
SpectraLDS: Provable Distillation for Linear Dynamical Systems

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 We present the first provable method for identifying symmetric linear dynamical
2 systems (LDS) with accuracy guarantees that are independent of the system’s
3 state dimension or effective memory. Our approach builds upon recent work that
4 represents symmetric LDSs as convolutions learnable via fixed spectral transfor-
5 mations. We show how to invert this representation—recovering an LDS model
6 from its spectral transform—yielding an end-to-end convex optimization procedure.
7 This distillation preserves predictive accuracy while enabling constant-time and
8 constant-space inference per token, independent of sequence length. We evaluate
9 our method, SpectraLDS, as a component in sequence prediction architectures and
10 demonstrate that accuracy is preserved while inference efficiency is improved on
11 tasks such as language modeling.

12 1 Introduction

13 The emergence of attention-based transformer architectures has revolutionized sequence modeling
14 tasks, particularly in natural language processing [34] and large-scale sequence-to-sequence learning
15 [16, 27]. These transformer models rely on the self-attention mechanism, which allows each token in
16 a sequence to attend to every other token, enabling strong contextual understanding. However, this
17 approach suffers from quadratic complexity in sequence length, making it computationally expensive
18 for longer sequences. Recent research has thus explored alternative, more efficient architectures that
19 preserve expressiveness while reducing computational costs for long sequences. Among these are
20 attention-free approaches such as convolution-based or state-space models (SSMs) [9, 2, 22, 26],
21 which can offer sub-quadratic or even near-linear time generation.

22 The most basic SSM, which is the starting point for all aforementioned models, are linear dynamical
23 systems (LDS), a foundational framework for modeling sequential dependencies in control theory,
24 signal processing, and learning theory [20]:

$$x_t = Ax_{t-1} + Bu_t, \quad \hat{y}_t = Cx_t + Du_t. \quad (1)$$

25 Here u_t represents the input sequence, x_t encodes the past state information, and \hat{y}_t approximates
26 the target sequence. By maintaining a fixed latent state, LDSs enable efficient inference and allow
27 for efficient memory utilization. However, gradient-based approaches for learning LDSs suffer from
28 exploding or vanishing gradients, particularly when modeling systems with long-term memory or
29 large hidden state spaces [14].

30 To address these limitations, Agarwal et al. [2] leveraged the spectral filtering method [14] and
31 introduced the Spectral Transform Unit (STU), a convex relaxation that shifts from learning the hidden
32 transition matrix A directly to a reparameterization of the LDS impulse response that learns how
33 input signals convolve with fixed spectral filters. This approach provably preserves the expressiveness
34 of an LDS while making the training problem far more tractable. In practice, it has proven robust for

systems requiring long-range memory and empirical results show that hybrid STU models—with alternating attention and STU layers—can match or even surpass purely attention-based architectures, as well as other popular hybrid state-space models, on long-context tasks such as language modeling and time-series prediction [22]. Furthermore, unlike self-attention, which requires $O(L^2)$ operations per L tokens during training and identically for inference with KV-caching, the STU operates in $O(L \log L)$ operations per L tokens during training and $O(L \log^3 L)$ operations per L tokens during inference through algorithms based on the Fast Fourier Transform [1, 22].

However, although STUs capture LDS-like dynamics using spectral filtering, there has been no straightforward way to convert (or “distill”) a trained STU layer back into explicit LDS form. Moreover, it has remained unclear whether every STU layer can be represented as a LDS with provably small error. Such a distillation from an STU layer to an LDS layer would permit recurrent inference in place of convolution operations, reducing the cost of generating L tokens during inference to $O(L)$, while maintaining the training robustness guarantees of the STU.

Such a distillation is especially desirable in the light of the resurgence of SSMs as scalable alternatives to Transformers in long-context tasks [9, 10, 32]. Thus, our work allows to retain the STU’s performance on long sequences while enabling **constant-time per-token generation**—providing an appealing alternative to both long convolutions and Transformer-based self-attention caching.

1.1 Our Contribution

We present a novel technique for distilling STU filters into LDS form, achieving a substantial reduction in operations—from $O(\log^3 L)$ to $O(\log L)$ per token¹ during generation—while preserving the STU’s expressivity and performance. Moreover, due to the training stability guarantees of the STU architecture, even when learning a marginally stable symmetric LDS or an LDS with high hidden dimension, this distillation procedure provides the **first provable method** to directly learn the parameters of a **symmetric LDS of arbitrarily high effective memory and with bounded noise**. Specifically:

- We show how to convert a learned STU into explicit system matrix parameters whose recurrence can be computed in constant time. This is spelled out in Algorithm 2.
- We provide theoretical analysis of such a distillation, in Theorem 1, and empirically demonstrate that the new LDS form incurs negligible degradation in modeling quality while improving autoregressive generation speed.
- We demonstrate in section 6 that by applying SpectraLDS to trained STUs, we maintain the same accuracy and attain constant per-token computational generation cost.
- As a consequence of our theoretical analysis we show in section 5.3 how we can efficiently convert a symmetric LDS of **arbitrary high dimension**, via an intermediate STU learning step, to one of constant dimension and maintain the same accuracy.

2 Related Work

State-space models have long been a cornerstone of control theory and signal processing. The simplest variants, linear dynamical systems, provide a succinct way to capture temporal dependencies via a hidden state propagated in constant time. Classical methods include the Kalman filter [20], which remain widely used due to its robust theoretical properties and computational efficiency.

In recent years, a new wave of SSMs have emerged as efficient alternatives to attention-based methods for long-sequence tasks, promising sub-quadratic or even near-linear complexity without compromising expressive power. Models like S4 [10] and its diagonal variants [11] exploit structured state matrices to learn long-range dependencies, while works such as Hyena, Mega, and Mamba [32, 24, 9] incorporate gating or convolution-based parameterizations to compete against (and sometimes outperform) transformers in language modeling, time-series analysis, and other long-context applications. Their growing popularity and the challenges of training SSMs underscore the need for methods with greater training robustness, efficiency, and performance guarantees. The

¹We expect the actual reduction to be much larger: from \sqrt{L} , which is a practical method for generation, as opposed to $\log^3 L$, which is a complex algorithm.

83 Spectral Transform Unit (STU) [2] lies squarely in this tradition, offering a powerful convex relaxation
 84 for training LDS-like systems that achieves impressive empirical results on long-context tasks. Our
 85 work builds directly on this line of research, introducing the first method to distill a learned STU
 86 layer into an explicit LDS with provable guarantees, thereby unifying the convex training advantages
 87 of spectral filtering with the real-time inference benefits of a recurrent LDS.

88 Our contributions also align with the long tradition of system identification for LDSs, where the
 89 aim is to learn the hidden transition and output matrices (A, B, C, D) from observed sequences.
 90 Early influential approaches—such as the Ho–Kalman algorithm [15], Kung’s method [21], and the
 91 Eigensystem Realization Algorithm (ERA) [18]—rely on linear-algebraic decompositions (e.g.,
 92 SVD) of Hankel matrices consistent with observations. Modern variants allow for single-trajectory
 93 identification [29], and subsequent refinements like MOESP and N4SID [17] added stochastic
 94 noise modeling, while prediction-error and maximum-likelihood methods improved estimation
 95 accuracy and statistical efficiency. More recent lines of work incorporate regularization and spectral
 96 methods (e.g., stable-spline kernels, sparse identification) to yield more robust or interpretable LDS
 97 representations. In the context of distillation, these methods still require matrix decompositions that
 98 scale super-linearly in the problem size.

99 In contrast to many previous approaches, since the STU’s parameterization avoids direct recon-
 100 struction of the system matrix A , distillation from a STU layer into an LDS remains agnostic to
 101 hidden dimension. Moreover, our method uses a fixed and abstract Hankel matrix—rather than
 102 having to construct it anew from observed data—we can perform a significant part of the distillation
 103 computation offline.

104 Finally, we highlight the recent, closely related work of [26], which distills state-space-like re-
 105 currences from convolution-based sequence models. Their *Laughing Hyena* method accelerates
 106 long convolutional filters by approximating them with a diagonal LDS, thus allowing constant-time
 107 generation at inference. While this approach generalizes to any convolution-based model (e.g.,
 108 Hyena [32]), it does not provide formal guarantees on the distillation quality. In contrast, we focus on
 109 the STU’s spectral filters, which have expressive power comparable to a symmetric LDS with real
 110 eigenvalues, and present the first theoretical framework to convert the filters into such an LDS with
 111 provable bounds on distillation quality. By leveraging the STU’s fixed bank of spectral filters, our
 112 method preserves long-sequence expressiveness while achieving a symmetric LDS realization with
 113 guaranteed approximation error (see Section 5).

114 3 Token Generation and Complexity for Language Modeling

115 In this section, we summarize the autoregressive generation costs for three model classes, Transform-
 116 ers, Convolutional Models, and RNNs, considering a prompt length of T and the generation of K
 117 tokens by each model, with L the length of the convolutional filters (i.e., the maximum sequence
 118 length). We show the runtimes and memory requirements for each of the listed models in Table 1.

119 **Attention.** Processing a prompt requires $O(T^2)$ time. However, token-by-token generation
 120 can be accelerated to $O(T + K)$ for each generated token via key-value caching, with a total of
 121 $O(T^2 + K(T + K)) = O(T^2 + TK + K^2)$ operations and requiring $O(T + K)$ space [34, 26].

122 **Convolutional Model.** A Convolutional Model with k convolutional filters will require $O(kN)$
 123 operations to autoregressively generate a new output given N inputs, and thus a naive autoregressive
 124 convolutional implementation will require $O(k \cdot K \cdot (T + K))$ operations to generate K tokens.
 125 A more refined “Epoched Future Fill” algorithm with prompt prefilling can reduce this to $O(k \cdot$
 126 $(T \log T + K^{3/2} \sqrt{\log K}))$ to generate K tokens. The “Continuous Future Fill” algorithm has
 127 theoretical guarantees of $O(k \cdot (T \log T + K \log^2 K))$ operations, although it suffers from numerical
 128 instability and has not been implemented or used in practice [1]. For the guarantees of the STU
 129 architecture, we require $k = O(\log L)$ and in practice we choose $k = 24$ [22].

130 **RNN (LDS).** For an RNN with state dimension h , autoregressive generation requires $O(h)$ operations
 131 per token generated and $O(h)$ memory, allowing generation of K tokens with $O(h \cdot (T + K))$
 132 operations. As we will prove, for an LDS with representation capacity comparable to an STU with k
 133 filters, we require $h = O(k) = O(\log L)$.

Method	Prefill + Generation Runtime	Cache Size	Runtime with $K, T = O(L)$
Standard Conv	$(TK + T \log T + K^2)k$	$T + K$	$L^2 \log L$
Standard Attn.	$T^2 + TK + K^2$	$T + K$	L^2
EpochedFF	$(T \log T + K^{3/2} \sqrt{\log K})k$	K	$L^{3/2} (\log L)^{3/2}$
ContinuousFF	$(T \log T + K \log^2 K)k$	K	$L \log^3 L$
SpectraLDS (ours)	$(T + K)h$	h	$L \log L$

Table 1: Comparison of architecture runtime and memory requirements for generating K tokens from a length T prompt. $O(\cdot)$ is omitted for brevity. $k, h = O(\log L)$.

4 Problem Background

In this section, we survey the fundamentals relevant to our approach. First, we discuss linear dynamical systems and the inherent challenges of training them directly on tasks requiring long memory. We then present the main theoretical results of spectral filtering and outline how the Spectral Transform Unit (STU) leverages fixed spectral filters to model linear recurrences without explicitly learning the transition matrix A . Finally, we set the stage for our method of distilling STUs back into linear dynamical systems.

4.1 Linear Dynamical Systems

Linear dynamical systems (LDS) have been widely used in control theory to represent time-dependent processes, forming the basis of classical state-space formulations and optimal control methods [20, 23, 3, 19, 6]. Concretely, we consider an input sequence $u_1, u_2, \dots, u_t \in \mathbb{R}^n$, and the corresponding output sequence $y_1, y_2, \dots, y_t \in \mathbb{R}^m$. The hidden state $x_t \in \mathbb{R}^d$ summarizes the system’s memory of past inputs, with the evolution of the system being represented as

$$\begin{aligned} x_t &= A x_{t-1} + B u_t, \\ y_t &= C x_t + D u_t, \end{aligned}$$

where $A \in \mathbb{R}^{d \times d}$, $B \in \mathbb{R}^{d \times n}$, $C \in \mathbb{R}^{m \times d}$, and $D \in \mathbb{R}^{m \times n}$. We omit dynamics and observation noise terms for simplicity with this derivation, although we test our methods on signals with noise.

Expanding the LDS to the Convolutional Form. In a noiseless environment, we can expand the LDS equations as follows:

$$y_t = C x_t + D u_t = C(A x_{t-1} + B u_t) + D u_t = \dots = \sum_{i=0}^{t-1} C A^i B u_{t-i} + D u_t.$$

If any eigenvalue $|\lambda_i(A)| > 1$, the system becomes unstable and y_t may tend to infinity in magnitude. Even for $|\lambda_i(A)| < 1$, systems with $\|A\| \approx 1$ are prone to failure due to large A^i powers in backpropagation, as a noisy algorithm may approximate A with spectral radius greater than 1 during training. If $|\lambda_i(A)| < 1 - \delta$ for some spectral gap $\delta > 0$, then

$$y_t = \sum_{i=0}^{\tau} C A^i B u_{t-i} + \varepsilon_k, \quad \|\varepsilon_k\| \leq \varepsilon,$$

where $\tau = O(\frac{1}{\delta} \log \frac{1}{\varepsilon})$ and the effective memory is thus on the order $\frac{1}{\delta}$ [2]. As $\delta \rightarrow 0$, learning A directly becomes unstable for large contexts [5, 30, 28], highlighting the need for methods such as spectral filtering. Since the D matrix serves as a skip connection that can be easily computed, we fix it to a 0-matrix and omit its consideration for the remainder of this paper. We will sometimes use the shorthand $\text{LDS}(C, A, B)$ to refer to a linear dynamical system with those parameters and $D = 0$.

Additionally, for the remainder of this paper, we restrict our attention to systems where A is a symmetric real matrix. An LDS with symmetric A can be diagonalized, making it equivalent to an LDS with a diagonal A . Without loss of generality, we therefore assume A is diagonal with eigenvalues $\alpha_1, \dots, \alpha_d$.

164 **Spectral Filtering.** With initial state $\vec{0}$ and defining $\mu(\alpha) = (1, \alpha, \alpha^2, \dots)$, we can extend the
 165 convolutional representation:

$$y_t = \sum_{i=0}^L C A^i B u_{t-i} = \sum_{i=0}^L C \left(\sum_{\ell=1}^d \alpha_\ell^i (e_\ell \otimes e_\ell) \right) B u_{t-i} = \sum_{\ell=1}^d (c_\ell \otimes b_\ell) \sum_{i=1}^L \mu(\alpha_\ell)(i) \cdot u_{t-i}$$

166 To circumvent the non-convex optimization problem of finding α that best fit an LDS, [14] propose
 167 the spectral filtering algorithm, which learns an approximation of $\mu(\alpha)$ in a convex manner. They
 168 prove that given eigenvalue-eigenvector pairs $\{\sigma_j, \phi_j\}_{j=1}^L$ of the Hankel matrix Z ,

$$Z := \int_0^1 \mu(\alpha) \mu(\alpha)^\top d\alpha, \quad Z_{i,j} := \frac{2}{(i+j)^3 - (i+j)}$$

169 any $\mu(\alpha)$ with $0 \leq \alpha \leq 1$ can be approximated by the top K eigenvectors $\{\phi_1, \dots, \phi_K\}$ of Z
 170 with an exponentially decreasing error in K . Thus, if y_t is generated by a PSD linear dynamical
 171 system, we have the following result:

$$y_t \approx \sum_{\ell=1}^d (c_\ell \otimes b_\ell) \sum_{i=1}^L \tilde{\mu}(\alpha_\ell)(i) \cdot u_{t-i} = \sum_{k=1}^K M_k \left(\sum_{i=0}^{t-1} \phi_k(i) u_{t-i} \right)$$

172 where we define $\tilde{\mu}(\alpha) := \sum_{i=1}^k \langle \mu(\alpha), \phi_i \rangle \phi_i$ and learn suitable parameters M_k . Rather than depend-
 173 ing on powers of A , learning an LDS with this parameterization remains convex in $\{M_k\}$, since
 174 eigenvectors $\{\phi_k\}$ from the matrix Z are computed offline.

175 **The Spectral Transform Unit.** To account for learning negative eigenvalues of A , the spectral
 176 filtering construction can be adapted by introducing positive and negative sets of feature maps. If
 177 $\{\sigma_j, \phi_j\}$ are the eigenvalue-eigenvector pairs of Z , then for each time t and each respective filter ϕ_j ,
 178 we define the projections of the inputs onto the spectral basis:

$$U_{t,j}^+ = \sum_{i=0}^{t-1} u_{t-i} \cdot \phi_j(i), \quad U_{t,j}^- = \sum_{i=0}^{t-1} u_{t-i} \cdot (-1)^i \cdot \phi_j(i).$$

179 One then forms the output by learning linear combinations of both $U_{t,k}^+$ and $U_{t,k}^-$ and an optional
 180 autoregressive term [12, 2]:

$$y_t^{\text{SF}} = \underbrace{\sum_{j=1}^k M_j^{\phi^+} U_{t-2,j}^+ + \sum_{j=1}^k M_j^{\phi^-} U_{t-2,j}^-}_{\text{Spectral Filtering component}} + \underbrace{\hat{y}_{t-2} + \sum_{i=1}^3 M_i^u u_{t+1-i}}_{\text{AR component}} \quad (2)$$

181 Without the autoregressive component, we compute $y_t^{\text{SF}} = \sum_{j=1}^k M_j^{\phi^+} U_{t,j}^+ + \sum_{j=1}^k M_j^{\phi^-} U_{t,j}^-$. The
 182 above expression is considered the *Spectral Transform Unit*, where $\{M_k\}$ is the set of parameters to
 183 be learned using a differentiable algorithm. Empirical evidence [22] shows that hybrid STU models
 184 can compete with or even outperform purely attention-based architectures.

185 **Error Bounds for Spectral Approximation.** We repeat the result from [25] and [2] stating that
 186 given any LDS parameterized by A, B, C where A is a symmetrix matrix with $\|A\| \leq 1$, there exist
 187 matrices $M_1^{\phi^+}, \dots, M_k^{\phi^+}, M_1^{\phi^-}, \dots, M_k^{\phi^-}$ such that for all L and for all input sequences u_1, \dots, u_L ,
 188 $\|u_t\| \leq 1$, the following holds for all $t \in [L]$:

$$\|y_t^{\text{LDS}} - y_t^{\text{SF}}\| \sim e^{-\frac{k}{\log L}}.$$

189 where k is the number of spectral filters, y_t^{LDS} is the sequence generated by the LDS, and y_t^{SF} is the
 190 sequence generated by spectral filtering.

191 Therefore, one can approximate any LDS that meets these specifications up to error ε by selecting
 192 $k = O(\log L \log(\frac{1}{\varepsilon}))$ spectral filters. Thus, the STU can capture LDS dynamics with only a
 193 logarithmic number of filters k , providing a compact and stable representation even for systems with
 194 high effective memory ($\|A\| \approx 1$).

4.2 Distilling STU into an LDS

While the STU avoids direct learning of A , it still implements LDS-like dynamics via spectral filtering. A natural question is whether we can recover explicit parameters $(\tilde{A}, \tilde{B}, \tilde{C})$ from the learned STU, enabling an equivalent recurrence evolving in $O(1)$ time per token, rather than $O(L)$ when convolving spectral filters with input sequences. This can be achieved by approximating the convolution kernel of the STU by the implicit convolution kernel of an LDS. Such a distillation would bridge the gap between the stable convex training of the STU and the fast inference of a recurrent LDS.

5 Algorithm and Main Result

We now present our main theoretical result and accompanying algorithm, which shows how to recover an accurate LDS representation from the learned spectral filters. Concretely, we demonstrate that each STU filter can be approximated by a linear combination of geometrically decaying LDS filters.

5.1 A General Transformation from Spectral Filters to LDS

Our main result is given in Algorithm 2. We denote a single spectral filter, scaled by $\frac{1}{1-\alpha}$, by

$$\mu_L(\alpha) = (1 \quad \alpha \quad \alpha^2 \quad \dots \quad \alpha^{L-1}).$$

Let $\phi_1, \dots, \phi_k \in \mathbb{R}^L$ be the spectral filters of length L . We write the first k filters in matrix form as $\Phi_{1:k} \in \mathbb{R}^{k \times L}$, such that the i -th row is ϕ_i .

A first observation is that we can write any impulse response for an LDS approximately in the spectral basis. This is a direct consequence of the spectral filtering methodology for learning an LDS [14].

Algorithm 1 FindSpectralRepresentation

- 1: **Input:** Scalar LDS parameter $\alpha \in \mathbb{R}$, representation size k .
- 2: **Output:** Spectral parameters $m \in \mathbb{R}^k$.
- 3: Construct the impulse response vector $\mu_L(\alpha) \in \mathbb{R}^L$ as $\mu_L(\alpha) = (1, \alpha, \alpha^2, \dots, \alpha^{L-1})$.
- 4: Generate a random input vector $u \in \mathbb{R}^L$.
- 5: **return** best spectral fit of the system over random input, (the factor $\gamma = b \cdot c$ is assumed to be one)

$$m = \arg \min_{m \in \mathbb{R}^k} |m^\top \Phi_{1:k} u - \mu_L(\alpha)^\top u|^2.$$

As a consequence of results from spectral filtering for learning LDSs, we show in Appendix A.3 that the procedure FindSpectralRepresentation returns a vector m for which, for some $c > 0$,

$$\|m^\top \Phi_{1:k} - \mu_L(\alpha)\| \leq c e^{-\frac{k}{\log L}}.$$

We proceed to use this subroutine to find a distillation to the spectral filters.

Algorithm 2 Spectral Filters to LDS Filters

- 1: **Input:** The first k spectral filters matrix $\Phi_{1:k} \in \mathbb{R}^{k \times L}$; and parameter $h > k$.
- 2: **Output:** The transformation matrix \tilde{M} .
- 3: Sample h randomly chosen independent vectors $\alpha_1, \dots, \alpha_h$ with each $\alpha \in [0, 1]$.
- 4: Construct the matrix $\mu_L(\alpha_{1:h}) \in \mathbb{R}^{h \times L}$ where each row is $\mu_L(\alpha_i) = (1, \alpha_i, \alpha_i^2, \dots, \alpha_i^{L-1})$.
- 5: For each scalar impulse response, find the spectral representation by

$$m_i = \text{FindSpectralRepresentation}(\alpha_i).$$

- 6: Let M be the $h \times k$ matrix whose rows are m_i .
 - 7: **return** $\tilde{M} := M^{-1} \in \mathbb{R}^{k \times h}$
-

Our main performance guarantee is given in the following theorem.

216 **Theorem 1.** *As long as $h \geq k$, Algorithm 2 returns w.h.p. a matrix \widetilde{M} such that*

$$\left\| \Phi_{1:k} - \widetilde{M} \mu_L(\alpha_1, \dots, \alpha_h) \right\| \leq c \lambda_{\max} h e^{-\frac{k}{\log L}},$$

217 *where λ_{\max} is the largest eigenvalue of the Penrose-Moore pseudo inverse of the matrix M .*

218 The significance of Theorem 1 is that it allows us to translate between the representation of spectral
219 filters and linear dynamical systems.

220 We note that it is not immediate to upper bound λ_{\max} . Indeed, for $h \sim k$, this can be exponentially
221 large in k , as it corresponds to the condition number of a Vandermonde matrix [4]. However, we note
222 experimentally that as h grows, λ_{\max} quickly becomes smaller. This is an overparametrization effect,
223 which we show experimentally in Appendix A.2. We provide analysis of Theorem 1 in Appendix A.3
224 and a separate Algorithm 3 with improved practical performance in Appendix A.4.

225 5.2 Converting \widetilde{M} into an LDS.

226 As a result of Theorem 1, with $A := \text{Diag}(\alpha_1, \dots, \alpha_h)$ and $\mu := \mu_L(\alpha_1, \dots, \alpha_h)$, we can replace
227 the costly STU convolutions:

$$\begin{aligned} U_{t,j}^+ &= \sum_{i=0}^{t-1} u_{t-i} \cdot \phi_j(i) \approx \sum_{i=0}^{t-1} (\widetilde{M}_j \mu)(i) \cdot u_{t-i} = \widetilde{M}_j \sum_{i=0}^{t-1} A^i \vec{1} u_{t-1}^\top = \text{LDS}(\widetilde{M}_j, A, \vec{1})(u_{1:L}^\top) \\ U_{t,j}^- &= \sum_{i=0}^{t-1} u_{t-i} \cdot \phi_j(i) \approx \widetilde{M}_j \sum_{i=0}^{t-1} (-A)^i \vec{1} u_{t-1}^\top = \text{LDS}(\widetilde{M}_j, -A, \vec{1})(u_{1:L}^\top) \end{aligned}$$

228 This provides the basis of our autoregressive inference advantage, as rather than computing
229 $U_{t,j}^+$ and $U_{t,j}^-$ as pure convolutions, we can maintain the hidden state for $\text{LDS}(\widetilde{M}_j, A, \vec{1})$ and
230 $\text{LDS}(\widetilde{M}_j, -A, \vec{1})$ for $O(h)$ computation during inference. For practical efficiency, we can com-
231 pute all $U_{t,1}^+, \dots, U_{t,k}^+, U_{t,1}^-, \dots, U_{t,k}^-$ simultaneously with a single LDS by leveraging the similarities
232 in the state updates (see Appendix A.9).

233 5.3 LDS to LDS Distillation

234 A direct consequence of our approach is that we can distill any high-dimensional symmetric LDS
235 into a low-dimensional LDS with bounded error. For an LDS with input dimension d_{in} and output
236 dimension d_{out} , spectral filtering provides an ε -approximation with only $O(d_{in} \cdot d_{out} \cdot \log L \cdot \log(\frac{1}{\varepsilon}))$
237 parameters regardless of the hidden dimension.

238 Accepting that $h\lambda_{\max}$ is $O(1)$ for $h \gg k$, as justified in Appendix A.2, we can then convert this
239 spectral representation by application of our distillation procedure into an LDS with state dimension
240 $d_{in} \cdot h$ with $O(d_{in} \cdot d_{out} \cdot \log L \cdot \log(\frac{1}{\varepsilon}))$ learned parameters and $O(d_{in} \cdot d_{out} \cdot \log L \cdot \log(\frac{1}{\varepsilon}) + d_{in} \cdot h)$
241 total parameters. In other words, the combination of spectral filtering for LDS learning and the
242 following distillation step yields a practical method to reduce state dimension while preserving the
243 system’s dynamics within tight error bounds. Empirically, for tasks as difficult as modeling language,
244 we only require $k \leq 24$ filters for strong performance, and state dimension $h \geq 80$ of the distilled
245 LDS suffices to retain performance.

246 6 Experiments

247 To demonstrate the effectiveness of our distillation algorithm, we begin by illustrating that a low-
248 dimensional LDS can effectively approximate the spectral filters in Figure 2 and further examine
249 the eigenvalues of the resulting system in Appendix A.9. Building on this, to further quantify the
250 effectiveness of Algorithm 3 in fitting the spectral filters with practically efficient LDSs, we plot
251 the reconstruction error of the spectral filters across different initial subset sizes of LDS impulses
252 in Figure 9 in Appendix A.10. These results show that our algorithm, guided by Theorem 1, not
253 only identifies an initial small state dimension that achieves a low error, but also, by greedily adding

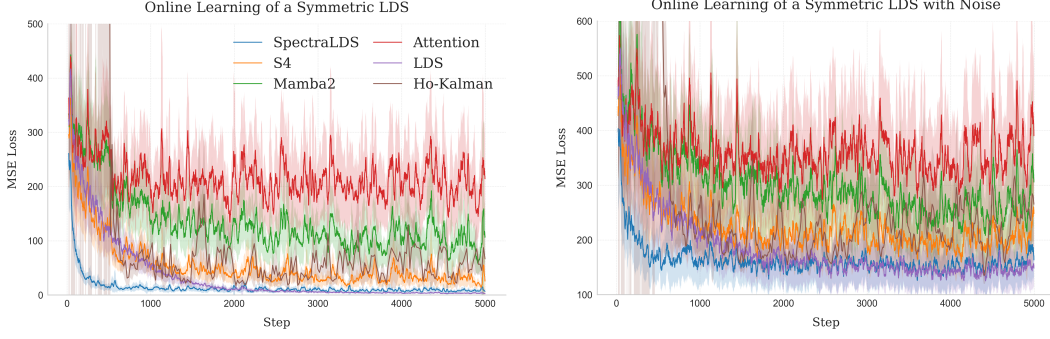


Figure 1: Comparison of SpectraLDS and other methods learning an arbitrary symmetric LDS with and without noise. The shaded region shows the 95% confidence interval over 8 runs. Each model leveraged default configurations except the LDS, which required a lower learning rate to converge. More details are available in Appendix A.7.

254 1D-LDS increments, progressively reduces the approximation error to a sufficiently small level
 255 without an excessive increase in state dimension.

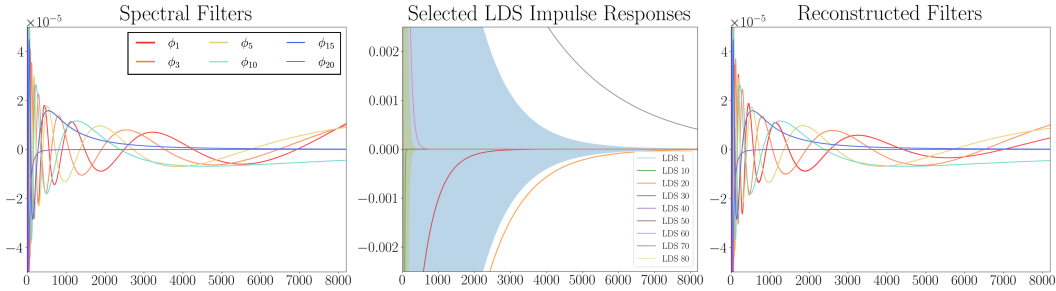


Figure 2: Fit of Spectral Filters by an LDS of state dimension 80 where x-axis represents the time domain. Filters are normalized to be comparable. The blue shading on the middle figure represents a filter quickly alternating (negative eigenvalue). A complete comparison for $k = 24$ without normalization is provided in Appendix A.8.

256 To validate our method quantitatively, we perform synthetic experiments comparing SpectraLDS
 257 against other benchmarks on learning high-memory LDSs (see Appendix A.11). We find that our
 258 method significantly outperforms strong baselines in both sample efficiency and reconstruction accu-
 259 racy, confirming that our approach is well suited for learning systems with long-range dependencies.

260 Turning to the large-scale evaluation, we distill a 340M-parameter FlashSTU model [22] into an
 261 LDS-based architecture and compare their performance across a suite of language benchmarks. From
 262 the results in Table 2, we point out that despite the change from convolution-based spectral filters to
 263 an explicit LDS representation for the STU layers, the performance remains identical across all tasks.
 264 This observation supports our claim that the STU can be closely approximated by a low-dimensional
 265 LDS without compromising predictive accuracy. We provide details of the experimental setup and
 266 hyperparameters for the models used in Appendix A.14.

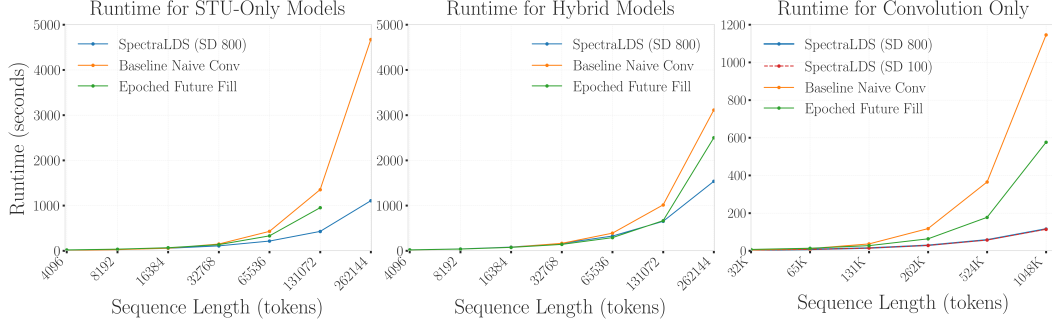


Figure 3: Runtime for generating sequences of increasing length across STU implementations. The naive convolution approach exhibits quadratic growth, the FutureFill variants show logarithmic growth, and the distilled STU-to-LDS layers achieve linear growth. The STU-Only EPOCHED Future Fill OOMs for the largest sequence length. As shown in the rightmost figure, the SpectraLDS models have nearly identical runtime despite varied state dimension. More results are available in A.12.

Model	MMLU	Hella.	PIQA	BoolQ	Wino.	CSQA	OBQA	ARC-e	ARC-c	Average
Flash STU 340M	26.58	30.46	65.34	60.12	51.85	20.48	20.60	54.08	23.29	39.20
SpectraLDS 340M	26.58	30.45	65.29	60.12	50.99	20.15	20.20	54.17	23.29	39.03
Flash STU Std. Err.	0.37	0.47	1.11	0.86	1.40	1.16	2.06	1.02	1.24	—
SpectraLDS Std. Err.	0.37	0.46	1.11	0.86	1.40	1.15	2.07	1.02	1.24	—
Transformer 340M	26.81	30.41	64.64	61.10	51.62	19.98	18.80	55.47	21.84	38.96

Table 2: Evaluation of a 340M-parameter FlashSTU model and its corresponding distilled LDS representation of state dim. 160 across multiple language benchmarks. Despite converting convolution-based spectral filters into an explicit LDS formulation, performance remains statistically equivalent.

Finally, we present various measures of inference speed to illustrate the constant-time per token generation provided by the distilled STU. In Figure 3, we compare the inference speed of a distilled STU model against a naive convolutional approach and a numerically stable FutureFill variant [1]. In the 12 STU layer model, the naive convolution exhibits quadratic runtime growth with sequence length; the FutureFill variants achieve logarithmic growth; and the distilled STU-to-LDS model demonstrates the best performance with linear growth.

In the hybrid model with 6 attention and 6 STU layers, we find the distilled LDS implementation still provides a significant performance increase. Additionally, we note increases in the LDS state dimension have little impact on the overall runtime, indicating that LDS operations are not a compute bottleneck (see Appendix A.12). For both the hybrid and STU-only models up to 16,384 tokens, the distilled LDS, naive convolution, and EPOCHED Future Fill all have similar runtimes. All performance benchmarks were conducted on a single H100 GPU, with each generation process evaluated separately to ensure consistent measurements. We provide full experiment details in Appendix A.13.

Looking ahead, further investigation is warranted to better understand how convolution, LDS, and attention layers interact at the hardware level, and to optimize their coordination for even greater speedup. Additionally, further work is required to determine if the LDS layer is stable below float64.

To conclude, as optimized transformer implementations suffer from KV-Cache memory bottleneck rather than compute bottlenecks [9], and the LDS layers have drastically lower memory requirements, we anticipate that, **with appropriate optimization, the inference speed of hybrid attention-STU architectures will be independent of the amount of LDS layers.**

7 Conclusion and Discussion

We have provided the first provable technique for learning the parameters of a symmetric LDS with arbitrarily high effective memory. By leveraging their convex methodology, we show how spectral filters can be distilled into an explicit LDS representation, enabling the construction of a state-space model with constant-time inference and theoretical guarantees on the loss bounds.

References

- [1] Naman Agarwal, Xinyi Chen, Evan Dogariu, Vlad Feinberg, Daniel Suo, Peter Bartlett, and Elad Hazan. Futurefill: Fast generation from convolutional sequence models, 2024.
- [2] Naman Agarwal, Daniel Suo, Xinyi Chen, and Elad Hazan. Spectral state space models, 2024.
- [3] Brian D.O. Anderson and John B. Moore. *Linear Optimal Control*. Prentice-Hall, 1971.
- [4] Bernhard Beckermann. The condition number of real vandermonde, krylov and positive definite hankel matrices. *Numerische Mathematik*, 85(4):553–577, 2000.
- [5] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.
- [6] Dimitri P Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 2005.
- [7] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.
- [8] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [9] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024.
- [10] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022.
- [11] Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models, 2022.
- [12] E. Hazan, H. Lee, K. Singh, C. Zhang, and Y. Zhang. Spectral filtering for general linear dynamical systems, 2018.
- [13] Elad Hazan and Karan Singh. Introduction to online nonstochastic control. *arXiv preprint arXiv:2211.09619*, 2022.
- [14] Elad Hazan, Karan Singh, and Cyril Zhang. Learning linear dynamical systems via spectral filtering, 2017.
- [15] B. L. Ho and R. E. Kalman. Effective construction of linear state-variable models from input/output functions. *Automatica*, 3(1):49–59, 1966.
- [16] A. Huang, S. Subramanian, J. Sum, K. Almubarak, S. Biderman, and S. Rush. The annotated transformer, 2022.
- [17] I. W. Jamaludin, N. A. Wahab, N. S. Khalid, S. Sahlan, Z. Ibrahim, and M F. Rahmat. N4sid and moesp subspace identification methods. In *2013 IEEE 9th International Colloquium on Signal Processing and its Applications*, pages 140–145, 2013.
- [18] Jer-Nan Juang and Richard S. Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics*, 8(5):620–627, 1985.
- [19] Thomas Kailath. *Linear Systems*. Prentice-Hall, 1980.
- [20] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82.1:35–45, 1960.
- [21] S. Y. Kung. A new identification and model reduction algorithm via singular value decompositions. In *Proc. 12th Asilomar Conf. Circuits, Systems and Computers*, pages 705–714, 1978.

- 336 [22] Y. I. Liu, W. Nguyen, Y. Devre, E. Dogariu, A. Majumdar, and E. Hazan. Flash stu: Fast spectral
337 transform units, 2024.
- 338 [23] David Luenberger. An introduction to observers. *IEEE Transactions on Automatic Control*,
339 16(6):596–602, 1971.
- 340 [24] Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan
341 May, and Luke Zettlemoyer. Mega: Moving average equipped gated attention, 2023.
- 342 [25] Annie Marsden, Evan Dogariu, Naman Agarwal, Xinyi Chen, Daniel Suo, and Elad Hazan.
343 Provable length generalization in sequence prediction via spectral filtering, 2024.
- 344 [26] Stefano Massaroli, Michael Poli, Daniel Y. Fu, Hermann Kumbong, Rom N. Parnichkun,
345 Aman Timalina, David W. Romero, Quinn McIntyre, Beidi Chen, Atri Rudra, Ce Zhang,
346 Christopher Ré, Stefano Ermon, and Yoshua Bengio. Laughing hyena distillery: Extracting
347 compact recurrences from convolutions, 2023.
- 348 [27] OpenAI. Gpt-4 technical report, 2024.
- 349 [28] A. Orvieto, S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De. Resurrecting
350 recurrent neural networks for long sequences, 2023.
- 351 [29] Samet Oymak and Necmiye Ozay. Non-asymptotic identification of lti systems from a single
352 trajectory. In *2019 American control conference (ACC)*, pages 5655–5661. IEEE, 2019.
- 353 [30] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks,
354 2013.
- 355 [31] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito,
356 Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in
357 pytorch. 2017.
- 358 [32] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua
359 Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional
360 language models, 2023.
- 361 [33] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases
362 enables input length extrapolation, 2022.
- 363 [34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez,
364 Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: We claim that we have identified a method to distill spectral filters to linear dynamical system form and demonstrate in the experiments section that our method successfully works.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: At the end of the experiments section, we mention the current limitations of our distillation. Additionally, throughout we demonstrate understanding of the limitations of our approach, such as restricting our results to linear dynamical systems with symmetric system matrices.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: Our main theoretical result is Theorem 1, in which we mention complete assumptions made and demonstrate when and why they would hold both theoretically and experimentally in the main paper and in the Appendix.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Yes, we describe our algorithm completely in 3 and 2, as well as in A.5 and A.4. Our code repository will also be open sourced.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: A link to the code will be provided in the final version. However, we omit the GitHub repository with all of our code at the moment in order not to violate the double blind policy and keep the paper anonymous.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Experiment details are expansively explained in 6, A.14, A.2, A.11, and in many other instances throughout the paper.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: 3, 2, and other graphs where relevant include error bars and statistical significance.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In 6, A.14, and in all places where relevant, the paper describes completely the computer resources needed and used. Furthermore, since many of our results relate to inference speed, we explain the hardware used and time of execution/memory extensively.

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The paper conforms simply in every respect with the NeurIPS Code of Ethics as explained.

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: [NA]

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: [NA]

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All previous work is fully credited and cited throughout.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our new assets are well documented and the documentation is provided.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: [NA]

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

463 Answer: [NA]

464 Justification: [NA]

465 **16. Declaration of LLM usage**

466 Question: Does the paper describe the usage of LLMs if it is an important, original, or
467 non-standard component of the core methods in this research? Note that if the LLM is used
468 only for writing, editing, or formatting purposes and does not impact the core methodology,
469 scientific rigorousness, or originality of the research, declaration is not required.

470 Answer: [NA]

471 Justification: [NA]

A Appendix

A.1 Experimental details and notation

All experiments were performed on Nvidia H100-80GB GPUs in PyTorch [31]. All layers except the LDS leverage bfloat16 precision, whereas the LDS layers require float64 precision. All tests, unless otherwise stated, use $k = 24$ spectral filters and replace the k filters with an LDS with state-dimension $h = 80$. To fit the additional 24 negative filters of the negative component (i.e., ϕ_j^- , where $\phi_j^-[i] = (-1)^{i-1}\phi_j[i]$, as is needed to compute $U_{t,j}^-$ purely convolutionally), we expand to an LDS with state-dimension $h = 160$.

For the layer-level inference speed benchmarks, we leverage the STU with tensor-dot approximation (STU-T) [2], the same variant used in the FlashSTU language model [22]. It is worth noting that the STU-to-LDS distillation leads to LDS layers with comparable speed regardless of whether the tensor-dot approximation is employed, and thus this approximation only accelerates the STU layers in benchmarks. A formal definition of this approximation is as follows:

STU Tensor-Dot Approximation: [2] introduced an important optimization, the tensor-dot approximation, wherein each tensor $M \in \mathbb{R}^{d_{\text{in}} \times 2k \times d_{\text{out}}}$, representing a concatenation of the tensors $M_1^{\phi^+}, \dots, M_k^{\phi^+}, M_1^{\phi^-}, \dots, M_k^{\phi^-}$, is learned as $M^{(1)} \times M^{(2)}$ for $M^{(1)} \in \mathbb{R}^{d_{\text{in}} \times 2k}$ and $M^{(2)} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$. This approximation allows for a reduction in convolutions as, with input x_1, x_2, \dots, x_ℓ , we can compute $y_\ell^{\text{SF}} \approx \sum_{i=1}^\ell (x_{\ell-i+1}^\top M^{(2)}) \odot M_{\text{filters}}[i]$, where $M_{\text{filters}} = [\phi_1, \dots, \phi_k, \phi_1^-, \dots, \phi_k^-]^\top M^{(1)} \in \mathbb{R}^{L \times d_{\text{out}}}$ and \odot refers to the Schur product (i.e. $(x \odot y)_j = x_j \cdot y_j$). This allows for only d_{out} convolutions with the tensor-dot approximation, as opposed to $k \cdot d_{\text{in}}$ convolutions without it. Although this method can reduce expressivity and does not inherit the same marginal-stability guarantees, it maintains competitive empirical performance while yielding significant improvements in efficiency.

Additionally, we frequently refer to the impulse response of the LDS and STU models. The impulse response of a linear sequence model $f : \mathbb{R}^L \rightarrow \mathbb{R}^1$ is the vector or convolutional kernel $\psi \in \mathbb{R}^L$ that is equivalent to f (i.e. $f(x_{[1, \dots, L]}) = \psi * x_{[1, \dots, L]}$), and thus:

$$\psi[t] = f(\underbrace{[0, \dots, 1, \dots, 0]}_{1 \text{ at position } L-t+1})$$

We only consider the impulse response of $f : \mathbb{R}^L \rightarrow \mathbb{R}^1$, but the impulse response is closely related to the derivative with respect to the inputs and is generalized identically. For an STU model with $d_{\text{in}} = d_{\text{out}} = 1$ and parameters $M_1^{\phi^+}, \dots, M_k^{\phi^+}, M_1^{\phi^-}, \dots, M_k^{\phi^-} \in \mathbb{R}$, the impulse response ψ_{SF} is thus described by $\psi_{\text{SF}}[t] = \sum_{j=1}^k M_j^{\phi^+} \phi_j[t] + (-1)^{t-1} \sum_{j=1}^k M_j^{\phi^-} \phi_j[t]$. Similarly, for an LDS with $d_{\text{in}} = d_{\text{out}} = 1$ and parameters $a, b, c \in \mathbb{R}$, the impulse response ψ_{LDS} is described by $\psi_{\text{LDS}}[t] = ca^{t-1}b$.

A.2 Experimental Results on the Condition Number of M , the Spectral Coefficients Matrix

In Figure 4, we present experimental results on the condition number of M as defined in Section 5. Recall that $M \in \mathbb{R}^{h \times k}$ is the spectral coefficients matrix produced by Algorithm 2; its i -th row m_i^\top stores the coefficients that express the LDS impulse response $\mu_L(\alpha_i)$ (with geometric decay factor α_i) in the spectral basis $\Phi_{1:k}$. Starting at $h = k$ with $k = 48$, we repeatedly add additional independent vectors α_i and measure the maximum singular value of the constructed M^{-1} . The blue central line shows the mean maximum singular value across 10 experiments for a given value of h , with the shaded region showing the maximum and minimum of the largest singular values across experiments. α were drawn independently from the distribution in Figure 6, which is near uniform but oversamples values in the range $[-1, -0.9] \cup [0.9, 1]$ and greatly oversamples in the range $[-1, -0.99] \cup [0.99, 1]$. To justify our statement in section 5.2 that $\lambda_{\text{max}} \cdot h$ can be considered $O(1)$ in a practical domain, we additionally plot $\lambda_{\text{max}} \cdot h$ under the same experimental setup in Figure 5.

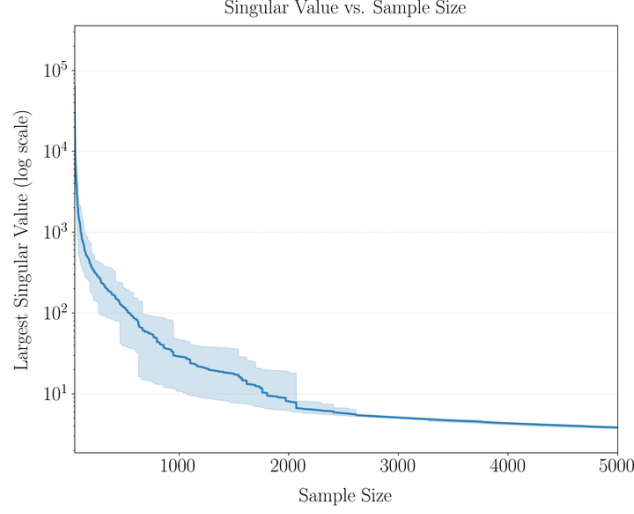


Figure 4: Largest Singular Value as we increase h .

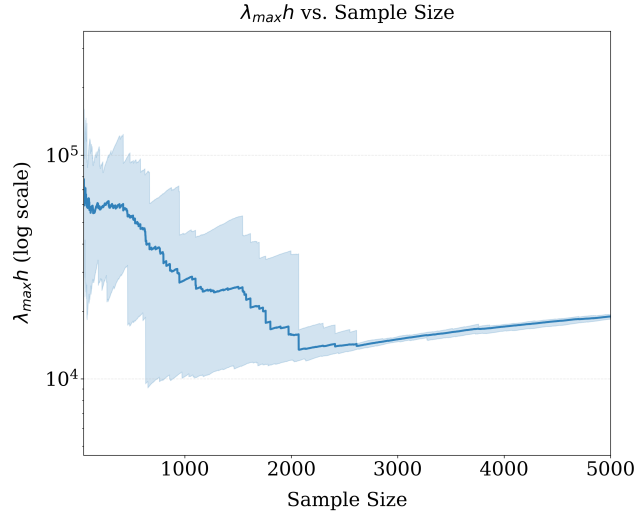


Figure 5: $\lambda_{max} \cdot h$ as we increase h .

518 A.3 Analysis of Theorem 1

Proof of Theorem 1. Let

$$\bar{\mu}_L(\alpha) = (\alpha - 1) \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{L-1} \end{bmatrix} \in \mathbb{R}^L,$$

519 be the L -impulse response of the one dimensional linear dynamical system with parameters
 520 $\alpha, b = 1, c = \alpha - 1$. Lemma 11.3 from [13] asserts that for any $\alpha \in [0, 1]$, there exist real
 521 coefficients $m_1, \dots, m_k \in \mathbb{R}^k$ and constant $c > 0$, such that any sequence of inputs $u_{L:1} \in \mathbb{R}^L$ with
 522 $\|u_{L:1}\|_\infty \leq 1$,

$$\left| \sum_{i=1}^k m_i \langle \phi_i, u_{L:1} \rangle - \langle \bar{\mu}_L(\alpha), u_{L:1} \rangle \right| \leq c e^{-\frac{k}{\log L}}.$$

523 The RHS term represents the evolution of the LDS, with system matrices b, c that are assumed to be
 524 identity, and $\bar{\mu}_L(\alpha)$ is the system evolution.

525 As the above is true for all $u_{L:1}$, for each α_j , there exist m_j such that,

$$\forall j \quad , \quad \|m_j^\top \Phi_{1:k} - \bar{\mu}_L(\alpha_j)\| \leq c e^{-\frac{k}{\log L}}. \quad (3)$$

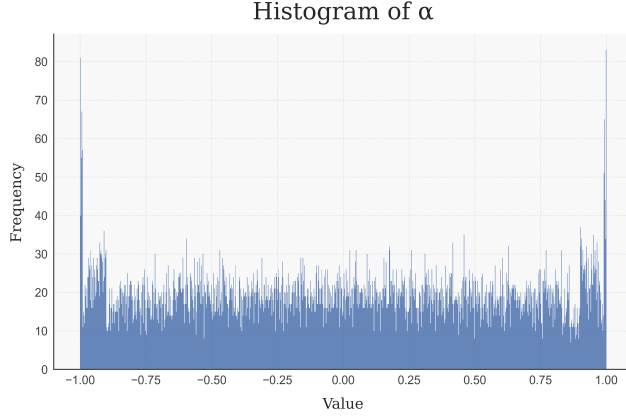


Figure 6: Distribution of α

526 Let $M \in \mathbb{R}^{h \times k}$ be the matrix whose rows are $m^j \in \mathbb{R}^k$:

$$M = \begin{pmatrix} -m^1 - \\ -m^2 - \\ \vdots \\ -m^h - \end{pmatrix}.$$

527 Let $\mathcal{E} = M \Phi_{1:k} - \bar{\mu}_L(\alpha_1, \dots, \alpha_h)$. By the triangle inequality and (3),

$$\|\mathcal{E}\|_1 \leq \sum_{j=1}^h \|m_j^\top \Phi_{1:k} - \bar{\mu}_L(\alpha_j)\| \leq c h e^{-\frac{k}{\log L}}. \quad (4)$$

Thus, assuming that M is full rank, multiplying \mathcal{E} by the Penrose-Moore pseudo-inverse of M and using Holder's inequality, we get

$$\|\Phi_{1:k} - M^{-1} \bar{\mu}_L(\alpha_1, \dots, \alpha_h)\| = \|M^{-1} \mathcal{E}\| \leq \|M^{-1}\|_* \|\mathcal{E}\|_1 \leq \lambda_{\max} \cdot c h e^{-\frac{k}{\log L}}.$$

528 It remains to argue that M is full rank. This follows since $\Phi_{1:k}$ is an orthogonal basis, and the matrix
 529 $\mu_L(\alpha_{1:h})$ is a Vandermonde matrix. Thus, both matrices are full rank. \square

530 A.4 The Practical Algorithm

Algorithm 3 LDS Impulse Responses to Matrix Representation

```

1: Input: Learned STU filters  $\Phi_{1:k} \in \mathbb{R}^{k \times L}$ 
2: Output: System matrices  $\{A, B, C\}$  of a discrete-time LDS.
3: Initialize  $\Psi \in \mathbb{R}^{N \times L}$  and  $\Theta \in \mathbb{R}^{N \times k}$  as empty matrices.
4: for  $i = 1, \dots, N$  do
5:   Sample  $(a_i, b_i, c_i)$  with  $|a_i| \leq 1$ 
6:   Train an STU of filter length  $L$  on the 1D-LDS defined by  $(a_i, b_i, c_i)$ 
7:   Let  $\psi_i = (cb, cab, \dots, ca^{L-1}b) \in \mathbb{R}^{1 \times L}$  be the LDS impulse response; update the  $i^{\text{th}}$  row of
      $\Psi$  by  $\Psi[i] \leftarrow \psi_i$ .
8:   Let  $\theta_i \in \mathbb{R}^{1 \times k}$  be the corresponding learned STU weights; update the  $i^{\text{th}}$  row of  $\Theta$  by
      $\Theta[i] \leftarrow \theta_i$ .
9: end for
10: Repeatedly sample a row-subset  $I_{\text{sub}} \subset [1, \dots, N]$  of size  $h_{\text{start}}$  and keep the sample that
    minimizes  $\|\Theta_{\text{sub}}^\dagger \Psi_{\text{sub}} - \Phi_{1:k}\|_F^2$ , where  $\Psi_{\text{sub}} := \Psi[I_{\text{sub}}]$  and  $\Theta_{\text{sub}} := \Theta[I_{\text{sub}}]$ .
11: for  $t = h_{\text{start}} + 1, \dots, h$  do
12:    $i^* \leftarrow \operatorname{argmin}_{i \in \{1, \dots, N\} \setminus I_{\text{sub}}} \|\Theta[I_{\text{sub}} \cup \{i\}]^\dagger \Psi[I_{\text{sub}} \cup \{i\}] - \Phi_{1:k}\|_F^2$ 
13:    $I_{\text{sub}} \leftarrow I_{\text{sub}} \cup \{i^*\}$ 
14: end for
15: Initialize  $\widetilde{M} \leftarrow \Theta_{\text{sub}}^\dagger$ 
16: repeat
17:   Update  $\widetilde{M} \leftarrow \widetilde{M} - \eta \nabla_{\widetilde{M}} \left( \|\Phi_{1:k} - \widetilde{M} \Psi_{\text{sub}}\|_F^2 \right)$ 
18: until convergence
19: Set  $A \leftarrow \operatorname{diag}(\{a_i\}_{i \in I_{\text{sub}}})$ ,  $B \leftarrow (b_i)_{i \in I_{\text{sub}}}$ ,  $C \leftarrow \widetilde{M} \operatorname{diag}(\{c_i\}_{i \in I_{\text{sub}}})$ 
20: return  $\{A, B, C\}$ 

```

531 A.5 Recovering the Best-Fit LDS for the Spectral Filters

532 We now elaborate on Algorithm 3, which converts the STU filters into an explicit LDS representation.

533 **1. Generating and Fitting Random LDSs.** We first sample many random (a, b, c) system parameters to produce a large collection of one-dimensional LDSs:

$$(1\text{D-LDS}) \quad x_t = a x_{t-1} + b u_t, \quad y_t = c x_t,$$

536 where $a \in \mathbb{R}$ satisfies $|a| \leq 1$. In particular, we skew toward generating LDSs with a matrices that have high eigenvalues, and therefore high effective memory, in order to accurately capture the full shape of the spectral filters later in the algorithm (i.e., see Figure 6).

538 Following the findings of [25], which show that an STU trained on sufficiently long LDS outputs can achieve minimal approximation error, we train an STU to approximate each LDS by feeding in a series of Gaussian inputs u_t into both the LDS and an STU of comparable filter length L . We optimize the STU weights to align its output with that of the LDS, and measure the approximation error using the mean squared error (MSE) between their outputs, discarding any instance whose approximation exceeds a threshold. This procedure corresponds to training the STU to best fit the impulse response for each 1D-LDS due to [14]’s learning guarantees (Appendix A.6). Thus, we yield a dataset of paired (LDS, STU) systems where the STU’s learned spectral filters successfully replicate the LDS impulse response.

547 **2. Collecting LDS Impulses and STU Weights.** For each of the retained 1D-LDSs, we explicitly record its impulse response, $\psi_{(i)} \in \mathbb{R}^L$, as the vector $\psi_{(i)}[t] = c a^{t-1} b$ for $t = 1, \dots, L$. Stacking these impulse responses row by row yields a matrix

$$\Psi = \begin{pmatrix} -\psi_{(1)}- \\ -\psi_{(2)}- \\ \vdots \\ -\psi_{(N)}- \end{pmatrix} \in \mathbb{R}^{N \times L},$$

where N is the number of LDS examples that we kept.

Meanwhile, we also obtain the STU weights learned to replicate each LDS. In our implementation, each STU layer is parameterized by spectral coefficients $M_1^{\phi+}, \dots, M_k^{\phi+}$ and, if negative filters are used, $M_1^{\phi-}, \dots, M_k^{\phi-}$, which correspond to the positive and negative components of the spectral projection. We can combine these coefficients to form a vector $\theta_i \in \mathbb{R}^k$ (or \mathbb{R}^{2k} if we also incorporate the negative components). Stacking these vectors θ_i row-by-row produces a matrix $\Theta \in \mathbb{R}^{N \times k}$.

At a high level, each row of Ψ is an LDS impulse response, while each row of Θ describes how the STU forms that impulse vector from its spectral basis. Thus, Ψ and Θ capture a variety of 1D-LDSs whose shapes are well-represented by STU filters.

3. Subset Selection and Pseudoinverse Next, we use matrices Ψ and Θ to approximate the target STU filters $\Phi_{1:k}$. We wish to project the STU filters on the basis Ψ of LDS impulses in \mathbb{R}^L .

From the analysis in Section A.3, we look for a matrix \widetilde{M} such that

$$\Phi_{1:k} \approx \widetilde{M} \Psi.$$

We have shown that each row of the LDS impulse vectors Ψ can be approximated by the STU filters using the weights matrix Θ (i.e. $\Theta \Phi_{1:k} \approx \Psi$). Thus, one can invert the filter mapping and form \widetilde{M} :

$$\widetilde{M} = \Theta_{\text{sub}}^\dagger \quad \text{on a suitably chosen subset.}$$

Concretely, our implementation samples fixed-size subsets among the N LDSs and retains the subset I_{sub} that best reconstructs the spectral filters. We then greedily expand I_{sub} to reduce the approximation error of the reconstructed filters $\Phi_{1:k}$. Since each new example adds one dimension (i.e. one more row) to the potential LDS basis, our approach incrementally expands the underlying LDS state dimension only when necessary.

4. Gradient-Based Fine-Tuning. Once we have a suitable subset, we refine the matrix $\widetilde{M} = \Theta_{\text{sub}}^\dagger$ by gradient descent:

$$\widetilde{M} = \underset{\widetilde{M}}{\text{argmin}} \|\Phi_{1:k} - \widetilde{M} \Psi_{\text{sub}}\|_F^2.$$

Since Ψ_{sub} and $\Phi_{1:k}$ are both fixed, this is a simple least-squares objective. Empirically, we often observe an approximately 1.4x decrease in the MSE for the fits of the spectral filters once this stage completes.

5. Combining Into a Single LDS. Finally, we merge the chosen 1D-LDSs into a single higher-dimensional discrete-time system. In particular, suppose our final subset Ψ_{sub} includes h different 1D systems, with parameters $\{(a_i, b_i, c_i)\}_{i=1}^h$. The gradient refined matrix \widetilde{M} helps determine how to combine these 1D systems to produce the best approximation of the spectral filters. We then form a system:

$$\mathbf{A} = \text{diag}(a_1, \dots, a_h) \in \mathbb{R}^{h \times h}, \quad \mathbf{B} = \begin{pmatrix} b_1 \\ \vdots \\ b_h \end{pmatrix} \in \mathbb{R}^{h \times 1}, \quad \mathbf{C} = \widetilde{M} \cdot \text{diag}(c_1, \dots, c_h) \in \mathbb{R}^{k \times h}$$

where each (a_i, b_i, c_i) corresponds to one of the original 1D LDS components. The net dimension (i.e., the number of included components.) of the hidden state is h .

Thus, for any vector $v \in \mathbb{R}^{1 \times k}$, $v \cdot \Phi_{1:k} \approx v \cdot \langle CB, CAB, CA^2B, \dots \rangle$, and the coefficients learned on the spectral filters can be directly ported to the LDS. In this manner, the STU's spectral filters are translated into a classical LDS. Empirically, $h \approx 160$ has high accuracy in our experiments. Once we have parameters $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ in standard form, we can generate new sequences in $\mathcal{O}(h)$ time per token, exactly like a conventional recurrent model, while achieving the same performance as the STU's filters.

For a practical performance boost, although we fit each 1D-LDS with an STU with positive and negative filters, we then fit I_{sub} and \widetilde{M} by only considering the reconstruction error on the positive

591 filters and return an LDS that solely reconstructs the positive filters. Then replacing A with $-A$ leads
 592 to a new LDS that fits the negative spectral filters, and we can thus fit both positive and negative
 593 filters with state dimension $2h$. This approach typically leads to better reconstruction accuracy.

594 A.6 Why Matching Outputs Corresponds to Matching Impulse Responses

595 We start by recalling that Theorem 2.1 of [2] ensures that the outputs of an LDS and the spectral
 596 filtering (SF) approximation are close for any bounded input sequence $\{u_t\}$ with $\|u_t\| \leq a$ when A
 597 is a PSD matrix with $\|A\| \leq 1$. In particular, if we set $\{u_t\}$ to the sequence of all ones (so $a = 1$),
 598 the theorem guarantees that for every time t ,

$$\|y_t^{\text{LDS}} - y_t^{\text{SF}}\|^2 \leq \varepsilon,$$

600
 601 for some ε on the order of $c \|B\|_{\text{col}} \|C\|_{\text{col}} L^3 e^{-\left(\frac{\pi^2}{4}\right) \cdot \frac{k}{\log(L)}}$.

603 Let $\psi_{\text{LDS}}[i]$ denote the i -th impulse response of the LDS. Likewise, let $\psi_{\text{SF}}[i]$ be the corresponding
 604 impulse response of the spectral filtering approximation. We observe that if the input sequence is all
 605 ones up to time $r \geq t$, then the resulting output at time t is precisely the sum of the first t impulse
 606 responses:

$$y_t^{\text{LDS}} = \sum_{i=1}^t \psi_{\text{LDS}}[i], \quad \text{and} \quad y_t^{\text{SF}} = \sum_{i=1}^t \psi_{\text{SF}}[i].$$

607 Hence, the condition $\|y_t^{\text{LDS}} - y_t^{\text{SF}}\|^2 \leq \varepsilon$ for all t can be rewritten as

$$\left\| \sum_{i=1}^t \psi_{\text{LDS}}[i] - \sum_{i=1}^t \psi_{\text{SF}}[i] \right\|^2 \leq \varepsilon \quad \text{for each } t.$$

608 To see that each individual impulse $\psi_{\text{LDS}}[t]$ is close to $\psi_{\text{SF}}[t]$, note that

$$\psi_{\text{LDS}}[t] - \psi_{\text{SF}}[t] = \left(\sum_{i=1}^t \psi_{\text{LDS}}[i] - \sum_{i=1}^t \psi_{\text{SF}}[i] \right) - \left(\sum_{i=1}^{t-1} \psi_{\text{LDS}}[i] - \sum_{i=1}^{t-1} \psi_{\text{SF}}[i] \right).$$

609 As both $\left\| \sum_{i=1}^t \psi_{\text{LDS}}[i] - \sum_{i=1}^t \psi_{\text{SF}}[i] \right\|^2 \leq \varepsilon$ and $\left\| \sum_{i=1}^{t-1} \psi_{\text{LDS}}[i] - \sum_{i=1}^{t-1} \psi_{\text{SF}}[i] \right\|^2 \leq \varepsilon$, then

$$\|\psi_{\text{LDS}}[t] - \psi_{\text{SF}}[t]\|^2 \leq \left\| \sum_{i=1}^t \psi_{\text{LDS}}[i] - \sum_{i=1}^t \psi_{\text{SF}}[i] \right\|^2 + \left\| \sum_{i=1}^{t-1} \psi_{\text{LDS}}[i] - \sum_{i=1}^{t-1} \psi_{\text{SF}}[i] \right\|^2 \leq 2\varepsilon.$$

610
 611 Therefore, matching the outputs of the LDS and the STU within small error forces each impulse
 612 response of the LDS to be close to the corresponding impulse response of the SF approximation,
 613 which generalizes to any input sequence bounded as above. We thus conclude that matching outputs
 614 implies matching impulse responses.

615 A.7 Learning a Symmetric LDS with and without noise

616 Figure 1 compares SpectraLDS to other system-identification methods on the task of learning an
 617 arbitrary symmetric LDS with and without noise. The LDS signal has hidden dimension 256, input
 618 and output dimension 5, and maximum eigenvalue magnitude 0.99 with Gaussian initialization. Each
 619 step provides a Gaussian input sequence of length 100 with variance $1/d_{in}$ and the final output. For
 620 the learning with noise experiment, Gaussian noise with 0.5 variance was added to the hidden state at
 621 each step, and Gaussian noise with 5 variance was added to the output. Each architecture was trained
 622 with the default PyTorch RMSProp optimizer configuration, except for the LDS, which required
 623 a lower learning rate to converge. The Ho-Kalman matrices are recomputed every 20 steps for
 624 computational ease, while the other methods were updated every step, and the Ho-Kalman parameters
 625 are the maximum for an input of length 100 (that is $T_{\text{horizon}} = 99$, $T_1 = 49$, $T_2 = 49$, state_dim_est
 626 $= 48$). The shaded region in the figure shows the 95% confidence interval over 8 runs.

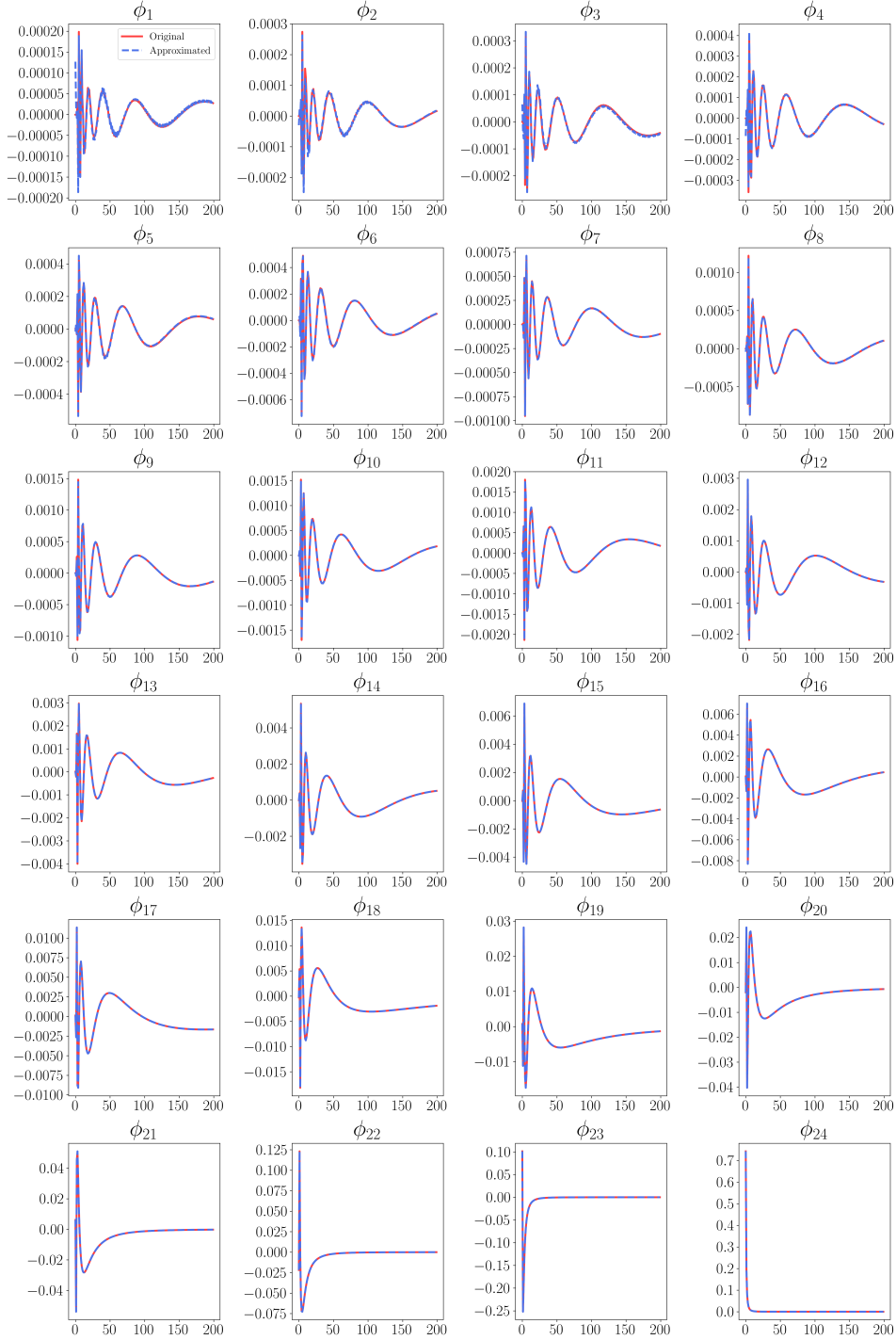


Figure 7: Visualization of the positive spectral filters and their approximation using an LDS with state dimension 80, as obtained via Algorithm 3. The figure illustrates that even with a low-dimensional state, the LDS can accurately fit the spectral filters, confirming the efficacy of our distillation process.

628 A.9 State Dynamics of the Linear Dynamical System

629 We train a linear dynamical system with state dimension 80 to fit the Spectral Filters using Algorithm 3
 630 and examine the eigenvalues of the resulting system in Figure 8. The reconstruction error of the
 631 spectral filters for the trained LDS is 1.23×10^{-12} . We transform the eigenvalues to visualize the
 632 full range of the values as they approach 1 and differentiate the negative and positive eigenvalues
 633 as blue and red (e.g, 0.97 and -0.97 would both map to ≈ 3.5 , although the first would join a red
 634 column and the second blue).

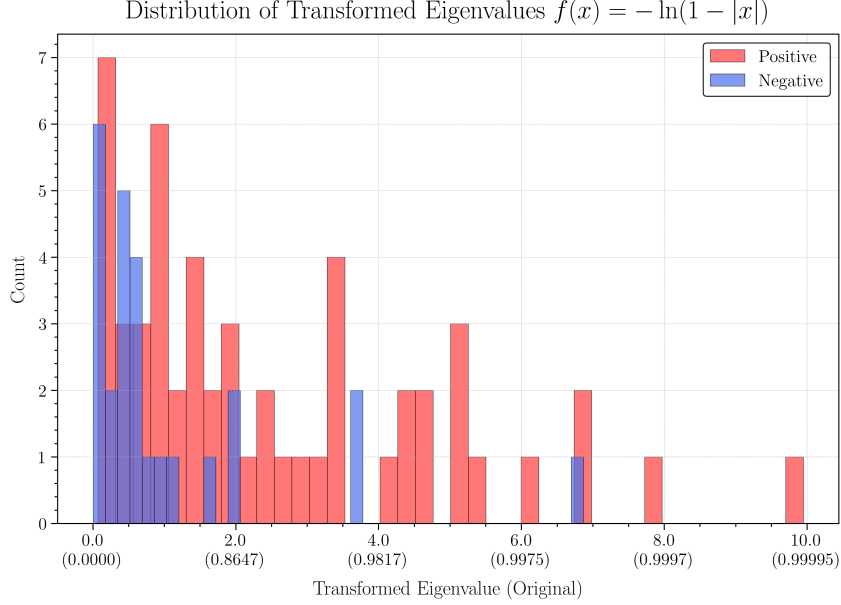


Figure 8: Distributions of Eigenvalues for LDS trained to predict first 24 Spectral Filters.

635 For practical efficiency, we can compute all $U_{t,1}^+, \dots, U_{t,k}^+, U_{t,1}^-, \dots, U_{t,k}^-$ simultaneously with a single
 636 LDS parameterized by $\left\{ \begin{bmatrix} \widetilde{M} & \mathbf{0} \\ \mathbf{0} & \widetilde{M} \end{bmatrix}, \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & -A \end{bmatrix}, \mathbf{1} \right\}$. This LDS has a state dimension of $2km$,
 637 where each of the m input dimensions have a hidden state updated by $\begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & -A \end{bmatrix}$. The hidden state
 638 dimension differs from $2k$ as we treat m as a batch dimension, resulting in $Bu_t \in \mathbb{R}^{2k \times m}$ rather
 639 than \mathbb{R}^{2k} . This mimics an LDS acting independently across each of the m dimensions, equivalent to
 640 treating m as a batch-axis, and this is the analog to how each of the k spectral filters convolve along
 641 each input dimension independently. The above Figure 8 showcases the eigenvalues of matrix A ,
 642 whereas for practical deployment, the state matrix is $\begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & -A \end{bmatrix}$.

643 A.10 Reconstruction Error of the Spectral Filters by Initial LDS State Dimension

644 We demonstrate the efficacy of Algorithm 3 in fitting the spectral filters with a low initial LDS state
 645 dimension. We generate 61784 1D-LDSs, with A 's eigenvalue distribution similar to Figure 6. For
 646 each LDS, we then fit it with an STU model with $k = 24$ filters and without an autoregressive term.
 647 This provides a large set of LDS impulse and STU impulse pairs, which allows us to relate the spectral
 648 filters to the LDS impulses. As described in Algorithm 3, we choose an initial subset of these LDSs,
 649 perform the matrix inversion procedure described in Algorithm 2, and measure the error. We greedily
 650 add additional LDS impulses, which corresponds to increasing h , to minimize the loss until we reach
 651 a desired state dimension. In Figure 9, we ablate the starting subset size for this procedure and
 652 measure the loss. Before running this procedure, we filter out any LDS-STU pairs where the STU did
 653 not learn the LDS to within a certain threshold. This figure does not include the final gradient descent
 654 step used to improve the fit, which typically leads to 30% to 50% improvements. Figure 7 and Figure

8 leverage an LDS with state dimension 80 and error $1.23 \cdot 10^{-12}$. In this procedure we only aim to fit the k positive spectral filters. To fit both the k positive spectral filters and k negative spectral filters, we can achieve the same error with double the state dimension, as described in Appendix A.9.

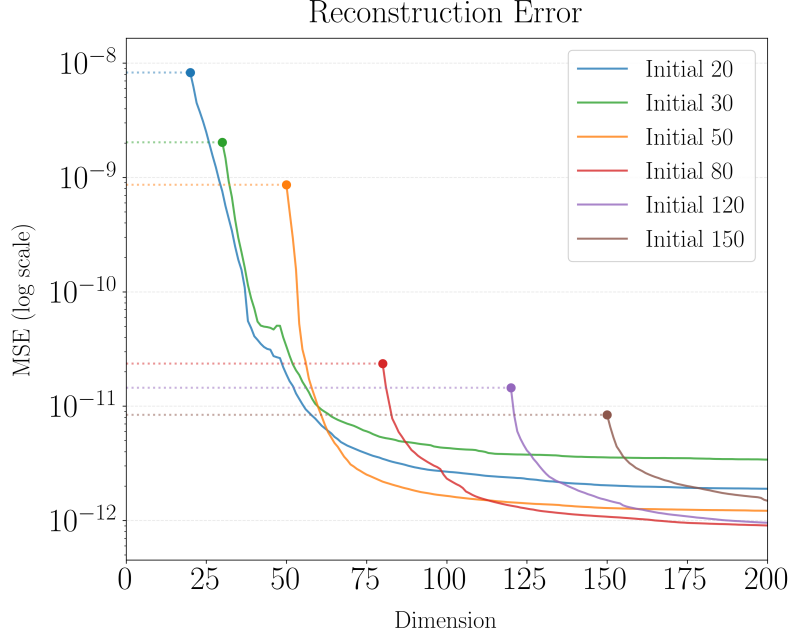


Figure 9: Reconstruction error of spectral filters as a function of LDS state dimension. Each curve starts from a different initial subset size of LDS impulses (chosen by best sample of that size out of 10^6 trials) and is then expanded greedily in single-dimensional increments until reaching a final state dimension of 200.

A.11 Experiments on Synthetic Tasks

We evaluate SpectraLDS against a strong baseline on a synthetic task, reporting results in Table 3 and Figure 10. A symmetric linear dynamical system with input dimension d_{in} and state dimension d_h is initialized by drawing all entries from a standard normal distribution, after which the update matrix A is rescaled so its largest eigenvalue has magnitude $1 - \delta$. We then distill a SpectraLDS model from an STU model trained with AdaGrad (learning rate 1.0) for 2000 steps; each step minimizes the mean-squared-error (MSE) between the STU output and the ground-truth LDS output over batches of 32 sequences of length seq_len . For the baseline, we fit a randomly initialized symmetric LDS with AdaGrad (learning rate 0.0001) under the same loss. To keep training times practical, most baseline runs use a shorter sequence length, and several runs were stopped early due to computational limits, marked with an asterisk. **When the state dimension satisfies $d_h \geq 1000$, baseline training with a learning rate of 0.01 becomes unstable and often diverges, whereas the STU remains stable even with a learning rate of 1.0.**

Performance Results											
Type	Len.	Delta	d_in	d_h	Avg MSE	MSE Std	Step 10 Loss	Step 100 Loss	Runs	Time (s)	Time Std (s)
LDS GD	8192	1×10^{-2}	10	100	2.15×10^{-2}	1.32×10^{-3}	3.49×10^{-1}	1.43×10^{-1}	4*	2153.12	14.16
LDS GD	8192	1×10^{-3}	10	100	2.53×10^{-2}	3.18×10^{-3}	4.53×10^{-1}	1.80×10^{-1}	3*	2147.61	18.64
LDS GD	8192	1×10^{-4}	10	100	3.03×10^{-2}	6.09×10^{-3}	5.99×10^{-1}	2.29×10^{-1}	3*	2150.73	28.05
LDS GD	1024	1×10^{-2}	10	100	9.54×10^{-3}	0.00×10^0	1.18×10^{-1}	6.72×10^{-2}	1*	248.33	0.00
LDS GD	1024	1×10^{-2}	10	1000	1.90×10^{-5}	5.93×10^{-6}	5.25×10^{-4}	3.39×10^{-5}	5	252.24	1.64
LDS GD	1024	1×10^{-3}	10	100	7.34×10^{-3}	1.02×10^{-3}	1.06×10^{-1}	4.97×10^{-2}	2*	248.68	0.13
LDS GD	1024	1×10^{-3}	10	1000	2.40×10^{-5}	1.71×10^{-6}	4.00×10^{-4}	3.71×10^{-5}	5	250.24	0.87
LDS GD	1024	1×10^{-4}	10	100	7.72×10^{-3}	2.52×10^{-4}	1.13×10^{-1}	5.31×10^{-2}	2*	248.32	0.38
LDS GD	1024	1×10^{-4}	10	1000	2.01×10^{-5}	4.53×10^{-6}	5.29×10^{-4}	3.53×10^{-5}	5	250.58	1.40
LDS GD	1024	1×10^{-5}	10	100	6.77×10^{-3}	6.43×10^{-4}	1.01×10^{-1}	4.84×10^{-2}	5	250.80	0.17
LDS GD	1024	1×10^{-5}	10	1000	1.94×10^{-5}	4.92×10^{-6}	4.79×10^{-4}	3.39×10^{-5}	5	251.51	0.67
SpectraLDS	8192	1×10^{-2}	10	100	4.21×10^{-4}	1.78×10^{-4}	1.33×10^{-2}	4.60×10^{-4}	5	42.14	0.02
SpectraLDS	8192	1×10^{-2}	10	1000	1.48×10^{-5}	1.96×10^{-6}	1.23×10^{-2}	5.68×10^{-5}	5	48.11	1.39
SpectraLDS	8192	1×10^{-3}	10	100	3.59×10^{-4}	1.67×10^{-4}	1.27×10^{-2}	4.02×10^{-4}	5	42.15	0.02
SpectraLDS	8192	1×10^{-3}	10	1000	1.44×10^{-5}	4.05×10^{-6}	1.52×10^{-2}	5.81×10^{-5}	5	47.40	0.03
SpectraLDS	8192	1×10^{-4}	10	100	3.18×10^{-4}	9.33×10^{-5}	1.30×10^{-2}	3.61×10^{-4}	5	42.10	0.01
SpectraLDS	8192	1×10^{-4}	10	1000	1.88×10^{-5}	2.68×10^{-6}	1.21×10^{-2}	6.17×10^{-5}	5	48.07	1.36
SpectraLDS	8192	1×10^{-5}	10	100	4.23×10^{-4}	2.40×10^{-4}	1.38×10^{-2}	4.60×10^{-4}	5	42.13	0.01
SpectraLDS	8192	1×10^{-5}	10	1000	1.51×10^{-5}	4.74×10^{-6}	1.36×10^{-2}	5.94×10^{-5}	5	47.40	0.02
SpectraLDS	8192	1×10^{-6}	10	100	2.68×10^{-4}	7.13×10^{-5}	1.16×10^{-2}	3.07×10^{-4}	5	42.12	0.01
SpectraLDS	8192	1×10^{-6}	10	1000	1.29×10^{-5}	2.05×10^{-6}	1.32×10^{-2}	5.29×10^{-5}	5	47.43	0.03
SpectraLDS	8192	1×10^{-7}	10	100	4.36×10^{-4}	1.16×10^{-4}	1.35×10^{-2}	4.79×10^{-4}	5	42.12	0.02
SpectraLDS	8192	1×10^{-7}	10	1000	1.43×10^{-5}	3.18×10^{-6}	1.44×10^{-2}	5.61×10^{-5}	5	47.38	0.02

Table 3: SpectraLDS performance on learning synthetic linear dynamical systems with maximum eigenvalue $1 - \delta$ against a strong baseline.

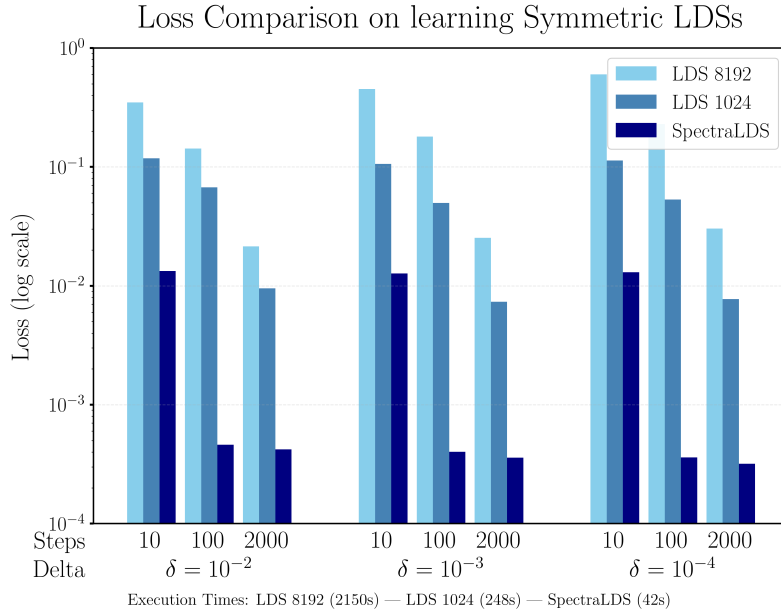


Figure 10: Comparison of learning performance on synthetic LDS tasks between SpectraLDS and a gradient-descent-updated LDS baseline. The plot shows the loss at different training steps under varying δ , input/output dimensions, and sequence lengths. SpectraLDS consistently achieves lower loss with significantly reduced runtime, especially at larger output dimensions. Full experimental details are provided in Table 3.

671 A.12 Layer Speeds of SpectraLDS and STU

672 We benchmark the inference speed of a single SpectraLDS layer across several state dimensions
673 and compare it with two accelerated schemes for computing the STU convolution: Epochd Future
674 Fill [1] and an STU that employs the tensor-dot approximation [22]. The SpectraLDS layer itself is
675 produced by applying STU-to-LDS distillation to the tensor-dot STU. For the timings reported in
676 Table 4, each model is evaluated on L (Seq. Len.) autoregressive convolutions on inputs of dimension
677 128, and the mean and standard deviation over five runs are recorded. All STU variants use a filter

length of 1,048,576 to accommodate the longest sequences. Although every architecture begins in the linear-scaling regime, only SpectraLDS continues to scale favorably as the sequence length increases. The resulting layer-speed comparison is visualized in Figure 11; benchmarks of SpectraLDS layers embedded in large language-model architectures appear in Appendix A.13.

Inference Time Performance (ms)										
Seq. Len.	STU Future Fill		STU Tensor-Dot		STU		SpectraLDS (SD 100)		SpectraLDS (SD 800)	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
32	30.86	1.20	31.87	6.43	134.96	22.13	21.27	3.86	18.98	0.73
64	36.52	0.72	28.92	0.42	123.16	2.54	22.40	0.14	22.27	0.29
128	49.87	0.52	40.20	6.63	135.94	2.28	29.49	0.18	29.72	0.58
256	75.73	0.35	45.98	1.12	162.74	2.90	43.54	0.27	43.62	0.47
512	136.67	7.90	70.24	0.61	211.66	2.99	71.40	0.56	73.01	0.88
1024	234.10	6.99	122.89	1.19	317.91	4.79	127.52	1.25	128.98	0.97
2048	441.09	3.69	226.47	8.02	525.65	5.09	241.16	4.43	252.58	8.78
4096	863.81	8.72	426.43	7.96	936.94	6.53	472.15	8.40	477.96	8.42
8192	1856.63	191.17	837.46	2.03	1772.30	20.46	921.55	10.23	929.06	9.76
16384	3377.29	33.16	1812.17	16.47	3434.19	30.83	1839.63	25.21	1846.04	17.43
32768	7139.35	515.40	4286.86	24.74	6721.83	19.65	3620.84	43.71	3686.90	26.45
65536	13485.25	234.22	11614.43	28.04	13478.53	89.72	7181.13	35.96	7362.73	78.96
131072	27252.83	240.34	36427.10	14.56	26703.77	323.97	14356.92	74.91	14649.56	162.32
262144	63437.26	123.25	117796.38	268.48	55775.49	275.16	28573.69	119.09	29156.42	113.55
524288	177502.89	651.96	365168.31	460.23	142896.29	621.72	57083.23	404.91	58027.04	276.39
1048576	576129.57	1425.12	1145862.83	1805.50	451607.11	721.73	114044.79	1215.07	115999.77	670.78

Table 4: Autoregressive Inference Time (ms) across model architectures (5 runs).

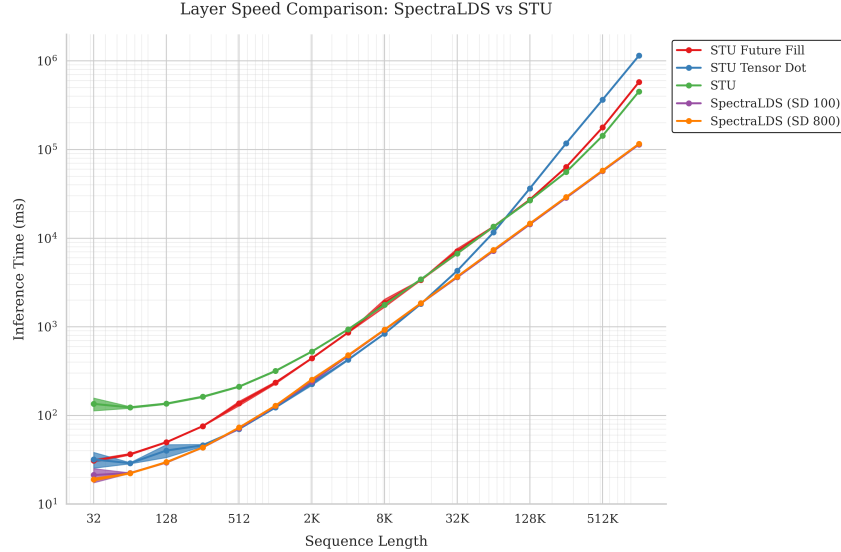


Figure 11: Autoregressive Inference Time (ms) across model architectures.

A.13 FlashSTU Ablations

A.13.1 Implementation Architecture Details

To perform the FlashSTU performance evaluations with STU-T, we employ the architecture depicted in Figure 12. We test token generation efficiency with a hybrid architecture, alternating between layers using STU-T and sliding window attention (SWA), and we additionally test with each layer using STU-T only. Each layer consists of RMS-Norm, followed by STU-T or SWA, followed by RMS-Norm, and then followed by an MLP. Inputs are tokenized by the o200k_base tokenizer and the FlashSTU model begins with an embedding layer, which is weight-tyed to the output unembedding layer. To start generation, we add special tokens such as `<|endoftext|>` and `<|endofprompt|>`.

692 The sliding window attention layers leverage Flash Attention v2 [8, 7] and ALiBi position embeddings
 693 [33]. The tested model has input dimension 896 and 12 layers, which has 550.31 million parameters
 694 for the hybrid model, and 535.99 million parameters for the STU-only model. All layers are
 695 run in bfloat16 except for the LDS distilled layers, which require float64 precision. The Flash
 696 STU-T leverages the STU with tensor-dot [22] approximation rather than the base STU layer for
 697 faster inference, and thus we perform the STU-to-LDS distillation on the STU with tensor-dot
 698 approximation. For tests with generation length up to 131072, we leverage STU filter length of
 699 131072. For the generation length of 262144, we leverage an STU filter length of 262144. For each
 700 runtime measurement, we first run a warmup generation, before reporting the mean of two generations
 701 of that length. All benchmarks include only inference time and not model setup or filter computation
 702 costs. Additionally, each MLP layer has hidden dimension $12\times$ the input dimension (MLP expansion
 703 factor). Other configuration details are identical to those in Table 7.

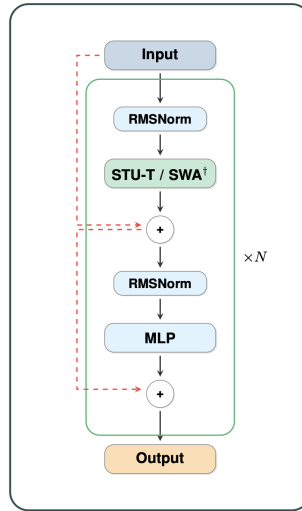


Figure 12: FlashSTU architecture [22].

704 A.13.2 Implementation Efficiency

705 Using the setup of Fig. 12, we time autoregressive generation for sequence lengths 4,096–262,144
 706 tokens under two architectures—(i) a hybrid network that interleaves STU-T and sliding-window
 707 attention layers and (ii) an STU-only network in which every layer is STU-T. Tables 5–6 report mean
 708 runtimes (over two runs after a warmup) for SpectraLDS with state dimensions 100–800 alongside
 709 the STU-T with naive convolutions (Base STU) and the EPOCHED FutureFill method. We note three
 710 main results. First, SpectraLDS runtimes grow nearly linearly with sequence length and are virtually
 711 independent of the chosen state dimension. Second, while the STU-T with naive convolutions is
 712 competitive at 4–8 k tokens, it becomes progressively slower, so that SpectraLDS is $\approx 2\times$ faster
 713 by 65 k tokens and over $4\times$ faster at 262 k tokens in the STU-only setting (and $2\times$ faster in the
 714 hybrid setting). Third, EPOCHED FutureFill narrows the gap at medium lengths but is still outpaced by
 715 SpectraLDS beyond 131 k tokens and, in the STU-Only architecture, exhausts memory (OOM) at 262
 716 k tokens, whereas SpectraLDS completes the run. Together these results demonstrate that SpectraLDS
 717 delivers the most favorable long-context scaling and remains robust across model hyper-parameters.

Seq. Len.	LDS SD 100	LDS SD 200	LDS SD 400	LDS SD 800	Base STU	FutureFill
4096	20.34	20.71	20.40	20.40	18.91	19.92
8192	40.48	41.32	40.69	40.82	38.09	38.39
16384	80.78	82.90	80.95	81.63	76.49	74.89
32768	161.43	163.92	162.13	163.30	164.70	141.16
65536	323.40	327.31	323.90	325.92	389.72	290.68
131072	646.66	653.96	648.99	651.35	1014.67	666.24
262144	1588.81	1639.24	1591.50	1536.20	3113.45	2498.89

Table 5: Hybrid Model Runtime (seconds) for generation across SpectraLDS with different state dimensions and Baseline and Epoched FutureFill implementations

Seq. Len.	LDS SD 100	LDS SD 200	LDS SD 400	LDS SD 800	Base STU	FutureFill
4096	14.04	13.87	13.87	13.74	12.14	19.35
8192	28.15	27.74	27.67	27.42	24.84	35.86
16384	55.71	55.51	55.34	54.73	57.32	67.96
32768	110.85	110.88	110.57	108.68	149.12	135.75
65536	220.11	221.55	221.13	215.00	429.20	329.91
131072	439.87	442.65	442.04	428.36	1352.40	952.70
262144	1104.97	1101.76	1132.18	1107.49	4672.60	OOM

Table 6: STU-Only Runtime (seconds) for generation across SpectraLDS with different state dimensions and Baseline and Epoched FutureFill implementations

718 A.14 Details of STU Model In Experiments

719 We summarize in Table 7 all relevant details for the FlashSTU model used in the language evaluations
720 in Table 2. The distilled LDS layer used in the language benchmarking experiments was obtained
721 by Algorithm 3 and has a state dimension of 160, incorporating both positive and negative spectral
722 components. The weights for the distilled model were directly transferred from the FlashSTU model
723 described below. The FlashSTU architecture is further described in Appendix A.13 and graphically
724 shown in Figure 12.

Model Architecture		
	Description	Flash STU
Parameter Count	Total number of parameters	340M
Embedding Dimension	Dimensionality of embedding space	1024
Number of Heads	Attention heads	4
Number of Layers	Transformer + STU layers	12
ALiBi Attention	Attention scores modification using linear biases	No
RoPE Theta	RoPE scaling factor for rotary embeddings	10,000
Sliding Window Size	Sliding window attention context lookback size	512
Sequence Length (Training)	Input sequence length during training	4,096
Sequence Length (Inference)	Input sequence length during inference via position interpolation	131,072
Vocabulary Size	Size of the model's vocabulary	200,064
MLP Expansion Factor	Expansion factor in MLP layers	4
Bias	Use of bias terms in linear layers	No
Dropout	Dropout rate	0.0
Number of Filters	Number of filters	24
Training and Optimization		
Epochs	Number of training epochs	1
Global Batch Size	Number of tokens processed per step	524,288
Micro Batch Size	Batch size per GPU	8
Gradient Accumulation Steps	Number of steps before performing a gradient update	8
Warmup Steps	Number of warmup steps	1,907
Evaluation Period	Evaluation frequency (steps)	50
Max Grad Norm	Maximum gradient norm for clipping	1.0
Optimizer Configuration		
Optimizer	Optimizer type	AdamW
Learning Rate Schedule	LR scheduling strategy	Linear decay with warmup
Max Learning Rate	Maximum learning rate	4.0×10^{-4}
Min Learning Rate	Minimum learning rate	4.0×10^{-5}
Torch Dtype	Data type for PyTorch tensors	bf16
Betas	Optimizer betas	(0.9, 0.999)
Epsilon	Optimizer epsilon	1.0×10^{-8}
Weight Decay	Weight decay factor	1.0×10^{-2}
AMSGrad	Use AMSGrad variant	No
Fused	Use fused optimizer	Yes
Optimization Techniques		
Activation Checkpointing	Enable activation checkpointing	Yes
Use Flash FFT	Enable Flash FFT	No
Use Tensordot Approx.	Enable tensordot approximation	Yes
Use Attention	Enable attention mechanism	Yes
Softcap	Softcap threshold	50.0
Torch Compile	Enable Torch compile optimization	Yes

Table 7: Model and training configuration details for the 340M Flash STU model.

Benchmarks were conducted with task-specific few-shot configurations. For each task, if the few-shot value is not set to -1, that number of examples is provided during evaluation; otherwise, the few-shot setting is left unchanged. Specifically, the few-shot settings are as follows: HellaSwag: 0 shots, MMLU: 5 shots, PIQA: 0 shots, BoolQ: 0 shots, Winogrande: -1, Commonsense_QA: 7 shots, OpenBookQA: -1, ARC-Easy: -1, ARC-Challenge: -1. The model employs bfloat16 precision except for the LDS computations, where it requires float64 precision, and a global batch size of 3 examples per task. All computations were performed on a single H100 GPU.

A.15 Example Generation

The following is an example completion from the FlashSTU 340M model after LDS distillation with temperature 1.0:

Prompt: After school each day, Deidre Smith would look over the essays her students submitted at the end of class and, without fail, each day Alex would surprise her. Today, he wrote about

Generated Text: After school each day, Deidre Smith would look over the essays her students submitted at the end of class and, without fail, each day Alex would surprise her. Today, he wrote

740 about 40 essays from his freshman class, all of which were collected over the past three years for The
741 Book of Sentiments: What Students Should Do When Reading Literature – First published in 1962,
742 this book was very well received. What people who read this book would not immediately accept or
743 appreciate the author's thought-provoking message of what it means to read literature.

744 What exactly are you doing with your newfound ability to read? You are reading? What emotions
745 do you feel at the time? What do you do with your newfound ability to read? These are just a few
746 examples of how much reading students think they can gain from the experience. It is no accident,
747 then, that each and every level of student will have difficulty with the text in this book. Students are
748 given a choice: choose an assignment that you are comfortable with and one that you would like to
749 read more deeply and critically. As the task to read becomes more difficult and the pressure for it is
750 much greater, students will struggle to put their best minds together when reading literature. To help
751 students overcome this hurdle, I have devised a book that makes the task a bit easier by providing all
752 the information they need from a reader. If a reader is too afraid to read, I have designed a page that
753 provides guidance from the textbook that will...