Fractional State Space Transition for Long Sequence Modeling

Ivan Kobyzev* **Abbas Ghaddar*** **Ali Nasiri-Sarvi** **Lifeng Shang** **Yufei Cui**
Huawei Noah's Ark Lab, Montreal Research Center, Canada
{ivan.kobyzev,abbas.ghaddar,shang.lifeng,yufei.cui}@huawei.com

Abstract

State Space Models (SSMs) compress sequence history into a bounded recurrent state, making the resulting memory law a central architectural choice for long-context performance. Most modern SSMs rely on ODE-based dynamics that lead to exponential forgetting, limiting their ability to retain information over broad temporal ranges. We introduce FRAC, a selective SSM architecture derived from fractional dynamics that replaces this exponential decay with power-law long memory. To make fractional dynamics practical, FRAC approximates the heavy-tailed target kernel with a finite-state, log-spaced sum of exponential modes. This construction turns fractional memory into an efficient recurrent module with parallel training and prefill, while retaining bounded-state autoregressive decoding. Extensive experiments, including 1.3B-parameter language modeling, demonstrate that FRAC consistently improves long-context performance over state-of-the-art SSM baselines while staying competitive on short-context. These results show that fractional dynamics provide a practical and effective prior for long-context SSMs. Code: https://github.com/anasiri/frac-ssm

1 Introduction

Due to the quadratic complexity of the Transformer self-attention module [62], a line of research has focused on developing more efficient linear alternatives based on State Space Models (SSMs) [22, 67, 10, 68]. SSMs efficiency is based on compressing the entire sequence history into a finite recurrent state. This compression constraint makes the model's internal memory a central design choice for long-context sequence modeling, which can be naturally understood through the memory law induced by the underlying dynamics.

Most SSMs, such as Mamba [20, 10, 35], base their internal memory design on ordinary differential equations (ODEs) and their discretizations [22, 24]. Despite their success and widespread adoption, the ODE dynamics underlying these models naturally induce exponential forgetting, causing the influence of past inputs to decay exponentially with lag [63]. That inductive bias is not optimal for modeling long sequences in which past events can retain non-negligible influence over broad temporal ranges. Fractional differential equations (FDEs) have been widely used in physics and applied mathematics to model such hereditary effects, as their solutions depend on the full history through heavy-tailed kernels [12, 43, 46, 44]. Thus, FDEs can offer an alternative foundation for designing the internal memory of SSMs, rather than refining information selection or routing within exponentially forgetting ODE-based recurrent models [69]. However, a major challenge in making FDEs practical for SSMs is that, unlike ODEs, FDEs are non-Markovian: the state at a given time depends on the entire past, making them not directly compatible with finite-state recurrent layers.

In this paper, we introduce FRAC, a novel selective SSM architecture derived from fractional dynamics. We develop a theoretical framework that makes fractional long memory compatible with efficient

*Equal contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

recurrent computation by approximating the target power-law kernel with a finite-state, log-spaced sum of exponential modes. The resulting FRAC layer retains the computational advantages of modern selective SSMs, while also supporting hardware-efficient parallel training, prefill, and bounded-state autoregressive decoding.

On synthetic long-tail and recall-retrieval benchmarks, FRAC achieves the strongest length extrapolation and highest recall performance among state-of-the-art SSMs, consistent with its intended heavy-tailed inductive bias and long-memory behavior. Furthermore, when training 1.3B-parameter language models from scratch, FRAC outperforms strong SSM baselines such as Mamba and Gated DeltaNet (GDN) [69] on long-context evaluations, while remaining competitive on standard short-context language-modeling evaluations. Taken together, these results suggest that the memory law induced by the underlying dynamics is a powerful architectural axis for designing efficient long-context sequence models.

2 Preliminaries

In this section, we review how dynamical systems with long memory can be modeled using FDEs, then we develop the theoretical framework that underlies the design of our FRAC model in Section 3.

2.1 Long memory in dynamical systems

Classical state-space models can be viewed as discrete-time counterparts of ordinary differential equations. In this setting, memory typically decays exponentially, so the influence of past events is controlled by a characteristic timescale. This is a natural model for many Markovian systems, where the future depends on the present state alone. However, many systems in nature do not behave this way. In anomalous diffusion, dielectric relaxation, viscoelasticity, and related transport phenomena, the present state depends on a broad and weighted history of the past rather than on a single characteristic timescale [46, 44]. Such systems exhibit long-memory or hereditary behavior, often described by kernels with heavy tails. Fractional calculus [12, 43] provides a convenient mathematical language for this regime. By allowing derivatives of non-integer order, it describes dynamics in which memory is distributed over the past rather than concentrated around a single exponential decay law.

2.2 Caputo fractional differential equations

To make the discussion above formal, let us start with the familiar first-order linear dynamics:

$$\dot{h}(t) = -\lambda h(t) + u(t), \qquad \lambda > 0. \tag{1}$$

This is the continuous-time analogue of a simple state-space update. Here the evolution of the state $h(t)$ is determined by the external input $u(t)$ together with the linear term $-\lambda h(t)$, which causes past information to decay at a rate set by λ. In this sense, the system is organized around a characteristic timescale of order λ^{-1}, which controls how quickly the influence of the past fades. As discussed in Section 2.1, this picture is no longer adequate when memory is distributed over the past rather than concentrated around a single scale.

A standard way to model this regime is to replace the ordinary derivative by a fractional derivative. In this paper, we use the Caputo fractional derivative [12]. For $0 < \alpha < 1$, it is defined by

$${}^{C}D_t^{\alpha} f(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{f'(\tau)}{(t-\tau)^{\alpha}}\, d\tau. \tag{2}$$

This expression makes the long-memory mechanism explicit: the derivative at time t depends on the entire past history on $[0, t]$, with recent increments weighted more strongly but with a long algebraic tail that keeps older events relevant. Replacing the ordinary derivative in (1) by the Caputo derivative gives the fractional relaxation equation:

$${}^{C}D_t^{\alpha} h(t) = -\lambda h(t) + u(t), \qquad \lambda > 0, \tag{3}$$

which we use as the basic continuous-time model of long-memory dynamics. When $\alpha = 1$, (3) reduces to the ordinary first-order system (1). For $0 < \alpha < 1$, the dynamics become nonlocal in time and the effective memory kernel becomes heavy-tailed. The parameter α therefore controls the

strength of the long-memory effect: values closer to 1 recover more local, ODE-like behavior, while smaller values produce slower forgetting and broader memory across timescales [44].

Eq. (3) gives the right continuous-time model of long-memory dynamics, but it is not yet in the form needed for a state-space layer. Direct discretizations of FDEs are typically history-dependent: the update at time t depends on the entire past trajectory rather than on a finite-dimensional recurrent state [12]. Rather than using a generic black-box fractional solver, we will construct a Markovian approximation of this dynamics that is compatible with efficient recurrent computation. Next, we develop this construction by expressing the fractional relaxation kernel in terms of Mittag–Leffler functions and then lifting it into a representation that admits a finite state-space realization.

2.3 Mittag–Leffler relaxation and fractional kernels

To understand what kind of memory law is induced by (3), we now ask the same question one would ask for an ordinary linear system: how does the state decay in the absence of input, and how does it respond to an external forcing? In the classical case, both are governed by the exponential function. In the fractional case, the corresponding role is played by the Mittag–Leffler function [19].

We first consider the homogeneous version of (3): ${}^{C}D_t^{\alpha} h(t) = -\lambda h(t)$. Its solution is: $h(t) = E_\alpha(-\lambda t^\alpha)\, h(0)$, where

$$E_\alpha(z) := \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + 1)} \tag{4}$$

is the one-parameter Mittag–Leffler function [42]. Thus, in the fractional setting, the exponential decay law of the ordinary system is replaced by Mittag–Leffler relaxation. We next return to the forced system (3). Its response to the input is described by the impulse-response kernel

$$g_{\alpha,\lambda}(t) := t^{\alpha-1} E_{\alpha,\alpha}(-\lambda t^\alpha), \qquad \text{where} \quad E_{\alpha,\beta}(z) := \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + \beta)} \tag{5}$$

is the two-parameter Mittag–Leffler function. The full solution to the FDE (3) can be written as:

$$h(t) = E_\alpha(-\lambda t^\alpha)\, h(0) + \int_0^t g_{\alpha,\lambda}(t-\xi)\, u(\xi)\, d\xi, \tag{6}$$

so the kernel $g_{\alpha,\lambda}$ describes how past inputs are accumulated over time [42].

Remark 1. Fractional relaxation can be informally associated with power-law memory. More precisely, the relevant Mittag–Leffler kernels are not exact power laws for all t, but they do have asymptotically polynomial tails as $t \to \infty$. For any fixed $\alpha \in (0,1)$ and $\lambda > 0$, the Mittag–Leffler homogeneous kernel satisfies [19]:

$$E_\alpha(-\lambda t^\alpha) = \frac{1}{\lambda\,\Gamma(1-\alpha)}\, t^{-\alpha} + O(t^{-2\alpha}), \qquad t \to \infty. \tag{7}$$

Likewise, the impulse-response kernel satisfies [19]:

$$g_{\alpha,\lambda}(t) = t^{\alpha-1} E_{\alpha,\alpha}(-\lambda t^\alpha) = \frac{\alpha}{\lambda^2\,\Gamma(1-\alpha)}\, t^{-\alpha-1} + O(t^{-2\alpha-1}), \qquad t \to \infty. \tag{8}$$

In particular, the homogeneous solution decays as $t^{-\alpha}$, in contrast to the exponential decay of the ordinary first-order system. This asymptotically polynomial tail is the source of the broader timescale coverage that motivates the fractional construction for long-memory sequence modeling.

At the same time, the solution (6) is still not in the finite-dimensional Markovian form needed for an efficient state-space layer. The next subsection addresses this issue by rewriting the fractional kernel in a form that can be approximated by a finite bank of exponential modes.

2.4 Diffusive representations of fractional kernels

The key structural fact we need is that the fractional kernel admits a representation as a continuous mixture of ordinary exponential decays.

Theorem 1. *For $0 < \alpha < 1$ and $\lambda > 0$, the kernels appearing in the solution* (6) *of the fractional differential equation* (3) *admit nonnegative diffusive representations. In particular, there exist nonnegative densities $R_{\alpha,\lambda}$ and $H_{\alpha,\lambda}$ such that*

$$E_\alpha(-\lambda t^\alpha) = \int_0^\infty e^{-t/\tau}\, R_{\alpha,\lambda}(\tau)\, d\tau, \qquad g_{\alpha,\lambda}(t) \quad = \int_0^\infty e^{-t/\tau}\, H_{\alpha,\lambda}(\tau)\, d\tau. \tag{9}$$

See Appendix A for more details. This representation is still infinite-dimensional. The next subsection shows how to approximate it on a bounded horizon by a finite bank of exponential modes, which is the form we will later turn into a discrete state-space transition.

2.5 Finite sum-of-exponentials (SoE) approximation on a bounded horizon

The diffusive representation of Theorem 1 still involves a continuum of timescales and therefore cannot be used directly as a finite-state recurrent transition. To obtain a finite memory bank, we approximate this integral on the bounded range of timescales relevant for the horizon of interest by a finite weighted sum of exponential modes. Because fractional kernels spread mass across many orders of magnitude in time, this construction is naturally organized on a logarithmic timescale grid. Related exponential-sum constructions of this type are standard in numerical methods for fractional kernels [28, 6]. The next theorem states the exact structural facts from this approximation that is the key component for our model in Section 3.

Theorem 2. *Fix $0 < \alpha < 1$, $\lambda > 0$, and $0 < T < \infty$. Then for every $\varepsilon > 0$ there exist $M \in \mathbb{N}$, $\tau_0 > 0$, and $q > 1$, a geometrically spaced bank of positive timescales $\tau_m = \tau_0 q^{m-1}$ for $m = 1, \dots, M$, and positive coefficients $c_m(\alpha, \lambda)$, $d_m(\alpha, \lambda)$ such that*

$$\sup_{t\in[0,T]} \left| E_\alpha(-\lambda t^\alpha) - \sum_{m=1}^{M} c_m(\alpha,\lambda) e^{-t/\tau_m} \right| \le \varepsilon, \tag{10}$$

$$\int_0^T \left| g_{\alpha,\lambda}(t) - \sum_{m=1}^{M} d_m(\alpha,\lambda) e^{-t/\tau_m} \right| dt \le \varepsilon. \tag{11}$$

Moreover, the coefficients may be chosen so that

$$\frac{d_m(\alpha,\lambda)}{c_m(\alpha,\lambda)} = \frac{1}{\lambda\tau_m}, \qquad m = 1, \dots, M,$$

and, for the large-timescale part of the geometric bank τ_m:

$$c_m(\alpha,\lambda) = \frac{(\log q)\sin(\pi\alpha)}{\pi\lambda}\, \tau_m^{-\alpha}\big(1 + O(\tau_m^{-\alpha})\big). \tag{12}$$

Proof. See Appendix B.

3 Method

We now turn the continuous-time fractional memory picture of Section 2 into the discrete selective recurrence underlying the FRAC layer, our selective fractional state-space module. The construction has three steps. First, we approximate the target long-memory kernel by a finite bank of exponential modes. Second, we discretize this mode bank exactly under a zero-order-hold (ZOH) assumption. Third, we make the resulting transition selective through token-dependent control variables and mode-wise read and write weights.

3.1 From finite SoE kernels to a memory bank

Consider the shared-bank SoE approximation from Theorem 2 with M memory modes. We now show that, on the horizon of interest, the fractional dynamical system is approximated by a bank of M first-order ODEs.

Proposition 3. *Fix $0 < T < \infty$ and $\varepsilon > 0$, and let $\{\tau_m, c_m(\alpha, \lambda), d_m(\alpha, \lambda)\}_{m=1}^{M}$ be the common-bank coefficients given by Theorem 2 on $[0, T]$ for this tolerance ε. Given $u \in L^\infty([0, T])$, denote by h the solution on $[0, T]$ of the fractional differential equation* (3) *with input u.*

Consider the system of ODEs

$$\dot{s}_m(t) = -\frac{1}{\tau_m} s_m(t) + a_m(\alpha, \lambda)\, u(t), \qquad m = 1, \ldots, M, \tag{13}$$

with initial conditions $s_m(0) = h(0)$ for $m = 1, \ldots, M$. If

$$a_m(\alpha, \lambda) := \frac{d_m(\alpha, \lambda)}{c_m(\alpha, \lambda)} = \frac{1}{\lambda \tau_m}, \qquad m = 1, \ldots, M, \tag{14}$$

and if the bank readout is defined by

$$\tilde{h}_M(t) := \sum_{m=1}^{M} c_m(\alpha, \lambda)\, s_m(t), \tag{15}$$

then

$$\sup_{t \in [0,T]} |h(t) - \tilde{h}_M(t)| \le \varepsilon \Big(|h(0)| + \|u\|_{L^\infty([0,T])} \Big). \tag{16}$$

In particular, $\tilde{h}_M$ approximates h uniformly on $[0, T]$.

For the proof see Appendix C. This gives a finite continuous-time realization of the target long-memory kernel. Next, we discretize this mode bank and turn it into the selective recurrent transition.

3.2 Frac State Transition

We now convert the finite continuous-time mode bank of Proposition 3 into a token-level recurrent transition by discretizing the mode dynamics under the standard zero-order-hold (ZOH) [20] procedure. On each interval, the control variables are frozen and the input is held constant, so each mode evolves as a scalar first-order linear system. Let u_t denote the held input and let $\Delta_t > 0$ denote the interval length. Applying the exact ZOH discretization to (13) on the interval $[0, \Delta_t)$, one gets:

$$s_{t,m} = \rho_{t,m} s_{t-1,m} + \beta_{t,m} u_t, \tag{17}$$

where

$$\rho_{t,m} = \exp(-\Delta_t / \tau_m), \qquad \beta_{t,m} = \frac{1 - \rho_{t,m}}{\lambda}. \tag{18}$$

Thus $\rho_{t,m}$ determines how much of the previous state is retained over the interval, while $\beta_{t,m}$ is the exact ZOH injection factor. To specialize this recurrence to the fractional setting, we use the self-similarity in λ from Remark 4. That remark shows that varying λ rescales the underlying timescale axis by the factor $\lambda^{-1/\alpha}$. We keep a shared geometric bank of base timescales $\{\tau_m\}_{m=1}^{M}$ and implement this effect through the token-dependent effective timescale

$$\tilde{\tau}_{t,m} = \frac{\tau_m}{\lambda_t^{1/\alpha_t}}. \tag{19}$$

Substituting $\tilde{\tau}_{t,m}$ for τ_m in (18) gives

$$\tilde{\rho}_{t,m} = \exp\Big(-\Delta_t \lambda_t^{1/\alpha_t} / \tau_m \Big), \qquad \tilde{\beta}_{t,m} = \frac{1 - \tilde{\rho}_{t,m}}{\lambda_t}. \tag{20}$$

Thus Δ_t, α_t, and λ_t determine both the mode retention factors and the exact ZOH injection factors. The learned routing introduced next provides additional content-dependent modulation over this fractional transition.

The recurrence (17) gives the nonselective update of the mode bank. We introduce token-dependent write weights $b_{t,m}$ and read weights $c_{t,m}$ over the modes. The resulting selective update is:

$$s_{t,m} = \tilde{\rho}_{t,m} s_{t-1,m} + \kappa_{t,m} u_t, \tag{21}$$

$$h_t = \sum_{m=1}^{M} c_{t,m} s_{t,m}, \tag{22}$$

where $\kappa_{t,m} = \tilde{\beta}_{t,m} b_{t,m}$. Thus $\tilde{\rho}_{t,m}$ and $\tilde{\beta}_{t,m}$ set the transition rule of the mode bank, while $b_{t,m}$ and $c_{t,m}$ determine how the input is distributed across modes and how the updated bank is read out.

On the read side, the fractional theory gives the asymptotics (12) for the readout coefficients c_m. We therefore use the log-timescale prior $-\alpha_t \log \tau_m$ and define

$$c_{t,m} = \mathrm{softmax}_m(-\alpha_t \log \tau_m + g_{\mathrm{read}}(u_t)_m)\,, \tag{23}$$

where g_{read} is a learned linear map from the content signal to mode-wise residual logits.

The write prior is not uniquely fixed by the theory. In the implementation used in this paper, however, we define $b_{t,m}$ using the same functional form as in (23), but with an independent linear map g_{write}. This gives the read and write pathways a common log-timescale addressing scheme over the mode bank, which empirically leads to better alignment between where information is written and how it is later retrieved. See Algorithm 1 in Appendix D for the details.

Remark 2. The transition (21) remains a first-order affine recurrence. For a fixed head and feature coordinate, stacking the M mode states into a vector $\mathbf{s}_t \in \mathbb{R}^M$ gives

$$\mathbf{s}_t = \mathbf{\Lambda}_t \mathbf{s}_{t-1} + \mathbf{b}_t, \qquad \mathbf{\Lambda}_t = \mathrm{Diag}(\tilde{\rho}_{t,1}, \ldots, \tilde{\rho}_{t,M}), \tag{24}$$

where $\mathbf{b}_t$ collects the write terms $\kappa_{t,m} u_t$. Because this is an affine recurrence, training and prefill can be implemented with a chunked parallel scan, following the standard scan composition used in modern selective state-space models [5, 20]. Autoregressive decoding uses the same rule in one-step recurrent form with cached states. We implement this computation with a custom Triton kernel.[2]

3.3 The FRAC layer

Figure 1 (left) summarizes the design of FRAC layer, which follows the common practice of recent linear-time sequence models [10, 69]. Starting from the hidden state, we first apply normalization, then an input projection, and pass the result to the FracMixer block. This returns the updated hidden representation, which is then processed by gated normalization and a final output projection.

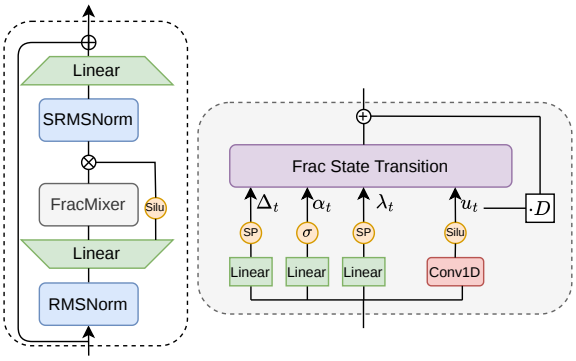


Figure 1: Overview of the FRAC layer (left), and FRACMixer block (right).

FracMixer is shown in Figure 1 (right). Following the common design pattern of recent linear models [10, 69], the input is split into a pre-convolution (control) branch and a post-convolution (content) branch. The control branch produces the token-wise variables Δ_t, α_t, and λ_t. As in Mamba [20], Δ_t is obtained from a linear projection with learned bias followed by a softplus (SP) transform, while α_t and λ_t are produced by separate linear projections followed by pointwise nonlinearities: sigmoid for α and SP for λ enforcing $0 < \alpha_t < 1$ and $\lambda_t > 0$. The content branch produces u_t by applying a local depthwise causal 1D convolution along the sequence. The Frac state transition module then applies the selective fractional recurrence across the sequence, as described in Section 3.2. Its output is combined with the direct feedthrough term Du_t, where D is a learnable parameter, mirroring the standard direct feedthrough term in classical state-space models [22].

4 Experiments

4.1 Synthetic Benchmarks

Heavy Tail Probing As a controlled probe of the memory law, we introduce a simple synthetic extrapolation task where sparse events must be accumulated with a fixed power-law decay over distance. We compare the FRACMixer block with its counterparts from prior linear-modeling work, including MAMBA2 [10], GDN [69], and MAMBA3 [35], as well as with vanilla self-attention [62].

[2] See Appendix F for additional implementation details and latency benchmarking.

All models use a single layer with 200K parameters, are trained only on sequences of length 512, and are evaluated on sequences up to 128K tokens.[3] Figure 2 shows that while all models degrade as the test context length increases, FRAC consistently exhibits the smallest performance decay. While GDN and MAMBA3 remain the closest competitors to our model, Attention rapidly drops to near-random performance starting at 8K. These results suggest that changing the memory law itself can lead to substantially better length generalization.

MADLab We next evaluate on MAD-Lab [53], a suite of synthetic tasks that probe sequence-modeling mechanisms including compression (Comp.), fuzzy in-context recall (F-ICR), memorization (Mem.), and selective copying (SC). All models are trained from scratch and have four layers with roughly 500K parameters, alternating sequence-mixing layers and SwiGLU channel-mixing layers.[4] As shown in Table 1, FRAC is on par with the strongest linear baselines and slightly improves the overall average. These results suggest that replacing the standard ODE-based memory law with an FDE-based one preserves the core mechanistic abilities of SSMs.

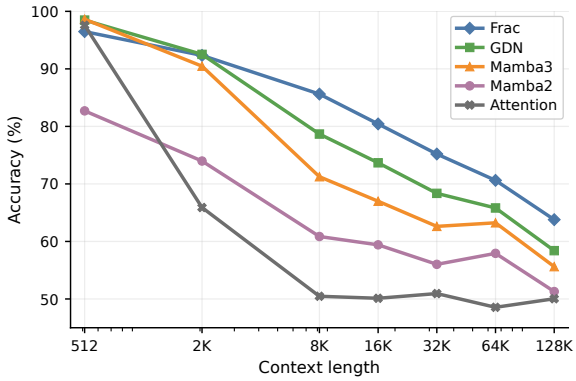


Figure 2: Model performance on heavy-tail synthetic extrapolation task.

Table 1: Performance (accuracy) on the synthetic MAD-Lab benchmark over 5 runs.

Model	Comp.	F-ICR	Mem.	SC	Avg.
MAMBA2	$88.3_{\pm 0.2}$	$7.2_{\pm 0.3}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	73.8
GDN	$88.9_{\pm 0.1}$	$9.7_{\pm 0.2}$	$98.8_{\pm 0.1}$	$99.9_{\pm 0.1}$	74.3
MAMBA3	$89.1_{\pm 0.2}$	$10.1_{\pm 0.4}$	$100.0_{\pm 0.0}$	$100.0_{\pm 0.0}$	74.8
FRAC	$90.4_{\pm 0.2}$	$11.5_{\pm 0.2}$	$99.9_{\pm 0.1}$	$99.9_{\pm 0.1}$	75.4

4.2 Language Modeling

Setup We compare FRAC with state-of-the-art models, including the linear models MAMBA2 [10], GDN [69], and MAMBA3 [35] (both -*SISO* and -*MIMO* variants), as well as Vanilla Transformer [66]. Following [69], we pretrain 1.3B-parameter LLMs from scratch on 100B tokens sampled from the deduplicated FineWeb-Edu [52, 3] corpus, training all models under the same standard protocol. We use the Llama-2 [60] tokenizer with a 32k-token vocabulary and perform training on fully packed sequences with a length of 4k. Following prior work [10, 68, 35], we evaluate models on long-context needle-in-a-haystack tasks [30] (**NIAH**) and **LongBench** [2], as well as short-context language modeling benchmarks (**LM Harness**) and real-world intensive recall-retrieval tasks [1] (**Recall-Retrieval**). A detailed description of the training, implementation, and evaluation protocols is provided in Appendix E.3.

NIAH Unlike prior works, we evaluate NIAH far beyond the 4K training sequence length, testing extrapolation up to 64K tokens. Figure 3 shows that FRAC performs on par with baseline linear models on short-context settings ($\leq$ 4K), while demonstrating substantially stronger length generalization once the context exceeds the training range. In particular, as the sequence length increases from 8K to 64K, the performance of FRAC drops more slowly than that of the baseline models. The main exception is GDN on *S-NIAH-1*, where its gated delta rule is especially effective at filtering repetitive context [69].

LongBench. Table 2 shows that FRAC improves long-context language processing tasks over both the Transformer and linear-model baselines. In particular, FRAC outperforms the strongest linear baseline GDN by 1.9% on average, and reports the best result on 8/14 tasks. The gains are smaller than on NIAH, which is expected because more than half of LongBench tasks have average input lengths below 8K tokens, so the Transformer remains competitive with linear recurrent models. Nevertheless, these results suggest that the fractional memory mechanism retains useful information over longer spans more effectively than standard exponential-decay recurrent models.

[3] Models and additional implementation details are presented in Appendix E.1.

[4] MADLab also includes plain ICR and noisy ICR tasks, which we do not report because performance saturates for all models. See Appendix E.2 for additional implementation details.

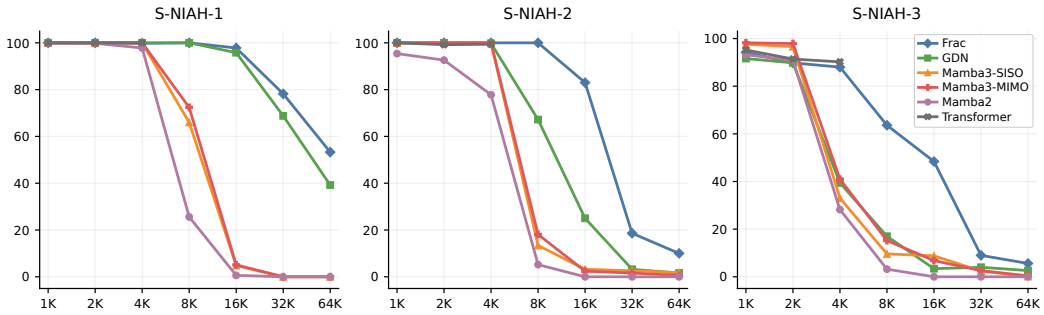


Figure 3: Model accuracies on three variants of the *needle-in-a-haystack* passkey retrieval task, with sequence lengths ranging from 1K to 64K. Transformer performance is zero after 4K.

Table 2: Model performance on 14 LongBench tasks grouped under 5 categories. The highest and second-highest scores are highlighted in bold and underline, respectively.

Model	Single-Doc QA			Multi-Doc QA			Summarization			Few-shot			Code		Avg.
	NQA	QQA	MFQ	HQA	2WM	Mus	GvR	QMS	MNs	TRC	TQA	SSM	LCC	RBP	
Transformer	8.7	10.6	19.0	6.1	15.5	4.0	7.2	16.6	11.2	25.0	19.1	11.3	**26.0**	25.6	14.7
MAMBA2	4.4	11.4	16.3	4.5	13.8	2.7	7.8	12.7	6.3	17.0	10.9	10.5	23.4	**26.5**	12.0
GDN	11.5	10.1	24.4	7.3	14.8	4.8	7.9	18.9	14.3	**40.0**	21.6	13.2	18.7	16.9	16.0
MAMBA3-*SISO*	11.2	12.4	22.6	8.4	**16.3**	**6.4**	9.8	**19.3**	9.6	13.5	18.2	14.5	20.6	18.3	14.4
MAMBA3-*MIMO*	5.0	10.4	15.3	8.2	12.5	5.6	10.5	14.3	8.4	8.5	8.6	11.6	20.7	19.9	11.4
FRAC	**13.2**	**14.0**	**25.8**	**8.5**	12.2	5.1	**13.8**	18.9	**16.3**	39.0	**24.5**	**28.2**	11.0	19.5	**17.9**

Table 3: Model performance on short-context language modeling and understanding tasks. The best and second-best scores for each metric are highlighted in bold and underline, respectively. The average is computed by excluding the first two columns.

Model	**Wiki.** ppl ↓	**LMB.** ppl ↓	**LMB.** acc ↑	**PIQA** acc ↑	**Hella.** acc_n ↑	**Wino.** acc ↑	**ARC-e** acc ↑	**ARC-c** acc_n ↑	**OBQA** acc ↑	**BoolQ** acc ↑	**Avg.**
Transformer	**15.3**	13.6	45.6	72.1	57.6	**59.5**	70.7	35.8	39.4	**61.9**	**55.3**
MAMBA2	16.7	13.5	40.9	72.1	57.1	54.0	71.2	38.3	40.3	56.2	53.8
GDN	16.4	13.6	42.8	71.8	54.7	57.3	70.5	37.5	39.3	59.9	54.2
MAMBA3-*SISO*	16.1	13.7	45.2	**72.8**	57.2	55.6	72.3	36.3	40.6	57.0	54.6
MAMBA3-*MIMO*	15.9	**12.3**	**47.3**	**72.8**	**57.9**	55.9	**73.2**	**39.2**	38.6	57.5	**55.3**
FRAC	16.5	13.5	44.9	72.4	56.4	55.6	71.6	38.0	**41.8**	60.0	55.1

LM Harness On short-context language understanding and commonsense reasoning tasks, FRAC achieves performance that is competitive with other linear-time baselines and the Transformer. As shown in Table 3, FRAC is only 0.2% behind the Transformer and MAMBA3-*MIMO* on average, while achieving perplexity comparable to the other models. Since these tasks typically involve short sequences of roughly 32–256 tokens, the results indicate that FRAC preserves short-context language understanding abilities despite its structural bias toward modeling heavy-tailed long-memory.

Recall-Retrieval A similar trend is observed on real-world recall- and retrieval-intensive tasks [1], where FRAC remains competitive with prior models. It ranks third overall, trailing the Transformer by 2.1% on average and the second best MAMBA3-*MIMO* by only 0.4%. It is worth noting that although the original tasks were designed to be challenging at very long sequences, we follow prior work [10, 35, 69] and evaluate on truncated 2K-token sequences, making them primarily short-context retrieval tasks in this setting. Nevertheless, FRAC retains strong short-context retrieval ability despite being structurally designed for long-context modeling.

Table 4: Model performance on 6 real-world recall-retrieval tasks. The highest and second-highest scores are highlighted in bold and underline, respectively.

Model	SWDE	SQD	FDA	TQA	NQ	Drop	Avg.
Transformer	42.8	39.6	27.4	**65.7**	**27.3**	**28.3**	**38.5**
MAMBA2	32.8	31.0	13.6	60.5	20.4	20.8	29.9
GDN	32.7	35.1	12.2	58.4	22.1	24.6	30.9
MAMBA3-*SISO*	32.2	39.8	26.7	61.5	24.8	25.7	35.1
MAMBA3-*MIMO*	34.5	**41.2**	**28.9**	64.2	25.8	26.1	36.8
FRAC	**43.2**	39.0	20.0	64.0	25.9	26.2	36.4

4.3 DNA modeling

We evaluate FRAC on genomic sequence modeling. Following HyenaDNA [49], we train 7M-parameter causal language models on the human genome HG38 dataset across sequence lengths ranging from 1K to 64K, testing the scaling law. Figure 4 shows that FRAC achieves on par or lower perplexity than MAMBA3 and GDN across all lengths, with larger gains at longer contexts. Full details are presented in Appendix E.4.

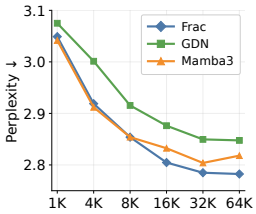


Figure 4: DNA perplexity vs. training sequence length.

5 Related Work

State-space sequence models. State-space models provide a principled route to efficient sequence modeling by viewing sequence layers as discretizations of continuous-time dynamical systems. Structured SSMs such as S4 [22] showed that parameterized linear ODEs can yield long-range sequence models with efficient convolutional or recurrent implementations, with subsequent work simplifying or extending this view through diagonal parameterizations [21] and input-dependent dynamics [24]. Our work follows this continuous-time perspective, but changes the underlying memory law: rather than starting from an ODE with exponential memory, FRAC starts from fractional dynamics and realizes the heavy-tail memory through a finite sum-of-exponentials approximation.

Selective SSMs and Mamba. Recent SSM language models improve discrete sequence modeling through input selectivity. Mamba [20] makes the transition and input/output projections token-dependent. MAMBA2 [10] develops the state-space duality view, connecting selective SSMs to semiseparable attention-like operators and enabling faster hardware-efficient implementations. Mamba-3 [35] further improves this line through a more expressive discretization, complex-valued state dynamics, and a multi-input multi-output (MIMO) recurrence. FRAC explores a different axis: it retains the efficient selective recurrent structure but changes the memory kernel being discretized. In particular, the multiple modes in FRAC should not be confused with Mamba-3's MIMO design. The modes in FRAC arise as quadrature modes over timescales in a finite sum-of-exponentials approximation to a fractional kernel, rather than as a MIMO parameterization of the recurrent map.

Linear attention, associative updates, and multiple memories. Several efficient sequence models approach bounded-state recurrence from the linear-attention side. RetNet [58] uses fixed decay factors across retention heads to cover different memory timescales, with each head using a single exponential decay. Gated Linear Attention [67] uses token-dependent forgetting, while delta-rule models [57, 68] improve associative memory updates. Gated DeltaNet [69] combines gating with delta-rule updates to improve retrieval, length extrapolation, and long-context understanding. Kimi Delta Attention [59] further develops this direction with finer-grained gating in a hybrid architecture. Mixture-of-Memories [13] instead increases effective memory capacity by routing tokens across multiple independent memory states. These approaches improve different aspects of bounded-state memory, including retention, updating, and capacity. FRAC instead derives the temporal decay law by introducing a heavy-tailed fractional memory kernel, making it largely complementary to the update- and capacity-oriented mechanisms mentioned above.

Fractional dynamics in machine learning. Fractional theory has found meaningful use in several areas of machine learning. Recent works have explored neural fractional differential equation models [9] and scalable training methods for them [32]. The work most closely related to ours is FADE [31], a fractional-attention differential-equation framework that solves a history-dependent neural integral equation over sampled past states using an iterative solver. Unlike FRAC, FADE does not provide a fixed-size recurrent state or a decoder language-model architecture. Fractional-order spiking neural networks apply the fractional formalism to enrich artificial neuron dynamics [17]. Fractional stochastic dynamics has been used in generative modeling, including fractional diffusion [50] and a protein-generation extension [37]. Our work instead approximates the fractional memory kernel with a finite SoE bank, giving a fixed-size recurrent state compatible with efficient scan computation and autoregressive decoding.

6 Conclusion

We introduced FRAC, a selective SSM architecture that turns fractional long-memory dynamics into an efficient finite-state recurrent layer. Experiments show that FRAC improves long-context performance over strong SSM baselines while remaining competitive on short-context benchmarks. Future work could combine fractional long-memory transitions with delta-rule style update mechanisms to obtain both stronger long-range inductive bias and more precise associative retrieval.

Acknowledgements

This work was conducted by the authors as part of their employment at *Huawei Technologies Canada Co., Ltd.* using internal company resources. The work received no external or third-party funding or support. The authors declare no competing interests.

References

[1] Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. Just read twice: closing the recall gap for recurrent language models, 2024. URL https://arxiv.org/abs/2407.05483.

[2] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172.

[3] Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Smollm-corpus, 2024. URL https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus.

[4] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI conference on artificial intelligence*, 34(05):7432–7439, 2020.

[5] Guy E. Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, 1990.

[6] Renu Chaudhary, Kai Diethelm, Afshin Farhadi, and Fred A. Fuchs. An efficient exponential sum approximation of power-law kernels for solving fractional differential equation, 2025. URL https://arxiv.org/abs/2508.20311.

[7] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[9] C. Coelho, M. Fernanda P. Costa, and L. L. Ferrás. Neural fractional differential equations. *ArXiv*, abs/2403.02737, 2024.

[10] Tri Dao and Albert Gu. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, pages 10041–10071, 2024.

[11] Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365.

[12] Kai Diethelm. *The Analysis of Fractional Differential Equations: An Application-Oriented Exposition Using Differential Operators of Caputo Type*. Springer, 2010.

[13] Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Tao Zhang, and Yu Cheng. Mom: Linear sequence modeling with mixture-of-memories. In *The Fourteenth International Conference on Learning Representations*, 2026.

[14] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246.

[15] Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1102. URL https://aclanthology.org/P19-1102.

[16] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

[17] Chengjie Ge, Yufeng Peng, Zihao Li, Qiyu Kang, Xueyang Fu, Xuhao Li, Qixin Zhang, Ren Junhao, and Zheng-Jun Zha. Fractional-order spiking neural network. In *The Fourteenth International Conference on Learning Representations*, 2026.

[18] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In Lu Wang, Jackie Chi Kit Cheung, Giuseppe Carenini, and Fei Liu, editors, *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5409. URL https://aclanthology.org/D19-5409.

[19] Rudolf Gorenflo, Anatoly A. Kilbas, Francesco Mainardi, and Sergei V. Rogosin. *Mittag-Leffler Functions, Related Topics and Applications*. Springer Publishing Company, Inc., 2014.

[20] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

[21] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, volume 35, pages 35971–35983, 2022.

[22] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*, 2022.

[23] Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian J. McAuley. Longcoder: A long-range pre-trained language model for code completion. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 12098–12107. PMLR, 2023. URL https://proceedings.mlr.press/v202/guo23j.html.

[24] Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid structural state-space models. In *International Conference on Learning Representations (ICLR)*, 2023.

[25] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580.

[26] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? In *Proceedings of the First Conference on Language Modeling (COLM)*, COLM, Philadelphia, PA, USA, October 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

[27] Luyang Huang, Shuyang Cao, Nikolaus Nova Parulian, Heng Ji, and Lu Wang. Efficient Attentions for Long Document Summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1419–1436. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.112. URL https://doi.org/10.18653/v1/2021.naacl-main.112.

[28] Shidong Jiang, Jiwei Zhang, Qian Zhang, and Zhimin Zhang. Fast evaluation of the Caputo fractional derivative and its applications to fractional diffusion equations. *Communications in Computational Physics*, 21(3):650–678, 2017.

[29] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

[30] Gregory Kamradt. Needle In A Haystack - pressure testing LLMs. *Github*, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main.

[31] Qiyu Kang, Wenjun Cui, Xuhao Li, Yuxin Ma, Xueyang Fu, Wee Peng Tay, Yidong Li, and Zhengjun Zha. Neural fractional attention differential equations. In *Advances in Neural Information Processing Systems*, 2025.

[32] Qiyu Kang, Xuhao Li, Kai Zhao, Wenjun Cui, Yanan Zhao, Weihua Deng, and Wee Peng Tay. Efficient training of neural fractional-order differential equation via adjoint backpropagation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39:17750–17759, 04 2025.

[33] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023.

[34] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion

Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026.

[35] Aakash Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, J Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *The Fourteenth International Conference on Learning Representations*, 2026.

[36] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. URL https://aclanthology.org/C02-1150.

[37] Xiao Liang, Wentao Ma, Eric Paquet, Herna Lydia Viktor, and Wojtek Michalowski. Prot-gfdm: A generative fractional diffusion model for protein generation. *ArXiv*, abs/2504.21092, 2025.

[38] Tianyang Liu, Canwen Xu, and Julian McAuley. RepoBench: Benchmarking repository-level code auto-completion systems. *ArXiv preprint*, abs/2306.03091, 2023. URL https://arxiv.org/abs/2306.03091.

[39] Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. OpenCeres: When open information extraction meets the semi-structured web. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3047–3056, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1309. URL https://aclanthology.org/N19-1309.

[40] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[41] Peng Lu, Jerry Huang, Qiuhao Zeng, Xinyu Wang, Boxing Chen, Philippe Langlais, and Yufei Cui. Mamba modulation: On the length generalization of mamba models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=QEU047bE8p.

[42] Yurii Luchko and Rudolf Gorenflo. An operational method for solving fractional differential equations with the caputo derivatives. *Acta Mathematica Vietnamica*, 24(2):207–233, 1999.

[43] Francesco Mainardi. *Fractional Calculus and Waves in Linear Viscoelasticity: An Introduction to Mathematical Models*. Imperial College Press, London, 2010.

[44] Francesco Mainardi and Rudolf Gorenflo. Time-fractional derivatives in relaxation processes: A tutorial survey. *Fractional Calculus and Applied Analysis*, 10(3):269–308, 2007.

[45] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations (ICLR 2017)*, Toulon, France, 2017. OpenReview.net. URL https://openreview.net/forum?id=Byj72udxe.

[46] Ralf Metzler and Joseph Klafter. The restaurant at the end of the random walk: recent developments in the description of anomalous transport by fractional dynamics. *Journal of Physics A: Mathematical and General*, 37(31):R161–208, 2004.

[47] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018.

[48] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, 2018.

[49] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, Stefano Ermon, Christopher Ré, and Stephen Baccus. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. In A. Oh, T. Naumann,

A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 43177–43201. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/86ab6927ee4ae9bde4247793c46797c7-Paper-Conference.pdf.

[50] Gabriel Nobis, Maximilian Springenberg, Marco Aversa, Michael Detzel, Rembert Daems, Roderick Murray-Smith, Shinichi Nakajima, Sebastian Lapuschkin, Stefano Ermon, Tolga Birdal, Manfred Opper, Christoph Knochenhauer, Luis Oala, and Wojciech Samek. Generative fractional diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[51] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144.

[52] Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

[53] Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Re, Ce Zhang, and Stefano Massaroli. Mechanistic design and scaling of hybrid architectures. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 40908–40950. PMLR, 21–27 Jul 2024.

[54] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive Transformers for Long-Range Sequence Modelling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

[55] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124.

[56] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[57] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, 2021.

[58] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2023. URL https://arxiv.org/abs/2307.08621.

[59] Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

[60] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/ARXIV.2307.09288. URL `https://doi.org/10.48550/arXiv.2307.09288`.

[61] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL `https://aclanthology.org/2022.tacl-1.31`.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[63] Shida Wang and Beichen Xue. State-space models with layer-wise nonlinearity are universal approximators with exponential decaying memory. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 74021–74038. Curran Associates, Inc., 2023.

[64] Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[65] Eric Wu, Kevin Wu, Roxana Daneshjou, David Ouyang, Daniel E. Ho, and James Zou. How medical AI devices are evaluated: limitations and recommendations from an analysis of FDA approvals. *Nature Medicine*, 27(4):582–584, 2021. ISSN 1546-170X. doi: 10.1038/s41591-021-01312-x. URL `https://doi.org/10.1038/s41591-021-01312-x`.

[66] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

[67] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Proceedings of the 41st International Conference on Machine Learning*, ICML, 2024.

[68] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS, 2024.

[69] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025.

[70] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question

answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259.

[71] Zhifan Ye, Kejing Xia, Yonggan Fu, Xin Dong, Jihoon Hong, Xiangchi Yuan, Shizhe Diao, Jan Kautz, Pavlo Molchanov, and Yingyan Celine Lin. Longmamba: Enhancing mamba's long-context capabilities via training-free receptive field enlargement. In *The Thirteenth International Conference on Learning Representations*, 2025.

[72] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[73] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proceedings of the VLDB Endowment*, 16(12):3848–3860, 2023.

[74] Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. QMSum: A new benchmark for query-based multi-domain meeting summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.472. URL https://aclanthology.org/2021.naacl-main.472.

A Discussion of Theorem 1

The statement of this theorem is a standard, yet non-trivial, result in complex analysis. The exact reference can be found in the comprehensive book of [43, Appendix E]. Specifically, the construction of densities in diffusive representation is given there. First, for $0 < \alpha < 1$ and $\lambda > 0$:

$$E_\alpha(-\lambda t^\alpha) = \int_0^\infty e^{-rt}\, K_\alpha(r;\lambda)\, dr, \qquad t \geq 0,$$

where

$$K_\alpha(r;\lambda) = \frac{1}{\pi}\, \frac{\lambda\, r^{\alpha-1} \sin(\pi\alpha)}{r^{2\alpha} + 2\lambda r^\alpha \cos(\pi\alpha) + \lambda^2}, \qquad r > 0.$$

Also for $0 < \alpha \leq \beta < 1$ and $\lambda > 0$:

$$t^{\beta-1} E_{\alpha,\beta}(-\lambda t^\alpha) = \int_0^\infty e^{-rt}\, K_{\alpha,\beta}(r;\lambda)\, dr, \qquad t > 0,$$

where

$$K_{\alpha,\beta}(r;\lambda) = \frac{1}{\pi}\, \frac{\lambda \sin\big(\pi(\beta-\alpha)\big) + r^\alpha \sin(\pi\beta)}{r^{2\alpha} + 2\lambda r^\alpha \cos(\pi\alpha) + \lambda^2}\, r^{\alpha-\beta}, \qquad r > 0.$$

In both cases, the spectral density is nonnegative.

To get the formulas (9), we need to change the variables $\tau = 1/r$. Then

$$E_\alpha(-\lambda t^\alpha) = \int_0^\infty e^{-t/\tau}\, R_{\alpha,\lambda}(\tau)\, d\tau,$$

where

$$R_{\alpha,\lambda}(\tau) = \frac{1}{\pi}\, \frac{\lambda \tau^{\alpha-1} \sin(\pi\alpha)}{\lambda^2\tau^{2\alpha} + 2\lambda\tau^\alpha \cos(\pi\alpha) + 1}, \qquad \tau > 0. \tag{25}$$

Similarly, putting $\beta = \alpha$:

$$g_{\alpha,\lambda}(t) := t^{\alpha-1} E_{\alpha,\alpha}(-\lambda t^\alpha) = \int_0^\infty e^{-t/\tau}\, H_{\alpha,\lambda}(\tau)\, d\tau,$$

where

$$H_{\alpha,\lambda}(\tau) = \frac{1}{\pi}\, \frac{\tau^{\alpha-2} \sin(\pi\alpha)}{1 + 2\lambda\tau^\alpha \cos(\pi\alpha) + \lambda^2\tau^{2\alpha}}, \qquad \tau > 0. \tag{26}$$

Remark 3. From the formulas it is clear that the two diffusive densities are related by the exact identity

$$H_{\alpha,\lambda}(\tau) = \frac{1}{\lambda\tau}\, R_{\alpha,\lambda}(\tau), \qquad \tau > 0. \tag{27}$$

Remark 4. It is important for the next steps to note the self-similarity relations in the λ-scaling. In particular, it follows from the explicit formulas for the densities that

$$R_{\alpha,\lambda}(\tau) = \lambda^{1/\alpha}\, R_{\alpha,1}\big(\lambda^{1/\alpha}\tau\big),$$
$$H_{\alpha,\lambda}(\tau) = \lambda^{2/\alpha-1}\, H_{\alpha,1}\big(\lambda^{1/\alpha}\tau\big),$$

so varying λ rescales the underlying timescale axis by the factor $\lambda^{-1/\alpha}$.

B Proof of Theorem 2

Proof. We will use the diffusive representations (9) proved in Appendix A:

$$E_\alpha(-\lambda t^\alpha) = \int_0^\infty e^{-t/\tau}\, R_{\alpha,\lambda}(\tau)\, d\tau, \qquad g_{\alpha,\lambda}(t) = \int_0^\infty e^{-t/\tau}\, H_{\alpha,\lambda}(\tau)\, d\tau,$$

where the densities are given in (25) and (26).

Both densities are strictly positive on $(0,\infty)$. Setting $t=0$ for $E_\alpha(-\lambda t^\alpha)$ in (9), we get:

$$\int_0^\infty R_{\alpha,\lambda}(\tau)\,d\tau = E_\alpha(0) = 1.$$

The second equality is easily seen from the power series definition of the function E_α (4). In particular, $R_{\alpha,\lambda} \in L^1(0,\infty)$.

Fix $\varepsilon > 0$, and set $\varepsilon_0 := \frac{\varepsilon}{2}\min(1,\lambda)$.

As we have shown, $R_{\alpha,\lambda} \in L^1(0,\infty)$, so we may choose $0 < \tau_- < \tau_+ < \infty$ such that

$$\int_{(0,\tau_-)\cup(\tau_+,\infty)} R_{\alpha,\lambda}(\tau)\,d\tau < \varepsilon_0.$$

For $E_\alpha(-\lambda t^\alpha)$ kernel, using the inequality $e^{-t/\tau} \le 1$:

$$\sup_{t\in[0,T]} \int_{(0,\tau_-)\cup(\tau_+,\infty)} e^{-t/\tau} R_{\alpha,\lambda}(\tau)\,d\tau \le \int_{(0,\tau_-)\cup(\tau_+,\infty)} R_{\alpha,\lambda}(\tau)\,d\tau < \varepsilon_0 \le \frac{\varepsilon}{2}. \tag{28}$$

For $g_{\alpha,\lambda}(t)$ kernel, Fubini's theorem and the identity (27) yield

$$\begin{aligned}
\int_0^T \int_{(0,\tau_-)\cup(\tau_+,\infty)} e^{-t/\tau} H_{\alpha,\lambda}(\tau)\,d\tau\,dt &= \int_{(0,\tau_-)\cup(\tau_+,\infty)} \Big(\int_0^T e^{-t/\tau}\,dt\Big) H_{\alpha,\lambda}(\tau)\,d\tau \\
&= \frac{1}{\lambda}\int_{(0,\tau_-)\cup(\tau_+,\infty)} \big(1 - e^{-T/\tau}\big) R_{\alpha,\lambda}(\tau)\,d\tau \\
&\le \frac{1}{\lambda}\int_{(0,\tau_-)\cup(\tau_+,\infty)} R_{\alpha,\lambda}(\tau)\,d\tau < \frac{\varepsilon_0}{\lambda} \le \frac{\varepsilon}{2}.
\end{aligned} \tag{29}$$

Now let

$$A = \log \tau_-, \qquad B = \log \tau_+.$$

For a sufficiently large $M \in \mathbb{N}$ (to be justified below), let

$$h = \frac{B-A}{M}, \qquad x_m = A + \Big(m - \tfrac{1}{2}\Big)h, \qquad m = 1,\dots,M.$$

Define

$$\tau_m := e^{x_m} = e^{A+h/2} e^{(m-1)h} = \tau_0 q^{m-1},$$

where

$$\tau_0 := e^{A+h/2} > 0, \qquad q := e^h > 1.$$

Thus $\{\tau_m\}_{m=1}^M$ is a geometrically spaced bank. With the change of variables $\tau = e^x$, $d\tau = e^x\,dx$, the truncated integrals in (9) become

$$\int_{\tau_-}^{\tau_+} e^{-t/\tau} R_{\alpha,\lambda}(\tau)\,d\tau = \int_A^B F(t,x)\,dx, \qquad \int_{\tau_-}^{\tau_+} e^{-t/\tau} H_{\alpha,\lambda}(\tau)\,d\tau = \int_A^B G(t,x)\,dx,$$

where

$$F(t,x) := e^{-te^{-x}} e^x R_{\alpha,\lambda}(e^x), \qquad G(t,x) := e^{-te^{-x}} e^x H_{\alpha,\lambda}(e^x).$$

The functions F and G are continuous on the compact rectangle $[0,T]\times[A,B]$, hence uniformly continuous there (by Heine–Cantor Theorem).

Let

$$\omega_F(\delta) := \sup\big\{|F(t,x) - F(t,y)| :\ t\in[0,T],\ x,y\in[A,B],\ |x-y|\le\delta\big\},$$

and define ω_G analogously. Then $\omega_F(\delta)\to 0$ and $\omega_G(\delta)\to 0$ as $\delta\to 0^+$. The midpoint-rule estimate therefore gives

$$\sup_{t\in[0,T]} \left| \int_A^B F(t,x)\,dx - h\sum_{m=1}^M F(t,x_m) \right| \le (B-A)\,\omega_F(h/2), \tag{30}$$

$$\sup_{t\in[0,T]} \left| \int_A^B G(t,x)\,dx - h\sum_{m=1}^M G(t,x_m) \right| \le (B-A)\,\omega_G(h/2). \tag{31}$$

Choose M large enough so that

$$(B-A)\,\omega_F(h/2) \le \frac{\varepsilon}{2}, \qquad (B-A)\,\omega_G(h/2) \le \frac{\varepsilon}{2T}.$$

By the definitions of F and G, the midpoint approximations in (30) and (31) take the form:

$$h\sum_{m=1}^{M} F(t,x_m) = h\sum_{m=1}^{M} e^{-te^{-x_m}}\, e^{x_m} R_{\alpha,\lambda}(e^{x_m}) = h\sum_{m=1}^{M} e^{-t/\tau_m}\,\tau_m R_{\alpha,\lambda}(\tau_m)$$

$$h\sum_{m=1}^{M} G(t,x_m) = h\sum_{m=1}^{M} e^{-te^{-x_m}}\, e^{x_m} H_{\alpha,\lambda}(e^{x_m}) = h\sum_{m=1}^{M} e^{-t/\tau_m}\,\tau_m H_{\alpha,\lambda}(\tau_m).$$

This motivates the definitions for the coefficients:

$$c_m(\alpha,\lambda) := h\,\tau_m\,R_{\alpha,\lambda}(\tau_m), \qquad d_m(\alpha,\lambda) := h\,\tau_m\,H_{\alpha,\lambda}(\tau_m), \qquad m=1,\dots,M. \tag{32}$$

Since $R_{\alpha,\lambda}(\tau)$ and $H_{\alpha,\lambda}(\tau)$ are positive for all $\tau > 0$, we have $c_m(\alpha,\lambda) > 0$ and $d_m(\alpha,\lambda) > 0$. Moreover,

$$\frac{d_m(\alpha,\lambda)}{c_m(\alpha,\lambda)} = \frac{H_{\alpha,\lambda}(\tau_m)}{R_{\alpha,\lambda}(\tau_m)} = \frac{1}{\lambda\tau_m}, \qquad m=1,\dots,M.$$

Combining (28) and (30), we obtain

$$\sup_{t\in[0,T]} \left| E_\alpha(-\lambda t^\alpha) - \sum_{m=1}^{M} c_m(\alpha,\lambda) e^{-t/\tau_m} \right| \le \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.$$

Likewise, by (29), (31), and the bound

$$\int_0^T |f(t)|\,dt \le T \sup_{t\in[0,T]} |f(t)|,$$

we get

$$\int_0^T \left| g_{\alpha,\lambda}(t) - \sum_{m=1}^{M} d_m(\alpha,\lambda) e^{-t/\tau_m} \right| dt \le \frac{\varepsilon}{2} + T\cdot\frac{\varepsilon}{2T} = \varepsilon.$$

Finally, from the explicit formula for $R_{\alpha,\lambda}$,

$$R_{\alpha,\lambda}(\tau) = \frac{\sin(\pi\alpha)}{\pi\lambda}\,\tau^{-\alpha-1}\big(1+O(\tau^{-\alpha})\big), \qquad \tau\to\infty.$$

Since $h = \log q$, it follows from (32) that

$$c_m(\alpha,\lambda) = (\log q)\,\tau_m R_{\alpha,\lambda}(\tau_m) = \frac{(\log q)\sin(\pi\alpha)}{\pi\lambda}\,\tau_m^{-\alpha}\big(1+O(\tau_m^{-\alpha})\big)$$

for the large-timescale part of the geometric bank. This completes the proof. □

Remark 5. We use the uniform norm to approximate $E_\alpha(-\lambda t^\alpha)$, while the approximation of $g_{\alpha,\lambda}(t) = t^{\alpha-1}E_{\alpha,\alpha}(-\lambda t^\alpha)$ is stated in the L^1-norm. This asymmetry is forced by the structure of the kernels themselves: for $0<\alpha<1$, $g_{\alpha,\lambda}$ has an integrable singularity at $t=0^+$, whereas any finite sum of exponentials is bounded there, so no uniform pointwise approximation can hold near zero. However, L^1-bound is sufficient for our main goal, the proof of Proposition 3.

Remark 6. Although Theorem 2 is an existence result, the numerical-analysis literature on sum-of-exponentials approximations for fractional kernels [28, 6] shows that the required number of modes grows only poly-logarithmically with the target accuracy and time horizon. Our kernels $E_\alpha(-\lambda t^\alpha)$ and $g_{\alpha,\lambda}$ exhibit the same multi-timescale power-law structure, so these works support the expectation that only a modest number of modes is needed in our setting as well. In practice, this means that a bank of only a few tens of modes can already cover many orders of magnitude of timescale at useful accuracy. See Appendix G for the practical ablation on the number of modes M.

C Proof of Proposition 3

Proof. For each mode $m = 1, \dots, M$, the ODE

$$\dot{s}_m(t) = -\frac{1}{\tau_m} s_m(t) + a_m(\alpha, \lambda)\, u(t), \qquad s_m(0) = h(0),$$

has the explicit solution

$$s_m(t) = e^{-t/\tau_m} h(0) + a_m(\alpha, \lambda) \int_0^t e^{-(t-\xi)/\tau_m} u(\xi)\, d\xi. \tag{33}$$

Therefore, the readout

$$\tilde{h}_M(t) = \sum_{m=1}^{M} c_m(\alpha, \lambda) s_m(t)$$

takes the form

$$\begin{aligned} \tilde{h}_M(t) = h(0) \sum_{m=1}^{M} c_m(\alpha, \lambda) e^{-t/\tau_m} \\ + \int_0^t \sum_{m=1}^{M} c_m(\alpha, \lambda) a_m(\alpha, \lambda) e^{-(t-\xi)/\tau_m} u(\xi)\, d\xi. \end{aligned} \tag{34}$$

With the choice

$$a_m(\alpha, \lambda) = \frac{d_m(\alpha, \lambda)}{c_m(\alpha, \lambda)}, \qquad m = 1, \dots, M,$$

this becomes

$$\tilde{h}_M(t) = h(0) \sum_{m=1}^{M} c_m(\alpha, \lambda) e^{-t/\tau_m} + \int_0^t \sum_{m=1}^{M} d_m(\alpha, \lambda) e^{-(t-\xi)/\tau_m} u(\xi)\, d\xi. \tag{35}$$

On the other hand, the full solution h for the fractional differential equation (3) is given by (6):

$$h(t) = E_\alpha(-\lambda t^\alpha)\, h(0) + \int_0^t g_{\alpha,\lambda}(t-\xi) u(\xi)\, d\xi.$$

Subtracting (35) from (6), we obtain

$$\begin{aligned} |h(t) - \tilde{h}_M(t)| \le |h(0)| \left| E_\alpha(-\lambda t^\alpha) - \sum_{m=1}^{M} c_m(\alpha, \lambda) e^{-t/\tau_m} \right| \\ + \left| \int_0^t \Big(g_{\alpha,\lambda}(t-\xi) - \sum_{m=1}^{M} d_m(\alpha, \lambda) e^{-(t-\xi)/\tau_m} \Big) u(\xi)\, d\xi \right|. \end{aligned} \tag{36}$$

By (10), the first term is bounded by $|h(0)|\, \varepsilon$. For the second term, using $\|u\|_{L^\infty([0,T])} < \infty$, the change of variables $r = t - \xi$, and then (11), we get

$$\begin{aligned} &\left| \int_0^t \Big(g_{\alpha,\lambda}(t-\xi) - \sum_{m=1}^{M} d_m(\alpha, \lambda) e^{-(t-\xi)/\tau_m} \Big) u(\xi)\, d\xi \right| \\ &\le \|u\|_{L^\infty([0,T])} \int_0^t \left| g_{\alpha,\lambda}(t-\xi) - \sum_{m=1}^{M} d_m(\alpha, \lambda) e^{-(t-\xi)/\tau_m} \right| d\xi \\ &= \|u\|_{L^\infty([0,T])} \int_0^t \left| g_{\alpha,\lambda}(r) - \sum_{m=1}^{M} d_m(\alpha, \lambda) e^{-r/\tau_m} \right| dr \\ &\le \|u\|_{L^\infty([0,T])} \int_0^T \left| g_{\alpha,\lambda}(r) - \sum_{m=1}^{M} d_m(\alpha, \lambda) e^{-r/\tau_m} \right| dr \\ &\le \|u\|_{L^\infty([0,T])}\, \varepsilon. \end{aligned}$$

Substituting these two bounds into (36) and taking the supremum over $t \in [0, T]$, we conclude that

$$\sup_{t\in[0,T]} |h(t) - \tilde{h}_M(t)| \le \varepsilon\Big(|h(0)| + \|u\|_{L^\infty([0,T])}\Big),$$

which is exactly (16). This completes the proof. □

D Frac State Transition Algorithm

For compactness, Algorithm 1 is written in vectorized form over heads H with head dimension d_h. Equivalently, the same update is applied independently to each head with its own controls Δ_t, α_t, and λ_t.

In our implementation, the maps g_{read} and g_{write} are parameterized as gated linear projections of the form

$$g_{\text{read}}(u) = \sigma(p_{\text{read}})\, W_{\text{read}} u, \qquad g_{\text{write}}(u) = \sigma(p_{\text{write}})\, W_{\text{write}} u,$$

where p_{read} and p_{write} are learnable scalar parameters and $W_{\text{read}}, W_{\text{write}}$ are learnable matrices.

Algorithm 1 Frac state transition for one token t (single-step recurrent form)

Require: input $u_t \in \mathbb{R}^{H\times d_h}$, step sizes $\Delta_t \in \mathbb{R}^H_{>0}$, fractional controls $\alpha_t \in (0,1)^H$ and $\lambda_t \in \mathbb{R}^H_{>0}$, previous mode state $S_{t-1} \in \mathbb{R}^{H\times M\times d_h}$, base timescale bank tensor $\tau \in \mathbb{R}^M_{>0}$ content-dependent linear maps $g_{\text{read}}, g_{\text{write}} : \mathbb{R}^{H\times d_h} \to \mathbb{R}^{H\times M}$

▷ Fractional specialization of the ZOH coefficients

1: $\tilde{\tau}_t \leftarrow \tau \oslash \lambda_t^{1/\alpha_t}$
2: $\tilde{\rho}_t \leftarrow \exp(-\Delta_t \oslash \tilde{\tau}_t)$
3: $\tilde{\beta}_t \leftarrow 1 - \tilde{\rho}_t$

▷ Read and write weights over the mode bank

4: $c_t \leftarrow \mathrm{softmax}_m(-\alpha_t \log \tau + g_{\text{read}}(u_t))$
5: $b_t \leftarrow \mathrm{softmax}_m(-\alpha_t \log \tau + g_{\text{write}}(u_t))$
6: $\kappa_t \leftarrow \tilde{\beta}_t \odot b_t$

▷ Selective recurrent update and readout

7: $S_t \leftarrow \tilde{\rho}_t \odot S_{t-1} + \kappa_t \odot u_t$
8: $h_t \leftarrow \sum_{m=1}^M c_{t,:,m} \odot S_{t,:,m,:}$
9: **return** $h_t,\ S_t$

Remark 7. The precise ZOH discretization, as we derive in (20), gives $\tilde{\beta}_t = (1 - \tilde{\rho}_t)/\lambda_t$. In Algorithm 1, however, we use $\tilde{\beta}_t = 1 - \tilde{\rho}_t$ (Step 3). We made this modeling choice to decouple timescale control from the write amplitude, which we found improves training stability. When λ is input-independent, the two forms are equivalent up to a constant rescaling of the state and readout. All our experiments in this paper use $\tilde{\beta}_{t,m} = 1 - \tilde{\rho}_{t,m}$, as in Algorithm 1.

Remark 8. The ZOH derivation in Section 3.2 is exact for a fixed fractional mode bank with frozen values of α, λ, and Δ. The selective FRAC layer applies the same transition parameterization token-wise, with α_t, λ_t, and Δ_t frozen within each interval, while the learned weights $b_{t,m}$ and $c_{t,m}$ provide content-dependent write and read routing over the resulting mode bank. Thus, the fractional theory specifies the per-token memory geometry and ZOH transition form, while selectivity adds content-dependent routing over this fractional transition. Accordingly, the full selective layer can be viewed as an architectural generalization of the fixed-system fractional construction, which serves as a principled inductive bias for heavy-tailed memory.

E Experimental Setting

E.1 Heavy-Tail Synthetic Probing

We use a controlled synthetic task to isolate long-range extrapolation under a heavy-tail aggregation law. Each sequence contains sparse ± 1 events embedded in a background token, with inter-event

gaps drawn from a truncated Zipf distribution. The label is the sign of a fixed power-law weighted sum over preceding events,

$$y = \sum_k \frac{v_k}{(L-k)^\gamma},$$

where the sum runs over all event positions k, each event carries value $v_k \in \{-1, +1\}$, and $\gamma = 0.1$ is chosen so that the target depends on a strongly heavy-tailed accumulation over distance. We train all models on sequences of length 512 and evaluate zero-shot extrapolation at $512, 2$K, 8K, 16K, 32K, 64K, and 128K. All models use a single sequence-mixing layer and, as shown in Table 5, are approximately parameter-matched at about 0.2M parameters. For MAMBA3, we use the stable SISO variant, since the MIMO implementation in TileLang does not support hidden sizes small enough to match the parameter counts of the other models. Table 6 reports mean accuracy and standard deviation across runs for each model at the tested sequence lengths.

Table 5: Model configurations for the heavy-tail synthetic benchmarking.

Model	Hidden	Heads	Head dim	Params
Attention	128	8	16	199.3K
MAMBA2	160	8	40	200.7K
MAMBA3	160	8	40	201.3K
GDN	96	8	32	203.3K
FRAC	176	8	44	195.2K

Table 6: Accuracy and standard deviation on heavy-tail synthetic benchmark across 512-128K sequence lengths on 10 runs. Bold indicates the best score at each sequence length.

Model	512	2K	8K	16K	32K	64K	128K
Attention	$97.6_{\pm 0.8}$	$65.9_{\pm 3.7}$	$50.5_{\pm 3.3}$	$50.1_{\pm 3.0}$	$50.9_{\pm 3.2}$	$48.6_{\pm 1.7}$	$50.0_{\pm 3.7}$
MAMBA2	$82.7_{\pm 2.9}$	$74.0_{\pm 3.6}$	$60.9_{\pm 4.5}$	$59.4_{\pm 5.4}$	$56.0_{\pm 3.5}$	$57.9_{\pm 3.2}$	$51.3_{\pm 4.3}$
MAMBA3	$\mathbf{98.6_{\pm 0.7}}$	$90.5_{\pm 2.7}$	$71.3_{\pm 7.5}$	$67.0_{\pm 5.0}$	$62.6_{\pm 4.3}$	$63.2_{\pm 4.4}$	$55.6_{\pm 3.8}$
GDN	$98.5_{\pm 0.6}$	$\mathbf{92.5_{\pm 2.8}}$	$78.7_{\pm 3.4}$	$73.7_{\pm 3.5}$	$68.4_{\pm 2.0}$	$65.8_{\pm 2.5}$	$58.4_{\pm 3.8}$
FRAC	$96.5_{\pm 2.5}$	$92.3_{\pm 1.1}$	$\mathbf{85.6_{\pm 3.0}}$	$\mathbf{80.4_{\pm 1.4}}$	$\mathbf{75.2_{\pm 0.4}}$	$\mathbf{70.6_{\pm 2.0}}$	$\mathbf{63.8_{\pm 1.9}}$

E.2 MadLab Synthetic Suite

Setup MADLab [53] is a synthetic benchmark suite designed to probe token-level sequence manipulation and recall mechanisms. We consider the six standard tasks. *Compression* (Comp.) tests whether a model can retain and compress information from the input sequence. *In-context recall* (ICR) evaluates associative retrieval of values from keys presented in context. *Noisy in-context recall* (N-ICR) adds distractor tokens to the same basic retrieval problem of ICR. *Fuzzy in-context recall* (F-ICR) makes the association less direct by requiring recall from longer or less trivially matched motifs. *Selective copying* (SC) measures the ability to copy only the relevant marked tokens from a sequence, and *memorization* (Mem.) tests direct token-level memorization of the training distribution.

Model Configurations We report the performance of four linear models: our FRAC, MAMBA2, GDN, and MAMBA3. As in Appendix E.1, we use only the MAMBA3-*SISO* variant, since the MIMO implementation does not support sufficiently small hidden dimensions. All models use the same 4-layer architecture: two sequence-mixing layers, each followed by a SwiGLU channel-mixing layer. All models use fixed hidden size $d = 128$ following the original protocol in the MADLab paper, and causal convolution size $d_{\text{conv}} = 4$. We matched the parameter count to make all models have approximately 0.5M parameters. MAMBA2 uses expansion 2.0, while all other models use expansion 1.5. All models use 8 heads, except for FRAC which uses 6 heads. For FRAC-specific hyperparameters we use the same setting as for language modeling experiments, the justification for which can be found in the next subsection.

Training protocol Models are trained from scratch with AdamW, batch size 128 for 200 epochs, cosine learning-rate schedule, minimum learning rate 10^{-6}, bfloat16 precision, and 1280 test examples. We sweep learning rates $\{1e^{-4}, 5e^{-4}, 1e^{-3}\}$ and weight decay values $\{0, 0.1\}$. For each

task and model, we select the best hyperparameter setting by the mean official MADLab score across five seeds, and report the mean and standard deviation over the five runs for the selected setting. We use the official MADLab baseline configurations for all task settings, including sequence length, vocabulary size, motif size, noise level, copy length, and multi-query flags. We do not report performance for N-ICR and ICR in the main paper as all models saturated on these two tasks.

E.3 Language Modeling

Pretraining Data We leverage the deduplicated FineWeb-Edu (220B tokens total) subsets of the SmolLM-Corpus [3] as pretraining data in all of our experiments. FineWeb-Edu is a deduplicated, high-quality subset of educational web data filtered from the FineWeb-v1 collection [52]. For our experiments, we randomly sample a 100B-token subset from deduplicated FineWeb-Edu, measured after tokenization with the Llama-2 [60] tokenizer of a vocabulary size of 32k tokens.

Model Configurations In our main experiment, all models are matched to roughly the same parameter count of 1.3B. For the baseline linear models, we adopt the default configuration of the 1B-parameter models from their respective publicly available repositories. We use the Qwen3 [66] architecture as the backbone for the Transformer model, as it achieves state-of-the-art performance among open-weight pure self-attention dense models. All models use a hidden size of 2048, an expansion factor for the intermediate state of 2.0[5], 16 heads, while the number of layers is determined by each model's original configuration and adjusted so that the total parameter count is around 1.3B. More precisely, we use 48 layers for MAMBA2, MAMBA3-*SISO*, and FRAC; 44 layers for MAMBA3-*MIMO*; and 22 and 26 layers for GDN and the Transformer, respectively.

For FRAC, we apply min-max clipping of $\Delta \in [10^{-4}, 1.0]$, following Mamba [10, 35], and also clip our $\lambda \in [0.25, 4.0]$ for numerical stability. We set the number of modes to $M = 16$. This choice follows the scale suggested by Remark 6: a bank of a few tens of geometrically spaced modes is expected to cover many orders of magnitude of timescale at useful accuracy. For implementation, we use a power-of-two mode count to improve hardware utilization in our Triton kernels, making $M = 16$ the smallest power-of-two value in this regime. In the ablation study of Appendix G, $M = 8$ degrades performance, while $M = 32$ gives only the same or a small improvement at a higher computational cost. For the timescale grid τ, we use geometrically spaced base timescales between $\tau_{\min} = 1$ and $\tau_{\max} = 2^{17}$. The upper endpoint is chosen to exceed the longest extrapolation length considered in our experiments, $64\text{k} = 2^{16}$, so the slowest base mode remains active over the full evaluation horizon. The lower endpoint gives the bank access to local recurrent memory, while the logarithmic spacing allocates the remaining modes across intermediate scales. It is worth noting that linear baseline models like Mamba and GDN have no explicit maximum-timescale parameter analogous to $\tau_{\max}$, while the Transformer can directly attend to the full context.

Importantly, introducing this finite mode bank does not give FRAC a larger recurrent state than the baseline linear models. To quantify this, we compare the recurrent SSM state tensors used during autoregressive decoding, excluding short-convolution buffers and architecture-specific auxiliary caches. In our configurations, FRAC, MAMBA2, and MAMBA3 use $d_{\text{inner}} = 4096$ and $H = 16$, giving $d_h = d_{\text{inner}}/H = 256$. With $M = 16$ modes, FRAC maintains $S_t^{\text{FRAC}} \in \mathbb{R}^{H \times M \times d_h}$, corresponding to $16 \times 16 \times 256 = 65{,}536$ recurrent scalars per layer. The principal SSM state in MAMBA2 and MAMBA3 has shape $S_t^{\text{Mamba}} \in \mathbb{R}^{H \times d_h \times N}$; with $N = 128$, this corresponds to $16 \times 256 \times 128 = 524{,}288$ scalars per layer. Similarly, GDN maintains a matrix-valued state $S_t^{\text{GDN}} \in \mathbb{R}^{H \times d_k \times d_v}$; with $d_k = 96$ and $d_v = 192$, this gives $16 \times 96 \times 192 = 294{,}912$ scalars per layer. Thus, Mamba2&3 and GDN maintain respectively $8\times$ and $4.5\times$ as many recurrent-state scalars as FRAC. We also note that recurrent-state size is different from trainable parameter count, and each architecture has to allocate weights to other parts of the token mixer. For example, FRAC includes the projections producing Δ_t, α_t, and λ_t, together with its read/write projections, whereas GDN allocates parameters to its query, key, value, gating, and output projections.

Implementation Details Each model is pretrained on 2 compute nodes, each equipped with 8 *modern parallel compute cards* with 80GB of memory per card. We use the Hugging Face Transformers library [64] as our pretraining framework. For baselines, we use the official Mamba2&3[6]

[5] Except for GDN, where we set it to 3.0 as in their default configuration.

[6] `https://github.com/state-spaces/mamba`

and GDN[7] implementations, and the Qwen3 implementation in the Transformers library for the Transformer baseline. For all models, we use the AdamW [40] optimizer with a learning rate decay setting the initial learning rate to 6e-4 with 10% warm-up steps, with a global batch size of 1M tokens. We use a weight decay of 0.1 for all models, following prior work. The selection of the learning rate was based on preliminary training runs, where it produced stable loss curves across all models. A higher learning rate, such as 1e-3, led to early training divergence, whereas learning rates substantially below 6e-4 produced worse loss curves for most models. To accelerate pretraining, we use Fully Sharded Data Parallel (FSDP) [73] and mixed-precision training [47]. Pretraining each model was completed in roughly 4 to 5 days (see Table 8).

Evaluation Protocol We mostly follow the evaluation protocol from Mamba2&3 and GDN by conducting comprehensive evaluations of models we pretrain from scratch on 4 benchmark suites[8]:

- **LM Harness** which includes perplexity evaluation on Wikitext (Wiki.) [45] and LAMBADA (LMB.) [51], as well as several commonsense reasoning tasks: PIQA [4], HellaSwag (Hella) [72], WinoGrande (Wino) [56], ARC-Easy (ARC-e) and ARC-Challenge (ARC-c) [8], OpenBookQA (OBQA) [48], and BoolQ [7].
- **NIAH** From the RULER [26] benchmark, we include three single *needle-in-a-haystack* tasks: *S-NIAH-1* (pass-key retrieval from repetitive filler text), *S-NIAH-2* (retrieving a target number embedded in natural-text distractors), and *S-NIAH-3* (retrieving a target UUID embedded in natural-text distractors).
- **LongBench** We consider 14 tasks from the LongBench (v1) benchmark [2] and follow the authors' categorizations: Single-document QA (NarrativeQA [33], MultiFieldQA-en, Qasper [11]), Multi-document QA (HotpotQA [70], 2WikiMQA [25], MuSiQue [61]), Summarization (GovReport [27], MultiNews [15], QMSum [74]), Few-shot learning (TREC [36], TriviaQA [29], SAMSum [18]), and Code Completion (LCC [23], RepoBench-P [38]).
- **Real-World Recall-Retrieval** We measure performance on real-world retrieval tasks, including: SWDE [39], SQuAD (SQD) [55], FDA [65], TriviaQA (TQA) [29], NQ [34], and Drop [14]. Following prior work [10, 35, 69], we evaluate using the cloze-completion prompt format of [1], with all sequences truncated up to 2k tokens.

Table 7: Characteristics of the LongBench tasks, including acronym, number of examples, and evaluation metric. The last column reports the mean and standard deviation of example lengths, computed using the Llama-2 tokenizer used in our experiments. Eight out of 14 tasks have an average sequence length less than 8K.

Task	Acr.	# Ex.	Metric	Length
NarrativeQA	NQA	200	F1	18405 ± 8976
Qasper	QQA	200	F1	3619 ± 1880
MultiFieldQA-en	MFQ	150	F1	4559 ± 2473
HotpotQA	HQA	200	F1	9149 ± 2849
2WikiMQA	2WM	200	F1	4885 ± 2541
MuSiQue	Mus	200	F1	11018 ± 1560
GovReport	GvR	200	ROUGE	8169 ± 5239
QMSum	QMS	200	ROUGE	10546 ± 5043
MultiNews	MNs	200	ROUGE	2114 ± 1554
TREC	TRC	200	Acc.	5176 ± 2140
TriviaQA	TQA	200	F1	8209 ± 4029
SAMSum	SSM	200	ROUGE	6258 ± 3107
LCC	LCC	500	Edit Sim.	1235 ± 1086
RepoBench-P	RBP	500	Edit Sim.	4206 ± 2698

[7] `https://github.com/fla-org/flash-linear-attention/tree/main`

[8] Except for Real-World Recall-Retrieval tasks, all benchmarks are implemented through the LM Eval Harness library [16].

E.4 DNA Modeling

We follow the DNA language-modeling setup of HyenaDNA [49][9] and Mamba [20]. We use the HG38 human reference genome data distributed through the HyenaDNA preprocessing pipeline, and train models with a character-level DNA tokenizer under the standard autoregressive next-token prediction objective. Given the training set, we truncate each example to a fixed sequence length. After training, we evaluate perplexity on the test set using the same truncation length as during training. We conduct this experiment for the following lengths $\{1K, 4K, 8K, 16K, 32K, 64K\}$, with the resulting test perplexities shown in Figure 4, comparing FRAC against GDN and MAMBA3-*SISO*. All models use the same 8-layer decoder-only language-model core with hidden size 256, convolution size 4 and eight heads. We approximately match model size to 7M parameters by adjusting the model-specific expansion. More precisely, we use expansion factors of 1.75, 2.0, and 1.5 for MAMBA3, GDN, and FRAC, respectively. Otherwise, we use the same model-specific hyperparameters as in the language-modeling experiments in Appendix E.3. We sweep learning rates over the grid $\{1e{-}4, 5e{-}4, 1e{-}3, 2e{-}3, 4e{-}3, 8e{-}3, 1e{-}2, 2e{-}2\}$ and select the best on the validation set. We use AdamW with weight decay 0.1, and the same cosine learning-rate schedule.

F FracMixer Efficient Implementation

In our implementation, the dominant components of FRACMIXER are implemented using custom Triton kernels. The inputs to the FRAC STATE TRANSITION block are first computed with standard matrix multiplications, followed by a fused Triton kernel that constructs the scan parameters, including the read/write mode weights and the Δ, α, and λ factors. FRAC STATE TRANSITION scan is computed by a sequence of chunk-local Triton kernels for cumulative decays, within-chunk state construction, cross-chunk state passing, chunk-local read/write mixing, and the final scan output. In addition, the output skip connection is fused in that kernel. Although the fractional memory bank introduces multiple timescale modes, the computation is still organized around efficient chunked tensor operations. Compared with MAMBA3-*MIMO*, FracMixer uses a simpler real-valued diagonal-mode update and does not employ complex rotations, trapezoidal two-tap state-inputs, or MIMO rank projections. However, MAMBA3-*MIMO* uses a lower-level CuTe/TileLang implementation, which is more optimized than our Triton, and a smaller rank $R=4$ compared to our $M=16$ modes.

Table 8: Inference throughput in tokens/s (higher is better) for prefill at 1K, 4K, and 16K context lengths, and single-step decoding throughput. All prefill measurements use batch size 1. *Decode throughput for the Transformer is reported at 16K, while the corresponding values at 1K and 4K are 75 and 72 tokens/s, respectively. The last column reports training throughput (also in tokens/s) following the experimental setup described in Appendix E.3.

Model	**Prefill**@1K	**Prefill**@4K	**Prefill**@16K	**Decode**	**Train**@4K
Transformer	**55,839**	**79,173**	56,297	**62***	218,184
MAMBA2	15,362	41,060	67,571	<u>34</u>	**278,620**
GDN	10,455	35,481	58,384	20	215,357
MAMBA3-*SISO*	<u>19,517</u>	44,998	<u>74,314</u>	27	238,924
MAMBA3-*MIMO*	13,692	30,710	46,707	25	200,994
FRAC	16,213	<u>49,946</u>	**77,490**	33	<u>241,598</u>

Table 8 reports the inference prefill throughput (tokens/s) of the 1.3B models of Appendix E.3.[10] We also report autoregressive decoding throughput, averaged over the generation of 64 tokens. It is worth noting that higher prefill throughput at longer contexts reflects improved GPU utilization and better amortization of fixed overheads. For instance, a single 16K-token sequence contains more tokens to process than a 4K-token sequence.

We observe that the self-attention Transformer achieves the highest throughput on short sequences (1–4K) as well as on single token decoding, mainly due to the highly optimized FlashAttention kernel. Among linear models, FRAC and MAMBA3-*SISO* are the closest to the Transformer in the

[9] `https://github.com/HazyResearch/hyena-dna`

[10] Throughput is measured on a single card of the same *modern compute accelerator* used for training, at sequence lengths of $\{1K, 4K, 16K\}$ with batch size 1. All models are evaluated under identical conditions.

short-context regime. However, at 16K, the quadratic complexity of self-attention becomes more pronounced, causing the Transformer to lag behind linear models. In this long-context regime, FRAC achieves the highest throughput, in the same range as MAMBA3-*SISO* and higher than MAMBA3-*MIMO*. In the training setting of Appendix E.3, throughput depends on both the forward and backward implementations. We observe that all models fall within a similar throughput range, with MAMBA2 being the most efficient and slightly outperforming our FRAC.

It is worth noting that the diagonal state transition can introduce a numerical issue analogous to the parallel formulation of GLA [67]: expressing the decay between positions t and k as a ratio of cumulative exponentials can be numerically unstable. We therefore compute the intra-chunk mixing coefficient directly using cumulative log-decay differences:

$$\Gamma_{t,k} = \sum_{m=1}^{M} c_{t,m}\kappa_{k,m} \exp(\ell_{t,m} - \ell_{k,m}), \qquad t \geq k,$$

where $\ell_{t,m} := \sum_{j=t_{\text{start}}}^{t} \log \tilde{\rho}_{j,m}$ denotes the cumulative log-decay of mode m from the beginning of the current chunk to position t. This formulation computes the decay between positions k and t directly in log space and is analogous to the stable formulation used in GLA [67]. GLA uses a second level of chunking to retain tensor-core acceleration for its relatively large reduction dimension ($d_k \geq 64$). In FRAC, the corresponding reduction is over only $M = 16$ modes, making it inexpensive to compute directly within a fused Triton kernel without tensor cores. We therefore use a single level of chunking, with parallel intra-chunk computation and chunk-level state passing, without requiring the additional sub-chunking used by GLA.

G FRAC Ablation

We conduct a set of ablation experiments on the FRACMixer block to validate our architectural design choices. To this end, we define a language modeling setup in which FRAC models with 390M parameters are pretrained from scratch on a 30B-token (4K packed sequences) subset of the corpus used in the experiments in Appendix E.3. Specifically, we use a model with hidden size 1024, intermediate expansion factor 2.0, and 36 layers. Otherwise, all remaining configuration details follow those of the FRAC 1.3B model in Appendix E.3. We validate these ablations by measuring long-document language modeling perplexity on several datasets with 16K-token sequences[11]: ProofPile[12], PG19 [54], and GovReport [27]. This evaluation protocol is better suited for small-scale models and pretraining experimental settings [41, 71].

First, we observe that making α non-learnable ($\alpha = 1$) or removing the prior term for write[13] (*w/o* write prior) causes a drastic drop in language-modeling performance, validating our design choices. In addition, dropping the direct feedthrough term D (*w/o* D) leads to worse performance, aligning with prior observations that D is an important component in modern SSM mixers [20, 10, 35, 69]. Second, we observe that reducing the number of modes to $M = 8$ consistently degrades performance, whereas increasing it to $M = 32$ yields performance comparable to the baseline at the cost of 5% more parameters and 7% slower computation. We therefore use $M = 16$ in all experiments.

Table 9: Perplexity scores of 390M-parameter FRAC models under ablations of FRAC design choices.

Model	ProofPile ppl↓	PG19 ppl↓	GovReport ppl↓
FRAC	**47.2**	**28.4**	11.3
$\alpha = 1$	61.3	36.2	19.7
w/o write prior	68.5	41.6	18.2
pure multiscale bank	76.3	47.1	17.2
w/o softmax R/W	80.4	36.9	15.5
learned timescales	54.1	34.3	13.5
w/o D	70.9	46.4	23.6
$M=8$	51.4	33.9	14.5
$M=32$	47.9	29.1	**10.6**

To further isolate which components stemming from the FDE motivation are responsible for the gains, we conduct three additional ablations. First, we remove all fractional inductive biases and train a pure multiscale memory bank baseline. Here, we fix $\alpha = 1$, remove the read and write fractional

[11] Training is performed at 4K sequence length.

[12] `https://github.com/zhangir-azerbayev/proof-pile`

[13] Removing $-\alpha_t \log \tau$ in line 5 of Algorithm 1.

priors, and use freely learned timescales initialized independently rather than on a geometric grid (*pure multiscale bank*). Second, we remove the softmax-normalized read and write weights while keeping learnable α (*w/o* softmax R/W). Third, we keep our FRAC layer with learnable α, priors on read and write, and softmax normalization, and only replace τ on a geometric grid with freely learned timescales initialized independently (*learned timescales*). As shown in Table 9, all 3 ablations lead to worse performance compared to FRAC. These results further validate the importance of the fractional parameterization, the geometric timescale grid, and the normalized read/write routing.

Finite SoE approximation quality To directly evaluate the approximation quality of the finite mode bank, we consider the homogeneous fractional relaxation equation (3) with $u(t) = 0$ and $h(0) = 1$, whose exact solution is $h(t) = E_\alpha(-\lambda t^\alpha)$. Using the normalized coordinate $s = \lambda^{1/\alpha} t$ we approximate $E_\alpha(-s^\alpha)$ using positive mixtures of the same geometrically spaced timescales $\tau \in [1, 2^{17}]$ as in FRAC. The resulting range for the normalized time is $s \in [4, 6553.6]$ because we use the normalized step $\Delta\lambda^{1/\alpha} = 0.1$, so the 64K-token horizon corresponds to $s = 0.1 \times 65536 = 6553.6$, and we start at $s = 4\tau_{\min} = 4$ to focus on the regime covered by the exponential bank. For each $\alpha \in \{0.10, 0.11, \ldots, 0.99\}$, we optimize the coefficients c_m of the fixed exponential bank under the simplex constraint $c_m \geq 0$, $\sum_m c_m = 1$, minimizing the maximum absolute error of SoE approximation:

$$\varepsilon_{\alpha,M} = \max_s \left| E_\alpha(-s^\alpha) - \sum_{m=1}^{M} c_m e^{-s/\tau_m} \right|.$$

We repeat this for $M \in \{8, 16, 32\}$ and report the mean of $\varepsilon_{\alpha,M}$ over all values of α in Table 10. The $M = 16$ bank used in our experiments approximately halves the mean maximum error relative to $M = 8$, while increasing to $M = 32$ provides only a marginal further improvement.

Table 10: Approximation error of the finite geometric mode bank.

Modes (M)	8	16	32
Mean maximum error	1.34×10^{-3}	6.82×10^{-4}	6.45×10^{-4}

H Limitations and Broader Impact

H.1 Limitations

Our work has several limitations. First, FRAC realizes the fractional kernel through a finite sum-of-exponentials approximation over a fixed geometric bank of timescales. This makes the model practical and scan-compatible, but the approximation quality depends on the number of modes and on whether the selected timescale range covers the dependencies required by the task. In applications where the relevant memory scale lies outside this range, or where the task is dominated by short-tailed dependencies requiring rapid forgetting, the advantage of the method may be reduced. Second, the computational cost of FRAC scales with the number of modes M. The transition within each mode is simple and diagonal, but the layer maintains a bank of M recurrent states and performs mode-wise read and write operations at every step. Thus, while the method remains linear in sequence length and bounded-state, efficient large-scale use requires care in kernel design and memory layout. Finally, our empirical evaluation focuses on settings where long-range dependencies are central, including synthetic long-memory tasks and long-context language modeling. While these experiments directly test the intended use case of FRAC, they do not exhaust the possible domains where fractional memory may be useful. Evaluating FRAC in additional long-horizon settings, such as video, scientific sequences, and persistent memory for agentic systems, is an important direction for future work.

H.2 Broader Impact

The potential positive impact of this work is to improve the efficiency and long-context capability of sequence models by providing a bounded-state recurrent architecture with an improved memory law. This may support applications where long-range dependencies are important, including language modeling, scientific sequences, time-series data, video, and persistent memory for agentic systems. At the same time, stronger long-context models can inherit the usual risks of capable sequence

models, including misuse in text or code generation, surveillance, automated decision support, and systems that reproduce biases or private information present in training data. Future releases of large pretrained checkpoints based on FRAC should therefore include application-appropriate safeguards, such as dataset review, misuse evaluation, privacy protections, and access controls when needed.

I Data and Asset Licenses

We use public datasets, benchmarks, tokenizers, and evaluation tools, and cite their original papers or repositories. FineWeb/FineWeb-Edu pretraining data, including the 10BT/100BT samples, and the FineWeb-Edu component of SmolLM-Corpus are released under ODC-By v1.0 and subject to the Common Crawl Terms of Use; the Llama 2 tokenizer is used under the Llama 2 Community License; the heavy-tail benchmark and MADLab tasks are generated programmatically and use no external data; and RULER/NIAH code and protocols are released under Apache-2.0. For long-document perplexity evaluation of Appendix G, PG19 is released under Apache-2.0, with text derived from pre-1919 Project Gutenberg books; ProofPile is released under Apache-2.0; and GovReport is released under CC BY 4.0.

For LM Evaluation Harness tasks, WikiText is CC BY-SA 3.0/GFDL; HellaSwag is MIT; WinoGrande is CC BY for data and Apache-2.0 for code; ARC-Easy/Challenge are CC BY-SA 4.0; OpenBookQA is Apache-2.0; and BoolQ is CC BY-SA 3.0. For LongBench and other long-context evaluations, LongBench code is MIT and constituent datasets retain their original terms: NarrativeQA, 2WikiMultiHopQA, TriviaQA, and Natural Questions are Apache-2.0, with Natural Questions also subject to applicable Wikipedia terms; Qasper, MuSiQue, and GovReport are CC BY 4.0; HotpotQA, SQuAD 1.1, and the Hugging Face `ucinlp/drop` distribution are CC BY-SA 4.0; QMSum is MIT; SAMSum and RepoBench-P are CC BY-NC-ND 4.0; and MultiNews is for non-commercial research and educational use only.

The HG38 human reference genome data used for DNA modeling are downloaded following the HyenaDNA repository preprocessing instructions; the HyenaDNA codebase is Apache-2.0, and the underlying UCSC Genome Browser sequence data are freely available under UCSC data-use conditions.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims made in the Abstract are supported by the theoretical development in Section 2, the model construction in Section 3, and the experimental results in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Appendix H.1

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: The main theoretical results state their assumptions in the paper, and the proofs are provided in Appendices A, B, and C.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The paper describes the model architecture and state transition in Sections 3 and Appendix D, and reports the experimental protocols, model configurations, and hyperparameters in Section 4 and Appendix E. These details are intended to make the main synthetic and language-modeling results reproducible.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

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

   Answer:[Yes]

   Justification: All datasets used in the experiments are publicly available or synthetically generated as described in the paper. The model implementation is available at `https://github.com/anasiri/frac-ssm`

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specifies the main experimental settings in Section 4, including the evaluated tasks, baselines, model configurations, training setup, and evaluation protocol. Additional model-size and hyperparameter details are provided in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: For the synthetic experiments, the paper reports mean and standard deviation over multiple random seeds, reflecting variability across independent training runs. For language-modeling experiments, estimating standard errors would require multiple 1.3B-scale pretraining runs, which is computationally infeasible given our limited resources. We therefore omit error bars, following the common practice in large-model studies of prioritizing broad evaluation and comparisons to state-of-the-art baselines.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provided the information on the computer resources in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: This paper conforms to the NeurIPS Code of Ethics in every aspect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

Justification: The paper discusses broader impacts in Appendix H.2.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not involve the release of data or models requiring special safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We follow the protocol and list licenses in Appendix I.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We release the implementation of FRAC at `https://github.com/anasiri/frac-ssm`, with documentation provided in the repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development in this research does not use LLMs as an important, original, or non-standard component. Any use of LLMs was limited to writing, editing, or formatting assistance and did not affect the methodology, scientific rigor, or originality of the work.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.