Improving on DLDMD with Convolutional Layers

Susana Munguia, Joseph Diaz

San Diego State University

May 2022

Introduction

"A central question in applied dynamical systems is one of generating models from measured data so as to facilitate prediction of unmeasured or future states. While the need for such work was clear to researchers several decades ago, the ongoing shift occurring in the physical sciences away from first principles modeling to data-driven modeling, coupled with the rise in computing power and algorithm design, has added impetus to the search for mathematically robust and generalizable model generating methods." - Abstract of [1].

Prediction from time series and the Koopman Operator

We seek a predictive model for a time series $\{\boldsymbol{y}_j\}_{j=1}^{N^T+1}$, which are measurements of an unknown dynamical system of the form

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

Classic approaches exist, we want something more generalizable. Consider the Koopman operator $\mathcal{K}$,

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

where $g : \mathcal{M} \to \mathbb{C}$ is an observable and $\boldsymbol{\varphi}(t; \boldsymbol{x}) = \boldsymbol{y}(t)$ is the flow map affiliated with $\boldsymbol{x}$, which is a global, linear representation of the flow by [4]. The trade-off is that, generally, this representation is infinite dimensional.

How to use $\mathcal{K}$

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

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

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

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

EDMD

Finding $\{\lambda_\ell\}$, $\{\phi_\ell\}$ analytically is impossible in general, so we numerically approximate them with Extended Dynamic Mode Decomposition (EDMD), by supposing that

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

For discrete time step δt, the dynamics can be described as

$$\mathcal{K}^{\delta t} g(\boldsymbol{x}) = \sum_{\ell=1}^{N_O} \psi_\ell(\boldsymbol{x})(\boldsymbol{K}_O^T \boldsymbol{a})_\ell + r(\boldsymbol{x}; \boldsymbol{K}_O)$$

where 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 represents some residual.

Where data^{TM} comes in

This where the data comes in. 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 Ψ_j is an observable of the original data $\boldsymbol{y}_j$. $\boldsymbol{K}_O$ approximates $\mathcal{K}$ for a one-step mapping and we can find with an SVD on Ψ_-. Finding the eigenvalues, eigenfunctions, and Koopman modes comes down to an eigen-decomposition and the dynamics are approximated as

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

for some representation $\boldsymbol{\Psi}$ of the initial condition. This is regular, data-driven EDMD; adding Machine Learning, we arrive at...

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.

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 to use for EDMD to do it work.

The Loss function

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

where

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

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

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

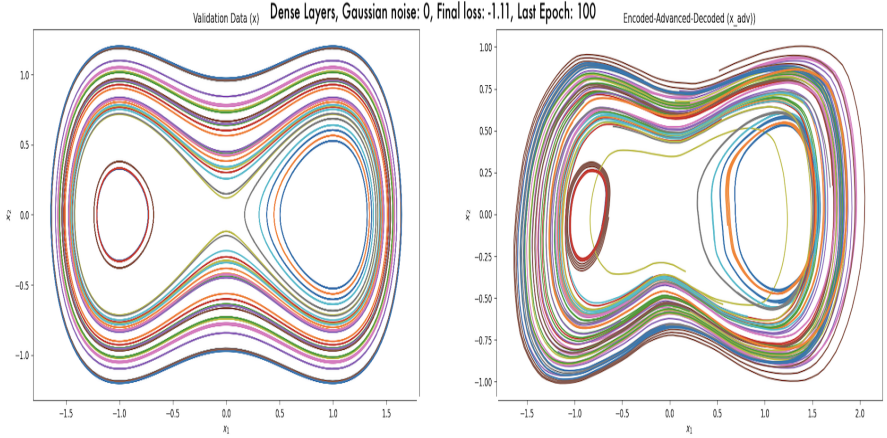
and $\boldsymbol{W}_g$ is vector representation of the weights of $\mathcal{E}$ and $\mathcal{D}$. The α's are regularization parameters that appropriately each component of the loss function.

Drawbacks of the DLDMD

As stated by [1],

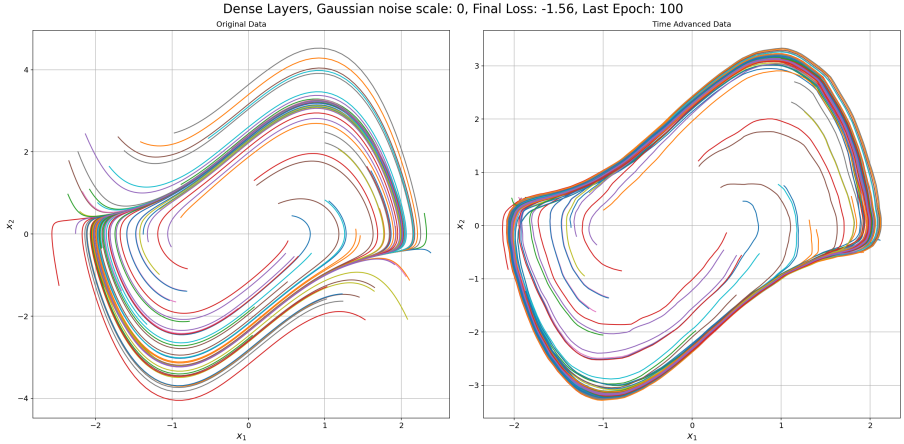
- The optimal latent space parameter N_0 is user chosen.
  - Improvement: Determine optimal way to approach Latent dimensionality
- Data generated is sampled uniformly [Unrealistic]
  - Improvement: Improve method to cope with sparseness ie sparse AE
- Chaotic Systems
  - additional innovations, such as incorporating delay embeddings in the EDMD step
- **Noisy data**
  - Suggestion: HAVOK, Our Brainstorm: **Change Loss Function + Auto Encoder**

Duffing: Noisy Data Implementation



Break point : 0.0254374

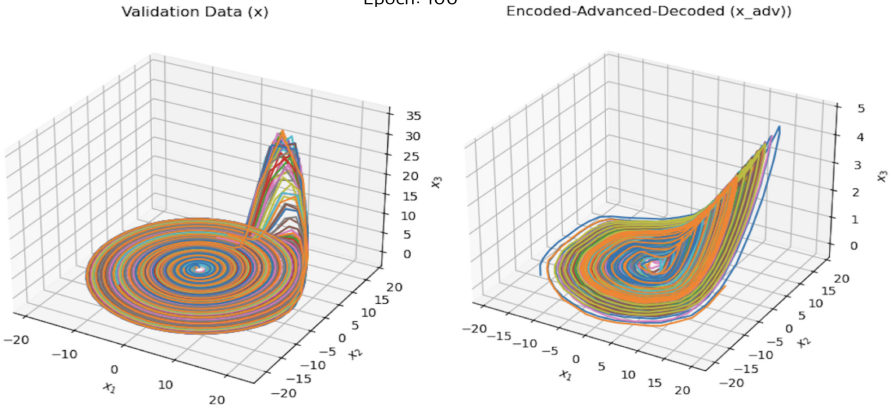
Van Der Pol : Noisy Data Implementation



Break point : 0.25625

Rossler: Noisy Data Implementation

Dense Layers, Gaussian noise: 0, Final loss: 1.16 Last Epoch: 100



Break point : 1.25

Loss Function: change ?

According to [3], amongst widely used loss functions, MAE and MSE are robust to noise.

Table 2: Accuracies under different noise rates (η) for all datasets (for Imdb, η's are halved). The last column gives accuracies under class conditional noise. In all the cases, standard deviation is shown only when it is more than 0.01

Data	loss	$\eta = 0\%$	$\eta = 30\%$	$\eta = 60\%$	CC
MNIST	CCE	0.9936	0.9138	0.5888	0.5725 (±0.0291)
	MAE	0.9916	0.9886	0.9799	0.9713
	MSE	0.9921	0.9868	0.9766	0.8505 (±0.0473)
RCV1	CCE	0.9126	0.8738	0.7905	0.7418 (±0.025)
	MAE	0.8732 (±0.0107)	0.8688	0.8637 (±0.0201)	0.8587
	MSE	0.9014	0.8943	0.8682 (±0.0120)	0.8315
Cifar 10	CCE	0.7812	0.5598 (±0.0170)	0.3083	0.1896
	MAE	0.7810 (±0.0190)	0.7011 (±0.0264)	0.5328 (±0.0251)	0.61425 (±0.0320)
	MSE	0.8074	0.7027	0.5257 (±0.0146)	0.6249 (±0.0359)
Imdb	CCE	0.8808	0.7729	0.6466	0.7858 (±0.0135)
	MAE	0.8813	0.8500	0.7352 (±0.0145)	0.8382 (±0.0127)
	MSE	0.8816	0.7725 (±0.0105)	0.6506 (±0.0103)	0.7874
News wire	CCE	0.7842	0.6905	0.4670	0.4973 (±0.0148)
	MAE	0.8081	0.7553	0.6357 (±0.0106)	0.6535
	MSE	0.7916	0.6626	0.4078 (±0.0172)	0.4377 (±0.0140)
News group	CCE	0.8006	0.7571	0.6435	0.5629
	MAE	0.7890	0.7749	0.7319	0.6772
	MSE	0.7999	0.7553	0.6347	0.5519

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

where

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

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

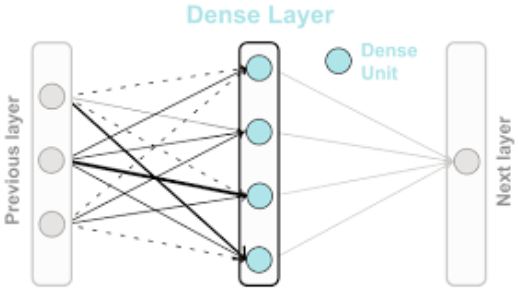
$$\mathcal{L}_{\text{pred}} = \frac{1}{N_T} \sum_{j=1}^{N_T} \left|\left| \boldsymbol{y}_{j+1} - \mathcal{D}\left(V \tilde{\Lambda}^j V^{-1} \mathcal{E}(x) \right) \right|\right|_2$$

The DLDMD loss function is already *quite* robust!

Convolution Autoencoder/ Decoder

Original DLDMD

Dense Layers in AE/D



Our change

Convolutional layers in AE/D

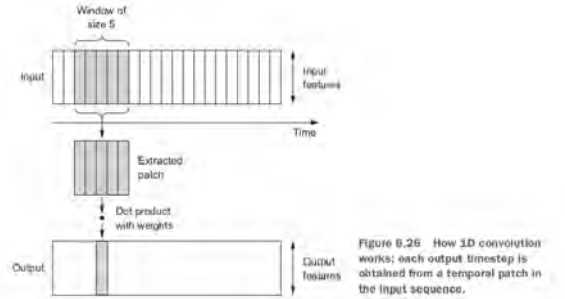
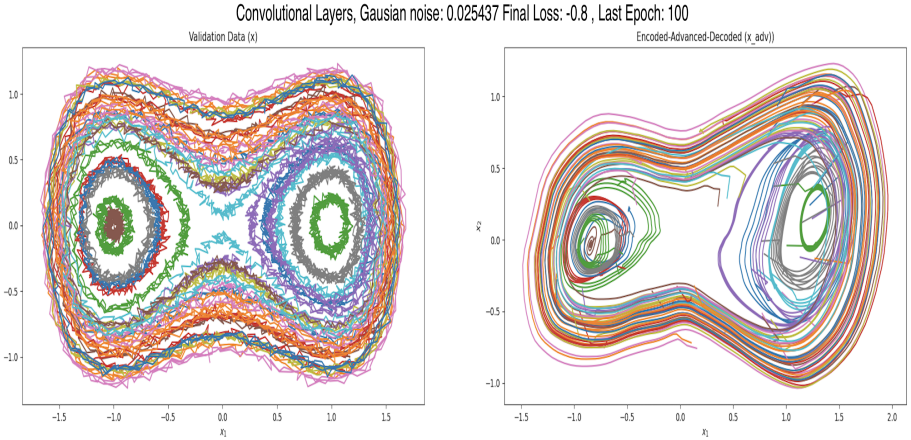


Figure 6.26 How 1D convolution works: each output timestep is obtained from a temporal patch in the input sequence.

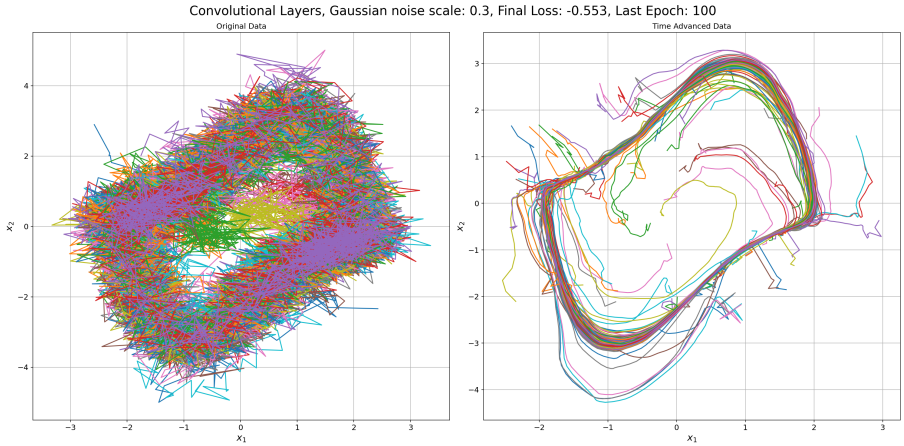
Duffing: CAE + noise



Previously, Break point: 0.0254374, Final Loss: 0.0529

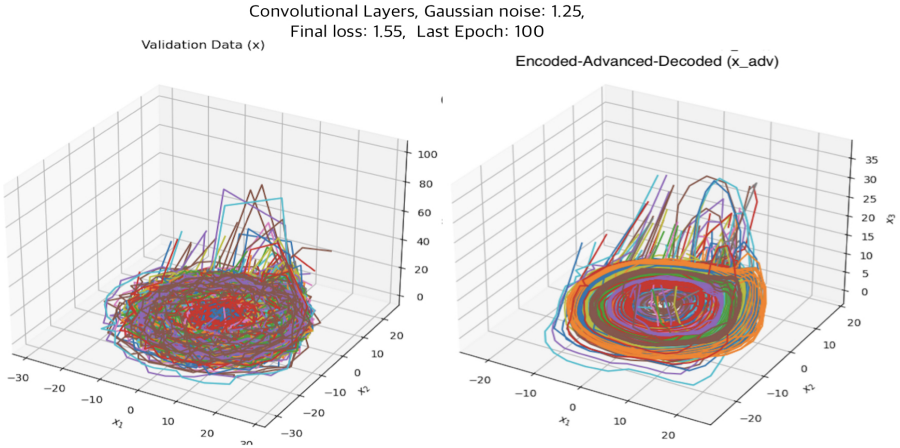
Previously, Break point: 0.0254374

Van Der Pol: CAE + noise



Previously, Break point: 0.25625 , Final Loss: 0.223

Rossler: CAE + noise



Previously, Break point: 1.25 , Final Loss: 1.698

Discussion

Getting good results for other chaotic systems like Lorenz 63 and the Chua circuit will require more sophisticated techniques, since (DL)DMD performs poorly on them even with perfect data. Some approaches [2] consider using Hankel matrices and Takens' Embedding to handle such cases, and integrating such an approach with convolutional layers for de-noising is a natural next step.

A caveat to all things ML is the crucial and sometimes arbitrary choice of hyper parameters.

The results for such a kernel size of the filter for the convolutional layers surprised us quite a bit, the author of [1] was even more surprised.

Conclusion

While not as scientifically satisfying as classical methods, data-driven methods like DMD fit nicely into the modern inquiry into systems from a data scientific perspective.

Using machine learning to extend and improve on data-driven numerical methods like DMD has born fruit in our heavily data dependent world and these models have proven their validity from a practical point of view.

References

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

S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz.
Chaos as an intermittently forced system.
Nature Comm., 8(1), 2017.

A. Ghosh, H. Kumar, and P. S. Sastry.
Robust loss functions under label noise for deep neural networks.
2017.

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

The End... ?

Questions?

What's the difference?

DLDMD

Using Koopman Operator Theory (DMD) and Autoencoders, find the best set of observables which best approximates the flow in the latent dimension which then can be projected into the original coordinates using the decoder.

- from data, find future prediction of dynamics [indefinite steps in future]

SINDy

Use sparse regression (NN) to find governing equations of given time-series data.

- from data, find dynamical equations

One-step time prediction of time

Use NN to construct best mapping from $[t_0, t_N] \longrightarrow [t_1, t_{N+1}]$

- from data, find future prediction of dynamics [one steps in future]