State-Free Inference of State-Space Models: The *Transfer Function* Approach

Rom N. Parnichkun [1 2 *] **Stefano Massaroli** [2 3 *] **Alessandro Moro** [1 *]
Jimmy T.H. Smith [2 4] **Ramin Hasani** [2 5] **Mathias Lechner** [2 5] **Qi An** [1]
Christopher Ré [4] **Hajime Asama** [1] **Stefano Ermon** [4] **Taiji Suzuki** [1 3]
Michael Poli [2 4 †] **Atsushi Yamashita** [1 †]

Abstract

We approach designing a state-space model for deep learning applications through its dual representation, the *transfer function*, and uncover a highly efficient sequence parallel inference algorithm that is *state-free*: unlike other proposed algorithms, state-free inference does not incur any significant memory or computational cost with an increase in state size. We achieve this using properties of the proposed frequency domain transfer function parametrization, which enables direct computation of its corresponding convolutional kernel's spectrum via a single Fast Fourier Transform. Our experimental results across multiple sequence lengths and state sizes illustrates, on average, a 35% training speed improvement over S4 layers – parametrized in time-domain – on the Long Range Arena benchmark, while delivering state-of-the-art downstream performances over other attention-free approaches. Moreover, we report improved perplexity in language modeling over a long convolutional Hyena baseline, by simply introducing our transfer function parametrization. Our code is available at https://github.com/rukelire/RTF.

1. Introduction

Central to the success of a certain class of sequence modeling layers are linear recurrences, which unlike the nonlinear case (Hochreiter & Schmidhuber, 1997; Chung et al., 2014; Kidger et al., 2020; Massaroli et al., 2021), are compatible with exact sequence parallel algorithms i.e., parallel scans (Blelloch, 1990; Martin & Cundy, 2018; Smith et al., 2023; Gu & Dao, 2023; Katsch, 2023), or (with time-invariance) the Fast Fourier Transform (FFT) (Gu et al., 2022b;a; Zhang et al., 2023). Such recurrent layers, often referred to in deep learning simply as *state-space models*, depending on their parametrization, also boast efficient constant time and memory autoregressive inference, lowering latency and memory costs.

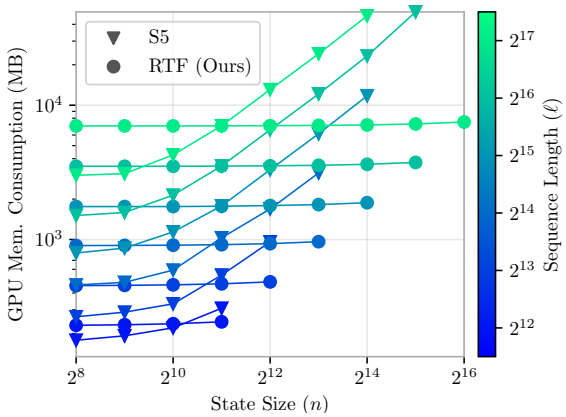

Figure 1: An illustration depicting the scaling of memory consumption on a scan-based algorithm (S5) and the proposed state-free inference algorithm denoted as RTF. We note that with larger state sizes, inference with S5 becomes prohibitively memory-intensive.

Despite recent advancements, current SSMs exhibit certain limitations that this paper aims to address.

With the goal of enabling parallel inference, many algorithms such as S5 (Smith et al., 2023), LRU (Orvieto et al., 2023) S4 (Gu et al., 2022b) and DSS (Gupta et al., 2022) employ a modal (diagonal) SSM representation, wherein the state transition matrix A is diagonal, potentially limiting the model's expressive capacity for a given state dimension. Additionally, along with Mamba (Gu & Dao, 2023), S5 and LRU rely on the parallel scan directive (Martin &

*Equal contribution †Equal senior authorship [1]The University of Tokyo [2]Liquid AI [3]RIKEN [4]Stanford University [5]Massachusetts Institute of Technology. Correspondence to: Rom N. Parnichkun <parnichkun@robot.t.u-tokyo.ac.jp>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

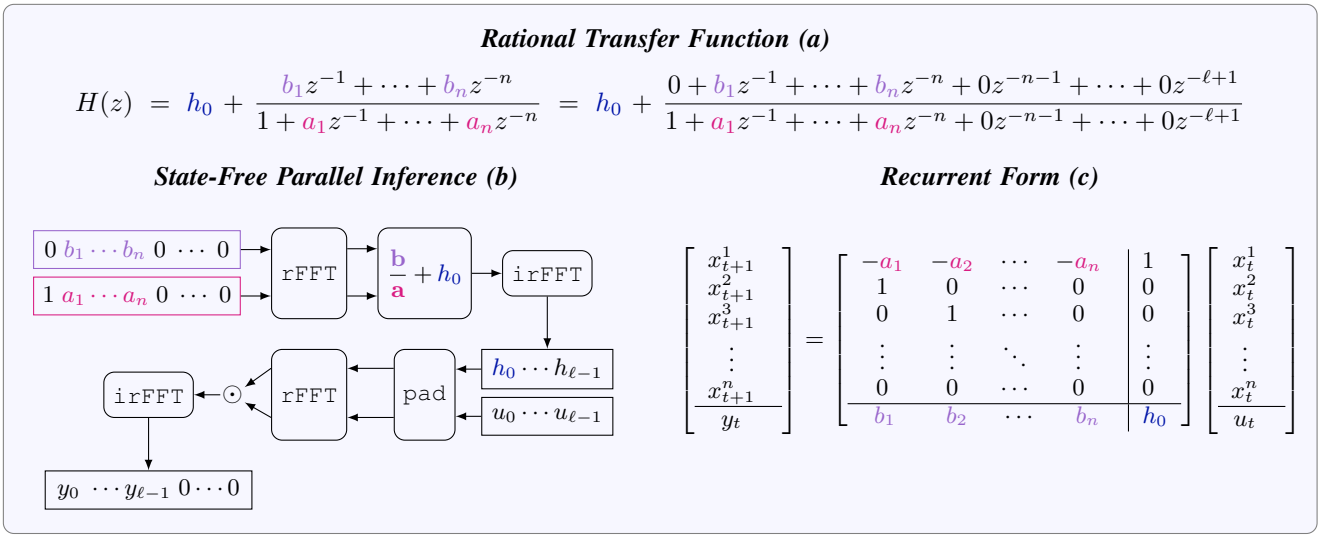

Figure 2: ***(a)*** The rational transfer function (RTF) representation comprises numerator and denominator polynomial coefficients **b** and **a**, and the feedforward term h_0. ***(b)*** illustrates the proposed state-free parallel inference algorithm. The key to efficient state-free inference lies in casting **b** and **a** onto the sequence length for computing the convolutional filter $(h_i)_{i \in [\ell]}$. ***(c)*** illustrates the recurrent form of RTF which can be used for fast single-step inference. Here we denote the i-th state at time t as x_t^i.

Cundy, 2018; Blelloch, 1990) which incurs considerable memory costs at large state sizes[1], due to the materialization of states over the sequence length, as made evident in Figure 1.

The expensive space requirement is alleviated with S4 (Gu et al., 2022b), S4D (Gu et al., 2022a), and Space-Time (Zhang et al., 2023) by an algorithm that admits what we denote as **state-additive** space complexities, in which the parallel inference algorithm collapses the state dimension n onto the sequence length dimension ℓ, enabling space complexities of $\mathcal{O}(\ell + n)$ in place of the much greater **state-multiplicative** $\mathcal{O}(\ell n)$ complexity of scan-based algorithms. To realize the aforementioned state-additive space complexity, S4 and S4D leverage fast Cauchy and Vandermonde matrix-vector product algorithms (Pan, 2001). These algorithms used in computing the convolutional kernel for S4 and S4D scale as $\mathcal{O}((\ell + n) \log^2(\ell + n))$, bottlenecking the faster $\mathcal{O}(\ell \log \ell)$ required to execute the downstream convolution.

We approach solving these issues through a thorough frequency analysis of state-space models and unveil a parallel inference algorithm that admits **state-free** space and time complexities of $\mathcal{O}(\ell)$ and $\mathcal{O}(\ell \log \ell)$ respectively. Additionally, the proposed algorithm operates over a complete representation, the **Rational Transfer Function** (RTF) representation, which unlike diagonal SSMs (Gu et al., 2022a; Gupta et al., 2022; Smith et al., 2023), fully encapsulates the functional space of any linear time-invariant state-space model, including ones parameterized with dense matrices. Parallel inference with RTF solely relies on the Fast Fourier Transform (FFT) algorithm – a widely used and optimized algorithm, alleviating the need for additional custom low-level optimizations to obtain efficient subquadratic complexities. Figure 2 illustrates an overview of the parametrization, parallel inference, and sequential inference algorithms of our proposed SSM.

In order to validate the proposed parametrization, we conducted experiments across a range of tasks, models, and importantly state sizes, including Long Range Arena (LRA), language modeling, and synthetic tasks. Notably, in LRA our proposed model obtained state-of-the-art accuracy (Table 1) among other attention-free models, and faster training speeds in comparison to S4 and S4D across state sizes (Figure 3). We approached language modeling by embedding RTF into a Hyena model (Poli et al., 2023a), effectively replacing the original convolutional filter parameterized with MLPs with transfer functions, and observed improved perplexity over the Hyena Filter baseline when trained on WikiText103 (Table 4).

2. Preliminaries and Related Work

We discuss sequence modeling, convolution-based sequence processing units and their state-space realization.

2.1. Sequence Modeling with Convolutions

Let $\mathbb{S}_\ell^d$ denote the space of length-ℓ vector-valued sequences, $\mathbb{S}_\ell := \{(u_t)_{t \in [\ell]} : u_t \in \mathbb{R}^d\} \equiv \mathbb{R}^{\ell \times d}$. We denote the time index with a subscript roman letter and additional dimensions with greek superscripts, e.g. x_t^α for $t \in [\ell]$ and

[1] Even when the states are only materialized in SRAM (Gu & Dao, 2023), as SRAMs are limited in size.

$\alpha \in [d]$. Any map from $\mathbb{S}_\ell^d$ into itself is herein referred to as a *sequence processor*. Complex *deep* learning architectures tailored for sequence modeling typically involve the composition of simpler, parametric sequence processors in a multi-layer fashion. In this work, we focus on causal sequence processors $u \mapsto y$, where the output y_t at any given time $t \in [\ell]$ is a function of solely the preceding inputs, i.e. $\partial y_t / \partial u_j = 0$ for all $t < j$ and $u \in \mathbb{S}_\ell^d$. This constraint is crucial, for instance, in auto-regressive training of decoder-only language models (Radford et al., 2018) or analogous modeling tasks of temporal dynamics (see e.g. Chen et al., 2021).

The ideal sequence processing layer is expected to fulfill several design criteria, balancing factors such as expressivity, computational and memory efficiency, favorable training dynamics, and parametric efficiency. Of particular interest in this work are those sequence processors that utilize *single-input single-output* (SISO) discrete convolutions as their fundamental components, a.k.a. *linear time invariant (LTI) systems*, with convolutional filters being implicitly parameterized.

A **single-input single-output causal convolution** between an *input* $u \in \mathbb{S}_\ell^1$ and a *filter* $h \in \mathbb{S}_\ell^1$ (often called the *impulse response* function) is defined as

$$(h * u)_t = \sum_{j=0}^{t} h_{t-j} u_j \quad \text{for all } t \in [\ell]. \tag{1}$$

The class of *implicit* convolutions represent the filter as a parametric function $f_\theta : t \mapsto h_t := f_\theta(t)$.

SISO convolution operators can be represented by structured (Toeplitz) matrices that admit a fast multiplication algorithm with efficient sub-quadratic complexity $\mathcal{O}(\ell \log \ell)$. They serve as the fundamental building blocks on various classical signal processing pipelines such as audio systems (Oppenheim et al., 1999) and visual systems (Gonzalez & Woods, 2008).

A notable modern example of sequence processors that make use of implicit convolutions as their core operation on the temporal dimension is the Hyena architecture (Poli et al., 2023a). Given three sequences $q, k, v \in \mathbb{S}_\ell^d$ obtained from the input $u \in \mathbb{S}_\ell^d$ through three dense linear projections $\mathbb{R}^d \to \mathbb{R}^d$ followed by three *short* convolutions, Hyena realizes a map $u \mapsto \mathcal{H}u : \mathbb{S}_\ell^d \to \mathbb{S}_\ell^d$, defined element-wise for all $t \in [\ell]$ and $\alpha \in [d]$ as

$$(\mathcal{H}u)_t^\alpha = u_t^\alpha + \sum_{\beta=0}^{d-1} \sum_{j=0}^{t} \mathsf{T}^{\alpha\beta} q_t^\beta h_{t-j}^\beta k_j^\beta v_j^\beta \tag{2}$$

where $\{h_t^\alpha : t \in [\ell], \alpha \in [d]\} \in \mathbb{S}_\ell^d$ is a collection of implicit *long* convolution filters and $\mathsf{T} \in \mathbb{R}^{d \times d}$ is an *output* projection that mixes channels across the sequence length. Hyena applies d SISO convolutions, independently on each channel. This *multi* SISO approach has been successful in other convolution-based sequence processors such as S4 (Gu et al., 2022b;a) or H3 (Fu et al., 2023) (as well as linear input-varying models (Gu & Dao, 2023)).

2.2. State-Space Realization of Convolutions

This work delves deep into the design of the individual SISO filters h_t, tailored for sequence processing architectures leveraging classical frequency-domain analysis techniques from signal processing and control theory.

More specifically, we specialize on those filters that admit a finite-dimensional state-space (lumped) realization, i.e. the input-output relation of their induced convolution operator can be expressed as:

$$\begin{aligned} x_{t+1} &= \mathsf{A}x_t + \mathsf{B}u_t \\ y_t &= \mathsf{C}x_t + h_0 u_t \end{aligned}, \quad t \mapsto h_t = \begin{cases} h_0 & t = 0 \\ \mathsf{C}\mathsf{A}^{t-1}\mathsf{B} & t > 0 \end{cases} \tag{3}$$

with a finite-dimensional *state* $x_t \in \mathbb{R}^n$ ($n \ll \ell$), *input* $u_t \in \mathbb{R}$, and *output* $y_t \in \mathbb{R}$. Our trainable degrees of freedom are the matrices $\mathsf{A} \in \mathbb{R}^{n \times n}$, $\mathsf{B} \in \mathbb{R}^{n \times 1}$, $\mathsf{C} \in \mathbb{R}^{1 \times n}$, and $h_0 \in \mathbb{R}$. The initial condition $x_0 \in \mathbb{R}^n$ is usually set to zero such that $u \mapsto y$ is a pure convolution. A major advantage of having a state-space realization is the possibility to switch between its convolution mode, for training, and recurrent mode, for efficient auto-regressive generation (see Massaroli et al., 2023 and Section A for further details and denominations).

State-space representations Parametrization of lumped convolutional filters with temporal dynamics, i.e., state-space parametrization present several challenges. Firstly, recurrence with dense transition matrices A are computationally expensive, amounting to a computational complexity of $\mathcal{O}(\ell n^2)$. To make such systems feasible various recent works proposing efficient state-space models have resorted to diagonalization (Gu et al., 2022a; Smith et al., 2023; Orvieto et al., 2023) and low-rank add-ons (Gu et al., 2022b) of A. As will be further uncovered when analyzing the dual representation, transfer functions, these restrictions impose a constraint on the expressivity of its convolutional filter h, given a fixed state-size n. Moreover, despite various works on optimizing parallel inference efficiency, associative scans utilized in (Martin & Cundy, 2018; Smith et al., 2023; Orvieto et al., 2023; Gu & Dao, 2023) still incur considerable memory costs due to its state-multiplicative complexity of $\mathcal{O}(\ell n)$, whereas fast Cauchy and Vandermonde matrix-vector products (Pan, 2001) utilized in (Gu et al., 2022b;a) present an improved state-additive space complexity of $\mathcal{O}(\ell + n)$, but heavily rely on custom platform specific low-level optimizations.

3. Training SSMs in the frequency domain

Linear time-invariant dynamical systems (1) are completely characterized by their **impulse response** h, and in the case they admit a state-space realization (3), their system matrices $(\mathsf{A}, \mathsf{B}, \mathsf{C}, h_0)$.

3.1. Transfer Function Representation

An alternative **complete representation** of (3) is its *transfer function* $H : \mathbb{C} \to \mathbb{C}$, defined as the $\mathcal{Z}$-transform of the impulse response $H(z) \coloneqq \sum_{t\in\mathbb{N}} h_t z^{-t}$ for all $z \in \mathbb{C}$ where the sum converges. The transfer function of a state-space model $(\mathsf{A}, \mathsf{B}, \mathsf{C}, h_0)$ is a *proper*[2] *rational function* of z,

$$\begin{aligned} H(z) &= h_0 + \mathsf{C}(z\mathsf{I} - \mathsf{A})^{-1}\mathsf{B} \\ &= h_0 + \frac{b_1 z^{-1} + \ \cdots \ + b_n z^{-n}}{1 + a_1 z^{-1} + \ \cdots \ + a_n z^{-n}}. \end{aligned} \tag{4}$$

Refer to A.2 for complete derivations. As discrete convolutions are the dual operation to element-wise multiplication under $\mathcal{Z}$-transform, the input-output relation of any LTI system can be equivalently characterized by $H(z)$,

$$y_t = (h * u)_t \quad \Leftrightarrow \quad Y(z) = H(z)U(z)$$

where H is defined outside the circle in the complex plane whose radius is the amplitude of the largest eigenvalue of the state transition matrix A. The $\mathcal{Z}$-transform is a projection of the sequence onto a power basis $z^{-t} = r^{-t}e^{-i\omega t}$ for $r, \omega \in \mathbb{R}$. This basis is not orthogonal unless $r = 1$. That is the basis of the discrete-time Fourier transform $\mathcal{F}$. Hence, the discrete-time Fourier transform of the signal h is defined as $\mathcal{F}[h](e^{i\omega}) = H(e^{i\omega}) \coloneqq \sum_{t\in\mathbb{N}} h_t e^{-i\omega t}$, i.e. it is the transfer function $H(z)$ evaluated at $z = e^{i\omega}$. We say that sequences live in the *time domain* and their $\mathcal{Z}$ (or $\mathcal{F}$) transforms in the *frequency domain*.

We argue that parametrizing state-space models via their transfer function (i.e. making (a, b) the *learnable* parameters), encompasses previous representations of SSMs such as using structured matrices (Fu et al., 2023; Gu et al., 2022b) or *modal canonical forms* (Gu et al., 2022a; Orvieto et al., 2023; Smith et al., 2023; Fu et al., 2023).

Coordinate invariance of the transfer function Notably, the transfer function is an *invariant* of the system: if an invertible change of variables is applied to the state-space representation, the transfer function parameters (a, b) remain unchanged. Without loss of generality let $h_0 = 0$.

> **Lemma 3.1.** *Coefficients a, b are **invariant** under any invertible change of variables.*

Proof. The proof is classic and can be found in (Chen, 1998) and follows from the definition of *equivalence transformation*. Consider the state-space matrices under a change of variables $\hat{x} = \mathsf{K}x$, for some invertible $\mathsf{K} \in \mathbb{R}^{n\times n}$

$$\hat{\mathsf{A}} = \mathsf{K}\mathsf{A}\mathsf{K}^{-1}, \ \ \hat{\mathsf{B}} = \mathsf{K}\mathsf{B}, \ \ \hat{\mathsf{C}} = \mathsf{C}\mathsf{K}^{-1}.$$

The transformed transfer function $\hat{H}(z)$ is given by

$$\hat{H}(z) = \mathsf{C}\mathsf{K}^{-1}[\mathsf{K}(z\mathsf{I} - \mathsf{A})\mathsf{K}^{-1}]^{-1}\mathsf{K}\mathsf{B} = H(z)$$

□

This emergent coordinate invariance should be of warning to most attempts at modeling filters by directly learning either dense or structured state-space matrices $(\mathsf{A}, \mathsf{B}, \mathsf{C})$ as such: there are infinitely many equivalent state-space realizations that map to the same system. This also demonstrates that dense SSM parametrizations are inefficient in their use of parameters with respect to its expressivity.

Expressivity of the transfer function Any impulse response h that can be represented using dense matrices—of $n^2 + 2n + 1$ parameters with stable dynamics—can also be described using rational transfer functions with just $2n + 1$ parameters.

This is demonstrated in the derivations presented in Section A.3. It illustrates that one can calculate the parameters of the transfer functions (a, b, h_0), given any state-space parameterization $(\mathsf{A}, \mathsf{B}, \mathsf{C}, h_0)$, through the following method:

$$\begin{aligned} a &= \mathsf{poly}(\mathsf{eig}(\mathsf{A})), \\ b &= \mathsf{poly}(\mathsf{eig}(\mathsf{A} - \mathsf{B}\mathsf{C})) + \mathsf{poly}(\mathsf{eig}(\mathsf{A}))(h_0 - 1), \end{aligned} \tag{5}$$

in which $\mathsf{poly}(r)$ computes the coefficients of a polynomial given its roots $r_0, \ldots, r_n$.

Parallel to change of variable techniques such as diagonalization of A employed in time-domain state-space realizations, partial fraction decomposition of transfer functions can not only provide alternative representations of state-space models, but also intuitive insights on the expressivity of these models.

As an example, by simply taking the first order partial fraction decomposition of a rational transfer function $H(z)$, i.e.,

$$H(z) = \sum_{i=1}^{n} \frac{r_i}{z - \lambda_i} + h_0 \tag{6}$$

in which $r_i, \lambda_i \in \mathbb{C}$, we obtain the diagonal time-domain parameterization. Its equivalence can be shown by simply breaking down the geometric series $r_i/(z - \lambda_i) = r_i(1/z + \lambda_i/z^2 + \lambda_i^2/z^3 + \ldots)$, and applying the inverse $\mathcal{Z}$-transform (z^{-j} is an impulse at time-step j), resulting in the diagonal SSM convolutional kernel $h_t = \sum_{i\in[n]} r_i \lambda_i^{t-1}$ for $t > 0$.

Looking further, we observe that, like (4), it contains $2n+1$ trainable parameters, but does not permit repeated roots, i.e. $r_1/(z - \lambda_1) + r_2/(z - \lambda_1)^2$, thereby demonstrating its limited expressivity.

[2] i.e. such that the denominator's order is not less than the numerator's one.

Algorithm 1 RTF Kernel Generation

Input: RTF params (a, b, h_0), truncation length ℓ
$\bar{b},\ \bar{a} \leftarrow \texttt{pad}(b, a, (1, \ell - n - 1))$ # Padding a and b to ℓ
$\bar{a}_0 \leftarrow 1$ # Set denominator monic poly. term.
$B,\ A \leftarrow \texttt{FFT}_\ell(\bar{b}, \bar{a})$ # Polynomial eval.
$H \leftarrow B/A + h_0$ # Construct rational function

3.2. State-Free Parallel Inference

For attaining sub-quadratic parallel inference speeds, the approach taken by S4, S4D, and SpaceTime predominantly hinges on the efficient computation of its length-ℓ truncated impulse response h_t:

$$h_t = \begin{cases} h_0 & t = 0 \\ \mathsf{C}\mathsf{A}^{t-1}\mathsf{B} & 0 < t \leq \ell \\ 0 & t > \ell \end{cases}, \tag{7}$$

or its corresponding spectrum $\mathsf{FFT}_\ell(h)$ for downstream integration with the sub-quadratic convolution algorithm, $\mathsf{FFTConv}(u, h)$, described in (Burrus & Parks, 1985; Selesnick & Burrus, 2017; Fu et al., 2024).

Adopting a parallel approach for rational transfer function, we reveal that h_t can be computed in a state-free manner, incurring space and time complexities of $\mathcal{O}(\ell)$ and $\mathcal{O}(\ell \log \ell)$, respectively. This is achieved through the evaluation of the **truncated transfer function** $H_\ell(z)$ across the roots of unity, as delineated below.

Firstly, we demonstrate that an impulse response of length-ℓ, when expressed in the $\mathcal{Z}$-domain as $H_\ell(z) = \sum_{t=0}^{\ell-1} h_t z^{-t}$, can be efficiently transformed into its time-domain representation in the following manner.

Lemma 3.2. *Let $\mathbb{T}_m$ denote the set of the m roots of unity, i.e. $\mathbb{T}_m := \{z^k : z = e^{2\pi i/m}\}_{k\in[m]}$. Then, for all $t \in [\ell]$ and $m \geq \ell$ it holds*

$$h_t = \mathsf{iFFT}_m\big((H_\ell(z))_{z\in\mathbb{T}_m}\big)_t \tag{8}$$

Proof.

$$\begin{aligned} \mathsf{iFFT}_m\big((H_\ell(z))_{z\in\mathbb{T}_m}\big)_t &= \frac{1}{m}\sum_{z\in\mathbb{T}_m} H_\ell(z) z^t \\ &= \frac{1}{m}\sum_{z\in\mathbb{T}_m}\sum_{j=0}^{\ell-1} h_j z^{t-j} \\ &= \frac{1}{m}\sum_{j=0}^{\ell-1} h_j \begin{cases} m & t-j=0 \\ 0 & \text{otherwise} \end{cases} \\ &= h_t. \end{aligned} \tag{9}$$

□

Additionally, observe that the inverse application of Lemma 3.2 results in the following insight.

Evaluating a truncated transfer function $H_\ell(z)$ at the roots of unity, outputs the spectrum of the impulse response, that is:

$$(H_\ell(z))_{z\in\mathbb{T}_m} = \mathsf{FFT}_m(h). \tag{10}$$

In order to truncate the rational transfer function, we devise a "tail" $\tilde{H}_\ell(z)$, such that $H_\ell(z) = H(z) - \tilde{H}_\ell(z)$, as follows.

Lemma 3.3. *Let the "tail", $\tilde{H}_\ell(z)$ be a $\mathcal{Z}$-domain representation a lumped LTI system $(\mathsf{A}, \mathsf{B}, \mathsf{C}, h_0)$ for $t > \ell$, i.e. $\tilde{H}_\ell(z) = \sum_{t=\ell+1}^{\infty} \mathsf{C}\mathsf{A}^{t-1}\mathsf{B}z^{-t}$, then*

$$\tilde{H}_\ell(z) = \mathsf{C}\mathsf{A}^\ell z^{-\ell}(z\mathsf{I} - \mathsf{A})^{-1}\mathsf{B}. \tag{11}$$

Proof.

$$\begin{aligned} \sum_{t=\ell+1}^{\infty} \mathsf{C}\mathsf{A}^{t-1}\mathsf{B}z^{-t} &= \mathsf{C}\mathsf{A}^{-1}\left[\sum_{t=\ell+1}^{\infty} \mathsf{A}^t z^{-t}\right]\mathsf{B} \\ &= \mathsf{C}\mathsf{A}^{-1}\left[\mathsf{A}^{\ell+1}z^{-\ell-1}(\mathsf{I} - \mathsf{A}z^{-1})^{-1}\right]\mathsf{B} \\ &= \mathsf{C}\mathsf{A}^\ell z^{-\ell-1}(\mathsf{I} - \mathsf{A}z^{-1})^{-1}\mathsf{B} \\ &= \mathsf{C}\mathsf{A}^\ell z^{-\ell}(z\mathsf{I} - \mathsf{A})^{-1}\mathsf{B}. \end{aligned} \tag{12}$$

□

Since $z^{-\ell} = 1 \forall z \in \mathbb{T}_\ell$, we can derive the length-ℓ truncated transfer function in the following manner,

$$\begin{aligned} H_\ell(z) = H(z) - \tilde{H}_\ell(z) = \tilde{\mathsf{C}}(z\mathsf{I} - \mathsf{A})^{-1}\mathsf{B}, \\ \tilde{\mathsf{C}} = \mathsf{C}(\mathsf{I} - \mathsf{A}^\ell) := \tilde{b}, \end{aligned} \tag{13}$$

Nonetheless, in practice, we circumvent the computation of A^ℓ, by directly optimizing $\tilde{b}$ during the training phase, and only apply the inverse correction $\mathsf{C} = \tilde{b}(\mathsf{I} - \mathsf{A}^\ell)^{-1}$, upon deployment, i.e. autoregressive inference. This is equivalent to the approach taken by (Gu et al., 2022b;a; Zhang et al., 2023), on the "truncated SSM generating function".

To evaluate the truncated rational function, we recognize that:

1. Rational functions are composed of polynomials.

2. Evaluating polynomials on the roots of unity, is equivalent to applying a fast Fourier transform over its coefficients.

Lemma 3.4. *Let α_k be the k-th order coefficient of a polynomial. Then for all $k, t \in [m]$, $z = e^{2\pi i/m}$, it holds*

$$\sum_{k=0}^{m-1} \alpha_k z^{-tk} = \mathsf{FFT}_m(\alpha)_t, \tag{14}$$

Proof. By definition of the Fourier Transform. □

In light of Lemma 3.4, it becomes evident that for any n-th order truncated rational transfer function parameterized by $(a, \tilde{b}, h_0)$, by setting $a_0 = 1$, $\tilde{b}_0 = 0$ and $a_k, \tilde{b}_k = 0$ for $k > n$ (zero padding of polynomial coefficients), the spectrum of the impulse response can be computed with:

$$H_\ell(z^t) = \frac{\sum_{k=0}^{\ell-1} \tilde{b}_k z^{-tk}}{\sum_{k=0}^{\ell-1} a_k z^{-tk}} + h_0 = \frac{\mathsf{FFT}_\ell(\tilde{b})_t}{\mathsf{FFT}_\ell(a)_t} + h_0, \tag{15}$$

as demonstrated in Algorithm 1. Finally to obtain h_t, we simply apply Equation 8.

Importantly, the proposed parallel inference algorithm relies solely on the FFT algorithms, which have space and time complexities of $\mathcal{O}(\ell)$ and $\mathcal{O}(\ell \log \ell)$, respectively. The ubiquitous FFT algorithm is widely used and already have low-level optimizations applied across several platforms, subsequently optimizing RTF across those platforms.

3.3. Fast Companion Recurrence

Rational transfer functions could directly be translated into a structured state-space model of the following form:

$$x_{t+1} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_n \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} x_t + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u_t \tag{16}$$
$$y_t = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} x_t + h_0 u_t.$$

The structure (companion form) permits fast companion recurrence via the combination of shift operations, dot products resulting in single time-step space and time complexities of $\mathcal{O}(n)$. Refer to Section B.1 for the full derivation.

Moreover, as discussed in (Massaroli et al., 2023), the companion realization of a state-space model can be leveraged to perform fast *prefilling*, in which the state x_t can be obtained from $u_0, \ldots, u_t$ with computation complexity of $\mathcal{O}(\ell \log_2 \ell)$. Fast prefilling is applicable in extensive language modeling applications, where the model, upon receiving a length-ℓ prompt from the user, autoregressively generates subsequent prompts using a constant-time recurrent algorithm as described above. For state-space realizations that are not in companion form, they must first be transformed into the companion form using Equation (5) to perform fast prefilling.

Unlike SpaceTime (Zhang et al., 2023) that shares the same A matrix but trains both B and C, we adhere to the true companion form during training, in which the B matrix is a constant as shown in Equation (16), while b (C matrix) is trained.

3.4. Stable Parametrization

To prevent numerical instabilities, it is important to configure SSMs to exhibit stable dynamics. The choice of parameters for the state-transition matrix A significantly influences their stability. For rational transfer functions, the roots of the denominator polynomial (the pole) must lie within the complex unit circle, i.e. $|r| \leq 1$ to prevent unstable dynamics (Chen, 1998).

Unlike diagonal SSMs, with first order roots (Equation (6)), ensuring that the coefficients of a high order polynomial $\sum_{i=0}^{n} a_{n-i} z^i$ are such that its roots remain within the complex unit circle presents a complex challenge, as highlighted in (Alomari & Chesneau, 2022). SpaceTime (Zhang et al., 2023) adopts Montel's method (Horn & Johnson, 1985; Alomari & Chesneau, 2022), a technique that, for a Monic polynomial (where $a_0 = 1$), constrains the remaining coefficients in a manner described by:

$$\sum_{i=1}^{n-1} |a_i| \leq 1. \tag{17}$$

However, as depicted in Figure 4, the application of Montel's method not only ensures that the roots are confined within the unit circle but also limits them to a specific subset of the stable region. This limitation could potentially diminish performance, a phenomenon supported by the findings in Table 5.

To mitigate this, we propose an alternative initialization strategy for the SSM coefficients, aiming to position them as far as possible from violating Montel's constraint:

$$\mathrm{argmin}_a \left(\sum_{i=1}^{n-1} |a_i|\right) = \mathbf{0}, \tag{18}$$

where $a, \tilde{b} = \mathbf{0}$. We denote this initialization scheme as the **zero** initialization. Our ablation tests (Table 5) and comparisons against SpaceTime on the Long Range Arena (Tay et al., 2021) benchmark (Table 1) show enhanced training stability and consequently, improved performance when adopting the zero initialization scheme.[3]

[3] Unless explicitly stated otherwise, all results presented in this paper adopts the zero initialization scheme with $h_0 = 1$.

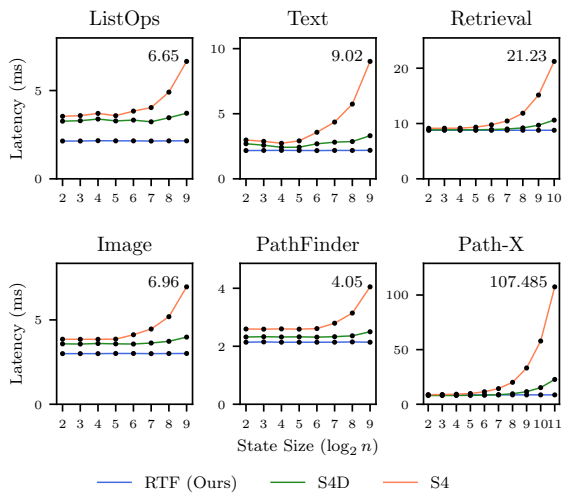

Figure 3: Latency profiles for a single RTF, S4D, and S4 layer at various state sizes. It is evident that RTF consistently exhibits superior parallel inference speeds, with its lower latency across a range of tasks and state sizes.

4. Experimental Results

In this section, we conduct an empirical evaluation of RTF in comparison to other state-space models and sequence models. Section 4.1 is dedicated to assessing memory usage and processing speed. Sections 4.2 and 4.3 examine the ability for SSMs to memorize and model long-range dependencies. Finally, their ability to model language is assessed in sections 4.4 and 4.5.

4.1. Efficiency Profiling

We profiled GPU memory usage between a parallel scan-based S5 model and RTF across different sequence lengths and state sizes at channel dimensions of $d = 1024$. The results depicted in Figure 1 reveal a consistent trend, wherein the memory consumption for the scan operation rises in conjunction with state size and sequence length, while it solely escalates with sequence length for RTF. This phenomenon can be attributed to the aforementioned state-free characteristic of RTF's inference algorithm, which casts its parameters with size of the state dimension onto the sequence length for parallel inference. We also observed a similar trend for the inference latency which is further detailed in Appendix C.1.

Next, we profiled inference latency across different SSMs of varying state-sizes over a suite of six LRA tasks, facilitating speed comparisons across a wide range of model architectures. Figure 3 reports the median inference latency per SSM layer across 75 training iterations.

The results show a recurring trend, wherein RTF's inference latency remained consistent regardless of state size and conversely, S4D and S4 experienced slower speeds particularly at higher orders, due to the utilization of the slower Vandermonde or Cauchy matrix-vector product algorithms respectively, which have computational complexity of $\mathcal{O}((\ell + n)\log^2(\ell + n))$ as opposed to RTF's $\mathcal{O}(\ell \log \ell)$.

4.2. Modeling Long Range Dependencies

The Long Range Arena (LRA) benchmark has become a common ground for testing various sequence models including SSMs (Gu et al., 2022b;a; Smith et al., 2023; Hasani et al., 2023) and Transformers (Vaswani et al., 2017; Choromanski et al., 2021). It is composed of six classification tasks with long range input sequences of lengths ranging from 1024 to 16384. We conducted these experiments on RTF along with S4, S4D, and SpaceTime (Zhang et al., 2023) as presented in Table 1.

RTF obtained strong results in several LRA tasks, including attaining state-of-the-art performance on `Retrieval`, and among attention-free approaches, the average score. However for Path-X, RTF was unable to learn a policy beyond random guessing when the state-size was fixed to 64, prompting an increase to 2048. Nevertheless, due to RTF's state-free parallel inference algorithm, this increase in state-size did not impact GPU memory consumption nor training speed as evidenced in Figure 3.

4.3. Synthetic Memorization Tasks

Recurrences have traditionally struggled with vanishing and exploding gradients, making memorization tasks challenging (Bengio et al., 1994; Pascanu et al., 2013). To evaluate the memorization capabilities of our state-space model, we benchmark them against two synthetic memorization tasks: *Copying* and *Delay*.

The *Copying* task, akin to (Arjovsky et al., 2016), presents SSMs with 1024 length sequences of 64 discrete states sampled uniformly, which the model is then tasked to recall all 1024 tokens in order. Each model was given 10k training samples for 50 epochs, and was tested with 1000 unseen samples.

The *Delay* task, which was also used to ablate HiPPO SSM initialization schemes (Gu et al., 2023), simply tests the model's ability to delay a continuous white noise by 1000 time steps. As reported by Gu et al., LSTMs and Transformers struggle on this seemingly simple task, and are unable to improve beyond a random guessing policy. The primary distinction between *Copying* and *Delay* is whether the input data is discrete or continuous. More detailed experimental setup could be found in C.3.

From the results reported in Table 2, we observed that at higher state-sizes, RTF could more accurately copy and delay data. S4 on the other hand struggled on *Copying*,

Table 1: Long range arena benchmark results. We included results reported in (Gu et al., 2022b; Smith et al., 2023; Ren et al., 2023) and additionally ran SpaceTime (Zhang et al., 2023) based on the official implementation with hyperparameters identical to RTF. We also included results of self-pretrained (SPT) Transformers (Amos et al., 2024) denoted with + Causal SPT. † indicate the use of an increased state-size and ✗ indicates that the model was unable to train beyond a random guessing policy.

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg.
Transformer	36.37	64.27	57.46	42.44	71.4	✗	53.66
Luna-256	37.25	64.57	79.29	47.38	72.72	✗	58.54
Transformers + Causal SPT	59.15	88.81	90.38	76.0	88.49	88.05	81.81
Mega $\mathcal{O}(\ell^2)$	**63.14**	**90.43**	91.25	90.44	96.01	97.98	**88.21**
H3	57.5	88.2	91.0	87.3	93	91.8	84.8
CCNN	43.6	84.08	✗	88.9	91.51	✗	68.02
Liquid-S4	62.75	89.02	91.2	89.5	94.8	96.66	87.32
S5	62.15	89.31	91.4	88.0	95.33	**98.58**	87.46
S4	61.29	88.25	90.90	89.2	94.2	96.35	86.69
S4D	60.74	87.03	90.68	89.18	95.42	97.32	86.72
SpaceTime	56.4	87.8	91.45	86.27	✗	✗	70.32
RTF (Ours)	61.59	89.72	**92.04**	**90.51**	**96.11**	96.32†	87.71

Table 2: Results on synthetic memorization tasks. The state-size of the model is denoted with the number trailing the model name, i.e. S4-64 is an S4 model with $n = 64$.

Model	*Copying* acc. ↑	*Delay* RMSE ↓
S4-64	**29.3**	**0.41**
RTF-64	22.1	0.45
S4-128	34.2	**0.39**
RTF-128	**93.3**	0.45
S4-256	35.0	**0.33**
RTF-256	**100**	0.44
S4-512	33.1	**0.22**
RTF-512	**100**	0.38
S4-1024	33.2	0.029
RTF-1024	**100**	**0.006**

showing no improvements beyond the state-size of 256. It is also worth noting that on both synthetic tasks, unlike the discrete-time RTF SSM, S4, being continuous-time required careful consideration of the initialization and interplay between the time-constant Δ and the transition matrix A for reasonable performance.

4.4. Laughing Hyena Distillation

Hyena (Poli et al., 2023a) and MultiHyena (Massaroli et al., 2023) operators utilize a diverse array of filters, encompassing short convolutional filters - filters implicitly parameterized by multi-layer perceptrons (MLP) (Poli et al., 2023a; Sitzmann et al., 2020; Romero et al., 2022), and diagonal SSMs (Massaroli et al., 2023). Notably, Hyena operators with MLP-parameterized filters have demonstrated superior performance compared to other convolutional and recurrent methods, as highlighted in (Akyürek et al., 2024; Bhattamishra et al., 2024). Despite their effectiveness, these filters lack constant-time autoregressive inference speeds desired in applications such as language modeling. This limitation has led to the investigation of distilling MLP-based filters into SSMs, a process detailed in Laughing Hyena (Massaroli et al., 2023).

Here, we look into distillation of MLP-based filters, using a 160M parameter multi-head StripedHyena (Poli et al., 2023b) language model, trained on The Pile (Gao et al., 2021), and compare distillation performances between RTF and a diagonal SSM employed in Laughing Hyena (LH), both of which boast highly efficient $\mathcal{O}(n)$ autoregressive algorithms. Table 3 reports distillation errors and downstream LM-Evaluation-Harness scores (Gao et al., 2023).

Interestingly, despite the theoretically superior expressiveness of RTF models, we observed that the modal representation employed in LH exhibits more favorable training dynamics for distillation at state-sizes 16 and 64, as evidenced by the distillation MSE. However with $n = 4$, RTF outperforms LH while maintaining comparable downstream evaluation performances to the baseline model, making it a good candidate for unlocking efficient constant-speed autoregressive inference on Hyena language models.

4.5. WikiText103 Language Modeling

In addition to evaluating the language modeling capabilities of state space models through distillation techniques, their performance when directly trained on autoregressive cross-entropy loss (Radford et al., 2018) was investigated on the well-established WikiText-103 dataset. We used a Hyena operator and replaced its filters with RTF, which we refer to as *Hyena-RTF*.

Table 3: This table illustrates downstream evaluation scores from LM-Evaluation-Harness (Gao et al., 2023). The number trailing the model names indicate its state-size.

Model	Winogrande acc. ↑	PIQA acc. ↑	HellaSwag acc. norm. ↑	OpenbookQA acc. norm. ↑	Distillation MSE ↓
Baseline (160M)	52.09	61.64	29.68	29.4	-
LH-4	51.7	62.02	29.76	29.6	0.032
RTF-4	51.7	61.04	29.82	29.6	**0.018**
LH-16	52.25	61.75	29.73	28.6	**0.009**
RTF-16	52.96	61.64	29.85	29.8	0.013
LH-64	49.57	61.59	29.8	29.6	**0.007**
RTF-64	53.43	61.81	29.85	29.2	0.011

Table 4: WikiText103 language modeling perplexity scores. The results are taken from (Poli et al., 2023a). Each model listed below contains ∼125M parameters.

Model	Perplexity ↓
Transformer	18.6
Hybrid H3	18.5
Linear Attention	25.6
Hyena	18.5
Hyena-S5 (Smith et al., 2023)	18.3
Hyena-RTF (Ours)	**18.0**

As shown in Table 4, Hyena-RTF outperforms both the Transformer and Hyena baselines on WikiText103. Additionally, RTF without the Hyena operator structure was compared against S4 and S4D on a pilot experiment further described in Appendix C.5.1, which similarly indicated relatively strong language modeling capability among other LTI SSMs. These results signal a promising potential for further scaling RTF on larger models and datasets.

5. Conclusion

In this study, we explore *state-space model* (SSM) parametrization via their dual representation, transfer functions. We systematically unveiled the realization of SSMs through rational transfer functions (RTF), demonstrating state-of-the-art efficiency through a state-free parallel inference algorithm, while maintaining the expressiveness of a dense SSM. Our experiments revealed that RTFs are effective for modeling long-range dependencies and processing language, and also exhibits improvements in comparison to the S4 model across synthetic memorization tasks with higher state-sizes. The results of our investigation suggest that RTFs hold significant potential for modeling signals across a variety of other domains.

Acknowledgements

T.S. was partially supported by JSPS KAKENHI (20H00576) and JST CREST (JPMJCR2015).

Impact Statement

This paper introduces research aimed at contributing to the advancement of the field of Machine Learning. It is designed to craft highly efficient models that lessen computational expenses. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Akyürek, E., Wang, B., Kim, Y., and Andreas, J. In-context language learning: Architectures and algorithms, 2024.

Alomari, M. W. and Chesneau, C. Bounding the zeros of polynomials using the frobenius companion matrix partitioned by the cartesian decomposition. *Algorithms*, 15(6), 2022. ISSN 1999-4893. doi: 10.3390/a15060184. URL https://www.mdpi.com/1999-4893/15/6/184.

Amos, I., Berant, J., and Gupta, A. Never train from scratch: Fair comparison of long-sequence models requires data-driven priors. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PdaPky8MUn.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1120–1128. JMLR.org, 2016.

Baevski, A. and Auli, M. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxZX20qFQ.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Bhattamishra, S., Patel, A., Blunsom, P., and Kanade, V. Understanding in-context learning in transformers and LLMs by learning to learn discrete functions. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ekeyCgeRfC.

Blelloch, G. E. Prefix sums and their applications. In *Sythesis of parallel algorithms*, pp. 35—60. Morgan Kaufmann Publishers Inc., 1990. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.6430.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Burrus, C. S. and Parks, T. Convolution algorithms. *Citeseer: New York, NY, USA*, 6:15, 1985.

Chen, C.-T. *Linear System Theory and Design*. Oxford University Press, Inc., USA, 3rd edition, 1998. ISBN 0195117778.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=a7APmM4B9d.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 933–941. JMLR.org, 2017.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. Hungry Hungry Hippos: Towards language modeling with state space models. In *International Conference on Learning Representations*, 2023.

Fu, D. Y., Kumbong, H., Nguyen, E., and Ré, C. FlashFFTConv: Efficient convolutions for long sequences with tensor cores. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gPKTTAfYBp.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL https://arxiv.org/abs/2101.00027.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

Gonzalez, R. C. and Woods, R. E. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J., 2008. ISBN 9780131687288 013168728X 9780135052679 013505267X.

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

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 35971–35983. Curran Associates, Inc., 2022a.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Gu, A., Johnson, I., Timalsina, A., Rudra, A., and Re, C. How to train your HIPPO: State space models with generalized orthogonal basis projections. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=klK17OQ3KB.

Gupta, A., Gu, A., and Berant, J. Diagonal state spaces are as effective as structured state spaces. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35 - 36th Conference on Neural Information Processing Systems, NeurIPS 2022*, Advances in Neural Information Processing Systems. Neural information processing systems foundation, 2022. Publisher Copyright: © 2022 Neural information processing systems foundation. All rights reserved.; 36th Conference on Neural Information Processing Systems, NeurIPS 2022 ; Conference date: 28-11-2022 Through 09-12-2022.

Hasani, R., Lechner, M., Wang, T.-H., Chahine, M., Amini, A., and Rus, D. Liquid structural state-space models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=g4OTKRKfS7R.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015. doi: 10.1109/ICCV.2015.123.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus), 2023.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Horn, R. A. and Johnson, C. R. *Matrix Analysis*. Cambridge University Press, 1985.

Katsch, T. Gateloop: Fully data-controlled linear recurrence for sequence modeling, 2023.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33: 6696–6707, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Linsley, D., Kim, J., Veerabadran, V., Windolf, C., and Serre, T. Learning long-range spatial dependencies with horizontal gated recurrent units. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/ec8956637a99787bd197eacd77acce5e-Paper.pdf.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In Lin, D., Matsumoto, Y., and Mihalcea, R. (eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015.

Martin, E. and Cundy, C. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyUNwulC-.

Massaroli, S., Poli, M., Sonoda, S., Suzuki, T., Park, J., Yamashita, A., and Asama, H. Differentiable multiple shooting layers. *Advances in Neural Information Processing Systems*, 34:16532–16544, 2021.

Massaroli, S., Poli, M., Fu, D. Y., Kumbong, H., Parnichkun, R. N., Romero, D. W., Timalsina, A., McIntyre, Q., Chen, B., Rudra, A., Zhang, C., Re, C., Ermon, S., and Bengio, Y. Laughing hyena distillery: Extracting compact recurrences from convolutions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OWELckerm6.

Nangia, N. and Bowman, S. ListOps: A diagnostic dataset for latent tree learning. In Cordeiro, S. R., Oraby, S., Pavalanathan, U., and Rim, K. (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 92–99, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-4013. URL https://aclanthology.org/N18-4013.

Oppenheim, A. V., Schafer, R. W., and Buck, J. R. *Discrete-Time Signal Processing*. Prentice-hall Englewood Cliffs, second edition, 1999.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Pan, V. Y. *Structured Matrices and Polynomials: Unified Superfast Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2001. ISBN 0817642404.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/pascanu13.html.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: towards larger convolutional language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023a.

Poli, M., Wang, J., Massaroli, S., Quesnelle, J., Nguyen, E., and Thomas, A. Stripedhyena: Moving beyond transformers with hybrid signal processing models. 2023b.

Radev, D. R., Muthukrishnan, P., and Qazvinian, V. The ACL Anthology network corpus. In Kan, M.-Y. and Teufel, S. (eds.), *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries (NLPIR4DL)*, pp. 54–61, Suntec City, Singapore, August 2009. Association for Computational Linguistics. URL https://aclanthology.org/W09-3607.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. 2018.

Ren, L., Liu, Y., Wang, S., Xu, Y., Zhu, C., and Zhai, C. Sparse modular activation for efficient sequence modeling, 2023.

Romero, D. W., Kuzina, A., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. Ckconv: Continuous kernel convolution for sequential data. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=8FhxBtXSl0.

Sandberg, I. W. On the theory of linear multi-loop feedback systems. *Bell System Technical Journal*, 42(2):355–382, 1963.

Selesnick, I. W. and Burrus, C. S. Fast convolution and filtering. In *Digital Signal Processing Fundamentals*, pp. 185–208. CRC Press, 2017.

Sitzmann, V., Martel, J. N., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

Smith, J. T., Warrington, A., and Linderman, S. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Zhang, M., Saab, K., Poli, M., Dao, T., Goel, K., and Ré, C. Effectively modeling time series with simple discrete state spaces. *International Conference on Learning Representations*, 2023.

Supplementary Material

Contents

1 Introduction **1**

2 Preliminaries and Related Work **2**
- 2.1 Sequence Modeling with Convolutions . . . 2
- 2.2 State-Space Realization of Convolutions . . . 3

3 Training SSMs in the frequency domain **4**
- 3.1 Transfer Function Representation . . . 4
- 3.2 State-Free Parallel Inference . . . 5
- 3.3 Fast Companion Recurrence . . . 6
- 3.4 Stable Parametrization . . . 6

4 Experimental Results **7**
- 4.1 Efficiency Profiling . . . 7
- 4.2 Modeling Long Range Dependencies . . . 7
- 4.3 Synthetic Memorization Tasks . . . 7
- 4.4 Laughing Hyena Distillation . . . 8
- 4.5 WikiText103 Language Modeling . . . 8

5 Conclusion **9**

A Linear System Theory **15**
- A.1 Overview and Basics . . . 15
- A.2 Transfer Function Realization of Lumped LTI Systems . . . 16
- A.3 From State-Space to Transfer Function (Massaroli et al., 2023) . . . 17
- A.4 From Transfer Function to State-Space (Massaroli et al., 2023) . . . 17

B RTF: Further Details **19**
- B.1 Fast Companion Recurrence . . . 19
- B.2 Initialization and Stability . . . 19
- B.3 Alternative Inference Algorithms . . . 21
  - B.3.1 RTF Kernel Generation via Long Polynomial Division . . . 21
  - B.3.2 Multi-Input Multi-Output RTF . . . 21

C Experiments **23**
- C.1 Memory and Latency Profiling Experiments . . . 23

C.2 Long Range Arena Benchmark . . . 23
C.2.1 Model Architecture Details . . . 23
C.2.2 Long Range Arena Benchmark Details . . . 24
C.3 Synthetic Memorization Tasks . . . 25
C.3.1 *Copying* Task . . . 25
C.3.2 *Delay* Task . . . 26
C.4 Laughing Hyena Distillation Task . . . 26
C.5 WikiText103 Language Modeling . . . 27
C.5.1 Pilot Experiments . . . 27
C.5.2 Model Architecture Details . . . 27

Author Contribution

R.N.P. Developed the algorithm, theory, code base, and manuscript. Managed and conducted experiments.

S.M. Developed the algorithm, theory, and manuscript. Supervised research.

A.M. Developed the code base and manuscript. Conducted experiments and secured compute.

J.S. Reviewed manuscript and assisted in writing.

R.H., **M.L.** Reviewed manuscript and secured compute.

Q.A., **C.R.**, **H.A.**, **S.E.**, **T.S.** Supervised research.

M.P. Developed the algorithm, theory and manuscript. Supervised research.

A.Y. Supervised research and secured compute.

A. Linear System Theory

This section delves into linear system theory, including denomination of various characteristics such as lumpedness, time-invariance, etc., and also includes the analysis and derivation of $\mathcal{Z}$-domain transfer functions.

A.1. Overview and Basics

Linear Systems: Linear systems consist of a series of linear equations generally expressed as:

$$y = \mathsf{G}u, \tag{A.1.1}$$

in which $u \in \mathbb{R}^{\ell}$, $y \in \mathbb{R}^{\ell}$, and $\mathsf{G} \in \mathbb{R}^{T \times T}$ are the input, output, and the transformation matrix, respectively. These systems adhere to the principles of linearity, including additivity and homogeneity. For the purpose of processing sequences, they can also written as:

$$y_t = \sum_{j=t_0}^{t} \mathsf{G}_{t,t-j} u_j, \tag{A.1.2}$$

in which, $\mathsf{G}_{t,t-j}$ scales the input signal u_j for the output, based on the absolute time t and the relative time $t - j$.

Time-Invariance A linear time-invariant (LTI) system simply discards the absolute time dependence in (A.1.2) as follows:

$$y_t = \sum_{j=t_0}^{t} h_{t-j} u_j. \tag{A.1.3}$$

These systems are equivalent to convolutions characterized by h, with a shorthand notation $y_t = (h * u)_t$. h is also known as the system's *impulse response*. As $y = h * \delta = h$, in which δ is the Kronecker delta (impulse) function.

Lumped Systems: Lumped LTI systems (Chen, 1998) are LTI systems that can be characterized with a finite and discrete (lumped) set of states. They can be formulated as a state-space model:

$$\begin{aligned} x_{t+1} &= \mathsf{A}x_t + \mathsf{B}u_t \\ y_t &= \mathsf{C}x_t + h_0 u_t, \end{aligned} \tag{A.1.4}$$

where $\mathsf{A} \in \mathbb{C}^{N \times N}$, $\mathsf{B} \in \mathbb{C}^{N \times 1}$, $\mathsf{C} \in \mathbb{C}^{1 \times N}$, and $h_0 \in \mathbb{R}$. Unrolling the recurrence, its connection to the convolutional operation could be made clear:

$$\begin{aligned} y_0 &= \mathsf{C}x_0 + h_0 u_0 \\ y_1 &= \mathsf{C}(\mathsf{A}x_0 + \mathsf{B}u_0) + h_0 u_1 \\ y_2 &= \mathsf{C}(\mathsf{A}(\mathsf{A}x_0 + \mathsf{B}u_0) + \mathsf{B}u_1) + h_0 u_2 \\ &\vdots \\ y_t &= h_0 u_t + \sum_{j=1}^{t} \mathsf{C}\mathsf{A}^{j-1}\mathsf{B}u_{t-j} + \mathsf{C}\mathsf{A}^t x_0 \\ y_t &= (h * u)_t + \mathsf{C}\mathsf{A}^t x_0, \text{ where } h_t = \begin{cases} h_0 & t = 0 \\ \mathsf{C}\mathsf{A}^{t-1}\mathsf{B} & t > 0 \end{cases}. \end{aligned} \tag{A.1.5}$$

Note that all lumped LTI systems have complex exponential convolutional kernels. Non-lumped systems are not restricted to exponential convolutional kernels but cannot be directly expressed using a fixed and finite state-space, i.e. they have a non-constant time autoregressive inference complexity. Convolutional filters implicitly parameterized by MLPs such as CKConv (Romero et al., 2022) and (Poli et al., 2023a) are examples of non-lumped linear time-invariant systems.

A.2. Transfer Function Realization of Lumped LTI Systems

Control Theorists Derivation: By applying the shift forward operator (z) in $\mathcal{Z}$-domain to the state-space equations, we can obtain its *transfer function* as follows.

$$\begin{aligned}
x_{k+1} &= \mathsf{A}x_k + \mathsf{B}u_k && \text{state dynamics} \\
X(z)z &= \mathsf{A}X(z) + \mathsf{B}U(z) && \mathcal{Z}\text{-transform} \\
(z\mathsf{I} - \mathsf{A})X(z) &= \mathsf{B}U(z) && (z\mathsf{I} - \mathsf{A}) \text{ is also known as the resolvent matrix} \\
X(z) &= (Iz - \mathsf{A})^{-1}\mathsf{B}U(z) \\
H(z) &= \frac{Y(z)}{U(z)} = \mathsf{C}(z\mathsf{I} - \mathsf{A})^{-1}\mathsf{B} + h_0 && \text{substituted } X(z) \text{ into } Y(z) = \mathsf{C}X(z) + h_0U(z)
\end{aligned} \tag{A.2.1}$$

Alternative Derivation: (Massaroli et al., 2023) The transfer function can also be derived by direct $\mathcal{Z}$-transform of the impulse response h_t of the system. This derivation is useful to highlight the region of convergence of the transfer function.

$$\begin{aligned}
H(z) &= h_0 + \sum_{t=1}^{\infty} z^{-t}\mathsf{C}\mathsf{A}^{t-1}\mathsf{B} && h_0 \text{ is pulled out via } h_0z^0 = h_0 \\
&= h_0 + \mathsf{C}\left[\sum_{t=1}^{\infty} z^{-t}\mathsf{A}^{t-1}\right]\mathsf{B} && \text{multiplication distributes over sum.} \\
&= h_0 + z^{-1}\mathsf{C}\left[\sum_{t=1}^{\infty} z^{-(t-1)}\mathsf{A}^{t-1}\right]\mathsf{B} && \text{multiply by } z/z \\
&= h_0 + z^{-1}\mathsf{C}\left[\sum_{t=0}^{\infty} (z^{-1}\mathsf{A})^{t}\right]\mathsf{B} && \text{change of index and collect like terms}
\end{aligned} \tag{A.2.2}$$

We look at the convergence of the series $\sum_{t=0}^{\infty} \|z^{-1}\mathsf{A}\|_2^t$. We have

$$\begin{aligned}
\|z^{-1}\mathsf{A}\|_2 &\leq \|z^{-1}\|_2\|\mathsf{A}\|_2 \\
&= \|r^{-1}e^{-i\omega}\|_2\|\mathsf{A}\|_2 && \text{using } z \coloneqq re^{i\omega} \in \mathbb{C},\ r, \omega \in \mathbb{R} \\
&\leq r^{-1}\|\mathsf{A}\|_2 = r^{-1}\rho(\mathsf{A})
\end{aligned}$$

The series converges to $1/(1 - r^{-1}\rho(\mathsf{A}))$ if and only if $r^{-1}\rho(\mathsf{A}) < 1$ i.e. for $r > \rho(\mathsf{A})$. Thus, in the exterior of the disk with radius $\rho(\mathsf{A})$, $\mathbb{D}_{\rho(\mathsf{A})} \coloneqq \{z \in \mathbb{C} : |z| > \rho(\mathsf{A})\}$, $\sum_{t=0}^{\infty}(z^{-1}\mathsf{A})^t$ converges to $(\mathsf{I} - z^{-1}\mathsf{A})^{-1}$ and

$$z \in \mathbb{D}_{\rho(\mathsf{A})} \;\Rightarrow\; H(z) = h_0 + z^{-1}\mathsf{C}(\mathsf{I} - z^{-1}\mathsf{A})^{-1}\mathsf{B} = h_0 + \mathsf{C}(z\mathsf{I} - \mathsf{A})^{-1}\mathsf{B}$$

The transfer function $H(z) = h_0 + \mathsf{C}(z\mathsf{I} - A)^{-1}\mathsf{B}$ of a stable lumped discrete-time system is defined outside the disc in the complex plane that encloses all the eigenvalues of A.

Further dissecting $H(z) = h_0 + \mathsf{C}(z\mathsf{I} - A)^{-1}\mathsf{B}$, note that to compute the inverse, $\det(z\mathsf{I} - \mathsf{A})$ is a nth order Monic polynomial, and $\mathsf{C}[\mathsf{Adj}(z\mathsf{I} - \mathsf{A})]\mathsf{B}$ is a $n - 1$ order polynomial (for the SISO case), hence the general form of a transfer function can be written in the form of the following rational function (this is discussed in greater detail in A.3):

$$H(z) = \frac{b_1z^{-1} + b_2z^{-2} + \cdots + b_nz^{-n}}{1 + a_1z^{-1} + a_2z^{-2} + \cdots + a_nz^{-n}} + h_0 \quad \rightarrow \text{Rational function form.} \tag{A.2.3}$$

The SISO rational coefficient form has $2n + 1$ parameters. With partial fraction decomposition, the rational function can be broken down into its first order partial decomposition, resulting in a modal representation:

$$H(z) = \sum_{i=1}^{n} \frac{r_i}{z - \lambda_i} + h_0 \quad \rightarrow \text{Modal form,} \tag{A.2.4}$$

in which $r, \lambda \in \mathbb{C}$. This form parameterizes the *poles* (λ) and its associated magnitude (r). The modal form has $2n + 1$ trainable parameters. It is worth noting that the first order partial fraction decomposition does not permit any form of repeated roots, for this reason, it is not a complete representation of a lumped LTI systems.

Another way in which rational functions can be structured is called the zero-pole-gain (ZPK) representation:

$$H(z) = k\frac{\prod_{i=1}^{n-1}(z - z_i)}{\prod_{i=1}^{n}(z - \lambda_i)} + h_0 \quad \to \text{Zero-Pole-Gain form,} \tag{A.2.5}$$

in which, k, z, and λ are the gain, zeros, and poles respectively. The ZPK form has $2n + 1$ trainable parameters.

A.3. From State-Space to Transfer Function (Massaroli et al., 2023)

We detail an implementation oriented method to compute the coefficients $(a_i)_{i=1}^n, (b_i)_{i=1}^n$ of a SSM's transfer function. Expanding the inverse of the resolvent matrix, recall that

$$H(z) = \mathsf{C}[z\mathsf{I} - \mathsf{A}]^{-1}\mathsf{B} + h_0 = \frac{\mathsf{C}\,\mathsf{Adj}(z\mathsf{I} - \mathsf{A})\mathsf{B} + \mathsf{det}(z\mathsf{I} - \mathsf{A})h_0}{\mathsf{det}(z\mathsf{I} - \mathsf{A})} \tag{A.3.1}$$

This shows that the denominator coefficients $(a_i)_{i=1}^n$ are simply the coefficients of the characteristic polynomial of matrix A. They can be easily obtained by 1. computing the eigenvalues of A and 2. calculating the coefficients of the polynomial whose roots are such eigenvalues. On the other hand, the numerator apparently involves more complex symbolic manipulation. This can be simplified recalling a classic matrix-determinant identity:

Lemma A.1 ((Sandberg, 1963)). *Let* M, B, *and* C *respectively denote matrices of orders* $n \times n$, $n \times 1$, *and* $1 \times n$. *Then,*

$$\mathsf{det}(\mathsf{M} + \mathsf{BC}) = \mathsf{det}(\mathsf{M}) + \mathsf{C}\,\mathsf{Adj}(\mathsf{M})\mathsf{B}.$$

Applying Lemma A.1 to (A.3.1) we obtain

$$H(z) = \frac{\mathsf{det}(z\mathsf{I} - \mathsf{A} + \mathsf{BC}) + \mathsf{det}(z\mathsf{I} - \mathsf{A})(h_0 - 1)}{\mathsf{det}(z\mathsf{I} - \mathsf{A})}.$$

Let $\mathsf{poly}(r)$ denote the coefficients of the polynomials with roots $r = (r_1, \dots, r_n)$. Then $a = \mathsf{poly}(\mathsf{eig}(\mathsf{A}))$. Since A and $\mathsf{A} - \mathsf{BC}$ are of equal dimension, their characteristic polynomials have equal order and therefore

$$b = \mathsf{poly}(\mathsf{eig}(\mathsf{A} - \mathsf{BC})) + \mathsf{poly}(\mathsf{eig}(\mathsf{A}))(h_0 - 1)$$

```
def get_tf_from_ss(A,B,C,h0):
    a = poly(eig(A))
    b = poly(eig(A - outer(B,C))) + (h0-1)*a
    return a, b
```

Listing 1: State-space → transfer function conversion code

A.4. From Transfer Function to State-Space (Massaroli et al., 2023)

Chen's derivation The derivation is based on the steps reported for the continuous-time *multi-input multi-output* case in (Chen, 1998) adapted to single-input single-output Transfer Functions.

Let $H(z) = \frac{q(z)}{p(z)} + h_0$, we define a pseudo-state v such that

$$p(z)V(z) = U(z) \quad \Leftrightarrow \quad V(z) = \frac{1}{p(z)}U(z). \tag{A.4.1}$$

Then, we define the state $x_t := (x_t^1, \dots, x_t^n) \in \mathbb{R}^n$ as

$$x_t = (v_{t-1}, v_{t-2}, \cdots, v_{t-n}) \quad \Leftrightarrow \quad \mathcal{Z}\{x\}(z) = X(z) = \begin{bmatrix} z^{-1} \\ \vdots \\ z^{-n} \end{bmatrix} V(z). \tag{A.4.2}$$

From (A.4.1) we have

$$\begin{aligned}
V(z) + a_1 z^{-1}V(z) + \cdots + a_n z^{-n}V(z) = U(z) \quad &\Leftrightarrow \\
V(z) = -a_1 z^{-1}V(z) - \cdots - a_n z^{-n}V(z) + U(z) \quad &\Leftrightarrow \\
v_t = -a_1 v_{t-1} - \cdots - a_n v_{t-n} + u_t \quad &\Leftrightarrow \quad \text{time-delay prop. of } \mathcal{Z}\text{-transform} \\
x_{t+1}^1 = -a_1 x_t^1 - \cdots - a_n x_t^n + u_t \quad &\Leftrightarrow \quad \text{by def. of state (A.4.2).}
\end{aligned}$$

Thus, we have the overall recurrence

$$
\begin{aligned}
x^1_{t+1} &= -a_1 x^1_t - \cdots - a_n x^n_t + u_t \\
x^2_{t+1} &= x^1_t \\
&\vdots \\
x^n_{t+1} &= x^{n-1}_t
\end{aligned}
$$

which can be written in matrix form as

$$
x_{t+1} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_n \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} x_t + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u_t
$$

The output spectrum is then given by

$$
\begin{aligned}
Y(z) = H(z)U(z) &= \frac{q(z)}{p(z)} U(z) + h_0 U(z) \\
&= q(z)V(z) + h_0 U(z) && \text{by def. of } V(z).
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
Y(z) = q(z)V(z) + h_0 U(z) &= \begin{bmatrix} b_1 & b_2 & \cdots & b_N \end{bmatrix} \begin{bmatrix} z^{-1} \\ z^{-2} \\ \vdots \\ z^{-n} \end{bmatrix} V(z) + h_0 U(z) \\
&= \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} X(z) + h_0 U(z)
\end{aligned}
$$

and the output equation in time-domain is given by

$$
y_t = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} x_t + h_0 u_t.
$$

yielding state-space matrices (A.4.3).

$$
\left[\begin{array}{c|c} \mathsf{A} & \mathsf{B} \\ \hline \mathsf{C} & h_0 \end{array}\right] = \left[\begin{array}{ccccc|c} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n & 1 \\ 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ \hline b_1 & b_2 & \cdots & b_{n-1} & b_n & h_0 \end{array}\right]. \tag{A.4.3}
$$

B. RTF: Further Details

B.1. Fast Companion Recurrence

The recurrent step of a generic SSM (3) with dense system matrices usually requires $\mathcal{O}(n^2)$ operations due to the matrix-vector product $\mathsf{A}x_t$. We show how the recurrence of SSMs in *companion canonical form*, i.e. with system's matrices (A.4.3), requires only $\mathcal{O}(n)$ operations.

Lemma B.1. *The recurrent step of a state-space model in companion canonical form* (A.4.3) *can be evaluated in* $\mathcal{O}(n)$ *time and memory.*

Proof. The companion state matrix A can be broken down into a lower shift matrix L_n and a low-rank term. Particularly, with e_1 the first element of the canonical basis of $\mathbb{R}^n$ and $a = (a_1, \dots, a_n)$, we have

$$\mathsf{A} = \mathsf{L}_n - e_1 \otimes a.$$

It follows that the recurrent update can be simplified to

$$\begin{aligned} x_{t+1} &= (\mathsf{L}_n - e_1 \otimes a)\, x_t + \mathsf{B}u_t \\ y_t &= \mathsf{C}x_t + h_0 u_t \end{aligned}$$

The peculiarity of this formulation is that we never need to construct the full transition matrix to perform the recurrence. In particular we have:

$$\begin{aligned} x^1_{t+1} &= u_t - a^\top x_t \\ x^{2:n}_{t+1} &= \mathsf{shift}(x_t) \\ y_t &= b^\top x_t + h_0 u_t \end{aligned}$$

Thus, each step only requires two inner products (n multiplications and n sums each) and one shift operation, totaling $\mathcal{O}(n)$ operations. □

B.2. Initialization and Stability

Initialization schemes can significantly impact the performance of SSMs, as explored in (Gu et al., 2022b), (Gu et al., 2023), (Orvieto et al., 2023), and (Zhang et al., 2023).

Intriguingly, rational transfer functions allow for initialization schemes that can be directly translated from explicitly parameterized convolutional kernels, as demonstrated below:

$$K_{\mathsf{FIR}}(z) = k_0 + k_1 z^{-1} + k_2 z^{-2} + \cdots + k_{m-1} z^{-(m-1)} \tag{B.2.1}$$

where K_{FIR} represents the z-domain representation of an m-length finite impulse response (a convolutional kernel of size m). It could easily be seen that by simply setting $h_0 = k_0$, $a_i = 0$ and $b_i = k_i$ for $i \in [m]$, the rational transfer function would represent the convolutional kernel. This implies that initialization approaches developed for explicitly parameterized convolutional models, such as (He et al., 2015) and (Glorot & Bengio, 2010), can be directly applied to the rational transfer function representation.

Besides initialization, it is generally desirable for SSMs to be stable, meaning that the roots of the rational transfer function denominator (poles) should reside within a complex unit circle in the z-domain (Chen, 1998). When employing a polar representation of the kernel eigenvalues (poles), in which the roots are parameterized by $\lambda = re^{i\theta}$, the roots r can easily be restricted to $|r| \leq 1$ in various ways such as $r = \exp(-\exp(\nu))$, where $\nu \in \mathbb{R}^n$ as described in (Orvieto et al., 2023). However, for rational transfer functions, where the denominator is represented as a polynomial, ensuring the stability of the SSM is more challenging. Alomari & Chesneau presents several methods for constraining polynomial coefficients, for their roots to lay within the complex unit circle. One such method, Montel's method (Horn & Johnson, 1985), constrains the polynomial roots as follows:

$$\sum_{i=0}^{n-1} |\alpha_i| \leq 1, \tag{B.2.2}$$

This can be implemented straightforwardly using a softmax or an l_1 norm over $n+1$ parameters, and then selecting n parameters from this set, as shown in the following code snippet:

```
def get_constrained_coefs(coefs_plus_scalar):
    """
    coefs_plus_scalar: torch.Tensor of shape [n+1]
    """
    return (coefs/sum(coefs.abs()))[:n] # returns n coefficients that are constrained
    according to Montel's method.
```

Spacetime (Zhang et al., 2023) also utilizes this approach to bound the gradients of their SSMs during training. However, we have found that Montel's method could excessively constrain the SSMs, potentially leading to diminished performance, as shown in Table 5.

Table 5: An ablation of different initialization and parameter constraining approaches.

Model	Wikitext-103 (25 epochs) ppl. ↓	LRA Image acc. ↑
RTF + Xavier Init. + Montel Constraint	26.512	89.2
RTF + Xavier. Init.	-	90.0
RTF + Impulse Init.	**26.093**	**90.1**

Next, we use a 2nd order polynomial case, as a visual illustration of the over-constraining occurring with Montel's method over the parameter space. Given a polynomial $z^2 + \alpha_1 z + \alpha_0$, its roots can be analytically computed with:

$$r = \frac{-\alpha_1 \pm \sqrt{\alpha_1^2 - 4\alpha_0}}{2}. \tag{B.2.3}$$

In the case that $\alpha_1^2 - 4\alpha_0 < 0$, the quadratic equation becomes a summation of a real term and an imaginary term, therefore we can constrain the root to be within the unit circle by computing its norm as follows:

$$\sqrt{\left(\frac{\alpha_1}{2}\right)^2 - \left(\frac{\sqrt{\alpha_1^2 - 4\alpha_0}}{2}\right)^2} \leq 1 \tag{B.2.4}$$

$$\frac{\alpha_1^2 - \alpha_1^2 + 4\alpha_0}{4} \leq 1 \tag{B.2.5}$$

$$\alpha_0 \leq 1. \tag{B.2.6}$$

This shows that the two equations that govern the possible stable regions (for pairs of conjugate roots) are, $\alpha_0 \leq 1$ and $\alpha_0 > \frac{1}{4}\alpha_1^2$. Figure 4 illustrate the space of stable coefficients with a green-blue colormap along with the space of coefficients that obey Montel's constraints in pink. Notice that a sizable portion of the coefficient space that represents a stable SSM with low decay rates is not accessible with the constraint, which hurts SpaceTime's expressivity and enforces a short term bias to the model.

We observed empirically (i.e., Table 5) that setting both numerator and denominator parameters to zeros, and setting $h_0 = 1$, as formulated below,

$$H_\delta(z) = 1 + \frac{0}{z^n}, \tag{B.2.7}$$

generally resulted in RTF having faster training convergence, while simultaneously avoiding instability issues that may be caused via other initialization schemes. The improved stability of this initialization scheme is likely due to it being optimal with respect to satisfying the Montel constraint as follows:

$$\text{argmin}_\alpha \left(\sum_{i=0}^{n-1} |\alpha_i|\right) = \mathbf{0}. \tag{B.2.8}$$

We denote this as the **zero initialization** scheme, and use it throughout all our experiments unless stated otherwise.

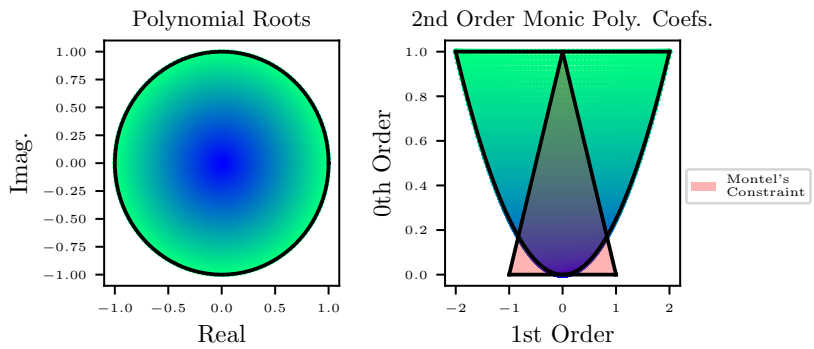

Figure 4: The space of stable roots of a 2nd order polynomial with conjugate roots is illustrated with a green-blue colormap. The figure on the right overlays the space of coefficients that obey Montel's constraints in pink.

B.3. Alternative Inference Algorithms

B.3.1. RTF Kernel Generation via Long Polynomial Division

Given a rational transfer function (TF) representing an infinite length convolutional kernel:

$$H(z) = h_0 + \frac{N(z)}{D(z)} = h_0 + \frac{\sum_0^{n-1} b_i z^i}{\sum_0^n a_i z^i} = h_0 + h_1 z^{-1} + h_2 z^{-2} + \dots, \tag{B.3.1}$$

we would like to directly obtain the truncated (finite length) representation of such a kernel, in order to 1. train RTF numerators that directly correspond to the recurrent form without the need to correct for truncation (which could offer significant speedups in online learning tasks such as reinforcement learning), 2. directly evaluate the truncated transfer function $H(z)$ at 2ℓ points, avoiding the need to convert the frequency domain kernel into time domain for causal padding.

We could take the approach of constructing an infinite length *tail* function, which upon being subtracted from the original TF, results in truncation as follows:

$$H_\ell(z) = H(z) - \tilde{H}_\ell(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + \dots + h_{\ell-1} z^{-\ell+1}. \tag{B.3.2}$$

To satisfy such an equation, we observe that $\tilde{H}_\ell(z) = h_\ell z^{-\ell} + h_{\ell+1} z^{-(\ell+1)} + \dots$, which could be obtained from the original rational transfer function via long division of $N(z)z^L$ against $D(z)$ as shown below:

$$\frac{N(z)z^{\ell-1}}{D(z)} = \underbrace{h_0 z^{\ell-1} + h_1 z^{\ell-2} + h_2 z^{\ell-3} + \dots + h_{\ell-1}}_{C(z)} + \underbrace{h_\ell z^{-1} + h_{\ell+1} z^{-2} + \dots}_{\tilde{H}_\ell(z) z^{-\ell+1}} \tag{B.3.3}$$

$$= C(z) + \tilde{H}_\ell(z) z^{-\ell+1} = C(z) + \frac{R(z)}{D(z)}, \tag{B.3.4}$$

$$\tilde{H}_\ell(z) = \frac{R(z)}{D(z) z^{\ell-1}}. \tag{B.3.5}$$

The naive long division algorithm takes $2np$ operations, in which $p = \ell - n + 1$, however with fast Toeplitz matrix inversion algorithms described in (Pan, 2001), such an algorithm could operate with complexity of $\mathcal{O}(\ell \log \ell)$, assuming $n \ll \ell$.

Next, by simply constructing the truncated transfer function $H_\ell(z)$ via Equation (B.3.2), the padded convolutional kernel in frequency domain can be obtained via transfer function evaluation at 2ℓ points of unity.

B.3.2. Multi-Input Multi-Output RTF

A multi-input multi-output (MIMO) LTI SSM could be represented using a $d \times d$ matrix of numerator polynomials, that shares a denominator polynomial, forming a rational function for each input to output pair. Chen shows that such a system

could be converted back into an SSM realizing the companion form (16) as follows:

$$
\begin{aligned}
x_{k+1} &= \begin{bmatrix} -a_0\mathsf{I}_d & -a_1\mathsf{I}_d & \dots & -a_{n-2}\mathsf{I}_d & -a_{n-1}\mathsf{I}_d \\ \mathsf{I}_d & 0 & \dots & 0 & 0 \\ 0 & \mathsf{I}_d & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & \mathsf{I}_d & 0 \end{bmatrix} x_k + \begin{bmatrix} \mathsf{I}_d \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u \\
y &= \mathsf{C}x_k + \mathsf{D}u,
\end{aligned}
\tag{B.3.6}
$$

in which I_d is a rank d identity matrix, $\mathsf{C} \in \mathbb{R}^{d \times nd}$ corresponds to the matrix of numerator coefficients and $\mathsf{D} \in \mathbb{R}^{d \times d}$. a_i is the denominator polynomial coefficient at order i. We can observe that such a system's C matrix becomes excessively large, making it not competitive in terms of both parallel inference and autoregressive inference speeds against other MIMO systems. For this reason, we focus on the multi SISO (2) companion realization, in which the SSMs are independent across the channel dimension, channel mixing is only done afterwards with a linear projection.

C. Experiments

C.1. Memory and Latency Profiling Experiments

- Experiments were conducted using JAX (Bradbury et al., 2018) on a single A100 80GB GPU for the memory profiling experiments, and on a single H100 80GB GPU for the latency profiling experiments.
- S5 implementation was taken directly from (Smith et al., 2023).
- The memory profiling was done on a single SSM layer with channel size $d = 1024$, whereas the latency profiling was done using $d = 128$.
- Due to S5 being a Multi-Input Multi-Output (MIMO) SSM and RTF being a Single-Input Single-Output SSM, there are few additional points to note on interpreting the results:
  - For fairness we considered a RTF layer with channel mixing, which includes an additional output linear projection layer that mixes the channel dimensions.
  - The RTF layer with channel mixing is equivalent to a block diagonal MIMO SSM with a combined state size of $n_M = dn$. Mamba (Gu & Dao, 2023) makes use of the term *state expansion factor* (e), which describes the state size per channel. For a multi-SISO SSM such as RTF, $e = n$, whereas for a MIMO SSM such as S5, $e = n/d$.
  - Figure 1 and Table 6 compare each SSM layer's memory usage across multiple state sizes (n), whereas Figure 5 and Table 7 compare SSM layer's parallel inference latency across multiple the expansion factors (e).
- For each sequence length ℓ, we collected profiling speeds for RTF and S5 with state-sizes ranging from 256 up to $\ell/2$.
- Table 6 lists the exact peak memory usage in MB. Runs which ran out of the 80GB GPU memory is denoted as `OOM` (Out Of Memory).
- Figure 5 and Table 7 illustrates the median parallel inference latencies (across 100 iterations) in milliseconds.

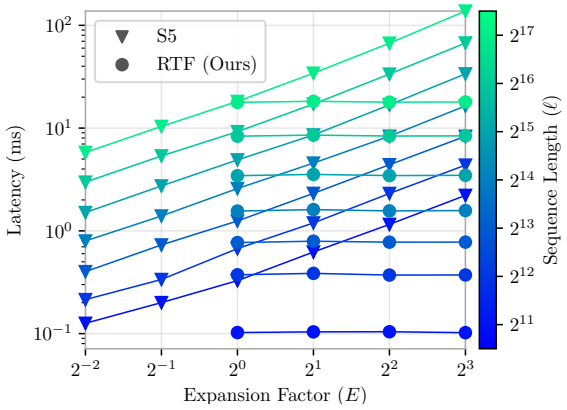

Figure 5: This figure illustrates the scaling of parallel inference latency on S5 and RTF across various sequence lengths and state sizes. When comparing equal expansion factors, it becomes evident that RTF provides lower latencies across different sequence lengths.

C.2. Long Range Arena Benchmark

C.2.1. Model Architecture Details

For fair comparisons with S4 (Gu et al., 2022b) and S4D (Gu et al., 2022a), we employed the same model backbone, block design, and architectural hyperparameters as employed by S4. Each model contains a linear encoder and decoder that projects the inputs and outputs to an appropriate channel dimension. Simply put, each layer is a combination of a SSM layer, an activation function (GELU (Hendrycks & Gimpel, 2023)), followed by an output linear projection layer, and another activation function (GLU (Dauphin et al., 2017)), with skip connections (He et al., 2016) and normalization applied before each every SSM and linear layer. Each channel in a SSM layer comprises of a SISO SSM with the ability

Table 6: Comparison of peak memory usage of S5 and RTF across different state-sizes and sequence lengths in MB.

n \ ℓ	Model	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
2^8	S5	178.0	266.0	459.0	794.01	1510.0	3010.0
	RTF	230.0	454.04	902.0	1760.0	3510.0	7010.0
2^9	S5	192.0	288.01	480.0	864.01	1590.0	3090.0
	RTF	232.04	456.04	904.0	1760.0	3510.0	7010.0
2^{10}	S5	220.0	332.0	592.0	1140.0	2140.0	4270.0
	RTF	236.0	460.04	908.0	1760.0	3510.0	7010.0
2^{11}	S5	308.0	544.02	1030.0	1780.0	3530.0	7030.0
	RTF	244.0	468.04	916.0	1770.0	3520.0	7020.0
2^{12}	S5	-	960.04	1690.0	3310.0	6560.0	13060.0
	RTF	-	484.04	932.0	1790.0	3540.0	7040.0
2^{13}	S5	-	-	3130.0	6130.0	12130.0	24130.0
	RTF	-	-	964.0	1820.0	3570.0	7070.0
2^{14}	S5	-	-	-	11750.0	23250.0	46250.0
	RTF	-	-	-	1880.0	3630.0	7130.0
2^{15}	S5	-	-	-	-	49500.0	OOM
	RTF	-	-	-	-	3750.0	7250.0
2^{16}	S5	-	-	-	-	-	OOM
	RTF	-	-	-	-	-	7500.0

to share the the transition matrix A across channels through the number of SSMs hyperparameter (Num. SSM). This sets the number of unique A matrices (or rational function denominator) which are then equally dispersed across the channel dimensions. Additional hyperparameter details are outlined in Tables 8 and 9.

Experiments using S4 and S4D models used the `PyKeops` implementation, available in the official S4 github repository (Gu et al., 2022b). Fused FFTConv (Fu et al., 2024) algorithms were **not** used for the RTF implementation.

C.2.2. Long Range Arena Benchmark Details

The long range arena (LRA) benchmark (Tay et al., 2021) features 6 unique tasks within lengths of 1K-16K steps. These tasks involve diverse modalities and objectives, pushing models to reason about similarity, structure, and visuospatial relationships.

We offer additional context and specifics for each dataset from the LRA (Tay et al., 2021) that we examine, following the identical data pre-processing procedures as those used by (Gu et al., 2022b).

- `ListOps` An extended dataset introduced by (Nangia & Bowman, 2018). This task involves calculating the integer outcome of mathematical expressions encoded in prefix notation with brackets. Nested operations (min, max, etc.) and operands (0-9) are represented as one-hot vectors (17 unique values, brackets and operators combined). Sequence lengths vary, with max length of 2048. The dataset contains 10 distinct classes, each representing a possible integer outcome, with 96,000 training, 2,000 validation, and 2,000 test sequences.
- `IMDB` Sentiment dataset from (Maas et al., 2011). This task involves classifying movie reviews into positive or negative sentiment categories based on sequences of integer tokens (encoded as one-hot vectors, 129 unique values). Sequence length varies, with a maximum length of 4,096. The dataset consists of 25,000 training and 25,000 test examples.
- `Retrieval` This is derived from the ACL Anthology network corpus introduced by (Radev et al., 2009). The datasets requires determining if two provided textual citations, encoded as a sequence of integer tokens, are the same. Characters are converted into a one-hot vector with 97 unique values. The two paired sequences can have different lengths, with a maximum sequence length of 4,000. There are two categories, signifying whether the citations are equivalent or not. The dataset comprises 147,086 training pairs, 18,090 validation pairs, and 17,437 test pairs.
- `Image` The task utilizes the CIFAR-10 dataset introduced by (Krizhevsky, 2009). It involves classifying a 32×32

Table 7: Comparison of parallel inference latency of a single SSM layer in milliseconds across different sequence lengths and expansion factors (e) of RTF and S5. We report the median value across 100 runs.

e \ ℓ	Model	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
0.25	S5	0.126	0.214	0.401	0.798	1.508	2.980	5.805
	RTF	-	-	-	-	-	-	-
0.5	S5	0.200	0.336	0.730	1.396	2.737	5.364	10.384
	RTF	-	-	-	-	-	-	-
1	S5	0.328	0.672	1.248	2.582	4.906	9.232	18.184
	RTF	0.102	0.373	0.770	1.558	3.443	8.345	17.824
2	S5	0.623	1.194	2.317	4.566	8.573	17.137	34.219
	RTF	0.104	0.385	0.793	1.610	3.543	8.547	18.220
4	S5	1.156	2.312	4.403	8.334	16.830	33.545	67.033
	RTF	0.104	0.372	0.777	1.564	3.451	8.372	17.895
8	S5	2.214	4.363	8.328	16.438	33.714	67.336	137.632
	RTF	0.102	0.372	0.777	1.577	3.473	8.402	17.963

Table 8: Table with the hyperparameters used for classification datasets. BN and LN refer to Batch Normalization and Layer Normalization.

	Layers	Channels	SSM State Size	Num. SSM	Norm.	Batch Size	Epochs
ListOps	6	256	4	1	BN	32	50
Text	6	256	4	1	BN	16	32
Retrieval	6	256	4	1	BN	64	20
Image	6	512	64	1	BN	50	200
Pathfinder	6	256	64	256	BN	64	200
Path-X	6	256	64	256	BN	16	50

grayscale CIFAR-10 image, presented as a one-dimensional raster scan, into one of ten categories. All sequences have the same length (1,024). The dataset comprises 45,000 training examples, 5,000 validation examples, and 10,000 test examples.

- `Pathfinder` This is derived from the Pathfinder challenge, as presented by (Linsley et al., 2018). It involves a 32 × 32 grayscale image that displays a start and an end point, each represented by a small circle. The image contains several dashed lines. The objective is to determine whether a dashed line (or path) connects the start and end points. There are two classes, signifying whether a valid path exists or not. All sequences have the same length (1,024). The dataset includes 160,000 training examples, 20,000 validation examples, and 20,000 test examples.

- `Path-X` This is a variant of the Pathfinder challenge. With a longer sequence and more complex, in this version, the images are 128 × 128 pixels, leading to sequences that are sixteen times longer.

C.3. Synthetic Memorization Tasks

Both implementations of *Copying* (Arjovsky et al., 2016) and *Delay* (Gu et al., 2023) were taken directly from the official S4 repository (Gu et al., 2022b), and was modified to enable drop in replacements of our RTF SSMs under identical conditions.

C.3.1. *Copying* Task

Each model is first fed a ℓ_{mem} length sequence of integer tokens randomly sampled from $0, ..., d-2$, and then fed a ℓ_{mem} length sequence of token number $d-1$ to recall the initial sequence. Table 10 lists the task hyperameters.

The overall model architecture is identical to that described in Section C.2.1. Each model was trained with 4 layers, 1024 channel dimensions, and the number of SSM was set to 1 (for weight sharing). Additionally, we initialized the RTF

Table 9: Table with the layer hyperparameters used for classification datasets.

	Model	Dropout	LR	WD	SSM LR	SSM WD
ListOps	S4	0.0	0.01	0.05	0.001	0.0
	S4D	0.0	0.01	0.05	0.001	0.0
	RTF	0.0	0.002	0.07	-	-
Text	S4	0.0	0.01	0.05	0.001	0.0
	S4D	0.0	0.01	0.05	0.001	0.0
	RTF	0.1	0.005	0.05	0.0001	0.025
Retrieval	S4	0.0	0.01	0.05	0.001	0.0
	S4D	0.0	0.01	0.05	0.001	0.0
	RTF	0.0	0.002	0.0	1e-6	0.0
Image	S4	0.1	0.01	0.05	0.001	0.0
	S4D	0.1	0.01	0.05	0.001	0.0
	RTF	0.1	0.006	0.05	0.005	0.05
Pathfinder	S4	0.0	0.004	0.05	0.001	0.0
	S4D	0.0	0.004	0.05	0.001	0.0
	RTF	0.1	0.002	0.05	-	-
Path-X	S4	0.0	0.002	0.05	0.001	0.0
	S4D	0.0	0.002	0.05	0.001	0.0
	RTF	0.1	0.001	0.05	0.001	0.0

Table 10: *Copying* Task Hyperparameters.

Configuration	Value
ℓ_{mem}	1024
Vocab. size d	64
Train-set Size	10000
Test-set Size	1000
Batch Size	8
Epochs	50
LR	0.001
WD	0.0

parameters by uniformly sampling from a range of 0 to 1, and applying the Montel constraint to limit the poles to a stable location.

C.3.2. *Delay* TASK

The models are given a signal of length ℓ_{seq} and are tasked to output the original signal shifted by ℓ_{delay} timesteps. The input is a white noise signal bandlimited to 1000 Hz. A single layer SSM with channel dimensions of 4 without a non-linear activation function was used for this experiment. Table 11 lists the task hyperameters.

C.4. Laughing Hyena Distillation Task

- The baseline 160M parameter MultiHyena-Attention hybrid model consists of 6 Attention layers and 6 MultiHyena layers.
- The distillation task aims to replace the Hyena filters in the 6 MultiHyena layers with an RTF or a modal SSM.
- Each MultiHyena layer consist of 256 independent SISO convolutional filters, which are projected to 768 dimensions as described in (Massaroli et al., 2023).

Table 11: *Delay* Task Hyperparameters.

Configuration	Value
ℓ_{seq}	4000
ℓ_{delay}	1000
Batch Size	64
Epochs	20
LR	0.001
WD	0.0

Table 12: WikiText103 language modeling perplexity scores (25 epochs).

Model	Perplexity ↓
S4-4	26.86
S4D-4	26.98
RTF-4	**26.36**
S4-64	26.82
S4D-64	26.67
RTF-64	**26.01**
S4-256	-
S4D-256	26.70
RTF-256	**26.32**

Table 13: Wikitext103 Hyperparameters.

Configuration	Value
Sequence length	1024
Batch Size	16 (128 global)
Epochs	100
LR	0.001
WD	0.25
Dropout	0.25
SSM State Size	64
Channels	768
Layers	12
Low-Rank Dims.	384

- Both LH and RTF were trained for 1e6 iterations, on the AdamW (Loshchilov & Hutter, 2019) optimizer with learning rates set to $1\text{e}{-4}$.

C.5. WikiText103 Language Modeling

C.5.1. Pilot Experiments

We additionally compared S4, S4D, and RTF on WikiText103 under the modified Transformer backbone (Baevski & Auli, 2019), from the official S4 repository (Gu et al., 2022b), via drop-in replacements of S4 with S4D and RTF, while keeping the original hyperparameters. Table 12 shows perplexity scores for the models across multiple state-sizes, trained for 25 epochs on two 40GB A100 GPUs. The results show a consistent trend of RTF outperforming S4 and S4D across multiple state-sizes.

C.5.2. Model Architecture Details

For our main WikiText103 experiment, we constructed Hyena-RTF by simply replacing the Hyena Filters in the Hyena Hierarchy model (Poli et al., 2023a) implemented in the `HazyResearch/safari` Github repository, with our RTF SSM. We also made slight modifications to the Hyena operator's output linear projection, by inserting an additional low-rank linear layer and a GELU (Hendrycks & Gimpel, 2023) activation, before the final output linear projection. This is to functionally mimic the low-rank MIMO SSM + non-linear activation function that Hyena-S5 (Smith et al., 2023) employs. It is worth noting that the additional low-rank layer does not increase parameter count since the original output linear projection also loses rank for compatibility of dimensions. We observed that the zero-initialization alone was not enough for the model to stay within the stable region across training – an important property for extrapolative tasks such as language generation. Therefore, we instead adopt the Xavier initialization (Glorot & Bengio, 2010) over the rational function coefficients and apply the Montel constraint via an $\ell 1$ penalization as shown in Section B.2. Table 13 lists the hyperparameters used to train our Hyena-RTF model.