MULTI-MIXER MODELS: FLEXIBLE SEQUENCE MODELING WITH SHARED REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Softmax attention is the cornerstone of modern large language models. However, its memory requirements scale linearly with sequence length, and its compute requirements scale quadratically. Linear recurrent models, such as linear attention and state space models, have become widely studied as alternatives to softmax attention due to their linear compute and constant memory requirements. While these sub-quadratic token mixing methods (mixers) achieve promising efficiency gains and competitive results on a wide range of benchmarks, current linear recurrent models still lag behind on tasks that require long-context retrieval or in-context learning. A growing body of work studies hybrid architectures that attempt to mitigate these trade-offs by statically interleaving or merging attention and recurrent blocks. In this work, we explore a new axis of developing hybrid models: across the token sequence. We propose ORYX, a hybrid model that can flexibly switch between different mixers throughout a sequence, e.g., quadratic attention for rich context utilization and linear recurrences to reduce quadratic computation. ORYX ties at least 90% of its parameters across mixers, enabling attention and recurrent modes to operate over shared internal representations. We validate our design with Mamba-2 and Gated DeltaNet variants, up to 1.4B models. Under fixed token budgets and a mixed-training strategy, ORYX achieves comparable or better performance than its single-mixer baselines. At the 1.4B scale, all instances of ORYX outperform their respective baselines by at least **0.7** percentage points on averaged language modeling tasks. On real-world retrieval tasks, ORYX remains comparable to the Transformer baseline even when processing less than 10% of the tokens in the attention mode. These results demonstrate that attention and linear recurrent models can share internal representations, and motivate sequence-axis hybridization as a promising direction.

1 INTRODUCTION

The Transformer architecture and its core softmax attention mechanism remain the dominant foundation for modern large language models (LLMs). However, softmax attention maintains a key-value (KV) cache of all previous tokens and queries the full cache at each step, incurring quadratic compute and linear memory requirements in sequence length. These costs have motivated the recent proliferation of sub-quadratic alternatives, in particular, linear recurrent models, such as linear attention (Katharopoulos et al., 2020; Yang et al., 2025a) and state-space models (SSMs) (Gu & Dao, 2024; Dao & Gu, 2024).[1] These linear models are characterized by their constant-size recurrent state, which is updated after each token, and linearly-scaling compute. While they have demonstrated promising results in many settings, pure recurrent approaches still lag behind on tasks that require strong retrieval or in-context learning abilities (Waleffe et al., 2024; Arora et al., 2025a). Such trade-offs have motivated the development and deployment of hybrid architectures, which combine these linear layers with softmax attention to balance performance and efficiency (Waleffe et al., 2024; Kimi Team et al., 2025; NVIDIA et al., 2025; Qwen Team, 2025).

Existing hybrid models fall under two main paradigms: inter-layer designs, which interleave softmax attention and linear layers, and intra-layer designs, which fuse the output of attention and linear mechanisms within a single layer or block. For both, the computational cost per token and capabilities are largely defined by the predetermined architecture. In practice, the required modeling

[1] In this work, we use attention to refer to the canonical quadratic softmax attention mechanism, and use linear to refer to the general class of linear recurrent mechanisms.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

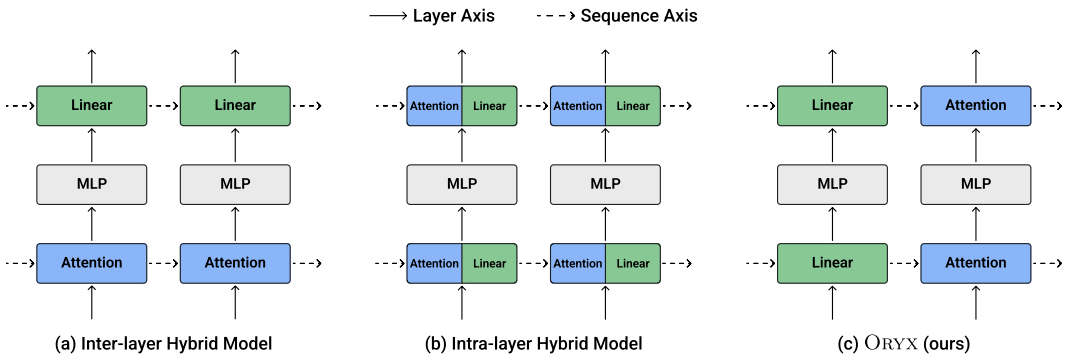


Figure 1: Comparison of different hybrid architectures. (a) Inter-layer hybrid models interleave different mixers along the layer axis; (b) Intra-layer hybrid models fuse different mixers within a single layer; (c) ORYX is a sequence-axis hybrid model that can switch between different mixers across the sequence, allowing different segments of the input to be processed by varying mechanisms.

capabilities and desired trade-off vary by task: retrieval tasks may benefit from richer attention-based context utilization, whereas standard language generation tasks may be adequately served by linear computation. This suggests a complementary form of hybridization: instead of choosing a fixed mixture of mechanisms, *can a model operate with different mixers throughout the sequence?*

In this work, we propose ORYX, a sequence-axis hybrid architecture that supports operating in *both* quadratic-attention and linear-recurrent regimes throughout a sequence (Figure 1), trained by a chunked mixed-mode strategy that enables flexible switching during deployment (§ 3). A key challenge in supporting switching is that attention and recurrent mixers parameterize and update their state differently. To bridge these mechanisms, ORYX layers maintain both a KV cache and a linear recurrent state, updated jointly at each timestep with key-value pairs obtained from shared weights across mixer types. Thus, both mechanisms update states from a shared representation space, avoiding separate state-update features for each mixer. Consequently, when the mechanism switches, the new selected mixer can continue from a compatible state accumulated over all preceding tokens. This selection flexibility during inference could enable compute allocation at the prompt or token level. For example, intermediate generation could use the linear mixer to reduce computation costs, while attention could be reserved for segments requiring stronger retrieval or context utilization.

We instantiate this design with Mamba-2 (Dao & Gu, 2024) and Gated DeltaNet (GDN) (Yang et al., 2025a) as the linear mixer, with both variants *sharing more than* 90% *of parameters across mixer modes*. Although our experimental validation focuses on Mamba-2 and GDN, we underscore that our overall design is not specific to these choices and can extend to other mixers that admit comparable key-value associations. Our multi-mixer models can switch between attention and linear sequence processing while incurring little to no perplexity degradation (Figure 3), and *under matched parameters and training token budgets*, ORYX remains competitive with pure softmax attention, Mamba-2, and GDN baselines on language modeling performance across scale. At the 1.4B scale, both attention and linear modes of Mamba-2 and GDN variants of ORYX outperform their respective baselines by *more than* 0.7 percentage points on average across downstream language modeling evaluations (Table 1). On a variety of real-world retrieval tasks and synthetic retrieval tasks, ORYX is able to achieve comparable performance compared to the Transformer baseline with $<10\%$ of the tokens processed in the attention mode. Using this mixed-inference mode, ORYX *substantially exceeds linear baselines* by a margin of at least 8.6 percentage points on real-world retrieval tasks and 38.6 percentage points on needle-in-a-haystack (NIAH) tests (Table 3), demonstrating that ORYX is a promising method to improve the retrieval capabilities of linear models. Our findings suggest that *attention and linear recurrent mechanisms can share similar representations* despite differing largely in methodology, opening new opportunities to study their interplay.

We first introduce preliminary background on sequence mixers and their connections in Section 2. We use their commonality to motivate the design of the multi-mixer ORYX block in Section 3 and highlight the main architectural components. Section 4 evaluates the language modeling and mode-switching ability of the ORYX model and ablates core design choices.

2 PRELIMINARIES

Notation. We use the term **mixer** to denote a transformation that mixes information along a particular axis of the input. A **sequence mixer** (e.g., softmax attention) mixes information across the sequence dimension, whereas a **channel mixer** (e.g., an MLP) mixes across the feature or channel dimension. We focus on sequence mixer designs in this paper and use the term mixer to refer to sequence mixer unless otherwise specified. We denote scalars by non-bold symbols (e.g., a), vectors by bold lowercase symbols (e.g., $\boldsymbol{q}$), and matrices by bold uppercase symbols (e.g., $\boldsymbol{X}, \boldsymbol{W}$). Subscripts index the first axis by default, while superscripts are reserved for identifiers.

Shared Key-Value Association View. While softmax attention (Vaswani et al., 2017) and modern linear models, e.g., state-space models (Dao & Gu, 2024), linear attention (Katharopoulos et al., 2020), and fast-weight programmers (Schlag et al., 2021; Yang et al., 2025a), differ in how they store and update their state, they can all be unified under the associative memory view (Wang et al., 2025; Liu et al., 2024; Behrouz et al., 2025a). Under this view, each mechanism maintains a memory of key-value associations and uses queries to retrieve relevant values from memory. Their core representations, namely the queries, keys, and values, are obtained through linear projections of the input, parameterized by weight matrices $\boldsymbol{W}^Q, \boldsymbol{W}^K$, and $\boldsymbol{W}^V$. For input $\boldsymbol{x} \in \mathbb{R}^{1\times D}$, we have

$$\boldsymbol{q} = \boldsymbol{x}\boldsymbol{W}^Q, \quad \boldsymbol{k} = \boldsymbol{x}\boldsymbol{W}^K, \quad \boldsymbol{v} = \boldsymbol{x}\boldsymbol{W}^V,$$

where $\{\boldsymbol{W}^Q, \boldsymbol{W}^K\} \in \mathbb{R}^{D\times D_k}, \boldsymbol{W}^V \in \mathbb{R}^{D\times D_v}$. For input sequence $\boldsymbol{X} := [\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_T] \in \mathbb{R}^{T\times D}$, we use $\boldsymbol{Q} = [\boldsymbol{q}_1; \boldsymbol{q}_2; \ldots; \boldsymbol{q}_T]$ to denote all queries, and define $\boldsymbol{K}$ and $\boldsymbol{V}$ similarly. This viewpoint provides the basis for the tied-projection design in Section 3.

Softmax Attention. For causal softmax attention, the output $\boldsymbol{o}_t \in \mathbb{R}^{1\times D_v}$ at timestep t is a weighted aggregation of past values $\boldsymbol{V}_{\le t}$ based on the softmax-normalized similarity between current query $\boldsymbol{q}_t$ and past keys $\boldsymbol{K}_{\le t}$: $\mathrm{softmax}\big(\boldsymbol{q}_t\boldsymbol{K}_{\le t}^\top\big)\,\boldsymbol{V}_{\le t}$. [2] In parallel form, it is expressed as

$$\boldsymbol{O} = \mathrm{Mixer}_{\mathrm{attention}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) := \mathrm{softmax}\big(\boldsymbol{L}^U + (\boldsymbol{Q}\boldsymbol{K}^\top)\big)\,\boldsymbol{V},$$

where $\boldsymbol{L}^U \in \mathbb{R}^{T\times T}, \boldsymbol{L}^U_{ij} = -\infty \cdot \mathbb{I}[i < j]$ is the causal mask. Its state, the KV cache, stores the key and value vectors $\boldsymbol{k}_t, \boldsymbol{v}_t$ from all previous time steps and therefore grows linearly with sequence length. The mixer output $\boldsymbol{O}$ is passed through the output projection $\boldsymbol{W}^O \in \mathbb{R}^{D_v\times D}$ to get the final block output in the original input dimension, $\boldsymbol{Y} = \boldsymbol{O}\boldsymbol{W}^O$. As this output projection step is present in other models, we will ignore it for clarity from this point onward.

Linear Recurrent Neural Networks (RNNs). In contrast to softmax attention, linear RNNs, which encompass linear attention variants and modern state space models (SSMs), are grounded in their recurrent structure. A key characteristic of these models is their fixed-size states, which enable runtime and memory efficiency. Despite not having an explicit cache of key-value pairs, the states *are* updated with key-value associations at each timestep. In general, a structured transition matrix $\boldsymbol{A}_t \in \mathbb{R}^{D_k\times D_k}$ adjusts the prior state $\boldsymbol{S}_{t-1} \in \mathbb{R}^{D_k\times D_v}$ while the current key-value interaction is incorporated via an outer-product. The output is determined using a simple readout with the current query.

$$\boldsymbol{S}_t = \boldsymbol{A}_t\boldsymbol{S}_{t-1} + \boldsymbol{k}_t^\top\boldsymbol{v}_t, \qquad \boldsymbol{o}_t = \boldsymbol{q}_t\boldsymbol{S}_t,$$

Thus, its state and output can still be expressed through query, key, and value representations.

Mamba-2. The discretized Mamba-2 SSM (Dao & Gu, 2024) is one instantiation of a linear RNN that uses a scalar times identity transition, $\boldsymbol{A}_t = \alpha_t\boldsymbol{I}$, where α_t is an input-dependent decay factor. [3] Its parallel form relies on decay-based lower-triangular mask $\boldsymbol{\Gamma}$ applied with a Hadamard product ($\circ$), which draws connections to attention variants and highlights the value retrieval mechanism.

$$\boldsymbol{O} = \mathrm{Mixer}_{\text{Mamba-2}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{X}) := \big(\boldsymbol{\Gamma} \circ \big(\boldsymbol{Q}\boldsymbol{K}^\top\big)\big)\,\boldsymbol{V}, \quad \boldsymbol{\Gamma} = \begin{bmatrix} 1 & & & \\ \alpha_2 & 1 & & \\ \alpha_3\alpha_2 & \alpha_3 & 1 & \\ \vdots & & & \ddots \end{bmatrix}$$

[2] We ignore positional encodings and other complementary components, e.g., head structure, head dimension normalization, for clarity.

[3] In Mamba-2, queries, keys, and values are referred to as C, B, x respectively, but we utilize attention terminology to draw similarities. We also ignore the discretization parameters and the tied nature of α_t and $\boldsymbol{v}_t$ in this section for clarity.

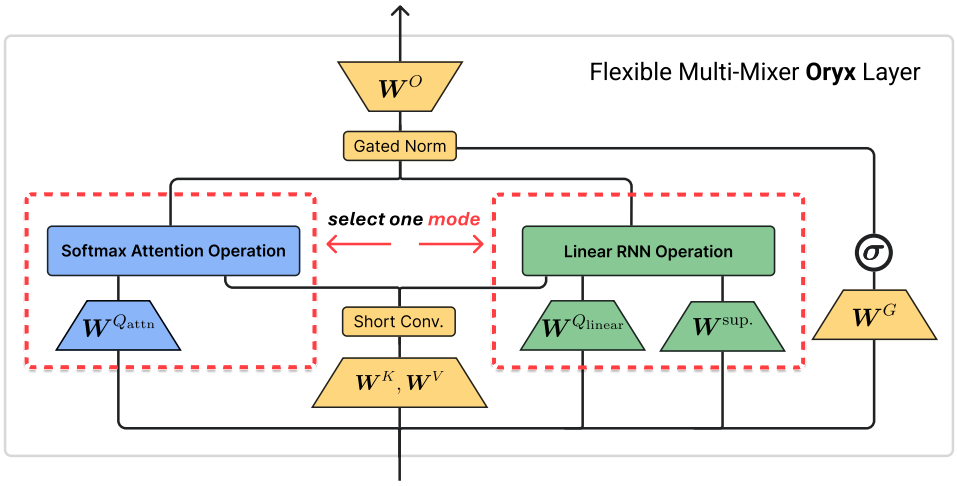


Figure 2: The general ORYX block shares the core representations between softmax attention and a linear recurrent mechanism through tied key-value projections and uses additional components critical to the linear mechanism's performance, e.g., the short convolution and gate. During a forward pass, the shared key-value representation updates both the KV cache and linear state, allowing either the softmax attention mixer or linear mixer to be chosen as the mode of operation at each timestep.

Gated DeltaNet. Fast-weight programmers (Schlag et al., 2021; Yang et al., 2025b), such as Gated DeltaNet (Yang et al., 2025a), can also be viewed under this lens. While the memory is queried the same way, the gated delta update rule enables a more expressive state update mechanism

$$\boldsymbol{S}_t = \left(\alpha_t \left(\boldsymbol{I} - \beta_t \boldsymbol{k}_t^\top \boldsymbol{k}_t\right)\right) \boldsymbol{S}_{t-1} + \beta_t \boldsymbol{k}_t^\top \boldsymbol{v}_t, \quad \boldsymbol{o}_t = \boldsymbol{q}_t \boldsymbol{S}_t,$$

where α_t, β_t are both data-dependent scalars. Like other sub-quadratic alternatives, it also retains a parallel representation. Reusing the decay mask $\boldsymbol{\Gamma}$ from the Mamba-2 formulation, we have

$$\boldsymbol{O} = \text{Mixer}_{\text{GDN}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}; \boldsymbol{\Gamma}, \boldsymbol{\beta}) := \left(\boldsymbol{\Gamma} \circ \left(\boldsymbol{Q}\boldsymbol{K}^\top\right)\right) \left[\boldsymbol{I} + \text{strictLower}\left(\text{diag}(\boldsymbol{\beta}) \left(\boldsymbol{\Gamma} \circ \left(\boldsymbol{K}\boldsymbol{K}^\top\right)\right)\right)\right]^{-1} \text{diag}(\boldsymbol{\beta}) \boldsymbol{V}.$$

While these sequence mixers differ in how they store (e.g., KV cache or fixed state), normalize (e.g., softmax, decay mask), and update (e.g., delta rule) key-value associations, their common query-key-value interaction structure provides a useful lens for connecting softmax attention and linear models. This connection motivates the design of our ORYX block in Section 3, where tied projections are used to update both attention and recurrent states from shared representations.

3 DESIGNING THE MULTI-MIXER ORYX SHARED BLOCK

In this section, we describe the sequence-mixing ORYX shared block that supports operating in both softmax attention and linear recurrent mechanisms. The resulting block maintains compatible attention and recurrent states and includes architectural components core to the modeling abilities of each mechanism while sharing most of its parameters across mixers (Figure 2). We instantiate the linear mixer as Mamba-2 and Gated DeltaNet (GDN) for concreteness, but our design choices *can* be applied with other linear models with similar query-key-value interactions.

Shared Key-Values and Mixer-Specific Queries. Motivated by the key-value associative memory view (§ 2), ORYX ties the key and value projections across mixer modes, enabling one set of representations, calculated from one forward pass, to update both the attention KV cache and linear recurrent state. While the query projections can also be shared, our empirical results suggest that tying all three core representations across mixers hinders model performance (see details in § 4.3), and thus, we use unique weights for each mixer's query projection for stronger performance. We conjecture that the differences in their state update and output rules may require different query vectors to extract mode-specific crucial information, a hypothesis supported by empirical results.

Incorporating Additional Linear Model Components. ORYX incorporates the short convolution, multiplicative gate, and pre-output projection normalization — common components in current linear models — to retain these important inductive biases. In our implementation, the short

convolution is applied to only the shared keys and values before they are passed into the selected $\text{Mixer}_{\boldsymbol{M}}$; the queries are not convolved due to being mixer-specific.

For a selected mixer mode $\boldsymbol{M} \in \{\text{attention}, \text{Mamba-2}, \text{GDN}\}$, the block computes: [4]

$$\boldsymbol{O} = \text{Mixer}_{\boldsymbol{M}}(\boldsymbol{X}\boldsymbol{W}^{Q_{\boldsymbol{M}}}, \boldsymbol{X}\boldsymbol{W}^{K}, \boldsymbol{X}\boldsymbol{W}^{V}, \boldsymbol{X}; \boldsymbol{W}^{\text{sup.}}),$$

where $\boldsymbol{W}^{\text{sup.}}$ denotes the additional mixer-specific supporting parameters, either data-dependent or independent, such as discretization or delta-rule parameters. The mixer output is gated element-wise and normalized before the final shared output projection

$$\boldsymbol{Y} = \text{GatedRMSNorm}\left(\boldsymbol{O}, \sigma(\boldsymbol{X}\boldsymbol{W}^{G})\right)\boldsymbol{W}^{O},$$

where σ is an activation function, usually SiLU (Hendrycks & Gimpel, 2016), and $\boldsymbol{W}^{G} \in \mathbb{R}^{D \times D_v}$ is the shared gate projection. The exact normalization and supporting parameters depend on the linear mechanism used; we detail the specifics of our variants in Section B. Most parameters, including $\boldsymbol{W}^{K}$, $\boldsymbol{W}^{V}$, $\boldsymbol{W}^{G}$, and $\boldsymbol{W}^{O}$, are shared across mixers. Thus, these shared representations can be calculated with one shared forward pass. The general block structure is visualized in Figure 2, and an abstracted version of the pseudocode can be found at Listing 1.

Maintaining Compatible States and Head Structures. At each timestep, ORYX computes shared key and value representations and uses them to update both the attention KV cache and linear recurrent state. While the selected mixer determines the block output, the joint update allows both states to retain the same token history. We note that while both mixer states are maintained, only the updated state of the selected mixer is needed for output computation. However, one issue is that different mixers often use differing head structures. For instance, modern Transformers use attention in a multi-head (MHA) or grouped-query head (GQA) structure, while Mamba-2 adopts a multi-value (MVA) structure. To enable the effective sharing of weight projections, we use the same head structure across mixers, matching that of attention, i.e., MHA in our experiments. Despite this constraint applied on the linear mixers, we find that they remain empirically effective.

Chunked Mixed-Mode Training. To enable robust mode switching capabilities at inference time, we train ORYX with chunked mixed-mode training (see ablation in § 4.3). Each training sequence is partitioned into fixed-length chunks, e.g., of 128 tokens, and each chunk is randomly assigned a mixer mode, e.g., attention or linear mode. All ORYX blocks use the same chunk assignment, and our experiments find that a 1:3 attention-to-linear chunk training ratio balances performance.

4 EMPIRICAL RESULTS AND PROPERTIES OF ORYX

We evaluate the empirical performance and capabilities of ORYX in the following section. Sections 4.1.1 and 4.1.2 discusses the language modeling and retrieval performance of our models when operating using only one mixer mode. Section 4.2 then explores the mode switching capabilities of the model, and Section 4.3 ablates architectural and training design choices.

ORYX Setup. Our model follows the Transformer++ setup originating from Touvron et al. (2023) and detailed in Gu & Dao (2024), which includes interleaved SwiGLU MLPs. We use ORYX-TM to denote a self-attention (Transformer)/Mamba-2 shared-block model, and ORYX-TG to denote a self-attention/GDN shared-block model. We set the dimensions of the query, key, and value to be equal, i.e., $D_k = D_v$ following standard Transformer convention; however, we fix the head dimension $D_{k,v} = 128$ across all scales to preserve the recurrent state size. As the gate increases the active parameters, the resulting mixer-to-MLP parameter ratio increases from the baselines Transformer's 4:8 to 5:8. As the MLPs and most sequence-mixer projections are shared across mixers, more than 90% of the weights are jointly used across modes.[5]

Baseline Setup. Transformer baselines follow the architecture of Touvron et al. (2023) and head structure of Brown et al. (2020). Mamba-2 baselines interleave Mamba-2 blocks, using $D_k = 128, D_v = 64$ and expansion factor of 2, with SwiGLU MLPs at a 6:6 parameter ratio. Gated DeltaNet baselines similarly interleave GDN blocks, using $D_k = 128, D_v = 256$, with MLPs at the same 6:6 ratio. To parameter-match models, we increase the MLP widths of the baselines to compensate for the gate in ORYX blocks (the short convolution leads to a negligible increase).

[4] Rotary embeddings, the short convolution, etc., are abstracted away in the Mixer class. Section B details the exact architecture for each ORYX variant.

[5] The percentage calculation excludes the embedding in the total parameter count.

Table 1: Downstream language modeling evaluations on parameter-matched models trained with 100B FineWeb-Edu tokens. We compare baseline models, ORYX-TM (Transformer/Mamba-2), and ORYX-TG (Transformer/Gated DeltaNet) at each parameter scale. Best results for each size are **bolded**, and second best are underlined. The ORYX results reported below use the same mixer for the entire model and sequence. Under a fixed training token budget, both modes of our dual mixer ORYX model achieve comparable or better performances than that of single mixer baselines.

	Family	Mixer	LAMB. ppl ↓	LAMB. acc ↑	HellaS. acc_n ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc_n ↑	WinoGr. acc ↑	OBQA acc ↑	**Avg.** acc ↑
810M	Baseline	Transformer	13.6	46.2	57.0	73.1	71.3	37.3	58.3	28.6	53.1
		Mamba-2	13.4	45.6	58.2	72.7	72.8	**40.1**	56.0	**31.2**	53.8
		Gated DeltaNet	12.1	47.4	58.3	72.5	73.3	39.8	58.6	29.6	54.2
	ORYX-TM	Transformer	12.5	48.0	58.0	73.6	**73.7**	39.4	59.1	31.0	**54.7**
		Mamba-2	**11.9**	**48.2**	57.9	**73.9**	**73.7**	39.0	**59.6**	30.6	**54.7**
	ORYX-TG	Transformer	13.4	46.2	58.1	72.8	71.5	37.5	58.5	27.8	53.2
		Gated DeltaNet	12.8	45.8	**58.4**	73.1	72.4	38.5	59.4	30.8	54.0
1.4B	Baseline	Transformer	11.4	49.9	60.6	74.1	73.6	42.1	58.0	31.2	55.6
		Mamba-2	11.2	48.6	60.9	74.7	74.5	42.7	58.1	30.4	55.7
		Gated DeltaNet	10.6	49.9	61.9	75.0	75.0	42.2	60.9	31.4	56.6
	ORYX-TM	Transformer	11.1	50.2	61.3	75.0	75.7	42.0	58.0	31.6	56.3
		Mamba-2	**10.5**	**50.4**	**62.1**	**75.2**	75.3	**43.6**	58.5	31.2	56.6
	ORYX-TG	Transformer	10.9	49.9	61.8	74.9	75.4	41.7	59.8	31.8	56.5
		Gated DeltaNet	10.6	50.0	**62.1**	75.1	**76.2**	43.1	**62.0**	**32.2**	**57.3**

Experimental Setup. We pretrained models at four different scales, each with 100B tokens of FineWeb-Edu (Lozhkov et al., 2024) using the GPT-2 tokenizer (Radford et al., 2019) at 2K context length. A cosine scheduler with 10% warmup was used, and AdamW (Loshchilov & Hutter, 2019) with $\beta = (0.9, 0.95)$ and 0.1 weight decay was used as the optimizer. For baselines, peak learning rate was set to $5\times$ that of Brown et al. (2020), following Dao & Gu (2024). A factor of $10\times$ was used for ORYX models. This increase is to mitigate the reduced mixer-specific gradients from chunked-training and different optimization landscapes induced by tied representations. Empirically, we find it improves both mixer performances while preserving training stability. Training used bf16 mixed precision and a global batch size of 1M tokens for 1B models, and 0.5M for the rest.

Evaluation Setup. The language modeling abilities of models were evaluated with a suite of standard commonsense reasoning and language understanding tasks: LAMBADA (Paperno et al., 2016; Radford et al., 2019), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2019), Arc-Easy and Challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2019), and OpenbookQA (Mihaylov et al., 2018). We also measured the retrieval capabilities of the 1.4B models on both synthetic needle-in-the-haystack (NIAH) tasks (Hsieh et al., 2024) and real-world retrieval tasks in cloze format (Arora et al., 2024; 2025a): SWDE (Arora et al., 2025b; Lockard et al., 2019), SQuAD (Rajpurkar et al., 2018), FDA (Arora et al., 2025b), TriviaQA (TQA) (Joshi et al., 2017), NQ (Kwiatkowski et al., 2019), and DROP (Dua et al., 2019). A context length of 2K was used for all retrieval tasks.

4.1.1 LANGUAGE MODELING PERFORMANCE OF INDIVIDUAL MIXERS

Table 1 reports each ORYX model's performance when using one of its mixer modes in isolation, i.e., either softmax attention or linear mechanism used for the entire sequence, to determine whether each mode remains useful on its own. Across scales, ORYX modes remain competitive with their corresponding single-mixer baselines *despite* being trained for the same number of tokens. At the 1.4B scale, both attention and linear modes of ORYX-TG and ORYX-TM outperform the baselines by at least 0.7 percentage points on average. Notably, despite these tasks being evaluated "out-of-distribution" — the mode switching training does not explicitly train the entire model with all chunks assigned to the same mixer — the models are still able to generalize to this edge case (the probability that a sample is processed entirely by the linear mechanism is $(3/4)^{16} \approx 1\%$ and $(1/4)^{16} \approx 2.3 \times 10^{-8}\%$ for softmax attention). These results suggest that sharing key-value representations and tying the majority of weights do *not* prevent either mixer from learning effective standalone behavior.

Table 2: Retrieval results for 1.4B baseline and ORYX models on real-world and synthetic retrieval tasks at 2K context length. Results reported below use the same mixer for the entire model and sequence. ORYX-TM denotes Transformer/Mamba-2 shared blocks, and ORYX-TG, Transformer/Gated DeltaNet. The isolated modes of ORYX remain comparable to their corresponding baselines.

		Real-World Retrieval							Synthetic Retrieval			
Family	**Mixer**	SWDE	SQuAD	FDA	TQA	NQ	DROP	**Avg.**	NIAH-1	NIAH-2	NIAH-3	**Avg.**
Baseline	Transformer	42.3	42.8	55.2	65.9	26.7	22.9	42.6	99.0	90.4	95.0	94.8
ORYX-TM	Transformer	50.1	41.7	57.6	63.9	28.3	22.5	44.0	99.8	99.4	81.6	93.6
ORYX-TG	Transformer	44.5	41.2	46.4	64.8	27.8	21.7	41.1	99.4	98.6	99.4	99.1
Baseline	Mamba-2	20.6	36.1	21.0	61.7	22.4	20.3	30.3	90.4	15.8	33.2	46.5
ORYX-TM	Mamba-2	22.2	36.4	19.4	63.7	21.8	22.3	31.0	50.8	17.0	41.0	36.3
Baseline	Gated DeltaNet	27.4	36.4	24.2	63.3	23.8	21.6	32.8	99.4	39.6	32.6	57.2
ORYX-TG	Gated DeltaNet	29.3	37.9	18.2	64.0	23.5	22.0	32.5	99.4	60.2	82.8	80.8

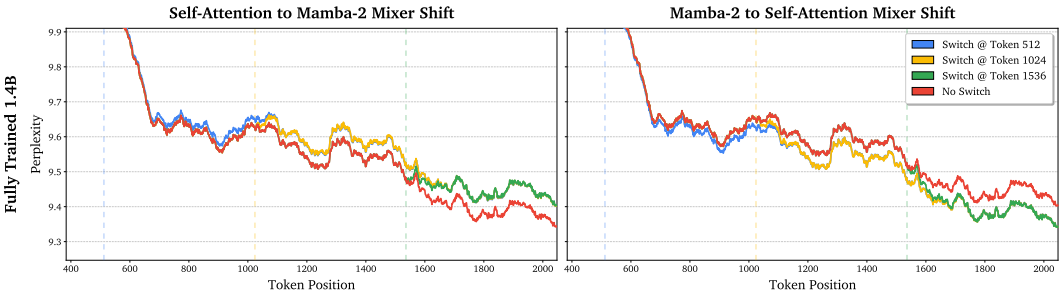


Figure 3: Smoothed token-level perplexity across token position for pretrained 1.4B ORYX-TM model trained with chunked mixed-mode training. After switching from softmax attention to Mamba-2 and vice versa at different positions, perplexity rapidly approaches the corresponding no-switch baseline, indicating that the mixers share compatible representations.

4.1.2 RETRIEVAL CAPABILITIES OF INDIVIDUAL MIXERS

The single-mode retrieval results show that ORYX generally preserves the retrieval capabilities of its constituent mixers when using the same mixer for the entire sequence. Both Mamba-2 and GDN variants achieve comparable real-world retrieval performance except for the GDN variant on the FDA dataset, which requires extracting information from unstructured data. Notably, the GDN mode of ORYX-TG *substantially* improves NIAH performance over its baseline, achieving more than $2\times$ the accuracy on NIAH-3 and more than $1.5\times$ the accuracy on NIAH-2. We emphasize that these results on the linear mechanisms were achieved despite using only half and two-thirds of the total state size of the respective Mamba-2 and GDN baselines, which suggests that shared-block model and training can improve downstream capabilities.[6]

4.2 FLEXIBLE MODE SWITCHING DURING INFERENCE

While ORYX can be deployed in an attention-only or linear-only mode, our chunk-level mode switching mechanism enables a new axis of exploring hybrid models. The models can flexibly change between mixers during prefill with little to no degradation in perplexity as demonstrated in Figure 3. In the pretrained ORYX-TM 1.4B model, after a switch, the perplexity rapidly approaches that of the corresponding no-switch baseline in the selected mode. The results in Section D display the same behaviors generalize to non-chunk-aligned switch boundaries and multiple switches for both TM and TG models, demonstrating the ability of various mixers to share internal representations across sequences.

We further evaluate the mode switching ability on retrieval tasks by splitting each sample into a context segment (**Context**) and a prompt/generation segment (**Prompt + Gen**), then processing the two segments with different modes. For synthetic NIAH tasks, the context consists of the haystack, and the prompt consists of the needle query. For real-world tasks, because the boundary between context and prompt is less distinct due to the cloze format, we designate the first 97.5% of tokens as the

[6] The total recurrent state size of Mamba-2 is $2D_k D_{\text{model}}$, GDN is $1.5D_k D_{\text{model}}$, and ORYX is $D_k D_{\text{model}}$.

Table 3: Cross-mode retrieval results for 1.4B baseline and ORYX models on real-world and synthetic retrieval tasks at 2K context length, grouped by the mixer used for prompt prefill and generation (**Prompt + Gen**). Baselines use the same mixer for the entire sequence, while ORYX use a different mixer for prefilling the context (**Context**) than for the prompt prefill and generation. The ORYX modes can remain comparable to baselines despite switching modes, suggesting that the underlying representations required for retrieval are preserved across modes.

			Real-World Retrieval							Synthetic Retrieval			
Prompt + Gen	**Family**	**Context**	SWDE	SQuAD	FDA	TQA	NQ	DROP	**Avg.**	NIAH-1	NIAH-2	NIAH-3	**Avg.**
Transformer	Baseline	Transformer	42.3	42.8	55.2	65.9	26.7	22.9	42.6	99.0	90.4	95.0	94.8
	ORYX-TM	Mamba-2	50.9	40.9	56.4	64.5	28.2	22.4	43.9	99.8	97.6	58.0	85.1
	ORYX-TG	Gated DeltaNet	46.1	41.1	47.9	64.1	28.0	20.9	41.4	99.0	97.6	95.8	97.5
Mamba-2	Baseline	Mamba-2	20.6	36.1	21.0	61.7	22.4	20.3	30.4	90.4	15.8	33.2	46.5
	ORYX-TM	Transformer	21.8	37.1	17.8	63.2	21.3	21.6	30.5	44.6	14.6	51.0	36.7
Gated DeltaNet	Baseline	Gated DeltaNet	27.4	36.4	24.2	63.3	23.8	21.6	32.8	99.4	39.6	32.6	57.2
	ORYX-TG	Transformer	29.5	38.2	17.1	63.9	24.0	21.2	32.3	99.4	73.6	88.2	87.1

context and the remaining tokens as the prompt. Table 3 shows that mode-switching often preserves the retrieval performance of the target prompt/generation mixer, especially on real-world retrieval tasks. We highlight that both ORYX models are able to prefill with the linear mode and generate with softmax attention, achieving comparable results to the Transformer baseline and *significantly surpassing their linear baselines*. In particular, with mixed inference mode, ORYX-TM achieves 13.5 percentage points higher on average for real-world retrieval tasks, and 38.6 percentage points higher for NIAH tests. For ORYX-TG, the numbers are 8.6 and 40.3, respectively. We note that synthetic evaluations are more sensitive to switching direction and mixer choice, particularly the Transformer-to-Mamba-2 setting. These results suggest that exact retrieval may stress the compatibility of shared representations more strongly than real-world retrieval for certain models.

This new capability may enable use cases such as models that vary mechanisms depending on the task, e.g., attention for retrieval-heavy questions, or paradigms where large portions of thinking traces are generated with the linear mixer and verified with the attention. We note that models that deploy with mode switching enabled will require the storage of both the KV cache and linear state, but memory costs would be dominated by that of the KV cache at longer context lengths.

4.3 ARCHITECTURE AND TRAINING ABLATIONS

In the following section, we ablate the architectural and training choices that enable the strong performance and mode switching abilities of ORYX models. We explore the impact of the mixer-specific query and additional linear model components on performance and the importance of the chunked mixed-mode training on mode-switching capabilities. Ablations are conducted at the 380M scale, unless specified otherwise, with the standard training regime as the final models except trained to Chinchilla scaling law token count ($20\times$ tokens-to-parameter ratio) (Hoffmann et al., 2022). When ablating our ORYX architecture, we utilize sequence mixed-mode training, where an entire sequence is assigned to only one mixer instead of our final chunked mixed-mode training, as both training regimes result in comparable pretraining loss and findings are consistent under both.

Mixer-Specific Queries. The final shared ORYX block uses mixer-specific query projections while sharing key and value projections across mixer modes. While query weights can be shared across mixers, which would reduce the total parameter count, it would not reduce the active parameter count and empirically degrades performance (Table 4a). Thus, we can conclude that attention and recurrent mixers can share the state updating representations, i.e., the key and values, but benefit from separate readout components, i.e., the query, due to their differences in state parameterization. We find similar mechanism-specific design requirements for ORYX-TG. For instance, SiLU activations after the short convolution are important, and the query-key normalization should only be applied to the GDN components (§ B).

Additional Architectural Components. Our ablations find that the usage of the short convolution is critical for both softmax attention and the linear model performance, while adding the gate further improves perplexity (Table 4b). In contrast, applying element-wise gating without the short convolution hurts both performances, underscoring the complex interactions between components that arise when designing shared blocks. Because ORYX uses separate queries for each mixer and supports mode switching during inference, we apply the short convolution only to the shared keys and values.

Notably, when these components, normalization, convolution, and gate, are added to the Transformer baseline, the language modeling performance does not improve (Table 5a). Thus, the strong ORYX results observed are unlikely to be caused solely by the addition of these components, but rather by the interaction of the various inductive biases when sharing core representations.

Chunked Mixed-Mode Training. Our chunk-level mixed-mode training regime is critical in enabling robust mode switching for our ORYX model. As an alternative, we train our models with a sequence-level mixed-mode scheme where an entire training sequence is processed by only one mixer. While these models achieve comparable pretraining losses to chunk-level (Table 5b), the mode switching capability does *not consistently* manifest. The sequence-level trained models seem to possess mode switching capabilities from softmax attention to Mamba-2, but suffer from large perplexity degradation when shifting from Mamba-2 to softmax attention at certain scales. Figure 4 displays the smoothed perplexities at each token position for the 130M and 380M models trained with and without chunk switching, and Section D discusses further. Our ablation suggests that the mixer transitions induced by the chunk-based training encourages the representations to remain more compatible throughout a sequence as compared to sequence-based training. We note that chunked mixed-mode training incurs an increase in computational overhead compared to sequence-level mixed training. While softmax attention strictly forgoes the quadratic cost of calculating outputs for unselected sequence chunks, the linear RNN must roll its recurrent state forward across all chunks to preserve output correctness, resulting in a fixed linear compute cost. We analyze general compute usage in Section C and leave overhead reduction for future work.

5 RELATED WORK

While some methodologies have been introduced that can change the computational pathway for a given input, e.g., context-dependent segmentation for tokenization (Hwang et al., 2025), there is comparatively little prior work that is directly motivated by the prospect of using different sequence mixers throughout generation or prefill. Our method, to the best of our knowledge, is among the first to study the flexible switching of sequence mixers in a language modeling setting. Concurrent work HAM (Lufkin et al., 2026) combines a linear RNN with attention by using the linear model to compress the full sequence while reserving the KV cache for certain routed tokens, resulting in linear RNN and sparse attention hybrid. TransMamba (Li et al., 2025) processes a sequence with a mixture of softmax attention and Mamba-2 with shared weights, but has a predetermined, fixed switch location at each layer. This constraint prevents the flexible usage of various mixers depending on the sequence context, which is further compounded by their unidirectional switch, i.e., only from softmax attention to Mamba once. Recent SLA2 (Zhang et al., 2026) also enables mixer-choice through a learned router but is formulated as sparse-linear attention for video diffusion rather than flexible switching for autoregressive prefill or generation. Our work provides a general framework for mixer selection in autoregressive language modeling and can serve as a basis for objectives such as learned routing or learned sparsity. Section A further covers related works.

6 DISCUSSION AND CONCLUSION

In our work, we introduce ORYX, a multi-mixer architecture that enables sequence-axis hybridization where different mixer mechanisms can be utilized across a sequence. ORYX maintains shared representations across the softmax attention and linear mechanism through tied projections, and achieves strong language modeling performance, preserves retrieval capabilities, and supports mode switching during prefill and generation. One consideration to highlight is that our design requires the model to keep and update both the KV cache and state memory at each timestep throughout generation if mode switching is used, but for longer context, the memory cost is dominated by the KV cache. Reducing the overhead of maintaining multiple states remains an important direction.

Our results suggest several extensions for developing sequence-level hybrid models. While our current work relies on static assignment, there exist many approaches to learnable mode-switching, e.g., using RL to train routers that dynamically route tokens using FLOPs saved as the reward. More broadly, the sequence-axis hybrid paradigm *opens a new design space* for adaptive compute. For instance, linear recurrent modes could process portions of a sequence with less computation, and quadratic attention could be selectively used for segments that require richer context utilization. Beyond these applications, our work suggests that the representations learned by different mechanisms can overlap and be used jointly, raising crucial questions of when and how different models develop compatible understandings and representations.

AI USE STATEMENT

In this work, we used generative AI tools for the occasional method implementation, results interpretation, and feedback on methodology and experiments. We have not used generative AI tools for developing conceptual frameworks, and the rest of the required disclosure tasks, e.g., dataset generation, proof writing, dataset manipulation, and mathematical claims, are not applicable to this work.

Additionally, we used generative AI tools for the formatting of tables, minor paper writing changes to improve readability, and identifying relevant literature. We have reviewed all AI-assisted work. We have checked the literature suggested by the LLM, verified the LLM-generated code, and ensured that the data in tables were correct. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

REPRODUCIBILITY STATEMENT

The architecture is described in detail in Sections 3 and 4 and Section B, and the chunked-mode training is detailed in Section 3 and Section B.1. The experimental and evaluation setup are detailed in Section 4.

REFERENCES

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, 2025a. URL https://arxiv.org/abs/2402.18668.

Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes, 2025b. URL https://arxiv.org/abs/2304.09433.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2024. URL https://arxiv.org/abs/2501.00663.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization, 2025a. URL https://arxiv.org/abs/2504.13173.

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures, 2025b. URL https://arxiv.org/abs/2512.24695.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

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.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

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.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models, 2024. URL https://arxiv.org/abs/2411.13676.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, and Yu Cheng. Mom: Linear sequence modeling with mixture-of-memories, 2025. URL https://arxiv.org/abs/2502.13685.

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.

Yunhao Fang, Weihao Yu, Shu Zhong, Qinghao Ye, Xuehan Xiong, and Lai Wei. Artificial hippocampus networks for efficient long-context modeling, 2025. URL https://arxiv.org/abs/2510.07318.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

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, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti

Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

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, 2022. URL https://arxiv.org/abs/2111.00396.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2016. URL https://arxiv.org/abs/1606.08415.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

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.

Jiaxi Hu, Yongqi Pan, Jusen Du, Disen Lan, Xiaqiang Tang, Qingsong Wen, Yuxuan Liang, and Weigao Sun. Comba: Improving bilinear rnns with closed-loop control, 2025. URL https://arxiv.org/abs/2506.02475.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time, 2022. URL https://arxiv.org/abs/2202.10447.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling, 2025. URL https://arxiv.org/abs/2507.07955.

IBM Research. Granite 4.0 models documentation. https://www.ibm.com/granite/docs/models/granite, 2025. Accessed: 2025-02-06.

Kazuki Irie, Morris Yau, and Samuel J. Gershman. Blending complementary memory systems in hybrid quadratic-linear transformers, 2025. URL https://arxiv.org/abs/2506.00744.

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.

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.

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.

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/.

Aakash Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles, 2026. URL https://arxiv.org/abs/2603.15569.

Yixing Li, Ruobing Xie, Zhen Yang, Xingwu Sun, Shuaipeng Li, Weidong Han, Zhanhui Kang, Yu Cheng, Chengzhong Xu, Di Wang, and Jie Jiang. Transmamba: Flexibly switching between transformer and mamba, 2025. URL https://arxiv.org/abs/2503.24067.

Xi Victoria Lin, Akshat Shrivastava, Liang Luo, Srinivasan Iyer, Mike Lewis, Gargi Ghosh, Luke Zettlemoyer, and Armen Aghajanyan. Moma: Efficient early-fusion pre-training with mixture of modality-aware experts, 2024. URL https://arxiv.org/abs/2407.21770.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners, 2024. URL https://arxiv.org/abs/2407.14207.

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 (eds.), *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)*, pp. 3047–3056, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1309. URL https://aclanthology.org/N19-1309/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Leon Lufkin, Tomás Figliolia, Beren Millidge, and Kamesh Krishnamurthy. Hybrid associative memories, 2026. URL https://arxiv.org/abs/2603.22325.

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.

NVIDIA, :, Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, Aleksandr Shaposhnikov, Alex Kondratenko, Alexander Bukharin, Alexandre Milesi, Ali Taghibakhshi, Alisa Liu, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amir Klein, Amit Zuker, Amnon Geifman, Amy Shen, Anahita Bhiwandiwalla, Andrew Tao, Anjulie Agrusa, Ankur Verma, Ann Guan, Anubhav Mandarwal, Arham Mehta, Ashwath Aithal, Ashwin Poojary, Asif Ahamed, Asit Mishra, Asma Kuriparambil Thekkumpate, Ayush Dattagupta, Banghua Zhu, Bardiya Sadeghi, Barnaby Simkin, Ben Lanir, Benedikt Schifferer, Besmira Nushi, Bilal Kartal, Bita Darvish Rouhani, Boris Ginsburg, Brandon Norick, Brandon Soubasis, Branislav Kisacanin, Brian Yu, Bryan Catanzaro, Carlo del Mundo, Chantal Hwang, Charles Wang, Cheng-Ping Hsieh, Chenghao Zhang, Chenhan Yu, Chetan Mungekar, Chintan Patel, Chris Alexiuk, Christopher Parisien, Collin Neale, Cyril Meurillon, Damon Mosk-Aoyama, Dan Su, Dane Corneil, Daniel Afrimi, Daniel Lo, Daniel Rohrer, Daniel Serebrenik, Daria Gitman, Daria Levy, Darko Stosic, David Mosallanezhad, Deepak Narayanan, Dhruv Nathawani, Dima Rekesh, Dina Yared, Divyanshu

Kakwani, Dong Ahn, Duncan Riach, Dusan Stosic, Edgar Minasyan, Edward Lin, Eileen Long, Eileen Peters Long, Elad Segal, Elena Lantz, Ellie Evans, Elliott Ning, Eric Chung, Eric Harper, Eric Tramel, Erick Galinkin, Erik Pounds, Evan Briones, Evelina Bakhturina, Evgeny Tsykunov, Faisal Ladhak, Fay Wang, Fei Jia, Felipe Soares, Feng Chen, Ferenc Galko, Frank Sun, Frankie Siino, Gal Hubara Agam, Ganesh Ajjanagadde, Gantavya Bhatt, Gargi Prasad, George Armstrong, Gerald Shen, Gorkem Batmaz, Grigor Nalbandyan, Haifeng Qian, Harsh Sharma, Hayley Ross, Helen Ngo, Herbert Hum, Herman Sahota, Hexin Wang, Himanshu Soni, Hiren Upadhyay, Huizi Mao, Huy C Nguyen, Huy Q Nguyen, Iain Cunningham, Ido Galil, Ido Shahaf, Igor Gitman, Ilya Loshchilov, Itamar Schen, Itay Levy, Ivan Moshkov, Izik Golan, Izzy Putterman, Jan Kautz, Jane Polak Scowcroft, Jared Casper, Jatin Mitra, Jeffrey Glick, Jenny Chen, Jesse Oliver, Jian Zhang, Jiaqi Zeng, Jie Lou, Jimmy Zhang, Jinhang Choi, Jining Huang, Joey Conway, Joey Guman, John Kamalu, Johnny Greco, Jonathan Cohen, Joseph Jennings, Joyjit Daw, Julien Veron Vialard, Junkeun Yi, Jupinder Parmar, Kai Xu, Kan Zhu, Kari Briski, Katherine Cheung, Katherine Luna, Keith Wyss, Keshav Santhanam, Kevin Shih, Kezhi Kong, Khushi Bhardwaj, Kirthi Shankar, Krishna C. Puvvada, Krzysztof Pawelec, Kumar Anik, Lawrence McAfee, Laya Sleiman, Leon Derczynski, Li Ding, Lizzie Wei, Lucas Liebenwein, Luis Vega, Maanu Grover, Maarten Van Segbroeck, Maer Rodrigues de Melo, Mahdi Nazemi, Makesh Narsimhan Sreedhar, Manoj Kilaru, Maor Ashkenazi, Marc Romeijn, Marcin Chochowski, Mark Cai, Markus Kliegl, Maryam Moosaei, Matt Kulka, Matvei Novikov, Mehrzad Samadi, Melissa Corpuz, Mengru Wang, Meredith Price, Michael Andersch, Michael Boone, Michael Evans, Miguel Martinez, Mikail Khona, Mike Chrzanowski, Minseok Lee, Mohammad Dabbah, Mohammad Shoeybi, Mostofa Patwary, Nabin Mulepati, Najeeb Nabwani, Natalie Hereth, Nave Assaf, Negar Habibi, Neta Zmora, Netanel Haber, Nicola Sessions, Nidhi Bhatia, Nikhil Jukar, Nikki Pope, Nikolai Ludwig, Nima Tajbakhsh, Nir Ailon, Nirmal Juluru, Nishant Sharma, Oleksii Hrinchuk, Oleksii Kuchaiev, Olivier Delalleau, Oluwatobi Olabiyi, Omer Ullman Argov, Omri Puny, Oren Tropp, Ouye Xie, Parth Chadha, Pasha Shamis, Paul Gibbons, Pavlo Molchanov, Pawel Morkisz, Peter Dykas, Peter Jin, Pinky Xu, Piotr Januszewski, Pranav Prashant Thombre, Prasoon Varshney, Pritam Gundecha, Przemek Tredak, Qing Miao, Qiyu Wan, Rabeeh Karimi Mahabadi, Rachit Garg, Ran El-Yaniv, Ran Zilberstein, Rasoul Shafipour, Rich Harang, Rick Izzo, Rima Shahbazyan, Rishabh Garg, Ritika Borkar, Ritu Gala, Riyad Islam, Robert Hesse, Roger Waleffe, Rohit Watve, Roi Koren, Ruoxi Zhang, Russell Hewett, Russell J. Hewett, Ryan Prenger, Ryan Timbrook, Sadegh Mahdavi, Sahil Modi, Samuel Kriman, Sangkug Lim, Sanjay Kariyappa, Sanjeev Satheesh, Saori Kaji, Satish Pasumarthi, Saurav Muralidharan, Sean Narentharen, Sean Narenthiran, Seonmyeong Bak, Sergey Kashirsky, Seth Poulos, Shahar Mor, Shanmugam Ramasamy, Shantanu Acharya, Shaona Ghosh, Sharath Turuvekere Sreenivas, Shelby Thomas, Shiqing Fan, Shreya Gopal, Shrimai Prabhumoye, Shubham Pachori, Shubham Toshniwal, Shuoyang Ding, Siddharth Singh, Simeng Sun, Smita Ithape, Somshubra Majumdar, Soumye Singhal, Stas Sergienko, Stefania Alborghetti, Stephen Ge, Sugam Dipak Devare, Sumeet Kumar Barua, Suseella Panguluri, Suyog Gupta, Sweta Priyadarshi, Syeda Nahida Akter, Tan Bui, Teodor-Dumitru Ene, Terry Kong, Thanh Do, Tijmen Blankevoort, Tim Moon, Tom Balough, Tomer Asida, Tomer Bar Natan, Tomer Ronen, Tugrul Konuk, Twinkle Vashishth, Udi Karpas, Ushnish De, Vahid Noorozi, Vahid Noroozi, Venkat Srinivasan, Venmugil Elango, Victor Cui, Vijay Korthikanti, Vinay Rao, Vitaly Kurin, Vitaly Lavrukhin, Vladimir Anisimov, Wanli Jiang, Wasi Uddin Ahmad, Wei Du, Wei Ping, Wenfei Zhou, Will Jennings, William Zhang, Wojciech Prazuch, Xiaowei Ren, Yashaswi Karnati, Yejin Choi, Yev Meyer, Yi-Fu Wu, Yian Zhang, Yigong Qin, Ying Lin, Yonatan Geifman, Yonggan Fu, Yoshi Subara, Yoshi Suhara, Yubo Gao, Zach Moshe, Zhen Dong, Zhongbo Zhu, Zihan Liu, Zijia Chen, and Zijie Yan. Nvidia nemotron 3: Efficient and open intelligence, 2025. URL https://arxiv.org/abs/2512.20856.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, Kai Chen, Mark Chen, Enoch Cheung, Aidan Clark, Dan Cook, Marat Dukhan, Casey Dvorak, Kevin Fives, Vlad Fomenko, Timur Garipov, Kristian Georgiev, Mia Glaese, Tarun Gogineni, Adam Goucher, Lukas Gross, Katia Gil Guzman, John Hallman, Jackie Hehir, Johannes Heidecke, Alec Helyar, Haitang Hu, Romain Huet, Jacob Huh, Saachi Jain, Zach Johnson, Chris Koch, Irina Kofman, Dominik Kundel, Jason Kwon, Volodymyr Kyrylov, Elaine Ya Le, Guillaume Leclerc, James Park Lennon, Scott Lessans, Mario Lezcano-Casado, Yuanzhi Li, Zhuohan Li, Ji Lin, Jordan Liss, Lily, Liu, Jiancheng Liu, Kevin Lu, Chris Lu, Zoran Martinovic, Lindsay McCal-

lum, Josh McGrath, Scott McKinney, Aidan McLaughlin, Song Mei, Steve Mostovoy, Tong Mu, Gideon Myles, Alexander Neitz, Alex Nichol, Jakub Pachocki, Alex Paino, Dana Palmie, Ashley Pantuliano, Giambattista Parascandolo, Jongsoo Park, Leher Pathak, Carolina Paz, Ludovic Peran, Dmitry Pimenov, Michelle Pokrass, Elizabeth Proehl, Huida Qiu, Gaby Raila, Filippo Raso, Hongyu Ren, Kimmy Richardson, David Robinson, Bob Rotsted, Hadi Salman, Suvansh Sanjeev, Max Schwarzer, D. Sculley, Harshit Sikchi, Kendal Simon, Karan Singhal, Yang Song, Dane Stuckey, Zhiqing Sun, Philippe Tillet, Sam Toizer, Foivos Tsimpourlas, Nikhil Vyas, Eric Wallace, Xin Wang, Miles Wang, Olivia Watkins, Kevin Weil, Amy Wendling, Kevin Whinnery, Cedric Whitney, Hannah Wong, Lin Yang, Yu Yang, Michihiro Yasunaga, Kristen Ying, Wojciech Zaremba, Wenting Zhan, Cyril Zhang, Brian Zhang, Eddie Zhang, and Shengjia Zhao. gpt-oss-120b & gpt-oss-20b Model Card, 2025. URL https://arxiv.org/abs/2508.10925.

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.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random feature attention, 2021. URL https://arxiv.org/abs/2103.02143.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

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.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Jingze Shi and Bingheng Wu. Wonderful matrices: Combining for a more efficient and effective foundation model architecture, 2024. URL https://arxiv.org/abs/2412.11834.

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.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context, 2025. URL https://arxiv.org/abs/2512.23675.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Asher Trockman, Hrayr Harutyunyan, J Zico Kolter, Sanjiv Kumar, and Srinadh Bhojanapalli. Mimetic initialization helps state space models learn to recall. *arXiv preprint arXiv:2410.11135*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. An empirical study of mamba-based language models, 2024. URL https://arxiv.org/abs/2406.07887.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory, 2025. URL https://arxiv.org/abs/2501.12352.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2024. URL https://arxiv.org/abs/2408.15664.

Xun Wu, Shaohan Huang, Wenhui Wang, and Furu Wei. Multi-head mixture-of-experts, 2024. URL https://arxiv.org/abs/2404.15045.

Yuxin Wu and Kaiming He. Group normalization, 2018. URL https://arxiv.org/abs/1803.08494.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention, 2021. URL https://arxiv.org/abs/2102.03902.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2024. URL https://arxiv.org/abs/2312.06635.

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.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025. URL https://arxiv.org/abs/2502.11089.

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.

Jintao Zhang, Haoxu Wang, Kai Jiang, Kaiwen Zheng, Youhe Jiang, Ion Stoica, Jianfei Chen, Jun Zhu, and Joseph E. Gonzalez. Sla2: Sparse-linear attention with learnable routing and qat, 2026. URL https://arxiv.org/abs/2602.12675.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right, 2025. URL https://arxiv.org/abs/2505.23884.

Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. Mixture of attention heads: Selecting attention heads per token, 2022. URL https://arxiv.org/abs/2210.05144.

APPENDIX

A ADDITIONAL RELATED WORK

Linear Layers. There has been a growing body of work focused on mitigating the memory and compute requirements of the softmax attention mechanism. These alternatives either aim to approximate the softmax attention mechanism (Katharopoulos et al., 2020; Choromanski et al., 2022; Peng et al., 2021; Xiong et al., 2021; Hua et al., 2022; Beltagy et al., 2020) or are inspired by other mechanisms. Many recent linear attention style models can be viewed as fast weight programmers (Yang

Table 4: **Left:** Untied, or disjoint, query weights outperform shared query weights for both types of RMS normalization, and default RMSNorm outperforms grouped RMSNorm. **Right:** Adding both an element-wise gate and short convolution on the key and values of a disjoint query model leads to the best performance. Model parameter counts are matched across variants; gated models increase the parameter count in the mixer block, so the MLP width is increased in non-gated models to compensate.

Query	RMSNorm	Attn ppl ↓	Mamba-2 ppl ↓
Shared	Default	16.10	16.00
	Grouped	17.73	17.77
Disjoint	Default	15.93	15.82
	Grouped	17.47	17.57

Conv?	Gate?	Attn ppl ↓	Mamba-2 ppl ↓
✓	✓	15.09	15.28
	✗	15.35	15.36
✗	✓	16.67	16.59
	✗	15.91	15.78

et al., 2025a; Schlag et al., 2021; Yang et al., 2025b; Hu et al., 2025), and the traditional state-space model (SSM) from control theory has also influenced many popular works (Gu & Dao, 2024; Dao & Gu, 2024; Gu et al., 2022; Smith et al., 2023). Trockman et al. (2024) showed that initializing such SSM layers to mimic attention layers improves performance on recall tasks, hinting at representational compatibility between the two layer types. Beyond linear attention and SSMs, test-time training (TTT) layers redefine the state as adjustable model weights during inference instead of an external tensor (Zhang et al., 2025; Tandon et al., 2025; Behrouz et al., 2024; 2025b). While these models do reduce the overall overhead needed for training and deployment, there exists a fundamental trade-off: the fixed-state size of linear RNNs forces lossy compression. This weakness becomes apparent in retrieval or in-context learning heavy tasks (Waleffe et al., 2024; Arora et al., 2025a).

Hybrid Models. Given efficiency benefits and strong modeling capabilities, linear layers are increasingly incorporated alongside standard softmax attention within hybrid models to mitigate this weakness (NVIDIA et al., 2025; Qwen Team, 2025; OpenAI et al., 2025; Kimi Team et al., 2025; IBM Research, 2025; Gemma Team et al., 2025). Current hybrid model design can mainly be split into two approaches: inter-layer and intra-layer. The majority of current hybrid models follow the inter-layer format where different mixer mechanisms are interleaved on a layer basis where each layer only utilizes a single sequence mixer, e.g., three Mamba layers followed by one softmax attention layer. Intra-layer designs integrate different types of mixer within a single layer or block. For instance, Hymba (Dong et al., 2024) utilizes sliding-window softmax attention and Mamba-2 in parallel within the same layer while maintaining completely separate parameters for both mixers prior to merging their outputs. Irie et al. (2025); Fang et al. (2025) utilize linear models as a mechanism to retain information discarded from the context of sliding-window, and thus can also be viewed as a variant of intra-layer hybrid models. Native Sparse Attention enables some sequence-level adaptivity by selecting the context processed by its attention pathways, but its per-token compute budget is largely fixed based on the configured sparsity pattern (Yuan et al., 2025). More broadly, most current hybrid models either use a static mixture of mixer types or restrict adaptivity to routing among attention pathways, dedicating a "fixed" architecture to each token, which can be computationally inefficient given the heterogeneous nature of sequences and queries. In contrast, ORYX enables the sequence mixing mechanism to be determined at each timestep.

Routing Mechanisms. The potential and usage of the ORYX block has parallels with routing-based layers and mechanisms. While the mixer selection was chosen manually in our paper, the ability to utilize both mechanisms freely enables the ability for a token to decide the selected mixer, similar to current mixture-of-expert routing (Fedus et al., 2022; Shazeer et al., 2017; Wang et al., 2024). While general MoE layers typically utilize homogeneous experts, Lin et al. (2024) partitions experts based on modality, similar to how our approach partitions the sequence mixer based on quadratic versus linear mechanisms. Beyond channel mixers, i.e., MLPs, routing has also enabled the selection of certain attention heads within the sequence mixer (Wu et al., 2024; Zhang et al., 2022). With the popularization of linear models with fixed-size states, methods have been proposed that route tokens to specific states to prevent overwriting of past information in other states (Du et al., 2025).

Table 5: **Left:** Additional architecture components added to fully pretrained, parameter-matched Transformer baselines do not benefit downstream language modeling performance. **Right:** The performance of chunked mixed-mode training is comparable to sequence mixed-mode training but results in stronger, more robust mode switching abilities, highlighted in Figure 3 and Figure 4.

Transformer 1.4B	Avg LM acc ↑
Baseline	55.6
+ Norm	55.4
+ Norm + Conv + Gate	55.6

Training	Attn ppl ↓	Mamba-2 ppl ↓
Sequence Assignment	15.09	15.28
Chunk Assignment	15.08	15.27

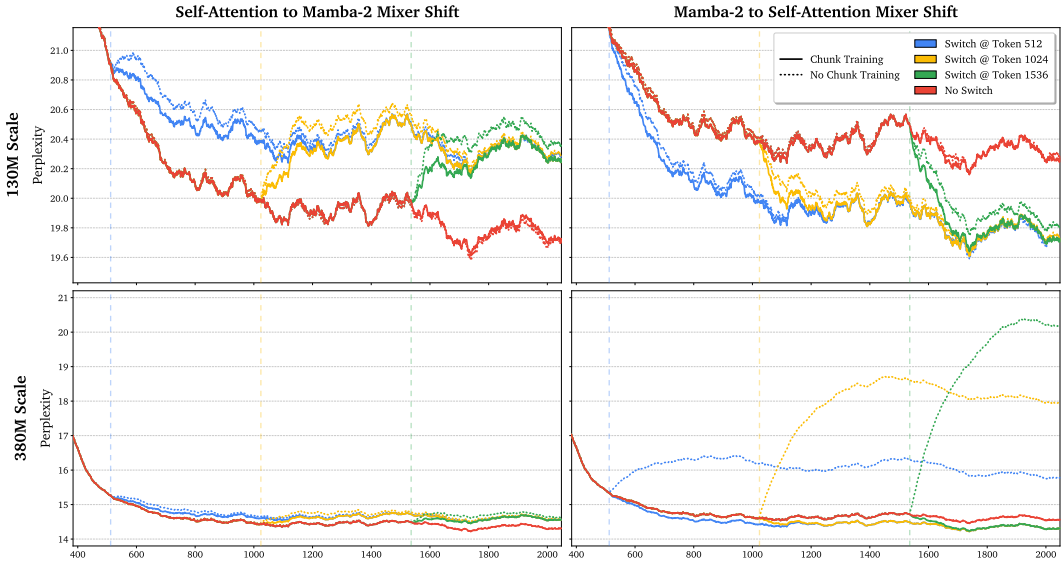


Figure 4: Smoothed perplexity across token index position for ORYX-TM models trained with and without chunk-level switching. Models trained without mode chunk switching perform slightly worse after switching mixers and can suffer from strong degradation in some cases.

B ORYX ARCHITECTURE DETAILS AND ADDITIONAL ABLATIONS

Mamba-2 ORYX. The RMSNorm used in the Mamba-2 variant of the ORYX block uses vanilla RMSNorm applied after the element-wise gating, following Dao & Gu (2024). We find that GroupNorm (Wu & He, 2018), used in Hymba (Dong et al., 2024), harms the shared block's Mamba-2 performance, but find that GroupNorm can be utilized if the Mamba-2 mixer were replaced with one that operates well with GroupNorm, e.g., Gated DeltaNet (Yang et al., 2025a). Unlike standard Mamba blocks, we also remove the activations after the convolution to maintain better consistency with the standard Transformer design, and find this does not significantly impact performance. In addition, we find that applying the RoPE positional embedding globally to queries and to the shared key empirically helps both mixers' performance (Table 6). Notably, integrating RoPE for both mixers improves Mamba-2's perplexity more significantly than it does for self-attention. While it is unclear why this is the case, this finding may be attributed to the beneficial nature of operating with complex SSMs (Lahoti et al., 2026) or incorporating direct positional encoding information rather than the decay's indirect information (Shi & Wu, 2024).

GDN ORYX. Following (Yang et al., 2025a), the pre-output projection norm uses the pre-gate, GroupNorm design. GDN requires the L2 normalization of queries and keys, which we find empirically hurts the performance of the attention mode significantly when applied (Table 7). Thus, we only apply L2 normalization to the queries and keys passed into the GDN mixer. Unlike the Mamba-2 variant of ORYX, the post-short convolution activation is important for modeling and retrieval performance, thus we add the standard SiLU activation to both mixers' queries and the shared keys and values (Table 8). Furthermore, rotary embeddings are only applied to the attention mixer's query and keys, as the application to GDN hurts retrieval performance. The mixer-specific appli-

Table 6: Applying RoPE to both self-attention and Mamba queries and keys improves performance compared to applying it solely to self-attention. This holds true regardless of whether a short convolution is used on the key-value components.

Conv?	RoPE Applied	Attn ppl ↓	Mamba-2 ppl ↓
✓	Both	15.54	15.54
	Attn	15.59	15.70
✗	Both	15.85	15.94
	Attn	16.05	16.03

cation of the architectural components, such as the embeddings and norms, can be subsumed by the mixer kernels with little loss in efficiency, as the main projection still shared, but we leave the implementation for future work.

Table 7: GDN requires L2 norm applied to its query and key, but we find not normalizing those of the attention mixer leads to the best results.

L2 Norm on Attn	Attn ppl ↓	GDN ppl ↓
Query + Key	28.68	16.16
Key	15.18	15.32
None	15.08	15.06

Table 8: **Left:** SiLU activation on the keys and values is critical for performance. While applying RoPE only in the attention mixer performs similar to that of applying RoPE in both attention and GDN for pretraining loss, it performs better on retrieval-based tasks. **Right:** Using attention-only RoPE, we find that applying the SiLU activation on the queries for both attention and GDN leads to strong performance and better retrieval than other configurations.

RoPE Applied	Shared KV Act.	Attn ppl ↓	GDN ppl ↓
Both	SiLU	15.08	15.06
	Id.	15.21	15.30
Attn	SiLU	15.09	15.07
	Id.	15.23	15.29

Attn Q act.	GDN Q act.	Attn ppl ↓	GDN ppl ↓
SiLU	SiLU	15.09	15.07
	Id.	15.31	15.28
Id.	SiLU	15.04	15.06
	Id.	15.23	15.26

B.1 Pseudocode

C General FLOPs Analysis

In this section, we provide a general FLOPs analysis of a forward pass of the Oryx sequence mixer algorithm and compare it against softmax attention and the Mamba-2 recurrent mechanism. The analysis is intended to compare asymptotic costs of the base sequence-mixing operation under a simplified chunked implementation, thus, we omit the compute associated with the weight projections and additional components, e.g., gate, and hardware-specific effects. As a refresher, modern recurrent linear models utilize a chunk forward algorithm during training or prefill, in which the total sequence length T is partitioned into chunks of size C and the outputs and states are calculated in parallel. Thus, $\boldsymbol{Q}$ is partitioned into $[\boldsymbol{Q}_1; \cdots; \boldsymbol{Q}_{T/C}]$ where $\boldsymbol{Q}_i \in \mathbb{R}^{C \times D_k}$, and the same is applied to $\boldsymbol{K}, \boldsymbol{V}$. The comprehensive algorithm can be found at Yang et al. (2024); Dao & Gu (2024).

Linear RNN (Mamba-2) FLOPs count. The forward pass of a chunked linear RNN can generally be decomposed into four steps. We mainly consider the large matrix multiplication operations and ignore smaller element-wise multiplication operations.

1. **Chunk state:** Here, each chunk i computes its contribution to the recurrent state $\boldsymbol{S}_i \in \mathbb{R}^{D_k \times D_v}$ via $\boldsymbol{K}_i^\top \boldsymbol{V}_i$ which results in $2CD_kD_v$ FLOPs per chunk and $2TD_kD_v$ for the entire sequence.

Sample implementations can be found for GatedRMSNorm[a] and ShortConv[b].

Listing 1: Abstracted pseudocode for the Mamba-2 variant of the ORYX shared block. We omit chunk-based mixed-mode forwarding, the joint state update, and implementation details for clarity and focus on the general representations and components.

```
# Learnable parameters/modules:
#   W_K, W_V, W_G, W_O, W_Q_attn, W_Q_mamba, W_sup_mamba
#   ShortConv, GatedRMSNorm

def OryxBlock(X, mode):
    K = X @ W_K
    V = X @ W_V
    G = SiLU(X @ W_G)

    K, V = ShortConv(K, V)

    if mode == "attention":
        Q = RoPE(X @ W_Q_attn)
        O = SoftmaxAttention(
            Q=Q,
            K=RoPE(K),
            V=V,
        )

    elif mode == "mamba2":
        Q = RoPE(X @ W_Q_mamba)
        O = Mamba2Mixer(
            x=V,
            B=RoPE(K),
            C=Q,
            params=W_sup_mamba,
        )

    Y = GatedRMSNorm(input=O, gate=G) @ W_O
    return Y
```

[a] https://github.com/state-spaces/mamba/blob/main/mamba_ssm/ops/triton/layernorm_gated.py

[b] https://github.com/Dao-AILab/causal-conv1d

2. **State passing:** The chunk states are updated to incorporate prior state information via a scan on the T/C total states of size $D_k \times D_v$. This results in approximately $2TD_kD_v/C$ FLOPs.

3. **Intra-chunk output:** Here, the output from the intra-chunk interactions are calculated via $(\boldsymbol{L}_i \circ \boldsymbol{Q}_i\boldsymbol{K}_i^\top)\boldsymbol{V}_i$ which results in $2C^2D_k + 2C^2D_v$ per chunk, when ignoring the mask. The total FLOPs for all chunks are then $2TCD_k + 2TCD_v$.

4. **Inter-chunk output:** Finally, the output from the cumulative prior hidden states must be calculated and added to the output arising from the intra-chunk calculations. Here, the $\boldsymbol{Q}_i\boldsymbol{S}_{i-1}$ calculation results in $2CD_kD_v$ per chunk and $2TD_kD_v$ in total. The summation of the inter-chunk and intra-chunk outputs results in TP FLOPs, which is negligible, thus ignored.

In total, the overall compute required for a single forward pass is approximately $4TD_kD_v + 2TC(D_k + D_v) + 2TD_kD_v/C$ which can be simplified to $8TC^2 + 2TC \approx 8TC^2$ when assuming $C = D_k = D_v$.

Softmax Attention FLOPs count. The standard causal softmax attention mechanism computes the quadratic interactions among the sequence constrained by a causal mask. While hardware-aware implementations exist (Dao et al., 2022), they do not decrease the overall FLOPs count. Similar to above, we ignore operations such as softmax and mainly focus on matmuls for clarity. For the score

computation $\boldsymbol{QK}^\top$, as only $T(T+1)/2$ pairs are valid due to causality, the total FLOPs required is $T(T+1)D_k$. The aggregation of $\boldsymbol{V}$ follows the same argument, resulting in a $T(T+1)D_v$ FLOPs. Thus, the total FLOPs required is $T(T+1)(D_k+D_v)$ which is around $2T^2C$ when assuming $C = D_k = D_v$.

ORYX FLOPs count. As ORYX uses shared key-value representations to update both the attention KV cache and linear state, the linear state needs to be updated at all instances. However, because the mixer selection determines the output, the FLOPs required for generating the output depends on δ which we will assign as the probability that a chunk is routed to the linear mechanism. Thus, we can consider the three aspects of ORYX's compute.

1. **Constant linear state update:** All chunks of the sequence need to update the state, resulting in $2TD_kD_v + 2TD_kD_v/C$ total FLOPs when accounting for the chunk state and state passing portions of the linear model.
2. **Linear output:** As only δ chunks are assigned to the linear mixer and thus require computation, the overall number of FLOPs used is approximately $\delta(2TCD_k + 2TCD_v + 2TD_kD_v)$ when accounting for the intra- and inter-chunk operations.
3. **Attention output:** Here, we make the assumption that the routing δ selects chunks uniformly at random to help with analysis. Under this simplifying condition, the computation cost in expectation is $(1-\delta)T(T+1)(D_k+D_v)$.

When combined, the total compute required for an ORYX forward pass is $2TD_kD_v + 2TD_kD_v/C + \delta(2TCD_k + 2TCD_v + 2TD_kD_v) + (1-\delta)T(T+1)(D_k+D_v)$ which is approximately $2TC^2(1+3\delta) + 2(1-\delta)T^2C$ when assuming $C = D_k = D_v$ and ignoring lower order terms.

Compute Tradeoff. When comparing the general FLOPs usage of ORYX and pure attention, ORYX uses fewer FLOPs when $(1-\delta) + (1+3\delta)C/T < 1$. Rearranging, we find that in general, if $T > (\frac{1}{\delta} + 3)C$, the shared block utilizes less compute than pure attention. This result is intuitive as a large δ would reduce the majority of the quadratic attention compute required, while a small δ would require most of the attention computation plus the recurrent state overhead. In our one-to-three attention-to-linear split setting with $C = 128$, the ORYX block would utilize less FLOPs than that of softmax attention if the context length were 2048 ($T = 2048 \geq 555$).

D ADDITIONAL MODE SWITCHING RESULTS

For instance, ORYX-TM models trained with sequence-level switching display the ability to freely switch from softmax-attention to Mamba-2 processing and vice versa without performance degradation at the 130M and 1.4B scale. However, the 380M and 810M models can only switch from softmax-attention to Mamba-2 without issue: switching from Mamba-2 to softmax-attention leads to notable performance degradations as measured by a spike in token-index perplexity. We display the perplexity of all Chinchilla-token trained models in Figure 5.

The cause of this divergence remains unclear, although our ablations suggest this is not an architectural issue. Reducing learning rate mitigates the divergence and increasing training does not solve the issue. We hypothesize this is due to the switch from softmax-attention to Mamba-2 involves switching from a more expressive to less expressive mixer which is easier than vice versa, which is why we see the ability appear naturally more often.

We find that chunked mixed-mode training enables switching at all scales (Figure 6), switching at non-chunk boundaries, i.e., non-multiples of 128 (Figure 8), and multiple switches within a single sequence (Figure 9).

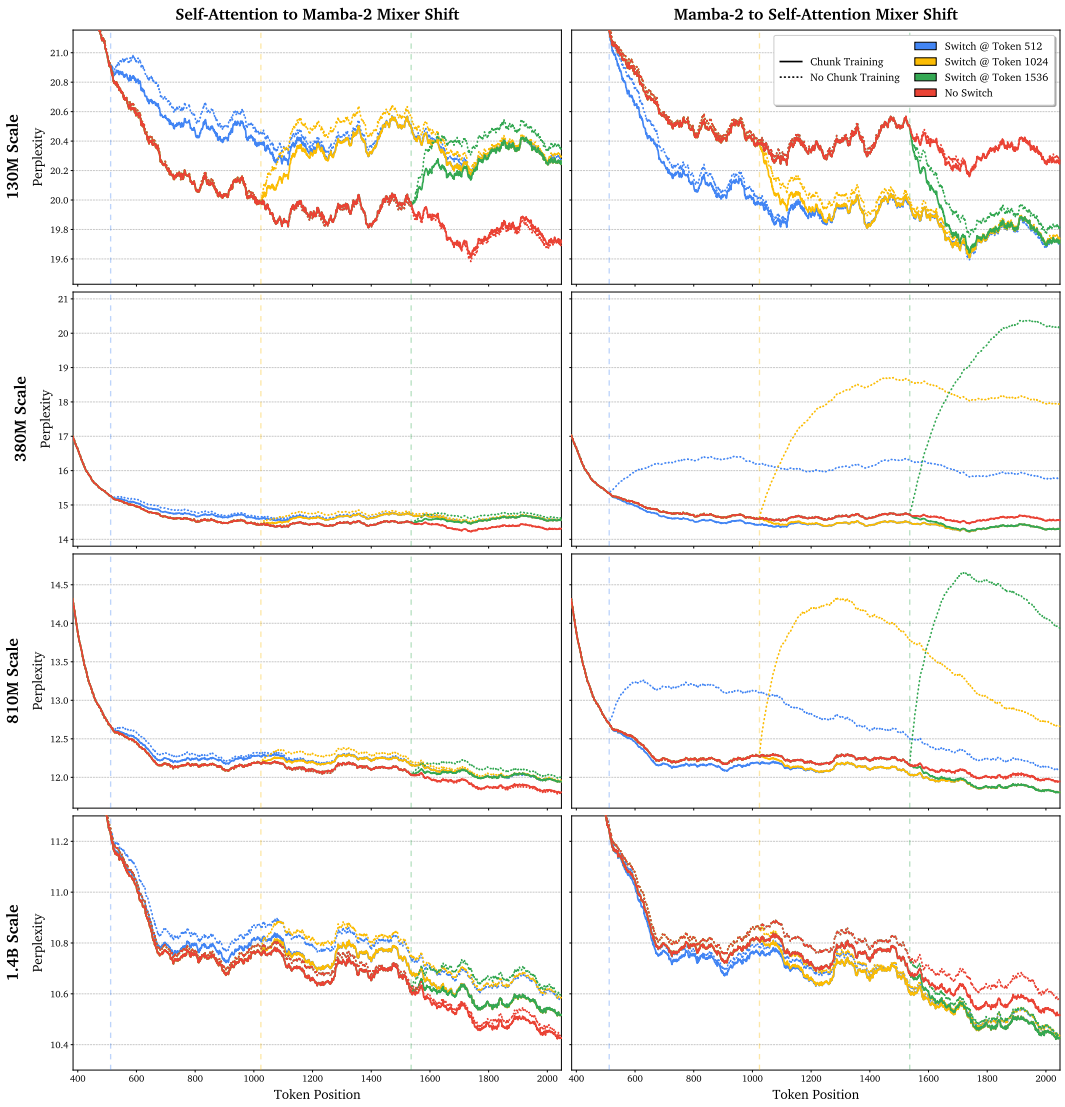


Figure 5: Smoothed perplexity across token index position for ORYX-TM models at all four scales trained with and without chunk-level switching.

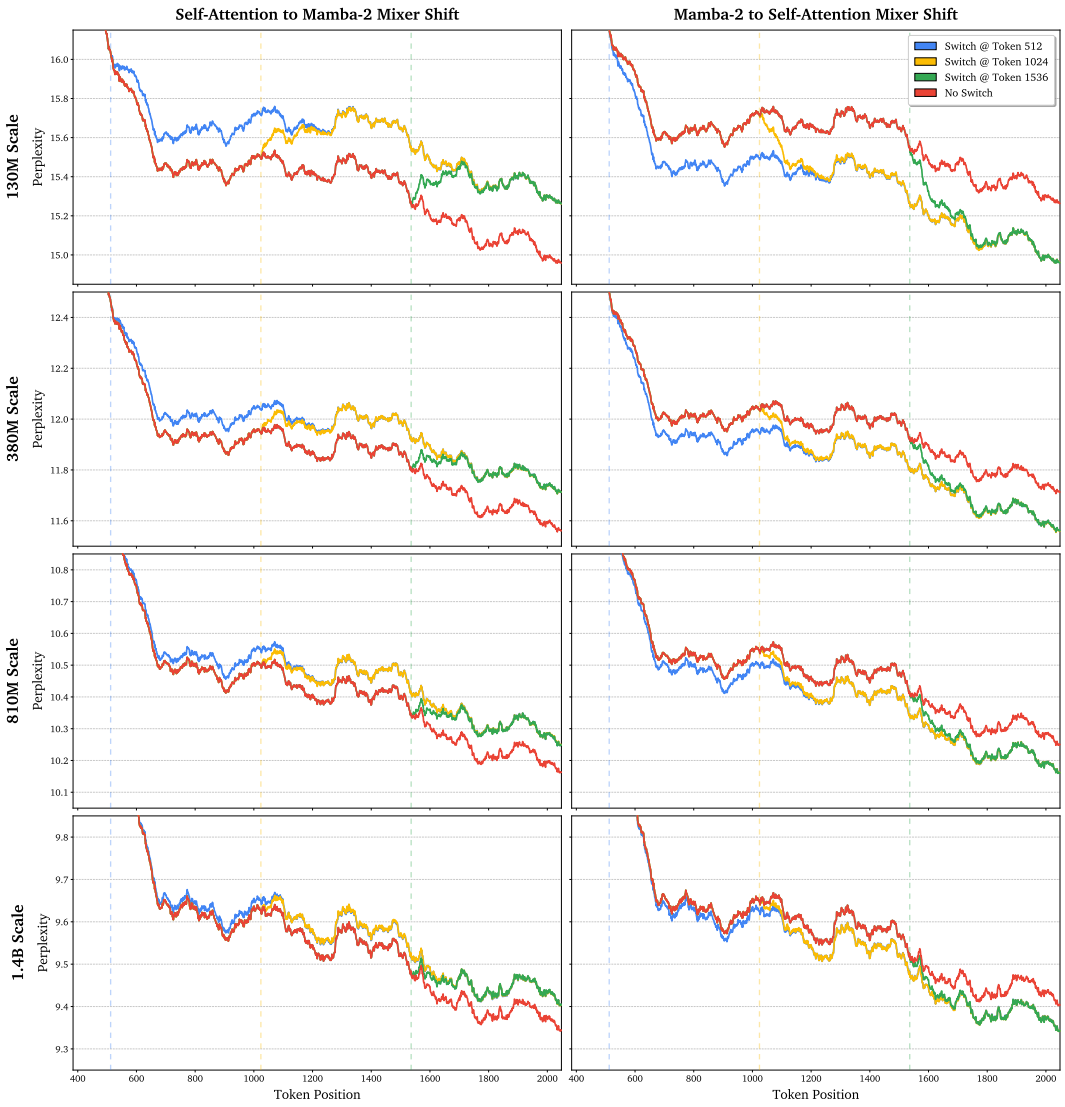


Figure 6: Smoothed perplexity across token index position for all chunked mixed-mode ORYX-TM models across scale.

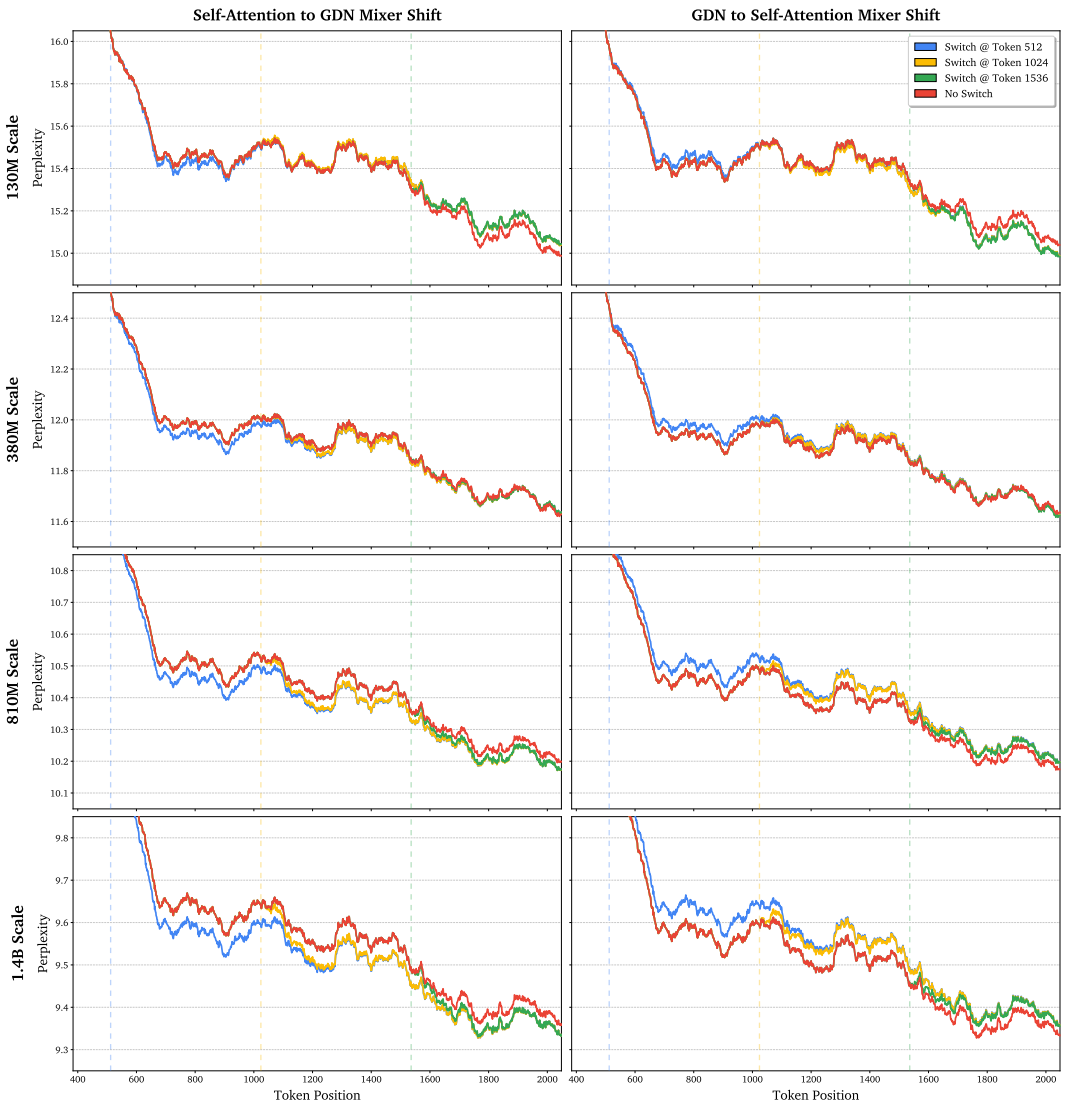


Figure 7: Smoothed perplexity across token index position for all chunked mixed-mode pretrained ORYX-TG models across scale.



Figure 8: Smoothed perplexity across token index position at non-chunk boundaries for chunked mixed-mode pretrained ORYX-TM 1.4B model.



Figure 9: Smoothed perplexity across token index position for multiple switches for chunked mixed-mode pretrained ORYX-TM 1.4B model.

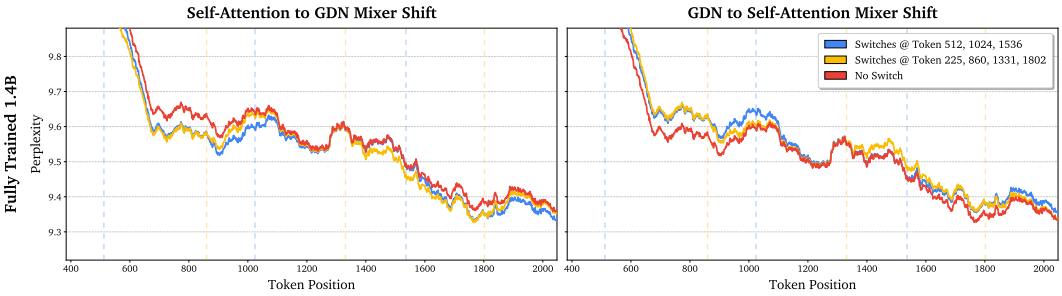


Figure 10: Smoothed perplexity across token index position for multiple switches for chunked mixed-mode pretrained ORYX-TG 1.4B model.