A Code

We have released our code at `https://github.com/niniack/koopman_hybrid`.

B Yin-Yang MLP Training Details

Dataset

The Yin-Yang dataset [Kriener et al., 2022] is a two-dimensional, publicly available classification task consisting of three categories: “Yin", “Yang", and “Dot", allowing for easy visualization of the model’s decision boundary. To begin with a problem which would prove straightforward for a neural network, we removed the “Dot" class. Hence, in the modified dataset, each point in the dataset falls in either one of the two sides in the“Yin-Yang” symbol, and we leave the dots on both sides of the symbol empty.

Scaling and Embedding

Each time, before layer replacement, we insert 10 additional Linear+ReLU layers and train for 200 epochs to scale the network. We generate a trajectory by collecting the scaled network’s activations as it runs inference.

Figure 6 provides an example of a trajectory generated by a layer in a trained network and its scaled variant, where we note that the start and end states align between both networks. In addition, the final values of states S7,8 (second column) are an augmented value set to -1. Here, we are scaling an 8×6 Linear + ReLU layer, hence the augmentation affects the last 2 states. The process generates a matrix $D \in \mathbb{R}^{8\times 12}$, where the number of rows is determined by the number of states and the number of columns is determined by the number of scaling layers $+ 2$. Repeating this process allows us to take advantage of more data, in turn producing a better fit DMD model. For this layer, if we use r trajectories, we obtain the data matrix $\mathbf{D} = [D_0\ D_1 \cdots D_{r-1}] \in \mathbb{R}^{8\times 12r}$.

Architecture and Training

Table 1 displays the architecture for the original MLP used to train on the Yin-Yang dataset. The MLP contains three hidden layers (IDs 1-3).

ID	Type	Input Dim	Output Dim
0	Linear + ReLU	2	8
1	Linear + ReLU	8	6
2	Linear + ReLU	6	4
3	Linear + ReLU	4	3
4	Linear	3	2

Table 1: Architecture of the multi-layer perceptron (MLP) model for binary classification on the Yin-Yang dataset.

We trained the original classifier on a single NVIDIA RTX 3080 using PyTorch for 5000 epochs to an accuracy of 98.4%, using the Adam with decoupled weight decay (AdamW) optimizer. We used a learning rate of 5e-3, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay of 1e-2. We train on 2000 randomly generated samples from the dataset, with a seed of 42 and a batch size of 1000 samples.

Hyperparameter Tuning and Scaling

To scale a hidden layer, we insert 10 additional Linear+ReLU layers directly before the layer we are interested in replacing. Before training the new layers, we searched for a learning rate and the AdamW betas, using Ray Tune (v2.34.0). Table 2 presents the search space.

We conducted this search for each hidden layer. Table 3 presents the final hyperparameters, along with the accuracy each scaled model achieved on the dataset.

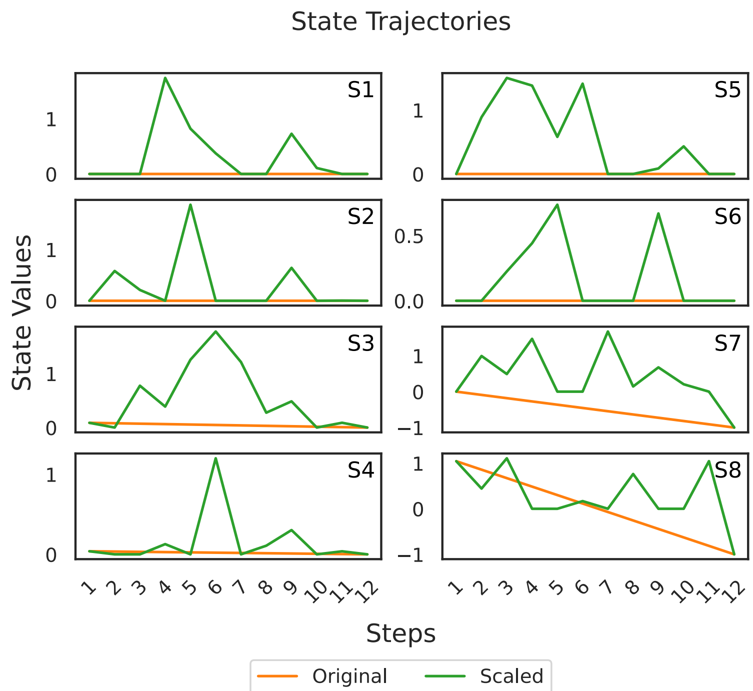

Figure 6: A sample trajectory with 8 states (S1-8) from an original (orange) and scaled (greened) 8×6 Linear + ReLU layer in an MLP trained on the Yin-Yang dataset. States S7,8 are augmented with -1 on the output to allow for a trajectory of system states with uniform dimensionality.

Hyperparameter	**Search Space**
Learning rate	QLogUniform(1e−3, 5e−3, 1e−3)
β_1	QLogUniform(0.2, 0.9, 1e−1)
β_2	QLogUniform(0.5, 0.99, 1e−2)
Weight decay	1e−3
Batch size	512

Table 2: Hyperparameter search space for the Yin-Yang scaled MLPs.

ID	**LR**	**β Values**	**θ Decay**	**Batch**	**Test Acc. (%)**
1	2e-3	[0.8, 0.8]	1e-4	512	98.89
2					98.88
3					98.89

Table 3: Final hyperparameters and accuracy for scaled models trained on the Yin-Yang dataset, where the original model achieves an accuracy of 98.88%.

C MNIST MLP Training Details

Dataset

We also conduct experiments with the MNIST digits dataset [Lecun et al., 1998], which is a 10-way digit classification task containing 60,000 training samples and 10,000 test samples.

Scaling and Embedding.

We scale with 10 Linear+ReLU layers. With the original layers frozen, the new layers are trained on the MNIST training set using the AdamW optimizer (see Appendix for a discussion on training and hyperparameters). As before, we build trajectories by collecting the network's activations as it runs inference. For example, if we analyze the 32×16 Linear + ReLU layer, the augmentation affects the last 16 states and the process generates a matrix $D \in \mathbb{R}^{32\times12}$. If we repeat the process for r samples, we arrive at a data matrix $\mathbf{D} = [D_0\ D_1 \cdots D_{r-1}] \in \mathbb{R}^{32\times12r}$. For this network, we scale and replace all three hidden layers.

Architecture and Training

Table 4 shows the architecture for the MLP, with three hidden layers, used to train on the MNIST dataset.

ID	Type	Input Dim	Output Dim
0	Linear + ReLU	784	256
1	Linear + ReLU	256	128
2	Linear + ReLU	128	64
3	Linear + ReLU	64	32
4	Linear	32	10

Table 4: Architecture of the multi-layer perceptron (MLP) model for 10-way classification on the MNIST dataset.

We trained the MNIST classifer on a single NVIDIA RTX 3080 using PyTorch for 30 epochs to an accuracy of 97.20% . We use AdamW with a learning rate of $1\mathrm{e}{-2}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, a weight decay of $1\mathrm{e}{-1}$, and a batch size of 4096 samples. In addition, we use a learning rate scheduler, which reduces the learning rate by a factor of 0.5 at a loss plateau with a patience of 2 epochs.

Hyperparameter Tuning and Scaling

Hyperparameter	Search Space
Learning rate	QLogUniform($1\mathrm{e}{-3}, 3\mathrm{e}{-3}, 1\mathrm{e}{-3}$)
β_1	QLogUniform($0.6, 0.9, 1\mathrm{e}{-1}$)
β_2	QLogUniform($0.7, 0.99, 1\mathrm{e}{-2}$)
Weight decay	$1\mathrm{e}{-2}$
Batch size	4096

Table 5: Hyperparameter search space for the MNIST scaled MLPs.

For scaling, once again, we insert 10 additional Linear+ReLU layers and conduct a hyperparameter search before training. Table 5 presents the search space and Table 6 presents the final hyperparameters and test accuracies.

ID	LR	β Values	θ Decay	Batch	Test Acc. (%)
1	2e-3	[0.7, 0.7]	1e-2	4096	96.63
2		[0.9, 0.85]			96.71
3	3e-3	[0.9, 0.99]			96.82

Table 6: Final hyperparameters and accuracy for scaled models trained on the Yin-Yang dataset, where the original model achieves an accuracy of 98.88%.