Spectral Distillation: From Nonlinear Dynamics to Linear State-Space Models

Liane Galanti [1 †] **Devan Shah** [1 †] **Shlomo Fortgang** [1] **Elad Hazan** [1]

Abstract

Can nonlinear dynamical systems be learned through a compact linear state-space representation, without directly solving a non-convex system-identification problem? We give a provable pipeline for doing so. Starting from observations of an unknown nonlinear dynamical system, we first learn an implicit spectral predictor using Observation Spectral Filtering (OSF), a convex method that competes with the best linear observer for the system. We then apply spectral-to-LDS distillation to convert this predictor into an explicit recurrent linear dynamical system. Our main theorem shows that the average prediction error of the distilled LDS decomposes into an exponentially-small distillation term and the OSF learning term governed by the Luenberger complexity of the best observer. The guarantee is dimension-free: it depends on observer complexity rather than on the latent dimension needed to represent the nonlinear system. To our knowledge, this yields the first end-to-end provable method for extracting a best-in-hindsight LDS representation of nonlinear dynamics through convex learning followed by provable distillation. Experiments on linear LDS benchmarks and MuJoCo behavior cloning show that the train-then-distill pipeline produces compact LDS predictors that match or outperform directly trained baselines.

1. Introduction

Modern sequence modeling and control hinge on the ability to represent and learn dynamical systems. Linear dynamical systems (LDS) occupy a central role in this landscape: they admit compact state-space representations, efficient simulation, and a rich spectral theory that enables both analysis and algorithm design. However, most real-world systems are inherently nonlinear, and learning either a nonlinear state-space model or even its best linear approximation remains a fundamentally non-convex and fragile problem.

This paper asks a basic question: *can we extract a simple linear dynamical system that faithfully represents a nonlinear one, without ever solving a non-convex system identification problem?*

Our answer is affirmative. We present a provable nonlinear-to-LDS distillation pipeline that converts observations from an unknown nonlinear system into a compact linear dynamical system, using only convex optimization followed by a structured distillation step.

The key idea is to separate learning from representation. Rather than attempting to fit a state-space model directly, we first learn an implicit predictor using Observation Spectral Filtering (OSF), a convex method that operates in a fixed temporal feature space. OSF does not attempt to recover latent states; instead, it competes with the best linear observer that could explain the data, achieving regret bounds that depend only on an intrinsic measure of observability complexity.

We then convert this predictor into an explicit state-space model. Leveraging recent results on spectral representations of dynamical systems, we show how a learned spectral predictor can be distilled into a finite-dimensional LDS whose impulse responses approximate those of the predictor up to exponentially small error. The result is a standard recurrent model with memory and per-step cost depending only on the observation and output dimension, suitable for long-horizon simulation and control.

1.1. Our Contributions

This paper makes the following contributions.

- **A nonlinear-to-LDS distillation pipeline.** We give a two-stage procedure that first learns a nonlinear dynamical system through the convex OSF feature class, and then distills the resulting spectral predictor into an

[†]Equal contribution; author order chosen alphabetically. [1]Princeton University. Correspondence to: Liane Galanti <galanti@princeton.edu>, Devan Shah <devan.shah@princeton.edu>.

The 2nd Workshop on Connecting Low-rank Representations in AI at the 43rd International Conference on Machine Learning, Seoul, South Korea. Copyright 2026 by the author(s).

explicit recurrent LDS.

- **A dimension-free prediction guarantee.** We prove that with $k = \Theta(\log^2 T)$ spectral filters and $h \geq k$ SpectraLDS poles, the distilled LDS satisfies

$$\frac{1}{T}\sum_{t=1}^{T} \|y_t - \hat{y}_t^{\mathrm{LDS}}\|^2 \leq \widetilde{O}\left(\frac{Q_\star^2 \log(Q_\star)\log^7 T}{\sqrt{T}} + \lambda_{\max}(F)^2 \exp\left(-\frac{2k}{\log T}\right)\right).$$

  Thus the error decomposes into an exponentially small distillation term and the OSF learning term governed by the Luenberger complexity $Q_\star$.

- **An explicit compact representation.** Unlike OSF, which is an implicit convolutional predictor, the output of our method is a standard LDS with $O(h \cdot (d_{in} + d_{out}))$ memory and $O(h \cdot (d_{in} + d_{out}))$ per-step simulation cost. The guarantee depends on observer complexity rather than on the potentially enormous dimension of a nonlinear lift or discretization.

- **Empirical validation of train-then-distill.** We test the pipeline on linear LDS benchmarks and MuJoCo behavior cloning. Across these settings, the distilled LDS preserves the behavior of the spectral predictor and often outperforms directly trained LDS baselines.

Together, these results give a principled route from nonlinear observations to compact linear state-space models: learn in a convex spectral space, then distill into an interpretable recurrent LDS.

1.2. Related Work

Linear dynamical system identification. Classical system identification (Ljung, 1999) fits state-space parameters (A, B, C) directly, typically through non-convex optimization. Recent work has obtained non-asymptotic guarantees for LDS estimation under Gaussian noise (Hardt et al., 2018; Simchowitz et al., 2018; Oymak and Ozay, 2019; Sarkar and Rakhlin, 2019; Tsiamis and Pappas, 2019), with sample complexity scaling polynomially in the state dimension and condition number of the system. These results crucially assume the data are generated by a linear system; extending them to nonlinear settings has remained open.

Spectral filtering. Spectral filtering (Hazan et al., 2017; 2018) sidesteps non-convex identification by learning in a fixed convolutional feature space derived from the Hankel matrix. The OSF extension (Dogariu et al., 2025) combines this with a Luenberger-observer analysis to obtain regret guarantees for nonlinear systems through improper learning. SpectraLDS (Shah et al., 2025) converts spectral predictors into recurrent LDS form. Our work composes these ingredients into an end-to-end provable pipeline with a unified guarantee, and we describe these techniques more in the next section.

Koopman operator methods. Koopman theory (Mezić, 2005) represents nonlinear dynamics as a linear operator on an infinite-dimensional observable space, and finite-dimensional approximations underlie methods such as EDMD (Williams et al., 2015) and deep Koopman embeddings (Lusch et al., 2018; Kostic et al., 2022). These approaches typically require choosing a dictionary of observables and offer mostly asymptotic or empirical guarantees. Our pipeline can be viewed in a similar spirit, but the spectral-filter dictionary is universal and the guarantees are non-asymptotic.

Modern state-space sequence models. Structured state-space models such as S4 (Gu et al., 2022), S5 (Smith et al., 2023), the LRU (Orvieto et al., 2023), and Mamba (Gu and Dao, 2023) have shown that linear recurrences with carefully chosen parameterizations can match or exceed Transformer performance on long-sequence tasks. These works focus on architectural and empirical advances rather than provable system-identification guarantees. Our distilled LDS produces a compact recurrence in the same family, but with a provable approximation guarantee for nonlinear data-generating processes.

2. Background

We recall the two ingredients used by our pipeline. Observation Spectral Filtering (OSF) (Dogariu et al., 2025) learns a convex spectral predictor (Hazan et al., 2017) from past inputs and observations. Spectral-to-LDS distillation (Shah et al., 2025) then realizes the learned spectral predictor as an explicit recurrent state-space model.

2.1. Linear systems as convolutional predictors

A controlled linear dynamical system maps inputs $u_t \in \mathbb{R}^{d_u}$ to outputs $y_t \in \mathbb{R}^{d_y}$ through a hidden state $x_t \in \mathbb{R}^{d_x}$:

$$x_{t+1} = Ax_t + Bu_t, \qquad y_t = Cx_t + Du_t. \tag{1}$$

The direct term Du_t can be replaced by an expanded hidden state, and so we often consider $D = 0$. Taking $x_0 = 0$ and suppressing the direct term, the output expands as

$$y_t = \sum_{\ell=0}^{t-1} CA^\ell B\, u_{t-\ell}. \tag{2}$$

Thus an LDS is equivalently described by its impulse response $\{CA^\ell B\}_{\ell \geq 0}$.

This representation exposes the main difficulty in directly fitting LDSs. Long-memory systems require accurately learning high powers of A, and the map from A to the sequence $\{A^\ell\}_{\ell\geq 0}$ is non-convex and poorly conditioned when $\rho(A)$ is close to one. Spectral filtering (Hazan et al., 2017) avoids this difficulty by learning in a fixed convolutional feature space instead of identifying the latent transition matrix.

2.2. Spectral filtering

For $\alpha \in [0,1]$, define the length-L geometric response

$$\mu_L(\alpha) = (1, \alpha, \alpha^2, \ldots, \alpha^{L-1}).$$

Consider the Hankel spectral basis obtained from

$$Z_L = \int_0^1 \mu_L(\alpha)\mu_L(\alpha)^\top \, d\alpha. \tag{3}$$

Let $\phi_1^+, \ldots, \phi_k^+$ be the top k eigenvectors of Z_L, with corresponding eigenvalues $\sigma_1, \ldots, \sigma_k$. The Spectral Filtering algorithm (Hazan et al., 2017) is rooted in the observation that the span of these filters uniformly approximates length-L geometric responses, with approximation error decaying exponentially in $k/\log L$.

For a signal stream s_t, define the spectral features

$$U_{t,j}^{(s)} = \langle \phi_j^+, s_{t-1:t-L} \rangle = \sum_{\ell=1}^{L} \phi_j^+(\ell)\, s_{t-\ell}, j = 1, \ldots, k. \tag{4}$$

A spectral predictor over this stream has the form

$$\hat{y}_t^{\mathrm{SF}} = \sum_{r=1}^{m} R_r s_{t-r} + \sum_{j=1}^{k} M_j U_{t,j}^{(s)}. \tag{5}$$

The temporal filters ϕ_j^+ are fixed in advance, while the readout matrices R_r and M_j are learned. Squared loss is therefore convex in the trainable parameters, even though the resulting predictor can represent long-memory impulse responses for an LDS with PSD matrix A. To extend this result to a symmetric LDS (Hazan et al., 2018), where the transition matrix A is symmetric, we can expand our filter set to include the negative spectral filters $\phi_i^- = [(\phi_i^+)_t \cdot (-1)^t]_{t=0}^{L-1}$. Thus, when we refer to k spectral filters ϕ_i, we refer to $k/2$ positive filters ϕ_i^+ and $k/2$ negative filters ϕ_i^-. For ease of explanation, we redefine the spectral features and predictor to use the combined filter set $\phi_1, \ldots \phi_k$.

2.3. Observation Spectral Filtering

OSF (Dogariu et al., 2025) extends spectral filtering to the input-output prediction setting. At time t, the learner predicts y_t from both the control history and the observation history. The form of the predictor is motivated by the Luenberger observer for a controlled linear system:

$$\hat{x}_{t+1} = A\hat{x}_t + Bu_t + L(y_t - C\hat{x}_t); \quad \hat{y}_t = C\hat{x}_t$$

The observer error evolves according to the closed-loop matrix

$$\widetilde{A} = A - LC.$$

Expanding the observer prediction expresses y_t as a combination of recent inputs, recent observations, and long convolutions over both streams, with convolution kernels generated by powers of $\widetilde{A}$.

The difficulty faced by the best observer is summarized by the Luenberger complexity

$$\begin{aligned} Q(\widetilde{A}) &= \frac{1}{1-\rho(\widetilde{A})} \max\{1, \log \kappa_{\mathrm{diag}}(\widetilde{A})\}, \\ Q_\star &= \inf_L Q(A - LC). \end{aligned} \tag{6}$$

This quantity depends on the spectral gap and diagonalization conditioning of the best closed-loop observer, rather than directly on the hidden-state dimension.

With k spectral filters and finite memory order n, OSF uses the predictor

$$\begin{aligned} \hat{y}_t^{\mathrm{OSF}} = &\sum_{r=1}^{n} J_r u_{t-r} + \sum_{j=1}^{k} \sigma_j^{1/4} M_j U_{t,j}^{(u)} + \\ &\sum_{r=1}^{n} P_r y_{t-r} + \sum_{j=1}^{k} \sigma_j^{1/4} N_j U_{t,j}^{(y)}, \end{aligned} \tag{7}$$

where

$$U_{t,j}^{(u)} = \langle \phi_j, u_{t-1:t-L} \rangle, \qquad U_{t,j}^{(y)} = \langle \phi_j, y_{t-1:t-L} \rangle. \tag{8}$$

Here $J_r, M_j \in \mathbb{R}^{d_y \times d_u}$ and $P_r, N_j \in \mathbb{R}^{d_y \times d_y}$. The parameters (J, M, P, N) are learned over a bounded convex constraint set. Since the features are fixed, the prediction is linear in all trainable parameters, and squared loss remains convex. For nonlinear controlled dynamics,

$$x_{t+1} = f(x_t, u_t), \qquad y_t = g(x_t),$$

OSF is used as an improper learner: it does not recover the latent state or fit a nonlinear model, but learns a predictor directly from filtered histories of inputs and observations, competing with the best finite-dimensional linear observer over the horizon. Under the boundedness and Lipschitz assumptions of (Dogariu et al., 2025), choosing $k = \Theta(\log^2 T)$ yields an average prediction bound of the form

$$\frac{1}{T}\sum_{t=1}^{T} \|y_t - \hat{y}_t^{\mathrm{OSF}}\|^2 \leq \widetilde{O}\left(\frac{Q_\star^2 \log(Q_\star) \log^7 T}{\sqrt{T}}\right), \tag{9}$$

after converting the bounded regret guarantee to squared loss. The important point for our purposes is that the rate is controlled by observer complexity $Q_\star$, not by the dimension of the finite-dimensional linear observer used in the comparison argument.

2.4. Spectral-to-LDS distillation

OSF learns a predictor written in terms of length-L convolutions. Spectral-to-LDS distillation converts those convolutional features into recurrent coordinates.

For $\alpha_i \in [-1, 1]$ and a signal stream s_t, define

$$z_{t+1}^{(s,i)} = \alpha_i z_t^{(s,i)} + s_t. \tag{10}$$

The implicit convolution of this recurrence is

$$(1, \alpha_i, \alpha_i^2, \ldots, \alpha_i^{L-1}).$$

Running h such recurrences in parallel gives a dictionary of geometric filters. SpectraLDS chooses modes $\alpha_1, \ldots, \alpha_h$ and a mixing matrix $F \in \mathbb{R}^{k \times h}$ such that

$$\Phi_{1:k} \approx F\, \mu_L(\alpha_{1:h}), \tag{11}$$

where $\Phi_{1:k} = [\phi_1, \ldots, \phi_k]$ stacks the spectral filters and $\mu_L(\alpha_{1:h}) = [\mu_L(\alpha_1), \ldots, \mu_L(\alpha_h)]$ stacks the geometric responses. Therefore each spectral feature can be replaced by a linear combination of recurrent states:

$$\langle \phi_j, s_{t-1:t-L} \rangle \approx \sum_{i=1}^{h} F(j,i)\, z_t^{(s,i)}. \tag{12}$$

Applying this replacement to both $s_t = u_t$ and $s_t = y_t$ in (7) turns the learned OSF predictor into a recurrent LDS. The finite-memory terms $\sum_{r=1}^{m} J_r u_{t-r}$ and $\sum_{r=1}^{m} P_r y_{t-r}$ are implemented by adding recurrent connections. The two stages are complementary: OSF learns the predictor in a convex spectral feature space, and distillation realizes its temporal filters as recurrent coordinates. The resulting LDS's prediction error decomposes into the OSF learning error and the cost of replacing spectral-filter convolutions by recurrent geometric responses.

3. Nonlinear-to-LDS Distillation

Algorithm 1 states the full pipeline for the controlled input-output setting. We fix horizon T and filter length $L = T$.

Theorem 3.1 (Nonlinear-to-LDS distillation)**.** *Let (f, g) be a nonlinear dynamical system s.t.:*

1. *For all t, the inputs, states, and outputs are bounded: $\|u_t\|, \|x_t\|, \|y_t\| \le R$,*
2. *The dynamics f and observations g are 1-Lipschitz,*
3. *The system has an observable discretized linear lifting (see Appendix B),*

which are the assumptions of (Dogariu et al., 2025). Run Algorithm 1 with filter length $L = T$, $k = \Theta(\log^2 T)$ filters, and h SpectraLDS modes with $k \le h \le 4k$. Then, with probability at least $1 - \delta$,

$$\frac{1}{T}\sum_{t=1}^{T} \|y_t - \hat{y}_t^{\mathrm{LDS}}\|^2 \le \widetilde{O}\left(\frac{Q_\star^2 \log(Q_\star) \log^7 T}{\sqrt{T}} + \lambda_{\max}(F)^2 \exp\left(-\frac{2k}{\log T}\right)\right).$$

where $\widetilde{O}(\cdot)$ suppresses logarithmic factors in T as well as fixed problem-dependent constants including R, D, m, and dimension factors. The distilled predictor uses $O((d_{in} + d_{out})(h + m))$ memory and per-step computation.

The bound has two terms. The OSF learning term $\widetilde{O}(Q_\star^2 \log Q_\star \log^7 T/\sqrt{T})$ depends on the Luenberger complexity of the best linear observer. The distillation term $\lambda_{\max}(F)^2 \exp\left(-\frac{2k}{\log T}\right)$ is the cost of replacing the learned convolutional predictor by a recurrent one. SpectraLDS observes empirically that $\lambda_{\max}(F)$ is bounded and decreasing in h (see Appendix A.2 of (Shah et al., 2025)). The proof of Theorem 3.1 is given in Section A.

4. Experiments: linear systems and behavior cloning

The central theoretical contribution of this work is the first provable method for extracting a best-in-hindsight LDS representation of a nonlinear system via convex optimization. We showcase this contribution in two regimes. First, we study synthetic LDS recovery, where the target class is itself a linear dynamical system. Second, we study MuJoCo (Todorov et al., 2012) behavior cloning, where the data are generated by nonlinear closed-loop expert policies and performance is measured by rollout reward. In these experiments, we consider the STU and OSF without the recurrent connections J_r and P_r (i.e., $n = 0$).

4.1. Learning linear dynamical systems

We first consider prediction from a ground-truth LDS. Each task samples a system

$$x_{t+1} = Ax_t + Bu_t, \qquad y_t = Cx_t$$

and trains a model to predict y_t from a length-m history. We evaluate on two LDS families: *symmetric* LDSs, whose transition matrices have real eigenvalues, and *asymmetric* LDSs, whose non-normal transitions typically have complex eigenvalues and oscillatory responses. We compare the STU $\rightarrow$ SpectraLDS pipeline against directly training a Diagonal LDS, training General LDS, and against training the more-general Linear FIR baselines. For the STU, we use $k = 48$ spectral filters. Each architecture has a $+y$ variant, which is observational and receives past outputs, and a $-y$

Algorithm 1 Nonlinear-to-LDS Distillation

Require: Inputs $u_{1:T}$, observations $y_{1:T}$, filters k, AR order m, LDS modes h, filter length $L = T$

1: **Spectral learning.** Run OSF (Dogariu et al., 2025) with k spectral filters $\phi_1, \dots, \phi_k$ and AR order n on the controlled trajectory $(u_{1:T}, y_{1:T})$ to learn the readout parameters (J_r, M_j, P_r, N_j) and obtain $\hat{y}_t^{\mathrm{OSF}}$ as in (7).

2: **Distillation.** Apply SpectraLDS (Shah et al., 2025) to $\Phi_{1:k} = [\phi_1 \cdots \phi_k]^\top$ to obtain modes $\alpha_1, \dots, \alpha_h \in [-1, 1]$ and $F \in \mathbb{R}^{k \times h}$ such that

$$\Phi_{1:k} \approx F\, \mu_L(\alpha_{1:h}).$$

3: **Recurrent realization.** Initialize $z_1^{(u,i)} = 0 \in \mathbb{R}^{d_u}$ and $z_1^{(y,i)} = 0 \in \mathbb{R}^{d_y}$ for $i = 1, \dots, h$. At prediction time t, the states $z_t^{(u,i)}$ and $z_t^{(y,i)}$ contain only the histories up to time $t-1$. Form

$$\widetilde{U}_{t,j}^{(u)} = \sum_{i=1}^{h} F(j,i) z_t^{(u,i)}, \qquad \widetilde{U}_{t,j}^{(y)} = \sum_{i=1}^{h} F(j,i) z_t^{(y,i)}.$$

4: **Output.** Predict

$$\hat{y}_t^{\mathrm{LDS}} = \sum_{r=1}^{n} J_r u_{t-r} + \sum_{j=1}^{k} \sigma_j^{1/4} M_j \widetilde{U}_{t,j}^{(u)} + \sum_{r=1}^{n} P_r y_{t-r} + \sum_{j=1}^{k} \sigma_j^{1/4} N_j \widetilde{U}_{t,j}^{(y)}.$$

5: **State update.** After time t is processed, update

$$z_{t+1}^{(u,i)} = \alpha_i z_t^{(u,i)} + u_t, \qquad z_{t+1}^{(y,i)} = \alpha_i z_t^{(y,i)} + y_t.$$

The finite-memory AR terms are implemented by shift registers over past inputs and observations. The resulting predictor is an LDS with $O((h+m)(d_u + d_y))$ scalar state variables.

variant, which receives only inputs. Thus, the STU $+y$ variant is equivalent to OSF. For STU, all reported losses are computed after SpectraLDS conversion, so the curves evaluate the Diagonal LDS rather than the intermediate convolutional predictor.

We sweep learning rates over $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$, select the value minimizing the geometric mean of final test MSE over the symmetric and asymmetric tasks, and rerun the selected configuration over five seeds. Full data-generation, initialization, learning-rate sweep results, and optimization details are given in Appendix C.

Long-memory systems. For the most challenging setting, we choose $d_{\mathrm{in}} = d_{\mathrm{out}} = 16$, $d_{\mathrm{state}} = 64$, $m = 512$, and spectral radius $\rho(A) = 0.999$. In this regime the impulse response decays slowly, so accurate prediction requires retaining information far into the past. We compare parameter-matched STU, Diagonal LDS, and General LDS models at roughly 24,600 parameters, together with a Linear FIR baseline, which requires more than ten times as many parameters. We also include STU trained with Online Newton Step (ONS), which we can employ as the prediction is linear in the STU trainable parameters; in contrast, the directly trained LDS baselines remain non-convex in their recurrent parameterizations. We test the $+y$ variant of each architecture.

Table 1 and Figure 1 show that STU $\rightarrow$ SpectraLDS outperforms each baseline under Adam, while STU + ONS improves the error by another three orders of magnitude. The General LDS is more expressive than the Diagonal LDS, but it still does not close the gap to the spectral pipeline, suggesting that the main advantage is optimization rather than final model class. The wall-clock times in Table 1 also show that the spectral model is substantially faster to train than the full General LDS, whose recurrent rollout dominates computation. Linear FIR requires $m \cdot (d_{in} + d_{out}) \cdot d_{out}$ parameters, leading to overparameterization that challenges optimization. Appendix E shows that STU and its distilled SpectraLDS representation have at most 1.5% relative error.

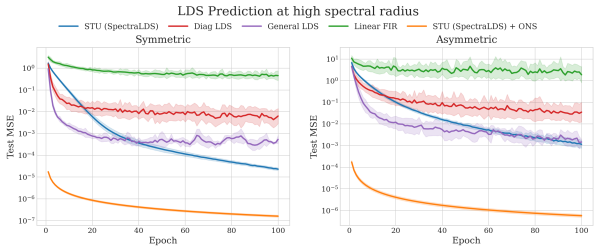

Figure 1. Test MSE versus epoch on the long-memory LDS task. Curves are geometric means over five seeds with geometric-standard-deviation bands, and the y-axis is log-scaled. STU $\rightarrow$ SpectraLDS is competitive under Adam and dominates when optimized with ONS, while Linear FIR fails despite having more than ten times as many parameters.

Table 1. Linear LDS prediction in the long-memory regime ($d_{\text{in}} = d_{\text{out}} = 16$, $d_{\text{state}} = 64$, $m = 512$, $\rho(A) = 0.999$). Final test MSE over five seeds, parameter count, and wall-clock time for one 100-epoch run on an NVIDIA H100. STU rows report the SpectraLDS-distilled recurrent predictor.

Model	# Params	Symmetric	Asymmetric	Time/run
STU	24,576	$2.3 \times 10^{-5} \pm 3.1 \times 10^{-6}$	$1.3 \times 10^{-3} \pm 5.5 \times 10^{-4}$	5.4 s
Diag LDS (param-match)	24,598	$9.2 \times 10^{-3} \pm 8.9 \times 10^{-3}$	$7.2 \times 10^{-2} \pm 1.0 \times 10^{-1}$	13.0 s
General LDS (param-match)	24,705	$6.0 \times 10^{-4} \pm 3.3 \times 10^{-4}$	$1.5 \times 10^{-3} \pm 4.5 \times 10^{-4}$	403.6 s
Linear FIR	262,144	$5.3 \times 10^{-1} \pm 3.4 \times 10^{-1}$	$2.9 \times 10^{0} \pm 2.9 \times 10^{0}$	4.3 s
STU + ONS (η=1, α=1)	24,576	$\mathbf{1.6 \times 10^{-7}} \pm 2.0 \times 10^{-8}$	$\mathbf{5.7 \times 10^{-7}} \pm 8.7 \times 10^{-8}$	121.6 s

Shorter-memory and larger-scale systems. We also test spectral radius $\rho(A) = 0.95$, where transients decay much faster than in the long-memory experiment. We evaluate: $(d_{\text{in}}, d_{\text{out}}, d_{\text{state}}, m) = (4, 4, 8, 256)$ and $(64, 64, 64, 512)$. At each scale we evaluate the $\pm y$ variants of STU $\rightarrow$ SpectraLDS, Diagonal LDS (parameter-matched and $h = 2d_{\text{state}}$), General LDS with $h = 2d_{\text{state}}$, and Linear FIR.

For the large-scale experiment, a parameter-matched General LDS is prohibitively expensive (Table 1), so we use $h = 2d_{\text{state}}$ instead; for the Diagonal LDS, the parameter-matched and $h = 2d_{\text{state}}$ variants achieve similar test MSE (Figure 2). Full parameter counts are in Appendix C.4.

Figure 2 shows the qualitative pattern persists: STU $\rightarrow$ SpectraLDS remains strong on both symmetric and asymmetric dynamics, though surpassed by Linear FIR on the smaller setting and General LDS at the larger one. Without $+y$, the diagonal models (SpectraLDS, Diagonal LDS) struggle on asymmetric systems, where real diagonal recurrences cannot represent complex modes of non-symmetric transitions. Linear FIR is competitive only in the smallest symmetric $-y$ setting, where its parameter count is not yet a bottleneck.

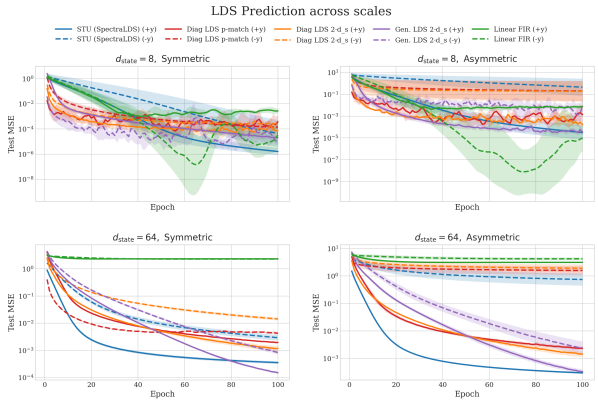

Figure 2. Test MSE versus epoch on linear LDS prediction at spectral radius 0.95. The top row shows the small scale ($d_{\text{in}} = d_{\text{out}} = 4$, $d_{\text{state}} = 8$, $m = 256$), and the bottom row shows the large scale ($d_{\text{in}} = d_{\text{out}} = 64$, $d_{\text{state}} = 64$, $m = 512$). Solid curves are $+y$ variants and dashed curves are $-y$ variants. Curves are geometric means over five seeds with geometric-standard-deviation bands, and the y-axis is log-scaled.

4.2. MuJoCo behavior cloning

We next test the same comparison on nonlinear control data. For each MuJoCo `-v5` task (Todorov et al., 2012), we roll out a deterministic Soft Actor-Critic (SAC) (Haarnoja et al., 2018) expert trained by the Minari team (Younis et al., 2024) for $T = 2{,}000{,}000$ steps and train a student to predict the expert action from a length-m window of past observations and previous actions. The trained student is then deployed in the environment and evaluated by episode reward. Note that training uses teacher-forced next-action prediction on expert trajectories, while evaluation is closed-loop reward in the environment. MuJoCo results therefore serve as empirical evidence that the train-then-distill recipe survives this distribution shift, not as a direct test of our prediction-error bound, which concerns open-loop prediction.

All students share the same encoder, decoder, optimizer, batch size, and training horizon. The only component that changes is the sequence-mixing block:

1. **STU $\rightarrow$ SpectraLDS**: trained as a tensor-dot STU approximation with $k = 16$ spectral filters and deployed after conversion to a Diagonal LDS.
2. **Diagonal LDS, hidden-dimension matched**: a directly trained Diagonal LDS with hidden dimension $h = d$.
3. **Diagonal LDS, parameter-matched**: a directly trained Diagonal LDS whose hidden dimension is chosen to match the STU sequence-block parameter count.

Every reward reported for STU is therefore the reward of the distilled recurrent LDS, not the pre-distillation convolutional model. All models are trained for 100 epochs with Adam at learning rate 3×10^{-4} and batch size 1024, with five seeds per environment-model pair.

The students are intentionally compact: all three student architectures use fewer parameters than the SAC actor deployed by the teacher on every environment. Within the student models, STU and the parameter-matched Diagonal LDS have essentially identical parameter counts by construction, while the hidden-dimension-matched Diagonal LDS

Table 2. Best-checkpoint MuJoCo behavior-cloning reward, re-evaluated over 1,000 episodes (mean ± std over five seeds). STU denotes the SpectraLDS-distilled recurrent model.

Env.	Teacher	STU	DiagLDS (h=d)	DiagLDS (p-match)
Walker2d	6956	**6466** ± 330	5842 ± 555	4766 ± 2045
Hopper	4135	**4038** ± 47	4007 ± 41	3928 ± 127
Ant	5224	5178 ± 100	**5196** ± 168	5103 ± 272
HalfCheetah	11471	**11198** ± 57	10972 ± 231	10838 ± 205
Humanoid	8110	**8021** ± 70	7922 ± 89	7948 ± 90
Pusher	−26.6	−27.4 ± 0.5	**−27.1** ± 0.3	−27.3 ± 0.6

is roughly twenty to twenty-five percent larger. Full environment dimensions, student parameter counts, and training details are given in Appendices D and D.5.

Table 2 and Figure 3 show that STU → SpectraLDS achieves the highest mean reward on four of the six environments: Walker2d, Hopper, HalfCheetah, and Humanoid. On Ant and Pusher, it remains close to the best directly trained Diagonal LDS baseline. The largest separation occurs on Walker2d, where the spectral pipeline exceeds the parameter-matched Diagonal LDS by roughly 1,700 reward. These results support the same conclusion as the LDS recovery experiments: training in the spectral parameterization and then distilling to an LDS is often more reliable than directly training the recurrent LDS, even when the final deployed model class is the same.

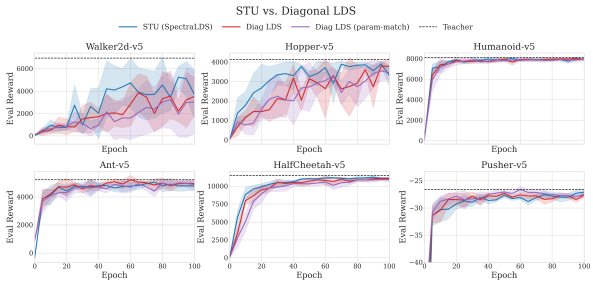

Figure 3. Evaluation reward during behavior-cloning training on six MuJoCo tasks. Curves show mean and standard deviation over five seeds. STU denotes the deployed SpectraLDS-distilled recurrent model.

5. Conclusion

We showed that a convex spectral predictor can be distilled into a recurrent LDS whose prediction error decomposes into a distillation term that decays exponentially in the LDS state dimension and an OSF learning term controlled by the observer condition number $Q_\star$, independent of the latent nonlinear dimension. Empirically, the distilled LDS matches or outperforms directly trained LDS baselines at equal parameter budgets on both linear recovery and MuJoCo behavior cloning, with the advantage appearing to come from the spectral parameterization's convexity rather than from a richer model class.

Several directions remain open. On the theory side, we expect that a refined choice of sampling distribution over the SpectraLDS modes α_i would yield sharper guarantees on $\lambda_{\max}(F)$ in the overparameterized regime $h \gtrsim k$, where the current bound is dominated by the pseudoinverse norm. Additionally, the bound in Theorem 3.1 controls average one-step prediction error under teacher forcing. It does not cover closed-loop rollout error, where errors compound over the horizon — the regime relevant to control and long-horizon simulation. Closing this gap is the most natural extension. Empirically, larger-scale experiments and integration with modern sequence models may establish nonlinear-to-LDS distillation as a building block for long-horizon forecasting and control.

References

E. Dogariu, A. Brahmbhatt, and E. Hazan. Universal learning of nonlinear dynamics. *arXiv preprint arXiv:2508.11990*, 2025.

A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

A. Gu, K. Goel, and C. Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

M. Hardt, T. Ma, and B. Recht. Gradient descent learns linear dynamical systems. In *Journal of Machine Learning Research*, volume 19, pages 1–44, 2018.

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007. doi: 10.1007/s10994-007-5016-8. URL https://doi.org/10.1007/s10994-007-5016-8.

E. Hazan, K. Singh, and C. Zhang. Learning linear dynamical systems via spectral filtering. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/165a59f7cf3b5c4396ba65953d679f17-Abstract.html.

E. Hazan, H. Lee, K. Singh, C. Zhang, and Y. Zhang. Spectral filtering for general linear dynamical systems. In

Advances in Neural Information Processing Systems, volume 31, pages 4634–4643, 2018.

V. Kostic, P. Novelli, A. Maurer, C. Ciliberto, L. Rosasco, and M. Pontil. Learning dynamical systems via koopman operator regression in reproducing kernel hilbert spaces. In *Advances in Neural Information Processing Systems*, 2022.

Y. I. Liu, W. Nguyen, Y. Devre, E. Dogariu, A. Majumdar, and E. Hazan. Flash STU: Fast spectral transform units. *arXiv preprint arXiv:2409.10489*, 2024.

L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 2nd edition, 1999.

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

I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41 (1-3):309–325, 2005.

A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, 2023.

S. Oymak and N. Ozay. Non-asymptotic identification of lti systems from a single trajectory. *American Control Conference*, pages 5655–5661, 2019.

T. Sarkar and A. Rakhlin. Near optimal finite time identification of arbitrary linear dynamical systems. In *International Conference on Machine Learning*, pages 5610–5618, 2019.

D. Shah, S. Fortgang, S. Druchyna, and E. Hazan. SpectraLDS: Provable distillation for linear dynamical systems. *arXiv preprint arXiv:2505.17868*, 2025.

M. Simchowitz, H. Mania, S. Tu, M. I. Jordan, and B. Recht. Learning without mixing: Towards a sharp analysis of linear system identification. In *Conference on Learning Theory*, pages 439–473, 2018.

J. T. H. Smith, A. Warrington, and S. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

A. Tsiamis and G. J. Pappas. Finite sample analysis of stochastic system identification. In *IEEE Conference on Decision and Control*, pages 3648–3654, 2019.

M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A data-driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

O. G. Younis, R. Perez-Vicente, J. U. Balis, W. Dudley, A. Davey, and J. K. Terry. Minari, Sept. 2024. URL https://doi.org/10.5281/zenodo.13767625.

A. Proof of Theorem 3.1

The proof has two ingredients. First, the OSF guarantee gives a mean-squared prediction bound for the learned spectral predictor. Second, replacing the OSF filters by their SpectraLDS approximation perturbs the predictions by an amount controlled by the filter approximation error and the norm of the learned readout. Combining these two bounds gives the result.

Throughout, $\widetilde{O}(\cdot)$ suppresses logarithmic factors in T and fixed problem-dependent constants. We state the proof for scalar outputs for readability; the vector-output case follows by using operator norms for the readout matrices and contributes only the corresponding output-dimension factor.

Step 1: OSF mean-squared error. By Theorem 3.1 of (Dogariu et al., 2025),

$$A_T := \frac{1}{T}\sum_{t=1}^{T} \|y_t - \hat{y}_t^{\mathrm{OSF}}\|^2 \le \widetilde{O}\left(\frac{Q_\star^2 \log(Q_\star)\log^7 T}{\sqrt{T}}\right). \tag{13}$$

Step 2: SpectraLDS filter approximation. Run Algorithm 2 of (Shah et al., 2025) on $\Phi_{1:k}$ to obtain modes $\alpha_1, \ldots, \alpha_h$ and a transformation matrix $F \in \mathbb{R}^{k\times h}$ (denoted M_f in (Shah et al., 2025)). We write

$$\widetilde{\phi}_j = \sum_{i=1}^{h} F(j,i)\,\mu_L(\alpha_i), \qquad j = 1, \ldots, k.$$

Define the rowwise approximation error

$$\eta_{k,h}(L) := \max_{j\le k} \|\phi_j - \widetilde{\phi}_j\|_1. \tag{14}$$

By Theorem 1 of (Shah et al., 2025), on its high-probability event,

$$\eta_{k,h}(L) \;\lesssim\; h\,\lambda_{\max}(F)\,\exp\left(-\frac{k}{\log L}\right), \tag{15}$$

where $\lambda_{\max}(F)$ is the largest singular value of the Moore–Penrose pseudoinverse of the $h \times k$ coefficient matrix produced by Algorithm 2 (called M in (Shah et al., 2025), with $F = M^+$). As shown in Appendix A.2 of (Shah et al., 2025), the product $h\,\lambda_{\max}(F)$ is empirically bounded for the sampling distribution over α_i used by Algorithm 2.

Step 3: Filter error to prediction error. Let $\hat{y}_t^{\mathrm{dist}}$ be the predictor obtained by replacing each ϕ_j in the OSF predictor with $\widetilde{\phi}_j$. As the finite-memory AR terms $\sum_r J_r u_{t-r}$ and $\sum_r P_r y_{t-r}$ in (7) are unchanged by the filter substitution and so do not contribute to the error, we have

$$\begin{aligned}\hat{y}_t^{\mathrm{OSF}} - \hat{y}_t^{\mathrm{dist}} &= \sum_{j=1}^{k} \sigma_j^{1/4} M_j \langle \phi_j - \widetilde{\phi}_j, u_{t-1:t-L}\rangle \\ &\quad + \sum_{j=1}^{k} \sigma_j^{1/4} N_j \langle \phi_j - \widetilde{\phi}_j, y_{t-1:t-L}\rangle.\end{aligned}$$

Using $\|u_t\|, \|y_t\| \le R$ and $\|\sigma_j^{1/4} M_j\|_{\mathrm{op}}, \|\sigma_j^{1/4} N_j\|_{\mathrm{op}} \le D$, we have

$$\left\|\left\langle \phi_j - \widetilde{\phi}_j, u_{t-1:t-L}\right\rangle\right\| \le R\|\phi_j - \widetilde{\phi}_j\|_1 \le R\eta_{k,h}(L),$$

and the same bound holds for the observation-history term. Therefore,

$$\|\hat{y}_t^{\mathrm{OSF}} - \hat{y}_t^{\mathrm{dist}}\| \le 2kDR\,\eta_{k,h}(L).$$

Averaging over time gives the bound

$$\frac{1}{T}\sum_{t=1}^{T} \|\hat{y}_t^{\mathrm{OSF}} - \hat{y}_t^{\mathrm{dist}}\|^2 \le 4k^2 D^2 R^2\,\eta_{k,h}(L)^2. \tag{16}$$

Substituting (15) with $L = T$ yields

$$\frac{1}{T}\sum_{t=1}^{T}\|\hat{y}_t^{\mathrm{OSF}} - \hat{y}_t^{\mathrm{dist}}\|^2 \le \widetilde{O}\left(k^2 D^2 R^2 h^2 \lambda_{\max}(F)^2 \exp\left(-\frac{2k}{\log T}\right)\right). \tag{17}$$

The predictor $\hat{y}_t^{\mathrm{dist}}$ is realized exactly by an LDS: each geometric response $\mu_L(\alpha_i)$ is implemented by a scalar recurrent mode, and the AR terms are implemented by finite shift registers. We denote this exact recurrent realization by $\hat{y}_t^{\mathrm{LDS}}$.

Step 4: Combine. For every t,

$$y_t - \hat{y}_t^{\mathrm{LDS}} = (y_t - \hat{y}_t^{\mathrm{OSF}}) + (\hat{y}_t^{\mathrm{OSF}} - \hat{y}_t^{\mathrm{dist}}).$$

Using

$$\|a+b\|^2 \le 2\|a\|^2 + 2\|b\|^2,$$

together with (13) and (16), gives

$$\frac{1}{T}\sum_{t=1}^{T}\|y_t - \hat{y}_t^{\mathrm{LDS}}\|^2 \le 2A_T + 8k^2 D^2 R^2\, \eta_{k,h}(L)^2.$$

Finally, substituting the SpectraLDS high-probability bound (15) with $L = T$ gives the explicit estimate

$$\frac{1}{T}\sum_{t=1}^{T}\|y_t - \hat{y}_t^{\mathrm{LDS}}\|^2 \le \widetilde{O}\left(\frac{Q_\star^2 \log(Q_\star)\log^7 T}{\sqrt{T}} + k^2 D^2 R^2 h^2 \lambda_{\max}(F)^2 \exp\left(-\frac{2k}{\log T}\right)\right).$$

Since $k = \Theta(\log^2 T)$, the factor k^2 is polylogarithmic in T and is absorbed by $\widetilde{O}(\cdot)$. The constants D and R, together with $h = O(k)$, are also absorbed by $\widetilde{O}(\cdot)$. Thus the same bound can be written in the cleaner form

$$\frac{1}{T}\sum_{t=1}^{T}\|y_t - \hat{y}_t^{\mathrm{LDS}}\|^2 \le \widetilde{O}\left(\frac{Q_\star^2 \log(Q_\star)\log^7 T}{\sqrt{T}} + \lambda_{\max}(F)^2 \exp\left(-\frac{2k}{\log T}\right)\right).$$

This proves the theorem. □

B. Assumption: Observable Discretized Linear Lifting

We recall the discretized lifting assumption used in (Dogariu et al., 2025). For simplicity, we state the autonomous version and consider:

$$x_{t+1} = f(x_t), \qquad y_t = g(x_t),$$

Fix a discretization scale $\varepsilon > 0$, and let $S = \{s^{(1)}, \dots, s^{(N)}\}$ be an ε-net of the radius-R ball in state space. Thus every bounded state x has a representative $\pi(x) \in S$ with

$$\|x - \pi(x)\| \le \varepsilon, \qquad N = |S| \le \left(\frac{2R}{\varepsilon}\right)^{d_x}.$$

Define a finite-state approximation to the nonlinear dynamics by

$$T(s) = \pi(f(s)).$$

Equivalently, define a transition matrix $A' \in \mathbb{R}^{N\times N}$ by

$$A'_{ij} = \mathbf{1}\{T(s^{(j)}) = s^{(i)}\}.$$

The lifted state $z_t \in \mathbb{R}^N$ is the one-hot vector representing the current grid point. Its evolution is linear:

$$z_{t+1} = A' z_t.$$

Define the observation matrix $C' \in \mathbb{R}^{d_Y \times N}$ by setting its jth column to the observation at the corresponding grid point:

$$C' e_j = g(s^{(j)}).$$

The discretized linear lifting is therefore the finite-dimensional LDS

$$z_{t+1} = A' z_t, \qquad y'_t = C' z_t.$$

By Lemma 5.5 of (Dogariu et al., 2025), if f and g are 1-Lipschitz and the trajectory is bounded, then this lifted LDS approximates the nonlinear output sequence over horizon T:

$$\sum_{t \le T} \|y'_t - y_t\| \le \frac{T^2}{2} \varepsilon.$$

Thus, choosing ε sufficiently small produces a high-dimensional LDS whose outputs track the nonlinear trajectory. The dimension $N \le (2R/\varepsilon)^{d_X}$ can be very large, but OSF is an improper learner: its runtime and parameter count do not depend on explicitly constructing or learning this lifted state.

The additional assumption in Theorem 3.1 is that the pair (A', C') is observable. That is, the observability matrix

$$\mathcal{O}(A', C') = \begin{bmatrix} C' \\ C'A' \\ C'(A')^2 \\ \vdots \\ C'(A')^{N-1} \end{bmatrix} \in \mathbb{R}^{N d_Y \times N}$$

has full column rank:

$$\operatorname{rank} \mathcal{O}(A', C') = N.$$

Equivalently, no nonzero lifted state direction is invisible to all future linear observations:

$$C'(A')^\ell v = 0 \quad \text{for all } \ell = 0, \dots, N-1 \quad \Longrightarrow \quad v = 0.$$

This observability condition is used only in the comparison argument. It ensures that a Luenberger observer can be formed for the lifted LDS and that the observer complexity $Q_\star$ appearing in the OSF bound is finite.

C. Linear LDS data-generation and training protocol

This appendix documents the data-generation procedure, predictor initializations, and training protocol used in the linear-LDS experiments of Section 4.1.

C.1. Data generation

Each ground-truth LDS (A, B, C) is sampled once per (seed, task) pair.

- **Symmetric** A: draw $Q \in \mathbb{R}^{d_{\text{state}} \times d_{\text{state}}}$ from the QR decomposition of a Gaussian matrix and eigenvalues $\lambda_i \overset{\text{i.i.d.}}{\sim} \text{Uniform}(-\rho, \rho)$; set $A = Q \operatorname{diag}(\lambda) Q^\top$.
- **Asymmetric** A: draw $\tilde{A}_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1/d_{\text{state}})$ and rescale to the target spectral radius via $A = \rho \tilde{A} / \max_i |\lambda_i(\tilde{A})|$. The eigenvalues are typically complex, producing oscillatory dynamics.
- $B \in \mathbb{R}^{d_{\text{state}} \times d_{\text{in}}}$ and $C \in \mathbb{R}^{d_{\text{out}} \times d_{\text{state}}}$ have entries $\mathcal{N}(0, 1/d_{\text{in}})$ and $\mathcal{N}(0, 1/d_{\text{state}})$, respectively.

Inputs are sampled i.i.d. as $u_t \sim \mathcal{N}(0, I_{d_{\text{in}}})$, and the state evolves from $x_0 = 0$ via $x_{t+1} = Ax_t + Bu_t$ with output $y_t = Cx_t$. We generate $T_{\text{train}} = 10{,}000$ training time steps and $T_{\text{test}} = 2{,}000$ test time steps; the test inputs are drawn with a different RNG seed from the same distribution, so train and test share the same dynamics but no shared input samples.

The predictor sees a length-m sliding window of past observations and predicts the current output. From a trajectory of length T this yields $N = T - m$ examples; the input at index i is $(u_{i+1}, \dots, u_{i+m})$ together with, in the $+y$ variant, the corresponding past outputs $(y_i, \dots, y_{i+m-1})$, and the target is y_{i+m}. So the training set contains $N_{\text{tr}} = T_{\text{train}} - m$ samples and the held-out test set contains $T_{\text{test}} - m$ samples.

C.2. Predictor initializations

All predictors are strictly linear; the only nonlinearity in any of the parameterizations is the $\tanh$ on Diagonal LDS eigenvalues. We use default PyTorch dtypes (float32 throughout training) and the random initialization given here, applied with the same seed used for the ground-truth LDS so that any seed-to-seed variance reflects a single jointly resampled configuration of data and model.

- **STU** (with k spectral filters): the spectral filters $\phi \in \mathbb{R}^{m\times k}$ are computed deterministically from the $m \times m$ Hankel matrix and held fixed. The mixer $M \in \mathbb{R}^{k\times d_{\text{in}}\times d_{\text{out}}}$ is initialized i.i.d. $\mathcal{N}(0, 1/(k \cdot d_{\text{in}}))$. For the experiments with $m = 512$, we choose SpectraLDS $h = 160$, and otherwise we choose $h = 120$.

- **Diagonal LDS** with hidden dimension h: the raw eigenvalue parameter is $\tilde{\lambda} \sim \text{Uniform}(-2, 2)^h$, and the trained eigenvalues are $\lambda = \tanh(\tilde{\lambda}) \cdot 0.999$, which keeps every $\lambda_i \in (-0.999, 0.999)$ for the entire training run and so guarantees stability without needing projection or clipping. The input matrix $B \in \mathbb{R}^{h\times d_{\text{in}}}$ has entries $\mathcal{N}(0, 1/d_{\text{in}})$ and the output matrix $C \in \mathbb{R}^{d_{\text{out}}\times h}$ has entries $\mathcal{N}(0, 1/h)$. We train the Diagonal LDS model convolutionally for maximal efficiency.

- **General LDS** with hidden dimension h: $A \in \mathbb{R}^{h\times h}$ is drawn from PyTorch's `nn.init.orthogonal_` and then rescaled in-place by $\rho_{\text{init}}/\rho(A)$ with $\rho_{\text{init}} = 0.95$ so that $\rho(A) = 0.95$ at initialization. After this, A is an unconstrained `nn.Parameter`; we do not project, clip, or otherwise normalize A during training, and rely only on the per-step gradient-norm clip described below to bound updates. The input and readout matrices use the same scheme as the Diagonal LDS.

- **Linear FIR**: the per-tap kernel $W \in \mathbb{R}^{m\times d_{\text{out}}\times d_{\text{in}}^{(\pm y)}}$ has entries $\mathcal{N}(0, 1/(m\, d_{\text{in}}^{(\pm y)}))$, where $d_{\text{in}}^{(\pm y)} = d_{\text{in}} + d_{\text{out}}$ in the $+y$ variant and d_{in} in the $-y$ variant. The factor of m in the variance is necessary to keep the predicted output's variance $O(1)$ at initialization; without it, the m-tap sum makes early-epoch predictions blow up by roughly $\sqrt{m}$ and Adam's first updates are catastrophically large.

C.3. Optimization and hyperparameter sweep

First-order baseline (Adam). All Adam-trained models are run for 100 epochs at batch size 256 on the small-scale and high-spectral-radius experiments and batch size 64 at the large scale (where the longer windows and larger hidden dimensions make the larger batch run out of memory). After each backward pass we apply `torch.nn.utils.clip_grad_norm_` with max norm `1.0`. Adam's other hyperparameters are PyTorch defaults ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, weight decay 0). For each (model, $\pm y$) we sweep the learning rate at seed 0 over the grid $\eta_{\text{Adam}} \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$, select the value minimizing the geometric mean of final-epoch test MSE across the symmetric and asymmetric tasks, and rerun at seeds $\{0, 1, 2, 3, 4\}$ with that LR.

Second-order learner for STU (ONS, used in the high-spectral-radius experiment). The STU is linear in its parameters. Collecting the spectral features into the flattened vector $g \in \mathbb{R}^{k\cdot d_{\text{in}}^{(\pm y)}}$ where $g = \text{vec}(g_{full})$ for $g_{full}[n, i] = \sum_t x_{\text{rev}}[t, i]\, \phi[t, n]$, the predictor reduces to $\hat{y} = M^\top g$ for a flattened mixer M. Thus, for MSE loss, Online Newton Step (Hazan et al., 2007) attains $O(\log T)$ regret. We use the standard ONS update

$$M \leftarrow M - \eta\, A_t^{-1}\, g_t\, (M^\top g_t - y_t)^\top,$$

with the inverse Hessian estimate maintained by Sherman–Morrison from $A_t = \alpha I + \sum_{s\le t} g_s g_s^\top$. We chose $\eta = 1$ and fixed $\alpha = 1$ after observing instability with our earlier $\alpha = 1 \times 10^{-3}$ default. Thus the final choice is $(\eta, \alpha) = (1, 1)$.

C.4. Per-experiment dimensions

The two cross-scale settings and the high-spectral-radius setting share the protocol above; only the dimensions differ.

In all settings the training set has $N_{\text{tr}} = T_{\text{train}} - m$ samples and the test set has $T_{\text{test}} - m$ samples; with $T_{\text{train}} = 10{,}000$ and $T_{\text{test}} = 2{,}000$, the longest-window cases ($m = 512$: cross-scale large and high spectral radius) leave $9{,}488$ training windows and $1{,}488$ test windows.

Table 3. Dimensions used in each experiment. The number of spectral filters k is fixed at 48 throughout; for the parameter-matched Diagonal and General LDS variants, the hidden dimension h is chosen so that the predictor matches STU's parameter count ($k(d_{\text{in}}^{(\pm y)})\, d_{\text{out}}$) at each scale.

Experiment	d_{in}	d_{out}	d_{state}	m	ρ	Batch
Cross-scale, small	4	4	8	256	0.95	256
Cross-scale, large	64	64	64	512	0.95	64
High spectral radius	16	16	64	512	0.999	256

C.5. Hyperparameter sweep

In this section, we include the results of the learning rate hyperparameter sweep for the learning linear dynamical systems experiments in Section 4.

Table 4. Learning-rate sweep at spectral radius $\rho = 0.999$ ($d_{\text{in}} = d_{\text{out}} = 16$, $d_{\text{state}} = 64$, $m = 512$, 100 epochs, seed 0). Final test MSE on symmetric and asymmetric tasks. For STU+ONS, we fixed $\eta = 1$ and tested $\alpha \in \{1, 10^{-3}\}$, choosing $\alpha = 1$.

Method	LR / η	Sym. MSE	Asym. MSE
STU $+y$ (Adam)	10^{-5}	1.48×10^{0}	9.23×10^{0}
	10^{-4}	6.89×10^{-2}	4.25×10^{-1}
	10^{-3}	$\mathbf{2.44 \times 10^{-5}}$	$\mathbf{9.61 \times 10^{-4}}$
	10^{-2}	1.08×10^{-3}	9.57×10^{-3}
	10^{-1}	9.64×10^{-1}	1.51×10^{0}
Diag LDS $+y$ (Adam)	10^{-5}	4.59×10^{-1}	1.99×10^{0}
	10^{-4}	$\mathbf{2.86 \times 10^{-3}}$	1.17×10^{-1}
	10^{-3}	7.71×10^{-3}	$\mathbf{2.10 \times 10^{-2}}$
	10^{-2}	8.23×10^{-3}	8.47×10^{-2}
	10^{-1}	1.37×10^{0}	8.35×10^{0}
General LDS $+y$ (Adam)	10^{-5}	6.35×10^{-1}	3.27×10^{0}
	10^{-4}	8.97×10^{-4}	1.10×10^{-2}
	10^{-3}	$\mathbf{3.73 \times 10^{-4}}$	$\mathbf{2.12 \times 10^{-3}}$
	10^{-2}	6.45×10^{-3}	6.06×10^{-2}
	10^{-1}	NaN	NaN
Linear FIR $+y$ (Adam)	10^{-5}	1.96×10^{0}	5.04×10^{0}
	10^{-4}	6.70×10^{-1}	1.19×10^{0}
	10^{-3}	$\mathbf{2.59 \times 10^{-1}}$	$\mathbf{1.19 \times 10^{0}}$
	10^{-2}	2.64×10^{1}	1.90×10^{2}
	10^{-1}	4.15×10^{3}	2.25×10^{4}

D. MuJoCo behavior-cloning data and training protocol

This appendix documents the teacher source, dataset construction, shared architecture, sequence-mixing-block details, training hyperparameters, and evaluation protocol used in the MuJoCo experiments of Section 4.2.

D.1. Environments

We evaluate the train-then-distill pipeline on six MuJoCo `-v5` tasks spanning a range of locomotion and manipulation difficulties. Table 6 lists each environment alongside its observation and action dimensions, along with the history length m, embedding dimension d, and number of spectral filters k used by the student. For SpectraLDS distillation, we use $h = 80$.

D.2. Teachers and trajectory data

For each of the six environments we distill a single Soft Actor-Critic (Haarnoja et al., 2018) teacher policy trained by the Minari team (Younis et al., 2024).

Table 5. Learning-rate sweep at the small scale ($d_{\text{in}} = d_{\text{out}} = 4$, $d_{\text{state}} = 8$, $m = 256$) and the large scale ($d_{\text{in}} = d_{\text{out}} = d_{\text{state}} = 64$, $m = 512$). We report final test MSE after 100 epochs of training at seed 0 with Adam. Bold marks the best LR per model and scale.

		$d_{\text{state}} = 8$		$d_{\text{state}} = 64$	
Method	LR	Sym	Asym	Sym	Asym
STU $+y$ (Adam)	10^{-1}	2.61×10^{-4}	5.79×10^{-2}	2.72×10^{0}	5.05×10^{0}
	10^{-2}	1.69×10^{-5}	5.72×10^{-7}	2.11×10^{-2}	4.30×10^{-2}
	10^{-3}	$\mathbf{1.08 \times 10^{-6}}$	$\mathbf{8.16 \times 10^{-6}}$	$\mathbf{3.33 \times 10^{-4}}$	$\mathbf{2.84 \times 10^{-4}}$
	10^{-4}	1.14×10^{-1}	1.83×10^{-1}	1.15×10^{-2}	2.29×10^{-2}
	10^{-5}	9.07×10^{-1}	3.07×10^{0}	8.95×10^{-1}	1.66×10^{0}
Diag LDS p-match $+y$ (Adam)	10^{-1}	1.39×10^{-2}	3.07×10^{-3}	1.16×10^{2}	2.71×10^{2}
	10^{-2}	$\mathbf{5.39 \times 10^{-4}}$	$\mathbf{1.90 \times 10^{-4}}$	6.92×10^{-3}	3.36×10^{-2}
	10^{-3}	7.85×10^{-5}	2.03×10^{-3}	4.08×10^{-3}	1.09×10^{-2}
	10^{-4}	6.73×10^{-4}	1.19×10^{-3}	$\mathbf{1.83 \times 10^{-3}}$	$\mathbf{2.42 \times 10^{-3}}$
	10^{-5}	3.35×10^{-1}	1.33×10^{0}	6.89×10^{-2}	1.56×10^{-1}
Diag LDS $2 \cdot d_{\text{state}}$ $+y$ (Adam)	10^{-1}	1.76×10^{-3}	1.11×10^{-2}	4.60×10^{0}	9.13×10^{0}
	10^{-2}	$\mathbf{4.32 \times 10^{-4}}$	$\mathbf{2.21 \times 10^{-4}}$	3.90×10^{-2}	8.17×10^{-2}
	10^{-3}	1.12×10^{-3}	2.07×10^{-3}	$\mathbf{8.64 \times 10^{-4}}$	$\mathbf{1.21 \times 10^{-3}}$
	10^{-4}	3.27×10^{-2}	1.25×10^{-1}	4.01×10^{-3}	1.05×10^{-2}
	10^{-5}	1.99×10^{0}	4.83×10^{0}	8.27×10^{-1}	1.43×10^{0}
Gen. LDS $2 \cdot d_{\text{state}}$ $+y$ (Adam)	10^{-1}	NaN	NaN	NaN	NaN
	10^{-2}	3.30×10^{-6}	7.94×10^{-5}	3.47×10^{-2}	1.52×10^{-1}
	10^{-3}	$\mathbf{7.24 \times 10^{-6}}$	$\mathbf{2.69 \times 10^{-5}}$	1.40×10^{-3}	2.53×10^{-3}
	10^{-4}	4.94×10^{-3}	5.64×10^{-3}	$\mathbf{1.56 \times 10^{-4}}$	$\mathbf{4.15 \times 10^{-4}}$
	10^{-5}	2.97×10^{0}	6.93×10^{0}	3.33×10^{-1}	8.05×10^{-1}
Linear FIR $+y$ (Adam)	10^{-1}	3.38×10^{1}	7.00×10^{1}	1.16×10^{5}	3.38×10^{5}
	10^{-2}	3.33×10^{-1}	8.90×10^{-1}	1.16×10^{3}	3.37×10^{3}
	10^{-3}	$\mathbf{1.18 \times 10^{-3}}$	$\mathbf{8.02 \times 10^{-3}}$	1.37×10^{1}	3.73×10^{1}
	10^{-4}	1.11×10^{-1}	2.60×10^{-1}	2.39×10^{0}	3.67×10^{0}
	10^{-5}	9.13×10^{-1}	2.36×10^{0}	$\mathbf{2.29 \times 10^{0}}$	$\mathbf{3.34 \times 10^{0}}$

Each teacher is rolled out for $T = 2{,}000{,}000$ time steps in its environment using a fixed seed and the deterministic action mode. The resulting trajectory $(o_t, a_t)_{t=0}^{T-1}$ is the only data the student ever sees: no rollouts of the student itself appear in the training distribution. From this trajectory we construct a sliding-window supervised dataset

$$\mathcal{D} \;=\; \big\{\, \big((o_{t-m+1:t},\, a_{t-m:t-1}),\; a_t\big) \;:\; t = m, \ldots, T-1 \,\big\},$$

in which the student observes a length-m window of past observations and previous actions at each time step and is trained, with mean-squared error against the teacher's action a_t, to predict the next action. We do not subsample or shuffle the trajectory, so the entire $T - m$ samples are used as training data for every student.

The "test" set in this experiment is the live environment itself: trained students are deployed deterministically and evaluated on $N_{\text{eval}} = 100$ episodes per checkpoint. We report best-checkpoint reward, defined as the mean episode reward of the checkpoint that achieves the highest mean reward over its $1{,}000$ deployment episodes. We save checkpoints every 5 epochs. Each episode runs to the environment's default time limit or natural termination under the standard `-v5` reward function.

D.3. Shared encoder, head, and action squash

To isolate the effect of the sequence-mixing block from the surrounding architecture, all three students share an identical encoder and decoder architecture with embedding dimension d that depends on the environment, (see Table 6).

- **Encoder** ($\mathbb{R}^{d_o+d_a} \to \mathbb{R}^d$): a two-layer GELU MLP applied independently at each time step. Both hidden layers have width d, no normalization layers, and GELU is applied between the two linear layers. The encoder concatenates the current observation with the previous action before applying the MLP, so each time step in the sliding window contributes a single d-dimensional embedding to the sequence-mixing block's input.

Table 6. The six MuJoCo environments used in our behavior-cloning experiments, listed with their observation dimension d_o, action dimension d_a, history length m, and the used embedding dimension d. Each student receives the concatenation $[o_t, a_{t-1}]$ at each time step, so the input dimension to the encoder is $d_o + d_a$. Environment descriptions are adapted from the Minari documentation (Younis et al., 2024) of the MuJoCo environments (Todorov et al., 2012).

Environment	Description	d_o	d_a	m	d	k
`HalfCheetah-v5`	2D quadruped with the goal of running	17	6	8	64	16
`Hopper-v5`	2D monoped with the goal of hopping	11	3	128	128	16
`Walker2d-v5`	2D biped with the goal of walking	17	6	128	128	16
`Ant-v5`	3D quadruped with the goal of running	105	8	8	64	16
`Humanoid-v5`	3D biped with the goal of walking	348	17	128	128	16
`Pusher-v5`	3D arm with the goal of pushing an object	23	7	128	128	16

- **Sequence-mixing block** ($\mathbb{R}^{m\times d} \to \mathbb{R}^d$): the only component that varies across the three students. It consumes the length-m sequence of encoder embeddings and produces a single d-dimensional summary. The three variants are detailed in Appendix D.4.
- **Decoder** ($\mathbb{R}^d \to \mathbb{R}^{d_a}$): a three-layer GELU MLP with first layer d, second layer $d/2$, and output d_a, with GELU between layers and a final `tanh` normalization and scaling to ensure the student's predictions live in the valid action box.

The only architectural variation across the three students lies in the sequence-mixing block.

D.4. Sequence-mixing-block variants

The three students differ only in their sequence-mixing block.

STU (the train-then-distill variant). At training time, the block is a tensor-dot Spectral Transform Unit (Liu et al., 2024) with $k = 16$ spectral filters. We use the tensor-dot variant for parameter efficiency at $d = 128$; Appendix F compares it against the STU on the linear-LDS task. At evaluation time, the trained STU is converted to a Diagonal LDS via SpectraLDS, and the converted Diagonal LDS is what we deploy. To allow for easier reuse of the published SpectraLDS conversions (Shah et al., 2025), we use the spectral filters computed with $m = 8192$ and $k = 48$ and at 80-dimensions, truncate, and consider the relevant subset of $k = 16$ filters. The STU runs in `float64` precision due to historical reasons, while other models are in the native `float32`. Based on later experimentation, we do not expect this contributes to any differences.

Diagonal LDS, C-matched ($h = d$). A directly-trained recurrence $x_{t+1} = \mathrm{diag}(\lambda)\, x_t + Bu_t,\ y_t = Cx_t$, with $\lambda = \tanh(\tilde{\lambda})\,(1 - 10^{-3})$ to keep eigenvalues strictly inside the unit disk.

Diagonal LDS, parameter-matched. Identical to the C-matched variant in form, but with h chosen so that $h(1 + 2d) = 2kd + d^2$, giving exactly the same parameter count as STU's tensor-dot block. This is the parameter-matched baseline for the train-then-distill comparison.

D.5. Per-student parameter counts

Table 7 reports the total trainable parameters of each student alongside the SAC teacher's parameter count. All three students are smaller than the SAC actor used at inference time on every environment, and roughly an order of magnitude smaller than the full SAC checkpoint (which carries two critic networks and their target copies in addition to the actor). Within the students, STU and the parameter-matched Diagonal LDS have essentially identical parameter counts by construction, while the C-matched Diagonal LDS is approximately twenty to twenty-five percent larger because the $h = d$ choice inflates its sequence-mixing block.

D.6. Optimization and training protocol

All three students are trained for 100 epochs on the $T - m$-sample dataset using Adam at a fixed learning rate of 3×10^{-4} and batch size 1,024. We use PyTorch's default Adam hyperparameters otherwise ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$,

Table 7. Total trainable parameters per student model and the corresponding SAC teacher across the six MuJoCo environments. The "SAC actor" column gives the parameter count of the policy network used at inference time, while "SAC full" adds the critic and target-critic networks carried in the SAC checkpoint and so reflects the size of the full SAC artifact. The SpectraLDS and parameter-matched LDS students use fewer parameters than the SAC actor on every environment, and within the students the C-matched Diagonal LDS is uniformly larger than the STU pipeline by roughly twenty to twenty-five percent.

Environment	STU	DiagLDS (h=d)	DiagLDS (p-match)	SAC actor	SAC full
`Walker2d-v5`	62,150	77,638	62,218	73,484	362,256
`Hopper-v5`	60,803	76,291	60,871	70,406	349,962
`Ant-v5`	22,568	26,216	22,604	97,040	477,972
`HalfCheetah-v5`	16,742	20,390	16,778	73,484	362,256
`Humanoid-v5`	106,641	122,129	106,709	163,874	802,854
`Pusher-v5`	63,111	78,599	63,179	75,534	371,474

weight decay 0). We do not sweep the learning rate in MuJoCo; we use 3×10^{-4} for every (environment, student) pair, and verified on a small set of pilot runs that all three students train stably at this rate. We run five seeds $\{0, 1, 2, 3, 4\}$ per environment-student pair.

Checkpoints are saved every five training epochs and at the end of training. Each checkpoint is evaluated on 100 episodes, and the reward we report in Table 2 is from taking the best checkpoint and re-evaluating it on 1,000 episodes.

E. SpectraLDS distillation fidelity

A key step in our evaluation pipeline is converting the trained STU predictor into a SpectraLDS (diagonal-LDS) predictor. The filter fit is approximate, so we verify here that this approximation does not meaningfully change the test MSE.

Figure 4 overlays the STU test MSE (from the original five-seed runs) and the SpectraLDS-distilled test MSE (from independent five-seed runs on the same seeds), for both the Adam and ONS optimizers on the high-spectral-radius setting ($\rho = 0.999$, $d = 16$, $d_{\text{state}} = 64$, $m = 512$). Each curve is the geometric mean over five seeds with geometric-standard-deviation bands; the solid line is the distilled model and the dashed line is the raw STU.

The two curves are visually indistinguishable in every panel. Quantitatively, the maximum relative error between the raw and distilled test MSEs, taken over all epochs and all seeds, is below 1.5% in all conditions.

F. STU vs. tensor-dot STU

Both STU variants begin from the same intermediate object. Given an input window $x \in \mathbb{R}^{m \times d_{\text{in}}}$ and the fixed spectral filters $\phi \in \mathbb{R}^{m \times k}$, the spectrally-filtered representation of the window is the matrix

$$Z \;=\; \phi^{\top} x \;\in\; \mathbb{R}^{K \times d_{\text{in}}}, \qquad Z_{k,i} \;=\; \langle \phi_{:,k},\, x_{:,i} \rangle.$$

Z has one row per spectral filter and one column per input channel; all of the time-window information that the predictor will use lives in this $k \times d_{\text{in}}$ matrix. Both STU variants then linearly map Z to a d_{out}-dimensional prediction y.

STU. The STU uses an unconstrained linear map from $\mathbb{R}^{k \times d_{\text{in}}}$ to $\mathbb{R}^{d_{\text{out}}}$, parameterized by a tensor $M \in \mathbb{R}^{k \times d_{\text{in}} \times d_{\text{out}}}$:

$$y \;=\; \sum_{k,i} M_{k,i,:}\, Z_{k,i}.$$

Every element of M is independently learned; the map has $k\, d_{\text{in}}\, d_{\text{out}}$ free parameters.

Tensor-dot STU. The tensor-dot variant replaces the single mixer tensor $M \in \mathbb{R}^{k \times d_{\text{in}} \times d_{\text{out}}}$ with two matrices,

$$M_{\text{in}} \in \mathbb{R}^{k \times d_{\text{in}}}, \qquad M_{\text{out}} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}},$$

constraining the mixer to factorize as $M_{k,i,o} = M_{\text{in},k,i} \cdot M_{\text{out},i,o}$. Two benefits follow.

First, the parameter count drops from $k\, d_{\text{in}}\, d_{\text{out}}$ to $k\, d_{\text{in}} + d_{\text{in}}\, d_{\text{out}}$ — at the MuJoCo dimensions $d_{\text{in}} = d_{\text{out}} = 128$ and $k = 16$ this is a $7.5\times$ reduction.

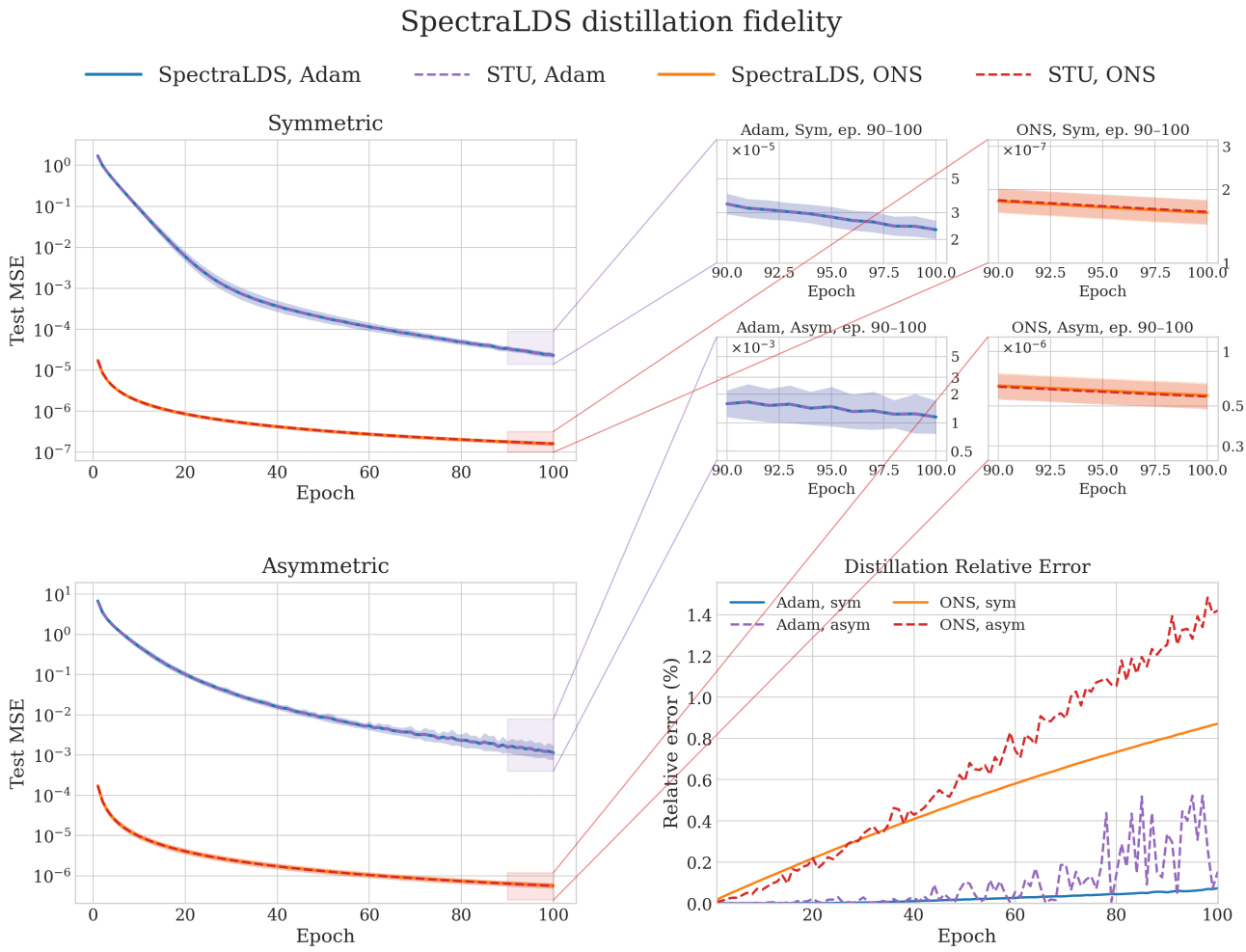

Figure 4. Test MSE versus epoch for the STU predictor (dashed) and the SpectraLDS-distilled predictor (solid), at $\rho = 0.999$. Each curve is the geometric mean over five seeds with geometric-standard-deviation bands; the y-axis is log-scaled. The solid and dashed curves overlap almost exactly in every panel, confirming that SpectraLDS distillation introduces negligible error.

Second, the factorization permits a cheaper convolutional implementation. In the STU, predicting y involves $k \cdot d_{\text{in}}$ distinct filter convolutions. The tensor-dot factorization lets us instead combine M_{in} with the spectral filters into d_{in} per-input-channel filters $f_{:,i} = \sum_k M_{\text{in},k,i}\, \phi_{:,k}$, then perform only d_{in} time-convolutions (one per input channel) before mixing through M_{out}. The number of convolutions thus drops from $k \cdot d_{\text{in}}$ to d_{in}.

However, the tensor-dot STU does have reduced representation capacity, which we analyze in the learning an LDS setting.

Comparison. Figure 5 reports test MSE versus epoch for both STU variants in $+y$ and $-y$ form, at both the small ($d_{\text{state}} = 8$, $m = 256$) and large ($d_{\text{state}} = 64$, $m = 512$) scales. Each curve is the geometric mean over five seeds with geometric-standard-deviation bands, on a log-scaled vertical axis. The STU outperforms the tensor-dot approximation STU in each setting as expected, however, the tensor-dot STU has approximately $k\times$ fewer parameters.

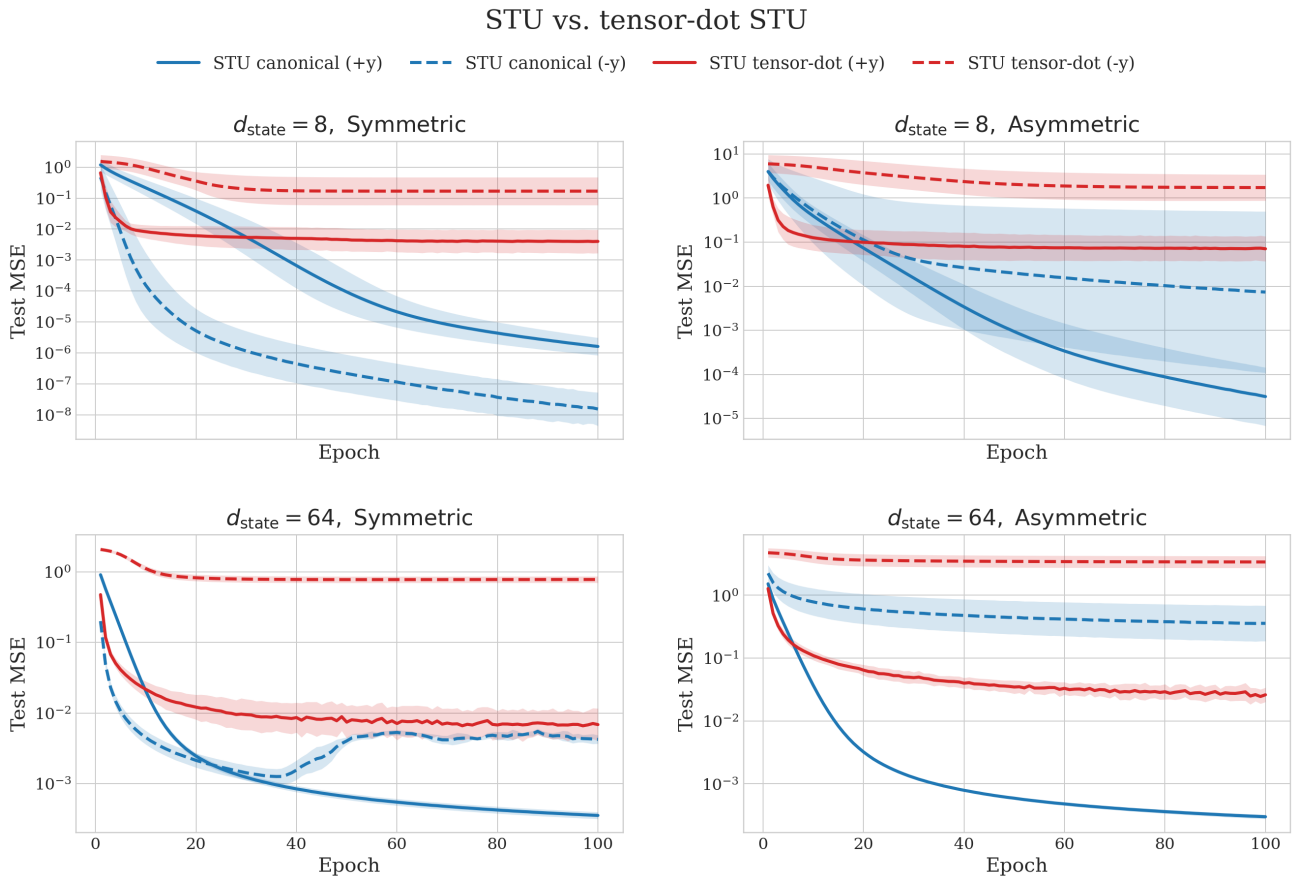

Figure 5. Test MSE versus epoch for STU (blue) and tensor-dot STU (red), both with $k = 48$, and each in $+y$ (solid) and $-y$ (dashed) variants, on the linear-LDS benchmark of Section 4.1. Top row: small scale ($d = 4$, $d_{\text{state}} = 8$, $m = 256$). Bottom row: large scale ($d = 64$, $d_{\text{state}} = 64$, $m = 512$). Curves are geometric means over five seeds with geometric-standard-deviation bands; the y-axis is log-scaled.