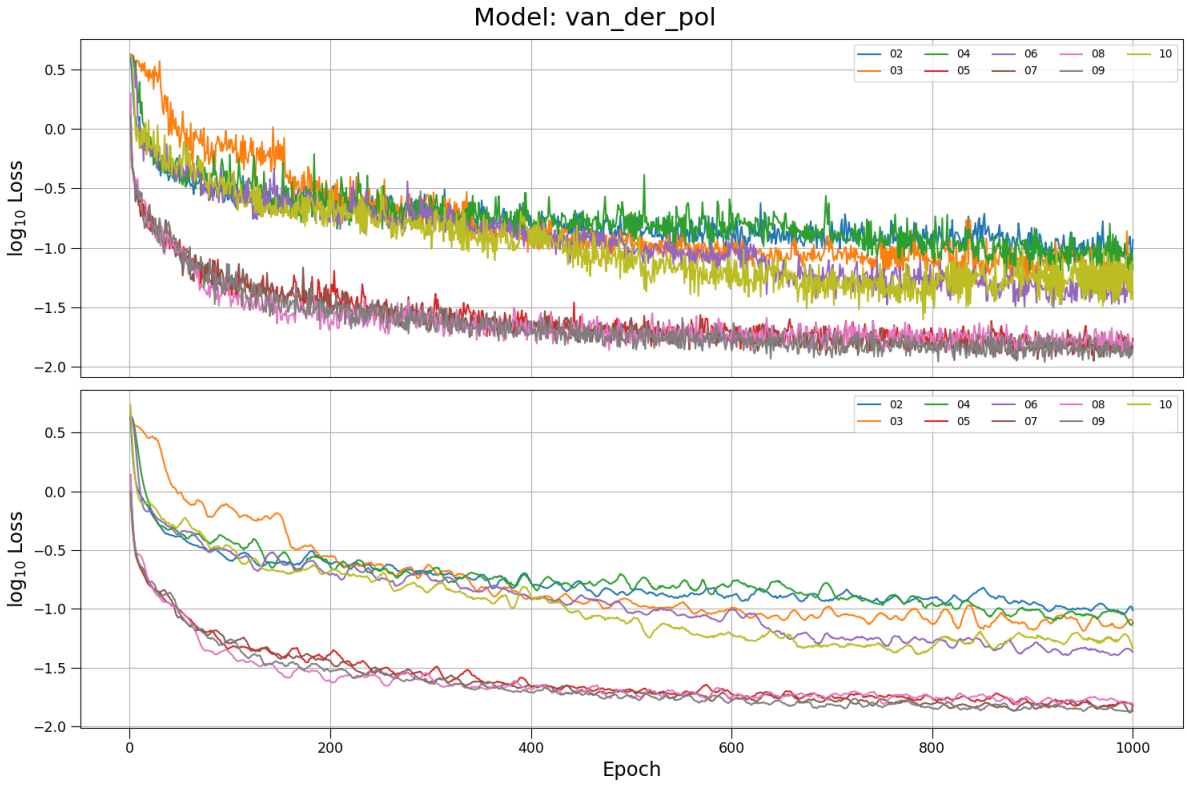

Model: duffing
1e−4
Average slope of linear fit of log10 Divergences
2
0
−2
−4
−6
−8
02
03
04
05
06
07
08
09
Lifting Dimension

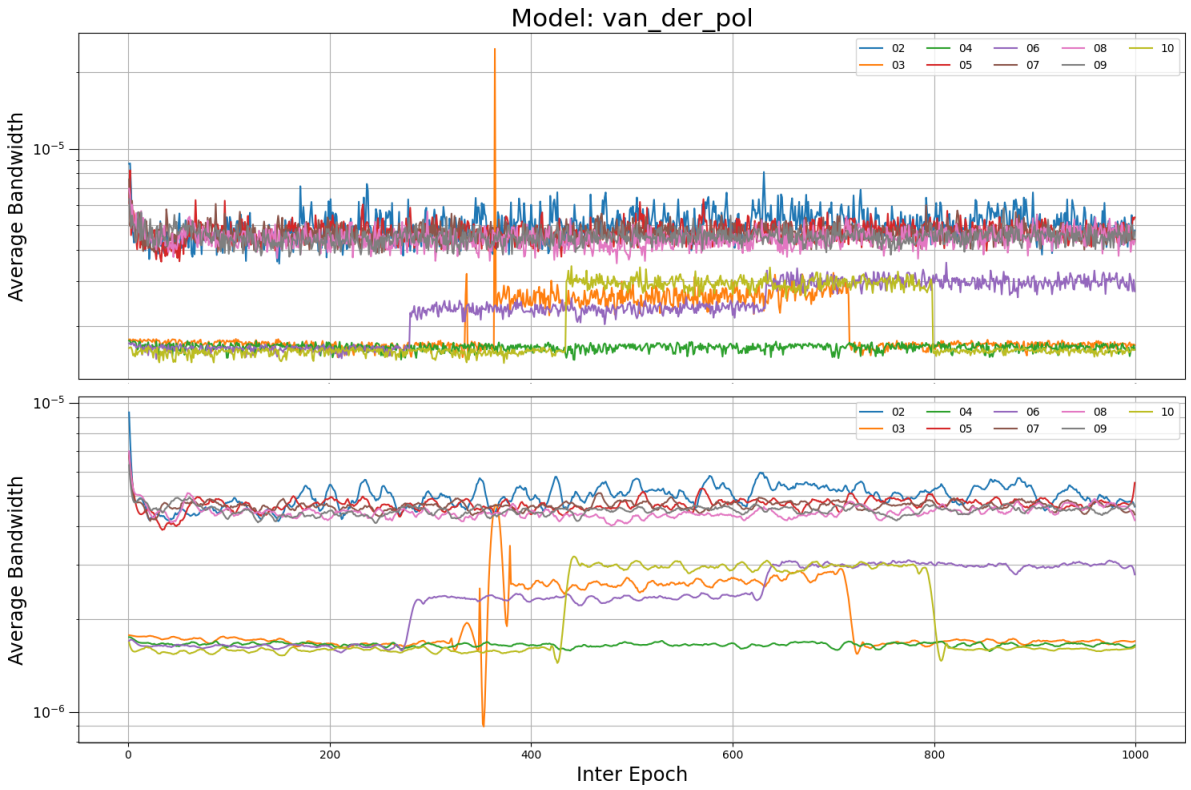

Model: duffing
log10 Loss
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
−1.75
02
03
04
05
06
0
200
400
600
800
1000
Epoch



3.2 Van der Pol


Model: van_der_pol
1e−4
Average slope of linear fit of log10 Divergences
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
02
03
04
05
06
07
08
09
Lifting Dimension


Model: van_der_pol
log10 Loss
02
03
04
05
06
07
08
09
10
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
200
400
600
800
1000
Epoch



3.3 Lorenz (Maybe)

CHAPTER 4
DISCUSSION

BIBLIOGRAPHY

[1] D. Alford-Lago, C. W. Curtis, A. T. Ihler, and O. Issan, *Deep Learning Enhanced Dynamic Mode Decomposition*, (2022).

[2] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, *Kernel density estimation via diffusion*, The Annals of Statistics, 38 (2010), pp. 2916 – 2957.

[3] S. Brunton, B. R. Noack, and P. Koumoutsakos, *Machine learning for fluid mechanics*, Ann. Rev. Fluid Mech., 52 (2020), pp. 477–508.

[4] V. A. Epanechnikov, *Non-parametric estimation of a multivariate probability density*, Theory of Probability & Its Applications, 14 (1969), pp. 153–158.

[5] K. Hornik, M. Stinchcombe, and H. White, *Multilayer Feedforward Networks are Universal Approximators*, Neural Networks, (1989).

[6] B. Koopman, *Hamiltonian systems and transformations in Hilbert space*, Proc. Nat. Acad. Sci., 17 (1931), pp. 315–318.

[7] B. Lusch, J. N. Kutz, and S. L. Brunton, *Deep learning for universal linear embeddings of nonlinear dynamics*, Nature Comm., 9 (2018), p. 4950.

[8] M. Williams, I. G. Kevrekidis, and C. W. Rowley, *A data-driven approximation of the Koopman operator: extending dynamic mode decomposition*, J. Nonlin. Sci., 25 (2015), pp. 1307–1346.