Applications of Koopman Mode Analysis to Neural Networks

Ryan Mohr,[1] **Maria Fonoberova,**[1] **Iva Manojlović,**[1] **Aleksandr Andrejčuk,**[1] **Zlatko Drmač,**[2] **Yannis Kevrekidis,**[3] **Igor Mezić,**[1, 4]

[1] AIMdyn, Inc., Santa Barbara, CA 93101
[2] University of Zagreb, 10000 Zagreb, Croatia
[3] Johns Hopkins University, Baltimore, MD 21218
[4] University of California, Santa Barbara, CA 93106
mohrr@aimdyn.com, mfonoberova@aimdyn.com, imanojlovic@aimdyn.com, aleksandr@aimdyn.com, drmac@math.hr, yannisk@jhu.edu, mezici@aimdyn.com

Abstract

We consider the training process of a neural network as a dynamical system acting on the high-dimensional weight space. Each epoch is an application of the map induced by the optimization algorithm and the loss function. Using this induced map, we can apply observables on the weight space and measure their evolution. The evolution of the observables are given by the Koopman operator associated with the induced dynamical system. We use the spectrum and modes of the Koopman operator to analyze the convergence versus non-convergence of the training process. Our methods can help to determine if (1) a bad initialization of the network weights was used—in particular, how the existence of eigenvalues clustering around 1 determines when to terminate the learning process—allowing a restart before training too long and (2) to speed up the training time. Additionally, we show that incorporating structured loss functions based on negative Sobolev norms can allow for the reconstruction of a multi-scale signal polluted by very large amounts of noise. Using these Sobolev based loss functions improves robustness and interpretability.

Introduction

The training of neural networks is a topic of much interest due to their wide spread use in a variety of application domains, from image recognition and classification to solving ordinary and partial differential equations. Unfortunately, the dimensionality of the problem often prevents rigorous analysis. Viewing a neural network training as dynamical system (Chang et al. 2017; Dietrich, Thiem, and Kevrekidis 2020; Chang et al. 2019) provides a framework for a mathematical approach to training.

Our objective is to introduce the wider community to various results in applying methods from dynamical systems, in particular the operator-theoretic approach, to extract insight into both the choosing of the neural network's architecture, such as its depth, for a given problem and insight into the networks training process.

We consider the training process as a dynamical system acting on the high-dimensional weight space. Each epoch is an application of the map induced by the optimization algorithm and the loss function. Using this induced map, we can apply observables on the weight space and measure their evolution. The evolution of the observables are given by a linear operator, the Koopman operator, associated with the induced dynamical system. We use the spectrum and modes of the Koopman operator to obtain insight into the training process. This viewpoint can help to determine if we have a bad initialization of the network weights, allowing a restart before training too long. Additionally, we incorporate structured loss functions based on negative Sobolev norms into our network. Using these new types of loss functions (1) can improve interpretability of the networks and (2) can allow for significant noise rejection when training on noisy multiscale signals.

The rest of the paper is structured as follows. In the next section, we mathematically formulate the training of a neural network as a dynamical system and introduce the Koopman operator viewpoint of analyzing dynamical systems. The following section, investigates the convergence of the training process through the lens of the Koopman operator's spectrum. In the final section before the conclusions, we apply loss functions inspired by negative Sobolev norms and show how they can be used for significant noise rejection when trying to reconstruct a signal.

Neural network training as a dynamical system.

Let $n(\mathbf{x};\mathbf{w})$, $n : X \times \mathbb{R}^n \to \mathbb{R}^d$, be a neural network, where $\mathbf{x} \in X \subset \mathbb{R}^m$ is the input feature vector, $\mathbf{w} \in \mathbb{R}^n$ is the vector of network weights (parameters), and the output of the network is a d-dimensional real vector. Let $L_{tr}(\mathbf{w})$ be the loss function of the network on the training set as a function of the network parameter weights. The optimization problem that is to be solved is

$$\mathbf{w}^* = \arg\min L_{tr}(\mathbf{w}). \tag{1}$$

The loss function L_{tr} and the chosen optimization algorithm (e.g. stochastic gradient descent) induce a nonlinear, discrete time map on the network weights that updates the weights at each epoch:

$$T : \mathbb{R}^n \to \mathbb{R}^n, \qquad \mathbf{w}_{t+1} = T(\mathbf{w}_t), \tag{2}$$

where $\mathbf{w}_t$ are the network weights at the beginning of epoch $t \in \mathbb{N}_0$; $\mathbf{w}_0$ represents the initialized network weights prior to training. This induced map T is a discrete dynamical system having the weight space as its state space.

Trying to directly analyze the map T can be quite difficult as it is implemented as a black box in whatever training framework one is using. Instead one can track the evolution of observables to gain insight into the dynamical system by studying the spectral properties of an induced linear operator that drives the evolution of the observable. Let $\mathcal{F}$ be a function space that is closed under composition with T; that is, if $f \in \mathcal{F}$, then $f \circ T \in \mathcal{F}$. An operator $U : \mathcal{F} \to \mathcal{F}$ can be defined via this composition $Uf = f \circ T$. This operator, called the Koopman or composition operator, is linear (Mezić 2005), albeit infinite-dimensional, even if T is nonlinear. Even though it is linear, it can still capture the full nonlinear behavior of T. In many cases, the Koopman operator has a spectral decomposition (Mezić 2005; Budišić, Mohr, and Mezić 2012; Mohr and Mezić 2014; Mezić 2019)

$$Uf = \sum_{j=1}^{\infty} c_j \lambda_j \phi_j + \int_{\mathbb{C}} z dE(z) f, \tag{3}$$

where $c_j \in \mathbb{C}$ is a coefficient, $\lambda_j \in \mathbb{C}$ is an eigenvalue, $\phi_j \in \mathcal{F}$ is an eigenfunction of U, and $dE(z)$ is a projection-valued measure (PVM). The set of eigenvalues form the point spectrum. The PVM is associated with the continuous spectrum of U; it takes sets in the complex plane and associates projection operators on $\mathcal{F}$ with them. We can also consider the case of vector-valued observables $\mathbf{f} = (f_1, \ldots, f_m)$, where $f_i \in \mathcal{F}$. In this case, there is an analogous decomposition to (3), whose only difference is that the scalar-valued coefficients c_j become vector-valued coefficients $\mathbf{m}_j \in \mathbb{C}^m$. These vector-valued coefficients are called Koopman modes, originally called shape modes (Mezić 2005). Data-driven algorithms, like the family of Dynamic Mode Decomposition algorithms, e.g. (Schmid and Sesterhenn 2008; Rowley et al. 2009; Jovanović, Schmid, and Nichols 2012, 2014; Hemati, Williams, and Rowley 2014; Williams, Kevrekidis, and Rowley 2015; Drmač, Mezić, and Mohr 2018; Drmač, Mezić, and Mohr 2019), are used to approximate the modes and eigenvalues using a trajectory from a single initial condition. Thus we do not need explicit access to U to analyze the dynamical system. We use the DMD_RRR algorithm from (Drmač, Mezić, and Mohr 2018) in the following work.

Insights into neural network training convergence via the Koopman spectrum.

The main idea here is that by monitoring the spectrum of the Koopman operator during the training process will give us a method to determine when the training process should be terminated so that good performance is given on the testing set without the network memorizing the set. Having this indicator allow the network to generalize better.

Given the trajectory of all network weights $\{\mathbf{w}_t\}$, where $t \in \mathbb{N}$ is the training epoch, we use two different observables, a delay-embedded version of the (cross-entropy) loss function, $L_{tr}(\mathbf{w}_t)$ and the full-state observable which returns the network weights $\mathbf{w}_t$ at each training epoch t. Inspecting the Koopman spectrum of either of these observables tells us how quickly the system is learning and when a fixed point is appearing. The Koopman mode decomposition (3) can be written as

$$L_{tr}(\mathbf{w}_t) = \sum_k \mathbf{m}_k \lambda_k^t \phi_k + \mathbf{e}_t, \tag{4}$$

where $\mathbf{m}_k$ is the Koopman mode normalized to norm 1, λ_k is the associated eigenvalue, ϕ_k is the reconstruction coefficient and $\mathbf{e}_t$ is the error term at epoch t. If all the eigenvalues not equal to 1 satisfy $|\lambda_k| < 1$, then the training is stable (Mauroy and Mezić 2016). In this case, the slowest eigenvalue determines the rate of convergence. The importance of a mode is determined by the absolute value of the reconstruction coefficient, with higher values giving the mode more importance. The further inside the unit circle the eigenvalues corresponding to the important modes are, the faster the training.

The standard MNIST data set was used for testing the methods. The network tested was a convolutional network with two convolutional layers with 16 and 32 kernels, each followed by 2×2 max-pooling. Each kernel is 5×5. Convolutional layers are followed by fully-connected layer with 100 neurons, which is then followed by another fully-connected layer with 10 neurons, after which follows softmax classification. The architecture is shown on Figure 1. A cross-entropy loss function was used with a learning rate of $\eta = 1e$-3. Different weight initialization (He or Xavier) were tested as well. After each epoch, cross-entropy loss on train and test sets were recorded along with the network weights. All networks were trained for 1000 epochs, and KMD analysis was applied to the snapshots for 3 different epoch ranges specified in the figures.

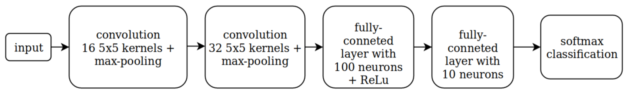
input
convolution 16 5x5 kernels + max-pooling
convolution 32 5x5 kernels + max-pooling
fully-conneted layer with 100 neurons + ReLu
fully-conneted layer with 10 neurons
softmax classification

Figure 1: Neural network architecture.

Figure 2 shows the results for the Xavier initialization scheme. The top left figure shows the cross entropy loss of the network evaluated on the training set at each epoch. The top right figure shows the cross entropy on the test set. The second row shows the KMD spectrum using the cross entropy loss function on the training set as the observable. The spectrum is computed using the first 40 (left), 100 (middle), and 500 (rights) snapshots of the observable. The third row shows the spectrum computed using the weight vectors as the vector-valued observable. The spectrum is computed using the first 50 (left), 100 (middle), and 500 (right) snapshots of the observable. The spectrum was computed using DMD_RRR. Eigenvalues inside the unit circle correspond to decaying modes, whereas eigenvalues outside the unit circle correspond to growing modes. Eigenvalues very close to the unit circle correspond to modes that change slower than modes associated with eigenvalues closer to zero.

Each of the initialization schemes trains very fast, with a large drop in the training error after a few epochs. The HE and Xavier initialization schemes seem to over-memorize the training set which we see as an increase in the cross entropy loss on the test set. However, the final errors on the test set are still lower than the final error for the random normal scheme on the test set.

Since for all initialization schemes, the training process is extremely fast, with a large drop in training set cross entropy, the eigenvalues clustered close to 0 in the second row of the figures (training set cross entropy observable) makes sense. As more snapshots are taken to be used to compute the spectrum, there seems to some important eigenvalues showing up close to 1, which would indicate that the training is nearing a fixed point for the training process. This trend seems clear in the final row of the plots which used the weights vector as the observable for the KMD computation. After 50 epochs, there are important eigenvalues clustered close to 0 and, as more snapshots are taken, important eigenvalues appear and cluster around 1.

Noise rejections using negative Sobolev norms in the loss function.

Here, we investigate the performance of different loss functions compared to the standard L^2 loss when learning a multiscale signal. The loss functions that we use are inspired by the functional form of negative-index Sobolev norms. For functions $f, h : \mathbb{T}^d \to \mathbb{R}$, where $\mathbb{T}^d$ is the d-dimensional torus $(\mathbb{R}/\mathbb{Z})^d$, the Sobolev norm of order $p = 2$ and index $s < 0$ of their difference can be computed via

$$\|f - h\|_{H^s}^2 = \sum_{\mathbf{k}\in\mathbb{Z}^d} \frac{|\hat{f}(\mathbf{k}) - \hat{h}(\mathbf{k})|^2}{(1 + (2\pi\|\mathbf{k}\|_2)^2)^{-s}}, \tag{5}$$

where $\hat{f}$ and $\hat{h}$ are the Fourier transforms of f and h, respectively. As the norm $\|\mathbf{k}\|_2$ of the wave vector increases, the contribution of the term $|\hat{f}(\mathbf{k}) - \hat{h}(\mathbf{k})|^2$ to the loss diminishes; discrepancies between f and h at small scales are not as important as discrepancies at larger/coarser scales.

We use the Sobolev loss functions to fully connected neural networks with L layers ($\ell = 0, \dots, L-1$) in two ways. Let $\mathbf{h} : \mathbb{R}^D \to \mathbb{R}^M$ be the function to be learned and $s : \mathbb{R} \to \mathbb{R}$ be a monotonically increasing function that defines a scale. Usually we will use the linear function $s(\ell) = \ell$ or the exponential function $s(\ell) = 2^\ell$.

Sobolev loss 1. For each hidden layer ℓ, we define an auxiliary output for that layer denoted by $\mathbf{f}_\ell = \mathbf{C}_\ell \mathbf{z}_\ell$, where $\mathbf{z}_\ell \in \mathbb{R}^{N_\ell}$ are the activation functions for layer ℓ and $\mathbf{C}_\ell \in \mathbb{R}^{M\times N_\ell}$ is a matrix. We add a loss function for each auxiliary output having the form

$$L_\ell(\mathbf{h}, \mathbf{f}_\ell) = \sum_{m=1}^{M} \sum_{\|\mathbf{k}\|_1 = s(\ell)} \frac{\left|\hat{\mathbf{h}}^{(m)}(\mathbf{k}) - \hat{\mathbf{f}}_\ell^{(m)}(\mathbf{k})\right|^2}{(1 + (2\pi\|\mathbf{k}\|_2)^2)^{1/2}}, \tag{6}$$

for $\ell \in \{1, \dots, L\}$ and where $\hat{\mathbf{h}}^{(m)}$ is the Fourier transform of the m-th component function of $\mathbf{h} = (h^{(1)}, \dots, h^{(M)})$. By applying a loss of different scales, $s(\ell)$, at each layer, we are enforcing a more interpretable network. Note that for $\ell = 0$, the denominator of (6) is 1 and by Parseval's identity, the expression is equivalent to the L^2 norm.

Sobolev loss 2. Here, instead of applying pieces of (5) at each layer, we apply it only at the output of the final layer, $\mathbf{f}_{L-1} = \mathbf{C}_{L-1}\mathbf{z}_\ell$:

$$L(\mathbf{h}, \mathbf{f}_{L-1}) = \sum_{m=1}^{M} \sum_{\ell=0}^{L-1} \sum_{\|\mathbf{k}\|_1 = s(\ell)} \frac{|\hat{\mathbf{h}}^{(m)}(\mathbf{k}) - \hat{\mathbf{f}}_{L-1}^{(m)}(\mathbf{k})|^2}{(1 + (2\pi\|\mathbf{k}\|_2)^2)^{1/2}}. \tag{7}$$

We apply the two methods on the multiscale signal $h(x) = x + \sin(2\pi x^4)$ (therefore M=1) on the interval $[0, 2]$. At each point x, we add noise $\eta(x)$ to the signal that is distributed according to

$$\eta \sim \epsilon \left(\max_{x\in[0,2]} h(x) - \min_{x\in[0,2]} h(x) \right) N(0, 1). \tag{8}$$

The noisy signal $\hat{h} = h + \eta$ is used as the data set. Figure 3 shows the performance of the different loss functions using the noisy signal to train. The left column contains the result for pure L^2 loss function, the middle column is Sobolev loss 1, and the third column is Sobolev loss 2. Each row, top to bottom, corresponds to noise levels $\epsilon = 0.05, 0.5$, and 1.0, respectively. The network had 9 hidden layers each containing 20 neurons and a single node output layer. Layers were fully-connected. As can be seen the pure L^2 loss function reconstructs the clean signal h poorly at every noise level, basically reconstructing the average of the signal. Of the two Sobolev loss functions, the first performs the best at reconstructing the clean signal, even in the presence of high amounts of noise. This is likely due to imposing a specific scale for each layer, rather than trying to force the network to try to disentangle the scales at the final layer. For each of the Sobolev loss, the linear scale function, $s(\ell) = \ell$ was used.

Conclusions

In this paper, we have presented results on using the Koopman operator to analyze aspects of neural networks. In order to do this, we consider the training process as a dynamical system on the weights. In particular, we used the spectrum of the Koopman operator to analyze the training process and determine when to terminate training and restart. We have also introduced structured loss functions based on negative Sobolev norms which allow for significant noise rejection when trained on a noisy multi-scale signal.

Acknowledgements

This work was partially supported under DARPA contract HR0011-18-9-0033 and AFOSR contract FA9550-17-C-0012.

References

Budišić, M.; Mohr, R.; and Mezić, I. 2012. Applied Koopmanism. *Chaos* 22(4): 047510.

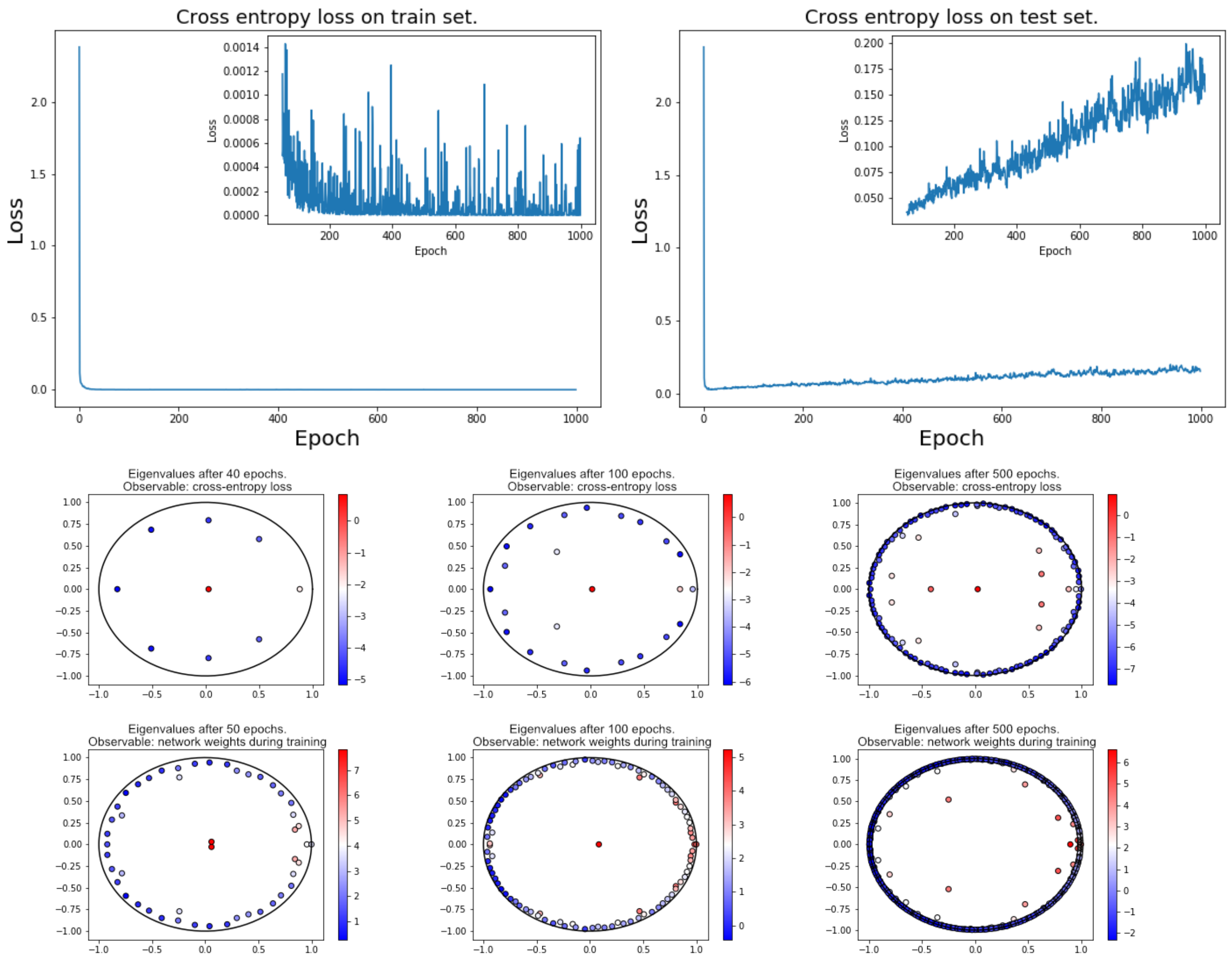

Figure 2: **KMD analysis of network 1 training with Xavier initialization scheme**. **(1st Row left)** Cross entropy loss of the network on the training set. **(1st Row right)** Cross entropy loss on the test set. **(2nd Row)** KMD eigenvalues using the cross entropy on the training set as the observable. Left: computed after 40 epochs. Middle: 100 epochs. Right: 500 epochs. **(3rd Row)** KMD eigenvalues using the weights vectors during training as the observable. Left: computed after 50 epochs. Middle: 100 epochs. Right: 500 epochs.

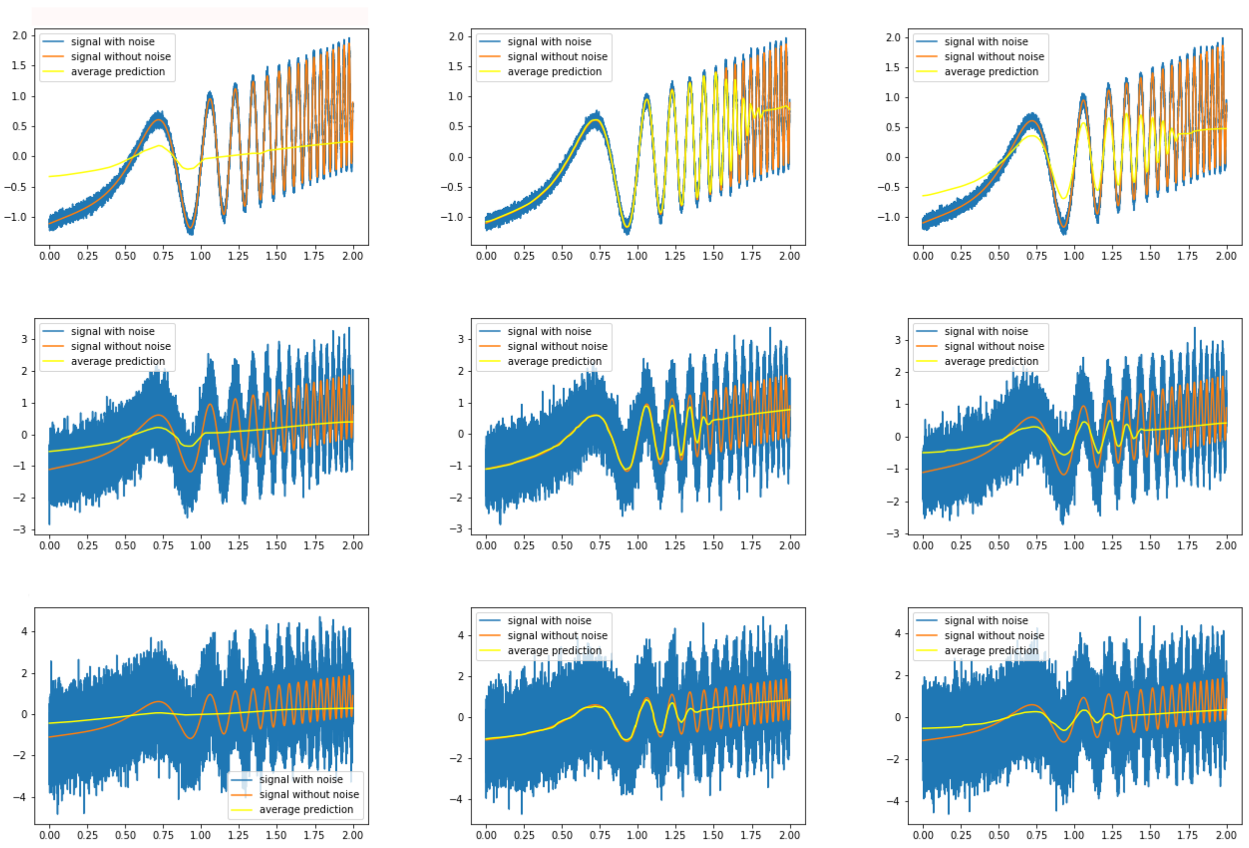

Figure 3: Reconstructing a mulitscale signal polluted by noise. **(Left column)** Pure L^2 loss function. **(Middle column)** Sobolev loss 1. **(Right column)** Sobolev loss 2. Each row, top to bottom, corresponds to noise levels $\epsilon = 0.05, 0.5$, and 1.0, respectively. The noisy signal $\hat{h}$ was used as the data set with the goal of reconstructing the clean signal h. A train/test split of 75/25 was used on $\{\hat{h}(x)\}$.

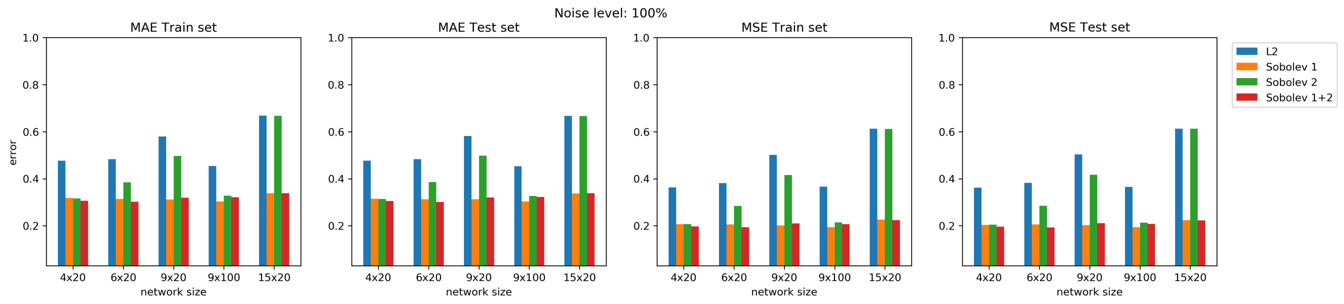

Figure 4: Reconstructing a mulitscale signal polluted by noise ($\epsilon = 1$ in Eq. (8)). The noisy signal $\hat{h} = h + \eta$ was used as the data set with the goal of reconstructing the clean signal h. A train/test split of 75/25 was used on $\{\hat{h}(x)\}$. Sobolev 1 and 2 are as described above. Sobolev $1 + 2$ means that both the Sobolev 1 method and the Sobolev 2 method were applied. It is clear that the best performance is given by the Sobolev 1 method. The same trend continues when using $\epsilon = 0.05$ and 0.5.

Chang, B.; Chen, M.; Haber, E.; and Chi, E. H. 2019. AntisymmetricRNN: A Dynamical System View on Recurrent Neural Networks. *arXiv.org*. arXiv:1902.09689(stat.ML).

Chang, B.; Meng, L.; Haber, E.; Tung, F.; and Begert, D. 2017. Multi-level Residual Networks from Dynamical Systems View. *arXiv.org*. arXiv:1710.10348(stat.ML).

Dietrich, F.; Thiem, T. N.; and Kevrekidis, I. G. 2020. On the Koopman Operator of Algorithms. *SIAM Journal on Applied Dynamical Systems* 19(2): 860–885. doi:10.1137/19M1277059. URL https://doi.org/10.1137/19M1277059.

Drmač, Z.; Mezić, I.; and Mohr, R. 2018. Data Driven Modal Decompositions: Analysis and Enhancements. *SIAM Journal on Scientific Computing* 40(4): A2253–A2285.

Drmač, Z.; Mezić, I.; and Mohr, R. 2019. Data Driven Koopman Spectral Analysis in Vandermonde–Cauchy Form via the DFT: Numerical Method and Theoretical Insights. *SIAM Journal on Scientific Computing* 41(5): A3118–A3151. doi:10.1137/18M1227688. URL https://doi.org/10.1137/18M1227688.

Hemati, M. S.; Williams, M. O.; and Rowley, C. W. 2014. Dynamic mode decomposition for large and streaming datasets. *Physics of Fluids* 26(11): 111701.

Jovanović, M. R.; Schmid, P. J.; and Nichols, J. W. 2012. Low-rank and sparse dynamic mode decomposition. *Center for Turbulence Research, Annual Research Briefs* 139–152.

Jovanović, M. R.; Schmid, P. J.; and Nichols, J. W. 2014. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids* 26(2): 024103.

Mauroy, A.; and Mezić, I. 2016. Global Stability Analysis Using the Eigenfunctions of the Koopman Operator. *IEEE Transactions on Automatic Control* 61(11): 3356–3369.

Mezić, I. 2005. Spectral Properties of Dynamical Systems, Model Reduction and Decompositions. *Nonlinear Dynamics* 41: 309–325.

Mezić, I. 2019. Spectrum of the Koopman operator, spectral expansions in functional spaces, and state-space geometry. *Journal of Nonlinear Science* 1–55.

Mohr, R.; and Mezić, I. 2014. Construction of Eigenfunctions for Scalar-type Operators via Laplace Averages with Connections to the Koopman Operator. *arXiv.org* 1–25.

Rowley, C. W.; Mezić, I.; Bagheri, S.; Schlatter, P.; and Henningson, D. S. 2009. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics* 641: 115–127.

Schmid, P. J.; and Sesterhenn, J. 2008. Dynamic Mode Decomposition of Numerical and Experimental Data. In *Sixty-First Annual Meeting of the APS Division of Fluid Dynamics*. San Antonio, Texas, USA.

Williams, M. O.; Kevrekidis, I. G.; and Rowley, C. W. 2015. A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science* 25(6): 1307–1346.