Multi-Mixer Models: Flexible Sequence Modeling with Shared Representations

Anonymous Authors[1]

Abstract

Softmax attention is the cornerstone of modern large language models but is hindered by its linear memory and quadratic compute requirements. Linear recurrent models, such as linear attention and state space models, have become widely studied as alternatives due to their linear compute and constant memory requirements. While these sub-quadratic methods 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, resulting in a growing body of work on hybrid architectures. In this work, we propose ORYX, which explores a new axis of developing hybrid models: across the sequence, the model has the ability to operate with different mixers, e.g., quadratic attention and linear recurrences, rather than fixing a single mixer throughout. ORYX ties more than 90% of its parameters across mixer types, enabling attention and recurrent modes to operate over shared internal representations. We validate our design with Mamba-2 and Gated DeltaNet variants up to the 1.4B model scale. Under fixed token budgets and a mixed-training strategy, all modes of the 1.4B ORYX outperform their respective baselines by at least **0.7** percentage points on average on downstream language modeling evaluations. On retrieval tasks, ORYX achieves performance comparable to the Transformer baseline even when processing only a tiny fraction ($<10\%$) of the tokens in the attention mode. These results suggest that attention and linear recurrent models can share internal representations, and motivate sequence-axis hybridization as a promising direction for efficient long-context language models.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

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 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). These 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, which interleave softmax attention and linear layers, and intra-layer designs, which fuse the output of attention and linear mechanisms within each layer. In both cases, the computational cost per token and capabilities are largely determined by the selected architecture prior to training. In practice, however, the required modeling capabilities and desired trade-off vary by task: retrieval and reasoning-heavy tasks may benefit from richer attention-based context utilization, whereas standard language generation tasks may be adequately served by linear recurrent computation. This suggests a complementary form of hybridization: instead of choosing a fixed mixture of mechanisms, can a model operate with different mixers across the sequence axis?

In this work, we propose ORYX, a sequence-axis hybrid architecture that supports operating with both quadratic-attention and linear-recurrent regimes throughout a sequence (Figure 1), and a chunked mixed-mode training strategy that enables flexible switching during deployment (Section 3). A

[1]In this work, we use attention to refer to the canonical quadratic softmax attention mechanism, and use specified names for the sub-quadratic variants, e.g., linear attention.

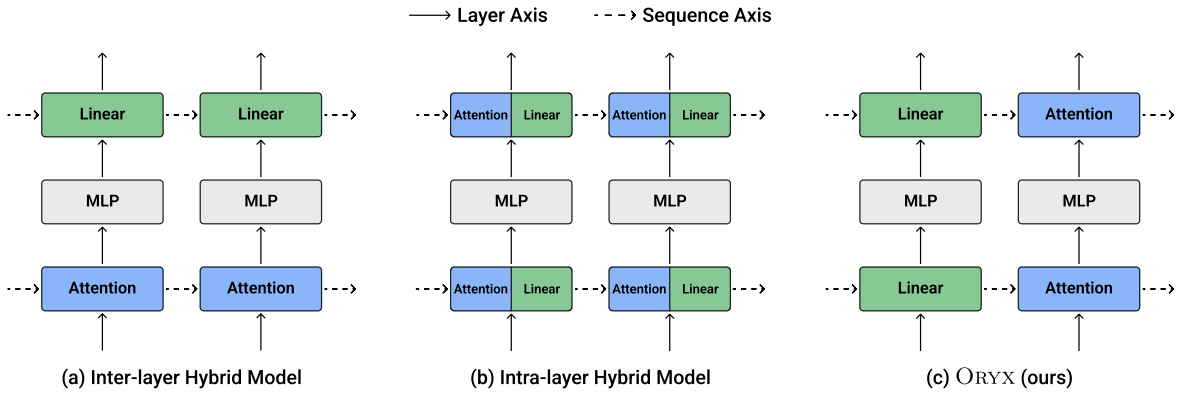

Figure 1. Comparison of different hybrid architectures. (a) Inter-layer hybrid models interleave different mixers along the layer axis; (b) Intra-layer hybrid models fuse different mixers within a single layer; (c) ORYX is a sequence-axis hybrid model that can switch between different mixers across the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

key challenge in supporting mode switching is that attention and recurrent mixers parameterize and update their state differently. To bridge these mechanisms, ORYX maintains the KV cache and linear recurrent state, updating both jointly at each timestep with shared key-value pairs obtained from tied projections. As a result, both mechanisms build their state from a shared representation space, avoiding separate state-update features for each mixer. Consequently, when the mixer mechanism switches, the new selected mixer can continue from a compatible state accumulated over all preceding tokens. The flexibility in mixer selection during inference could enable compute allocation at the prompt or token level. For example, reasoning traces could be generated with the linear mixer for lower-latency, and the answer could use the attention mode for better retrieval.

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. However, we underscore that our overall design is not specific to these choices and can extend to other mixers that admit comparable key-value association viewpoints. Across scale, our ORYX models can switch between attention and linear processing while incurring little to no degradation in output quality (Figure 3). Under matched parameters and *training token budgets*, ORYX remains competitive with pure softmax attention, Mamba-2, and GDN baselines on language modeling performance. At 1.4B parameters, 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). We also test the capability of ORYX on a variety of real world retrieval tasks and synthetic retrieval tasks. ORYX is able to achieve comparable retrieval performance compared to the Transformer baseline with $<10\%$ of the tokens processed in the attention mode. Using mixed-inference mode, ORYX significantly surpasses the linear model baselines by a margin of at least 8.6 percentage points on real-world retrieval tasks and at least 38.6 percentage points on needle-in-a-haystack (NIAH) tests (Table 3). Our results demonstrate that ORYX is a promising method for improving 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 cover a background on sequence mixers and their connections in Section 2. We use their commonality to motivate the main architectural design of the ORYX layer in Section 3. Section 4 evaluates the language modeling and mode-switching ability of the ORYX model.

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., the softmax attention operation) 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., 2023) 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). Under this view, each mechanism main-

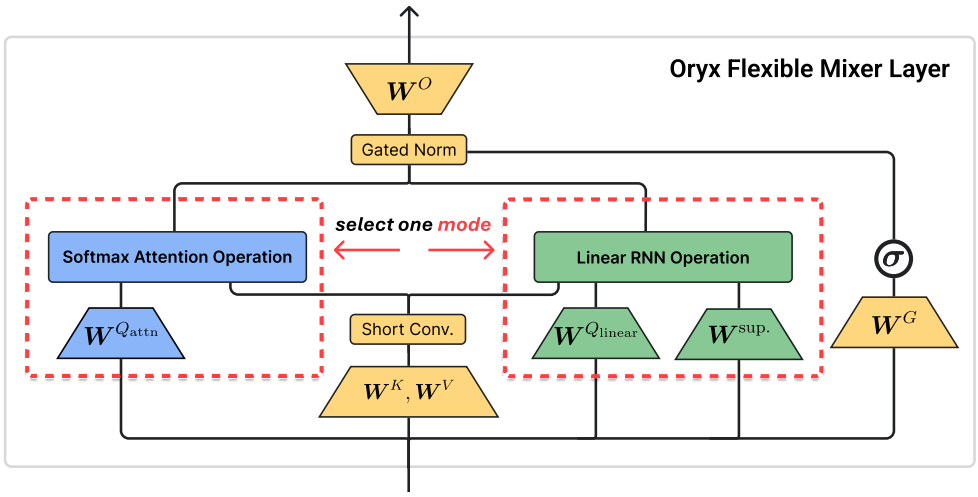

Figure 2. General ORYX layer. The layer shares the core representations between softmax attention and a linear recurrent mechanism through tied key-value projections and uses additional components critical to the linear mechanism's performance, e.g., the short convolution and gate. During a forward pass, 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.

tains a memory of key-value associations and uses queries to retrieve relevant values from that 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. We discuss the various instantiations of this viewpoint in Section B.

3. Designing the ORYX Shared Layer

In this section, we describe the sequence-mixing ORYX shared-layer that supports operating in both softmax attention and linear recurrent mechanisms. The resulting layer 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 (Section 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 queries can also be shared, our empirical results suggest that tying all three core representations across mixers hinders model performance (see details in Section C.1), 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 queries to extract mode-specific information, a belief supported by empirical results.

Incorporating Additional Linear Layer 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 layer computes[2]

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

where $\boldsymbol{W}^{\text{sup.}}$ denotes the additional mixer-specific supporting parameters, 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, 2023), and $\boldsymbol{W}^G \in \mathbb{R}^{D\times D_v}$ is the shared gate

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

projection. The exact normalization and supporting parameters depends on the linear mechanism used; we detail the specifics of our variants in Section C. 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 layer 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 layer output, the joint update allows both states to retain the same token history. However, one issue is that different mixers often use differing head structures. For instance, modern Transformers use softmax 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 softmax attention, i.e., MHA in our experiments. Despite this constraint applied on the linear mixers, we find that they still remain empirically effective on modeling.

Chunked Mixed-Mode Training. To enable robust mode switching capabilities at inference time (see ablation study in Section C.1), we train ORYX with chunked mixed-mode training. 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 mechanism. All layers use the same chunk assignment, and our experiments find that a 1:3 attention-to-linear chunk training ratio balances the performance of both modes.

4. Empirical Results and Properties of ORYX

Section 4.1 discusses the language modeling and retrieval results of our models when operating using only one mixer mode. Section 4.2 then explores the mode switching capabilities of the model, and Section C.1 ablates architectural and training design choices. Setup is in Section D.

4.1. Language Modeling and Retrieval with Isolated Mixer Modes

4.1.1. LANGUAGE MODELING PERFORMANCE

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 the usefulness of each mode 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.

4.1.2. RETRIEVAL CAPABILITIES

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 twice the accuracy on NIAH-3 and more than half 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 respectively Mamba-2 and GDN baselines, which suggests that shared-layer model and training can improve downstream capabilities.

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 F display the same behaviors for 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 context and the remaining tokens as the prompt. Table 3 shows that mode-

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. Full results can be found at Table 9.

	Family	Mixer	LAMB. ppl ↓	LAMB. acc ↑	HellaS. acc_n ↑	PIQA acc ↑	Arc-E acc ↑	Arc-C acc_n ↑	WinoGr. acc ↑	OBQA acc ↑	Average 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**

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 layers, and ORYX-TG, Transformer/Gated DeltaNet. The isolated modes of ORYX remain comparable in to their corresponding baselines.

		Real-World Retrieval							Synthetic Retrieval			
Family	**Mixer**	SWDE	SQuAD	FDA	TQA	NQ	DROP	Mean	NIAH-1	NIAH-2	NIAH-3	Mean
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		50.1	41.7	57.6	63.9	28.3	22.5	44.0	99.8	99.4	81.6	93.6
ORYX-TG		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		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		29.3	37.9	18.2	64.0	23.5	22.0	32.5	99.4	60.2	82.8	80.8

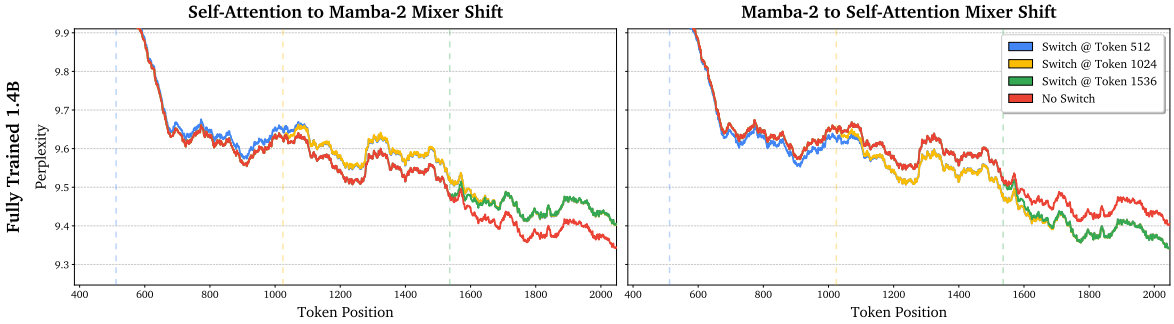

Figure 3. Smoothed token-level perplexity across token position for pretrained 1.4B ORYX-TM model trained with chunked mixed-mode training. After switching from softmax attention to Mamba-2 and vice versa at different positions, perplexity rapidly approaches the corresponding no-switch baseline, indicating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

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 compared to the Transformer baseline and significantly surpassing their linear model baselines. In particular, with the 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

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 and generation (**Prompt + Gen**). Baselines use the same mixer for the entire sequence, while ORYX use a different mixer for context prefill (**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

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 model configurations.

This new capability may enable use-cases such as models that change mechanisms depending on the task, e.g., paradigms where large portions of thinking traces are generated with the linear mode and verified with the attention one. While ORYX 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.

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 an autoregressive 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, preventing the flexible usage of various mixers depending on the sequence context. This 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 language modeling and can serve as a basis for learned routing or learned sparsity. We further cover related works in Section A.

6. Discussion and Conclusion

In our work, we introduce ORYX, an 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 across model scales, achieves strong language modeling performance, preserves retrieval capabilities, and supports mode switching during prefill and generation. One consideration 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 to be used. Although the linear state's memory cost is dominated by the KV cache at longer lengths, 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 using FLOPs saved as the reward. In addition, because the ORYX design can naturally incorporate other sequence mixers, certain layers that share similar head structures to softmax attention or can utilize its KV cache, e.g., 2-simplicial attention (Roy et al., 2025), may further improve performance. Finally, the training dynamics of our multi-mixer models remain largely unexplored and may be vastly different than that of standard models. As the number of mixers or model scale increases, it remains to be seen whether techniques, such as discriminative learning rates or specialized training objectives, may become more important.

More broadly, the sequence-axis hybrid paradigm opens a new design space for adaptive compute. For instance, linear recurrent modes could serve as efficient drafters for speculative decoding or could rapidly generate the longer intermediate reasoning traces prior to fully synthesizing the answer with quadratic attention. 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.

Impact 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 which we feel must be specifically highlighted here.

References

Arora, S., Timalsina, A., Singhal, A., Spector, B., Eyuboglu, S., Zhao, X., Rao, A., Rudra, A., and Ré, C. Just read twice: closing the recall gap for recurrent language models, 2024. URL https://arxiv.org/abs/2407.05483.

Arora, S., Eyuboglu, S., Zhang, M., Timalsina, A., Alberti, S., Zinsley, D., Zou, J., Rudra, A., and Ré, C. Simple linear attention language models balance the recall-throughput tradeoff, 2025a. URL https://arxiv.org/abs/2402.18668.

Arora, S., Yang, B., Eyuboglu, S., Narayan, A., Hojel, A., Trummer, I., and Ré, C. Language models enable simple systems for generating structured views of heterogeneous data lakes, 2025b. URL https://arxiv.org/abs/2304.09433.

Behrouz, A., Zhong, P., and Mirrokni, V. Titans: Learning to memorize at test time, 2024. URL https://arxiv.org/abs/2501.00663.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. Rethinking attention with performers, 2022. URL https://arxiv.org/abs/2009.14794.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Dong, X., Fu, Y., Diao, S., Byeon, W., Chen, Z., Mahabaleshwarkar, A. S., Liu, S.-Y., Keirsbilck, M. V., Chen, M.-H., Suhara, Y., Lin, Y., Kautz, J., and Molchanov, P. Hymba: A hybrid-head architecture for small language models, 2024. URL https://arxiv.org/abs/2411.13676.

Du, J., Sun, W., Lan, D., Hu, J., and Cheng, Y. Mom: Linear sequence modeling with mixture-of-memories, 2025. URL https://arxiv.org/abs/2502.13685.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs, 2019. URL https://arxiv.org/abs/1903.00161.

Fang, Y., Yu, W., Zhong, S., Ye, Q., Xiong, X., and Wei, L. Artificial hippocampus networks for efficient long-context modeling, 2025. URL https://arxiv.org/abs/2510.07318.

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

Gemma Team, Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., bastien Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.-T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A. M., Goedeckemeyer, A., Saade, A., Feng, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A. S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C. L., Choquette-Choo, C. A., Carey, C., Brick,

C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D. S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F., Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Plucińska, H., Singh, H., Mehta, H., Lehri, H. T., Hazimeh, H., Ballantyne, I., Szpektor, I., Nardini, I., Pouget-Abadie, J., Chan, J., Stanton, J., Wieting, J., Lai, J., Orbay, J., Fernandez, J., Newlan, J., yeong Ji, J., Singh, J., Black, K., Yu, K., Hui, K., Vodrahalli, K., Greff, K., Qiu, L., Valentine, M., Coelho, M., Ritter, M., Hoffman, M., Watson, M., Chaturvedi, M., Moynihan, M., Ma, M., Babar, N., Noy, N., Byrd, N., Roy, N., Momchev, N., Chauhan, N., Sachdeva, N., Bunyan, O., Botarda, P., Caron, P., Rubenstein, P. K., Culliton, P., Schmid, P., Sessa, P. G., Xu, P., Stanczyk, P., Tafti, P., Shivanna, R., Wu, R., Pan, R., Rokni, R., Willoughby, R., Vallu, R., Mullins, R., Jerome, S., Smoot, S., Girgin, S., Iqbal, S., Reddy, S., Sheth, S., Põder, S., Bhatnagar, S., Panyam, S. R., Eiger, S., Zhang, S., Liu, T., Yacovone, T., Liechty, T., Kalra, U., Evci, U., Misra, V., Roseberry, V., Feinberg, V., Kolesnikov, V., Han, W., Kwon, W., Chen, X., Chow, Y., Zhu, Y., Wei, Z., Egyed, Z., Cotruta, V., Giang, M., Kirk, P., Rao, A., Black, K., Babar, N., Lo, J., Moreira, E., Martins, L. G., Sanseviero, O., Gonzalez, L., Gleicher, Z., Warkentin, T., Mirrokni, V., Senter, E., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Matias, Y., Sculley, D., Petrov, S., Fiedel, N., Shazeer, N., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Alayrac, J.-B., Anil, R., Dmitry, Lepikhin, Borgeaud, S., Bachem, O., Joulin, A., Andreev, A., Hardin, C., Dadashi, R., and Hussenot, L. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

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

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces, 2022. URL https://arxiv.org/abs/2111.00396.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. Ruler: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

Hu, J., Pan, Y., Du, J., Lan, D., Tang, X., Wen, Q., Liang, Y., and Sun, W. Comba: Improving bilinear rnns with closed-loop control, 2025. URL https://arxiv.org/abs/2506.02475.

Hua, W., Dai, Z., Liu, H., and Le, Q. V. Transformer quality in linear time, 2022. URL https://arxiv.org/abs/2202.10447.

Hwang, S., Wang, B., and Gu, A. 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.

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

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

Kimi Team, Zhang, Y., Lin, Z., Yao, X., Hu, J., Meng, F., Liu, C., Men, X., Yang, S., Li, Z., Li, W., Lu, E., Liu, W., Chen, Y., Xu, W., Yu, L., Wang, Y., Fan, Y., Zhong, L., Yuan, E., Zhang, D., Zhang, Y., Liu, T. Y., Wang, H., Fang, S., He, W., Liu, S., Li, Y., Su, J., Qiu, J., Pang, B., Yan, J., Jiang, Z., Huang, W., Yin, B., You, J., Wei, C., Wang, Z., Hong, C., Chen, Y., Chen, G., Wang, Y., Zheng, H., Wang, F., Liu, Y., Dong, M., Zhang, Z., Pan, S., Wu, W., Wu, Y., Guan, L., Tao, J., Fu, G., Xu, X., Wang, Y., Lai, G., Wu, Y., Zhou, X., Yang, Z., and Du, Y. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. 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/.

Lahoti, A., Li, K. Y., Chen, B., Wang, C., Bick, A., Kolter, J. Z., Dao, T., and Gu, A. Mamba-3: Improved sequence modeling using state space principles, 2026. URL https://arxiv.org/abs/2603.15569.

Li, Y., Xie, R., Yang, Z., Sun, X., Li, S., Han, W., Kang, Z., Cheng, Y., Xu, C., Wang, D., and Jiang, J. Transmamba: Flexibly switching between transformer and mamba, 2025. URL https://arxiv.org/abs/2503.24067.

Lin, X. V., Shrivastava, A., Luo, L., Iyer, S., Lewis, M., Ghosh, G., Zettlemoyer, L., and Aghajanyan, A. Moma: Efficient early-fusion pre-training with mixture of modality-aware experts, 2024. URL https://arxiv.org/abs/2407.21770.

Liu, B., Wang, R., Wu, L., Feng, Y., Stone, P., and Liu, Q. Longhorn: State space models are amortized online learners, 2024. URL https://arxiv.org/abs/2407.14207.

Lockard, C., Shiralkar, P., and Dong, X. L. OpenCeres: When open information extraction meets the semi-structured web. In Burstein, J., Doran, C., and Solorio, T. (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/.

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

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

Lufkin, L., Figliolia, T., Millidge, B., and Krishnamurthy, K. Hybrid associative memories, 2026. URL https://arxiv.org/abs/2603.22325.

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

NVIDIA, :, Blakeman, A., Grattafiori, A., Basant, A., Gupta, A., Khattar, A., Renduchintala, A., Vavre, A., Shukla, A., Bercovich, A., Ficek, A., Shaposhnikov, A., Kondratenko, A., Bukharin, A., Milesi, A., Taghibakhshi, A., Liu, A., Barton, A., Mahabaleshwarkar, A. S., Klein, A., Zuker, A., Geifman, A., Shen, A., Bhiwandiwalla, A., Tao, A., Agrusa, A., Verma, A., Guan, A., Mandarwal, A., Mehta, A., Aithal, A., Poojary, A., Ahamed, A., Mishra, A., Thekkumpate, A. K., Dattagupta, A., Zhu, B., Sadeghi, B., Simkin, B., Lanir, B., Schifferer, B., Nushi, B., Kartal, B., Rouhani, B. D., Ginsburg, B., Norick, B., Soubasis, B., Kisacanin, B., Yu, B., Catanzaro, B., del Mundo, C., Hwang, C., Wang, C., Hsieh, C.-P., Zhang, C., Yu, C., Mungekar, C., Patel, C., Alexiuk, C., Parisien, C., Neale, C., Meurillon, C., Mosk-Aoyama, D., Su, D., Corneil, D., Afrimi, D., Lo, D., Rohrer, D., Serebrenik, D., Gitman, D., Levy, D., Stosic, D., Mosallanezhad, D., Narayanan, D., Nathawani, D., Rekesh, D., Yared, D., Kakwani, D., Ahn, D., Riach, D., Stosic, D., Minasyan, E., Lin, E., Long, E., Long, E. P., Segal, E., Lantz, E., Evans, E., Ning, E., Chung, E., Harper, E., Tramel, E., Galinkin, E., Pounds, E., Briones, E., Bakhturina, E., Tsykunov, E., Ladhak, F., Wang, F., Jia, F., Soares, F., Chen, F., Galko, F., Sun, F., Siino, F., Agam, G. H., Ajjanagadde, G., Bhatt, G., Prasad, G., Armstrong, G., Shen, G., Batmaz, G., Nalbandyan, G., Qian, H., Sharma, H., Ross, H., Ngo, H., Hum, H., Sahota, H., Wang, H., Soni, H., Upadhyay, H., Mao, H., Nguyen, H. C., Nguyen, H. Q., Cunningham, I., Galil, I., Shahaf, I., Gitman, I., Loshchilov, I., Schen, I., Levy, I., Moshkov, I., Golan, I., Putterman, I., Kautz, J., Scowcroft, J. P., Casper, J., Mitra, J., Glick, J., Chen, J., Oliver, J., Zhang, J., Zeng, J., Lou, J., Zhang, J., Choi, J., Huang, J., Conway, J., Guman, J., Kamalu, J., Greco, J., Cohen, J., Jennings, J., Daw, J., Vialard, J. V., Yi, J., Parmar, J., Xu, K., Zhu, K., Briski, K., Cheung, K., Luna, K., Wyss, K., Santhanam, K., Shih, K., Kong, K., Bhardwaj, K., Shankar, K., Puvvada, K. C., Pawelec, K., Anik, K., McAfee, L., Sleiman, L., Derczynski, L., Ding, L., Wei, L., Liebenwein, L., Vega, L., Grover, M., Segbroeck, M. V., de Melo, M. R., Nazemi, M., Sreedhar, M. N., Kilaru, M., Ashkenazi, M., Romeijn, M., Chochowski, M., Cai, M., Kliegl, M., Moosaei, M., Kulka, M., Novikov, M., Samadi, M., Corpuz, M., Wang, M., Price, M., Andersch, M., Boone, M., Evans, M., Martinez, M., Khona, M., Chrzanowski, M., Lee, M., Dabbah, M., Shoeybi, M., Patwary, M., Mulepati, N., Nabwani, N., Hereth, N., Assaf, N., Habibi, N., Zmora, N., Haber, N., Sessions, N., Bhatia, N., Jukar, N., Pope, N., Ludwig, N., Tajbakhsh, N., Ailon, N., Juluru, N., Sharma, N., Hrinchuk, O., Kuchaiev, O., Delalleau, O., Olabiyi, O., Argov, O. U., Puny, O., Tropp, O., Xie, O., Chadha, P., Shamis, P., Gibbons, P., Molchanov, P., Morkisz, P., Dykas, P., Jin, P., Xu, P., Januszewski, P., Thombre, P. P., Varshney, P., Gundecha, P., Tredak, P., Miao, Q., Wan, Q., Mahabadi, R. K., Garg, R., El-Yaniv, R., Zilberstein, R., Shafipour, R., Harang, R., Izzo, R., Shahbazyan, R., Garg, R., Borkar, R., Gala, R., Islam, R., Hesse, R., Waleffe, R., Watve, R., Koren, R., Zhang, R., Hewett, R., Hewett, R. J., Prenger, R., Timbrook, R., Mahdavi, S., Modi, S., Kriman, S., Lim, S., Kariyappa, S., Satheesh, S.,

Kaji, S., Pasumarthi, S., Muralidharan, S., Narentharen, S., Narenthiran, S., Bak, S., Kashirsky, S., Poulos, S., Mor, S., Ramasamy, S., Acharya, S., Ghosh, S., Sreenivas, S. T., Thomas, S., Fan, S., Gopal, S., Prabhumoye, S., Pachori, S., Toshniwal, S., Ding, S., Singh, S., Sun, S., Ithape, S., Majumdar, S., Singhal, S., Sergienko, S., Alborghetti, S., Ge, S., Devare, S. D., Barua, S. K., Panguluri, S., Gupta, S., Priyadarshi, S., Akter, S. N., Bui, T., Ene, T.-D., Kong, T., Do, T., Blankevoort, T., Moon, T., Balough, T., Asida, T., Natan, T. B., Ronen, T., Konuk, T., Vashishth, T., Karpas, U., De, U., Noorozi, V., Noroozi, V., Srinivasan, V., Elango, V., Cui, V., Korthikanti, V., Rao, V., Kurin, V., Lavrukhin, V., Anisimov, V., Jiang, W., Ahmad, W. U., Du, W., Ping, W., Zhou, W., Jennings, W., Zhang, W., Prazuch, W., Ren, X., Karnati, Y., Choi, Y., Meyer, Y., Wu, Y.-F., Zhang, Y., Qin, Y., Lin, Y., Geifman, Y., Fu, Y., Subara, Y., Suhara, Y., Gao, Y., Moshe, Z., Dong, Z., Zhu, Z., Liu, Z., Chen, Z., and Yan, Z. Nvidia nemotron 3: Efficient and open intelligence, 2025. URL https://arxiv.org/abs/2512.20856.

OpenAI, :, Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., Barak, B., Bennett, A., Bertao, T., Brett, N., Brevdo, E., Brockman, G., Bubeck, S., Chang, C., Chen, K., Chen, M., Cheung, E., Clark, A., Cook, D., Dukhan, M., Dvorak, C., Fives, K., Fomenko, V., Garipov, T., Georgiev, K., Glaese, M., Gogineni, T., Goucher, A., Gross, L., Guzman, K. G., Hallman, J., Hehir, J., Heidecke, J., Helyar, A., Hu, H., Huet, R., Huh, J., Jain, S., Johnson, Z., Koch, C., Kofman, I., Kundel, D., Kwon, J., Kyrylov, V., Le, E. Y., Leclerc, G., Lennon, J. P., Lessans, S., Lezcano-Casado, M., Li, Y., Li, Z., Lin, J., Liss, J., Lily, Liu, Liu, J., Lu, K., Lu, C., Martinovic, Z., McCallum, L., McGrath, J., McKinney, S., McLaughlin, A., Mei, S., Mostovoy, S., Mu, T., Myles, G., Neitz, A., Nichol, A., Pachocki, J., Paino, A., Palmie, D., Pantuliano, A., Parascandolo, G., Park, J., Pathak, L., Paz, C., Peran, L., Pimenov, D., Pokrass, M., Proehl, E., Qiu, H., Raila, G., Raso, F., Ren, H., Richardson, K., Robinson, D., Rotsted, B., Salman, H., Sanjeev, S., Schwarzer, M., Sculley, D., Sikchi, H., Simon, K., Singhal, K., Song, Y., Stuckey, D., Sun, Z., Tillet, P., Toizer, S., Tsimpourlas, F., Vyas, N., Wallace, E., Wang, X., Wang, M., Watkins, O., Weil, K., Wendling, A., Whinnery, K., Whitney, C., Wong, H., Yang, L., Yang, Y., Yasunaga, M., Ying, K., Zaremba, W., Zhan, W., Zhang, C., Zhang, B., Zhang, E., and Zhao, S. gpt-oss-120b & gpt-oss-20b Model Card, 2025. URL https://arxiv.org/abs/2508.10925.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The lambada dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., and Kong, L. Random feature attention, 2021. URL https://arxiv.org/abs/2103.02143.

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

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners, 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for squad, 2018.

Roy, A., Chou, T., Duvvuri, S. S., Chen, S., Yu, J., Wang, X., Zaheer, M., and Anil, R. Fast and simplex: 2-simplicial attention in triton, 2025. URL https://arxiv.org/abs/2507.02754.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers, 2021. URL https://arxiv.org/abs/2102.11174.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

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

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

Tandon, A., Dalal, K., Li, X., Koceja, D., Rød, M., Buchanan, S., Wang, X., Leskovec, J., Koyejo, S., Hashimoto, T., Guestrin, C., McCaleb, J., Choi, Y., and Sun, Y. End-to-end test-time training for long context, 2025. URL https://arxiv.org/abs/2512.23675.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Trockman, A., Harutyunyan, H., Kolter, J. Z., Kumar, S., and Bhojanapalli, S. Mimetic initialization helps state space models learn to recall. *arXiv preprint arXiv:2410.11135*, 2024.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Waleffe, R., Byeon, W., Riach, D., Norick, B., Korthikanti, V., Dao, T., Gu, A., Hatamizadeh, A., Singh, S., Narayanan, D., Kulshreshtha, G., Singh, V., Casper, J., Kautz, J., Shoeybi, M., and Catanzaro, B. An empirical study of mamba-based language models, 2024. URL https://arxiv.org/abs/2406.07887.

Wang, K. A., Shi, J., and Fox, E. B. Test-time regression: a unifying framework for designing sequence models with associative memory, 2025. URL https://arxiv.org/abs/2501.12352.

Wang, L., Gao, H., Zhao, C., Sun, X., and Dai, D. Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2024. URL https://arxiv.org/abs/2408.15664.

Wu, X., Huang, S., Wang, W., and Wei, F. Multi-head mixture-of-experts, 2024. URL https://arxiv.org/abs/2404.15045.

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

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention, 2021. URL https://arxiv.org/abs/2102.03902.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated linear attention transformers with hardware-efficient training, 2024. URL https://arxiv.org/abs/2312.06635.

Yang, S., Kautz, J., and Hatamizadeh, A. Gated delta networks: Improving mamba2 with delta rule, 2025a. URL https://arxiv.org/abs/2412.06464.

Yang, S., Wang, B., Zhang, Y., Shen, Y., and Kim, Y. Parallelizing linear transformers with the delta rule over sequence length, 2025b. URL https://arxiv.org/abs/2406.06484.

Yuan, J., Gao, H., Dai, D., Luo, J., Zhao, L., Zhang, Z., Xie, Z., Wei, Y. X., Wang, L., Xiao, Z., Wang, Y., Ruan, C., Zhang, M., Liang, W., and Zeng, W. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025. URL https://arxiv.org/abs/2502.11089.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Zhang, J., Wang, H., Jiang, K., Zheng, K., Jiang, Y., Stoica, I., Chen, J., Zhu, J., and Gonzalez, J. E. Sla2: Sparse-linear attention with learnable routing and qat, 2026. URL https://arxiv.org/abs/2602.12675.

Zhang, T., Bi, S., Hong, Y., Zhang, K., Luan, F., Yang, S., Sunkavalli, K., Freeman, W. T., and Tan, H. Test-time training done right, 2025. URL https://arxiv.org/abs/2505.23884.

Zhang, X., Shen, Y., Huang, Z., Zhou, J., Rong, W., and Xiong, Z. Mixture of attention heads: Selecting attention heads per token, 2022. URL https://arxiv.org/abs/2210.05144.

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 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). 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 towards 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 layer has parallels to 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).

B. Additional Preliminaries

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 $\boldsymbol{V}_{\leq t}$ according to the softmax-normalized similarity between the current query $\boldsymbol{q}_t$ and past keys $\boldsymbol{K}_{\leq t}$: $\mathrm{softmax}\big(\boldsymbol{q}_t\boldsymbol{K}_{\leq t}^\top\big)\boldsymbol{V}_{\leq t}$. [3] 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 layer output in the original input dimension, $\boldsymbol{Y} = \boldsymbol{O}\boldsymbol{W}^O$. As this step is present in other models as well, we will ignore it from here on out.

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

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 $\boldsymbol{A}_t \in \mathbb{R}^{D_k \times D_k}$ evolves 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 such instantiation of a linear RNN, using a scalar times identity transition, $\boldsymbol{A}_t = \alpha_t \boldsymbol{I}$, where α_t is an input-dependent decay factor. [4] Its parallel form can be written using a decay-based lower-triangular mask $\boldsymbol{\Gamma}$ applied with a Hadamard product ($\circ$), which draws connections to attention variants (Dao & Gu, 2024) and highlights the value retrieval mechanism.

$$\boldsymbol{O} = \text{Mixer}_{\text{Mamba-2}}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{X}) := \left(\boldsymbol{\Gamma} \circ (\boldsymbol{Q}\boldsymbol{K}^\top)\right) \boldsymbol{V}, \quad \boldsymbol{\Gamma} = \begin{bmatrix} 1 & & & \\ \alpha_1 & 1 & & \\ \alpha_2\alpha_1 & \alpha_2 & 1 & \\ \vdots & & & \ddots \end{bmatrix}$$

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 layer in Section 3, where tied projections are used to update both attention and recurrent states from shared representations.

C. ORYX Architecture Details and Ablations

C.1. 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 layer components on performance and the importance of the chunked mixed-mode training on mode-switching capabilities. Ablations are conducted at the 350M 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 layer 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 (Section C).

[4] 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.

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 layer, 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

Table 5. **Left:** Additional architecture components added to fully pretrained, parameter-matched Transformer baselines. The additional components do not benefit downstream language modeling performance. **Right:** The performance of chunked mixed-mode training is comparable to sequence mixed-mode training. However, chunk assignment 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

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 layers. 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 massive 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 125M and 350M models trained with and without chunk switching, and Section F 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 may incur 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 E and leave overhead reduction for future work.

C.2. Specific ORYX Variant Architectures

Mamba-2 ORYX. The RMSNorm used in the Mamba-2 variant of the ORYX layer 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 layer'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 layers, 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

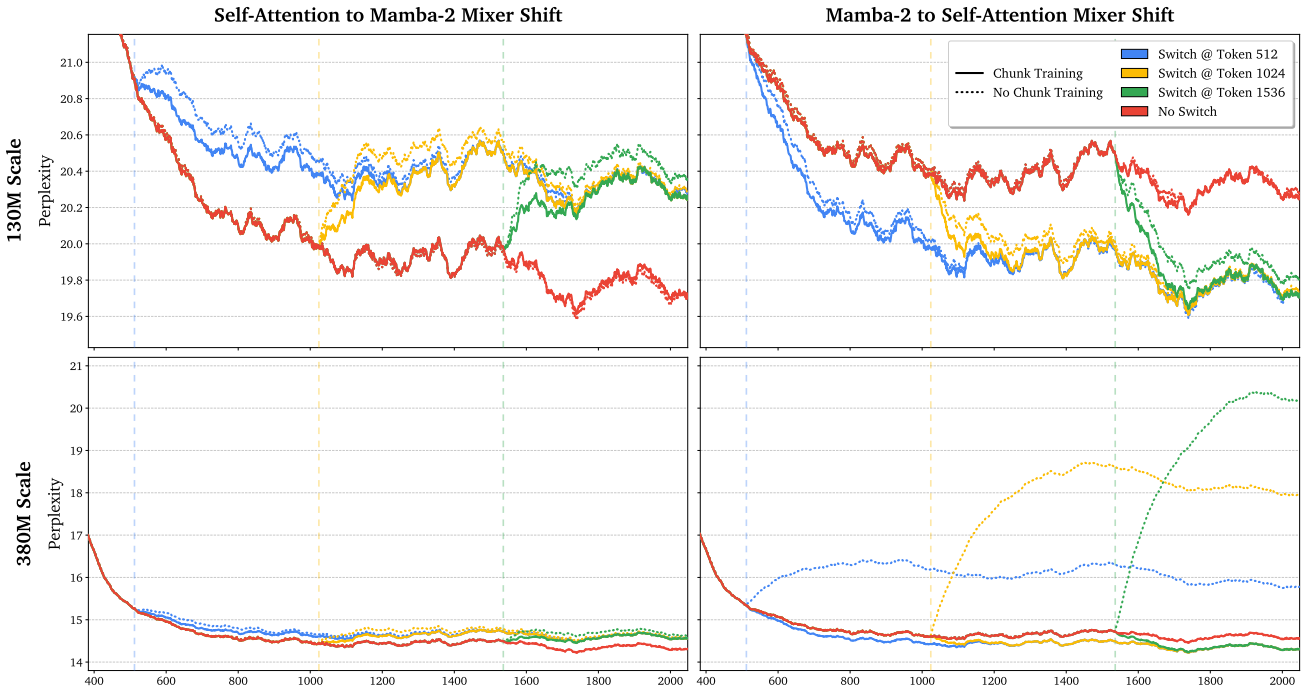

Figure 4. Smoothed perplexity across token index position for ORYX-TM models trained with and without chunk-level switching. Models trained without mode chunk switching perform slightly worse after switching mixers and can suffer from strong degradation in some cases.

positional embedding globally to queries and to the shared key empirically helps both mixer's performances (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 indirect information via the decay) (Shi & Wu, 2024).

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

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 application 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 projected values are still shared.

C.3. Pseudocode

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.

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

Listing 1. Abstracted pseudocode for the Mamba-2 variant of ORYX shared layer. 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 OryxLayer(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
```

D. Experimental Setup

ORYX Setup. Our model follows the Transformer++ setup originating from Touvron et al. (2023) and detailed in Gu & Dao (2024). We use ORYX-TM to denote a self-attention (Transformer)/Mamba-2 shared-layer model, and ORYX-TG to denote a self-attention/GDN shared-layer 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 mixer parameters, the resulting mixer-to-MLP parameter ratio increases from the baselines Transformer's 4:8 to 5:8. Since the MLPs and most projections are shared across mixers, more than 90% of the weights are jointly used across modes.[5]

Baseline Setup. Transformer baselines follow the Transformer++ architecture and head structure of Brown et al. (2020). Mamba-2 baselines interleave Mamba-2 layers, 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 layers, using $D_k = 128, D_v = 256$, with MLPs at the same ratio. To parameter-match models, we increase the MLP widths of the baselines to compensate for the additional gate parameters in ORYX layers (the short convolution leads to a negligible increase).

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 was used with 10% of total steps allocated to warmup, 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) at each scale following Dao & Gu (2024); a factor of $10\times$ was used for ORYX models due to its two mixer modes. Models were trained with bfloat16 mixed precision, and a global batch size of 1M tokens was used 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.

E. 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 a linear recurrent mechanism, Mamba-2 to be specific. 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.
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.

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

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 at uniform 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-layer 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 layer would utilize less FLOPs than that of softmax attention if the context length were 2048 ($T = 2048 \geq 555$).

F. Additional Language Modeling and 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 125M and 1.4B scale. However, the 350M and 760M 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).

Table 9. Full 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 ↑	Average acc ↑
130M	Baseline	Transformer	42.6	32.3	39.2	66.8	58.4	28.9	51.1	19.4	42.3
		Mamba-2	41.5	29.9	40.0	67.1	60.0	27.9	52.6	22.8	42.9
		Gated DeltaNet	**36.5**	32.2	**40.5**	**68.4**	**62.7**	28.7	51.6	22.0	**43.7**
	ORYX-TM	Transformer	38.2	**34.3**	39.3	67.7	58.7	28.1	**54.0**	22.4	43.5
		Mamba-2	40.3	31.1	39.8	67.4	59.0	**29.0**	53.7	**23.2**	43.3
	ORYX-TG	Transformer	39.3	32.7	39.9	67.3	59.7	28.6	50.9	21.4	42.9
		Gated DeltaNet	39.0	31.5	40.4	67.1	59.3	27.9	48.6	20.8	42.2
380M	Baseline	Transformer	19.5	41.1	51.0	70.8	68.2	33.6	55.7	23.8	49.2
		Mamba-2	18.3	41.3	**51.8**	**72.1**	68.5	35.2	56.6	27.0	50.4
		Gated DeltaNet	**16.5**	**42.4**	51.4	71.2	68.6	34.6	55.3	27.2	50.1
	ORYX-TM	Transformer	17.8	41.9	51.2	71.2	**71.0**	35.1	57.0	26.8	50.6
		Mamba-2	18.5	42.3	51.0	71.3	70.5	**36.3**	57.1	**28.0**	**50.9**
	ORYX-TG	Transformer	19.4	40.5	51.3	71.6	67.1	33.8	55.9	24.4	49.2
		Gated DeltaNet	18.1	40.4	51.5	71.3	68.6	34.8	**57.5**	25.6	50.0
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**

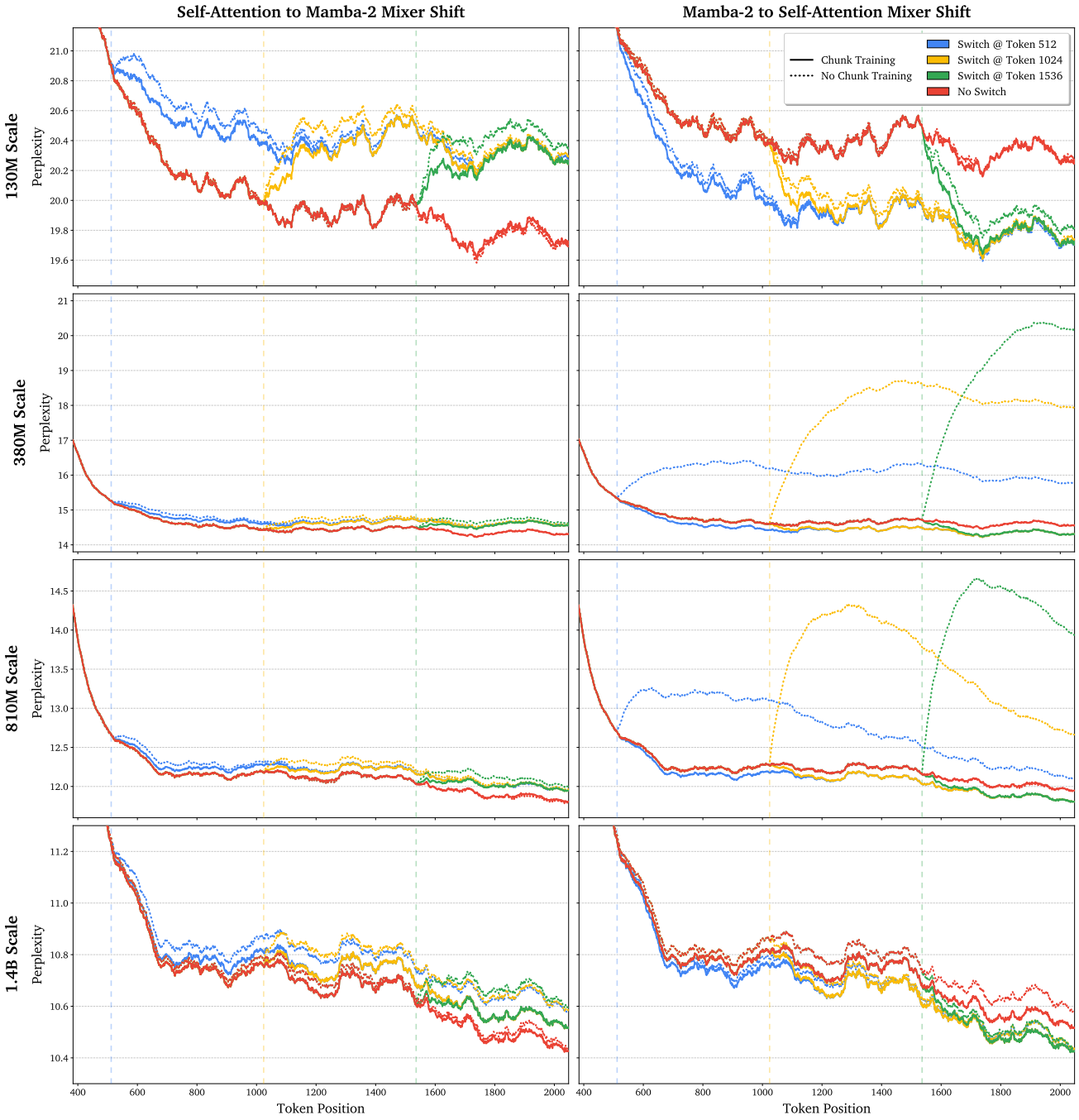

Figure 5. Smoothed perplexity across token index position for ORYX-TM models at all four scales trained with and without chunk-level switching.

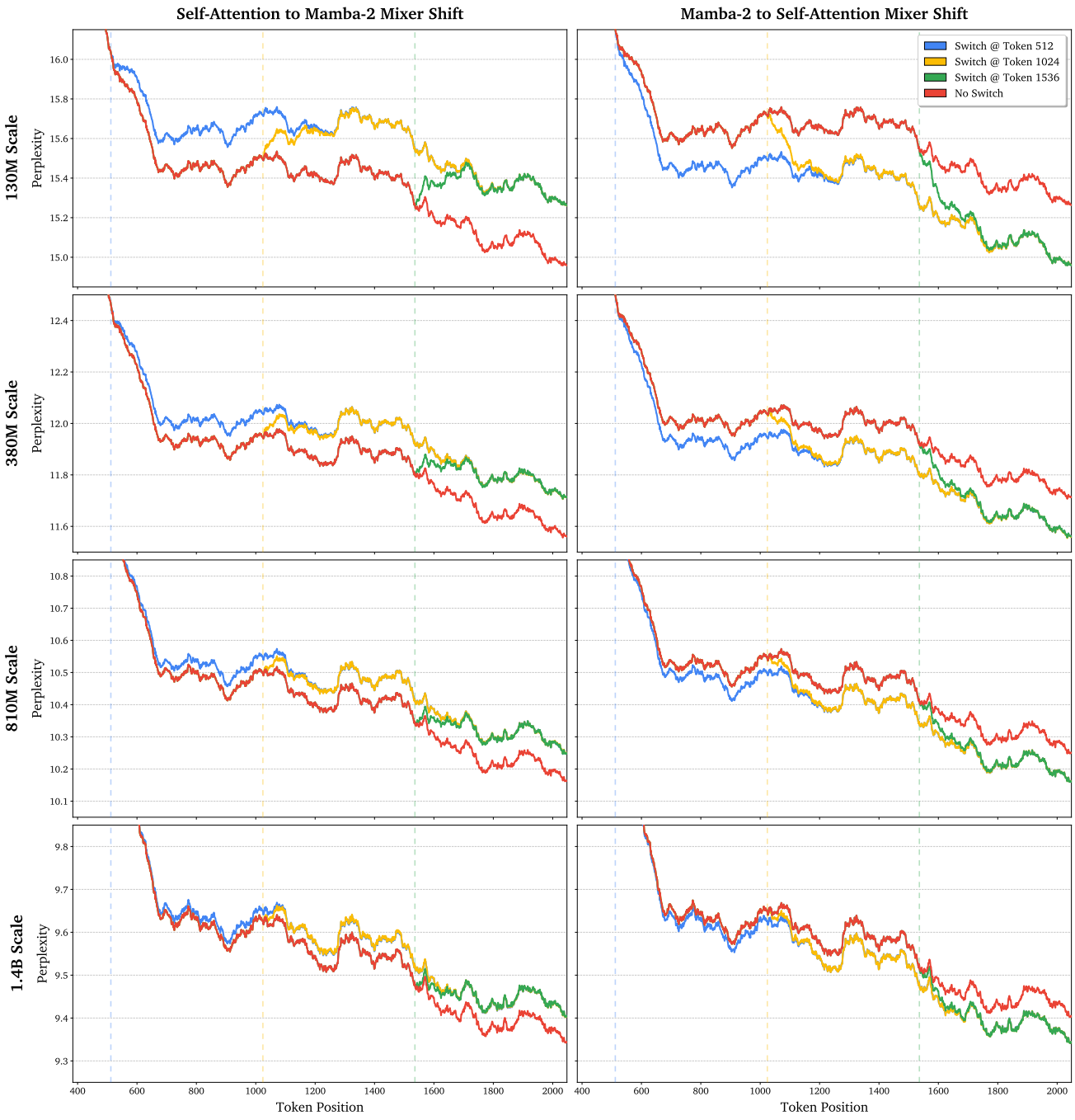

Figure 6. Smoothed perplexity across token index position for all chunked mixed-mode ORYX-TM models across scale.

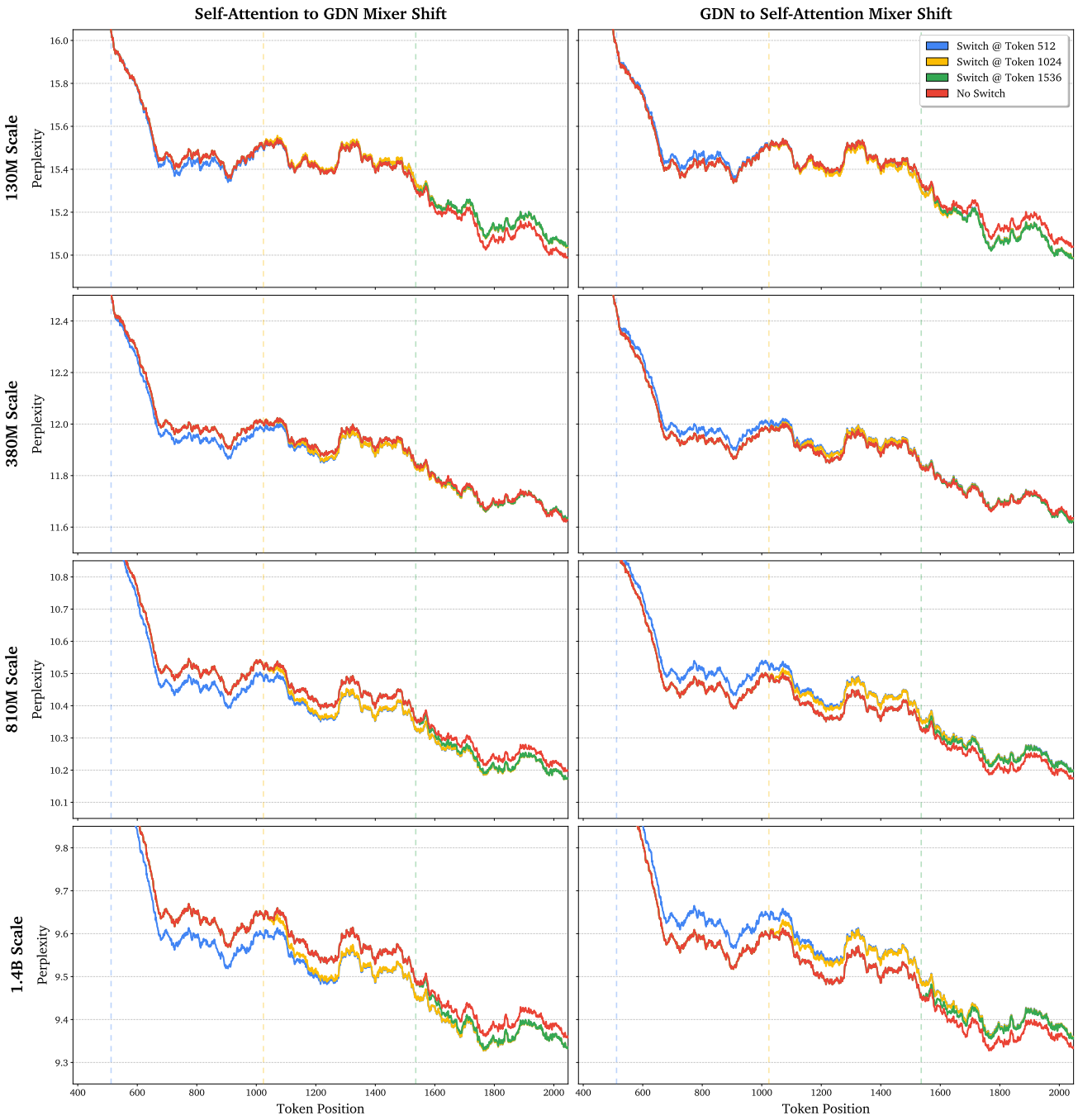

Figure 7. Smoothed perplexity across token index position for all chunked mixed-mode pretrained ORYX-TG models across scale.

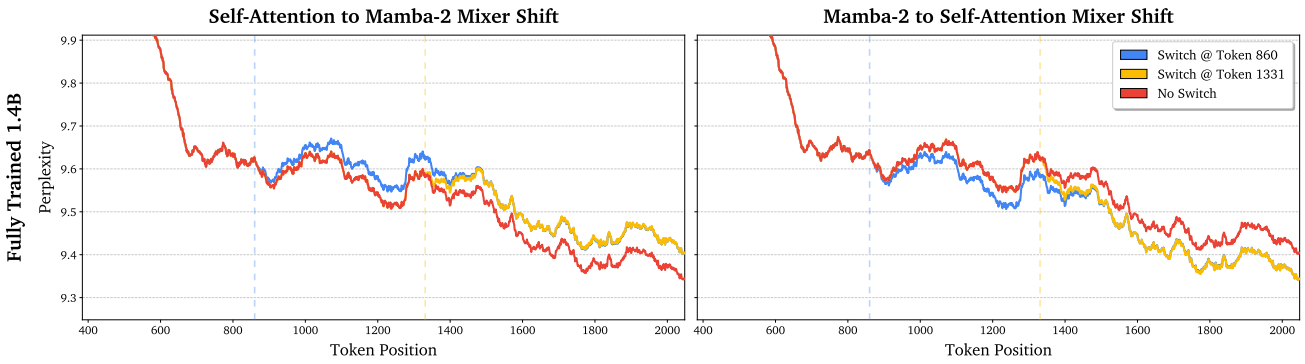

Figure 8. Smoothed perplexity across token index position at non-chunk boundaries for chunked mixed-mode pretrained ORYX-TM 1.4B model.

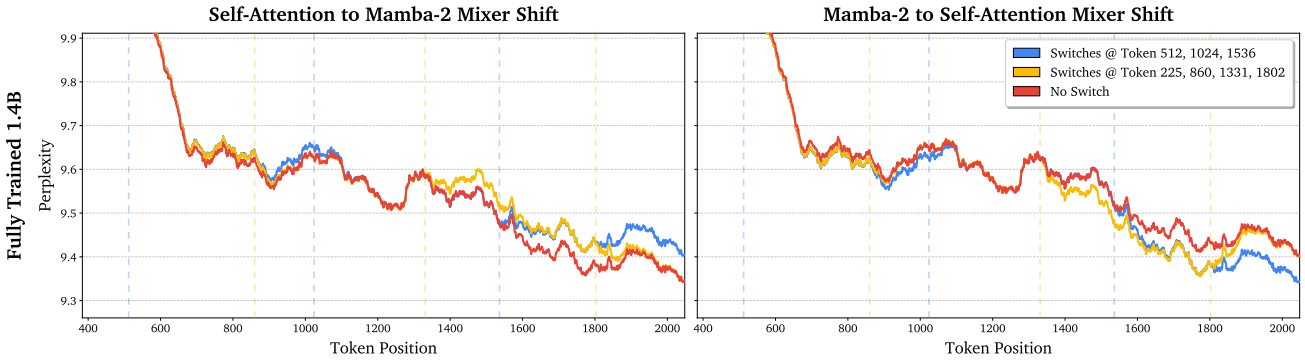

Figure 9. Smoothed perplexity across token index position for multiple switches for chunked mixed-mode pretrained ORYX-TM 1.4B model.

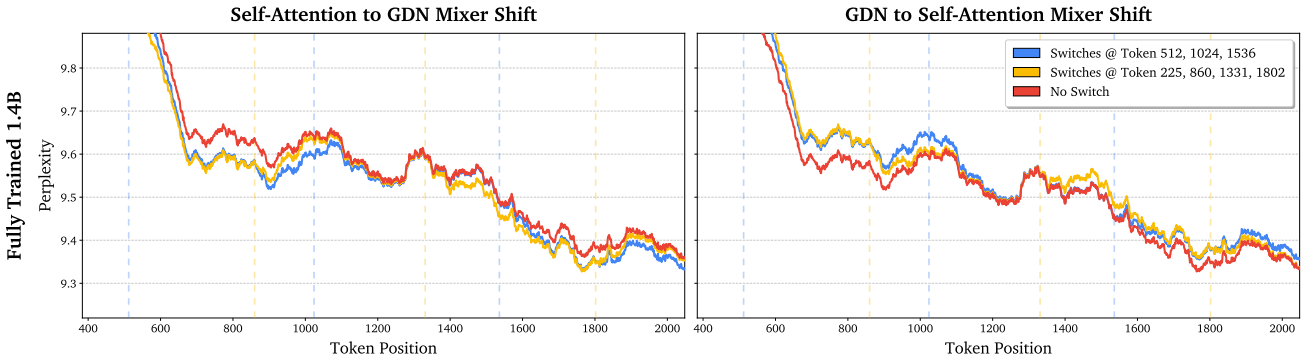

Figure 10. Smoothed perplexity across token index position for multiple switches for chunked mixed-mode pretrained ORYX-TG 1.4B model.