Orthros: Phase-Aware Heterogeneous Attention for Efficient Transformers

Yuhong Chou[1*], **Zehao Liu**[1], **Yuqi Pan**[2], **Qian Liu**[3], **Xianwei Chen**[1], **Jibin Wu**[1†]

[1]The Hong Kong Polytechnic University
[2]University of Chinese Academy of Sciences
[3]Independent Researcher

Abstract

Efficient attention mechanisms, such as linear and sliding window attention, aim to reduce the computational complexity of Transformers from quadratic to linear. However, their degraded in-context recall often necessitates interleaving full attention layers, leaving the quadratic bottleneck unresolved in long-context processing. To address this, we introduce Orthros attention, which shifts the efficiency paradigm from structural allocation to phase-aware decoupling. As a unified module, it dynamically switches its computation pattern across different phases of sequence processing, enabling linear complexity prefilling and full attention decoding. Our design grants Orthros attention a lower asymptotic complexity compared to vanilla Transformers in prevalent long-context scenarios characterized by heavy prefilling and light decoding. Beyond theoretical formulation and complexity analysis, extensive empirical evaluations demonstrate Orthros's superiority across multiple dimensions. First, we validate its fundamental language modeling capabilities, demonstrating scalability across varying model sizes and strong competitiveness in comprehensive architectural comparisons. Second, Orthros exhibits exceptional conversion feasibility via a data-efficient Transformer-to-Orthros conversion, yielding performance that even surpasses the original Transformer. Finally, Orthros achieves a $21.9\times$ prefilling speedup over Transformers with end-to-end inference gains, while preserving long-context capabilities, as demonstrated by near-perfect 128K needle-in-a-haystack results and Transformer-level performance on real-world in-context recall tasks. Overall, these advantages in high performance, flexibility, and efficiency position Orthros as a promising candidate to succeed the Transformer in industrial applications.

1 Introduction

Efficient attention mechanisms have been extensively investigated to address the growing prevalence of long-context scenarios in modern foundation models [52]. In applications such as high-resolution multimodal [4, 55], repository-level code analysis [9, 7], and long-horizon agentic workflows [48, 61, 62], interaction sequences routinely span hundreds of thousands to millions of tokens. Under such extreme lengths, the quadratic complexity of vanilla full attention [50] becomes a critical bottleneck. While methods like sliding window and linear attention [13, 17, 51, 43] reduce this complexity to a linear scale, their respective mechanisms of key-value (KV) eviction and state compression severely degrade in-context recall [3, 1], especially in extended contexts. To mitigate this issue, hybrid Transformers that spatially mix linear and full attention [30, 31, 15, 49, 6] have emerged as the prevailing industry solution to balance performance and efficiency. However, such hybridization implies that the model's overall asymptotic complexity remains bottlenecked by its most

*yuhong.chou@connect.polyu.hk †Corresponding author. jibin.wu@polyu.edu.hk

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

computationally intensive component: full attention. Consequently, hybrid Transformers ultimately yield only a constant-factor speedup, effectively eclipsing the linear complexity advantage inherent to the efficient attention modules.

We propose Orthros attention, an efficient attention mechanism that maintains a favorable subquadratic complexity property without KV eviction or compression, thereby preserving in-context recall in prevalent long-context scenarios. Its core insight lies in shifting the allocation of computational complexity from the structural dimension to the distinct phases. Specifically, Orthros attention applies sliding window attention across the entire sequence, while selectively performing additional full attention at positions tasked with next-token prediction. By adopting distinct computational paradigms, this mechanism perfectly aligns with modern hardware realities: it maintains highly efficient linear complexity during the compute-intensive prefilling phase, while the heavier decoding computation is effectively masked by inherent memory I/O bottlenecks. Through our algorithmic design, these heterogeneous computations are encapsulated within a unified Orthros attention layer, while simultaneously ensuring strict consistency between training and inference. By functioning as a self-contained module, this formulation does not require broader architectural changes, such as cross-layer attention and decoder-decoder designs, thereby preserving the plug-and-play capability to convert existing well-trained foundation models.

Empirically, we conduct comprehensive experiments to validate Orthros attention across diverse scales and settings. First, using a controlled token budget, we verify the architecture's scalability from 160M to 6.8B parameters and perform detailed component ablations to justify our design choices. Second, we pre-train a 1.3B parameter Orthros model from scratch on 100B tokens, demonstrating that it achieves performance highly comparable to strong Transformer baselines. Third, we validate its plug-and-play capability by converting well-trained base models, specifically Llama3.1-8B and Qwen3-8B. Compared to models continually trained on equivalent data volumes, the converted Orthros models deliver performance comparable to or exceeding the original architecture. Fourth, evaluations on in-context recall tasks show that our pre-trained models match Transformer-level performance; notably, in long-context extension, Orthros achieves near-perfect retrieval accuracy on the 128K needle-in-a-haystack test. Fifth, system-level profiling of prefilling efficiency reveals that Orthros yields up to a $21.9\times$ speedup at an extreme sequence length of one million tokens. Finally, end-to-end inference results confirm that this acceleration translates into tangible overall gains, with the doubled FLOP count during decoding effectively masked by the I/O-bound nature of the phase. Together, our theoretical and empirical results validate that the Orthros Transformer delivers performance and efficiency competitive with or superior to strong Transformer baselines, positioning it as a highly promising successor to the Transformer architecture.

2 Related Works

In this section, we first review efficient attention mechanisms, including linear and sliding window attention. Next, we discuss the decoder-decoder framework, a macro-architecture built upon cross-layer attention.

2.1 Efficient Attention

Given an input sequence of token vectors $\{\boldsymbol{x}_i\}_{i=1}^{T}$, where $\boldsymbol{x}_i \in \mathbb{R}^{1\times d_{\text{model}}}$ we define the query, key, and value vectors at position i as $\boldsymbol{q}_i = \boldsymbol{x}_i\boldsymbol{W}_Q$, $\boldsymbol{k}_i = \boldsymbol{x}_i\boldsymbol{W}_K$, and $\boldsymbol{v}_i = \boldsymbol{x}_i\boldsymbol{W}_V$, where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d_{\text{model}}\times d}$ are projection matrices. The standard causal self-attention [50] output $\boldsymbol{o}_t$ at position t is formulated as:

$$\boldsymbol{o}_t = \frac{\sum_{j=1}^{t} \exp\left(\frac{\boldsymbol{q}_t\boldsymbol{k}_j^\top}{\sqrt{d}}\right)\boldsymbol{v}_j}{\sum_{m=1}^{t} \exp\left(\frac{\boldsymbol{q}_t\boldsymbol{k}_m^\top}{\sqrt{d}}\right)}. \tag{1}$$

Since computing the output at each position t requires t inner products, evaluating the entire sequence from $t = 1$ to T incurs a quadratic computational complexity of $\mathcal{O}(T^2)$. Consequently, the community has actively pursued efficient attention implementations to mitigate this bottleneck.

By decomposing the exponential similarity via a kernel feature map $\phi(\cdot)$, standard linear attention [27] computes the output recursively using a hidden state matrix $\boldsymbol{S}_t$ and a normalization vector $\boldsymbol{z}_t$:

$$\boldsymbol{o}_t = \frac{\phi(\boldsymbol{q}_t)\boldsymbol{S}_t}{\phi(\boldsymbol{q}_t)\boldsymbol{z}_t}, \quad \text{where} \quad \boldsymbol{S}_t = \sum_{j=1}^{t} \phi(\boldsymbol{k}_j)^\top \boldsymbol{v}_j, \quad \boldsymbol{z}_t = \sum_{j=1}^{t} \phi(\boldsymbol{k}_j)^\top. \tag{2}$$

Following this paradigm, numerous linear attention variants have emerged [21, 37, 46, 38, 59, 10, 14, 11, 58]. Despite introducing diverse mechanisms for hidden state updates and output computations, these models universally preserve the desirable $\mathcal{O}(T)$ asymptotic complexity.

Alternatively, directly restricting the receptive field of self-attention to a local window of size w yields sliding window attention [24, 36, 44, 40]:

$$\boldsymbol{o}_t = \frac{\sum_{j=\max(1,t-w+1)}^{t} \exp\left(\frac{\boldsymbol{q}_t \boldsymbol{k}_j^\top}{\sqrt{d}}\right) \boldsymbol{v}_j}{\sum_{m=\max(1,t-w+1)}^{t} \exp\left(\frac{\boldsymbol{q}_t \boldsymbol{k}_m^\top}{\sqrt{d}}\right)}. \tag{3}$$

For both linear and sliding window attention, the computational overhead at each time step t is bounded by a constant state size or a fixed window size w. Therefore, processing the full sequence achieves an overall linear complexity of $\mathcal{O}(T)$.

2.2 Cross-Layer Attention and Decoder-Decoder Architectures

In contrast to self-attention, cross-attention [50, 8, 53] derives its queries and key-value (KV) pairs from entirely distinct token sequences. Formally, given a query token $\boldsymbol{x}_i$ and a context token $\boldsymbol{y}_j$, the projections become $\boldsymbol{q}_i = \boldsymbol{x}_i \boldsymbol{W}_Q$, $\boldsymbol{k}_j = \boldsymbol{y}_j \boldsymbol{W}_K$, and $\boldsymbol{v}_j = \boldsymbol{y}_j \boldsymbol{W}_V$, with the output computed via Equation 1. In modern LLMs, this mechanism is frequently used to share KV pairs across different layers, known as cross-layer attention.

Building upon cross-layer attention, the decoder-decoder framework [47, 32, 41, 54] is a specific hybrid macro-architecture that partitions the network into two stages. The first half, termed the *self-decoder*, applies efficient attention and generates a single, shared KV cache. The second half, the *cross-decoder*, relies entirely on full cross-layer attention to attend to this shared cache. In this way, the cross-decoder computation can be entirely skipped before token generation, while the global receptive field is maintained during the decoding phase.

While this architecture achieves a notable balance between efficiency and performance, it inherently possesses two critical limitations. First, relying on a single shared KV cache for half of the network constitutes an extreme sharing strategy, which existing studies on cross-layer attention suggest inevitably compromises performance. Second, this macro-design strictly limits architectural flexibility. Specifically, inserting any state-maintaining attention mechanisms (such as standard self-attention or linear attention) into the cross-decoder would fundamentally break its core efficiency property: the early exit capability before generation.

3 Method

First, Section 3.1 introduces Orthros attention, a prefilling-decoding heterogeneous attention mechanism that processes input and output sequences using distinct computation paradigms. Next, Section 3.2 details the training and inference schemes of the Orthros Transformer from a macro-architectural perspective. Finally, Section 3.3 analyzes the efficiency characteristics of the proposed design across various scenarios.

3.1 Orthros attention

Orthros attention is fundamentally designed around the inherent structure of autoregressive generation, where a sequence comprises a given prompt and subsequent generated tokens. Exploiting this structure, Orthros attention decouples context assimilation from output generation. Specifically, it applies sliding window attention across all tokens to efficiently assimilate local context and materialize

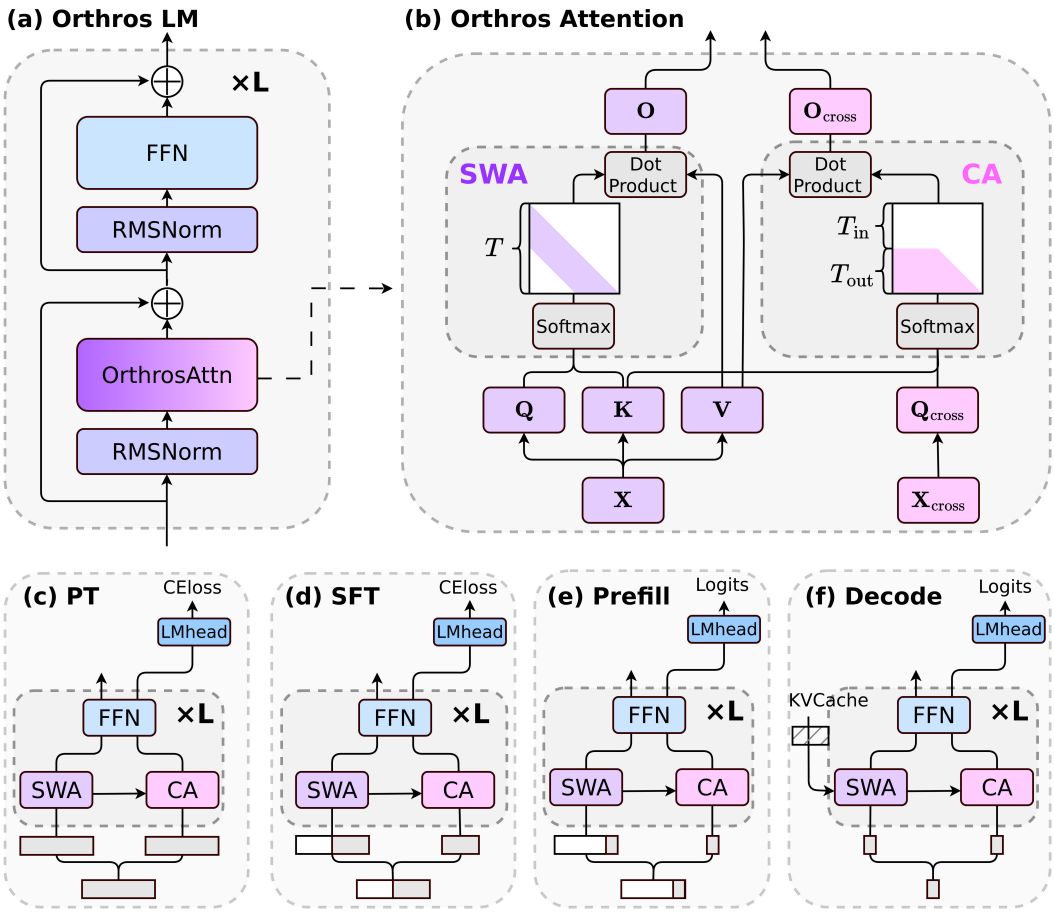


Figure 1: Overview of Orthros attention. The top two sub-figures describe the Orthros architecture and the bottom four sub-figures show the comparison of four sequence-processing regimes. SWA stands for sliding window attention, and CA stands for full cross-attention.

a layer-wise KV cache. For positions tasked with output generation, it additionally performs full cross-attention against this materialized KV cache to retrieve global information.

This mechanism naturally exhibits prefilling-decoding heterogeneity across inference stages: during the prefilling phase, the model predominantly executes the highly efficient sliding window attention, invoking full cross-attention exclusively at the final prompt token to initiate generation; conversely, during the decoding phase, each generation step processes a decoupled pair of tokens: one executes sliding window attention to update the KV cache, while the other performs full cross-attention to predict the subsequent output.

Formally, let T denote the total length of a tokenized sequence (excluding the <eos> token), which serves as the complete model input for next-token prediction. We partition this sequence into two functional segments. Let T_{out} denote the number of positions tasked with predicting the generated output tokens (spanning from the final prompt token to the last generated token predicting <eos>). Consequently, we define $T_{\text{in}} = T - T_{\text{out}}$ as the length of the preceding sequence. Given a sequence of base token vectors $\{\boldsymbol{x}_i\}_{i=1}^{T}$ and an additional set of output-specific token vectors $\{\boldsymbol{x}'_i\}_{i=T_{\text{in}}+1}^{T}$ maintained exclusively for the T_{out} output positions, we first compute the key and value vectors for all $i \in [1, T]$ via standard linear projections: $\boldsymbol{k}_i = \boldsymbol{x}_i \boldsymbol{W}_K$ and $\boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{W}_V$. For the queries, we apply distinct affine transformations to differentiate their functional roles. Specifically, the query for the base token is computed for all $i \in [1, T]$ as $\boldsymbol{q}_i = \boldsymbol{\alpha} \odot (\boldsymbol{x}_i \boldsymbol{W}_Q) + \boldsymbol{\beta}$. In contrast, the query for the output-specific token is derived exclusively for the output sequence $i \in [T_{\text{in}} + 1, T]$ as $\boldsymbol{q}'_i = \boldsymbol{\alpha}' \odot (\boldsymbol{x}'_i \boldsymbol{W}_Q) + \boldsymbol{\beta}'$. Here, $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{d_{\text{model}} \times d}$ denote the standard projection matrices,

while $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\alpha}', \boldsymbol{\beta}' \in \mathbb{R}^{1\times d}$ represent learnable transformation vectors, with $\odot$ denoting element-wise multiplication.

Orthros attention computes outputs via two distinct streams. The sliding window attention output $\boldsymbol{o}_t$ (with window size w) is computed for all $t \in [1, T]$, while the full cross-attention output $\boldsymbol{o}'_t$ is computed exclusively for the output sequence $t \in [T_{\text{in}} + 1, T]$:

$$\boldsymbol{o}_t = \frac{\sum_{j=\max(1,t-w+1)}^{t} \exp\left(\frac{\boldsymbol{q}_t \boldsymbol{k}_j^\top}{\sqrt{d}}\right) \boldsymbol{v}_j}{\sum_{m=\max(1,t-w+1)}^{t} \exp\left(\frac{\boldsymbol{q}_t \boldsymbol{k}_m^\top}{\sqrt{d}}\right)} \quad (4) \qquad \boldsymbol{o}'_t = \frac{\sum_{j=1}^{t} \exp\left(\frac{\boldsymbol{q}'_t \boldsymbol{k}_j^\top}{\sqrt{d}}\right) \boldsymbol{v}_j}{\sum_{m=1}^{t} \exp\left(\frac{\boldsymbol{q}'_t \boldsymbol{k}_m^\top}{\sqrt{d}}\right)} \quad (5)$$

Algorithm 1 outlines the forward pass of the Orthros attention layer, with blue highlights denoting deviations from standard attention. For computational convenience, the input tensor concatenates the T base tokens and the T_{out} output-specific tokens along the sequence dimension. The first T tokens generate the shared KV cache and the base queries Q for sliding window attention. Conversely, the subsequent T_{out} tokens exclusively generate the global queries Q_{cross} for full cross-attention. The resulting outputs from both streams are concatenated along the sequence dimension before the final linear projection. A visual representation of this architecture is provided in Figure 1(b).

Algorithm 1 OrthrosAttention_Layer

Require: Input tensor $X \in \mathbb{R}^{(T+T_{\text{out}})\times d_{\text{model}}}$, number of heads H, sliding window size W
Parameters: $W_Q, W_K, W_V, W_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$
$X, X_{\text{cross}} \leftarrow X_{:T,:}, X_{T:,:}$ ▷ Split base and output-specific tokens
$K, V \leftarrow XW_K, XW_V$ ▷ Generate shared KV cache
$Q, Q_{\text{cross}} \leftarrow \boldsymbol{\alpha} \odot (XW_Q) + \boldsymbol{\beta}, \boldsymbol{\alpha}' \odot (X_{\text{cross}} W_Q) + \boldsymbol{\beta}'$ ▷ Distinct query transformations
Split heads:
  $Q, K, V \in \mathbb{R}^{T\times H\times d_h}$, $Q_{\text{cross}} \in \mathbb{R}^{T_{\text{out}}\times H\times d_h}$ $(d_h = d_{\text{model}}/H)$
$O \leftarrow \text{Flash-Attn}(Q, K, V, \text{window_size} = W)$
$O_{\text{cross}} \leftarrow \text{Flash-Attn}(Q_{\text{cross}}, K, V, \text{window_size} = \infty)$
Merge heads: $O \in \mathbb{R}^{T\times d_{\text{model}}}$, $O_{\text{cross}} \in \mathbb{R}^{T_{\text{out}}\times d_{\text{model}}}$
$O \leftarrow \text{Concat}[O, O_{\text{cross}}] \in \mathbb{R}^{(T+T_{\text{out}})\times d_{\text{model}}}$
$Y \leftarrow OW_O$
return Y

At the macro-architectural level, integrating these output-specific tokens requires minimal modifications to the standard Transformer backbone. Non-attention modules like Feed-Forward Networks (FFNs) and Normalization operate strictly token-wise. Consequently, they naturally process the additional tokens alongside the base tokens without requiring any structural changes. As detailed in Algorithm 2, the Orthros backbone simply duplicates the embeddings of the T_{out} output positions and appends them to the sequence. After uniformly processing this concatenated sequence through L layers, the model extracts exclusively these T_{out} output-specific tokens to compute the final logits.

Algorithm 2 Orthros Transformer Backbone

Require: Input tensor $X \in \mathbb{R}^{T\times d_{\text{model}}}$, total sequence length T, output sequence length T_{out}, number of layers L
$Z \leftarrow \text{Concat}[X, X_{T-T_{\text{out}}:,:}] \in \mathbb{R}^{(T+T_{\text{out}})\times d_{\text{model}}}$ ▷ Duplicate the T_{out} output-specific tokens
for $\ell = 1$ to L **do**
  $A \leftarrow \text{OrthrosAttention_Layer}_\ell(\text{Attn_Norm}_\ell(Z))$
  $Z \leftarrow Z + A$
  $F \leftarrow \text{FFN_Layer}_\ell(\text{FFN_Norm}_\ell(Z))$
  $Z \leftarrow Z + F$
end for
$Z \leftarrow Z_{T:,:}$ ▷ Extract the T_{out} output-specific tokens
$Y \leftarrow \text{Norm}(Z)$
return Y

3.2 Training & Inference Schemes

The architectural elegance of the Orthros Transformer lies in its unified formulation, which ensures strict algorithmic consistency between training and inference. We categorize its operational pipelines into four distinct paradigms: Pre-training (PT), Supervised Fine-Tuning (SFT), Prefilling, and Decoding. Remarkably, all four scenarios are seamlessly instantiated without any structural modifications, simply by adjusting the context length T_{in} and the output length T_{out}. Figure 1(c)-(f) provides a visual representation of these configurations.

Pre-Training ($T_{\text{in}} = 0, T_{\text{out}} = T$): In standard causal language modeling, every token serves as a target for next-token prediction. Consequently, the entire sequence is duplicated ($T_{\text{out}} = T$) to instantiate the output-specific tokens, ensuring that full cross-attention is computed at every position to derive the training loss.

Supervised Fine-Tuning ($T_{\text{in}} = T - T_{\text{out}}, T_{\text{out}} \geq 1$): During SFT, the loss is computed exclusively over the generated responses, while the user prompts serve purely as context. Thus, T_{out} corresponds exactly to the total length of the target responses. Notably, alignment tuning paradigms, such as reinforcement learning on post-rollout data, share this identical formulation, as optimization is similarly restricted to the generated tokens.

Prefilling ($T_{\text{in}} = T - 1, T_{\text{out}} = 1$): When processing a given prompt during inference, all tokens except the last act strictly as contextual inputs ($T_{\text{in}} = T - 1$). The model only duplicates the final prompt token ($T_{\text{out}} = 1$) to perform full cross-attention and initiate the generation of the first output token.

Decoding ($T_{\text{in}} = 0, T_{\text{out}} = 1$): In the autoregressive decoding phase, each step processes exactly one generated token ($T = 1$). This single token is duplicated to form the decoupled pair introduced in Section 3.1: the base token updates the KV cache and computes sliding window attention, while the output-specific token performs full cross-attention against the historical cache to predict the subsequent token.

3.3 Complexity Analysis

We analyze the computational complexity of the Orthros Transformer with respect to the total sequence length $T = T_{\text{in}} + T_{\text{out}}$, model width d_{model}, and sliding window size W.

The model's computation comprises three primary components: sliding window attention ($\mathcal{O}(TW d_{\text{model}})$), full cross-attention ($\mathcal{O}(T_{\text{out}} T d_{\text{model}})$), and token-wise linear projections ($\mathcal{O}((T + T_{\text{out}}) d_{\text{model}}^2)$). In practice, since the window size W is typically small ($W \ll d_{\text{model}}$), the sliding window attention term is asymptotically dominated by the linear projections and can be safely omitted from the bottleneck analysis.

In prevalent long-context scenarios, the output length T_{out} is typically a lower-order term relative to T ($T_{\text{out}} \ll T$), or even bounded by a constant (e.g., structured tool invocations in agentic workflows). When $T_{\text{out}} \ll T$, the overall complexity is dominated by the cross-attention term $\mathcal{O}(T_{\text{out}} T d_{\text{model}})$, achieving a highly favorable subquadratic scaling. If $T_{\text{out}} \approx \mathcal{O}(1)$, the attention overhead becomes negligible, reducing the model to a strictly linear complexity $\mathcal{O}(T d_{\text{model}}^2)$ governed by the projections. Only in the extreme edge case where $T_{\text{out}} \approx \mathcal{O}(T)$ (e.g., pre-training) does the complexity degrade to the $\mathcal{O}(T^2 d_{\text{model}})$ scaling of a vanilla Transformer.

Finally, during the memory I/O-bound decoding phase, Orthros processes a decoupled pair of tokens per generation step. Fortunately, this additional computation incurs negligible latency overhead. Since the I/O complexity is strictly dominated by loading the weight matrices, the arithmetic cost of processing two tokens, or even a small batch of vectors, is entirely masked by the memory access latency. Furthermore, in batched decoding scenarios, this strategy of increasing active compute tokens enables the system to more rapidly transcend the I/O bottleneck, effectively escaping the regime of inefficient GPU utilization.

4 Experiments

We conduct comprehensive experiments to evaluate Orthros from five perspectives: First, we do scaling and ablation studies from 160M to 6.8B parameters to validate the scalability and the key

component design. Second, pre-training from scratch shows that Orthros achieves competitive or superior performance compared with Transformer and efficient attention architectures on benchmarks. Third, conversion experiments on Llama3.1 and Qwen3 base models demonstrate that existing full attention Transformers can be converted into Orthros with data-efficient continual training. Fourth, in-context recall evaluations show that Orthros achieves competitive performance among all architectures and near-perfect accuracy on the 128K needle-in-a-haystack test. Fifth, prefilling efficiency tests in a single GPU and multi GPUs show that Orthros substantially reduces long-context prefilling latency, achieving up to a $21.9\times$ speedup. Finally, end-to-end inference evaluations reveal that this prefilling advantage translates into tangible overall speedups. Moreover, the doubled FLOP count during decoding is effectively masked by the I/O-bound nature of this phase, strongly corroborating our theoretical predictions.

In all the experiments, Orthros uses sliding window attention of 128 window size. We report all the experiment settings in Appendix A.

4.1 Scaling and Ablation Study

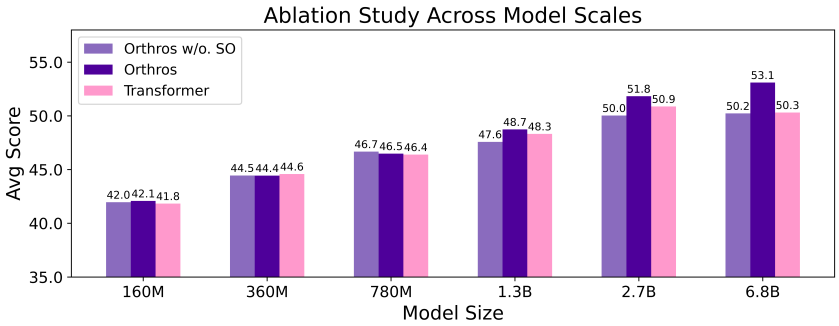


Figure 2: The ablation and scaling study of Orthros. The transformer serves as a baseline. Orthros w/o SO removes different scalers $\boldsymbol{\alpha}$ and offsets $\boldsymbol{\beta}$.

In this section. We conduct scaling and architecture component ablation studies. A key design choice in Orthros is the scale-offset (SO) [22], which specializes the shared query projection for different attention streams through lightweight element-wise transformations. This design preserves parameter sharing and I/O efficiency.

In the experiment, all ablation models are trained on 10B tokens with a sequence length of 2048 and a 0.25M tokens batch size. Orthros denotes our model and its w/o SO variant removes the learnable scale-offset transformation. We evaluate three models of different model sizes from 160M to 6.8B with `lm-evaluation-harness` framework [18] on Common Sense Reasoning benchmarks, which include ARC-challenge (ARC-C) [12], ARC-easy (ARC-E) [12], HellaSwag (HS) [60], OpenBookQA (OBQA) [35], PIQA [5], Social-IQa (SIQA) [42], WinoGrande (Wino) [29]. We report the average score in Figure 2, and full results are reported in Appendix B. These results show that Orthros exhibits competitive model scalability against Transformer baselines, with its scale-offset design providing a better trade-off between parameter efficiency and performance

4.2 Training from Scratch

Next, we pre-train Orthros from scratch and compare it with Transformer and efficient attention architectures, including GatedDeltaNet (GDN), Gated Linear Attention (GLA), RetNet, and sliding window attention (SWA). For the 1.3B-scale comparison, all models are trained on 100B tokens fineweb-edu [33] dataset with a sequence length of 2048 and a 2M tokens batch size. The SWA baseline uses a window size of 512, while Orthros uses a window size of 128, following our default setting.

We evaluate the models on Common Sense Reasoning benchmarks (Section 4.1), as shown in Table 1. Orthros achieves the best average performance among all compared architectures while maintaining comparable performance across individual tasks. This result indicates that Orthros preserves strong general language modeling ability.

Table 1: Common Sense Reasoning evaluation of different architectures at 1.3B scale. Higher is better for all metrics.

Model	ARC-C Norm	ARC-E	HS Norm	OBQA Norm	PIQA	SIQA	Wino	Avg.
Transformer	37.80	70.58	56.21	37.80	**72.91**	41.10	**57.46**	53.41
Orthros	**39.08**	**71.09**	**57.19**	**40.20**	71.82	**42.12**	56.35	**53.98**
GDN	38.82	70.66	56.28	**40.20**	71.44	40.63	55.01	53.29
GLA	36.86	67.97	53.29	38.20	70.73	40.07	56.20	51.90
RetNet	35.84	69.82	54.33	37.00	71.93	39.76	55.01	51.96
SWA	36.95	70.16	56.11	38.80	72.42	40.94	57.14	53.22

4.3 Conversion Training

Table 2: Evaluation results of converting Llama3.1 and Qwen3 to Orthros. Higher is better for all metrics.

Base	Model	ARC-C Norm	ARC-E	HS Norm	OBQA Norm	PIQA	SIQA	Wino	MMLU	Avg.
Llama3.1	Base	53.50	81.57	78.99	45.20	79.76	46.67	73.09	65.00	65.47
	Orthros CPT	57.08	**84.13**	**79.10**	**45.60**	**80.25**	**48.93**	**74.35**	68.65	**67.26**
	Base CPT	**57.51**	83.80	78.93	45.40	**80.25**	47.80	73.40	**68.89**	67.00
Qwen3	Base	57.34	81.73	**78.49**	42.20	**79.27**	**54.61**	**72.53**	76.82	67.87
	Orthros CPT	**59.39**	**84.81**	78.06	**44.80**	**79.27**	50.51	72.38	77.76	**68.37**
	Base CPT	57.59	83.29	77.59	43.20	78.78	50.00	71.98	**77.96**	67.55

We next study whether existing pretrained full attention models can be converted into Orthros with data-efficient continual pretraining. Specifically, we convert Llama3.1-8B [20] and Qwen3-8B [56] base models to Orthros by replacing their original attention modules with Orthros attention, and continue pre-training both the converted and base models on an 80B-token corpus of high-quality Common Crawl (CC) data [45] with a restricted mixture of math [34] and code [23]. To isolate the effect of architecture conversion from the effect of additional training data, we include two baselines for each model family: The first is the original base models (Base), which measure the performance before continual training. The second is Base CPT, which continues training the original full attention architecture on the same 80B-token budget.

We evaluate the models on Massive Multitask Language Understanding (MMLU) and Common Sense Reasoning benchmarks (Section 4.1). The results are shown in Table 2. Across both Llama3.1-8B and Qwen3-8B, Orthros CPT consistently outperforms both the Base and the same-architecture Base CPT baseline on average, indicating gains of Orthros conversion come from the architecture itself rather than merely from additional training. Additionally, these results demonstrate that Orthros can be obtained not only by pre-training from scratch, but also by converting strong pretrained full attention Transformer with a moderate continual training cost.

4.4 In-Context Recall Capacity

To evaluate whether Orthros avoids the inherent disadvantage of efficient architectures on recall-intensive tasks, we first compare the 1.3B Orthros models (trained from scratch in Section 4.2) on a suite of in-context recall-intensive benchmarks, which include DROP [16], NarrativeQA (NQ) [28], SWDE (SW) [2], TriviaQA (TQ) [25], FDA (FD) [2], SQuAD (SQ) [39]. Due to space limitations, full results are reported in Table 7 in Appendix B. The results show that Orthros achieves the best average performance among all architectures (34.4, versus 33.9 for the Transformer and at most 28.8 for the efficient baselines) and is comparable with the full attention Transformer on each task. This suggests that Orthros substantially avoids the inherent recall gap of efficient models, while retaining the efficiency benefits.

We further evaluate the long-range retrieval ability of Orthros on the needle-in-a-haystack benchmark [26]. For this experiment, we use the 8B Orthros model converted from Qwen3 (Section 4.3) and further perform 20B-token long-context extension training on ProLong data [19]. As shown in

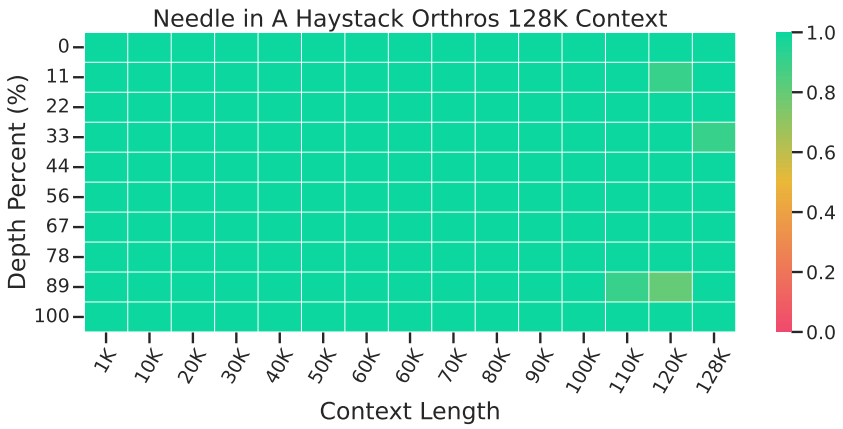


Figure 3: Orthros 128K Needle in a Haystack task result. Orthros achieves near-perfect accuracy.

Figure 3, Orthros maintains near-perfect retrieval accuracy at a context length of 128K. This result confirms that Orthros preserves strong in-context retrieval ability after long-context extension.

4.5 Prefilling Efficiency

We benchmark the prefilling efficiency of 8B-Orthros and compare it with Qwen3 [56]. Specifically, our prefilling benchmark is implemented using FLA-style [57] models with high-performance kernels. As shown in Figure 4, Orthros consistently achieves lower prefilling latency on a single GPU for input lengths from 8K to 128K and 8 GPUs for input lengths from 64K to 1M. The advantage becomes more pronounced as the sequence length increases, which is consistent with the linear-complexity prefilling design of Orthros.

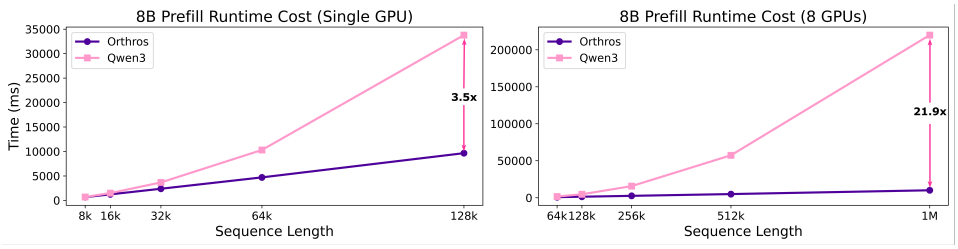


Figure 4: The left figure shows the time cost of Orthros and Qwen3 with a single GPU for prefilling, while the right figure presents 8-GPU prefilling time.

4.6 End-to-End Inference

We conduct end-to-end inference experiments on a single H100 GPU using the same SGLang inference engine. The Transformer and Linear Hybrid baselines follow the 8B architectures of Qwen3 and Qwen3.5, respectively. Orthros is built on the Qwen3-8B architecture using Orthros attention. Orthros-H extends Orthros with a layer-wise mixture of full-attention and linear-attention layers at a ratio of 1:3. In Orthros-H, the GDN layer, GDN as cross-attention part in which the query bypasses the convolution module. All experiments use a batch size of 2, with input and output sequence lengths of 128, 1K, and 8K tokens, utilizing the Triton backend for attention.

Prefilling and decoding. As shown in Table 3, Orthros and Orthros-H substantially reduce prefilling latency relative to Transformer and Linear Hybrid, respectively. Their decode-only throughput remains comparable to that of the corresponding baselines. The two hybrid models decode faster than the two full-attention decoding models.

Table 3: End-to-end inference on a single H100 GPU using SGLang. Time to first token (TTFT) and end-to-end latency (E2E) are measured in seconds; output and decode-only throughput are measured in tokens per second. Output throughput is the total number of generated tokens divided by E2E latency. Lower latency and higher throughput are better.

	128K input / 128 output				127K input / 1K output				120K input / 8K output			
Model	TTFT (s)	E2E (s)	Output (tok/s)	Decode (tok/s)	TTFT (s)	E2E (s)	Output (tok/s)	Decode (tok/s)	TTFT (s)	E2E (s)	Output (tok/s)	Decode (tok/s)
Transformer	77.25	84.91	3.0	33.7	76.16	137.72	14.9	33.2	67.77	548.36	29.9	34.1
Orthros	**17.02**	**24.65**	**10.4**	33.3	**17.30**	**76.35**	**26.8**	33.7	**15.21**	**488.83**	**33.5**	34.6
Linear Hybrid	25.57	28.13	9.1	98.8	25.17	46.31	44.2	96.8	22.75	189.30	86.5	98.3
Orthros-H	**10.18**	**12.77**	**20.0**	97.3	**9.88**	**30.72**	**66.7**	105.8	**9.30**	**173.89**	**94.2**	99.5

End-to-end latency. Faster prefilling translates into lower end-to-end (E2E) latency and higher output throughput, particularly when the output is short. As the output length increases, decoding occupies a larger share of the total time, reducing the relative E2E advantage.

5 Discussion, Limitation and Future Work

Discussion. In this work, we introduced Orthros attention, a novel efficient attention mechanism defined by its inherent prefilling-decoding heterogeneity. Extensive empirical evaluations demonstrate that Orthros delivers substantial gains in inference efficiency without compromising downstream performance or long-context capabilities, standing as a compelling alternative to strong Transformer baselines. Beyond these empirical successes, the introduction of prefilling-decoding heterogeneity invites a broader discussion on the fundamental trade-offs in architecture design and the evolving computational paradigms of modern LLMs:

- **Training–Inference Trade-off.** Orthros achieves substantial inference gains at the cost of doubling token computation during training to maintain strict training–inference consistency. However, conventional scaling analyses primarily assess architectural efficiency through the relationship between training FLOPs and loss, leaving inference efficiency outside the evaluation. This perspective can therefore undervalue architectures that trade additional training computation for sustained inference savings.
- **Prefill-Dominated Workloads and Compute Allocation.** Prefill-heavy applications, such as multimodal understanding and agentic LLM workflows, are particularly well suited to Orthros. Its heterogeneous design allocates less computation to processing inputs and more to generating outputs, reflecting an intuitive strategy for preserving output quality. However, how to fairly compare this redistribution of computation with the uniform computation of standard Transformers remains an open question.

Together, these considerations motivate a broader evaluation framework for heterogeneous architectures that accounts for training cost, phase-specific inference cost, and model quality across diverse workloads. Such a framework should make comparison criteria explicit and assess how architectural benefits vary with input and output lengths and deployment scale, enabling a more comprehensive assessment of computational efficiency.

Limitation and Future Work. While Orthros attention demonstrates significant potential, it currently requires computing a doubled sequence during training. In addition, the memory efficiency of its KV cache remains unaddressed. Beyond these limitations, several future efforts must be undertaken to translate this potential into reality for industrial applications. First, Mixture-of-Experts scenarios require specialized designs to prevent decoupled token pairs from being routed to different experts during decoding, which would otherwise exacerbate the I/O bound. Second, the agentic reinforcement learning paradigm for Orthros needs to be thoroughly investigated; this will help demonstrate its efficiency advantages in multi-turn and long-context agentic applications. Finally, adapting Orthros to modern inference frameworks requires further exploration, particularly regarding how to optimize its decoding mechanism across diverse scenarios, such as single-sample generation and batched decoding.

Acknowledgments and Disclosure of Funding

This work was partially supported by the Research Grants Council of the Hong Kong SAR (Grant No. PolyU25216423, PolyU15217424), and The Hong Kong Polytechnic University (P0056503, P0053699).

We are grateful to Yin Zheng and Zejun Ma for their insightful discussions and support.

References

[1] Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *arXiv:2312.04927*, 2023.

[2] 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. *arXiv preprint arXiv:2304.09433*, 2023.

[3] 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. *arXiv:2402.18668*, 2024.

[4] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[5] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[6] Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, et al. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

[7] Egor Bogomolov, Aleksandra Eliseeva, Timur Galimzyanov, Evgeniy Glukhov, Anton Shapkin, Maria Tigina, Yaroslav Golubev, Alexander Kovrigin, Arie Van Deursen, Maliheh Izadi, et al. Long code arena: a set of benchmarks for long-context code models. *arXiv preprint arXiv:2406.11612*, 2024.

[8] William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention. *Advances in Neural Information Processing Systems*, 37:86927–86957, 2024.

[9] Ruisheng Cao, Mouxiang Chen, Jiawei Chen, Zeyu Cui, Yunlong Feng, Binyuan Hui, Yuheng Jing, Kaixin Li, Mingze Li, Junyang Lin, et al. Qwen3-coder-next technical report. *arXiv preprint arXiv:2603.00729*, 2026.

[10] Yuhong Chou, Man Yao, Kexin Wang, Yuqi Pan, Ruijie Zhu, Yiran Zhong, Yu Qiao, Jibin Wu, Bo Xu, and Guoqi Li. Metala: Unified optimal linear approximation to softmax attention map. *Advances in Neural Information Processing Systems*, 37:71034–71067, 2024.

[11] Yuhong Chou, Zehao Liu, Ruijie Zhu, Xinyi Wan, Tianjian Li, Congying Chu, Qian Liu, Jibin Wu, and Zejun Ma. Zeco: Zero communication overhead sequence parallelism for linear attention. *arXiv preprint arXiv:2507.01004*, 2025.

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

[13] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 2978–2988, 2019.

[14] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

[15] Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Min-Hung Chen, Yoshi Suhara, Yingyan Celine Lin, et al. Hymba: A hybrid-head architecture for small language models. In *The Thirteenth International Conference on Learning Representations*, 2024.

[16] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*, 2019.

[17] Zichuan Fu, Wentao Song, Yejing Wang, Xian Wu, Yefeng Zheng, Yingying Zhang, Derong Xu, Xuetao Wei, Tong Xu, and Xiangyu Zhao. Sliding window attention training for efficient large language models. *arXiv preprint arXiv:2502.18845*, 2025.

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

[19] Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In *ACL*, 2025.

[20] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[21] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[22] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International conference on machine learning*, pages 9099–9117. PMLR, 2022.

[23] Siming Huang, Tianhao Cheng, Jason Klein Liu, Weidi Xu, Jiaran Hao, Liuyihan Song, Yang Xu, Jian Yang, Jiaheng Liu, Chenchen Zhang, et al. Opencoder: The open cookbook for top-tier code large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33167–33193, 2025.

[24] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. *arXiv e-prints*, art. arXiv:2310.06825, October 2023. doi: 10.48550/arXiv.2310.06825.

[25] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, art. arXiv:1705.03551, 2017.

[26] Greg Kamradt. Needle in a haystack - pressure testing LLMs. `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`, 2023.

[27] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

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

[29] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer, 2012.

[30] Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

[31] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

[32] Hongzhan Lin, Zhiqi Bai, Xinmiao Zhang, Sen Yang, Xiang Li, Siran Yang, Yunlong Xu, Jiaheng Liu, Yongchi Zhao, Jiamang Wang, et al. Reconstructing kv caches with cross-layer fusion for enhanced transformers. *arXiv preprint arXiv:2512.03870*, 2025.

[33] 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`.

[34] Rabeeh Karimi Mahabadi, Sanjeev Satheesh, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc-math: A 133 billion-token-scale high quality math pretraining dataset. *arXiv preprint arXiv:2508.15096*, 2025.

[35] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

[36] OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL `https://arxiv.org/abs/2508.10925`.

[37] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *Findings of the association for computational linguistics: EMNLP 2023*, pages 14048–14077, 2023.

[38] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

[39] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://aclanthology.org/D16-1264`.

[40] Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*, 2024.

[41] Liliang Ren, Congcong Chen, Haoran Xu, Young Jin Kim, Adam Atkinson, Zheng Zhan, Jiankai Sun, Baolin Peng, Liyuan Liu, Shuohang Wang, et al. Decoder-hybrid-decoder architecture for efficient reasoning with long generation. *arXiv preprint arXiv:2507.06607*, 2025.

[42] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *EMNLP*, 2019. URL `https://www.aclweb.org/anthology/D19-1454`.

[43] Xuyang Shen, Dong Li, Ruitao Leng, Zhen Qin, Weigao Sun, and Yiran Zhong. Scaling laws for linear complexity language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16377–16426, 2024.

[44] Varun Singh, Lucas Krauss, Sami Jaghouar, Matej Sirovatka, Charles Goddard, Fares Obied, Jack Min Ong, Jannik Straube, Aria Harley, Conner Stewart, et al. Arcee trinity large technical report. *arXiv preprint arXiv:2602.17004*, 2026.

[45] Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2459–2475, 2025.

[46] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[47] Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. *Advances in Neural Information Processing Systems*, 37:7339–7361, 2024.

[48] Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[49] Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[51] Dustin Wang, Rui-Jie Zhu, Steven Abreu, Yong Shan, Taylor Kergan, Yuqi Pan, Yuhong Chou, Zheng Li, Ge Zhang, Wenhao Huang, et al. A systematic analysis of hybrid linear attention. *arXiv preprint arXiv:2507.06457*, 2025.

[52] Xindi Wang, Mahsa Salmani, Parsa Omidi, Xiangyu Ren, Mehdi Rezagholizadeh, and Armaghan Eshaghi. Beyond the limits: A survey of techniques to extend the context length in large language models. *arXiv preprint arXiv:2402.02244*, 2024.

[53] You Wu, Haoyi Wu, and Kewei Tu. A systematic study of cross-layer KV sharing for efficient LLM inference. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 396–403, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-190-2. doi: 10.18653/v1/2025.naacl-short.34. URL https://aclanthology.org/2025.naacl-short.34/.

[54] You Wu, Ziheng Chen, Yizhen Zhang, Haoyi Wu, Chengting Yu, Yuchi Xu, Wenbo Su, Bo Zheng, and Kewei Tu. Yoco++: Enhancing yoco with kv residual connections for efficient llm inference. *arXiv preprint arXiv:2604.13556*, 2026.

[55] Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, et al. Qwen3-omni technical report. *arXiv preprint arXiv:2509.17765*, 2025.

[56] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[57] Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention.

[58] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

[59] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

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

[61] Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

[62] Yufan Zhuang, Xiaodong Yu, Jialian Wu, Ximeng Sun, Ze Wang, Jiang Liu, Yusheng Su, Jingbo Shang, Zicheng Liu, and Emad Barsoum. Self-taught agentic long context understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5525–5537, 2025.

A Experiment Settings

Table 4: Model configurations and training settings for the scaling and ablation study. All models are trained with a sequence length of 2048 for approximately 10B tokens. Orthros uses a sliding window size of 128.

Item	160M	360M	780M	1.3B	2.7B	6.8B
Model						
Architecture	Transformer / Orthros / Orthros w/o SO					
Hidden size	768	1024	1536	2048	2560	4096
Layers	16	24	24	24	32	32
Heads	6	8	12	16	20	32
Head dimension	128					
Vocabulary size	32000					
Torch dtype	bfloat16					
Window size	128 for Orthros and Orthros w/o SO					
Training						
Training tokens	10B					
Sequence length	2048					
Training steps	40K					
Global batch size	0.25M tokens					
Seed	42					
Optimizer	AdamW					
Learning rate	3×10^{-4}					
LR scheduler	Cosine					
Warmup steps	1500					
Minimum LR ratio	0.001					
Weight decay	0.05					
Gradient clipping	1.0					
Loss	Cross Entropy					
FSDP	Full shard					
Data parallel	8					

In this section, we report all the experiment settings. All the training was conducted in H100 GPUs, 80GB memory
In Table 4, we report all the experiment settings of the ablation and scaling experiments in Section 4.1. Section 4.2, 1.3B Models uses the same configuration as shown in Table 4. For efficient models, we use the implementation of FLA [57]. In Table 5, we report all the experiment settings of the ablation and scaling experiments in Section 4.3.

In Section 4.4, we use the 8B-Orthros converted from Qwen3-8B for in-context recall-intensive benchmarks and trained on 20B long context for the needle-in-a-haystack task. For Section 4.5, we use the 8B-Orthros converted from Qwen3-8B, sequence parallelism was enabled in multi GPUs experiment.

B Experiment Metrics

We report all ablation experiment results in this section, as showm in Table 6.

Table 7 reports the full results of the in-context recall-intensive evaluation at 1.3B scale discussed in Section 4.4. Trans. denotes the Transformer; DP denotes DROP, and the remaining abbreviations follow Section 4.4.

Table 5: Model configurations and continual training settings for Transformer-to-Orthros conversion. Base denotes the original pretrained checkpoint without additional training. Orthros Conversion replaces the original full attention modules with Orthros Attention. Base CPT continues training the original Transformer architecture under the same training setting.

Item	**Llama3.1-8B**			**Qwen3-8B**		
	Base	**Orthros Conv.**	**Base CPT**	**Base**	**Orthros Conv.**	**Base CPT**
Model						
Base checkpoint		Llama3.1-8B			Qwen3-8B-Base	
Architecture	Transformer	Orthros	Transformer	Transformer	Orthros	Transformer
Hidden size		4096			4096	
Layers		32			36	
Vocabulary size		128256			151936	
Torch dtype		bfloat16			bfloat16	
window size	–	128	–	–	128	–
continual Training						
Additional training	No	80B tokens		No	80B tokens	
Sequence length	–	32768		–	32768	
Global batch size	–	8M tokens		–	8M tokens	
Training steps	–	10000		–	10000	
Seed	–	42		–	42	
Optimizer	–	AdamW		–	AdamW	
Learning rate	–	3×10^{-5}		–	3×10^{-5}	
LR scheduler	–	Cosine		–	Cosine	
Warmup steps	–	500		–	500	
Minimum LR ratio	–	0.1		–	0.1	
Weight decay	–	0.1		–	0.1	
Gradient clipping	–	1.0		–	1.0	
FSDP	–	Full shard		–	Full shard	
Data parallel	–	32		–	32	
Activation checkpointing	–	Yes		–	Yes	

Table 6: Ablation study across different model scales. We report normalized accuracy for ARC-Challenge, ARC-Easy, HellaSwag, OpenBookQA, and PIQA, and accuracy for WinoGrande.

Param	**Model**	**ARC-C/Norm**	**ARC-E/Norm**	**HS/Norm**	**OBQA/Norm**	**PIQA/Norm**	**Wino**	**Avg.**
160M	Orthros w/o SO	25.86	**47.64**	33.14	**32.20**	62.62	50.28	41.96
	Orthros w/ SO	**26.19**	46.68	**34.26**	30.80	**63.33**	51.22	**42.08**
	Transformer	25.77	46.25	33.64	30.60	62.08	**52.64**	41.83
360M	Orthros w/o SO	**28.58**	**51.68**	38.01	32.20	64.74	51.46	44.45
	Orthros w/ SO	27.90	50.25	**38.70**	32.80	65.23	**51.78**	44.44
	Transformer	28.41	50.00	38.52	**33.80**	**65.94**	50.83	**44.58**
780M	Orthros w/o SO	29.35	53.07	41.86	**35.00**	67.14	**53.59**	**46.67**
	Orthros w/ SO	**29.78**	53.41	**43.32**	33.40	67.57	51.38	46.48
	Transformer	29.44	**53.75**	42.52	33.40	**67.95**	51.30	46.39
1.3B	Orthros w/o SO	30.03	55.05	44.83	34.40	**69.31**	51.85	47.58
	Orthros w/ SO	**31.48**	56.19	**46.64**	**34.80**	69.04	**54.30**	**48.74**
	Transformer	31.06	**57.07**	45.30	33.60	68.99	53.91	48.32
2.7B	Orthros w/o SO	33.11	58.21	48.85	35.80	69.59	54.62	50.03
	Orthros w/ SO	**34.22**	**62.92**	**50.22**	**36.80**	70.84	**55.96**	**51.83**
	Transformer	33.19	59.89	48.95	35.60	**71.71**	55.88	50.87
6.8B	Orthros w/o SO	32.59	60.44	49.39	36.00	70.02	52.96	50.23
	Orthros w/ SO	**35.84**	**65.57**	**53.43**	**36.60**	**72.69**	**54.46**	**53.10**
	Transformer	33.45	60.90	49.84	34.80	70.35	52.49	50.31

Table 7: In-context recall-intensive task evaluation at 1.3B scale. Higher is better for all metrics.

Model	**DP**	**NQ**	**SW**	**TQ**	**FD**	**SQ**	**Avg.**
Trans.	23.2	**22.4**	36.7	**62.3**	16.4	**42.6**	33.9
Orthros	**23.3**	20.8	**39.9**	59.8	**24.1**	38.7	**34.4**
GDN	22.8	20.6	21.8	58.2	12.0	35.1	28.4
GLA	21.4	16.7	21.4	58.6	10.7	33.6	27.1
RetNet	21.0	15.8	17.6	57.7	8.3	33.7	25.7
SWA	22.8	14.1	17.3	61.0	16.3	41.1	28.8

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are directly supported by the method, complexity analysis, and experiments. We explicitly state the target scope as long-context autoregressive language modeling and evaluate Orthros through scaling, conversion, recall, and efficiency experiments.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Orthros is designed for long-context autoregressive workloads where prefilling dominates the overall computation and the generated output is relatively short. Its asymptotic advantage becomes less pronounced when the output length is comparable to the input length, since readout tokens then require full cross-attention over a large fraction of the sequence. Also, RL training will be explored in future works.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: The theoretical results in this paper are primarily complexity analyses of Orthros Attention under different sequence-processing regimes. We state the relevant assumptions, including the input length, output length, model width, and sliding-window size, and provide the corresponding derivations in the Method section.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper describes the Orthros architecture, training and inference schemes, evaluated model scales, training budgets, baseline architectures, benchmark tasks, and prefilling efficiency settings needed to reproduce the main conclusions.

   Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [No]

   Justification: We have provided a complete process from the algorithm to the data and the experimental setup, which is sufficient to reproduce the experimental results.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specifies the main experimental settings needed to understand the results, including model scales, training budgets, baseline architectures, evaluation benchmarks, conversion settings, long-context evaluation setup. Additional implementation and hyperparameter details are provided in the appendix or supplemental material.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

Justification: We do not report error bars or statistical significance tests because the main experiments involve large-scale pre-training and conversion training up to 7B parameters, making multiple independent runs prohibitively expensive. Instead, we evaluate Orthros across multiple model scales, architectures, benchmarks, and efficiency settings to assess the consistency of the observed trends.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We report the compute resources used for the main experiments, including GPU type, number of GPUs, sequence lengths. The appendix or supplemental material provides additional details on the configuration for the main training, conversion, evaluation, and efficiency experiments.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that the research conducted in this paper conforms to it. The work focuses on model architecture and efficiency for long-context language modeling, and does not involve human subjects or private data collection.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss the impacts of Orthros as an efficiency-oriented architecture for long-context language modeling. Potential positive impacts include reducing the computational cost and energy consumption of long-context inference and making long-context capabilities more accessible, while potential negative impacts include enabling more efficient misuse of generative models, such as scalable misinformation generation or privacy-sensitive long-context analysis.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This paper focuses on model architecture and efficiency analysis and does not release high-risk datasets or pretrained language model checkpoints at submission time.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We properly cite the original creators of the existing models, datasets, benchmarks, and software packages used in this work, including model baselines, pretrained model checkpoints, evaluation benchmarks, and implementation frameworks.

    Guidelines:

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

13. **New assets**

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

    Answer: [No]

    Justification: We have provided a complete process from the algorithm to the data and the experimental setup, which is sufficient to reproduce the experimental results.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing experiments, human-subject studies, participant recruitment, or human annotation conducted by the authors.

    Guidelines:

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

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: This paper does not involve human-subject studies, participant recruitment, or human annotation conducted by the authors. Therefore, IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development in this work does not use LLMs. Any use of LLMs is limited to editing or formatting assistance and does not affect the methodology, experiments, scientific rigor, or originality of the research.

    Guidelines:

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