THE CURIOUS CASE OF IN-TRAINING COMPRESSION OF STATE SPACE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

State Space Models (SSMs), developed to tackle long sequence modeling tasks efficiently, offer both parallelizable training and fast inference. At their core are recurrent dynamical systems that maintain a hidden state, with update costs scaling with the state dimension. A key design challenge is striking the right balance between maximizing expressivity and limiting this computational burden. Control theory, and more specifically Hankel singular value analysis, provides a potent framework for the measure of energy for each state, as well as the balanced truncation of the original system down to a smaller representation with performance guarantees. Leveraging the eigenvalue stability properties of Hankel matrices, we apply this lens to SSMs *during training*, where only dimensions of high influence are identified and preserved. Our approach applies to Linear Time-Invariant SSMs such as Linear Recurrent Units, but is also extendable to selective models. Experiments show that in-training reduction significantly accelerates optimization while preserving expressivity, with compressed models retaining task-critical structure lost by models trained directly at smaller dimension. In other words, SSMs that begin large and shrink during training achieve computational efficiency while maintaining higher performance.

1 INTRODUCTION

State Space Models (SSMs) Gu et al. (2021); Hasani et al. (2022); Smith et al. (2022); Orvieto et al. (2023); Rusch & Rus (2025) have recently emerged as a powerful alternative to established sequence models such as Recurrent Neural Networks (RNNs) and Transformers. They combine the parallelizable training efficiency of scaled dot-product attention Vaswani (2017) with the computational and memory advantages of RNNs, enabling strong performance across large-scale language, vision, and audio modeling tasks Gu & Dao (2024); Goel et al. (2022); Nguyen et al. (2022).

Despite their efficient structure and recent progress in hardware-aware implementations, current SSMs remain computationally-intensive. While both memory and runtime scale with sequence length, the size of the SSM state further amplifies these costs. Reducing the state dimension therefore provides an effective strategy to simultaneously reduce memory usage and runtime. This can be achieved by leveraging techniques from structured compression. However, most existing approaches are commonly applied post-training: a large model is trained to completion and only compressed afterwards. Popular examples include knowledge distillation Hinton et al. (2015), post-training quantization Jacob et al. (2018), low-rank factorization Hu et al. (2022), and structured pruning Li et al. (2016). All of these methods typically require the costly upfront training of a large network.

In this article, we address the issue of costly pre-training by introducing CompreSSM, a principled in-training compression technique that effectively reduces the dimension of the SSM while largely preserving the expressive power of uncompressed models. We motivate our approach by the control-theoretic origins of SSMs Kalman (1960). In particular, we draw on balanced truncation Antoulas (2005), a classical Model Order Reduction (MOR) technique that approximates a high-dimensional state space system with a low-dimensional one while retaining its essential input–output behavior. Observing that the dominant Hankel Singular Values (HSVs) of an SSM are rank-preserving during training, we propose truncating dimensions associated with small Hankel singular values once they fall below a predefined relative threshold.

Main contributions. In the subsequent sections, we will demonstrate the following features of CompreSSM:

- We rigorously justify the validity of our in-training compression approach by establishing means to identify and track the HSVs of an SSM during training, and showing that dominant singular values are rank-preserving. Thus, SSM dimensions associated with small HSVs can be safely truncated.
- We show that CompreSSM is broadly applicable, including to SSMs with structured state matrices such as diagonal matrices, with simple extensions to Linear Time-Varying systems discussed.
- We provide an extensive empirical evaluation demonstrating that CompreSSM largely preserves the expressive power of uncompressed models.
- CompreSSM significantly accelerates training, while achieving similar or higher accuracy than larger uncompressed models by truncating large portions of the state early in training.

2 Mathematical Preliminaries

2.1 Discrete Linear Time Invariant Systems

Let $\mathcal{G}$ be a discrete *Linear Time-Invariant (LTI)* system described by state equations:

$$\begin{aligned} \boldsymbol{h}(k+1) &= \boldsymbol{A}\,\boldsymbol{h}(k) + \boldsymbol{B}\,\boldsymbol{x}(k)\,, \quad \boldsymbol{h}(0) = \boldsymbol{h}_0 \\ \boldsymbol{y}(k) &= \boldsymbol{C}\,\boldsymbol{h}(k) + \boldsymbol{D}\,\boldsymbol{x}(k), \end{aligned} \tag{1}$$

where $\boldsymbol{h} \in \mathbb{R}^n$ is the state, $\boldsymbol{x} \in \mathbb{R}^p$ the input, and $\boldsymbol{y} \in \mathbb{R}^q$ the output, with $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times p}$, $\boldsymbol{C} \in \mathbb{R}^{q\times n}$, and $\boldsymbol{D} \in \mathbb{R}^{q\times p}$.

A more general class of systems are *Linear Time-Varying (LTV)*, where the matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ are functions of time. Such systems become relevant in the context of *selective* SSMs, where the system matrices depend on the input. For now, we restrict to the LTI case as the base framework. The LTV case is discussed in Appendix A.2.

The LTI framework provides a tractable and well-understood setting in which powerful tools such as Gramians and balanced truncation can be developed. Before introducing these concepts, we formalize the standard assumptions of stability, controllability, and observability, which ensure that the system is both well-posed and non-degenerate. For precise definitions of terms and a detailed background presentation we refer the reader to chapters 4, 5, and 6 of Chen (1999).

Assumption 2.1. *The system is stable, i.e all the eigenvalues of $\boldsymbol{A}$ are of amplitude less than 1.*

Assumption 2.2. *The pair $(\boldsymbol{A}, \boldsymbol{B})$ is controllable, i.e., the state $\boldsymbol{h}$ can be be steered from any initial state to any final state in finite time.*

Assumption 2.3. *The pair $(\boldsymbol{A}, \boldsymbol{C})$ is observable, i.e., observing the output $\boldsymbol{y}$ and the input $\boldsymbol{x}$ of the system for some finite time suffices to determine the initial state $\boldsymbol{h}_0$.*

2.1.1 Controllability and Observability Gramians

The concepts of controllability and observability can be captured quantitatively by matrix-valued energy measures called Gramians. These respectively encode how easily internal states can be excited by inputs or observed from outputs.

Under assumptions 2.1 and 2.2, there exist a unique symmetric positive definite solution $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ to the discrete Lyapunov equation:

$$\boldsymbol{A}\boldsymbol{P}\boldsymbol{A}^{\mathrm{T}} - \boldsymbol{P} + \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} = 0, \qquad \boldsymbol{P} = \sum_{i=0}^{\infty} \boldsymbol{A}^i \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} (\boldsymbol{A}^{\mathrm{T}})^i \tag{2}$$

$\boldsymbol{P}$ is known as the *discrete controllability Gramian*. It intuitively captures how much energy from the input can reach each state dimension over time. Large entries indicate states that are easily influenced by inputs.

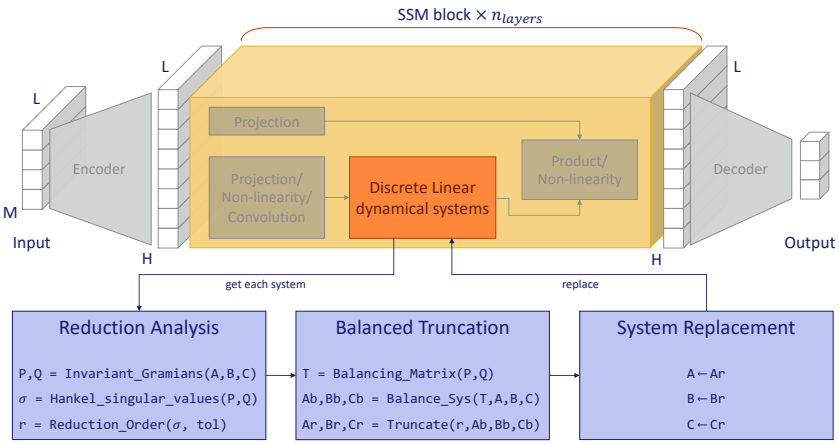


Figure 1: Overview of the proposed balanced truncation pipeline. The method applies at the level of the discrete linear dynamical systems inside SSM layers, independently of surrounding design choices such as projections, non-linearities, convolutions, or skip connections. Each dynamical system is isolated, balanced via its controllability and observability Gramians, and truncated according to Hankel singular values before being reinserted into the model.

Under assumptions 2.1 and 2.3, there exist a unique symmetric positive definite solution $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ to the discrete Lyapunov equation:

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{A} - \boldsymbol{Q} + \boldsymbol{C}^{\mathrm{T}}\boldsymbol{C} = 0, \qquad \boldsymbol{Q} = \sum_{i=0}^{\infty} (\boldsymbol{A}^{\mathrm{T}})^i \boldsymbol{C}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{A}^i \tag{3}$$

$\boldsymbol{Q}$ is known as the *discrete observability Gramian*. It similarly measures how much each state contributes to the outputs over time. Large entries correspond to state directions that are easily observed through the outputs.

2.1.2 Balanced Realizations

The notions of controllability and observability come together in the concept of a diagonal balanced realization, in which the system is transformed so that both Gramians are equal and diagonal. This provides a natural coordinate system where each state dimension has a well-defined "importance."

Definition 2.4 (State Space Realization)**.** A discrete-time linear system $\mathcal{G}$ is fully characterized by its input–output map

$$\mathcal{G} : \{\boldsymbol{x}(k)\}_{k\geq 0} \;\mapsto\; \{\boldsymbol{y}(k)\}_{k\geq 0}.$$

A *realization* of $\mathcal{G}$ is any quadruple of matrices $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ and state $\boldsymbol{h}(k) \in \mathbb{R}^n$ that *realizes* this input–output map via the dynamics given by Equation 1. Many different realizations can realize the same map. In particular, if $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ is a realization, then so is $(\boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}, \boldsymbol{T}^{-1}\boldsymbol{B}, \boldsymbol{C}\boldsymbol{T}, \boldsymbol{D})$ for any invertible $\boldsymbol{T} \in \mathbb{R}^{n\times n}$.

Definition 2.5 (Minimal/Balanced realizations)**.** A realization is called *minimal* if it is both controllable and observable. The corresponding state dimension n is called the *order* of the realization.

A realization is said to be *balanced* if $\boldsymbol{P} = \boldsymbol{Q}$. In this case we denote the common matrix by $\boldsymbol{W}$, and refer to it simply as *the Gramian* of the balanced system.

Theorem 2.6 (Antoulas (2005))**.** *Any stable, minimal discrete LTI system admits a balanced realization, in which the controllability and observability Gramians coincide as* $\boldsymbol{W} = diag(\boldsymbol{\sigma}) = diag(\sigma_1, \ldots, \sigma_n)$*, with* $\sigma_1 \geq \cdots \geq \sigma_n > 0$ *called "Hankel singular values" (HSV). This diagonal balanced realization can be explicitly constructed .*

The HSVs $\boldsymbol{\sigma}$ can also be computed in decreasing order from non-balanced realizations via:

$$\boldsymbol{\sigma} = \mathrm{sort}_{\downarrow}\big(\sqrt{\mathrm{spec}(\boldsymbol{P}\boldsymbol{Q})}\big). \tag{4}$$

The HSVs quantify the joint controllability and observability of each state. Large values indicate state directions that both strongly affect the output and are strongly influenced by the input, while small values correspond to weakly contributing states.

2.1.3 BALANCED TRUNCATION

Balanced truncation is a MOR scheme leveraging the ordering of Hankel singular values to obtain a lower-dimensional approximation of the system, while guaranteeing stability and error bounds.

Consider a stable minimal balanced realization $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ of $\mathcal{G}$ with Gramian $\boldsymbol{W} = \text{diag}(\boldsymbol{\sigma}) = \text{diag}(\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ where $\boldsymbol{\Sigma}_1$ is diagonal of size r and $\boldsymbol{\Sigma}_2$ of size $n - r$, with the smallest entry in $\boldsymbol{\Sigma}_1$ larger than the largest in $\boldsymbol{\Sigma}_2$. The state space matrices can be rewritten as:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{1,1} & \boldsymbol{A}_{1,2} \\ \boldsymbol{A}_{2,1} & \boldsymbol{A}_{2,2} \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_1 \\ \boldsymbol{B}_2 \end{bmatrix}, \ \boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_1 & \boldsymbol{C}_2 \end{bmatrix}, \tag{5}$$

with $\boldsymbol{A}_{1,1} \in \mathbb{R}^{r \times r}$, $\boldsymbol{B}_1 \in \mathbb{R}^{r \times p}$, and $\boldsymbol{C}_1 \in \mathbb{R}^{q \times r}$.

It is well established that the reduced system $\hat{\mathcal{G}}$ defined by $\boldsymbol{A}_{1,1}, \boldsymbol{B}_1, \boldsymbol{C}_1, \boldsymbol{D}$ is stable and

$$||\mathcal{G} - \hat{\mathcal{G}}||_\infty \leq 2 \sum_{i=r+1}^{n} \boldsymbol{\sigma}_i. \tag{6}$$

Balanced truncation thus provides a principled way to reduce model order while controlling the approximation error in terms of discarded Hankel singular values. This makes it a central tool for simplifying state space models while preserving their dominant dynamics.

2.2 SPECTRAL STABILITY OF HERMITIAN MATRICES

In practice, training SSMs with gradient descent modifies the learned state matrices incrementally. Understanding how the downstream Hankel singular values shift under such perturbations is therefore critical to establish in-training reduction protocols. For Hermitian matrices, Weyl's theorem provides a powerful tool.

Let $\boldsymbol{W}$ and $\boldsymbol{W}'$ be Hermitian matrices of size n (*i.e.* symmetric for real-valued matrices), and let $\delta\boldsymbol{W} = \boldsymbol{W}' - \boldsymbol{W}$. Also, $\forall i \in [1, \cdots, n]$, let $\lambda_i(\boldsymbol{W})$ represent the i-th largest eigenvalue of $\boldsymbol{W}$. The ordering $\lambda_1(\boldsymbol{W}) \geq \cdots \geq \lambda_n(\boldsymbol{W})$ can always be established as the eigenvalues of Hermitian matrices are guaranteed to be real.

Theorem 2.7 (Weyl (1912))**.** *$\forall i \in [1, \cdots, n]$, $\lambda_i(\boldsymbol{W})$ is Lipschitz-continuous on the space of Hermitian matrices with operator norm:*

$$|\lambda_i(\boldsymbol{W}') - \lambda_i(\boldsymbol{W})| \leq \max_{i=1,\cdots,n} (|\lambda_i(\delta\boldsymbol{W})|) = \max(|\lambda_1(\delta\boldsymbol{W})|, |\lambda_n(\delta\boldsymbol{W})|). \tag{7}$$

In other words, each of the eigenvalues of $\boldsymbol{W}$ can at most fluctuate by the largest absolute eigenvalue of the perturbation $\delta\boldsymbol{W}$, providing a bound on spectral variations under Hermitian perturbations. This bound will turn out to be crucial for our in-training reduction scheme (see Section 3.2).

3 THE PROPOSED IN-TRAINING REDUCTION SCHEME

Our general pipeline (illustrated in Figure 1) is designed to be applicable to all types of SSMs as it surgically acts on the dynamical systems within SSM layers of models regardless of the choice of projections, non-linear activations, convolutions, skip connection, etc.

To achieve significant gains in training time, we aim to reduce the model's hidden state dimension where possible early on in training. Typically, we attempt to reduce SSM state dimensions at snapshots of the model obtained at fixed intervals at early stages of training (*e.g.* during learning rate warm-up). Section 3.2 provides justification for the validity of this approach.

3.1 CompreSSM: the algorithm

At an given training step, we proceed per block. For an input feature vector sequence $\boldsymbol{x} \in \mathbb{R}^{H\times L}$, where H is the inner dimension and L the sequence length, common SSM blocks contain either a single Multi-Input Multi-Output (MIMO) system that transforms the sequence $\boldsymbol{x} \in \mathbb{R}^{H\times L}$ to $\boldsymbol{y} \in \mathbb{R}^{H\times L}$, or H independent per-channel Single-Input Single-Output (SISO) systems that transform $\boldsymbol{x}^i \in \mathbb{R}^{1\times L}$ to $\boldsymbol{y}^i \in \mathbb{R}^{1\times L}$, for $i \in 1, \cdots, H$. In the latter case, we proceed per channel.

For a given block (and possibly a given channel index), the reduction algorithm works as follows:

1. Extract the discrete linear system matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ from the model weights. We denote by n the current order (rank) of the system.
2. Solve Equation 2 and Equation 3 (using Equation 14 if $\boldsymbol{A}$ is diagonal, as is the case for many modern SSMs) to obtain the Gramians $\boldsymbol{P}$ and $\boldsymbol{Q}$ respectively.
3. Compute the Hankel singular values $\boldsymbol{\sigma}$ via Equation 4.
4. Find the smallest rank r such that the top-r singular values account for a predetermined threshold $\tau \in [0, 1]$ of the total energy,
$$r = \min\Big\{k \in \{1, \cdots, n\} : \sum_{i=1}^{k} \sigma_i \;\geq\; \tau \sum_{i=1}^{n} \sigma_i\Big\}, \tag{8}$$
5. If the rank is smaller than a given fraction of the initial state dimension (i.e. the reduction is large enough to warrant the trouble), compute the balancing transformation matrix $\boldsymbol{T}$ from Theorem 2.6. Otherwise, leave the system unchanged.
6. Transform the original system to its diagonal balanced realization,
$$(\boldsymbol{A}_b, \boldsymbol{B}_b, \boldsymbol{C}_b) = (\boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}, \boldsymbol{T}^{-1}\boldsymbol{B}, \boldsymbol{C}\boldsymbol{T}) \tag{9}$$
7. Truncate the balanced system down to rank r (with a slight abuse of tensor slicing notation),
$$(\boldsymbol{A}_r, \boldsymbol{B}_r, \boldsymbol{C}_r) = (\boldsymbol{A}_b[:r, :r], \boldsymbol{B}_b[:r, :], \boldsymbol{C}_b[:, :r]) \tag{10}$$
8. Replace the model weights for the dynamical system matrices. Based on the architecture this might require diagonalizing the truncated system to ensure computational consistency.
$$(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) \leftarrow (\boldsymbol{A}_r, \boldsymbol{B}_r, \boldsymbol{C}_r) \tag{11}$$

Note that the above algorithm can not deliver improved small models if there is no clear correlation between state dimension and model performance. There furthermore should be enough time in between successive reduction steps for the model to recover from pruning. For more details, we refer the reader to Section 4.2.

The fact that CompreSSM applies an *in-training* compression scheme enables a significant increase in performance per training time. Indeed, we validate the speedup experimentally in Section 4.2.

3.2 In-Training Reduction

The validity of our proposed in-training truncation relies on several non-trivial properties, which we first motivate intuitively before formalizing. First, the method requires tracking how the relative importance of individual states evolves with training. Second, even if we can measure importance continuously, training dynamics may render initially insignificant dimensions crucial at later stages. Early truncation of such dimensions would therefore be undesirable. Consequently, it is necessary that the relative ordering of importance remains approximately stable, at least for the dimensions with low initial contribution. Finally, since our balanced truncation approach relies on the energy contribution of each dimension relative to the total system energy, it is desirable that the cumulative importance of the bottom-r dimensions does not increase substantially during training. Otherwise, dimensions could converge to a regime of near-equal importance, making early truncation unjustified even if the ordering is preserved.

To leverage Hankel singular value analysis during training, we must first establish a protocol for tracking individual state importance as the system evolves under gradient updates. This existence of such a protocol is non-trivial and its development is central to this work.

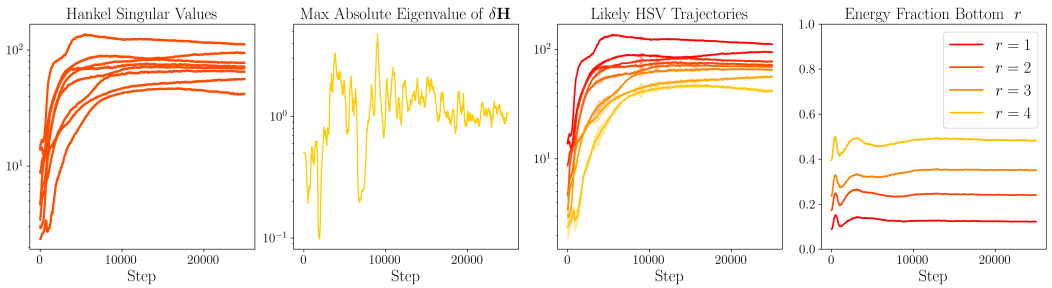


Figure 2: In-training per-step analysis of Hankel singular value dynamics for a single LRU block with state dimension of 8 on the MNIST dataset for the first 25k steps. The leftmost plot shows the raw HSVs (as a set). The middle-left plot depicts the maximum absolute eigenvalue of $\delta\boldsymbol{H}$ as described in Section 3.2. The middle-right plot overlays the maximum variation bound as an error margin around each HSV, with each shade now representing a highly probable path for a specific state dimension obtained by step by step linear sum assignment solving. The rightmost plot shows the relative contribution of the bottom r HSVs to the total energy.

Indeed, between gradient steps, model weights are updated according to the negative gradient of the loss with respect to the parameters. At the SSM level, this translates into the state matrices being incrementally updated such that a discrete system described by $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ becomes a *different* dynamical system $(\boldsymbol{A}', \boldsymbol{B}', \boldsymbol{C}')$, where

$$\boldsymbol{A}' = \boldsymbol{A} + \delta\boldsymbol{A}, \quad \boldsymbol{B}' = \boldsymbol{B} + \delta\boldsymbol{B}, \quad \boldsymbol{C}' = \boldsymbol{C} + \delta\boldsymbol{C}. \tag{12}$$

The omission of $\boldsymbol{D}$ is deliberate for both notational simplicity, but also since it is often fixed as a skip layer and not learned. Throughout this section, we adopt the convention that plain symbols denote pre-update quantities, while primed symbols denote their post-update counterparts.

Now, using the expressions of the controllability and observability Gramians given respectively by Equation 2 and Equation 3, one can see that both Gramians are continuous with respect to the gradient perturbation to the system matrices thus we can also consider,

$$\boldsymbol{P}' = \boldsymbol{P} + \delta\boldsymbol{P}, \quad \boldsymbol{Q}' = \boldsymbol{Q} + \delta\boldsymbol{Q}, \tag{13}$$

where $\delta\boldsymbol{P}$ is some continuous function of $(\delta\boldsymbol{A}, \delta\boldsymbol{B})$, and similarly $\delta\boldsymbol{Q}$ of $(\delta\boldsymbol{A}, \delta\boldsymbol{C})$.

Also recall that the Hankel singular values can be obtained as the square root of the eigenvalues of $\boldsymbol{PQ}$ (Equation 4). Generally, $\boldsymbol{PQ}$ need not be symmetric, but we use this form for computational efficiency. Noticing that $\boldsymbol{PQ}$ is similar to the symmetric positive definite matrix $\boldsymbol{P}^{1/2}\boldsymbol{Q}\boldsymbol{P}^{1/2}$, we let $\boldsymbol{H} = \sqrt{\boldsymbol{P}^{1/2}\boldsymbol{Q}\boldsymbol{P}^{1/2}}$. The matrix $\boldsymbol{H}$'s eigenvalues are exactly the Hankel singular values. In addition, by composition, $\boldsymbol{H}$ is continuous with respect to the perturbations to the system, so we write $\boldsymbol{H}' = \boldsymbol{H} + \delta\boldsymbol{H}$ with $\delta\boldsymbol{H}$ a continuous function of $(\delta\boldsymbol{A}, \delta\boldsymbol{B}, \delta\boldsymbol{C})$.

Lemma 3.1 (Continuity of Hankel singular values under training updates)**.** *By application of Weyl's Theorem 2.7 to the matrix $\boldsymbol{H}$ and its perturbation $\boldsymbol{H}'$, between gradient steps, each Hankel singular value can at most change by the largest absolute eigenvalue of $\delta\boldsymbol{H} = \boldsymbol{H}' - \boldsymbol{H}$.*

While the previous argument establishes that Hankel singular values evolve continuously with gradient updates, the remaining conditions—stability of relative ordering and low contribution of the bottom-r dimensions—cannot be guaranteed theoretically. However, empirical evidence strongly suggests that standard training dynamics are favorable in practice.

We examine the case of a single LRU block trained on the MNIST dataset with state dimension of 8 for visual clarity in Figure 2 (and provide plots for larger models and datasets in Appendix C.2). To keep track of the state dimension to HSV value correspondence, we overlay the maximum absolute eigenvalue of $\delta\boldsymbol{H}$, on top of each HSV. Empirically, this allows us to robustly identify probable HSV trajectories as the continuity bound is consistently small enough to ensure each eigenvalue is clearly isolated, with minimal bound overlaps and rare gradual HSV relative order crossings occurring (prediction of evolution established via linear sum assignment solution in such cases). Furthermore, the cumulative contribution of the bottom-r HSVs stabilizes rapidly. Indeed, after a

small initial number of steps, we observe that both the ordering of singular values remains constant, and dimensions of low importance seldom gain substantial relative energy during training.

In sum, these observations provide empirical justification for early in-training truncation: dimensions identified as negligible at early stages typically remain so throughout training. Consequently, truncation decisions made during training rarely conflict with the final importance ranking, making our approach both effective and robust in practice.

4 Experiments

4.1 Experimental Setup

In order to empirically validate in-training balanced truncation, we train a linear recurrent unit (LRU) Orvieto et al. (2023) on datasets of different complexity, ranging from MNIST to tasks from the long range arena (LRA) Tay et al. (2020).

We use the same training pipeline as Rusch & Rus (2025); Walker et al. (2025), accounting for additional quirks of LRU training like a learning rate factor for the sequence mixing layer. The LRA dataloaders are borrowed from Smith et al. (2022), the hyperparameters largely taken from (Orvieto et al., 2023, Table 10). We summarize them in Table 2.

Reduction starts from the full model order reported in column 4 of Table 2. For all datasets but IMDB and MNIST, we attempt four equidistant reduction steps during the warm-up period of the learning rate, which equals 10% of the total steps. Doing so ensures maximal speed up potential for the subsequent 90% of training, while also staying robust to large early training .

As MNIST is trained without learning rate decay, we attempt truncation during all of training. For IMDB, we include a waiting stage and attempt reduction inside a smaller time window. The details can be found in Appendix B.1. Generally, reductions are only executed if the reduced dimension is less than 95% of the current state dimension.

We train non-reduced models with each block initialized at the average final order of the compressed ones to allow for a fair comparison. To increase our baseline statistics and further establish correlations between state dimension and model performance, we train further models with different state dimensions.

4.2 Empirical Results

We repeat all experiments using five different random seeds and report mean top-3 as well as top-1 performance. Table 1 contains the top-3 results. The top-1 results can be found in the appendix (Table 3). The state dimensions reported for multi-block models are the averages of the SSM orders per-block. It is accompanied by Figure 4, which presents the performance visually and contains further non-reduced benchmark models.

For some datasets, for example AAN or Pathfinder, our baseline experiments reveal a small correlation between state dimension and model performance, given the other model parameters taken from Orvieto et al. (2023) (see entries 3 and 5 in Table 1 or figures 4c and 4e). On these datasets, CompreSSM can not deliver better performance for small models.

However, on datasets where state dimension does clearly correlate with model performance, CompreSSM improves small model performance. On CIFAR10, for example, compressed model performance almost stays constant as a function of the reduction tolerance (and thus for different state dimensions), while the non-compressed counterparts exhibit a approximately 10% performance drop (see entry 1 in Table 1 or Figure 3a). The MNIST results paint a similar picture (entry 5 in Table 1 or Figure 4a).

Similarly, on ListOps, the performance of non-compressed models drops significantly for state dimensions smaller than 120 (see Figure 4f). While the compressed models perform on par with their uncompressed counterparts for larger state dimensions, smaller models outperform the baseline.

On Pathfinder, we can observe a similar trend (Figure 4e), where the unreduced model performance does not show a strong correlation with the state dimension. Just for the smallest state dimension,

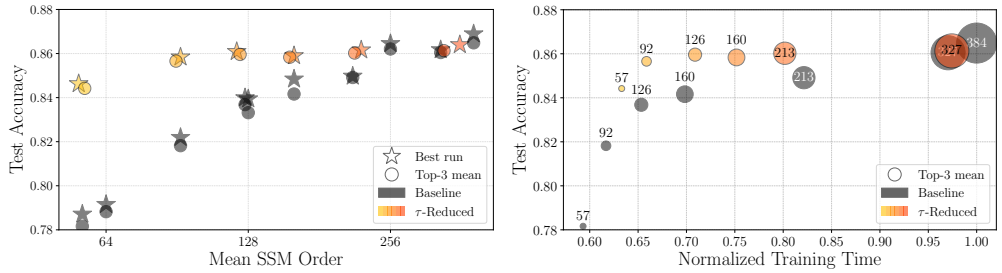
Table 1: Average final state dimension (mean ± std) and Top-3 runs mean performance with/without reduction for LRU under different tolerances τ.

Dataset	Metric	$\tau = 1.5 \cdot 10^{-1}$	$\tau = 1 \cdot 10^{-1}$	$\tau = 7 \cdot 10^{-2}$	$\tau = 5 \cdot 10^{-2}$	$\tau = 3 \cdot 10^{-2}$	$\tau = 2 \cdot 10^{-2}$	$\tau = 0$
CIFAR10	State dim	57.4 ± 1.5	92.6 ± 4.2	126.0 ± 4.0	160.8 ± 5.4	213.6 ± 6.1	327.2 ± 16.0	384
	CompreSSM	$\mathbf{84.4 \pm 0.2}$	$\mathbf{85.7 \pm 0.1}$	$\mathbf{86.0 \pm 0.1}$	$\mathbf{85.8 \pm 0.1}$	$\mathbf{86.0 \pm 0.2}$	$\mathbf{86.1 \pm 0.2}$	-
	Baseline	78.2 ± 0.7	81.8 ± 0.3	83.7 ± 0.2	84.2 ± 0.5	84.9 ± 0.0	86.0 ± 0.1	86.5 ± 0.3
ListOps	State dim	56.8 ± 3.4	81.8 ± 4.9	109.8 ± 3.9	135.4 ± 6.8	167.6 ± 5.7	213.8 ± 28.0	256
	CompreSSM	$\mathbf{48.3 \pm 0.7}$	$\mathbf{51.8 \pm 0.9}$	48.2 ± 1.1	47.5 ± 1.6	$\mathbf{49.2 \pm 0.3}$	47.1 ± 1.4	-
	Baseline	43.4 ± 0.4	46.3 ± 0.5	$\mathbf{49.4 \pm 1.8}$	$\mathbf{49.2 \pm 0.7}$	48.2 ± 2.1	$\mathbf{47.6 \pm 1.7}$	49.7 ± 0.8
AAN	State dim	53.6 ± 1.9	84.4 ± 1.4	111.0 ± 2.0	136.6 ± 2.9	170.0 ± 2.4	203.2 ± 13.7	256
	CompreSSM	87.2 ± 0.3	87.5 ± 0.1	87.4 ± 0.3	87.2 ± 0.0	87.6 ± 0.3	87.9 ± 0.2	-
	Baseline	$\mathbf{87.5 \pm 0.3}$	$\mathbf{87.9 \pm 0.3}$	$\mathbf{87.8 \pm 0.2}$	$\mathbf{87.8 \pm 0.5}$	$\mathbf{87.3 \pm 0.4}$	87.4 ± 0.5	$\mathbf{87.3 \pm 0.3}$
IMDB	State dim	95.0 ± 2.3	119.6 ± 2.2	136.8 ± 1.9	150.4 ± 1.2	165.0 ± 1.3	192.0 ± 0.0	192.0
	CompreSSM	82.2 ± 0.2	82.8 ± 0.1	83.7 ± 0.4	83.8 ± 0.3	84.1 ± 0.4	84.4 ± 0.2	-
	Baseline	$\mathbf{82.7 \pm 0.1}$	$\mathbf{83.5 \pm 0.1}$	$\mathbf{83.7 \pm 0.0}$	$\mathbf{84.0 \pm 0.4}$	$\mathbf{84.3 \pm 0.0}$	$\mathbf{84.5 \pm 0.1}$	84.7 ± 0.1
Pathfinder	State dim	34.6 ± 1.9	51.2 ± 1.7	65.6 ± 2.3	81.2 ± 1.6	105.0 ± 2.1	129.8 ± 5.2	256
	CompreSSM	96.6 ± 1.3	$\mathbf{97.9 \pm 0.1}$	97.6 ± 0.5	97.8 ± 0.4	98.0 ± 0.0	98.0 ± 0.1	-
	Baseline	$\mathbf{97.3 \pm 0.2}$	$\mathbf{97.9 \pm 0.1}$	$\mathbf{98.0 \pm 0.1}$	$\mathbf{98.1 \pm 0.0}$	$\mathbf{98.2 \pm 0.0}$	$\mathbf{98.1 \pm 0.1}$	98.3 ± 0.1
		$\tau = 4{\cdot}10^{-2}$	$\tau = 2{\cdot}10^{-2}$	$\tau = 1{\cdot}10^{-2}$	$\tau = 5{\cdot}10^{-3}$	$\tau = 2{\cdot}10^{-3}$	$\tau = 1{\cdot}10^{-3}$	$\tau = 0$
MNIST	State dim	12.7 ± 3.0	27.6 ± 1.8	46.8 ± 3.2	76.3 ± 7.5	148.1 ± 9.8	191.4 ± 4.7	256
	CompreSSM	$\mathbf{95.9 \pm 0.2}$	$\mathbf{96.9 \pm 0.0}$	$\mathbf{96.9 \pm 0.1}$	$\mathbf{96.9 \pm 0.1}$	97.0 ± 0.1	$\mathbf{97.2 \pm 0.3}$	-
	Baseline	92.6 ± 0.5	96.0 ± 0.2	95.9 ± 0.1	96.4 ± 0.2	$\mathbf{97.3 \pm 0.2}$	$\mathbf{97.3 \pm 0.1}$	97.3 ± 0.1

the unreduced model performs sees a small drop in performance and is outperformed by the top-1 reduced model.

The IMDB results reveal the importance of the prerequisites mentioned in Section 3.1 (see Figure 4d). Even after significantly reducing the parameters and increasing the droprate (see Appendix B), the unreduced only learn for roughly $8k$ steps before overfitting. However, in order to do in-training balanced truncation, the actual training phase needs to be long enough to allow for a couple of reduction steps with a large enough spacing so that the model can follow the training dynamics for a while without being pruned. Indeed, for non-aggressive pruning (that is, pruning with small tolerance τ), which requires less recovery time, the top reduced models often outperform the baseline.

As mentioned previously, an advantage of CompreSSM is that it comes with a training speedup; as the state dimensions get pruned, training speeds up. Indeed, Figure 3b demonstrates this effect on CIFAR10. In the case of state dimension 57, CompreSSM provides a 7% increase in test accuracy for just 4% more training time.



(a) Test accuracy vs. Final state dimension

(b) Test accuracy vs. Training time

Figure 3: Subfigure (a) shows the performance of different models trained on CIFAR10 as a function of the state dimension. Grey data indicates non-reduced models, and the shades of orange correspond to reduced models, with tolerance decreasing with redness. The circles represents the top-3 mean, while the star corresponds to the top-1 model. Subfigure (b) shows top-3 performance versus the normalized average training time. Marker diameter is proportional to the final model order (also annotated) and in-between models are omitted for visual decluttering.

5 Related Work

Model compression techniques in machine learning. The question of model compression has received considerable attention in the literature (Deng et al., 2020; Zhu et al., 2024), leveraging various techniques (and mixtures thereof) such as pruning (Han et al., 2015; Frantar & Alistarh, 2023), which removes redundant parameters to reduce network size; quantization (Xiao et al., 2022; Gholami et al., 2021), which compresses models by lowering numerical precision; low-rank factorization (Lin et al., 2024), which exploits structure in weight matrices to reduce dimensionality; and knowledge distillation (Hinton et al., 2015; Gou et al., 2020), where a smaller model learns to mimic a larger one.

In-training vs. post-training paradigms. Techniques such as quantization-aware training (Choi et al., 2018; Zhang et al., 2018), and dynamic pruning (Hoefler et al., 2021; Wimmer et al., 2022) perform in-training compression during the optimization process. In contrast, most methods like Deep Compression (Han et al., 2015) follow a post-training paradigm, applying pruning or quantization after convergence and relying on retraining or fine-tuning to recover accuracy.

Compression in SSMs. Work on compressing SSMs has primarily explored quantization. Post-training quantization has been applied to stabilize SSM inference under 8-bit constraints (Abreu et al., 2024; Chiang et al., 2024), while quantization-aware training has been used to maintain accuracy below 8 bits (Zhao et al., 2025) and to improve robustness for deployment on specialized or analog hardware (Siegel et al., 2024; 2025).

By contrast, control-theoretic MOR approaches have been applied to diagonal S4 layers, but only as a post-hoc step to initialize retraining (Ezoe & Sato, 2024). Elsewhere, regularizers based on the Hankel nuclear norm or modal ℓ_1 have been shown to encourage parsimonious state representations during training, though without explicit truncation events, and with a price to pay in terms of optimal performance (Forgione et al., 2024). Finally, $\mathcal{H}_2$-optimal reductions have been proposed as a competitor to balanced truncation in offline SSM MOR settings (Sakamoto & Sato, 2025).

To the best of our knowledge, our contribution is the first to propose a principled in-training model order reduction method applicable to a broad spectrum of SSMs.

6 Conclusion

In conclusion, we proposed a novel framework for principled compression of SSMs during training called CompreSSM. Drawing on classical control theory, we base our approach on balanced truncation. In particular, we show that Hankel singular values (HSVs), the corner stone of balanced truncation, preserve the rank of their dominant dimensions during training, which allows for safely truncating dimensions associated with smaller HSVs.

We test CompreSSM by training LRUs on a range of tasks of varying complexity. Provided there is a correlation between state dimension and model performance as well as enough training steps in between two reduction steps, we verify empirically that compressed models outperform their uncompressed counterparts while delivering better performance per unit of training time.

Finally, we provide an outlook on how to do in-training balanced truncation for linear time-varying systems. In this selective case, we propose averaging the dynamics over the input space and applying the reduction scheme to this time-independent system. For future work, we would like to extend CompreSSM to linear self-attention models, e.g., Gated Linear Attention Yang et al. (2023), Mamba2 Dao & Gu (2024), and Gated DeltaNet Yang et al. (2024), which is another class of models based on linear time-varying systems.

Reproducibility Statement

We provide an anonymized supplementary code package that implements CompreSSM for the LRU architecture with configuration runs used to produce the results in this paper. A public repository with identical code will be made public after review.

Use of Large Language Models (LLMs)

No original methods, model designs, or ideas originate from the use of LLMs. They have been put to use for LaTeX formatting, grammar and syntax and coding assistance only. The authors are the sole responsible parties for the contents of the work. LLMs are not eligible for authorship.

References

Steven Abreu, Jens Egholm Pedersen, Kade Heckel, and Alessandro Pierro. Q-S5: Towards Quantized State Space Models, 2024.

Athanasios C. Antoulas. *Approximation of Large-Scale Dynamical Systems*, chapter 7. Balancing and Balanced Approximations, pp. 207–247. SIAM Advances in Design and Control, 2005. doi: 10.1137/1.9780898718713.ch7. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898718713.ch7.

Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, 3 edition, 1999. ISBN 978-0-19-511777-6. URL https://app.knovel.com/hotlink/pdf/id:kt008LOK0F/linear-system-theory/frontmatter.

Hung-Yueh Chiang, Chi-Chih Chang, N. Frumkin, Kai-Chiang Wu, and Diana Marculescu. Quamba: a Post-Training Quantization Recipe for Selective State Space Models, 2024.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, P. Chuang, Vijayalakshmi Srinivasan, and K. Gopalakrishnan. PACT: Parameterized Clipping Activation for Quantized Neural Networks, 2018.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized Models and Efficient Algorithms through Structured State SpaceDuality. *arXiv preprint arXiv:2405.21060*, 2024.

By Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey, 2020.

Haruka Ezoe and Kazuhiro Sato. Model Compression Method for S4 With Diagonal State Space Layers Using Balanced Truncation, 2024.

Marco Forgione, Manas Mejari, and Dario Piga. Model Order Reduction of Deep Structured State-Space Models: a System-Theoretic Approach, 2024. URL https://arxiv.org/abs/2403.14833.

Elias Frantar and Dan Alistarh. SparseGPT: Massive Language Models can be Accurately Pruned in One-Shot, 2023.

A. Gholami, Sehoon Kim, Zhen Dong, Z. Yao, Michael W. Mahoney, and K. Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference, 2021.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's Raw! audio Generation with State-Space Models. In *International Conference on Machine Learning*, pp. 7616–7633. PMLR, 2022.

Jianping Gou, B. Yu, S. Maybank, and D. Tao. Knowledge Distillation: a Survey, 2020.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Song Han, Huizi Mao, and W. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, 2015.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid Structural State-Space Models. *arXiv preprint arXiv:2209.12951*, 2022.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*, 2015.

T. Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in Deep Learning: Pruning and growth for Efficient Inference and Training in Neural Networks, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-Rank Adaptation of Large Language Models. *ICLR*, 1(2):3, 2022.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

Rudolph Emil Kalman. A new Approach to Linear Filtering and Prediction Problems, 1960.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters for Efficient Convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Chi-Heng Lin, Shangqian Gao, J. Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. MoDeGPT: Modular Decomposition for Large Language Model Compression, 2024.

Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. S4nd: Modeling Images and Videos as Multidimensional Signals with State Spaces. *Advances in neural information processing systems*, 35:2846–2861, 2022.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting Recurrent Neural Networks for Long Sequences. In *International Conference on Machine Learning*, pp. 26670–26698. PMLR, 2023.

T Konstantin Rusch and Daniela Rus. Oscillatory State-Space Models. In *International Conference on Learning Representations*, 2025.

Hiroki Sakamoto and Kazuhiro Sato. Compression Method for Deep Diagonal State-Space Model Based on H2 Optimal Reduction, 2025.

Sebastian Siegel, Ming-Jay Yang, and John Paul Strachan. IMSSA: Deploying Modern State-Space Models on Memristive In-Memory Compute Hardware, 2024.

Sebastian Siegel, Ming-Jay Yang, Younes Bouhadjar, Maxime Fabre, Emre Neftci, and John Paul Strachan. QS4D: Quantization-Aware Training for Efficient Hardware Deployment of Structured State-Space Sequential Models, 2025.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified State Space Layers for Sequence Modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long-Range Arena: A Benchmark for Efficient Transformers. *arXiv preprint arXiv:2011.04006*, 2020.

A Vaswani. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017.

Benjamin Walker, Lingyi Yang, Nicola Muca Cirone, Cristopher Salvi, and Terry Lyons. Structured Linear CDEs: Maximally Expressive and Parallel-in-Time Sequence Models, May 2025. URL https://arxiv.org/abs/2505.17761.

Hermann Von Weyl. Das Asymptotische Verteilungsgesetz der Eigenwerte Linearer Partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). *Mathematische Annalen*, 71:441–479, 1912. URL https://api.semanticscholar.org/CorpusID:120278241.

Paul Wimmer, Jens Mehnert, and A. Condurache. Dimensionality Reduced Training by Pruning and Freezing Parts of a Deep Neural Network: a Survey, 2022.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models, 2022.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated Linear Attention Transformers with Hardware-Efficient Training. *arXiv preprint arXiv:2312.06635*, 2023.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated Delta Networks: Improving Mamba2 with Delta Rule. *arXiv preprint arXiv:2412.06464*, 2024.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and G. Hua. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks, 2018.

Leo Zhao, Tristan Torchet, M. Payvand, Laura Kriener, and Filippo Moro. Quantizing Small-Scale State-Space Models for Edge AI, 2025.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A Survey on Model Compression for Large Language Models. *Transactions of the Association for Computational Linguistics*, 12: 1556–1577, 11 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00704. URL https://doi.org/10.1162/tacl_a_00704.

A Appendix

A.1 Solving The Lyapunov Equations for Diagonal SSMs

For SSMs with diagonal state transition matrix $\boldsymbol{A} = \text{diag}(a_1, \ldots, a_n)$, which covers a lot of SSMs used today such as LRU Orvieto et al. (2023) and S5 Smith et al. (2022), Equation 2 and Equation 3 admit a simple, entry-wise closed-form solution:

$$\boldsymbol{P}_{ij} = \frac{\left(\boldsymbol{B}\boldsymbol{B}^\top\right)_{ij}}{1 - a_i a_j}, \quad \boldsymbol{Q}_{ij} = \frac{\left(\boldsymbol{C}\boldsymbol{C}^\top\right)_{ij}}{1 - a_i a_j} \quad \forall 1 \leq i, j \leq n \tag{14}$$

In the non-diagonal case, one can either solve the Lyapunov equations by vectorization or use the argument put forward by Orvieto et al. (2023) and realize that every state transition matrix $\boldsymbol{A} \in \mathbb{R}^{n,n}$ can be diagonalized over $\mathbb{C}$ up to a small perturbation.

A.2 Selective SSM case

Selective SSMs are built with dynamical systems that fall under the linear parameter varying (LPV) framework, with the parameter being the layer input. Such systems are also referred to as Linear Input Varying (LIV) and their general case state equations are given by,

$$\boldsymbol{h}(k+1) = \boldsymbol{A}(\boldsymbol{x}(k))\,\boldsymbol{h}(k) + \boldsymbol{B}(\boldsymbol{x}(k))\,\boldsymbol{x}(k)\,, \quad \boldsymbol{h}(0) = \boldsymbol{h}_0 \tag{15}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}(\boldsymbol{x}(k))\,\boldsymbol{h}(k) + \boldsymbol{D}(\boldsymbol{x}(k))\,\boldsymbol{x}(k), \tag{16}$$

For these systems, the controllability and observability Gramians are no longer stationary. In particular, for each possible input $\boldsymbol{x} \in \mathcal{X}$, one would in principle need to solve a set of Lyapunov inequalities of the form

$$\boldsymbol{A}(\boldsymbol{x})\boldsymbol{P}(\boldsymbol{x})\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{x}) - \boldsymbol{P}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{B}^{\mathrm{T}}(\boldsymbol{x}) \leq 0, \tag{17}$$

$$\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{Q}(\boldsymbol{x})\boldsymbol{A}(\boldsymbol{x}) - \boldsymbol{Q}(\boldsymbol{x}) + \boldsymbol{C}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{C}(\boldsymbol{x}) \leq 0, \tag{18}$$

so that the Gramians $\boldsymbol{P}$ and $\boldsymbol{Q}$ are input-dependent.

In practice, solving for fully input-dependent Gramians and applying the subsequent per input reduction is clearly neither computationally tractable nor practical. A common simplification is to seek input-invariant Gramians $\boldsymbol{P}, \boldsymbol{Q}$ that satisfy the inequalities for all $\boldsymbol{x} \in \mathcal{X}$; this reduces the problem to an LTI-like Lyapunov condition over all inputs, which can still be expensive for high-dimensional $\mathcal{X}$, when such a solution even exists. In practice this is still too constraining.

A cheaper alternative is simply averaging the dynamics over the input space:

$$\bar{\boldsymbol{A}} = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{A}(\boldsymbol{x}), \quad \bar{\boldsymbol{B}} = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{B}(\boldsymbol{x}), \quad \bar{\boldsymbol{C}} = \frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{C}(\boldsymbol{x}), \tag{19}$$

The caveat is that the mean system may not be stable, controllable, or observable; one may therefore need to regularize the mean matrices to satisfy these assumptions before applying a single global reduction based on the LTI approach.

B Experimental Details

The hyperparameters we used can be found in Table 2. For all but one LRA tasks we use the same ones as reported by (Orvieto et al., 2023, Table 10). The exception is IMDB, which we observed to overfit massively with the given hyperparameters. We mitigate this issue by increasing dropout and reducing the total number of layers.

On LRA tasks, the learning rate is warmed up from 10^{-7} to 10^{-3} for 10% of the total steps, before it is cosine-decayed back to 10^{-7}. For MNIST, the learning rate is fixed at $4 \cdot 10^{-4}$ for the entirety of training.

Table 2: Hyperparameters for LRU experiments. "h" refers to the dimension of the hidden state, "n" to the state space dimension, "Batch" to the batch size and "LR Factor" to the learning rate factor applied to the sequence mixer Orvieto et al. (2023).

Task	Depth	h	n	Steps	Batch	LR Factor	Weight Decay	Dropout
MNIST	1	8	256	$200k$	50	-	-	0.1
CIFAR10	6	512	384	$180k$	50	0.25	0.05	0.1
ListOps	6	128	256	$80k$	32	0.5	0.05	0.0
IMDB	1	256	192	$50k$	32	0.1	0.05	0.1
AAN	6	128	256	$100k$	64	0.5	0.05	0.1
Pathfinder	6	192	256	$500k$	64	0.25	0.05	0.0

B.1 Reduction Details

We find that LRU overfits on IMDB, even after reducing the number of parameters by 6 and doubling the dropout rate compared to Orvieto et al. (2023). Training LRU on this dataset, we furthermore observe an initial training period in which the loss plateaus. Just on IMDB, we thus wait for an initial $1k$ steps before doing the balanced truncation. Instead of doing 4 reduction steps until the end of warmup, we also just do 2 until $3k$ steps in order to give the model time to recover before entering the overfitting regime.

C Additional Results

C.1 Performance

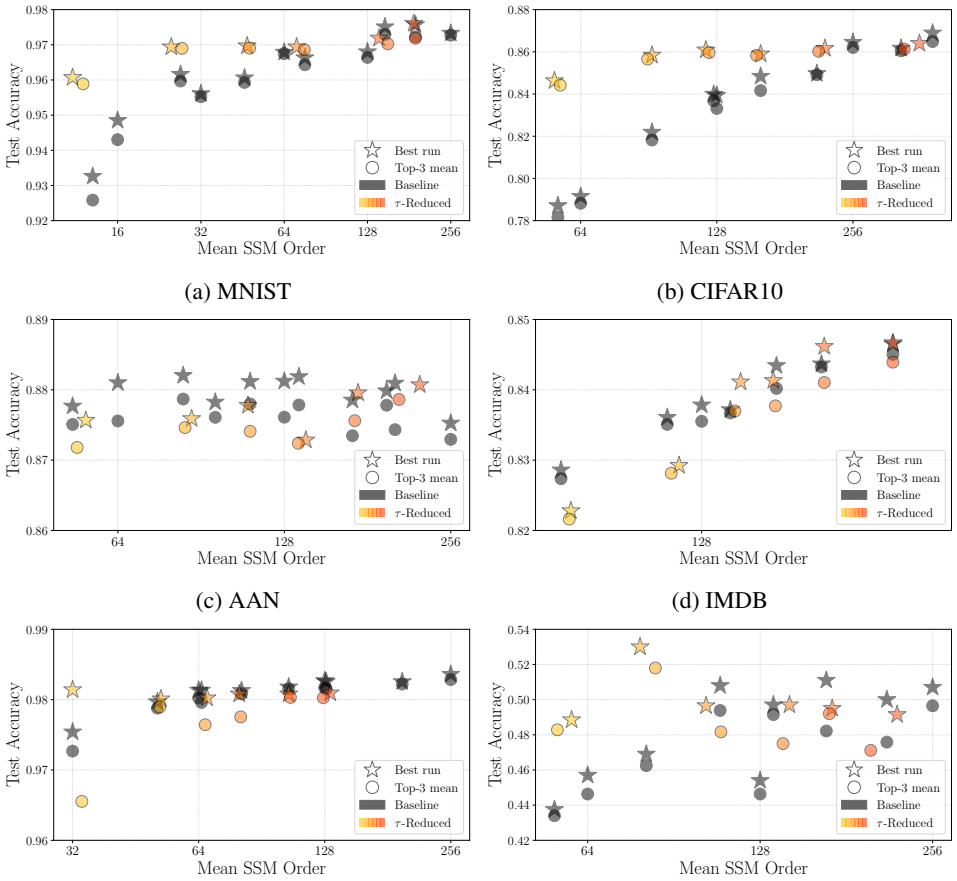
In Figure 4 we provide the state dimension vs test performance plots for all datasets.

Table 3: Final state dimension (mean $\pm$ std) and MAX performance with/without reduction for LRU under different tolerances τ.

Dataset	Metric	$\tau = 1.5 \cdot 10^{-1}$	$\tau = 1 \cdot 10^{-1}$	$\tau = 7 \cdot 10^{-2}$	$\tau = 5 \cdot 10^{-2}$	$\tau = 3 \cdot 10^{-2}$	$\tau = 2 \cdot 10^{-2}$	$\tau = 0$
CIFAR10	State dim	57.4 ± 1.5	92.6 ± 4.2	126.0 ± 4.0	160.8 ± 5.4	213.6 ± 6.1	327.2 ± 16.0	384
	CompreSSM	**84.6**	**85.8**	**86.1**	**85.9**	**86.2**	**86.4**	-
	Baseline	78.7	82.2	84.0	84.8	85.0	86.2	86.9
ListOps	State dim	56.8 ± 3.4	81.8 ± 4.9	109.8 ± 3.9	135.4 ± 6.8	167.6 ± 5.7	213.8 ± 28.0	256.0 ± 0.0
	CompreSSM	**48.9**	**53.0**	49.7	**49.7**	49.5	49.2	-
	Baseline	43.8	46.9	**50.8**	**49.7**	**51.1**	**50.0**	50.7
AAN	State dim	53.6 ± 1.9	84.4 ± 1.4	111.0 ± 2.0	136.6 ± 2.9	170.0 ± 2.4	203.2 ± 13.7	256
	CompreSSM	87.6	87.6	87.8	87.3	**88.0**	**88.1**	-
	Baseline	**87.8**	**88.2**	**88.1**	**88.2**	87.9	**88.1**	87.5
IMDB	State dim	95.0 ± 2.3	119.6 ± 2.2	136.8 ± 1.9	150.4 ± 1.2	165.0 ± 1.3	192.0 ± 0.0	192
	CompreSSM	82.3	82.9	**84.1**	84.1	**84.6**	**84.7**	-
	Baseline	**82.9**	**83.6**	83.7	**84.3**	84.4	**84.7**	84.7
Pathfinder	State dim	34.6 ± 1.9	51.2 ± 1.7	65.6 ± 2.3	81.2 ± 1.6	105.0 ± 2.1	129.8 ± 5.2	256
	CompreSSM	**98.1**	**98.0**	98.0	**98.1**	98.1	98.1	-
	Baseline	97.5	**98.0**	**98.1**	**98.1**	**98.2**	**98.3**	98.4
		$\tau = 3 \cdot 10^{-2}$	$\tau = 2 \cdot 10^{-2}$	$\tau = 1 \cdot 10^{-2}$	$\tau = 5 \cdot 10^{-3}$	$\tau = 2 \cdot 10^{-3}$	$\tau = 1 \cdot 10^{-3}$	$\tau = 0$
MNIST	Dim ($\pm$ std)	0.0 ± 0.0	27.6 ± 1.8	46.8 ± 3.2	76.3 ± 7.5	148.1 ± 9.8	191.4 ± 4.7	256
	CompreSSM	0.0	96.2	97.0	96.9	97.2	97.6	-
	Baseline	0.0	96.9	96.1	96.6	97.5	97.6	97.3

C.2 Empirical in-training Hankel stability

In Figures 5, 6, 7, and 8, we provide additional empirical evidence showing the validity of assumptions made on the dynamics of HSVs during training for multiple experiments with various number of layers and SSM state dimensions. The observations regarding a consistent ordering after a small number of training steps, and contributions of smaller values hold in all cases. Note that, unlike the analysis in the main text, which is established at each gradient step, we inspect HSVs at larger intervals on the order of thousands of steps. This leads to the computation of the exact continuity



(a) MNIST (b) CIFAR10

(c) AAN (d) IMDB

(e) Pathfinder (f) ListOps

Figure 4: Test performance vs. final state dimension for all our experiments. Stars correspond to best performance, circles to the mean of the top-3 runs. Grey shapes correspond to non-reduced models, and the shades of orange to reduced models, with tolerance decreasing with redness.

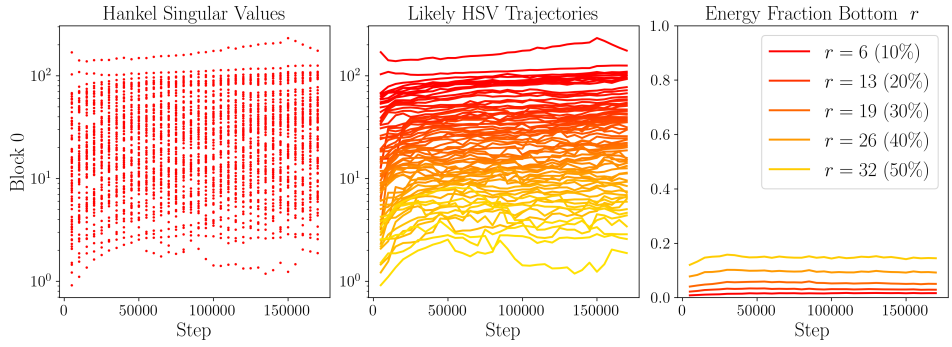


Figure 5: Single LRU block with state dimension of 64 on the MNIST dataset.

bounds becoming extremely noisy as perturbations grow. Nevertheless, we use linear sum assignment tracking as our most likely guess of HSV evolution as before.

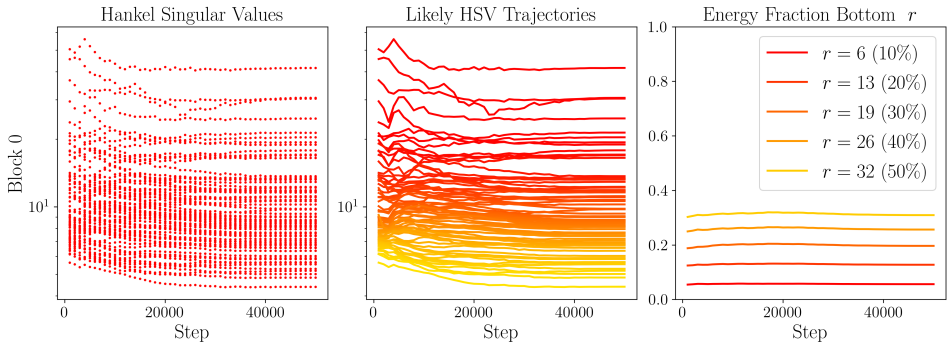


Figure 6: Single LRU block with state dimension of 64 on the IMDB dataset.

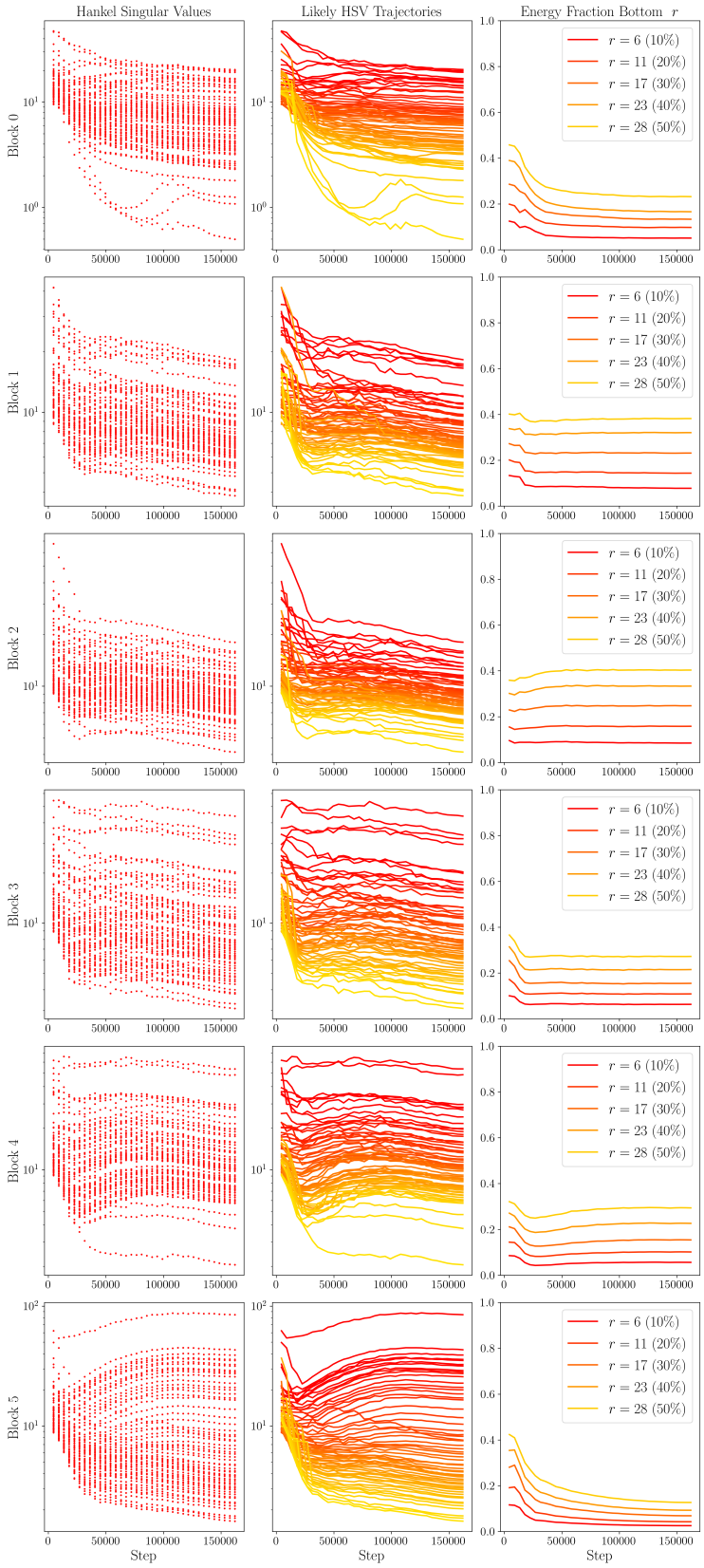


Figure 7: Six LRU blocks with state dimension of 57 on the CIFAR10 dataset.

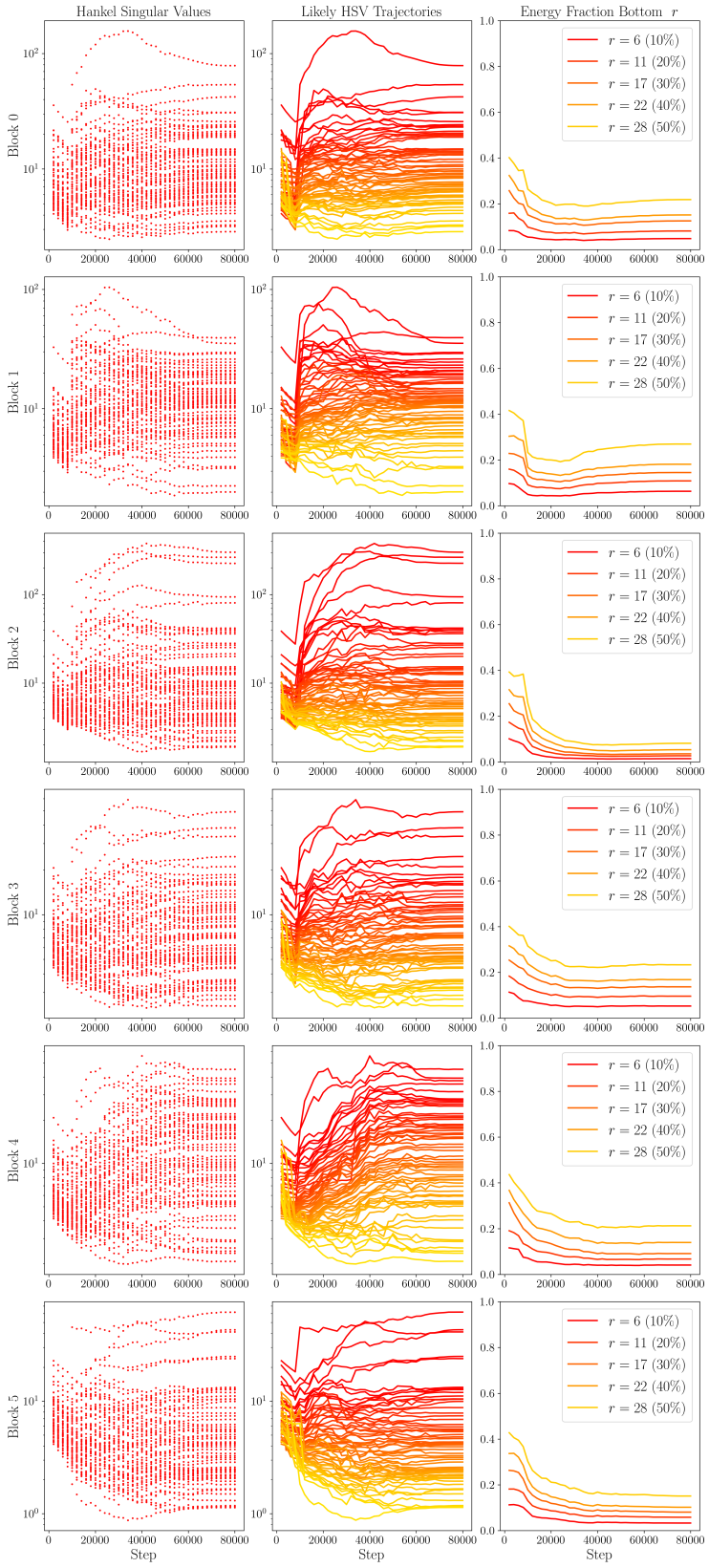


Figure 8: Six LRU blocks with state dimension of 56 on the ListOps dataset.