ICML 2026

Caracal: Causal Architecture via Spectral Mixing

Bingzheng Gan, Tianyi Zhang, Yusu Li, Jing Huang, Wei Shi, Yangkai Ding, Tao Yu

HUAWEI NORBERT WIENER RESEARCH CENTER

Agenda

INTRODUCTION

METHODOLOGY

MOTIVATION

EXPERIMENT

CONCLUSION

Introduction: The Architectural Bottlenecks

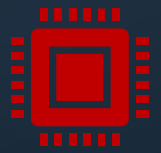
EFFICIENCY WALL

ATTENTION/GPT

Most effective but suffers from quadratic cost at context length. Becomes computationally prohibitive as sequence length grows.

HARDWARE DEBT

SSM/MAMBA



Requires specialized CUDA kernels for efficiency. Deeply coupled with hardware, making architectural innovation difficult.

CAUSALITY GAP

FFT/FNET

Inherently global by nature. Difficult to enforce causality during training, rendering them unsuitable for autoregressive tasks.

Introduction: The Caracal Solution

Addressing efficiency, portability, and causality simultaneously

`O(L log L)`

Efficient Scaling

Leveraging Fourier Spectral Mixing for near-linear long-sequence processing.

`Native PyTorch`

Hardware Agnostic

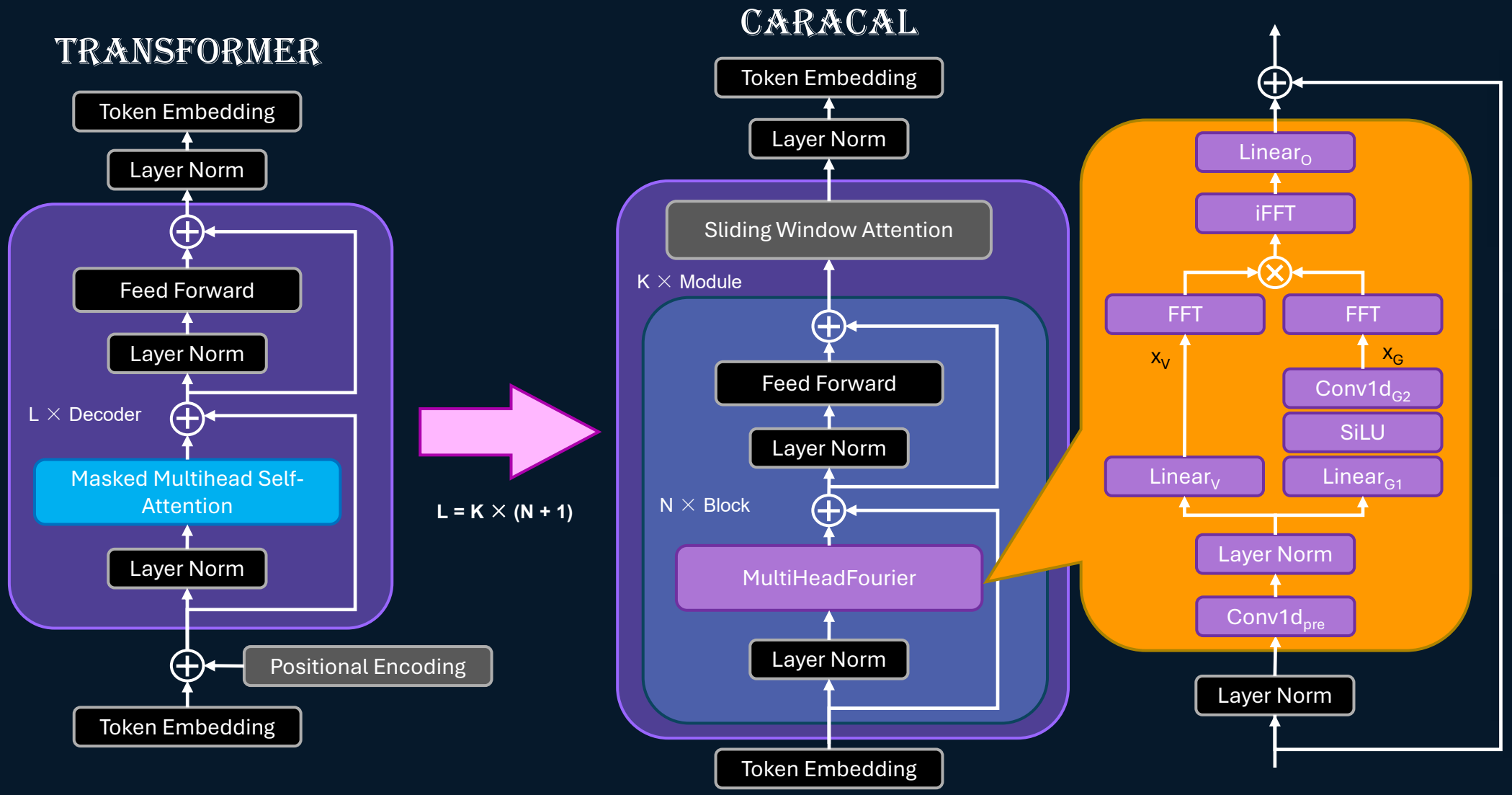
No custom CUDA kernels required. Robust portability across platform.

`Guaranteed Causality`

Rigorous Masking

Rigorous frequency-domain masking ensures perfect autoregressive behaviour.

Methodology: Overall Structure



1. Replacing MHA with MHF modules

2. Removing explicit PE module

3. Retaining selective SWA layers

Methodology: Spectral Mixing Steps

1. Local Inductive Bias

Use causal Conv1d_{pre} to capture local patterns (3-grams, k=3) and normalize it

$x_{norm} = \text{LayerNorm}(\text{Conv1d}_{pre}(x))$

2. Gated Signals

Project signals into context (Linear_V) and gated (Linear_{G1}-SiLU-Conv1d_{G2}) stream

$x_V = \text{Linear}_V(x)$

$x_G = \text{Conv1d}_{G2}(\text{SiLU}(\text{Linear}_{G1}(x_{norm})))$

3. Spectral Mixing

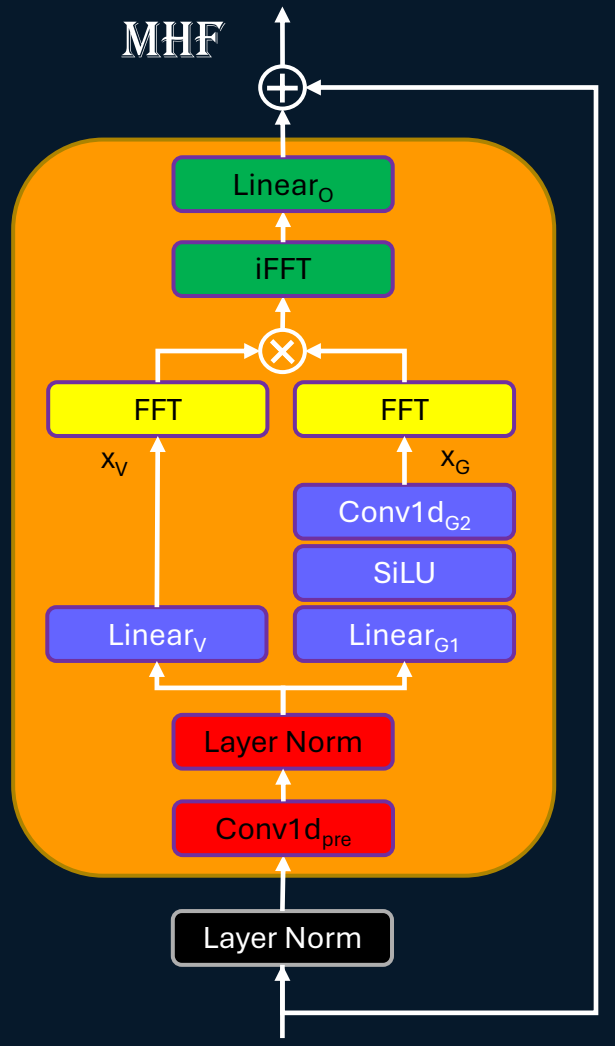
Pad to 2L to enforce causality, transform via FFT and multiply element-wise

$X_{FFT} = \text{FFT}(\text{PAD}(x_V)) \odot \text{FFT}(\text{PAD}(x_G))$

4. Reconstruction

Apply inverse FFT and truncate to L for causality, followed by Linear_O projection

$y = \text{Linear}_O(\text{TRUNCATE}(\text{iFFT}(X_{FFT})))$



Motivation: Attention vs FFT vs SSM

First-Principle View

Attention vs FFT

Both algorithm mix value vector of tokens globally via weighted sums. While attention computes dynamic, content-dependent weights at $O(L^2)$ cost, FFT relies on static, position-based complex weights at O(L log L) cost.

Causality Dilemma

Bypassing FFT Limits

Attention masks explicit weight matrix midway. FFT operates as an atomic sum with no intermediate matrix, preventing parallel token processing. Caracal bypasses this via a parallel pad-and-truncate spectral pipeline.

Data-Dependent Mixing

Bridging Mamba & FFT

Mamba adds data-dependency to S4 for adaptivity, but breaks parallel convolution, requiring low-level optimization. Caracal introduces data-dependent gating while strictly preserving parallel convolution.

Experiment: Setup & Baseline & Benchmark

Setup:

Scale all models from Tiny to Large configurations.
Scratch-Training on FineWeb-10B via unified setups.
Insert one SWA after every two MHF in Caracal.

Pre-trained on 10B FineWeb tokens with 512 context length
Global batch of 0.5M tokens using the AdamW optimizer
Cosine LR schedule peaking at 9e-4 after linear warmup
Betas (0.9, 0.95), 0.1 weight decay, and 1.0 grad clipping
Single-node training on 8x 24GB consumer-grade GPUs

Baselines:

Llama: Standard Transformer with RoPE, SwiGLU and FlashAttention;
Mamba & Mamba-2: SOTA pure SSMs with official library mamba_ssm;
Jamba: Hybrid architecture interleaving Mamba and Attention.

Model Size	d_{model}	n_{layer}	n_{head}	d_{head}
Tiny (T)	512	12	8	64
Small (S)	768	12	12	64
Medium (M)	1024	24	16	64
Large (L)	1536	24	16	96

Benchmarks:

Common Sense Reasoning: Hellaswag, Winogrande, ARC-E/C, PIQA, SIQA, BoolQ
Language Modelling: LAMNADA (reporting both Accuracy and Perplexity)
Long-Context Understanding: SWDE, FDA

Experiment: Effectiveness and Scaling

Size	Model(Params)	LMB. ppl ↓	LMB. acc ↑	Hella. acc ↑	ARC-e acc ↑	ARC-c acc ↑	Wino. acc ↑	BoolQ acc ↑	PIQA acc ↑	SIQA acc ↑	Avg. acc ↑
T	Llama(64M)	164.19	**22.53**	30.97	45.16	24.91	**50.59**	55.84	60.83	36.13	40.87
	Mamba(66M)	**129.88**	20.36	**31.87**	45.88	24.32	48.70	58.69	61.81	37.31	41.12
	Mamba2(64M)	191.20	17.62	31.36	**46.17**	24.15	49.72	55.93	60.88	36.28	40.26
	Jamba(71M)	158.41	21.99	31.29	46.13	24.57	50.20	52.97	**61.97**	**37.41**	40.82
	Caracal(63M)	219.90	20.51	30.46	44.91	**26.02**	50.28	**59.30**	61.64	36.03	**41.14**
S	Llama(124M)	79.94	**28.06**	34.26	47.98	25.34	**51.93**	56.70	62.95	36.90	43.02
	Mamba(129M)	86.33	25.25	**34.99**	**49.87**	25.60	50.20	**60.67**	63.87	**38.38**	**43.60**
	Mamba2(125M)	100.76	22.36	34.50	48.23	25.00	50.91	59.79	63.11	37.21	42.64
	Jamba(138M)	**60.48**	27.50	34.43	48.65	26.02	51.85	59.14	62.79	37.56	43.49
	Caracal(120M)	92.05	25.07	33.65	47.69	**26.96**	51.38	59.88	**64.69**	37.46	43.35
M	Llama(360M)	**32.65**	**34.06**	41.21	53.37	**29.35**	**52.41**	60.55	66.65	38.95	**47.07**
	Mamba(372M)	45.72	28.51	**41.86**	**54.17**	29.27	50.83	60.98	66.32	**39.15**	46.39
	Mamba2(357M)	51.97	29.38	41.17	53.75	29.18	50.99	60.86	66.65	38.79	46.35
	Jamba(409M)	42.44	33.09	41.62	52.95	27.73	52.17	60.00	66.05	38.69	46.54
	Caracal(345M)	38.50	32.25	39.89	51.81	28.07	52.25	**61.35**	**67.68**	38.49	46.47
L	Llama(757M)	**24.92**	**36.74**	44.97	55.22	29.44	52.64	61.47	69.21	**40.12**	48.73
	Mamba(793M)	34.34	34.02	**46.02**	59.97	31.23	51.93	**62.11**	67.03	39.66	49.00
	Mamba2(764M)	36.32	33.18	45.65	**60.61**	29.27	52.64	61.83	67.46	39.36	48.75
	Jamba(866M)	26.93	36.35	45.87	56.90	**31.40**	52.57	61.31	68.99	39.20	**49.07**
	Caracal(724M)	29.39	35.26	45.10	58.16	29.69	**53.20**	61.90	**69.26**	39.51	49.01

Direct benchmarking via Behrouz et al. (2025) recipe (340M params, 15B tokens, 4096 context length).

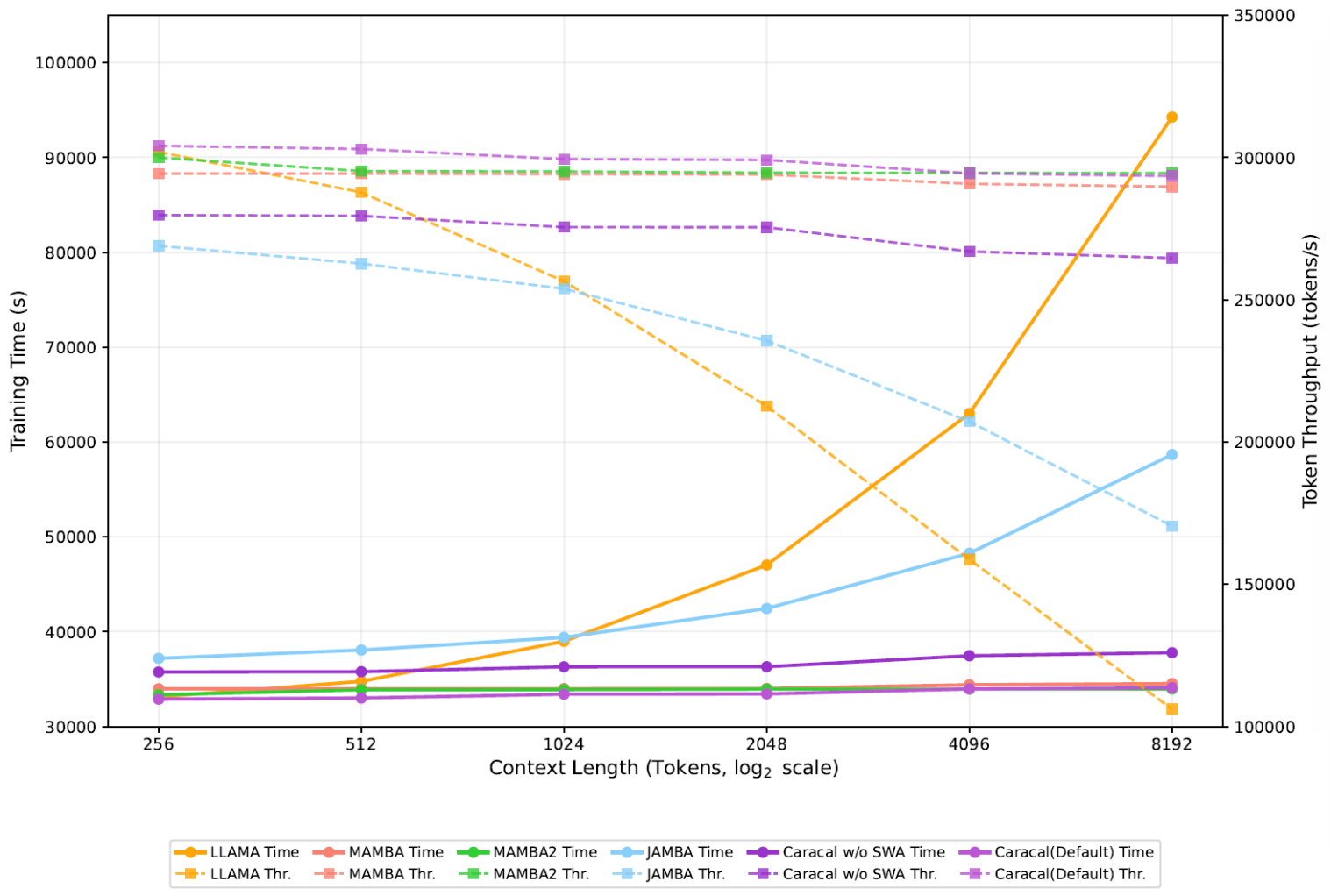
Caracal (Default) achieves SOTA; Caracal w/o SWA maintains a highly competitive rank.

Results validate robust core MHF capabilities alongside significant gains from SWA.

Caracal exhibits consistent and predictable performance gains as parameter counts increase. It achieves competitive performance with pure Transformer (Llama), pure SSM (Mamba/Mamba2) and hybrid architecture (Jamba) baselines.

Model	LMB. ppl ↓	LMB. acc ↑	Hella. acc ↑	ARC-e acc ↑	ARC-c acc ↑	Wino. acc ↑	BoolQ acc ↑	PIQA acc ↑	SIQA acc ↑	Avg. acc ↑
Transformer++	41.08	30.76	34.76	45.21	24.05	50.53	58.24	62.98	36.81	42.92
RetNet	49.73	28.24	34.15	44.27	23.62	50.91	59.72	62.61	36.79	42.54
GLA	43.02	28.73	35.96	54.19	24.29	50.00	58.39	64.05	37.13	44.09
Mamba	40.21	29.94	35.88	49.24	24.56	49.82	60.07	63.79	35.41	43.59
DeltaNet	47.30	28.43	35.95	52.68	25.37	49.63	58.79	63.52	37.96	44.04
TTT	34.19	30.06	35.71	53.01	26.11	50.08	59.83	63.97	37.32	44.51
Gated DeltaNet	30.94	34.11	38.12	55.28	26.77	51.60	59.54	63.08	34.89	45.42
Moneta	29.31	**35.70**	39.23	**55.96**	27.15	52.04	60.22	63.99	37.29	46.45
Yaad	**29.11**	34.09	39.86	54.75	**28.64**	51.12	60.29	64.93	33.82	45.94
Memora	30.44	33.68	39.17	53.40	27.99	51.23	59.29	65.21	34.10	45.51
Caracal w/o SWA	56.43	25.79	38.54	50.42	26.54	52.88	60.00	66.32	38.84	44.92
Caracal (Default)	37.27	32.70	**40.95**	50.46	27.90	**53.12**	**60.80**	**67.95**	**39.10**	**46.62**

Experiment: Efficiency and Speed



Llama exhibits quadratic complexity, causing severe speed degradation as L scales. Mamba and Mamba-2 achieve the highest speed due to their linear scaling. Hybrid model Jamba sits squarely between Transformers and SSMs. Caracal maintains an O(L log L) profile, yielding a 3x speedup over Llama at L = 8192.

The hybrid model is faster than the pure variant. Because SWA (window size 256) directly used highly optimized FlashAttention kernels.

Experiment: Ablation and Parameter

Model Variant	Avg. Acc (%)
Full Model (Default)	**49.01**
w/o SWA	48.22
w/o SWA & PC	47.82
with PE	48.94
SSLP	48.05

Ablation Study: Core Component Analysis

- Local Priors: Local inductive biases from sliding window attention (SWA) and pre-convolution (PC) are crucial for linguistic consistency, though pure MHF remains highly competitive.
- Positional Awareness: Explicit positional encoding (PE) provides no gain, confirming that MHF captures sufficient positional context.
- Gating Complexity: The two-stage design outperforms single-stage linear projection (SSLP) by developing more expressive filters.

Ratio (MHF:SWA)	Avg. Acc (%)
5:1	43.04
3:1	43.19
2:1 (Default)	**43.35**
1:1	43.26

Parameter Study: Layer Hybridization Ratio

- Optimal Ratio: Accuracy steadily improves as the introduction of SWA layers increases from a sparse 5:1 ratio, peaking at the default 2:1 ratio.
- Over-Refinement Disruption: Increasing SWA density to a 1:1 ratio degrades performance, suggesting that over-relying on SWA disrupts the synergy between frequency-domain global mixing and time-domain precision.

Conclusion

Core Architectural Contributions

- O(L log L) Efficiency: Addresses context-scaling bottlenecks by replacing global attention with a Fourier-based gated mixing module.
- Hardware Portability: Eliminates low-level kernels by relying exclusively on standard, universally optimized library operators.
- Guaranteed Causality: Overcomes the constraints of prior spectral models via a frequency-domain causal masking for autoregressive tasks.

Future work

- Frontier-Scale Scaling: Validates the architecture up to current parameters, with large-scale pre-training still required to verify its potential for trillion-parameter frontier models.
- Localized Retrieval: Targets the "resolution gap" in fine-grained information extraction, planning future architectural refinements like multi-scale gating to enhance localized retrieval.

Thank you