Forecasting Sequential Data Using Consistent Koopman Autoencoders — Supplementary Materials —

Omri Azencot [*1] **N. Benjamin Erichson** [*2] **Vanessa Lin** [2] **Michael W. Mahoney** [2]

1. Network architecture

In our evaluation, we employ an autoencoding architecture where the encoder and decoder are shallow and contain only three layers each. Using a simple design allows us to focus our comparison on the differences between the DAE model (Lusch et al., 2018) and ours. Specifically, we list the network structure in Tab. 1 including the specific sizes we used as well as the different activation functions. We recall that m represents the spatial dimension of the input signals, whereas κ is the bottleneck of our approximated Koopman operators. Thus, $p = 32 \cdot \alpha$ is the main parameter with which we control the width and expressiveness of the autoencoder. We facilitate fully connected layers as some of our datasets are represented on unstructured grids. Finally, we note that the only difference between our net architecture and the DAE model is the additional backward linear layer.

Type	Layer	Weight size	Bias size	Activation
Encoder	FC	$m \times p$	p	tanh
Encoder	FC	$p \times p$	p	tanh
Encoder	FC	$p \times \kappa$	κ	linear
Forward	FC	$\kappa \times \kappa$	0	linear
Backward	FC	$\kappa \times \kappa$	0	linear
Decoder	FC	$\kappa \times p$	p	tanh
Decoder	FC	$p \times p$	p	tanh
Decoder	FC	$p \times m$	m	linear

Table 1. Our network architecture, where $p = 16 \cdot \alpha$ with α controlling the width per encoder and decoder layer.

2. Computational requirements

The models used in this work are relatively shallow. The amount of parameters per model can be computed as follows $2(m+32+\kappa)\cdot 32\alpha + (4\cdot 32\alpha + \kappa + m) + 2\cdot\kappa^2$, corresponding to the number of weights, biases and Koopman operators, respectively. Notice that DAE is different than our model by having κ^2 less parameters.

We recorded the average training time per epoch, and we show the results for DAE and our models in Fig. 1 for many of our test cases. Specifically, the figure shows from left to right the run times for cylinder flow, noisy cylinder flow, sphere flow, linear pendulum, noisy linear pendulum, nonlinear pendulum, noisy nonlinear pendulum, SST and noisy SST. The behavior of our model is consistent in comparison to DAE for the different tests. On average, if DAE takes x milliseconds per epoch, than our model needs $\approx 1.8x$ time.

This difference in time is due to the additional penalty terms, and it can be asymptotically bounded by

$$\mathcal{O}(\mathcal{E}_{\text{bwd}}) + \mathcal{O}(\mathcal{E}_{\text{con}}) = \mathcal{O}(\lambda_s n m) + \mathcal{O}(\kappa^4) \ .$$

We note that the asymptotics for the forward prediction are equal to the backward component, i.e., $\mathcal{O}(\mathcal{E}_{\text{fwd}}) = \mathcal{O}(\lambda_s n m)$. The consistency term $\mathcal{E}_{\text{con}}$ is composed of a sum of sequence of cubes (matrix products) which can be bounded by κ^4, assuming matrix multiplication is $\mathcal{O}(\kappa^3)$ and thus it is a non tight bound. Moreover, while the quartic bound is extremely high, we note that a cheaper version of the constraint can be used in practice, i.e., $||CD - I||_F^2$. Also, since our models are loaded to the GPU where matrix multiplication computations are usually done in parallel, the practical bound may be much lower. Finally, the inference time is insignificant (≈ 1 ms) and it is the same for DAE and ours and thus we do not provide an elaborated comparison.

3. Backward prediction of dynamical systems

One of the key features of our model is that it allows for the direct backward prediction of dynamics. Namely, given an observation f_t, our network yields the forward prediction via $\hat{f}_{t+1} = \chi_d \circ C \circ \chi_e(f_t)$, as well as the backward estimate using $\check{f}_{t-1} = \chi_d \circ D \circ \chi_e(f_t)$. Time reversibility may be important in various contexts (Greydanus et al., 2019). For instance, given two different poses of a person, we can consider the trajectory from the first pose to the second or the other way around. Typically, neural networks require that we re-train the model in the reverse direction

[*]Equal contribution [1]Department of Mathematics at UC Los Angeles, CA, USA. [2]ICSI and Department of Statistics at UC Berkeley, CA, USA.. Correspondence to: Omri Azencot <azencot@math.ucla.edu>, N. Benjamin Erichson <erichson@berkeley.edu>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

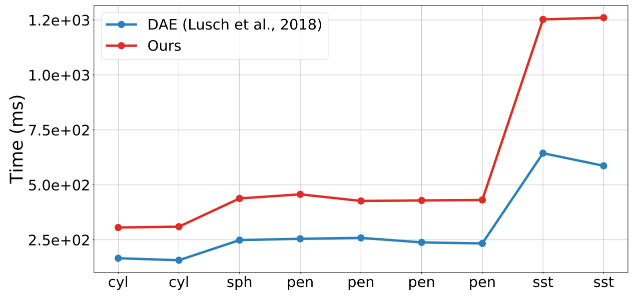

Figure 1. We show above the average run time for an epoch in milliseconds for several of the test cases in this work. In general, our model is almost two times slower than DAE during training.

to be able to predict backwards. In contrast, Koopman-based methods can be used for this task as the Koopman matrix is linear and thus back forecasting can be obtained simply via $\bar{f}_{t-1} = \chi_d \circ C^{-1} \circ \chi_e(f_t)$. We show in Fig. 2 the backward prediction error computed with $\bar{f}_{t-1}$ for the cylinder flow data using our model and the DAE (blue and red curves). In addition, as our model computes the matrix D, we also show the errors obtained for $\check{f}_{t-1}$. The solid lines correspond to the clean version of the data, whereas the dashed lines are related to its noisy version. Overall, our model clearly outperforms DAE by an order of magnitude difference, indicating the overfitting in DAE.

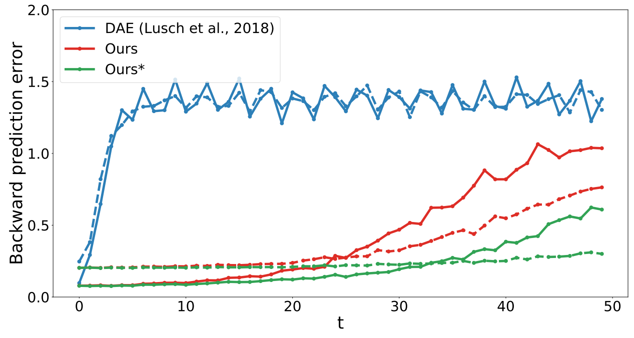

Figure 2. The cylinder flow data is used for backward prediction with our model (red) and DAE (blue). Our results hint that DAE overfits in the forward direction, whereas our network generalizes well when the time is reversed.

References

Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pp. 15353–15363, 2019.

Lusch, B., Kutz, J. N., and Brunton, S. L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, 2018.