MAMBA-3: IMPROVED SEQUENCE MODELING USING STATE SPACE PRINCIPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

The recent scaling of test-time compute for LLMs has restricted the practical deployment of models to those with strong capabilities that can generate high-quality outputs in an inference-efficient manner. While current Transformer-based models are the standard, their quadratic compute and linear memory bottlenecks have spurred the development of sub-quadratic models with linear-scaling compute with constant memory requirements. However, many recent linear-style models lack certain capabilities or lag behind in quality, and even their linear-time inference is not hardware-efficient. Guided by an inference-first perspective, we introduce three core methodological improvements inspired by the state-space model viewpoint of linear models. We combine a: 1) more expressive recurrence derived from discretization , 2) complex-valued state update rule that enables richer state tracking, and 3) multi-input, multi-output formulation together, resulting in a stronger model. Together with architectural refinements, our **Mamba-3** model achieves significant gains across retrieval, state-tracking, and downstream language modeling tasks. Our new architecture sets the Pareto-frontier for performance under a fixed inference budget and outperforms strong baselines in a head-to-head comparison.

1 INTRODUCTION

Test-time compute has emerged as a key driver of progress in AI, with techniques like chain-of-thought reasoning and iterative refinement demonstrating that inference-time scaling can unlock new capabilities (Wu et al., 2025; Snell et al., 2024). This paradigm shift makes inference efficiency (Kwon et al., 2023; Li et al., 2024) paramount, as the practical impact of AI systems now depends critically on their ability to perform large-scale inference during deployment. Model architecture design plays a fundamental role in determining inference efficiency, as architectural choices directly dictate the computational and memory requirements during generation. While Transformer-based models (Vaswani et al., 2017) are the current industry standard, they are fundamentally bottlenecked by linearly increasing memory demands through the KV cache and quadratically increasing compute requirements through the self-attention mechanism. These drawbacks have motivated recent lines of work on sub-quadratic models, e.g., state-space models (SSMs), which, despite utilizing only constant memory and linear compute, have comparable or better performance than their Transformer counterparts. Models that benefit the most from this new scaling paradigm perform well on the following three axes: (i) quality, (ii) capability, and (iii) inference efficiency.

Recent model architectures have tried to strike a balance between the three, but many fall short on at least one of these three axes. In particular, Mamba-2 and Gated DeltaNet (GDN), which have gained significant traction and adoption due to their inference efficiency, made architectural design choices that enable their linear compute requirements but sacrifice quality and capabilities (Dao & Gu, 2024; Yang et al., 2025a). For example, Mamba-2 was developed to improve training speed and simplicity over Mamba-1 (Gu & Dao, 2024), opting out of more expressive parameterizations of the underlying SSM and hindering the quality of the model (Dao & Gu, 2024). Linear attention-style models (Katharopoulos et al., 2020) have also been shown to lack certain capabilities, with poor state-tracking abilities, e.g., determining parity of bit sequences, being one of the most notable (Grazzi et al., 2025; Sarrof et al., 2024). In addition, despite these sub-quadratic models being prized for theoretically efficient inference, these inference algorithms are not hardware efficient. In particular, because these algorithms were developed from a training perspective, their decoding phase has low arithmetic intensity (the ratio of FLOPs to memory traffic), resulting in large portions of hardware remaining idle.

To develop more performant models from an inference-first paradigm, we introduce three core methodological changes on top of Mamba-2, influenced by a SSM-centric viewpoint of sub-quadratic models. While many recent models fall into the linear attention framework (Dao & Gu, 2024; Yang et al., 2025a; Sun et al., 2023), we find that the classical SSM toolbox (Kalman, 1960; Gopal, 1993) leads to natural interpretations and improvements on modeling.

Trapezoidal Discretization. We discretize the underlying continuous-time dynamical system with a trapezoidal methodology. The final recurrence is a more expressive superset of Mamba-2's recurrence and can be viewed as a convolution. We combine this new discretization with applied biases on the B, C, inspired by Yu & Erichson (2025), and find that their synergy is able to empirically replace the short causal convolution in language modeling which was previously hypothesized to be essential for recurrent models.

Complex-valued State-Space Model. By viewing the underlying SSM of Mamba-3 as complex-valued, we enable a more expressive state update than Mamba-2's. This change in update rule, designed to be lightweight for training and inference, overcomes the lack of state-tracking ability common in many current linear models. We emphasize that our complex-valued update rule is equivalent to a data-dependent rotary embedding and can be efficiently computed (Su et al., 2023).

Multi-Input, Multi-Output SSM. To improve FLOP-efficiency during decoding, we shift from outer-product-based state update to matrix-multiplication-based state update . In view of the signal processing foundations of SSMs, such a transition exactly coincides with the generalization from a single-input single-output (SISO) sequence dynamic to a multiple-input multiple-output (MIMO) one. Here, we found that MIMO is particularly suitable for inference, as the extra expressivity allows for more compute during state update, without increasing the state size and hence compromising speed.

These three SSM-centric methodological changes are core to our **Mamba-3** mixer primitive. We also make adjustments to the overall architecture to ensure more similarity to the baseline Transformer architecture. Mamba-3 swaps the pre-output projection norm with the more common QK-normalization (Team et al., 2025; OLMo et al., 2025) and makes the short convolution, a common component found in many other sub-quadratic models (Gu & Dao, 2024; Yang et al., 2025a; von Oswald et al., 2025), optional.

We empirically validate our new model on a suite of synthetic and language-modeling tasks.

- **Better Quality.** Mamba-3 matches or outperforms Mamba-2 and other open-source architectures on standard downstream language modeling evaluations. For example, Mamba-3-1.5B's average accuracy on all downstream tasks is better than that of its Transformer, Mamba-2, and Gated DeltaNet counterparts.
- **New Capabilities.** Mamba-3's complexification of the SSM state enables the model to solve synthetic state-tracking tasks that Mamba-2 cannot. We empirically demonstrate that the efficient RoPE-like calculation is able to near perfectly solve arithmetic tasks, while Mamba-3 without RoPE and Mamba-2 perform not better than random guessing.
- **Stronger Inference Efficiency.** Mamba-3's MIMO variant retains the same state size while enabling better hardware utilization compared to standard Mamba-3 and other models. Its improved performance without increased memory requirements pushes the pareto-frontier of inference efficiency.

2 PRELIMINARIES

2.1 NOTATION

Scalars are denoted by plain-text letters (e.g., x, y). Tensors, including vectors and matrices, are denoted by bold letters (e.g., $\boldsymbol{h}, \mathbf{C}$). The shape of the tensor can be inferred from the context. We denote the input sequence length as T, the model dimension as D, and the SSM state size as N. For time indices, we use subscripts (e.g., x_t for the input at time t). The Hadamard product between two tensors is denoted by $\odot$. For a vector of size $\boldsymbol{v} \in \mathbb{R}^d$, we denote $\mathrm{Diag}(\boldsymbol{v}) \in \mathbb{R}^{d \times d}$ as the diagonal matrix with the vector $\boldsymbol{v}$ as the diagonal, and for products of scalars across time steps, we use the notation $\alpha_{t \cdots s} = \alpha_{t:s}^{\times} = \prod_{i=s}^{t} \alpha_i$.

2.2 SSM PRELIMINARIES

State Space Models (SSMs) describe continuous-time linear dynamics via

$$\dot{\boldsymbol{h}}(t) = \mathbf{A}(t)\,\boldsymbol{h}(t) + \mathbf{B}(t)\,x(t), \qquad\qquad y(t) = \mathbf{C}(t)^\top \boldsymbol{h}(t),$$

where $\boldsymbol{h}(t) \in \mathbb{R}^N$ is the hidden state, $x(t) \in \mathbb{R}$ the input, and $\mathbf{A}(t) \in \mathbb{R}^{N\times N}$, $\mathbf{B}(t), \mathbf{C}(t) \in \mathbb{R}^N$. For discrete sequences with step size Δ_t, Euler's discretization gives the recurrence

$$\boldsymbol{h}_t = e^{\Delta_t \mathbf{A}_t}\,\boldsymbol{h}_{t-1} + \Delta_t\,\mathbf{B}_t\,x_t, \qquad\qquad y_t = \mathbf{C}_t^\top \boldsymbol{h}_t.$$

Mamba-2's parameterization. Mamba-2 (Dao & Gu, 2024) makes the SSM *data-dependent* and hardware-efficient by (i) projecting $A = \mathbf{A} \in \mathbb{R}_{<0}$, and $\mathbf{B}, \mathbf{C} \in \mathbb{R}^N$ from the current token and (ii) choosing transition matrix $A = \mathbf{A}$ as a data-dependent scalar. Writing $\alpha_t := e^{\Delta_t A_t} \in (0, 1)$ and $\gamma_t := \Delta_t$, the update becomes

$$\boldsymbol{h}_t = \alpha_t\,\boldsymbol{h}_{t-1} + \gamma_t\,\mathbf{B}_t\,x_t, \qquad y_t = \mathbf{C}_t^\top \boldsymbol{h}_t.$$

The scalar $A_t < 0$ is an input-dependent *forget-gate (decay)* α_t, and the parameter *selectivity* Δ_t jointly controls the forget-gate ($\alpha_t = \exp(\Delta_t A_t)$) and the input-gate ($\gamma_t = \Delta_t$): larger Δ_t forgets faster and up-weights the current token more strongly, while smaller Δ_t retains the hidden state with minimal contributions from the current token.

2.3 STRUCTURED MASKED REPRESENTATION AND STATE SPACE DUALITY

Dao & Gu (2024) show that a large class of SSMs admit a *matrix* form that vectorizes the time-step recurrence. For instance, Mamba-2's recurrence can be vectorized as a masked matrix multiplication,

$$\mathbf{Y} = (\mathbf{L} \odot \mathbf{C}\bar{\mathbf{B}}^\top)\mathbf{X} = \left(\begin{bmatrix} 1 & & & \\ \alpha_1 & 1 & & \\ \vdots & & \ddots & \\ \alpha_{T\ldots 1} & \cdots & \alpha_T & 1 \end{bmatrix} \odot \mathbf{C}\mathbf{B}^\top \right) \mathbf{X}, \tag{1}$$

where $\mathbf{L} \in \mathbb{R}^{T\times T}$ is the structured mask, $\mathbf{B}, \mathbf{C} \in \mathbb{R}^{T\times N}$, $\mathbf{X} \in \mathbb{R}^{T\times D}$ is the input to the SSM and $\mathbf{Y} \in \mathbb{R}^{T\times D}$ is its output. Within this form, Mamba-2 can be viewed as a type of linear attention by setting $\mathbf{Q} = \mathbf{C}$, $\mathbf{K} = \mathbf{B}$, $\mathbf{V} = \mathbf{X}$ and viewing $\mathbf{L}$ as a causal, data-dependent mask. When all $\alpha = 1$, the expression reduces to (causal) linear attention (Katharopoulos et al., 2020). A more detailed coverage of related linear-time sequence mixers can be found at Appendix A.

3 MODEL DESIGN FROM A STATE-SPACE VIEWPOINT

We introduce Mamba-3, with three new innovations rooted in classical state-space theory: trapezoidal discretization for more expressive dynamics, complex-valued state spaces for state-tracking, and multi-input multi-output (MIMO) to improve hardware utilization. These advances address the quality, capability, and efficiency limitations of current sub-quadratic architectures.

3.1 TRAPEZOIDAL DISCRETIZATION

Structured SSMs are naturally defined as continuous-time dynamical systems that map input functions, $x(t) \in \mathbb{R}$, to output functions, $y(t) \in \mathbb{R}$, for time $t > 0$. In sequence modeling, however, the data is only observed at discrete time steps, which then requires applying a *discretization step* to the SSM to transform its continuous-time dynamics into a discrete recurrence. The preliminary step in deriving Mamba-3's discretization is to apply the Variation of Constants formula (Proposition 5), which decomposes the hidden state into an exponentially decay term and a state update term "information" term dependent on the most recent inputs.

The first step in deriving the discretized recurrence is to approximate the "state-update" integral in equation 10. A straightforward choice, used in Mamba-2, is applying *Euler's rule* (Süli & Mayers, 2003), which approximates the integral by holding the (right) endpoint constant throughout the interval (Fig. 1). This yields Mamba-2's recurrence,

$$\begin{aligned} \boldsymbol{h}_t \;&=\; e^{\Delta_t A_t}\,\boldsymbol{h}_{t-1} \;+\; (\tau_t - \tau_{t-1}) e^{(\tau_t - \tau_t) A_t}\,\mathbf{B}_t\,x_t \\ &\approx\; e^{\Delta_t A_t}\,\boldsymbol{h}_{t-1} \;+\; \Delta_t\,\mathbf{B}_t\,x_t. \end{aligned} \tag{2}$$

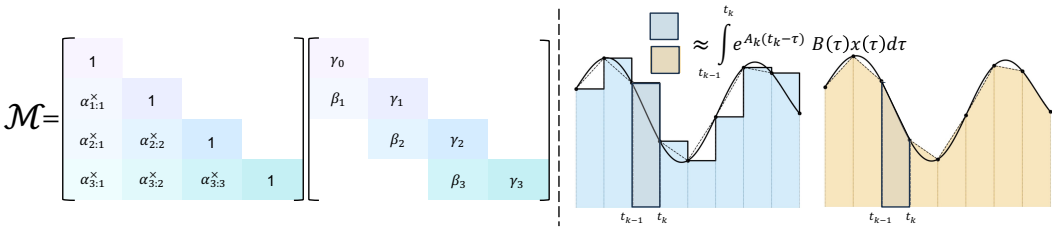

Figure 1: **Left:** The structured mask induced by the generalized trapezoid rule is a product of the decay and convolutional mask. **Right:** Euler (hold endpoint) vs trapezoidal rule (average endpoints).

However, Euler's rule provides only a first-order approximation to the "state-update" integral: local truncation error is $O(\Delta_t^2)$, which accumulates across steps to yield a global error of $O(\Delta_t)$ over the sequence. In contrast, we adopt a *generalized trapezoidal rule*, which provides a second-order accurate approximation of the integral, offering improved accuracy over the Euler's rule. Specifically, it approximates the integral with a *data-dependent, convex combination of both interval endpoints*. This generalization extends the classical trapezoidal rule (Süli & Mayers, 2003), which simply averages the interval endpoints, by allowing for a *data-dependent convex combination* (Fig. 1).

Proposition 1 (Generalized Trapezoidal Discretization)**.** *Approximating the state-update integral in equation 10 by the general trapezoidal rule yields the recurrence,*

$$\boldsymbol{h}_t = e^{\Delta_t A_t}\boldsymbol{h}_{t-1} + (1-\lambda_t)\Delta_t e^{\Delta_t A_t}\mathbf{B}_{t-1}x_{t-1} + \lambda_t\Delta_t\mathbf{B}_t x_t, \tag{3}$$

$$:= \alpha_t\boldsymbol{h}_{t-1} + \beta_t\mathbf{B}_{t-1}x_{t-1} + \gamma_t\mathbf{B}_t x_t, \tag{4}$$

where $\lambda_t \in [0,1]$ *is a data-dependent scalar,* $\alpha_t := e^{\Delta_t A_t}$, $\beta_t := (1-\lambda_t)\Delta_t e^{\Delta_t A_t}$, $\gamma_t := \lambda_t\Delta_t$.

Remark 1 (Expressivity). Our scheme is a generalization of a) The classical trapezoid rule which is recovered when $\lambda_t = \frac{1}{2}$. b) Mamba-2's Euler's rule, which is recovered when $\lambda_t = 1$.

Remark 2 (Error Rate). This is a second-order discretization with local truncation error $O(\Delta_t^3)$ and global error $O(\Delta_t^2)$ over the sequence under standard stability assumptions, provided that the trapezoidal parameter satisfies $\lambda_t = \frac{1}{2} + O(\Delta_t)$. However, our ablations indicate that not enforcing this constraint is the best for empirical performance. See Appendix B.2,B.3 for details.

3.1.1 Trapezoidal Discretization is a Convolutional Mask

We can view the generalized trapezoidal discretization as applying a *data-dependent* convolution of size two on the projected input, $\mathbf{B}_t x_t$, to the SSM. We now show that a similar vectorization to Equation (1) holds with the generalized trapezoidal discretization. Unrolling the recurrence starting from $\boldsymbol{h}_0 = \gamma_0\mathbf{B}_0 x_0$ results in $\boldsymbol{h}_T = \alpha_{T\cdots 2}(\gamma_0\alpha_1 + \beta_1)\,\mathbf{B}_0 x_0 + \cdots + \gamma_T\mathbf{B}_T x_T$.

Unrolling these rows shows that the mask induced by the trapezoidal update is no longer a fixed averaging of endpoints (as in the classical trapezoidal rule), but a *data-dependent convex combination* of the two interval endpoints. In the SSD representation, this corresponds to a mask $\mathbf{L}$:

$$\begin{bmatrix} \gamma_0 & & & \\ (\gamma_0\alpha_1+\beta_1) & & & \\ \alpha_2(\gamma_0\alpha_1+\beta_1) & \gamma_2 & & \\ \vdots & & \ddots & \\ \alpha_{T\cdots 2}(\gamma_0\alpha_1+\beta_1) & & \cdots & \gamma_T \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \alpha_1 & 1 & & \\ \alpha_2\alpha_1 & & & \\ \vdots & & \ddots & \\ \alpha_{T\cdots 1} & & \cdots & 1 \end{bmatrix} \begin{bmatrix} \gamma_0 & & & \\ \beta_1 & & & \\ 0 & \gamma_2 & & \\ \vdots & & \ddots & \\ 0 & & \cdots & \gamma_T \end{bmatrix}. \tag{5}$$

Here, the first factor is precisely the lower-triangular decay mask from Mamba-2, while the second factor encodes the size two convolution induced by the trapezoidal rule through the coefficients (β_t, γ_t). We provide a rigorous proof for this decomposition in Appendix B.1.

3.2 Complex-Valued SSMs

Modern SSMs are designed with efficiency as the central goal, motivated by the need to scale to larger models and longer sequences. For instance, successive architectures have progressively simplified the state transition matrix: S4 (Gu et al., 2022a) used complex-valued Normal plus Low Rank (NPLR) matrices, Mamba (Gu & Dao, 2024) reduced this to a diagonal of reals, and Mamba-2 (Dao & Gu, 2024) further simplified it to a single scalar. Although these simplifications largely maintain language modeling performance, recent works (Merrill et al., 2025; Sarrof et al., 2024; Grazzi et al., 2025) have shown that they degrade the capabilities of the model on simple state-tracking tasks such as parity and modular arithmetic, which can be solved by a one-layer LSTM.

This limitation, formalized in Theorem-1 of (Grazzi et al., 2024), arises from restricting the eigenvalues of the transition matrix to real numbers, which cannot represent "rotational" hidden state dynamics. For instance, consider the parity function on binary inputs $\{0, 1\}$, defined as $\sum_t x_t \bmod 2$. This task can be performed using update: $\boldsymbol{h}_t = \mathbf{R}(\pi x_t)\boldsymbol{h}_{t-1}$, where $\mathbf{R}(\cdot)$ is a 2-D rotation matrix. Such rotational dynamics cannot be expressed with real eigenvalues.

To recover this capability, we begin with complex SSMs (6), which are capable of representing state-tracking dynamics. We show that, under discretization (Proposition 5), complex SSMs can be formulated as a real SSMs with a *block-diagonal transition matrix composed of* 2×2 *rotation matrices* (Proposition 2). We then show that this is equivalent to applying *data-dependent rotary embeddings* on both the input and output projections $\mathbf{B}, \mathbf{C}$ respectively. This result establishes a theoretical connection between complex SSMs and data-dependent RoPE embeddings (Proposition 3). Finally, this allows for an efficient implementation of the complex-valued SSM via the "RoPE trick", enabling efficient complex-valued state transition matrix with minimal computational overhead over real-valued SSMs.

Proposition 2 (Complex-to-Real SSM Equivalence)**.** *Consider a complex-valued SSM*

$$\begin{aligned} \dot{\boldsymbol{h}}(t) &= \mathrm{Diag}\big(A(t) + i\boldsymbol{\theta}(t)\big)\,\boldsymbol{h}(t) + \big(\mathbf{B}(t) + i\hat{\mathbf{B}}(t)\big)\,x(t), \\ y(t) &= \mathrm{Re}\Big(\big(\mathbf{C}(t) + i\hat{\mathbf{C}}(t)\big)^\top \boldsymbol{h}(t)\Big), \end{aligned} \tag{6}$$

where $\boldsymbol{h}(t) \in \mathbb{C}^{N/2}$*,* $\boldsymbol{\theta}(t), \mathbf{B}(t), \hat{\mathbf{B}}(t), \mathbf{C}(t), \hat{\mathbf{C}}(t) \in \mathbb{R}^{N/2}$*, and* $x(t), A(t) \in \mathbb{R}$*. Under Euler discretization, this system is equivalent to a real-valued SSM*

$$\begin{aligned} \boldsymbol{h}_t &= e^{\Delta_t A_t}\,\mathbf{R}_t\,\boldsymbol{h}_{t-1} + \Delta_t \mathbf{B}_t x_t, \\ y_t &= \mathbf{C}_t^\top \boldsymbol{h}_t, \end{aligned} \tag{7}$$

with state $\boldsymbol{h}_t \in \mathbb{R}^N$*, projections*

$$\mathbf{B}_t = \begin{bmatrix} \mathbf{B}_t \\ \hat{\mathbf{B}}_t \end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{C}_t = \begin{bmatrix} \mathbf{C}_t \\ -\hat{\mathbf{C}}_t \end{bmatrix} \in \mathbb{R}^N,$$

and a transition matrix

$$\mathbf{R}_t \;=\; \mathit{Block}\Big(\{R(\Delta_t \boldsymbol{\theta}_t[i])\}_{i=1}^{N/2}\Big) \in \mathbb{R}^{N\times N}, \qquad R(\Theta) = \begin{bmatrix} \cos(\Theta) & -\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) \end{bmatrix}.$$

The proof is in Appendix C.1.

Proposition 2 shows that the discretized complex SSM has an equivalent real SSM with doubled state dimension (N), and a block-diagonal transition matrix multiplied with a scalar decay, where each 2×2 block is a data-dependent rotation matrix ($e_t^{\Delta_t A}\mathbf{R}_t$). We now show that the rotations can equivalently be absorbed into the input and output projections $\mathbf{B}_t, \mathbf{C}_t$, yielding an equivalent view that *complex SSMs are real SSMs equipped with data-dependent rotary embeddings (RoPE).*

Proposition 3 (Complex SSM, Data-Dependent RoPE Equivalence)**.** *Under the notation established in Proposition 2, consider the real SSM defined in Eq. 7 unrolled for T time-steps. The output of the above SSM is equivalent to that of a vanilla scalar transition matrix-based SSM (Eq. 2) with a data-dependent rotary embedding applied on the $\mathbf{B}, \mathbf{C}$ components of the SSM defined as:*

$$\boldsymbol{h}_t = e^{\Delta_t A_t}\boldsymbol{h}_{t-1} + (\prod_{i=0}^{t} \mathbf{R}_i^\top)\mathbf{B}_t x_t, \qquad \boldsymbol{y}_t = \left((\prod_{i=0}^{t} \mathbf{R}_i^\top)\mathbf{C}_t\right)^\top \boldsymbol{h}_t \tag{8}$$

where the matrix production represents right matrix multiplication, e.g., $\prod_{i=0}^{1} \mathbf{R}_i = \mathbf{R}_0\mathbf{R}_1$*. We denote employing the vanilla SSM to compute the Complex SSM as "RoPE trick".*

The proof is in Appendix C.2.

To observe the connection of complex SSMs to RoPE embeddings, note that in the above proposition, the data-dependent rotations $\mathbf{R}_i$ are aggregated across time-steps and applied to $\mathbf{C}, \mathbf{B}$, which, by the State Space Duality of Dao & Gu (2024), correspond to the Query ($\mathbf{Q}$) and Key ($\mathbf{K}$) components of Attention. Analogously, vanilla RoPE (Su et al., 2023) applies *data-independent* rotation matrices, where the rotation angles follow a fixed frequency schedule $\boldsymbol{\theta}[i] = 10000^{-2i/N}$.

Remark 3 (Generality). Proposition 3 extends to the fully general case where the transition is given by any complex matrix. By the complex diagonalization theorem, such a matrix is unitarily equivalent to a complex diagonal matrix, $\mathrm{Diag}\big(\mathbf{A}(t) + i\boldsymbol{\theta}(t)\big)$ with $\mathbf{A}(t) \in \mathbb{R}^N$. However, in practice, we restrict $\mathbf{A}(t)$ to a scalar, mirroring the simplification from Mamba to Mamba-2, to enable faster implementation by avoiding GPU memory bottlenecks.

Proposition 4 (Rotary Embedding Equivalence with Trapezoidal Discretization)**.** *Discretizing a complex SSM with the trapezoidal rule (Proposition 1) yields the recurrence*

$$\boldsymbol{h}_t = \alpha_t \boldsymbol{h}_{t-1} + \beta_t \left(\prod_{i=0}^{t-1} \mathbf{R}_i^\top\right) \mathbf{B}_{t-1} x_{t-1} + \gamma_t \left(\prod_{i=0}^{t} \mathbf{R}_i^\top\right) \mathbf{B}_t x_t,$$
$$\boldsymbol{y}_t = \left((\prod_{i=0}^{t} \mathbf{R}_i^\top)\mathbf{C}_t\right)^\top \boldsymbol{h}_t. \tag{9}$$

Here $\mathbf{R}_t$ is the block-diagonal rotation matrix defined in Proposition 3.

The proof is in Appendix C.3.

Remark 4 (RoPE Trick). Complex SSMs discretized with the general trapezoidal rule of a complex SSM naturally admit the RoPE trick we established for SSMs discretized with Euler's rule.

3.3 Multi-Input, Multi-Output

During the decoding phase of autoregressive inference, outputs are generated one token at a time, and performance is typically measured using in *Tokens generated Per Second (TPS)*. In this metric, sub-quadratic models, such as Mamba-2 (Dao & Gu, 2024), have a significant advantage over standard Transformer-style attention, since they feature a fixed-size hidden state (Equation (2)) rather than maintaining a key–value (KV) cache that grows linearly with the sequence length.

TPS, however, does not explicitly factor in hardware efficiency, where we aim to be in a compute-bound regime (as opposed to memory-bound) in order to fully utilize on-chip accelerators. To better characterize hardware efficiency, we would need to consider the arithmetic intensity of token generation. Recall that arithmetic intensity is defined as FLOPs divided by the number of input-output bytes, for a given op. In order to fully utilize both the accelerators and the bandwidth, we would like the arithmetic intensity to match the ops:byte ratio of the hardware, which in the case of NVIDIA H100-SXM5, is 295.2 bfloat16 ops per second with respect to the DRAM, and 31.9 bfloat16 ops per second with respect to the SRAM [Fleetwood].

Table 2(a) shows the arithmetic intensity for a single generation in the SSM component of Mamba (with respect to 2-byte data). We see that it falls far short of a compute-bound regime, and moreover it is not clear how one can adjust the existing parameters in Mamba to mitigate the lack of hardware efficiency. We note that this observation applies generally to other sub-quadratic models, such as causal linear attention.

Input	Output	FLOPs	Arithmetic Intensity
$H_t : (n, p)$ $x_t : (p)$ $a_t : (1)$ $b_t : (n)$ $c_t : (n)$	$y_t : (p)$	$5pn$	$\frac{5pn}{2(1+2n+p+np)}$ $\approx 2.5 = \Theta(1)$

(a) SISO (2-byte data).

Input	Output	FLOPs	Arithmetic Intensity
$H_t : (n, p)$ $x_t : (p, r)$ $a_t : (1)$ $b_t : (n, r)$ $c_t : (n, r)$	$y_t :$ (p, r)	$4nrp+$ $2np$	$\frac{p(4nr+2n)}{2(1+2nr+pr+np)}$ $\approx 2r = \Theta(r)$

(b) MIMO (2-byte data).

Figure 2: Arithmetic Intensity for (a) SISO, (b) MIMO. Batch and head dimensions cancel out.

In light of this, we made the following simple adjustment to our recurrent relation: instead of transforming the input $\mathbf{x}_t \in \mathbb{R}^p$ to state $\mathbf{H}_t \in \mathbb{R}^{n\times p}$ via an outer product, i.e., $\mathbf{H}_t \leftarrow a_t \mathbf{H}_{t-1} + \mathbf{b}_t \otimes \mathbf{x}_t$, we made such a transformation via a matrix product, i.e., $\mathbf{H}_t \leftarrow a_t \mathbf{H}_{t-1} + \mathbf{B}_t \mathbf{X}_t^\top$, where $\mathbf{B}_t \in \mathbb{R}^{n\times r}$ and $\mathbf{X}_t \in \mathbb{R}^{p\times r}$ are now matrices with an additional rank r. The emission from state to output similarly acquire an extra rank r, i.e., $\mathbf{Y}_t \in \mathbb{R}^{r\times p} \leftarrow \mathbf{C}_t^\top \mathbf{H}_t$, where $\mathbf{C}_t \in \mathbb{R}^{n\times r}, \mathbf{H}_t \in \mathbb{R}^{n\times p}$. This simple change increases the arithmetic intensity of recurrence, which now scales with the rank

r (Figure 2(b)). Hence, by increasing r, arithmetic intensity improves and shifts decode generation towards a more compute-bound regime. This increase in FLOPs during decode does not compromise runtime, as the operation is bounded by the I/O of state $\mathbf{H}_t \in \mathbb{R}^{n \times p}$.

Moreover, moving from outer-product-based state update to matrix-product-based coincides exactly with generalizing from SISO to MIMO SSM, with the rank r being the MIMO rank. Such a generalization recovers a key expressive feature of SSMs in classical literature; indeed, there has been previous work, namely Smith et al. (2023), that explored MIMO SSM as a drop-in replacement of attention, albeit not in the context of Mamba and not necessarily with inference in view. We note that training and prefilling is generally compute bound, resulting in MIMO incurring increased costs during these stages, while decoding, a memory-bound operation, sees very little increase in latency when utilizing MIMO over SISO.

Details of the MIMO formulation for Mamba-3 are provided in Appendix D.

3.4 Mamba-3 Architecture

The Mamba-3 block retains the overall layout of its predecessor while introducing several key modifications. Most notably, the SSD layer is replaced with the more expressive trapezoidal SSM defined in Proposition 4. The extra normalization layer, first introduced between Mamba-1 and Mamba-2 for training stability, is repositioned to follow the $\mathbf{B}, \mathbf{C}$ projection, mirroring the QK-Norm commonly used in modern Transformers (Henry et al., 2020; Wortsman et al., 2023). Inspired by the findings of Yu & Erichson (2025), which prove adding channel-specific bias to $\mathbf{B}$ in a blockwise variant of Mamba-1 grants universal approximation capabilities, Mamba-3 incorporates a head-specific, channel-wise bias into both the $\mathbf{B}$ and $\mathbf{C}$ components after its normalization. These learnable biases are data-independent parameters that are initialized to all ones and independent across $\mathbf{B}$ and $\mathbf{C}$ (ablations for bias parameterization can be found in Appendix G). Our trapezoidal discretization complements this bias, empirically eliminating the need for the original short causal convolution and its accompanying activation function (Section 4.3). Mamba-3 employs the SISO SSM by default, though we view its MIMO variant as a flexible option that can be toggled depending on inference requirements. The overall architecture follows the Llama design (Grattafiori et al., 2024), alternating Mamba-3 and SwiGLU blocks with pre-normalization.

4 Empirical Validation

We empirically validate our SSM-centric methodological changes through the Mamba-3 model on a host of synthetic and real world tasks. Section 4.1 compares our SISO-variant of Mamba-3 on language modeling and retrieval-based tasks, while Section 4.2 demonstrates inference efficiency of Mamba-3 and MIMO Mamba-3's benefits over SISO Mamba-3 under fixed inference compute. We ablate the impact of our new discretization and BC bias on performance and show that complexification of the SSM leads capabilities that prior SSMs such as Mamba-2 lacked in Section 4.3.

4.1 Language Modeling

All models are pretrained with 100B tokens of the FineWeb-Edu dataset (Penedo et al., 2024) with the Llama-3.1 tokenizer (Grattafiori et al., 2024) at a 2K context length with the same standard training protocol. Training and evaluation details can be found in Appendix E.

Across all four model scales, Mamba-3 outperforms popular baselines at various downstream tasks (Table 1). We highlight that Mamba-3 does not utilize the short convolution that has been empirically identified as an important component in many performant linear models (Allen-Zhu, 2025).

4.1.1 Retrieval Capabilities

Beyond standard language modeling, an important measure for linear models is their retrieval ability — how well they can recall information from earlier in the sequence (Arora et al., 2025a;b). Unlike attention models, which can freely revisit past context with the growing KV cache, linear models must compress context into a fixed-size state. This trade-off is reflected in the Transformer baseline's substantially stronger retrieval scores. To evaluate Mamba-3 under this lens, Table 2 compares it against baselines on both real-world and synthetic needle-in-a-haystack (NIAH) tasks (Hsieh et al., 2024), using our pretrained 1.5B models from Section 4.1. We restrict the task sequence length to 2K tokens to match the training setup and adopt the cloze-style format for our real-world tasks to mirror the next-token-prediction objective, following Arora et al. (2025b; 2024).

Mamba-3 is competitive on real-world associative recall and question-answering but struggles when extracting information from semi-structured or unstructured data. On synthetic NIAH tasks, how-

Table 1: Downstream language modeling evaluations on models trained with 100B FineWeb-Edu tokens. Best results for each size are **bolded**, and second best are underlined. All models are trained with the same procedure. Mamba-3 outperforms Mamba-2 and others at every model scale.

Model	FW-Edu ppl ↓	LAMB. ppl ↓	LAMB. acc ↑	HellaS. acc_n ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc_n ↑	WinoGr. acc ↑	OBQA acc ↑	Average acc ↑
Transformer-180M	16.89	45.0	32.5	39.0	**67.1**	59.8	27.9	51.2	21.8	42.8
Gated DeltaNet-180M	16.61	**35.9**	**33.7**	40.2	66.8	59.6	**28.5**	51.2	21.6	43.1
Mamba-2-180M	16.76	41.8	30.9	40.1	66.8	60.1	27.3	**52.0**	**23.2**	42.9
Mamba-3-180M (SISO)	**16.59**	37.7	32.5	**40.8**	66.1	**61.5**	27.9	**52.0**	22.8	**43.4**
Transformer-440M	13.03	21.2	**41.7**	50.5	69.9	67.6	34.6	**56.7**	**26.0**	49.6
Gated DeltaNet-440M	13.12	**19.0**	40.4	50.5	70.5	67.5	34.0	55.3	25.8	49.1
Mamba-2-440M	13.00	19.6	40.8	**51.7**	70.6	68.8	**35.0**	54.1	**26.0**	49.6
Mamba-3-440M (SISO)	**12.87**	19.6	40.2	**51.7**	**71.9**	**68.9**	34.4	55.8	**26.0**	**49.8**
Transformer-880M	11.42	15.0	44.7	57.2	72.6	71.6	39.2	57.7	26.8	52.8
Gated DeltaNet-880M	11.39	**12.7**	47.1	57.5	72.6	72.5	38.8	57.9	**30.6**	53.9
Mamba-2-880M	11.35	13.8	45.0	58.1	72.5	72.3	38.7	56.8	30.2	53.4
Mamba-3-880M (SISO)	**11.23**	12.9	**47.2**	**58.8**	**73.6**	**72.7**	**40.2**	**58.4**	30.0	**54.4**
Transformer-1.5B	10.51	11.1	**50.3**	60.6	73.8	74.0	40.4	58.7	29.6	55.4
Gated DeltaNet-1.5B	10.51	**10.8**	49.9	60.5	**74.3**	73.3	40.4	**61.5**	30.4	55.7
Mamba-2-1.5B	10.47	12.0	47.8	61.4	73.6	75.3	41.8	57.5	**32.6**	55.7
Mamba-3-1.5B (SISO)	**10.35**	10.9	49.4	**61.9**	73.6	**75.9**	**42.7**	59.4	32.0	**56.4**

Table 2: Retrieval capabilities measured by a mixture of real-world and synthetic retrieval tasks. Real-world retrieval tasks utilize cloze variants of the original datasets and are truncated to 2K length. Mamba-3 demonstrates strong associative recall and question-answering but suffers with information extraction of semi-structured and unstructured data. Mamba-3 has strong needle-in-a-haystack (NIAH) accuracy and generalizes outside its trained context.

Model (1.5B)	SWDE	SQUAD	FDA	TQA	NQ	Drop	NIAH-Single-1			NIAH-Single-2			NIAH-Single-3		
Context Length	2048						1024	2048	4096	1024	2048	4096	1024	2048	4096
Transformer	48.9	46.6	58.4	67.5	31.7	26.4	100.0	100.0	0.0	92.2	100.0	0.0	98.6	99.4	0.0
Gated DeltaNet	**32.7**	40.0	**28.3**	63.5	25.7	24.5	**100.0**	**100.0**	**99.8**	**100.0**	93.8	49.8	83.8	68.4	**34.2**
Mamba-2	30.7	39.1	23.7	64.3	25.1	**28.5**	**100.0**	99.6	62.0	**100.0**	53.8	11.8	**95.8**	**87.4**	13.4
Mamba-3 (SISO)	28.5	**40.1**	23.4	**64.5**	**26.5**	27.4	**100.0**	**100.0**	88.2	**100.0**	**95.4**	**50.6**	92.4	81.4	**34.2**

ever, Mamba-3 surpasses or matches baselines on most cases and notably demonstrates markedly better out-of-distribution retrieval abilities than its Mamba-2 predecessor.

4.2 Inference Efficiency

In this section, we investigate our methodological changes in the context of inference performance. We first present our inference benchmark in Section 4.2.1; we then establish a framework for comparing the inference performance in Section 4.2.2. Finally, we focus on the effectiveness of MIMO in Section 4.2.3.

4.2.1 Fast Mamba-3 Kernels

We complement Mamba-3's methodological advances with optimized kernels that deliver fast inference in practical settings. Specifically, we implement a new series of inference kernels for Mamba-3—using Triton for the forward (prefill) path and CuTe-DSL for decode—and compare their per-token decode latency against the released Triton kernels for Mamba-2 and Gated DeltaNet (GDN)[1] in Table 3. The evaluation uses the setting: a decode step at batch size 128 on a single H100 for 1.5B-parameter models with model dimension 2048, state dimension $\in \{64, 128\}$ in both FP32 and BF16 datatypes. Across all configurations, SISO achieves the lowest latency amongst baselines, while MIMO incurs only a minor overhead relative to SISO. This indicates that our CuTe-DSL decode implementation is competitive and that the additional components of Mamba-3 (trapezoidal update, complex-valued state, and MIMO projections) are lightweight. This supports our overall inference-first perspective: the Mamba-3 admits **simple, low-latency implementation** while providing strong empirical performance. A thorough analysis, including prefill and prefill with decode results are provided in Appendix H.

Model	FP32		BF16	
	$d_{\text{state}} = 64$	$d_{\text{state}} = 128$	$d_{\text{state}} = 64$	$d_{\text{state}} = 128$
Mamba-2	0.295	0.409	0.127	0.203
Gated DeltaNet	0.344	0.423	0.176	0.257
Mamba-3 (SISO)	0.261	0.356	0.106	0.152
Mamba-3 (MIMO)	0.285	0.392	0.136	0.185

Table 3: Latency (in milliseconds) comparison across models, precision, and d_{state} values. Both Mamba-3 SISO and MIMO are faster than the Mamba-2 and Gated DeltaNet at the commonly used bf16, $d_{\text{state}} = 128$ setting.

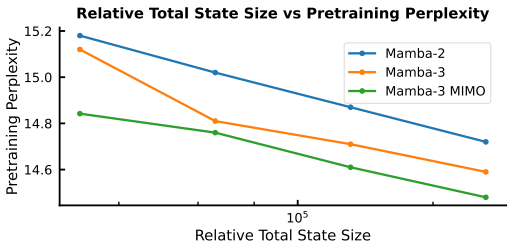

Figure 3: Exploration of state size (inference speed proxy) versus pretraining perplexity (performance proxy) across different Mamba variants. Mamba-3 MIMO drives the-Pareto frontier without increasing state size.

4.2.2 Pareto Frontier for Inference Efficiency

For Mamba and many variants of sub-quadratic models, the generation of tokens during decoding is heavily dominated by memory I/O due to the low arithmetic intensity of computing the recurrent update (c.f. Section 3.3). Furthermore, among the data being transferred, the latent state $\mathbf{H_t}$ dominates in terms of size. Indeed, from Table 3, we see that the runtime scales with d_{state}, which configures the size of the hidden state.

As d_{state} dominates the decode runtime for the subquadratic models considered in this paper, we opt to use it as a proxy for inference speed. By plotting the validation perplexity (itself a proxy for model performance) as a function of d_{state}, we aim to formulate a holistic picture about how the subquadratic models can trade off performance with inference speed.

Figure 3 shows such a Pareto front for the Mamba variants models considered in this paper. For each data point, we train a 440M parameter model to $2\times$ Chinchilla optimal tokens on the Fineweb-Edu dataset, where the model is configured with a d_{state} of $\{16, 32, 64, 128\}$. As expected, we observe an inverse correlation between validation loss and d_{state}; moreover, we noticed a general downward shift on the Pareto front moving from Mamba-2 to Mamba-3. A further downward shift is observed when moving from the SISO variant of Mamba-3 to the MIMO variant of Mamba-3 (where we set the Mimo rank $r = 4$ and decrease our MLP inner dimension to parameter match the SISO variants). We expand the comparison to include the Gated DeltaNet baseline in Figure 7. The results highlight both the expressivity gain coming our methodology change as well as the effectiveness of the MIMO mechanism in improving decoding efficiency.

4.2.3 MIMO enhances inference efficiency

MIMO, with its higher arithmetic intensity, increases the decoding FLOPs without significantly increasing decode runtime (Table 3)[2] The implication is that any performance gain from MIMO translates into efficiency gain in decoding: a conclusion supported by the downward shift of the MIMO pareto curve we observed in Section 4.2.2.

We aim to further verify the gain from MIMO by investigating its language-modeling capabilities. To that end, we train a 440M and 820M parameter MIMO models with MIMO rank $r = 4$ on 100B tokens on Fineweb-Edu (i.e., same setting as the 440M parameter run in Section 4.1; we are currently training the 1.5B model). To ensure the total parameter count equals SISO, we decrease the inner dimension of the MLP layers to compensate for the increase due to the MIMO projections.

On both validation perplexity and our suite of language evaluation tasks (Table 6), we see significant gain when moving from SISO to MIMO. Namely, we attain a perplexity gain of 0.16 on the 100B tokens run, and Figure 3 illustrates the downward shift in our validation loss. On the language evaluation front, we see significant gain on most tasks when compared to SISO, resulting in an overall gain of 1.2 point over SISO. This strongly supports MIMO as a SSM-centric technique to improve model quality without compromising decoding speed.

[1]Details on each kernel DSL and the exact kernel fusion structure is provided in Appendix H.

[2]The kernel for MIMO Mamba-3 in fact fuses the MIMO projection, and so the reported wall clock time is actually an overestimate for the pure SSM update.

Table 4: **Left**: Ablations on core modeling components of Mamba-3, results on test split of dataset. A combination of our BC bias and trapezoidal discretization makes the convolution optional. **Right**: Formal language evaluation (scaled accuracy, %). Higher is better. Models are trained on short sequences and evaluated on longer lengths to test length generalization. For Gated DeltaNet we report the variant with eigenvalue range $[-1, 1]$.

Model Variant (SISO)	ppl ↓
Mamba-3 − bias − trap	16.68
Mamba-3 − bias	16.49
Mamba-3	**15.72**
Mamba-3 + conv	15.85

(a) Component ablation (350M).

Model	Parity ↑	Arith. w/o ↑ brackets	Arith. w/ ↑ brackets
Mamba-3	100.00	98.51	87.75
Mamba-3 (w/o RoPE)	2.27	1.49	0.72
Mamba-3 (w/ Std. RoPE)	1.56	20.70	2.62
Mamba-2	0.90	47.81	0.88
Gated DeltaNet [-1,1]	100.00	99.25	93.50

(b) Performance comparison on formal language tasks. Results show that unlike Mamba-2, Mamba-3 features state tracking ability stemming from data-dependent RoPE embeddings. We used Mamba-3 (SISO) for these ablations.

4.3 SSM-Centric Methodological Ablations

Table 4a ablates the changes made to the core SSM component, mainly the introduction of BC bias and trapezoidal discretization. We report the pretraining test perplexity on models at the 440M scale, trained for Chinchilla optimal tokens. We find that the bias and trapezoidal SSM synergize well and make the short convolution utilized by many current linear models redundant.

We empirically demonstrate that data-dependent RoPE in Mamba-3 enables state tracking. Following Grazzi et al. (2025), we evaluate on tasks from the Chomsky hierarchy—Parity, Modular Arithmetic (without brackets), and Modular Arithmetic (with brackets)—and report scaled accuracies in Table 4b. Mamba-3 solves Parity and Modular Arithmetic (without brackets), and nearly closes the accuracy gap on Modular Arithmetic (with brackets). In contrast, Mamba-3 without RoPE, Mamba-3 with standard RoPE (Su et al., 2023), and Mamba-2 fail to learn these tasks. We use the state-tracking–enabled *Gated DeltaNet* variant of and observe that *Mamba-3* is competitive—matching parity and approaching its performance on both modular-arithmetic tasks. Experimental settings are covered in Appendix E.

5 Conclusion And Future Work

We introduce Mamba-3, an SSM model with three axes of improvement rooted in SSM principles: (i) *improved quality*, via trapezoidal discretization; (ii) *new capabilities*, through complex SSMs that recover state-tracking; and (iii) *higher inference efficiency*, with a MIMO formulation that raises arithmetic intensity. Mamba-3 delivers strong language modeling results and establishes a new Pareto frontier on the performance-efficiency axes with respect to strong baseline models. A limitation remains in retrieval, where fixed-state architectures lags attention-based models. We see **hybrid Mamba-3 architectures** that integrate retrieval mechanisms as a promising path, alongside broader application of our design principles to linear-time sequence models.

REFERENCES

Zeyuan Allen-Zhu. Physics of Language Models: Part 4.1, Architecture Design and the Magic of Canon Layers. *SSRN Electronic Journal*, May 2025. https://ssrn.com/abstract=5240330.

Aryaman Arora, Neil Rathi, Nikil Roashan Selvam, Róbert Csordás, Dan Jurafsky, and Christopher Potts. Mechanistic evaluation of transformers and state space models, 2025a. URL https://arxiv.org/abs/2505.15105.

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.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff, 2025b. URL https://arxiv.org/abs/2402.18668.

Aviv Bick, Kevin Y. Li, Eric P. Xing, J. Zico Kolter, and Albert Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models, 2025a. URL https://arxiv.org/abs/2408.10189.

Aviv Bick, Eric Xing, and Albert Gu. Understanding the skill gap in recurrent language models: The role of the gather-and-aggregate mechanism, 2025b. URL https://arxiv.org/abs/2504.18574.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2022. URL https://arxiv.org/abs/2009.14794.

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, 2018. URL https://arxiv.org/abs/1803.05457.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs, 2019. URL https://arxiv.org/abs/1903.00161.

Christopher Fleetwood. Domain specific architectures for ai inference. URL https://fleetwood.dev/posts/domain-specific-architectures.

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.

Madan Gopal. *Modern control system theory*. New Age International, 1993.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and et. al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Riccardo Grazzi, Julien Siems, Simon Schrodi, Thomas Brox, and Frank Hutter. Is mamba capable of in-context learning?, 2024. URL https://arxiv.org/abs/2402.03170.

Riccardo Grazzi, Julien Siems, Arber Zela, Jörg K. H. Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues, 2025. URL https://arxiv.org/abs/2411.12537.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022a. URL https://arxiv.org/abs/2111.00396.

Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *arXiv preprint arXiv:2206.11893*, 2022b. URL https://arxiv.org/abs/2206.11893.

Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces, 2022. URL https://arxiv.org/abs/2203.14343.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers, 2020. URL https://arxiv.org/abs/2010.04245.

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?, 2024. URL https://arxiv.org/abs/2404.06654.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying, 2024. URL https://arxiv.org/abs/2402.01032.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

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/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. Llm inference serving: Survey of recent advances and opportunities, 2024. URL https://arxiv.org/abs/2407.12391.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models, 2025. URL https://arxiv.org/abs/2404.08819.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL https://arxiv.org/abs/1809.02789.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, and et. al. 2 olmo 2 furious, 2025. URL https://arxiv.org/abs/2501.00656.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences, 2023. URL https://arxiv.org/abs/2303.06349.

Daniele Paliotta, Junxiong Wang, Matteo Pagliardini, Kevin Y. Li, Aviv Bick, J. Zico Kolter, Albert Gu, François Fleuret, and Tri Dao. Thinking slow, fast: Scaling inference compute with distilled reasoners, 2025. URL https://arxiv.org/abs/2502.20339.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. Can mamba learn how to learn? a comparative study on in-context learning tasks, 2024. URL https://arxiv.org/abs/2402.04248.

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, 2024. URL https://arxiv.org/abs/2406.17557.

Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. Rwkv-7 "goose" with expressive dynamic state evolution, 2025. URL https://arxiv.org/abs/2503.14456.

Pranav Rajpurkar, Jian Zhang, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *ACL 2018*, 2018.

Yuval Ran-Milo, Eden Lumbroso, Edo Cohen-Karlik, Raja Giryes, Amir Globerson, and Nadav Cohen. Provable benefits of complex parameterizations for structured state space models, 2024. URL https://arxiv.org/abs/2410.14067.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Yash Sarrof, Yana Veitsman, and Michael Hahn. The expressive capacity of state space models: A formal language perspective, 2024. URL https://arxiv.org/abs/2405.17394.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers, 2021. URL https://arxiv.org/abs/2102.11174.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Improving state-tracking in linear rnns via householder products, 2025. URL https://arxiv.org/abs/2502.10297.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling, 2023. URL https://arxiv.org/abs/2208.04933.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

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.

Endre Süli and David F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, and et. al. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

M. Tenenbaum and H. Pollard. *Ordinary Differential Equations: An Elementary Textbook for Students of Mathematics, Engineering, and the Sciences*. Dover Books on Mathematics. Dover Publications, 1985. ISBN 9780486649405. URL https://books.google.com/books?id=iU4zDAAAQBAJ.

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*, pp. 5998–6008, 2017. URL http://arxiv.org/abs/1706.03762.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie, Charlotte Frenkel, Razvan Pascanu, Blaise Agüera y Arcas, and João Sacramento. Mesanet: Sequence modeling by locally optimal test-time training, 2025. URL https://arxiv.org/abs/2506.05233.

Mitchell Wortsman, Peter J. Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D. Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, Jeffrey Pennington, Jascha Sohl-dickstein, Kelvin Xu, Jaehoon Lee, Justin Gilmer, and Simon Kornblith. Small-scale proxies for large-scale transformer training instabilities, 2023. URL https://arxiv.org/abs/2309.14322.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models, 2025. URL https://arxiv.org/abs/2408.00724.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2025a. URL https://arxiv.org/abs/2412.06464.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length, 2025b. URL https://arxiv.org/abs/2406.06484.

Annan Yu and N. Benjamin Erichson. Block-biased mamba for long-range sequence processing, 2025. URL https://arxiv.org/abs/2505.09022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

LLM Usage. We utilized Large Language Models to polish the writing in our submission as well as generate latex code for formatting tables and figures.

A Related Work

Linear-time sequence mixers. State-space models (SSMs) provide linear-time sequence mixing through explicit dynamical states and efficient scan/convolution implementations, offering significant computational advantages over quadratic-time attention mechanisms (Gu et al., 2022a; Smith et al., 2023; Gupta et al., 2022). Mamba-1 (Gu & Dao, 2024) introduced input-dependent selectivity to SSMs, while Mamba-2 (Dao & Gu, 2024) formalized the connection between SSMs and attention via structured state-space duality (SSD) (Katharopoulos et al., 2020; Choromanski et al., 2022). Despite matching transformers on standard language understanding benchmarks, these recurrent models exhibit limitations on tasks requiring precise algorithmic reasoning. Recent evaluations identified gaps in capabilities such as associative retrieval (Bick et al., 2025b; Arora et al., 2025a), exact copying (Jelassi et al., 2024), and in-context learning (Park et al., 2024; Grazzi et al., 2024). To address these limitations, DeltaNet enhances linear attention by replacing additive updates with delta-rule recurrence (Schlag et al., 2021), with recent work developing hardware-efficient, sequence-parallel training algorithms for this architecture (Yang et al., 2025b). This has catalyzed a broader effort to improve the algorithmic capabilities of linear-time models through architectural innovations including gating mechanisms, improved state transition dynamics, and hybrid approaches (Peng et al., 2025; Siems et al., 2025; Yang et al., 2025a; Paliotta et al., 2025; Bick et al., 2025a).

Expressivity and state tracking in recurrent mixers. Recent work characterizes the types of state that recurrent, constant-memory mixers can maintain, revealing algorithmic deficiencies in previous SSM-based models. Merrill et al. (2025) show that under finite precision, practical SSMs collapse to TC^0, leading to failures on tasks like permutation composition over S_5 unless the primitive is extended. Similarly, Yu & Erichson (2025) prove that a single-layer Mamba is not a universal approximator. Several modifications have been proposed to improve expressivity. For instance, the same work shows that a block-biased variant regains the universal approximation property with only minor changes, either through block decomposition or a channel-specific bias. Allowing negative eigenvalues or non-triangular transitions enables linear RNNs—including diagonal and Householder/DeltaNet forms—to capture parity and, under mild assumptions, regular languages (Grazzi et al., 2025). Complex-valued parameterizations provide another avenue for enhanced expressivity. Diagonal LTI SSMs demonstrate effectiveness for language modeling (Gu et al., 2022b; Orvieto et al., 2023), with complex variants achieving equivalent functions using smaller, well-conditioned parameters (Ran-Milo et al., 2024). However, the introduction of selectivity—the central innovation of modern SSMs (Gu & Dao, 2024)—narrowed the performance gap with Transformers by enabling input-dependent dynamics and achieving state-of-the-art results on language modeling benchmarks, leading practitioners to abandon complex states in favor of simpler real-valued architectures. We extend this line of work by reintroducing complex-valued state evolution that yields a real SSM with doubled dimensionality and block-diagonal rotations applied to the update rule—analogous through SSD (Dao & Gu, 2024) to how RoPE (Su et al., 2023) applies complex rotations to queries and keys in attention. The resulting data-dependent rotational structure expands stable dynamics to include oscillatory modes, enabling richer states while maintaining constant memory and linear-time complexity.

B Trapezoidal Discretization

Proposition 5 (Variation of Constants (Tenenbaum & Pollard, 1985))**.** *Consider the linear SSM*

$$\dot{\boldsymbol{h}}(t) = A(t)\,\boldsymbol{h}(t) + \mathbf{B}(t)\,x(t),$$

where $\boldsymbol{h}(t) \in \mathbb{R}^N$, $A(t) \in \mathbb{R}$ is a scalar decay, and $\mathbf{B}(t)x(t) \in \mathbb{R}^N$. For Δ_t discretized time grid $\tau_t = \tau_{t-1} + \Delta_t$, the hidden state satisfies

$$\boldsymbol{h}_t \;\approx\; e^{\Delta_t A_t}\,\boldsymbol{h}_{t-1} \;+\; \int_{\tau_{t-1}}^{\tau_t} e^{(\tau_t-\tau)A_t}\,\mathbf{B}(\tau)x(\tau)\,d\tau. \tag{10}$$

Proof. Since $A(t)$ is scalar, the homogeneous system $\dot{\boldsymbol{h}}(t) = A(t)\boldsymbol{h}(t)$ has solution

$$\boldsymbol{h}(t) = \phi(t,s)\,\boldsymbol{h}(s), \qquad \phi(t,s) = \exp\!\left(\int_s^t A(\xi)\,d\xi\right).$$

The Variation of Constants formula gives us,

$$\boldsymbol{h}(t) = \phi(t,s)\,\boldsymbol{h}(s) \;+\; \int_s^t \phi(t,\tau)\,\mathbf{B}(\tau)x(\tau)\,d\tau.$$

Setting $(s,t) = (t_{k-1}, t_k)$ yields the exact $\boldsymbol{h}_t$ given $\boldsymbol{h}_{t-1}$. We approximate $\int_s^t A(\xi)\,d\xi$ by setting $A(\tau) \approx A_k$ over $[t_{k-1}, t_k]$, which gives us,

$$\phi(t_k, t_{k-1}) = \exp\left(\int_s^t A(\xi)\,d\xi\right) \approx \exp\left(\int_s^t A_k\,d\xi\right) = e^{\Delta_k A_k},$$

Substituting these approximations in the Variation of Constants integral, we get the approximation

$$\boldsymbol{h}_t \;\approx\; e^{\Delta_t A_t}\,\boldsymbol{h}_{t-1} \;+\; \int_{\tau_{t-1}}^{\tau_t} e^{(\tau_t-\tau)A_t}\,\mathbf{B}(\tau)x(\tau)\,d\tau.$$

□

B.1 Trapezoid Discretization's Mask Matrix

Proof. When viewing the tensor contraction form, let us call $C = (T,N), B = (S,N), L = (T,S), X = (S,P)$ based on the Mamba-2 paper. With this decomposition of our mask, we can view $L = \text{contract}(TZ, ZS \to TS)(L_1, L_2)$.

The original contraction can be seen as

$$\text{contract}(TN, SN, TS, SP \to TP)(C, B, L, X)$$

We can now view it as

$$\text{contract}(TN, SN, TJ, JS, SP \to TP)(C, B, L_1, L_2, X)$$

This can be broken into the following:

$$\begin{aligned} Z &= \text{contract}(SN, SP \to SNP)(B, X) \\ Z' &= \text{contract}(JS, SNP \to JNP)(L_2, Z) \\ H &= \text{contract}(TJ, JNP \to TNP)(L_1, Z') \\ Y &= \text{contract}(TN, TNP \to TP)(C, H) \end{aligned}$$

Thus, we can view this step: $\text{contract}(ZS, SNP \to ZNP)(L_2, Z)$ as a conv of size two applied on Bx with the traditional SSD $L = L_1$ matrix. □

B.2 Trapezoidal Discretization Error Rate

Standard assumptions. We assume that: $A(t), \mathbf{B}(t), x(t)$ are bounded and C^2 on each timestep, so that $g(\tau)$ has two bounded derivatives; the map $\boldsymbol{h} \mapsto A(t)\boldsymbol{h} + \mathbf{B}(t)x(t)$ is Lipschitz in $\boldsymbol{h}$ which is true for linear systems; λ_t lies in a bounded interval so that the update is zero-stable.

Proof. Let $\boldsymbol{g}(\tau) \coloneqq e^{(t_k-\tau)A_k}\,\mathbf{B}(\tau)x(\tau)$ denote the integrand in the second term of Proposition 5. Since $A(t), \mathbf{B}(t), x(t)$ are C^2 on $[t_{k-1}, t_k]$, the function g has two bounded derivatives. A second-order Taylor expansion of g around t_{k-1} gives us,

$$\int_{t_{k-1}}^{t_k} g(\tau)\,d\tau = \Delta_t\,g(t_{k-1}) + \frac{\Delta_t^2}{2}\,g'(t_{k-1}) + \frac{\Delta_t^3}{6}\,g''(t_{k-1}) + O(\Delta_t^4).$$

Recall that the trapezoidal approximation to this integral is given by,

$$Q_\lambda = \Delta_t\Big[(1-\lambda_t)\,g(t_{k-1}) + \lambda_t\,g(t_k)\Big].$$

Expanding $g(t_k)$ using Taylor expansion: $g(t_k) = g(t_{k-1}) + \Delta_t g'(t_{k-1}) + \frac{\Delta_t^2}{2}g''(t_{k-1}) + O(\Delta_t^3)$. Substituting this into Q_λ,

$$\begin{aligned} Q_\lambda &= \Delta_t\Big[(1-\lambda_t)g(t_{k-1}) + \lambda_t g(t_k)\Big] \\ &= \Delta_t g(t_{k-1}) + \lambda_t\Delta_t^2 g'(t_{k-1}) + \lambda_t\frac{\Delta_t^3}{2}g''(t_{k-1}) + O(\Delta_t^4). \end{aligned}$$

Hence, the error is given by:

$$\int_{t_{k-1}}^{t_k} g(\tau)\,d\tau - Q_\lambda = \left(\tfrac{1}{2} - \lambda_t\right)\Delta_t^2 g'(t_{k-1}) + \left(\tfrac{1}{6} - \tfrac{\lambda_t}{2}\right)\Delta_t^3 g''(t_{k-1}) + O(\Delta_t^4).$$

Under the assumption that $\lambda_t = \frac{1}{2} + c_t\Delta_t$, where $c_t = O(1)$, then $\frac{1}{2} - \lambda_t = -c_t\Delta_t = O(\Delta_t)$ and thus the Δ_t^2 term is $O(\Delta_t^3)$. Therefore,

$$\int_{t_{k-1}}^{t_k} g(\tau)\,d\tau - Q_\lambda = O(\Delta_t^3),$$

which yields an $O(\Delta_t^3)$ local truncation error. Since the update $\boldsymbol{h}_k = e^{\Delta_t A_k}\boldsymbol{h}_{k-1} + Q_\lambda$ is linear and zero–stable for bounded λ_t, standard numerical ODE results imply an $O(\Delta_t^2)$ global error. □

B.3 Trapezoidal Parameterization

Parameterization	Form of λ_t	ppl ↓
Default	$\sigma(u_t)$	**15.72**
Fixed $1/2$	$\frac{1}{2}$	15.76
No trapezoid (Euler)	1	15.81

Table 5: **Ablations on λ_t parameterization in the trapezoidal update.**

Setting: All runs use the Mamba-3 (SISO) 440M model trained at Chinchilla scale, with the other architectural and optimization hyperparameters being the same as in Table 1.

The default model uses a data-dependent gate $\lambda_t = \sigma(u_t)$, where u_t is a learned projection of the current input token. In Table 5, we try different parameterizations for λ_t and find that the default parameterization empirically performs the best. Hence we choose the simpler default parameterization that does *not* enforce the $O(\frac{1}{2} + \Delta_t)$.

C Complex SSM Proofs

C.1 Proof of Proposition 2

Proposition 2 (Complex-to-Real SSM Equivalence)**.** *Consider a complex-valued SSM*

$$\begin{aligned}\dot{\boldsymbol{h}}(t) &= \mathrm{Diag}\big(A(t) + i\boldsymbol{\theta}(t)\big)\,\boldsymbol{h}(t) + \big(\mathbf{B}(t) + i\hat{\mathbf{B}}(t)\big)\,x(t),\\ y(t) &= \mathrm{Re}\Big(\big(\mathbf{C}(t) + i\hat{\mathbf{C}}(t)\big)^\top \boldsymbol{h}(t)\Big),\end{aligned} \tag{6}$$

where $\boldsymbol{h}(t) \in \mathbb{C}^{N/2}$, $\boldsymbol{\theta}(t), \mathbf{B}(t), \hat{\mathbf{B}}(t), \mathbf{C}(t), \hat{\mathbf{C}}(t) \in \mathbb{R}^{N/2}$, and $x(t), A(t) \in \mathbb{R}$. Under Euler discretization, this system is equivalent to a real-valued SSM

$$\begin{aligned}\boldsymbol{h}_t &= e^{\Delta_t A_t}\,\mathbf{R}_t\,\boldsymbol{h}_{t-1} + \Delta_t \mathbf{B}_t x_t,\\ y_t &= \mathbf{C}_t^\top \boldsymbol{h}_t,\end{aligned} \tag{7}$$

with state $\boldsymbol{h}_t \in \mathbb{R}^N$, projections

$$\mathbf{B}_t = \begin{bmatrix}\mathbf{B}_t\\ \hat{\mathbf{B}}_t\end{bmatrix} \in \mathbb{R}^N, \qquad \mathbf{C}_t = \begin{bmatrix}\mathbf{C}_t\\ -\hat{\mathbf{C}}_t\end{bmatrix} \in \mathbb{R}^N,$$

and a transition matrix

$$\mathbf{R}_t \;=\; \mathit{Block}\Big(\{R(\Delta_t\boldsymbol{\theta}_t[i])\}_{i=1}^{N/2}\Big) \in \mathbb{R}^{N\times N}, \qquad R(\Theta) = \begin{bmatrix}\cos(\Theta) & -\sin(\Theta)\\ \sin(\Theta) & \cos(\Theta)\end{bmatrix}.$$

Proof. We first present the derivation for $N = 2$; the block-diagonal structure for general even N follows by grouping pairs of coordinates.

Let $h_t + i\hat{h}_t$ denote the complexified hidden state, with parameters $A(t) + i\theta(t)$ and $B(t) + i\hat{B}(t)$ for the transition and input, respectively. By the variation of constants formula (Proposition 5), applying zero–order hold and Euler's rule over a step $[t_{k-1}, t_k]$ gives

$$h_k + i\hat{h}_k = e^{\Delta_t(A_t + i\theta_t)}(h_{k-1} + i\hat{h}_{k-1}) + \Delta_t(B_t + i\hat{B}_t)x_t.$$

Expanding the exponential,

$$e^{\Delta_t(A_t+i\theta_t)} = e^{\Delta_t A_t}\Big(\cos(\Delta_t\theta_t) + i\sin(\Delta_t\theta_t)\Big),$$

so in real coordinates $\boldsymbol{h}_t = \begin{bmatrix} h_t \\ \hat{h}_t \end{bmatrix} \in \mathbb{R}^2$ the recurrence becomes

$$\boldsymbol{h}_t = e^{\Delta_t A_t} \underbrace{\begin{bmatrix} \cos(\Delta_t\theta_t) & -\sin(\Delta_t\theta_t) \\ \sin(\Delta_t\theta_t) & \cos(\Delta_t\theta_t) \end{bmatrix}}_{R(\Delta_t\theta_t)} \boldsymbol{h}_{t-1} + \Delta_t \begin{bmatrix} B_t \\ \hat{B}_t \end{bmatrix} x_t.$$

Stacking across $N/2$ such pairs yields the block-diagonal transition

$$\boldsymbol{h}_t = e^{\Delta_t A_t}\,\mathrm{Block}\big(\{R(\Delta_t\theta_t[i])\}_{i=1}^{N/2}\big)\boldsymbol{h}_{t-1} + \Delta_t \begin{bmatrix} \mathbf{B}_t \\ \hat{\mathbf{B}}_t \end{bmatrix} x_t.$$

For the output,

$$y_t = \mathrm{Re}\Big((\mathbf{C}_t + i\hat{\mathbf{C}}_t)^\top (h_t + i\hat{h}_t)\Big) = \begin{bmatrix} \mathbf{C}_t \\ -\hat{\mathbf{C}}_t \end{bmatrix}^\top \boldsymbol{h}_t,$$

which defines the real projection $\mathbf{C}_t \in \mathbb{R}^N$ in the proposition. This proves the equivalence between complex SSM and the real block-diagonal system with rotations. □

C.2 Proof of Proposition 3

Proposition 3 (Complex SSM, Data-Dependent RoPE Equivalence)**.** *Under the notation established in Proposition 2, consider the real SSM defined in Eq. 7 unrolled for T time-steps. The output of the above SSM is equivalent to that of a vanilla scalar transition matrix-based SSM (Eq. 2) with a data-dependent rotary embedding applied on the $\mathbf{B}, \mathbf{C}$ components of the SSM defined as:*

$$\boldsymbol{h}_t = e^{\Delta_t A_t}\boldsymbol{h}_{t-1} + (\prod_{i=0}^{t}\mathbf{R}_i^\top)\mathbf{B}_t x_t, \qquad \boldsymbol{y}_t = \Big((\prod_{i=0}^{t}\mathbf{R}_i^\top)\mathbf{C}_t\Big)^\top \boldsymbol{h}_t \tag{8}$$

where the matrix production represents right matrix multiplication, e.g., $\prod_{i=0}^{1}\mathbf{R}_i = \mathbf{R}_0\mathbf{R}_1$. We denote employing the vanilla SSM to compute the Complex SSM as "RoPE trick".

Proof. Consider the SSM

$$\boldsymbol{h}_t \;=\; e^{\Delta_t A_t}\,\mathbf{R}_t\,\boldsymbol{h}_{t-1} \;+\; \mathbf{B}_t x_t, \qquad \boldsymbol{y}_t \;=\; \mathbf{C}_t^\top \boldsymbol{h}_t, \tag{11}$$

where (as in Proposition 3) $A_t \in \mathbb{R}$ is a scalar (so that $e^{\Delta_t A_t}$ is a scalar and commutes with rotations), and $\mathbf{R}_t$ is block-diagonal orthogonal/unitary, hence $\mathbf{R}_t^{-1} = \mathbf{R}_t^\top$.

Unrolling the recurrence with the convention that an empty product is the identity,

$$\boldsymbol{h}_t \;=\; \sum_{i=0}^{t}\Big(\prod_{s=i+1}^{t} e^{\Delta_s A_s}\mathbf{R}_s\Big)\,\mathbf{B}_i x_i. \tag{12}$$

Thus

$$\boldsymbol{y}_t \;=\; \mathbf{C}_t^\top \boldsymbol{h}_t \;=\; \sum_{i=0}^{t}\mathbf{C}_t^\top\Big(\prod_{s=i+1}^{t} e^{\Delta_s A_s}\mathbf{R}_s\Big)\mathbf{B}_i x_i. \tag{13}$$

Using unitarity property,

$$\prod_{s=i+1}^{t}\mathbf{R}_s = (\prod_{s=0}^{t}\mathbf{R}_s)(\prod_{s=0}^{i}\mathbf{R}_s)^{-1} = (\prod_{s=0}^{t}\mathbf{R}_s)(\prod_{s=0}^{i}\mathbf{R}_s^\top).$$

Since $e^{\Delta_s A_s}$ are scalars, they commute with rotations; hence

$$\boldsymbol{y}_t = \sum_{i=0}^{t} \mathbf{C}_t^\top \Big(\prod_{s=0}^{t} \mathbf{R}_s \Big) \Big(\prod_{s=i+1}^{t} e^{\Delta_s A_s} \Big) \Big(\prod_{s=0}^{i} \mathbf{R}_s^\top \Big) \mathbf{B}_i x_i \tag{14}$$

$$= \Big((\prod_{s=0}^{t} \mathbf{R}_s^\top) \mathbf{C}_t \Big)^\top \sum_{i=0}^{t} \Big(\prod_{s=i+1}^{t} e^{\Delta_s A_s} \Big) \Big(\prod_{s=0}^{i} \mathbf{R}_s^\top \Big) \mathbf{B}_i x_i. \tag{15}$$

Define the rotated parameters $\bar{\mathbf{C}}_t := \big(\prod_{s=0}^{t} \mathbf{R}_s^\top\big)\mathbf{C}_t$ and $\bar{\mathbf{B}}_i := \big(\prod_{s=0}^{i} \mathbf{R}_s^\top\big)\mathbf{B}_i$. Then

$$\boldsymbol{y}_t \;=\; \bar{\mathbf{C}}_t^\top \sum_{i=0}^{t} \Big(\prod_{s=i+1}^{t} e^{\Delta_s A_s} \Big) \bar{\mathbf{B}}_i x_i. \tag{16}$$

Equivalently, introducing the rotated state $\tilde{\boldsymbol{h}}_t := \big(\prod_{s=0}^{t} \mathbf{R}_s^\top\big)\boldsymbol{h}_t$,

$$\tilde{\boldsymbol{h}}_t \;=\; e^{\Delta_t A_t}\, \tilde{\boldsymbol{h}}_{t-1} \;+\; \bar{\mathbf{B}}_t x_t, \qquad \boldsymbol{y}_t \;=\; \bar{\mathbf{C}}_t^\top \tilde{\boldsymbol{h}}_t, \tag{17}$$

□

C.3 Proof of Proposition 4

Proposition 4 (Rotary Embedding Equivalence with Trapezoidal Discretization)**.** *Discretizing a complex SSM with the trapezoidal rule (Proposition 1) yields the recurrence*

$$\boldsymbol{h}_t = \alpha_t \boldsymbol{h}_{t-1} + \beta_t \left(\prod_{i=0}^{t-1} \mathbf{R}_i^\top \right) \mathbf{B}_{t-1} x_{t-1} + \gamma_t \left(\prod_{i=0}^{t} \mathbf{R}_i^\top \right) \mathbf{B}_t x_t,$$

$$\boldsymbol{y}_t = \left((\prod_{i=0}^{t} \mathbf{R}_i^\top) \mathbf{C}_t \right)^\top \boldsymbol{h}_t. \tag{9}$$

Here $\mathbf{R}_t$ is the block-diagonal rotation matrix defined in Proposition 3.

Proof. We begin from the complex SSM (as in Prop. 2)

$$\dot{\boldsymbol{h}}(t) = \mathrm{Diag}\big(A(t) + i\boldsymbol{\theta}(t)\big)\, \boldsymbol{h}(t) \;+\; \big(\mathbf{B}(t) + i\hat{\mathbf{B}}(t)\big)\, x(t),$$
$$y(t) = \mathrm{Re}\big((\mathbf{C}(t) + i\hat{\mathbf{C}}(t))^\top \boldsymbol{h}(t)\big),$$

where $A(t) \in \mathbb{R}$ is a scalar and $\boldsymbol{\theta}(t), \mathbf{B}(t), \hat{\mathbf{B}}(t), \mathbf{C}(t), \hat{\mathbf{C}}(t) \in \mathbb{R}^{N/2}$.

Recall from Prop. 5,

$$\boldsymbol{h}_t \approx e^{\Delta_t (A_t + i\boldsymbol{\theta}_t)}\, \boldsymbol{h}_{t-1} \;+\; \int_{\tau_{t-1}}^{\tau_t} e^{(\tau_t - \tau)(A_t + i\boldsymbol{\theta}_t)} \big(\mathbf{B}(\tau) + i\hat{\mathbf{B}}(\tau)\big)\, x(\tau)\, d\tau.$$

Applying Prop. 1 to the above integral, we get

$$\boldsymbol{h}_t = e^{\Delta_t (A_t + i\boldsymbol{\theta}_t)}\, \boldsymbol{h}_{t-1} \;+\; \beta_t\, e^{i\Delta_t \boldsymbol{\theta}_t} \big(\mathbf{B}_{t-1} + i\hat{\mathbf{B}}_{t-1}\big) x_{t-1} \;+\; \gamma_t \big(\mathbf{B}_t + i\hat{\mathbf{B}}_t\big) x_t, \tag{18}$$

wherem

$$\alpha_t := e^{\Delta_t A_t}, \qquad \beta_t := (1 - \lambda_t)\Delta_t e^{\Delta_t A_t}, \qquad \gamma_t := \lambda_t \Delta_t,$$

Since $e^{\Delta_t (A_t + i\boldsymbol{\theta}_t)} = \alpha_t\, e^{i\Delta_t \boldsymbol{\theta}_t}$ and as shown in Prop. 2, multiplication by $e^{i\Delta_t \boldsymbol{\theta}_t}$ is a block-diagonal rotation in real coordinates, we get the real N-dimensional recurrence

$$\boldsymbol{h}_t = \alpha_t\, \mathbf{R}_t\, \boldsymbol{h}_{t-1} \;+\; \beta_t\, \mathbf{R}_t\, \mathbf{B}_{t-1}\, x_{t-1} \;+\; \gamma_t\, \mathbf{B}_t\, x_t, \tag{19}$$
$$y_t = \mathbf{C}_t^\top \boldsymbol{h}_t,$$

where $\mathbf{R}_t = \mathrm{Block}\big(\{R(\Delta_t \boldsymbol{\theta}_t[i])\}_{i=1}^{N/2}\big)$ where $R(\Theta) = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix}$, and projections $\mathbf{B}_t = \begin{bmatrix} \mathbf{B}_t \\ \hat{\mathbf{B}}_t \end{bmatrix}$, $\mathbf{C}_t = \begin{bmatrix} \mathbf{C}_t \\ -\hat{\mathbf{C}}_t \end{bmatrix}$. Note that $\mathbf{R}_t$ is orthogonal, so $\mathbf{R}_t^{-1} = \mathbf{R}_t^\top$.

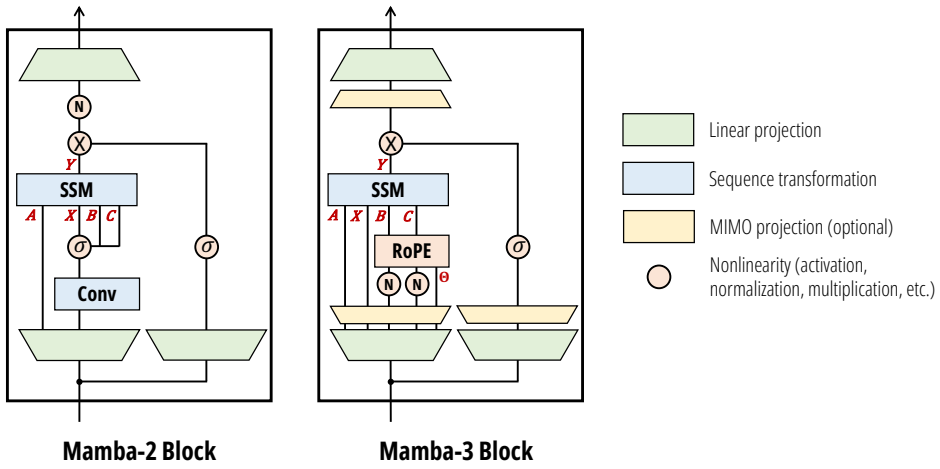

Figure 4: Contrasting Mamba-2 and Mamba-3 Architectures: Key updates include trapezoidal discretization, data-dependent RoPE embeddings, MIMO projections, QK normalization, and learnable biases.

We define the following,

$$\tilde{\boldsymbol{h}}_t := \Big(\prod_{s=0}^{t} \mathbf{R}_s^\top\Big)\boldsymbol{h}_t, \qquad \bar{\mathbf{B}}_t := \Big(\prod_{s=0}^{t} \mathbf{R}_s^\top\Big)\mathbf{B}_t, \qquad \bar{\mathbf{C}}_t := \Big(\prod_{s=0}^{t} \mathbf{R}_s^\top\Big)\mathbf{C}_t.$$

Left-multiplying equation 19 by $\prod_{s=0}^{t} \mathbf{R}_s^\top$ and using $\mathbf{R}_t^\top \mathbf{R}_t = I$,

$$\begin{aligned}\tilde{\boldsymbol{h}}_t &= \alpha_t\, \tilde{\boldsymbol{h}}_{t-1} \;+\; \beta_t\, \bar{\mathbf{B}}_{t-1}\, x_{t-1} \;+\; \gamma_t\, \bar{\mathbf{B}}_t\, x_t,\\ y_t &= \bar{\mathbf{C}}_t^\top\, \tilde{\boldsymbol{h}}_t.\end{aligned}$$

This is a vanilla scalar-transition SSM with data-dependent rotary embeddings absorbed into $\mathbf{B}, \mathbf{C}$ via cumulative products of $\mathbf{R}_s^\top$. □

D MIMO FOR MAMBA-3

With hindsight from Mamba and with inference in mind, we propose the following MIMO formulation:

Mamba with MIMO. With a given batch, head, and sequence position t, consider the input $\mathbf{U}_t \in \mathbb{R}^D$. Also denote $P, R \in \mathbb{N}$ as the head dimension and MIMO rank, respectively. We first obtain SSM parameters via a set of projections defined in terms of tensor contraction notation as follows:

$$\begin{aligned}\mathbf{B}_t &= \text{contract}(DNR, D \to NR)(\mathbf{W}_\mathbf{B}, \mathbf{U}_t) & \mathbf{C}_t &= \text{contract}(DNR, D \to NR)(\mathbf{W}_\mathbf{C}, \mathbf{U}_t),\\ \mathbf{X}'_t &= \text{contract}(PD, D \to P)(\mathbf{W}_{\mathbf{X}'}, \mathbf{U}_t) & \mathbf{X}_t &= \text{contract}(PR, P \to PR)(\mathbf{W}_\mathbf{X}, \mathbf{X}'_t),\end{aligned}$$

where $\mathbf{W}_\mathbf{B}, \mathbf{W}_\mathbf{C}, \mathbf{W}_{\mathbf{X}'}, \mathbf{W}_\mathbf{X}$ are model parameters. Additionally, we obtain the residual term $\mathbf{Z}_t$ in the same manner as $\mathbf{X}_t$ with weights $\mathbf{W}_{\mathbf{Z}'}$ and $\mathbf{W}_\mathbf{Z}$. The state update and the SSM output is then computed via the following MIMO SSM:

$$\mathbf{H}_t \;=\; a_t\, \mathbf{H}_{t-1} \;+\; \mathbf{B}_t \mathbf{X}_t^\top \in \mathbb{R}^{N \times P}, \qquad \mathbf{Y}_t \;=\; \mathbf{H}_t^\top \mathbf{C}_t \in \mathbb{R}^{P \times R}.$$

The intermediate output $\mathbf{Y}'_t$ is obtained via some residual function ϕ, $\mathbf{Y}'_t \leftarrow \phi(\mathbf{Y}_t, \mathbf{Z}_t)$. Finally, the layer output $\mathbf{O}_t \in \mathbb{R}^D$ is computed via the following down projections:

$$\mathbf{O}'_t = \text{contract}(PR, R \to P)(\mathbf{W}_{\mathbf{O}'}, \mathbf{Y}'_t) \qquad \mathbf{O}_t = \text{contract}(P, PD \to D)(\mathbf{W}_\mathbf{O}, \mathbf{O}'_t).$$

This formulation enhances the existing Mamba3 architecture by providing a lightweight parameterization that transforms the set of independent SISO SSMs within each head into a set of MIMO SSMs. Here, we note that the hardware-efficient chunking technique employed by Mamba2 for pretraining can be applied with little change, as the MIMO dimension r is orthogonal to the sequence dimension.

E Experimental Details

Language Modeling. Our pretraining procedures follow that of Dao & Gu (2024)'s section D.2. All models at each scale follow the same procedure and were trained with bfloat16. The Mamba family of models were trained using the standard expand factor of 2 and a dstate of 128 and head dimension of 64. The Transformer baselines follows Dao & Gu (2024), and the Gated DeltaNet baselines follow (Yang et al., 2025a). We utilize the Llama-3.1 tokenizer (Grattafiori et al., 2024) for all models.

We utilize LM Evaluation Harness (Gao et al., 2024) to test the zero-shot languag modeling capabilities of our pretrained model on LAMBADA (OpenAI version) (Paperno et al., 2016), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2019), Arc-Easy/Arc-Challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2019), and OpenBookQA(Mihaylov et al., 2018).

Real-World and Synthetic Retrieval. For our real-world retrieval tasks, we evaluate on the common suite consisting of SWDE (Arora et al., 2025b), SQUAD (Rajpurkar et al., 2018), FDA (Arora et al., 2025b), TriviaQA (Joshi et al., 2017), NQ (Kwiatkowski et al., 2019), and DROP (Dua et al., 2019). We utilize the cloze-formatted version of the aforementioned tasks provided by Arora et al. (2025b; 2024), as the original datasets are in a question-answering format, making it challenge for solely pretrained models. All tasks were truncated to match the training context length. The synthetic NIAH tasks (Hsieh et al., 2024) were also run with LM Evaluation Harness.

State-Tracking Synthetics. Training follows a sequence length curriculum that progresses from 3 -40 to 160, evaluated at 256. Each curriculum runs for 10^4 steps with batch size 256. We use 1 layer models for Parity and 3 layer models for Modular-arithmetic tasks. The state size is chosen to be 64, and we sweep $d_{\text{model}} \in \{32, 64\}$ and 8 learning rates logarithmically spaced between 10^{-4} and 10^{-2}, reporting the best validation accuracy.

F Additional Experimental Results

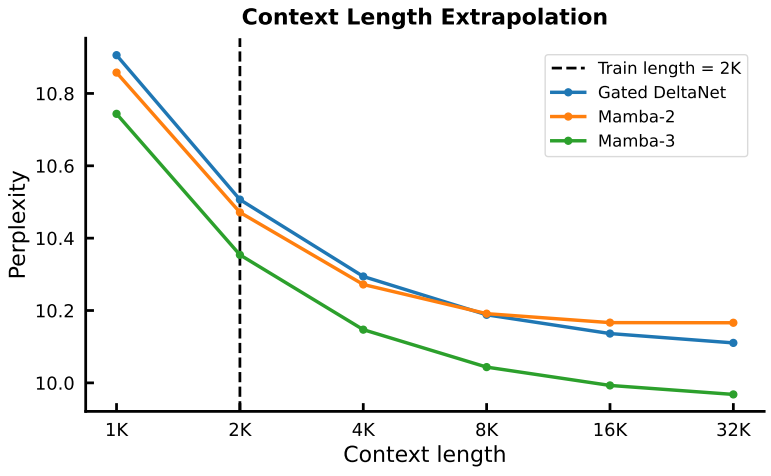

Figure 5: Pretrained 1.5B models' performance on the held-out FineWeb-Edu test set at varying context lengths. Mamba-3 exhibits strong length extrapolation while Mamba-2 falters at longer contexts.

Table 6: Downstream language modeling evaluations on parameter-matched pretrained models, including Mamba-3 MIMO. Mamba-3 MIMO's average accuracy on all tasks is more than 1 percentage point better than the next best (Mamba-3 SISO).

Model	FW-Edu ppl ↓	LAMB. ppl ↓	LAMB. acc ↑	HellaS. acc_n ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc_n ↑	WinoGr. acc ↑	OBQA acc ↑	Average acc ↑
Transformer-440M	13.03	21.2	41.7	50.5	69.9	67.6	34.6	**56.7**	26.0	49.6
Gated DeltaNet-440M	13.12	19.0	40.4	50.5	70.5	67.5	34.0	55.3	25.8	49.1
Mamba-2-440M	13.00	19.6	40.8	51.7	70.6	68.8	35.0	54.1	26.0	49.6
Mamba-3-440M	12.87	19.6	40.2	51.7	**71.9**	68.9	34.4	55.8	26.0	49.8
Mamba-3-MIMO-440M	**12.72**	**17.1**	**43.4**	**52.8**	70.8	**69.6**	**35.6**	56.3	**28.4**	**51.0**
Transformer-880M	11.42	15.0	44.7	57.2	72.6	71.6	39.2	57.7	26.8	52.8
Gated DeltaNet-880M	11.39	12.7	47.1	57.5	72.6	72.5	38.8	57.9	**30.6**	53.9
Mamba-2-880M	11.35	13.8	45.0	58.1	72.5	72.3	38.7	56.8	30.2	53.4
Mamba-3-880M	11.23	12.9	47.2	58.8	73.6	72.7	40.2	58.4	30.0	54.4
Mamba-3-MIMO-880M	**11.11**	**11.8**	**49.5**	**59.2**	**73.7**	**74.7**	**41.2**	**59.9**	28.6	**55.3**

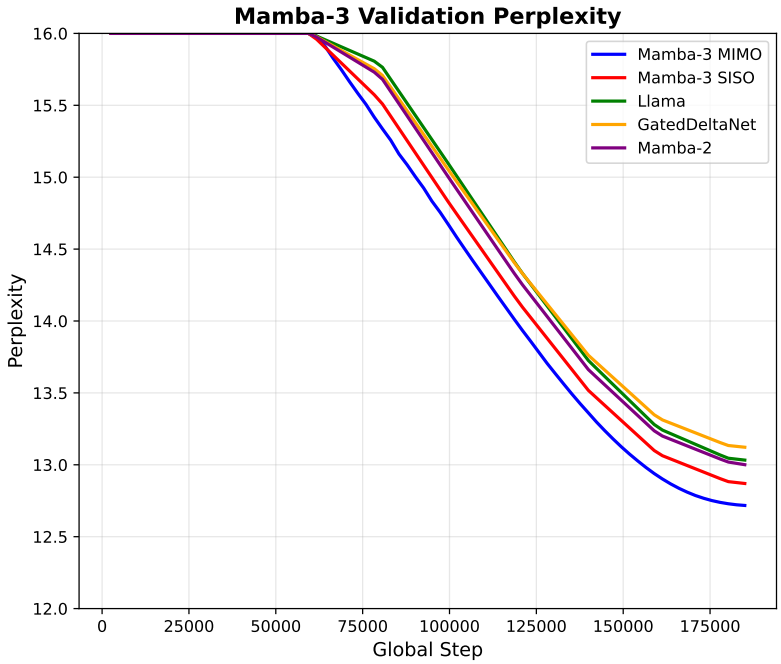

Figure 6: Mamba-3 demonstrates superior performance compared to strong baselines like Mamba-2, Llama, and Gated Deltanet. These are 440M models, trained and evaluated on FineWeb-Edu.

We also compare the effectiveness of state size usage of Mamba variants to a Gated DeltaNet baseline in Figure 7. We highlight the difficulty of directly comparing GDN versus Mamba-style models due to the differing head structure, multi-head compared to multi-value respectively. Our experiments hold GDN's v_expand to 2 and decrease the head dimension accordingly to vary the relative total state size. Similar to Figure 3, we train 440M models to 2× Chinchilla tokens and sweep across $d_{\text{state}} = \{32, 64, 128\}$ for the Mamba models and $d_{\text{head dim}} = \{32, 64, 128\}$ for GDN. We parameter match all models.

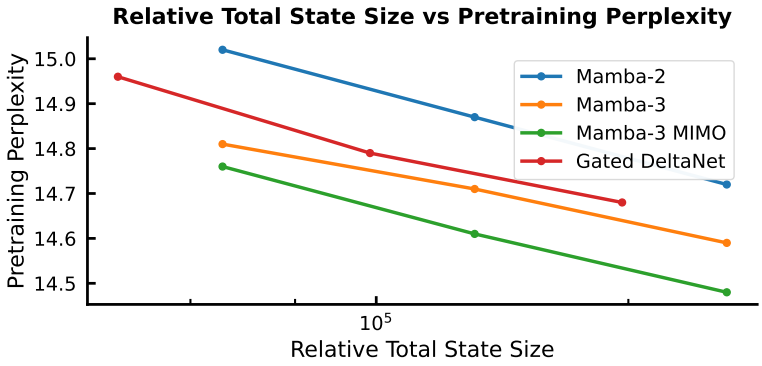

Figure 7: Exploration of state size (inference speed proxy) versus pretraining perplexity (performance proxy). Mamba-3 and Mamba-3 MIMO continue set the Pareto frontier.

G Architecture Ablations

We explore our model architecture's ablation in this section. All models are trained at the 440M scale to Chinchilla optimal number of tokens (20× tokens to parameters) with the same experimental procedures as our pretrained models as covered in Appendix E unless otherwise stated.

B,C Bias Parameterization. The Mamba-3 model's separate B and C biases are head-specific and channel-wise and added to both $\mathbf{B}$ and $\mathbf{C}$ after the QK-Norm. While the biases in the final Mamba-3 model are trainable, data-independent parameters and initialized to all ones, we explore various bias parameterizations in Table 7a. We find our models are not very sensitive to the initialization of the biases as long as they are positive. We choose the all-ones initialization due to it's simplicity.

We also explore the impact removing the B or C bias on performance in Table 7b (bias is initialized with our default parameterization when utilized). Unlike in Yu & Erichson (2025), which finds that B bias by itself is able to improve performance on Mamba-1, our experiments find that only having B bias hurts performance slightly and that B and C biases have synergetic properties.

Bias Init.	Trainable	ppl ↓
1.0	✓	15.72
0.0	✓	16.57
1.0	×	15.80
$\mathcal{U}(0,1)$	✓	15.76
$\mathcal{U}(-1,1)$	✓	16.07

(a) Effect of parameterization of the B and C bias on model performance, measured by pretraining perplexity. We find our default initialization of all-ones (first row) provides the best performance, but performance is not sensitive as long as biases are positive.

B Bias	C Bias	ppl ↓
×	×	16.52
✓	×	16.68
×	✓	15.98
✓	✓	15.69

(b) Applying a bias to both B and C leads to the best performance. Only applying B bias (Block-Biased (Yu & Erichson, 2025) Mamba-3 variant) does not provide significant gains over the no-bias baseline.

Table 7: Ablations on B, C bias initialization (left) and presence (right) for Mamba-3.

H Inference Kernel Latency Analysis

H.1 Kernel Implementations and Fusion Structure

In Table 3, we detail the DSL (Triton, CuTe, PyTorch) and the fusion level of the kernels used in our latency analysis. For Mamba-2 and Gated DeltaNet (GDN), we directly use the publicly released Triton kernels from the respective authors. For Mamba-3, we implement new inference kernels with a comparable fusion structure: the forward uses a Triton kernel fused with rotary position embeddings, while the decode path uses a CuTe kernel fused with gating and MIMO projection.

In Tables 8 and 9, we abbreviate IP = input projection, Conv = 1D convolution, Gate = gating, OP = output projection. Colors indicate implementation backend (Torch, Triton, CuTe).

Table 8: Kernel DSL and fusion structure for **forward** (prefill) kernels.

Model (Forward)	Kernel DSL	Fusion Level
Mamba-2	Triton	IP, Conv, SSM, Gate+OP
Gated DeltaNet	Triton	IP, Conv, Chunked Delta, Gate+OP
Mamba-3 (SISO)	Triton	IP, SSM+Rotary, Gate+OP
Mamba-3 (MIMO)	Triton	IP, SSM+Rotary, Gate+OP

Table 9: Kernel DSL and fusion structure for **decode** kernels.

Model (Decode)	Kernel DSL	Fusion Level
Mamba-2	Triton	IP, Conv, SSM, Gate+OP
Gated DeltaNet	Triton	IP, Conv, Recurrent Delta, Gate+OP
Mamba-3 (SISO)	CuTe + Triton	IP, Rotary, SSM+Gate+OP
Mamba-3 (MIMO)	CuTe + Triton	IP, Rotary, SSM+Gate+OP+MIMO

H.2 Extended Prefill and Prefill+Decode Latency Measurements

Models. We benchmark Mamba-3 1.5B (SISO), Mamba-2 1.5B, Gated DeltaNet (GDN) 1.5B, and a strong Transformer baseline implemented via the vLLM engine (v0.11.0) with Llama-3.2 1B.[3] All recurrent models are trained at the 1.5B scale with $d_{\text{model}} = 2048$ and 24 layers. For Mamba variants we set state size as 128 and head dimension 64; for GDN we use QK head dimension as 128.

Setting. Sequence lengths were swept over $L \in \{512, 1024, 2048, 4096, 16384\}$ for prefill, with an equal number of tokens decoded. For sequence lengths $\{512, 1024, 2048, 4096\}$, we use batch size of 128; for sequence lengths $\{16384\}$, we use batch size of 16. We use a single H100-SXM 80GB GPU and report wall-clock times (in seconds) over 3 repetitions.

Table 10: Prefill and Prefill+Decode latency across sequence lengths.

Model	512 tokens		1024 tokens		2048 tokens		4096 tokens		16384 tokens	
	Prefill	Prefill+Dec	Prefill	Prefill+Dec	Prefill	Prefill+Dec	Prefill	Prefill+Dec	Prefill	Prefill+Dec
vLLM (Llama-3.2-1B)	**0.26**	4.45	**0.52**	9.60	**1.08**	20.37	**2.08**	58.64	**1.52**	122.06
Gated DeltaNet	0.48	4.52	0.95	9.04	1.90	18.07	3.79	36.14	1.91	71.66
Mamba-2	0.48	4.62	0.96	9.24	1.91	18.48	3.81	36.94	1.92	57.90
Mamba-3 (SISO)	0.48	**4.33**	0.95	**8.64**	1.90	**17.29**	3.80	**34.57**	1.91	**53.97**

We observe that (i) Mamba-3 adds minimal forward-pass cost showing that the trapezoidal update, complex state tracking, and MIMO parameterization remain lightweight; (ii) decode latency is competitive across recurrent models; and (iii) recurrent mixers scale more gently with context length than vLLM Llama-3.2-1B, which grows much faster with L due to KV-cache overhead.

[3] https://huggingface.co/meta-llama/Llama-3.2-1B