Representing Neural Network Layers as Linear Operations via Koopman Operator Theory

Nishant Suresh Aswani **Saif Eddin Jabari** **Muhammad Shafique**
Division of Engineering, New York University Abu Dhabi (NYUAD), Abu Dhabi, UAE
Tandon School of Engineering, New York University (NYU), Brooklyn, USA
{nishantaswani, sej7, muhammad.shafique}@nyu.edu

Abstract

The strong performance of simple neural networks is often attributed to their nonlinear activations. However, a linear view of neural networks makes understanding and controlling networks much more approachable. We draw from a dynamical systems view of neural networks, offering a fresh perspective by using Koopman theory and its connections with dynamic mode decomposition (DMD). Together, they offer a framework for linearizing dynamical systems by embedding the system into an appropriate observable space. By reframing a neural network as a dynamical system, we demonstrate that we can replace the nonlinear layer in a pretrained multi-layer perceptron (MLP) with a finite-dimensional linear operator. In addition, we analyze the eigenvalues of DMD and the right singular vectors of SVD, to present evidence that time-delayed coordinates provide a straightforward and highly effective observable space for Koopman theory to linearize a network layer. Consequently, we replace layers of an MLP trained on the Yin-Yang dataset with predictions from a DMD model, achieving a model accuracy of up to 97.3%, compared to the original 98.4%. In addition, we replace layers in an MLP trained on the MNIST dataset, achieving up to 95.8%, compared to the original 97.2% on the test set.

Introduction

Trained neural networks arrive at a decision by applying a series of transformations to their inputs. Each layer plays a specific, albeit often difficult to interpret, role in achieving this goal. At every step, there is a nonlinear mapping to an abstract space, often resulting in a change of dimensionality. Under this view, neural networks are among the simplest instances of a discrete dynamical system [E, 2017]. Given the nature of our optimization tools, we do not arrive at systems with explicit formulae describing the dynamics. Nonetheless, we do find ourselves in a data rich regime, still allowing us to use powerful tools to study complex dynamical systems. In this work, we use Koopman operator theory, a well-established approach dynamical systems to represent nonlinear systems with a linear operator, making it particularly effective due to its data-driven nature.

While an emerging body of work applies Koopman theory to study neural networks, research at this intersection has largely focused on treating the *optimization* procedure as a dynamical system, targeting speedup in training and better understanding weight initialization [Dogra and Redman, 2020, Mohr et al., 2021]. Our work differs in how we draw the link between Koopman theory and neural networks, instead focusing on the *layers of the network* as a composition of dynamical systems. Outside of Koopman theory, this reframing is relatively well researched; a wealth of literature has described neural networks as dynamical systems, with significant effort directed towards developing a framework [E, 2017, Thorpe and van Gennip, 2022] for residual networks, resulting in new architectures [Lu et al., 2020] and training approaches [Chang et al., 2018]. Although our work draws from the same shift in perspective, we direct our attention towards finding linear representations

Preprint.

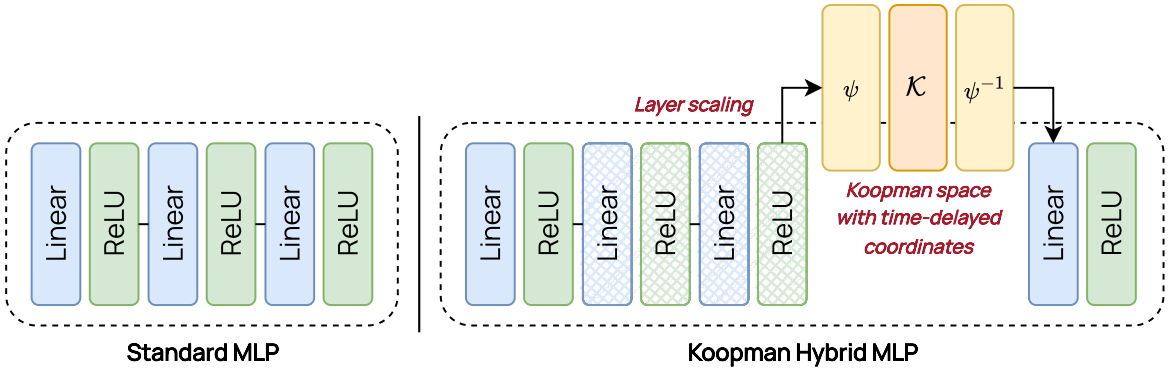

Figure 1: Comparing our Koopman hybrid approach to a standard model. (Left) A typical MLP with compositions of Linear (blue) + ReLU (green) layers; (Right) Our proposed layer approach, which includes scaling a layer (hatched boxes) and "lifting" the activations into the "Koopman space" via delay coordinates embedding (yellow and orange), consequently replacing the original layer with a linear representation to obtain a *Koopman hybrid model*.

of individual nonlinear layers in the neural network, investigating the impact on model performance, and working towards a more interpretable understanding of network layers.

The work most closely aligned with ours [Sugishita et al., 2024] investigates a similar perspective, validating the use of Koopman theory in linearizing neural networks. However, they focus on finding a linear representation of all the intermediate layers, limiting the architectural choices available for their analysis. Moreover, they primarily rely on a monomial embedding as the choice of observable function. In contrast, we successfully demonstrate the use of delay coordinates embedding—a straightforward yet powerful procedure—as an observable function, which is a key ingredient when implementing Koopman theory in practice.

Background

Koopman Theory

In the classical perspective, a discrete dynamical system which evolves the system state $\mathbf{x} \in \mathbb{R}^n$ from a step k to $k+1$ is described as:

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k), \tag{1}$$

where $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ is the nonlinear map. Typically, such systems are analyzed by linear approximation near fixed points, along with other well developed tools in dynamical systems theory.

Koopman operator theory [Koopman, 1931, Koopman and Neumann, 1932] provides an alternative approach to linearizing a nonlinear system by studying its evolution in the observable space, where $\psi : \mathbb{R}^n \to \mathbb{C}^m$ is a function which acts on a system state to generate an observable. Koopman theory proposes a linear, infinite-dimensional operator $\mathcal{K}$ which advances the observable of our system state $\psi(\mathbf{x}_k)$ from one discrete step to the next. The system is described as:

$$\psi(\mathbf{x}_{k+1}) = \mathcal{K}\psi(\mathbf{x}_k), \tag{2}$$

where $\mathbf{x}_k$ is first "lifted" into the observable space and then advanced by $\mathcal{K}$, producing $\psi(\mathbf{x}_{k+1})$, an evolved state in the observable space.

Koopman theory was originally developed in the context of Hamiltonian systems; there the Koopman operator is unitary (i.e. $\mathcal{K}^*\mathcal{K} = I$). Modern interest in Koopman theory was revived by extending it to dissipative systems Mezić [2005] and developing data-driven algorithms Rowley et al. [2009], Schmid [2010] to approximate the Koopman operator in finite-dimensional space, consequently expanding the real world applications. In the following section, we provide a brief walk-through of the data-driven algorithm.

In practice, Koopman analysis requires a finite-dimensional approximation of the operator, which is obtained by restricting the set of observables to a "Koopman-invariant subspace", such that the observable ψ and its advanced image $\mathcal{K}\psi$ lie in the same subspace [Brunton et al., 2021]. Looking towards the eigenvalue problem of a linear operator, $\mathcal{K}\Phi = \Lambda\Phi$, where Φ is a set of eigenfunctions and Λ is a diagonal matrix of corresponding eigenvalues, we identify eigenfunctions as a suitable candidate for observable functions. Upon action by the Koopman operator, the eigenfunctions remain in the original subspace; hence, they are "Koopman invariant." Then, setting our observables (ψ) in the eigenfunction basis as $\psi = \xi\Phi$, where ξ is a collection of coefficients that allow us to linearly combine our eigenfunctions (Φ), we can reinterpret the system. In fact, we can obtain the observables advanced a t number of steps with the equation

$$\psi(\mathbf{x}_k) = \mathcal{K}^t\xi\Phi(\mathbf{x}_0) = \Phi(\mathbf{x}_0)\Lambda^t\xi. \tag{3}$$

Data-Driven Koopman Analysis

Assuming we have access to $t+1$ snapshots of data from steps 0 to t, and our observables lift the state to dimension m, we can build a data matrix $\mathbf{D} \in \mathbb{C}^{m\times t}$ of observables. Here, $\mathbf{D}$ omits the first observable. Each column $\mathbf{D}_i \in \mathbb{C}^m$ is a state in the observable space which can be factorized as $\Phi(\mathbf{x}_0)\Lambda^i\xi$, where only the number of actions i applied by Λ varies between columns. Then, the entire data matrix can also be factorized as

$$\mathbf{D} = \Phi\,\mathrm{diag}(\xi)\,[\mathbf{I}\;\Lambda\cdots\Lambda^t] = \Phi\,\mathrm{diag}(\xi)\,\mathbf{M}. \tag{4}$$

This formulation can be recast to replace $\mathbf{M}$, a Vandermonde matrix of eigenvalues, by looking at its relationship with companion matrices. Vandermonde matrices diagonalize companion matrices in the manner $\mathbf{C} = \mathbf{M}^{-1}\mathrm{diag}(\xi)\mathbf{M}$ [Krake et al., 2022]. The relationship can equivalently be written as $\mathbf{MC} = \mathrm{diag}(\xi)\mathbf{M}$. Introducing this diagonalization to Equation 4, results in

$$\mathbf{D} = \Phi\,\mathrm{diag}(\xi)\,\mathbf{M} = \Phi\,\mathbf{M}\,\mathbf{C} = \mathbf{D}'\mathbf{C}, \tag{5}$$

where the final expression substitutes $\Phi\,\mathbf{M}$ with $\mathbf{D}'$. Notably, the companion matrix has a square structure with a row-shifted diagonal and a nonzero last column:

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & \cdots & 0 & c_0 \\ 1 & 0 & \cdots & 0 & c_1 \\ 0 & 1 & \cdots & 0 & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & c_{n-1} \end{bmatrix}. \tag{6}$$

With the exception of the final column, we know that each column $\mathbf{D}'_i$ is identical to $\mathbf{D}_{i+1}$ because we know the action of $\mathbf{C}$. The final column $\mathbf{D}'_t$, however, is a linear combination of all the previous snapshots. We can formulate the last expression in Equation 5 as a minimization problem to solve for $\mathbf{C}$.

Reformulating the framework as an algorithm results in an early version of the core dynamic mode decomposition (DMD) method. The more recent and standard, core DMD algorithm alleviates computational issues in the described approach. Our brief description largely follows the review by Schmid [Schmid, 2022], which provides a detailed look at developments in the DMD algorithm. Nevertheless, our exposition of the original algorithm emphasizes the use of observed measurements to analyze a nonlinear dynamical system. We build our work atop this data-driven foundation.

Koopman Theory for Neural Networks

Our work adapts the Koopman framework to study trained neural networks by treating them as sequential compositions of nonlinear transformations. Each layer is an individual nonlinear mapping that advances a set of states forward to the next step.

Reformulating Layers. Specifically, we consider trained multi-layer perceptrons (MLPs) as $\mathcal{F} = \mathbf{F}_{\ell-1} \circ \cdots \circ \mathbf{F}_1 \circ \mathbf{F}_0$, where each layer $\mathbf{F}_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$ for $i \in [0, \ell-1] \cap \mathbb{Z}$ consists of a linear operation and a ReLU. We treat the inputs, activations, and outputs as system states denoted by

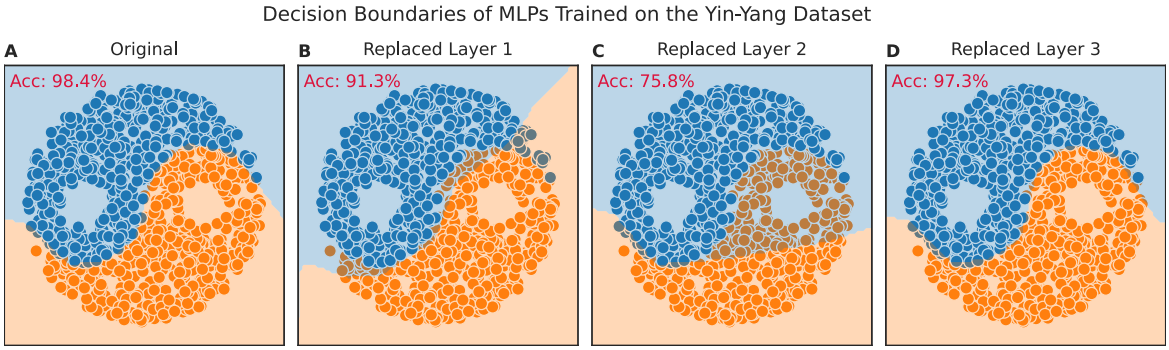

Figure 2: Decision boundaries of the original MLP, and its hybrid variants, trained on the Yin-Yang dataset. We test the models on 1000 samples of the dataset. (A) The original model achieves an accuracy of 98.4%. (B) The MLP where the first hidden layer of size 8×6 is replaced with a DMD model, achieves an accuracy of 91.3%. (C) MLP with second 6×4 hidden layer replaced, achieves an accuracy of 75.8%. (D) MLP with final 4×3 hidden layer replaced, achieves an accuracy of 97.3%.

$\mathbf{x}_i \in \mathbb{R}^{d_i}$, where $d_0 = n$ represents the input dimension and d_ℓ represents the output dimension. The dimensionality of our inputs and activations is directly equal to the number of states in our system. Following the Koopman framework, our approach lifts $\mathbf{x}_i$ to $\psi(\mathbf{x}_i)$ and replaces $\mathbf{F}_i$ with a linear operator $\mathcal{K}_i$. Assuming we apply this approach to the layer $\mathbf{F}_1$, we would represent model inference as

$$\mathbf{x}_l = \mathbf{F}_{\ell-1} \circ \cdots \circ (\psi^{-1} \circ \mathcal{K}_1 \circ \psi) \circ \mathbf{F}_0(\mathbf{x}_0), \tag{7}$$

where ψ^{-1} is the inverse function which returns the observable to the original space.

Dimensionality of the System State. Our goal is to replicate the output of $\mathbf{F}_1(\cdot) \in \mathbb{R}^{d_2}$. But for neural networks, the dimensionality of our system state may vary between layers; it is not guaranteed that $d_2 = d_1$. In our work, we admit two variants of fully-connected layers, ones that do not affect dimensionality ($d_2 = d_1$) and decoder layers that reduce dimensionality ($d_2 < d_1$). In the latter case, we augment the output of $\mathbf{F}_1(\cdot)$ by appending $d_1 - d_2$ negative integers to the system state. Given that our MLPs use the ReLU activation, except for this augmentation, the system never encounters negative integers.

Layer Scaling. Standard literature presents Koopman theory's applications to dynamical systems with inherent time steps, such as fluid flow and financial engineering [Brunton et al., 2016, Schmid, 2022]. Canonically, there are no time steps in model inference: each layer acts on its inputs only once to produce activations for the subsequent layer. However, DMD's performance improves with more samples [Schmid, 2022]. For this purpose, we apply *layer scaling*, which inserts and trains additional layers in an otherwise frozen neural network, without significantly affecting the original model's performance.

Consider $\mathcal{F} = \mathbf{F}_1 \circ \mathbf{F}_0$, a small, trained and frozen two-layer MLP, where we are interested in replacing the final layer $\mathbf{F}_1$. We refer to the activations of $\mathcal{F}$ as $\mathbf{f}_0, \mathbf{f}_1$. Then, we "scale" the network by inserting an additional set of layers $\mathcal{G} = \mathbf{G}_{k-1} \circ \cdots \circ \mathbf{G}_1 \circ \mathbf{G}_0$, where $\mathbf{G}_j : \mathbb{R}^{d_1} \to \mathbb{R}^{d_1}$ for $j \in [0, k-1] \cap \mathbb{Z}$. Then, with $\mathcal{F}$ still frozen, we train $\mathcal{G}$ alone to minimize the Huber loss between the outputs of $\mathbf{F}_1 \circ \mathbf{G}_{k-1}$ and $\mathbf{F}_1 \circ \mathbf{F}_0$. Consequently, we obtain a scaled network $\widetilde{\mathcal{F}} = \mathbf{F}_1 \circ \mathcal{G} \circ \mathbf{F}_0$. Intuitively, the new set of layers $\mathcal{G}$ answer: "what composition of k new layers leads one from $\mathbf{f}_0$ to $\mathbf{f}_1$?" We note that, because of how we train $\mathcal{G}$, ablating $\mathcal{G}$ from $\widetilde{\mathcal{F}}$ precisely returns $\mathcal{F}$. Introducing $\mathcal{G}$ allows us to collect a size $k+2$ set of activations, of uniform dimensionality d_1, to build a dataset $\mathbf{D} \in \mathbb{R}^{d_1 \times (k+2)}$, represented as

$$\mathbf{D} = [\mathbf{f}_0 \quad \mathbf{g}_0 \quad \mathbf{g}_1 \quad \cdots \quad \mathbf{g}_{k-1} \quad \mathbf{f}_1], \tag{8}$$

where $\mathbf{f}_i, \mathbf{g}_i$ represent the activations of the the layers $\mathbf{F}_i, \mathbf{G}_i$, respectively. To summarize, we train a set of new layers $\mathcal{G}$, which produces a set of new activations that interpolate between $\mathbf{f}_0$ and $\mathbf{f}_1$. As a

result, we can generally view our dataset as

$$\mathbf{D} = \begin{bmatrix} \mathbf{d}_0 & \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_{t-1} \end{bmatrix}. \tag{9}$$

For simplicity, we will refer to the dimensions as $\mathbf{D} \in \mathbb{R}^{d \times t}$.

Multiple Trajectories. From above, each input sample produces a "trajectory" of length t. We use r total trajectories and include them by appending them together, such that we have $\mathbf{D} \in \mathbb{R}^{d \times rt}$. Hence, r becomes a parameter in our model, controlling the number of samples we use to fit the DMD model.

Delay Coordinates as Observables. So far, we have elided a discussion of selecting an observable function. For neural networks, the number of states, equivalent to the number of rows in $\mathbf{D}$, corresponds to the input dimension of the layer we replace. Our analysis benefits from selecting an observable function which results in a higher-dimensional set of states. To achieve this, delay coordinates embedding, or Hankelization, is a popular choice, even when complete measurement data is available [Kamb et al., 2020]. Delay coordinates embedding lifts the state by augmenting it with past history, resulting in a Hankel matrix

$$\mathcal{H}_h\mathbf{D} = \begin{bmatrix} \mathbf{d}_0 & \cdots & \mathbf{d}_{rt-h-1} \\ \mathbf{d}_1 & \cdots & \mathbf{d}_{rt-h} \\ \vdots & \ddots & \vdots \\ \mathbf{d}_h & \cdots & \mathbf{d}_{rt-1} \end{bmatrix}, \tag{10}$$

where $\mathcal{H}_h\mathbf{D} \in \mathbb{R}^{(h+1)d \times (rt-h)}$. Each row vector is a subseries of the original timeseries. Hankelization is a simple method with a single hyperparameter, the delay parameter h, which determines the length of the subseries. Finally, we apply DMD to $\mathcal{H}_h\mathbf{D}$, resulting in a fitted DMD model. In the case of prediction, by convention, a delay parameter of h requires providing $h+1$ steps to the model. Given a trajectory of size t, we can predict $t-(h+1)$ steps. If we are interested in predicting just one step, then considering the equality $t-(h+1)=1$, we can set $h_{max} = t-2$ as the maximum delay parameter.

Figure 1 provides an overview of our approach. In summary, our approach scales a trained network, builds a trajectory of system states, Hankelizes the trajectory to train a DMD model, and uses the fitted DMD model's outputs in place of the activations from the hidden layer, essentially hybridizing the neural network with the aid of Koopman theory. We refer to the networks as *Koopman hybrid models*.

Hybridizing a Yin-Yang Network

Decision Boundaries of the Yin-Yang Network

Figure 2 illustrates the decision boundaries for the original network and three hybrid networks, each with a different hidden layer replaced by its corresponding DMD model. For this experiment, we fix the delay parameter h =10 and the number of trajectories $r = 1000$. The boundaries reflect that our approach preserves the network's performance to varying degrees, depending on the layer replaced. After scaling the network, but prior to replacement, the changes in the decision boundaries and accuracies compared to the original model are negligible, indicating that the differences in performance must be attributed to the DMD routine, possibly due to inadequate hyperparameters. However, as described earlier, the maximum delay parameter is tethered to the number of steps we can provide to the DMD model, limiting our exploration. Given that we scale with 10 layers, we must set h =10. We further explore these hyperparameters in the next section.

We hypothesize that the decline in performance indicates the complexity of the transformation undertaken by the layer we replace. Figure 2 shows that replacing the penultimate layer has a minimal effect on the decision boundary, suggesting that it is responsible for the simplest transformation in the network's decision-making process, whereas hidden layer 2 is responsible for the most complex.

Hybridizing an MNIST Network

To further evaluate our approach, we extend our analysis to an MLP trained on the MNIST dataset and present additional experiments to explore DMD hyperparameters.

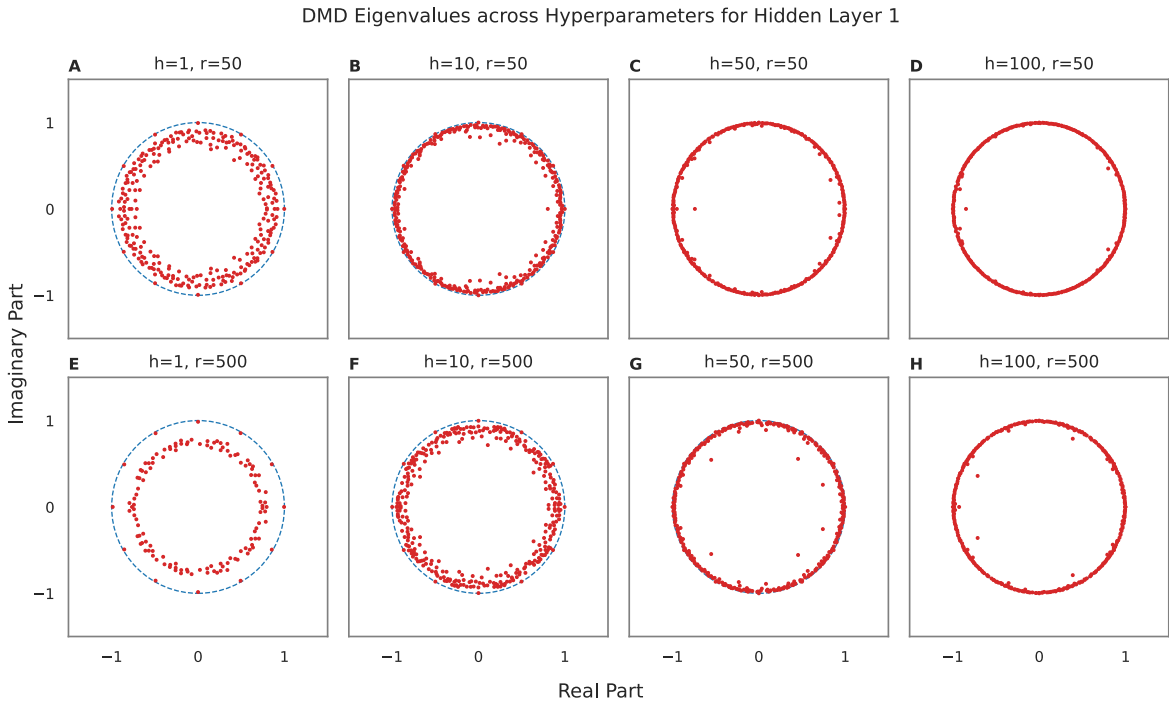

Figure 3: Eigenvalue plots from varying DMD hyperparameters when replacing the first hidden layer in the MNIST network. (A-D) $r = 50$ with $h \in (1, 10, 50, 100)$, where (A) produces a model with 56.48% accuracy and (B) 83.24% accuracy; (E-H) $r = 500$ with $h \in \{1, 10, 50, 100\}$ where (E) produces a model with 63.53% accuracy and (F) 90.21% accuracy.

Exploring the Eigenvalues from Dynamic Mode Decomposition

When Hankelizing and applying DMD to a trajectory matrix, we must select the delay parameter h and the number of trajectories r. Hence, we explore the implications of these hyperparameters on the DMD routine. Figure 3 shows the eigenvalues plotted alongside a unit circle for a few combinations of h, r for the first hidden layer of the network. We note that the eigenvalue plots for the two remaining hidden layers, provided in the appendix, follow a similar pattern.

Figure 3 illustrates that increasing the delay parameter (left to right) increases the number of eigenvalues and eigenmodes (not visualized), implying that there are more “building blocks" to work with. In addition, the eigenvalues tend to be pulled further out, towards the unit circle, suggesting more temporal patterns that do not decay over time. Comparing the top and bottom rows, we see that increasing the number of trajectories affects the size of the eigenvalues. In the first two columns, there are fewer eigenvalues resting on the unit circle when DMD is provided with more trajectories (E-F), compared to when provided with fewer trajectories (A-B). This may also suggest that DMD finds more decaying temporal patterns when exposed to a greater number of trajectories. Intuitively, this is sound, as it would be easier to identify more spurious patterns with a larger sample size.

It is tempting to associate improved model performance with the empirical observation of eigenvalues approaching the unit circle. In Figure 3, the settings from panel E result in an accuracy of 63.53% , while those from panel F yield an accuracy of 90.21% . A similar improvement appears between panel A to panel B, with accuracies of 56.48% and 83.24% , respectively. In both rows, the eigenvalues move outwards. But, comparing panels B and E, the correlation does not necessarily hold. We hypothesize that, while increasing the delay parameter helps the DMD model identify better “building blocks", it must be accompanied by an increase in trajectories to avoid reliance on spurious patterns.

Exploring the Singular Vectors from Singular Value Decomposition

Next, we explore delay embedding on the trajectory matrices, testing multiple options for the delay parameter h. Here, we fix $r = 500$. In Figure 4, we compute the singular value decomposition (SVD),

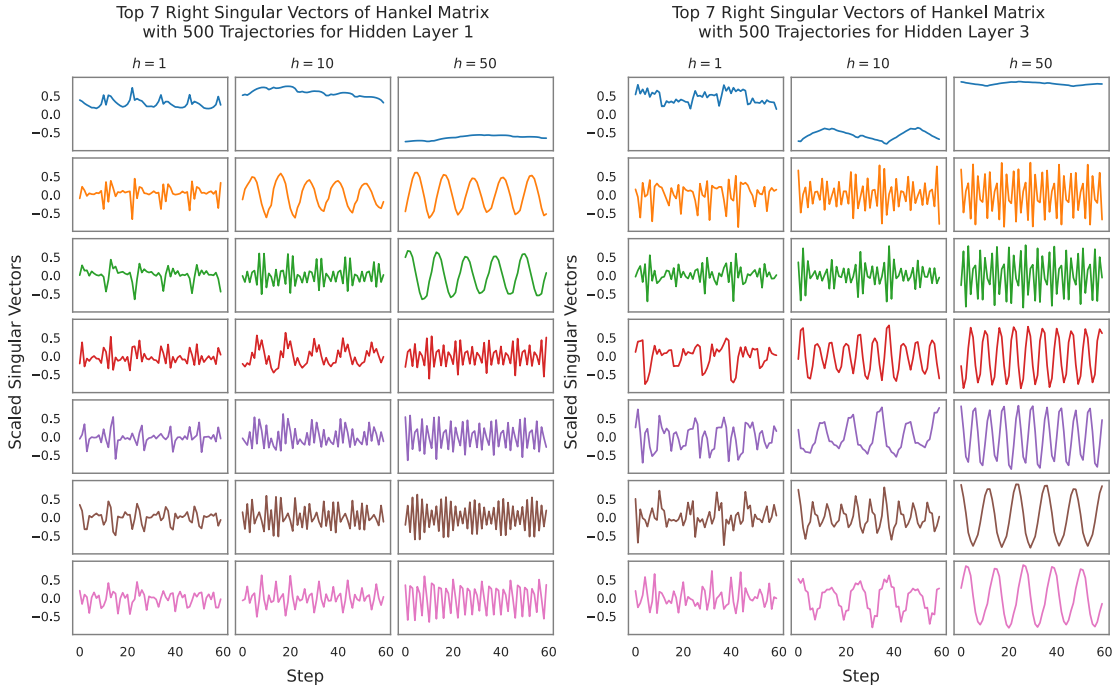

Figure 4: Right singular vector (RSV) plots of the Hankel matrix with various delay parameters. (Left) the Hankel matrix is generated from the first hidden layer with $h \in (1, 10, 50)$; $h = 1$ achieves 66.66% and $h = 10$ achieves 92.04% on the test set. (Right) the Hankel matrix is generated for the final hidden layer with $h \in (1, 10, 50)$; $h = 1$ achieves 28.22% and $h = 10$ achieves 95.80% on the test set.

$\mathcal{H}_h\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, and plot the right singular vectors $\mathbf{V}$. In systems with a single state, both the row and column vectors of the Hankel matrix contain time sub-series, allowing one to analyze either $\mathbf{U}$ or $\mathbf{V}$ for temporal patterns. In our case, because the system has multiple states, the left singular vectors do not exclusively represent temporal patterns. Instead, they are a spatio-temporal mix, making them a poor candidate for inspection. Hence, we plot $\mathbf{V}$ of the trajectory matrices generated for the first (left) and last (right) hidden layers.

For both layers, when $h = 1$ (indicating no Hankelization), the plotted vectors are noisy, complex objects. However, Hankelizing the matrix linearizes the system in the Koopman space, as evidenced by the simpler, more uniform sinusoidal curves that emerge. Interestingly, in the first hidden layer (Figure 4 left), the dominant singular vector has the lowest frequency, whereas in the final hidden layer (Figure 4 right), the dominant singular vectors have higher frequencies. In either case, given the improvement in performance, Hankelizing the trajectory matrix is a successful strategy to linearize our dynamics.

Network Performance After Layer Replacement

Our goal is to replicate the activations of a layer, while maintaining the model's performance. To that end, we insert the learned eigenmodes, eigenvalues, and coefficients (see Equation 3) from DMD in place of the layer of interest and run inference on the MNIST test dataset. While previous discussions have reinforced the selection of a high delay parameter, we are limited to a maximum $h = 10$ for replacement experiments as described earlier.

Figure 5 presents, for a combinations of parameters, the accuracies of all three hybrid MNIST classifiers, achieving up to 95.80% accuracy (replacing hidden layer 3), compared to the original 97.20% . Here, the best performing combination of available hyperparameters is $h = 10, r = 500$, regardless of the layer replaced. While increasing the number of trajectories clearly improves the model, the rate of improvement plateaus, as evidenced by the small difference in performance between $r = 50$ and $r = 500$, compared to $r = 10$ and $r = 50$. Supporting previous discussion, there is a

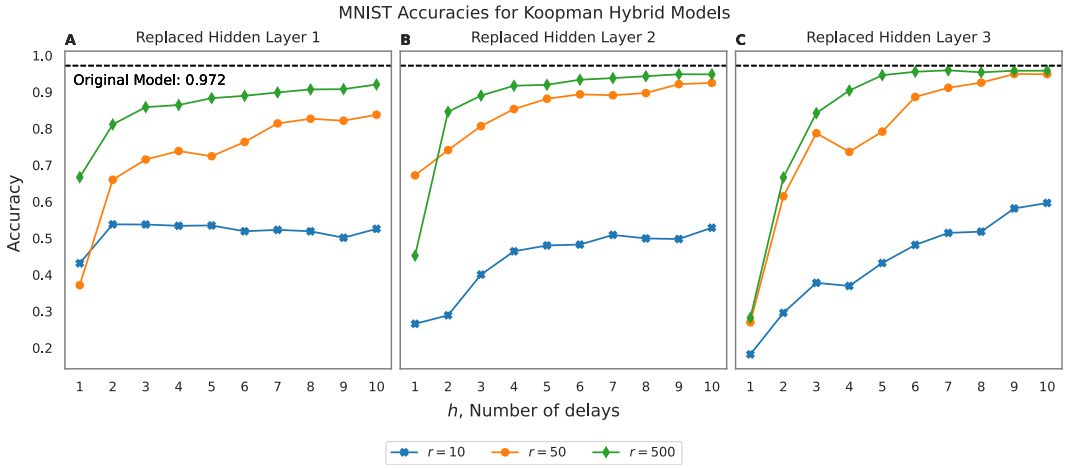

Figure 5: Accuracies on the MNIST test set for various DMD hyperparameters across Koopman hybrid models. The original model achieves 97.20% accuracy. (A) Accuracies after replacing hidden layer 1, where the best performing combination ($h = 10, r = 500$) achieves 92.04%; (B) Replaces hidden layer 2, where ($h = 10, r = 500$) achieves 94.81%; (C) Replaces hidden layer 3, where ($h = 10, r = 500$) achieves 95.80%.

positive trend in accuracy when increasing h, demonstrating that delay coordinates embedding is a viable observable function for linearizing neural network layers. Even for $r = 10$, increasing the delay parameter significantly improves the hybrid model's performance. Increasing either parameter increases the computational cost to fit the DMD model, so it is advisable to select the fewest trajectories with the smallest delay that still produces adequate results. Further, in line with results from Figure 2, we find varying levels of success in hybridizing the model depending on the layer. As before, replacing the final hidden layer is most successful, potentially highlighting a quality of how neural networks process data.

Conclusion

Under the lens of dynamical systems, we demonstrated the first application of Koopman theory and delay-coordinates embedding to linearize individual layers in a neural network. Adapting from extensive literature, we first reframed neural networks as a composition of different nonlinear maps, admitting neural activations as the states of a nonlinear dynamical system. We introduced layer scaling to augment the number of steps available for our states, building a trajectory matrix. Through analyzing the eigenvalues and right singular vectors of our trajectory matrices, we studied the use of delay-coordinates embedding as an observable function. For the Yin-Yang and MNIST datasets, we successfully replaced a fitted DMD model in trained MLPs, retaining significant performance and even visualizing the change in decision boundaries for the former dataset.

Considering our fresh perspective, our work is ripe with several questions to be addressed in future work. Can we further establish a link between what we observe about these components and how successful they will be in replacing a layer? Of particular importance, given that we arrive at a linear operator, is the potential to explicitly understand and manipulate a trained layer. To highlight a few questions here, what do the varying success rates in hybridization across layers suggest about the role of each layer? How do we further develop our approach to decompose a network into its transformations, allowing for a mechanistic understanding? And, with vast literature in linear control to draw from, can we "edit" trained models in the Koopman space? Finally, how do we build upon this framework to reliably accommodate any architectural choice (e.g. convolution, attention)?

Treating the layers of a neural networks as dynamical systems is a powerful framing, especially with the data-driven tendency of Koopman theory. We hope our work demonstrates the potential of this approach in advancing our understanding and control of neural networks.

Acknowledgements

This work was supported by the NYUAD Center for Interacting Urban Networks (CITIES), funded by Tamkeen under the NYUAD Research Institute Award CG001. The views expressed in this article are those of the authors and do not reflect the opinions of CITIES or their funding agencies.

References

S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz. Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control. *PLOS ONE*, 11(2): e0150171, Feb. 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0150171. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0150171. Publisher: Public Library of Science.

S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

B. Chang, L. Meng, E. Haber, F. Tung, and D. Begert. Multi-level Residual Networks from Dynamical Systems View, Feb. 2018. URL http://arxiv.org/abs/1710.10348. arXiv:1710.10348 [cs, stat].

A. S. Dogra and W. Redman. Optimizing Neural Networks via Koopman Operator Theory. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2087–2097. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/169806bb68ccbf5e6f96ddc60c40a044-Paper.pdf.

W. E. A Proposal on Machine Learning via Dynamical Systems. *Communications in Mathematics and Statistics*, 5(1):1–11, Mar. 2017. ISSN 2194-671X. doi: 10.1007/s40304-017-0103-z. URL https://doi.org/10.1007/s40304-017-0103-z.

M. Kamb, E. Kaiser, S. L. Brunton, and J. N. Kutz. Time-Delay Observables for Koopman: Theory and Applications. *SIAM Journal on Applied Dynamical Systems*, 19(2):886–917, Jan. 2020. doi: 10.1137/18M1216572. URL https://epubs.siam.org/doi/10.1137/18M1216572. Publisher: Society for Industrial and Applied Mathematics.

B. O. Koopman. Hamiltonian Systems and Transformation in Hilbert Space. *Proceedings of the National Academy of Sciences of the United States of America*, 17(5):315–318, May 1931. ISSN 0027-8424. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1076052/.

B. O. Koopman and J. v. Neumann. Dynamical Systems of Continuous Spectra. *Proceedings of the National Academy of Sciences of the United States of America*, 18(3):255–263, Mar. 1932. ISSN 0027-8424. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1076203/.

T. Krake, D. Weiskopf, and B. Eberhardt. Dynamic Mode Decomposition: Theory and Data Reconstruction, Feb. 2022. URL http://arxiv.org/abs/1909.10466. arXiv:1909.10466 [cs, math].

L. Kriener, J. Göltz, and M. A. Petrovici. The Yin-Yang dataset. In *Proceedings of the 2022 Annual Neuro-Inspired Computational Elements Conference*, NICE '22, pages 107–111, New York, NY, USA, May 2022. Association for Computing Machinery. ISBN 978-1-4503-9559-5. doi: 10.1145/3517343.3517380. URL https://dl.acm.org/doi/10.1145/3517343.3517380.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998. ISSN 1558-2256. doi: 10.1109/5.726791. URL https://ieeexplore.ieee.org/abstract/document/726791. Conference Name: Proceedings of the IEEE.

Y. Lu, A. Zhong, Q. Li, and B. Dong. Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations, Mar. 2020. URL http://arxiv.org/abs/1710.10121. arXiv:1710.10121 [cs, stat].

I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, 2005.

R. Mohr, M. Fonoberova, I. Manojlović, A. Andrejčuk, Z. Drmač, Y. Kevrekidis, and I. Mezić. Applications of Koopman Mode Analysis to Neural Networks. In *CEUR Workshop Proceedings*, volume 2964, page 182. CEUR-WS, 2021. URL https://pure.johnshopkins.edu/en/publications/applications-of-koopman-mode-analysis-to-neural-networks. ISSN: 1613-0073 Journal Abbreviation: 11th International Workshop on Enterprise Modeling and Information Systems Architectures. EMISA 2021.

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

P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

P. J. Schmid. Dynamic Mode Decomposition and Its Variants. *Annual Review of Fluid Mechanics*, 54(Volume 54, 2022):225–254, Jan. 2022. ISSN 0066-4189, 1545-4479. doi: 10.1146/annurev-fluid-030121-015835. URL https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-030121-015835. Publisher: Annual Reviews.

N. Sugishita, K. Kinjo, and J. Ohkubo. Extraction of nonlinearity in neural networks with Koopman operator. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(7):073401, July 2024. ISSN 1742-5468. doi: 10.1088/1742-5468/ad5713. URL https://dx.doi.org/10.1088/1742-5468/ad5713. Publisher: IOP Publishing.

M. Thorpe and Y. van Gennip. Deep limits of residual neural networks. *Research in the Mathematical Sciences*, 10(1):6, Dec. 2022. ISSN 2197-9847. doi: 10.1007/s40687-022-00370-y. URL https://doi.org/10.1007/s40687-022-00370-y.