ECE7115 ~~Multimodal VLM~~ LLM

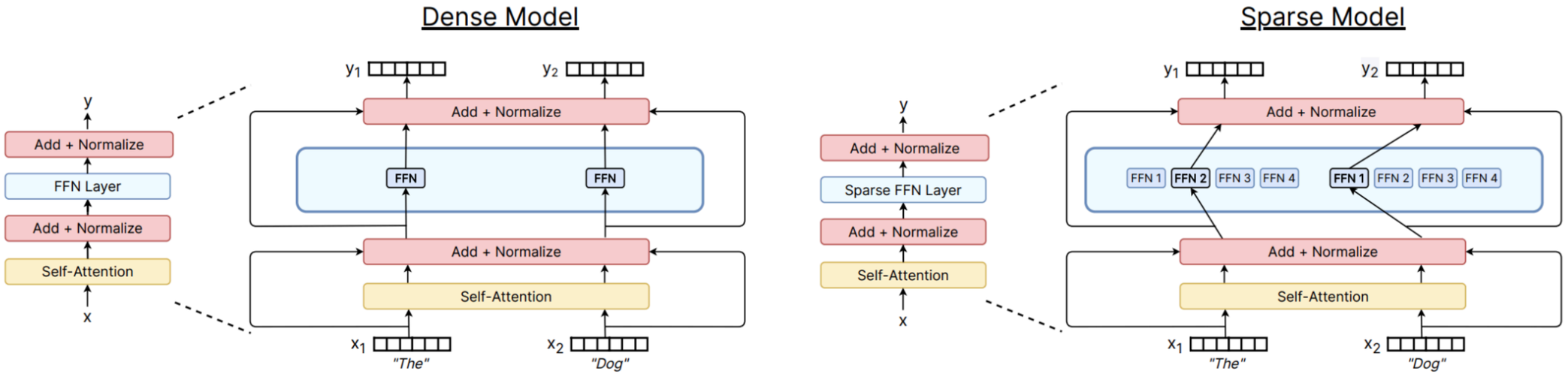
7. LLM Case Study: Architecture

Spring 2026

Namhyuk Ahn, Inha University

Last Week: MoE

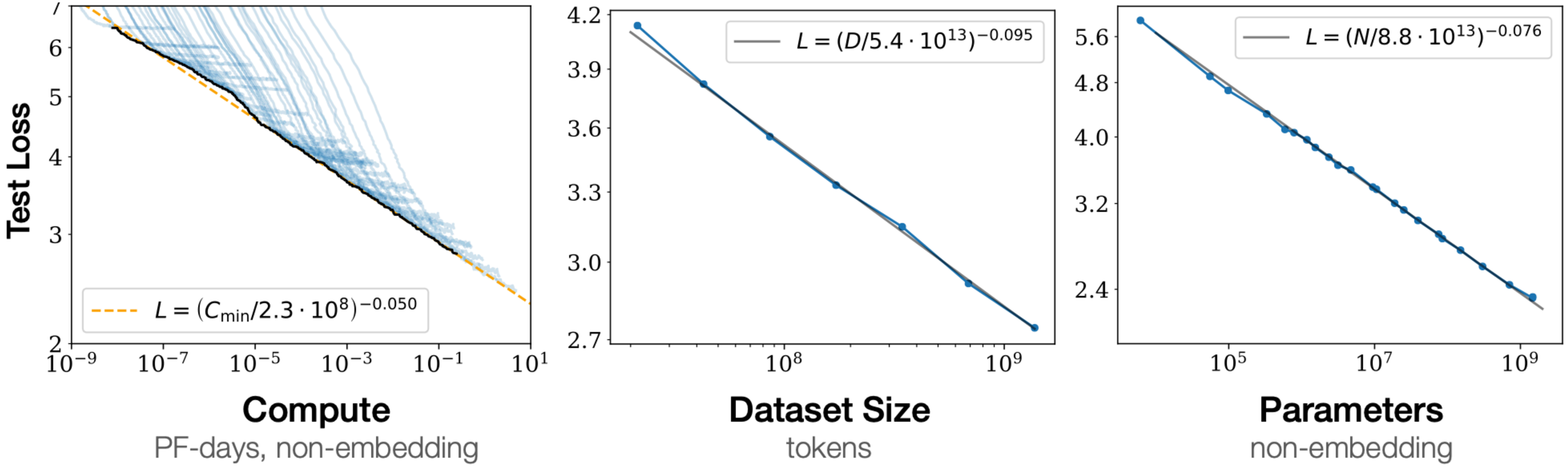
- MoE is a technique that uses many different sub-models (or "experts") to improve the quality of LLMs



- A very common architecture in recent models

Last Week: Scaling Laws

- Scaling laws: Simple and predictive rules for model performance
  - Old way: tune hyperparameters on big models
  - New way: tune on small models, extrapolate to large ones



Last Week: Scaling Laws

- Scaling laws are really surprising and useful tools

- Data scaling: understand how data affects models
- Model scaling: dramatically reduce costs for training
- Scaling as prediction: understand what problems can be brute forced

- Without scaling laws, we might…
  - Waste millions of dollars on "blind" trial-and-error
  - Produce "under-trained" models without knowing it
  - Fail to distinguish between "bad architecture" and "insufficient scale"
  - Lose the "extrapolation" advantage (train and pray 😢)

Image Credit: https://sebastianraschka.com/llm-architecture-gallery/

really nice blog post!

LLM: Architectural Change

Lecture Overview

- Walk through LLM models chronologically

- Part 1: 2023 - 2024 Q1
  - Key models: LLaMA v1-2, Mistral, Gemma, OLMo, DeepSeekMoE, …

- Part 2: 2024 Q1 - 2024 Q4
  - Key models: Qwen2, DeepSeek v2-3, Gemma v2, MiniMax-01, …

- Part 3: 2025 -
  - Key models: Gemma 3, Kimi, Qwen3, GLM, some Korean models, …

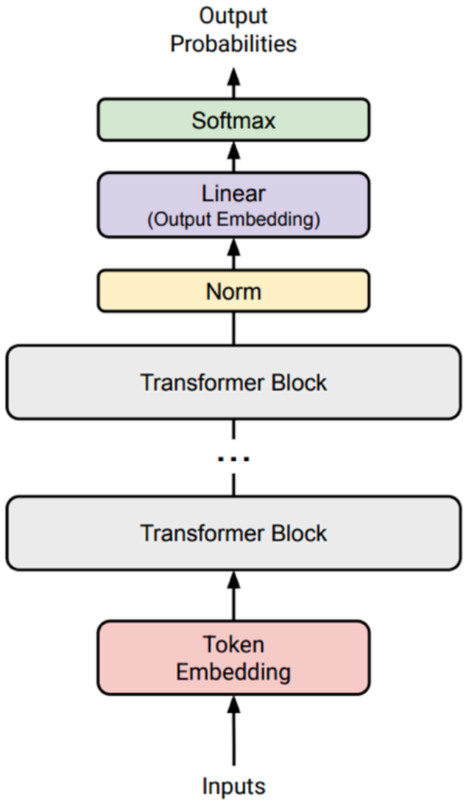
Disclaimer…

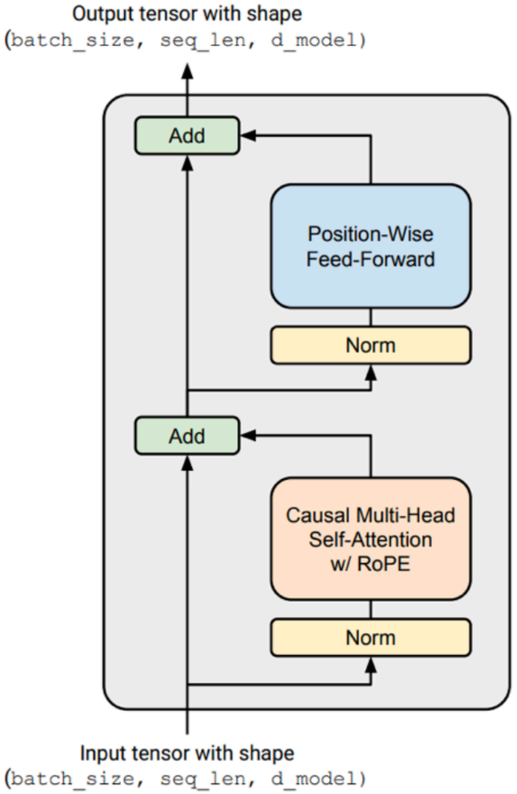
- Please note that some specs or details may be inaccurate
  - Some specs may be inaccurate; reviewing 20–30 technical reports made exhaustive verification
  - Certain settings only exist in the codebase, making manual verification of every detail highly difficult 😢 😢 😢
- I may have missed some key models, or included less crucial ones
- If you notice any errors or missing information, please let me know!

Part 1: 2023 - 2024 Q1

Recall: LLaMA Recipe [Touvron+ 2023]

- RMSNorm with pre-norm placement
- RoPE
- SwiGLU FFN, no bias
  - `d_ff = 8/3 x d_model`
- MHA
- Follow Chinchilla-optimal
  - Small model, more tokens





LLaMA: Hyperparameters [Touvron+ 2023]

Model	Params	Layers	d_model	Heads	KV Heads	Context	Vocab
7B	6.7B	32	4096	32	32 (MHA)	2048	32K
13B	13.0B	40	5120	40	40 (MHA)	2048	32K
33B	32.5B	60	6656	52	52 (MHA)	2048	32K
65B	65.2B	80	8192	64	64 (MHA)	2048	32K

- Optimizer: AdamW with LR cosine schedule
- Batch size: 4M tokens, Weight decay: 0.1
- Dataset: 1.0T (7B/13B), 1.4T (33B/65B) tokens (public only)
- Tokenizer: SentencePiece BPE

LLaMA: Takeaway [Touvron+ 2023]

- Established the "LLaMA recipe" as the de facto standard
  - RMSNorm + pre-norm + RoPE + SwiGLU + no bias

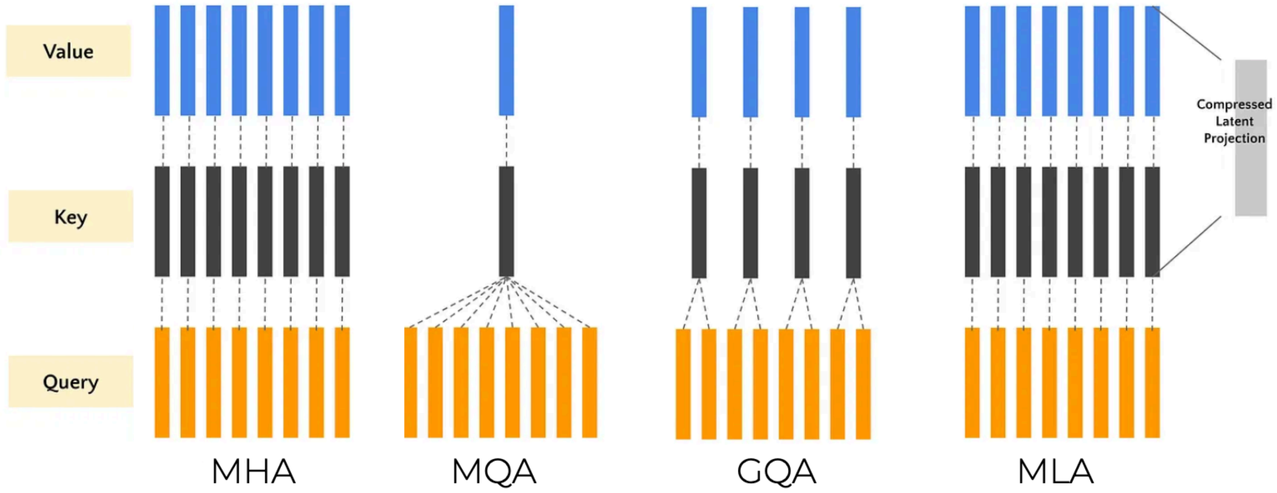
- Opened the era of open-weight foundation models

LLaMA 2: What Changed? [Touvron+ 2023]

- Architecture: Same as LLaMA 1 with two key changes:
  - Context length: 2K → 4K
  - GQA introduced for 34B and 70B models (8 KV heads)
  - It was one of the first major open models to adopt GQA at scale

  - But, 7B/13B still use MHA (same as LLaMA 1)

- Training data: 1.4T → 2.0T tokens



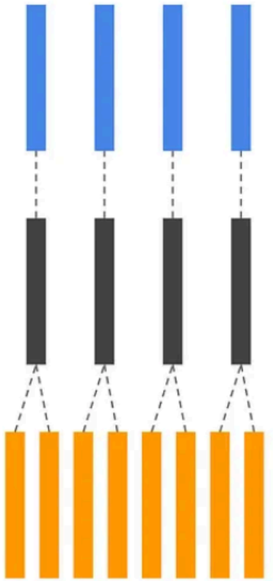
LLaMA 2: Hyperparameters [Touvron+ 2023]

Model	Params	Layers	d_model	Heads	KV Heads	d_ff	Context	Vocab
7B	7B	32	4096	32	32 (MHA)	11008	4096	32K
13B	13B	40	5120	40	40 (MHA)	13824	4096	32K
34B	34B	48	8192	64	8 (GQA)	22016	4096	32K
70B	70B	80	8192	64	8 (GQA)	28672	4096	32K

- Total compute: 3.3M A100 GPU hours
- Note: Training loss still not saturated after 2T tokens

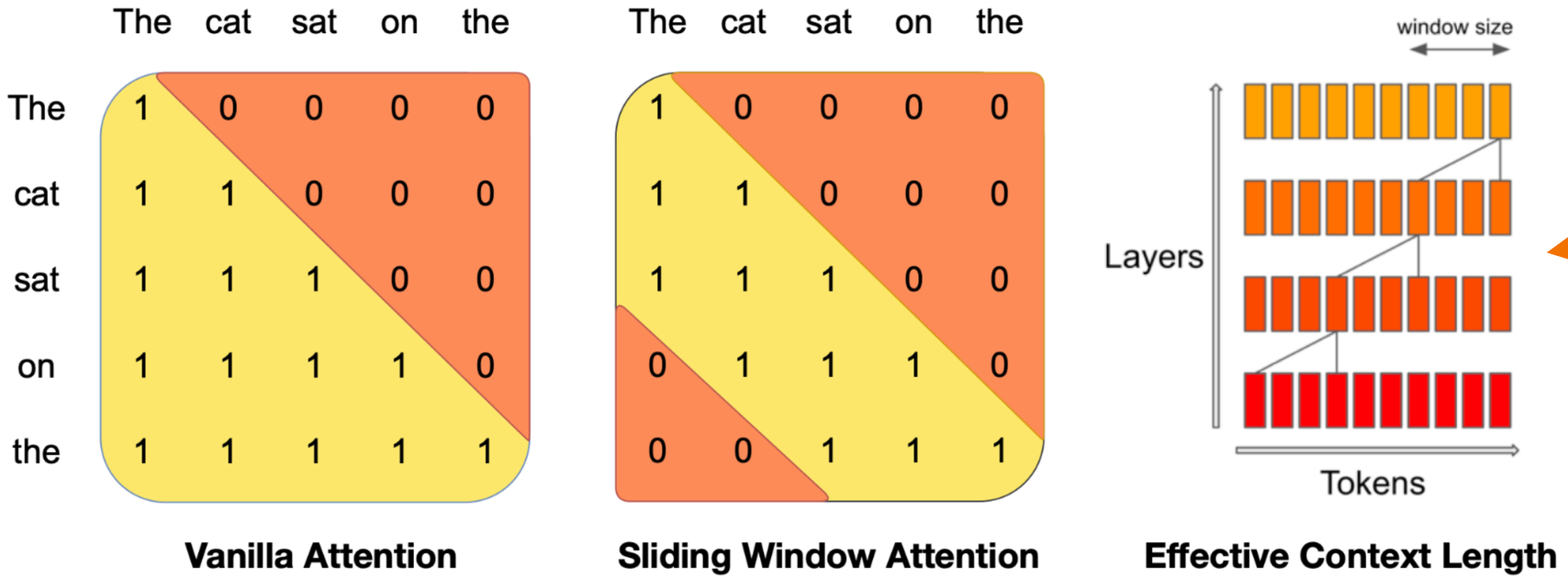
Mistral 7B [Jiang+ 2023]

- Basically, LLaMA recipe (RMSNorm, pre-norm, RoPE, SwiGLU)
- 8,192 context length
- Use GQA (32 query heads, 8 KV heads)
- Introduce sliding window attention (SWA)
  - Rolling buffer KV-cache
  - Pre-fill and chunking



GQA

Mistral 7B: Sliding Window Attention [Jiang+ 2023]



with 32 layers, theoretically the model can see 131K tokens

Figure 1: Sliding Window Attention. The number of operations in vanilla attention is quadratic in the sequence length, and the memory increases linearly with the number of tokens. At inference time, this incurs higher latency and smaller throughput due to reduced cache availability. To alleviate this issue, we use sliding window attention: each token can attend to at most W tokens from the previous layer (here, $W = 3$). Note that tokens outside the sliding window still influence next word prediction. At each attention layer, information can move forward by W tokens. Hence, after k attention layers, information can move forward by up to $k \times W$ tokens.

W = 4096, context length = 8192

Mistral 7B: Sliding Window Attention [Jiang+ 2023]

- Rolling buffer KV-cache
  - Now KV-cache can store up to W KV context, not # context length
  - Rolls KV-cache to overwrite the oldest context

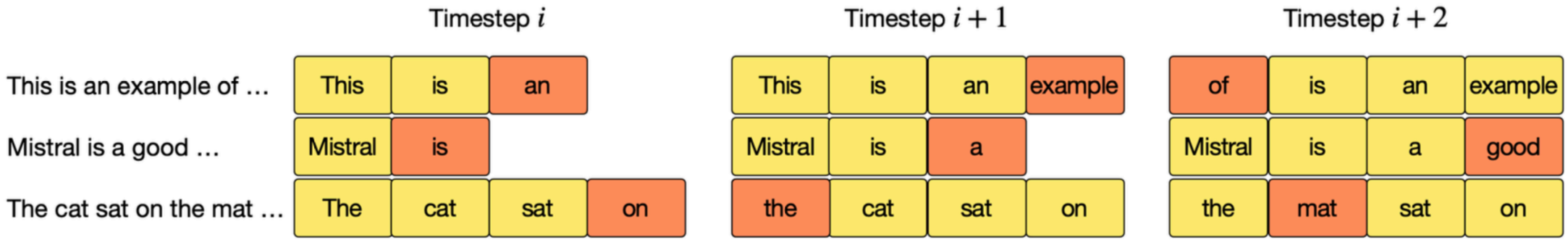


Figure 2: Rolling buffer cache. The cache has a fixed size of $W = 4$. Keys and values for position i are stored in position $i \bmod W$ of the cache. When the position i is larger than W, past values in the cache are overwritten. The hidden state corresponding to the latest generated tokens are colored in orange.

Mistral 7B: Sliding Window Attention [Jiang+ 2023]

- Pre-fill and chunking
  - Slices the pre-fill context to reduce memory peak

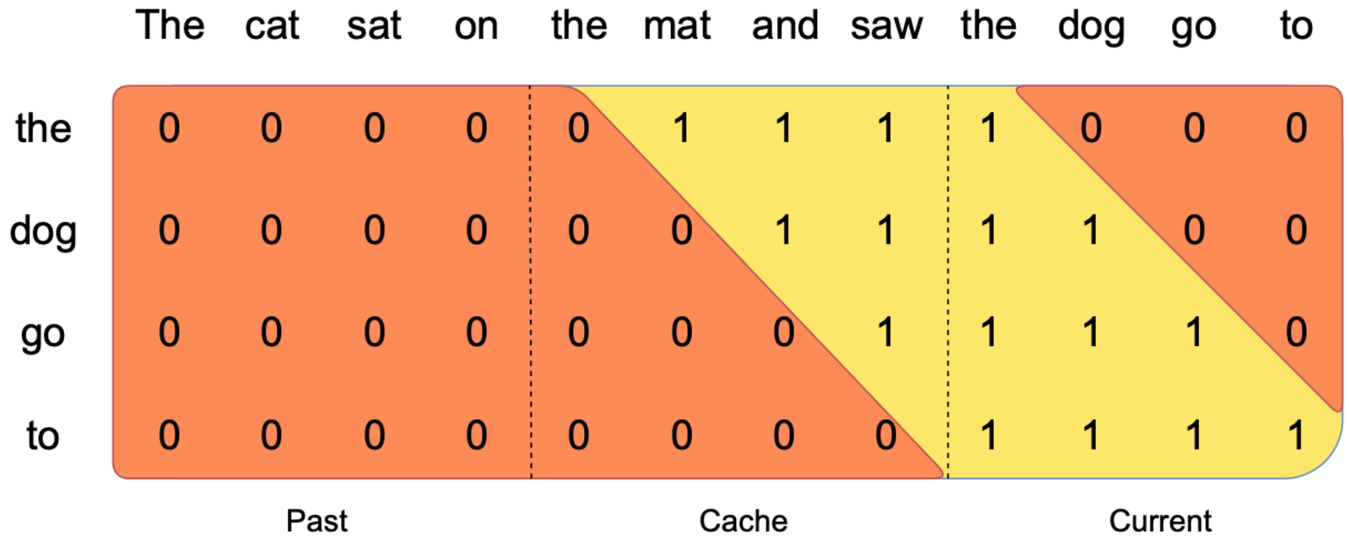


Figure 3: Pre-fill and chunking. During pre-fill of the cache, long sequences are chunked to limit memory usage. We process a sequence in three chunks, "The cat sat on", "the mat and saw", "the dog go to". The figure shows what happens for the third chunk ("the dog go to"): it attends itself using a causal mask (rightmost block), attends the cache using a sliding window (center block), and does not attend to past tokens as they are outside of the sliding window (left block).

Mistral 7B: Hyperparameters [Jiang+ 2023]

Model	Params	Layers	d_model	Heads	KV Heads	d_ff	Context	Vocab
Mistral 7B	7B	32	4096	32	8	14336	8192	32K

- Window size W = 4096

Mixtral 8x7B [Jiang+ 2024]

- Same base as Mistral 7B, but every FFN is replaced by MoE
  - Total/active params: 47B / 13B

$$y = \sum_{i=0}^{n-1} \mathbf{Softmax}(\mathbf{Top2}(x \cdot W_g))_i \cdot \mathbf{SwiGLU}_i(x).$$

routing logic

- Full 32K context (no sliding window, unlike Mistral 7B)
  - GQA: 32 Q heads, 8 KV heads

	Mixtral 8x7B
Total Params	47B
Active Params	13B
Layers	32
d_model	4096
d_ff	14336
Experts	8
Top-K	2
Context Length	32K

Mixtral 8x7B: Takeaway [Jiang+ 2024]

- 5x fewer active params than LLaMA 2 70B, comparable performance

Model	Active Params	MMLU	HellaS	WinoG	PIQA	Arc-e	Arc-c	NQ	TriQA	HumanE	MBPP	Math	GSM8K
LLaMA 2 7B	7B	44.4%	77.1%	69.5%	77.9%	68.7%	43.2%	17.5%	56.6%	11.6%	26.1%	3.9%	16.0%
LLaMA 2 13B	13B	55.6%	80.7%	72.9%	80.8%	75.2%	48.8%	16.7%	64.0%	18.9%	35.4%	6.0%	34.3%
LLaMA 1 33B	33B	56.8%	83.7%	76.2%	82.2%	79.6%	54.4%	24.1%	68.5%	25.0%	40.9%	8.4%	44.1%
LLaMA 2 70B	70B	69.9%	**85.4%**	**80.4%**	82.6%	79.9%	56.5%	25.4%	**73.0%**	29.3%	49.8%	13.8%	69.6%
Mistral 7B	7B	62.5%	81.0%	74.2%	82.2%	80.5%	54.9%	23.2%	62.5%	26.2%	50.2%	12.7%	50.0%
Mixtral 8x7B	13B	**70.6%**	84.4%	77.2%	**83.6%**	**83.1%**	**59.7%**	**30.6%**	71.5%	**40.2%**	**60.7%**	**28.4%**	**74.4%**

- First successful large-scale open-weight MoE model

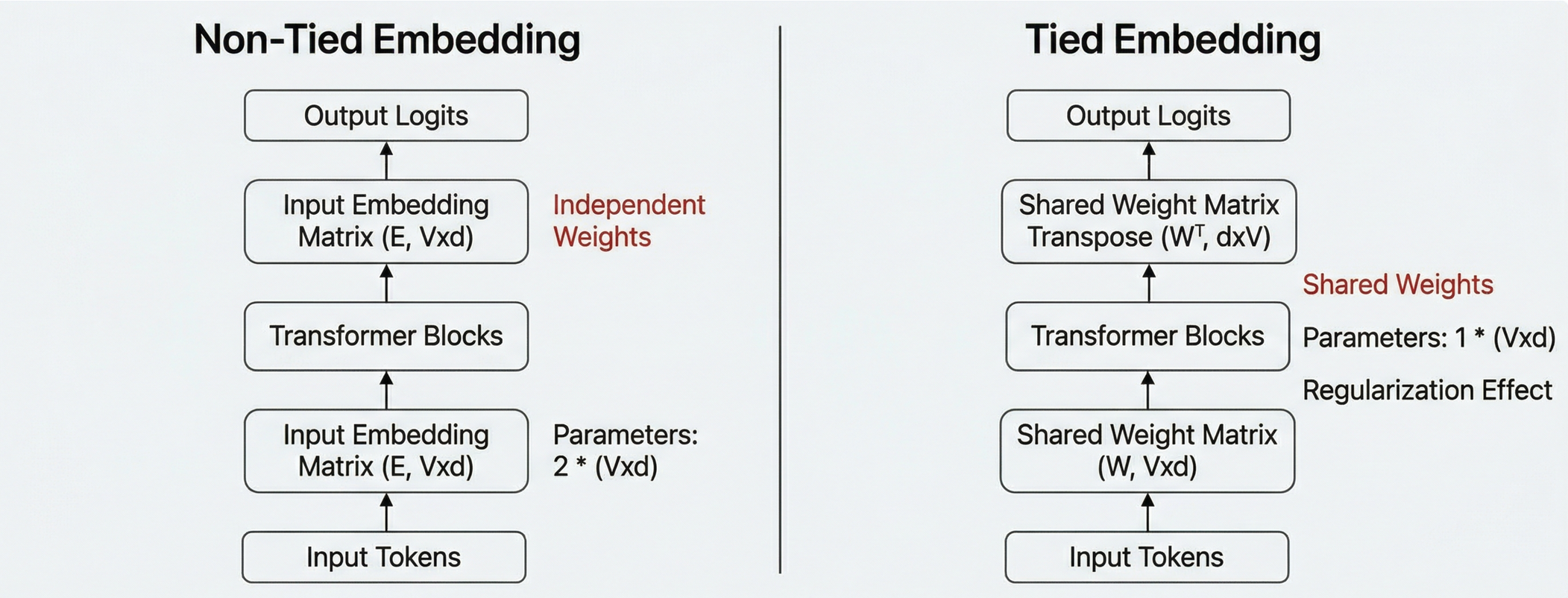
Gemma 1 [Gemma Team 2024]

- Google's first open-weight model family

- Architecture choices (some different from LLaMA recipe):
  - RMSNorm + Pre-norm + RoPE (same as LLaMA)
  - GeGLU instead of SwiGLU
  - Very large vocab: 256K (vs. LLaMA's 32K); tied embedding
  - Training tokens: 3T (2B) / 6T (7B)

Model	Params	Layers	d_model	Heads	KV Heads	d_ff	Context	Vocab
2B	~2.5B	18	2048	8	1 (MQA)	32768	8K	256K
7B	~8.5B	28	3072	16	16 (MHA)	49152	8K	256K

Tied Embedding [Press+ 2017]



OLMo [Groeneveld+ 2024]

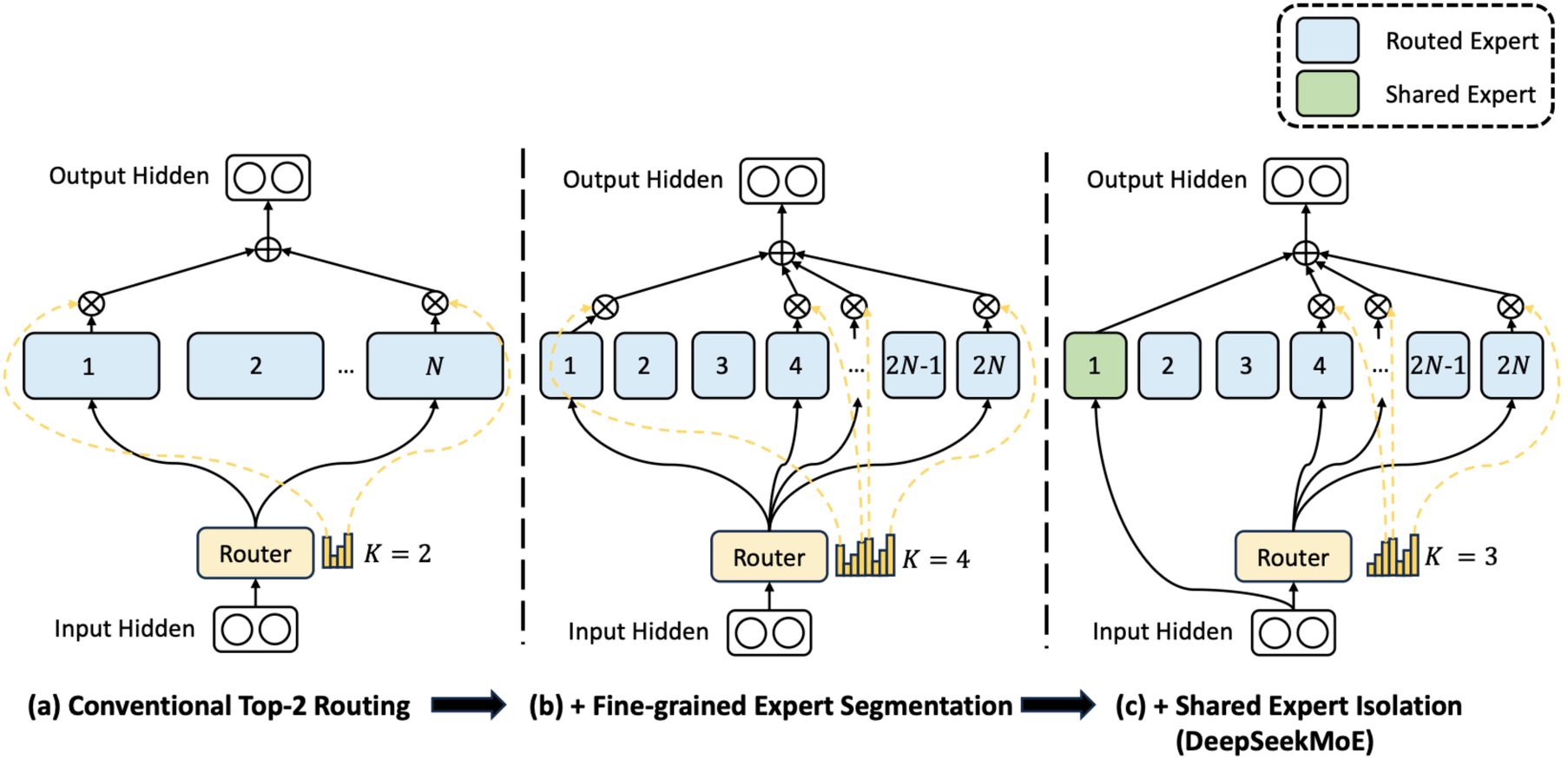
- Title: "OLMo: Accelerating the Science of Language Models"
  - Fully public: data (Dolma), code, checkpoints (500+), training logs

- Deliberate architectural choices that differ from the standard:
  - Non-parametric LayerNorm (not RMSNorm) for stability + speed
  - MHA (no GQA), linear LR schedule (not cosine)
  - All biases removed (same as LLaMA)

Model	Params	Layers	d_model	Heads	KV Heads	d_ff	Context	Vocab
1B	1B	16	2048	16	16 (MHA)	5504	2K	50K
7B	7B	32	4096	32	32 (MHA)	11008	2K	50K

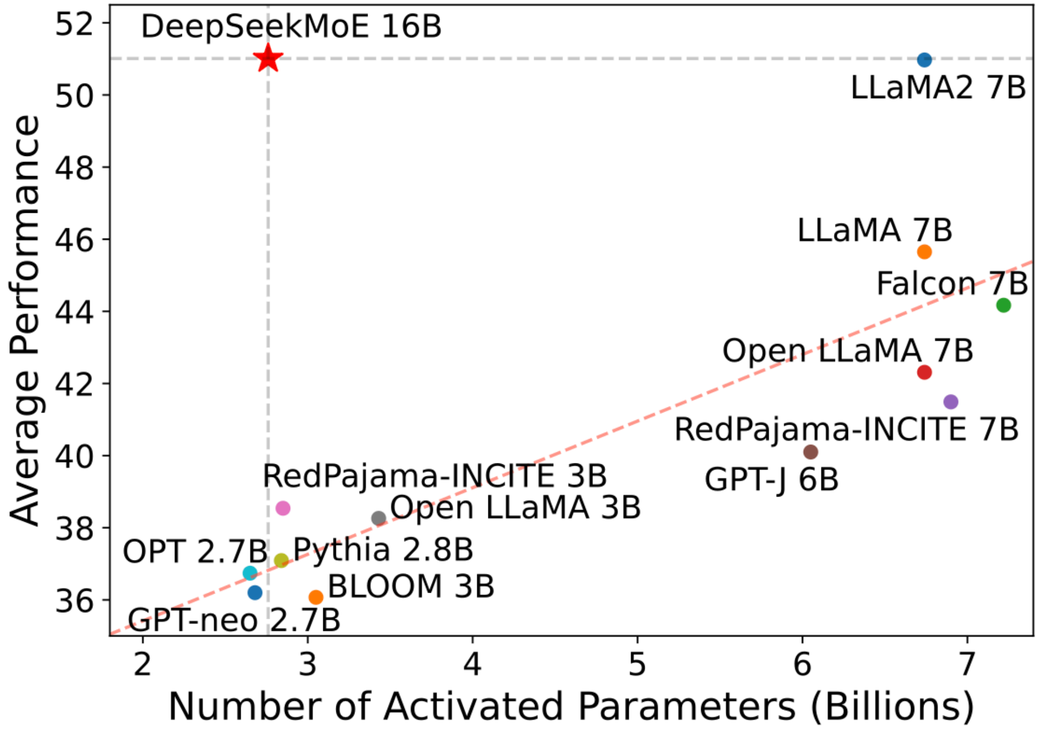
DeepSeekMoE [Dai+ 2024]

- Fine-grained experts and shared expert isolation



DeepSeekMoE [Dai+ 2024]

- DeepSeekMoE 16B achieves similar performance to DeepSeek 7B (dense) with ~40% compute
- DeepSeekMoE 145B matches DeepSeek 67B with ~28.5% compute



Load Balancing [Dai+ 2024]

- Expert-level balance loss
  - We saw this loss in MoE slide
  - Switch Transformer [Fedus+ 2022]

- Device-level balance loss
  - Balance across the devices

  - Partition all routed experts into D groups and deploy each group on a single device

$$\mathcal{L}_{\text{ExpBal}} = \alpha_1 \sum_{i=1}^{N'} f_i P_i,$$

$$f_i = \frac{N'}{K'T} \sum_{t=1}^{T} \mathbb{1}(\text{Token } t \text{ selects Expert } i),$$

$$P_i = \frac{1}{T} \sum_{t=1}^{T} s_{i,t},$$

$$\mathcal{L}_{\text{DevBal}} = \alpha_2 \sum_{i=1}^{D} f'_i P'_i,$$

$$f'_i = \frac{1}{|\mathcal{E}_i|} \sum_{j \in \mathcal{E}_i} f_j,$$

$$P'_i = \sum_{j \in \mathcal{E}_i} P_j,$$

DeepSeekMoE: Hyperparameters [Dai+ 2024]

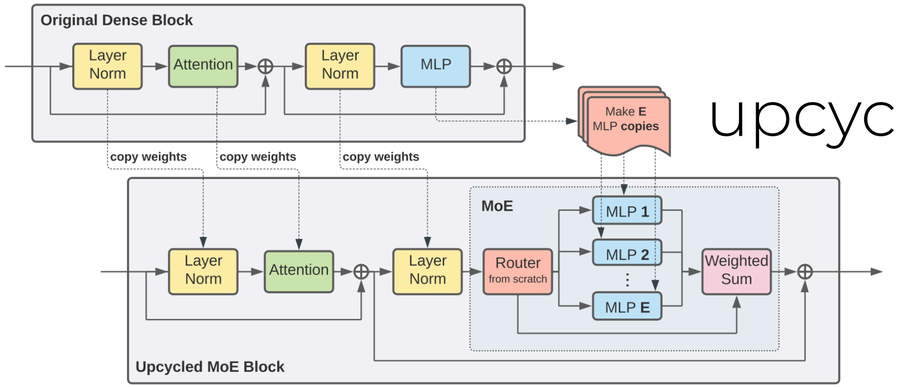
	2B (validation)	16B	145B
Total Params	~2.0B	~16.4B	~144.6B
Active Params	~0.3B	~2.8B	~22.2B
Layers	9	28	62
d_model	1280	2048	4096
Heads	10 (MHA)	16 (MHA)	32 (MHA)
Total Experts	1s + 63r	2s + 64r	4s + 128r
Active Experts	1s + 7r	2s + 6r	4s + 12r
Expert Size	0.25x FFN	0.25x FFN	0.125x FFN
Training Tokens	100B	2T	245B

s: shared
r: routed

MiniCPM [Hu+ 2024]

- Title: "MiniCPM: Unveiling the Potential of Small Language Models"

- Key contribution
  - WSD learning rate scheduler
  - Extensive scaling experiments
  - MoE upcycling (dense → init MoE)



Model	Params	Layers	d_model	Heads	KV Heads	d_ff	Vocab (tied)
1.2B	1.2B	52	1536	24	8 (GQA)	3840	73440
2.4B	2.4B	40	2304	36	36 (MHA)	5760	122753

MiniCPM: Scaling Laws Experiments [Hu+ 2024]

- Key technique: they use μP initialization [Yang+ 2022]
  - + Fix the aspect ratio, and scale up the overall model size

Name	N (B)	d_m	d_{ff}	d_h	n_h	L
9M	0.009	320	800	64	5	8
30M	0.036	512	1280	64	8	12
70M	0.066	640	1600	64	10	14
0.1B	0.109	768	1920	64	12	16
0.17B	0.166	896	2240	64	14	18
0.2B	0.241	1024	2560	64	16	20
0.5B	0.499	1344	3360	64	21	24

- Optimal batch size and optimal learning rate

MiniCPM: Scaling Laws Experiments [Hu+ 2024]

- `scale_emb = 12, scale_depth = 1.4, init_std = 0.1, lr = 0.01`

Name	Specific Operation
Embedding Output Scaling	Multiply the output of the embedding by $scale\_emb$
Residual Connection Scaling	Scale the output tensor of a block before adding to each residual connection in each layer by $scale\_depth / \sqrt{\text{num\_layers}}$
Initialization of Tensors	Set the initialization standard deviation of each two-dimensional tensor parameter to $init\_std / \sqrt{d_m / d_{base}}$, and set other parameters' initialization to 0.1
Learning Rate Scaling of Tensors	Adjust the learning rate of each two-dimensional tensor parameter to $1/(d_m/d_{base})$ times the learning rate of other parts (or the overall learning rate)
LM Head Scaling	Adjust the output logits to $1/(d_m/d_{base})$ times the original value

Table 7: List of operations used when applying tensor program techniques.

MiniCPM: Scaling Laws Experiments [Hu+ 2024]

- Optimal batch size

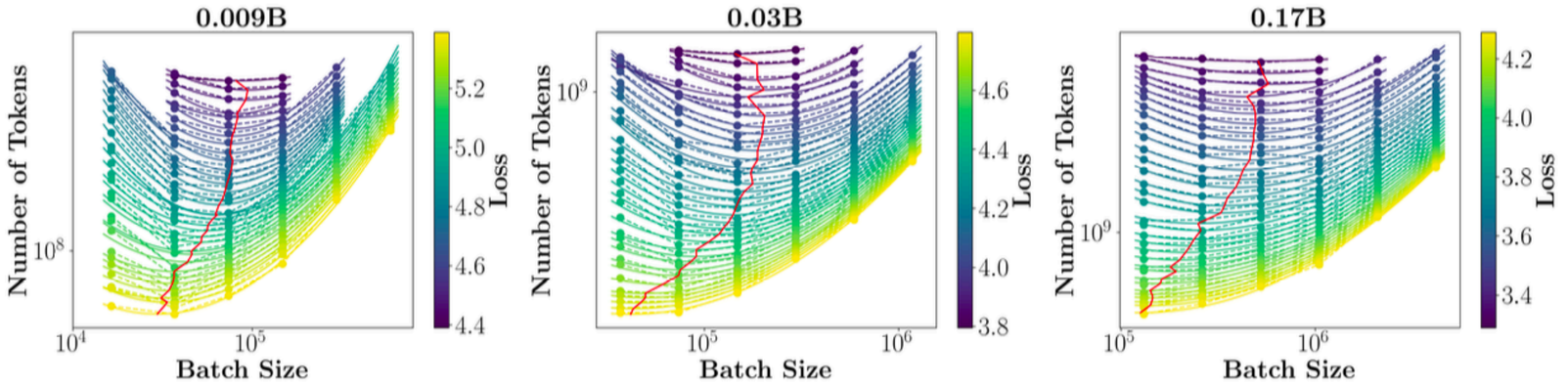


Figure 1: We demonstrate the loss curve of three size models trained using different batch sizes. Each vertical line formed by points with a gradient color represents a training curve. Lighter colors denote higher loss.

similar trend to Kaplan's experiment

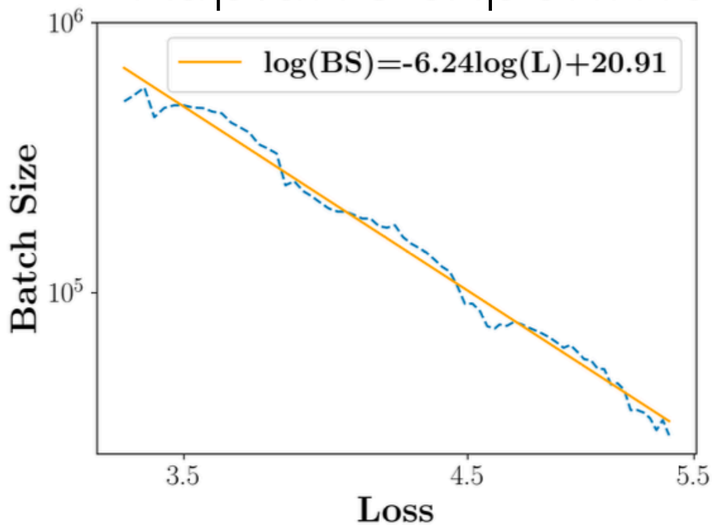


Figure 2: The connected optimal batch sizes.

MiniCPM: Scaling Laws Experiments [Hu+ 2024]

- Optimal learning rate
  - With μP initialization, optimal learning rate does not shift very much
  - Even when we scale up the model from 0.04b to 2.1b, relative similar

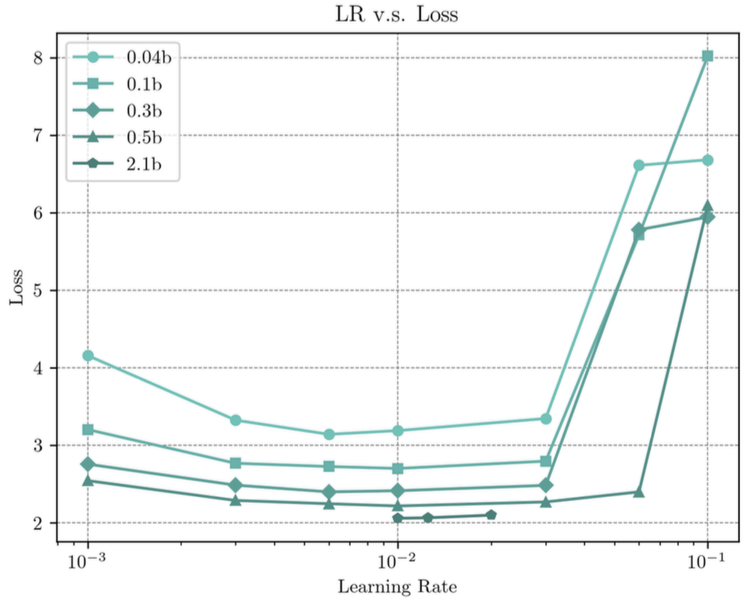
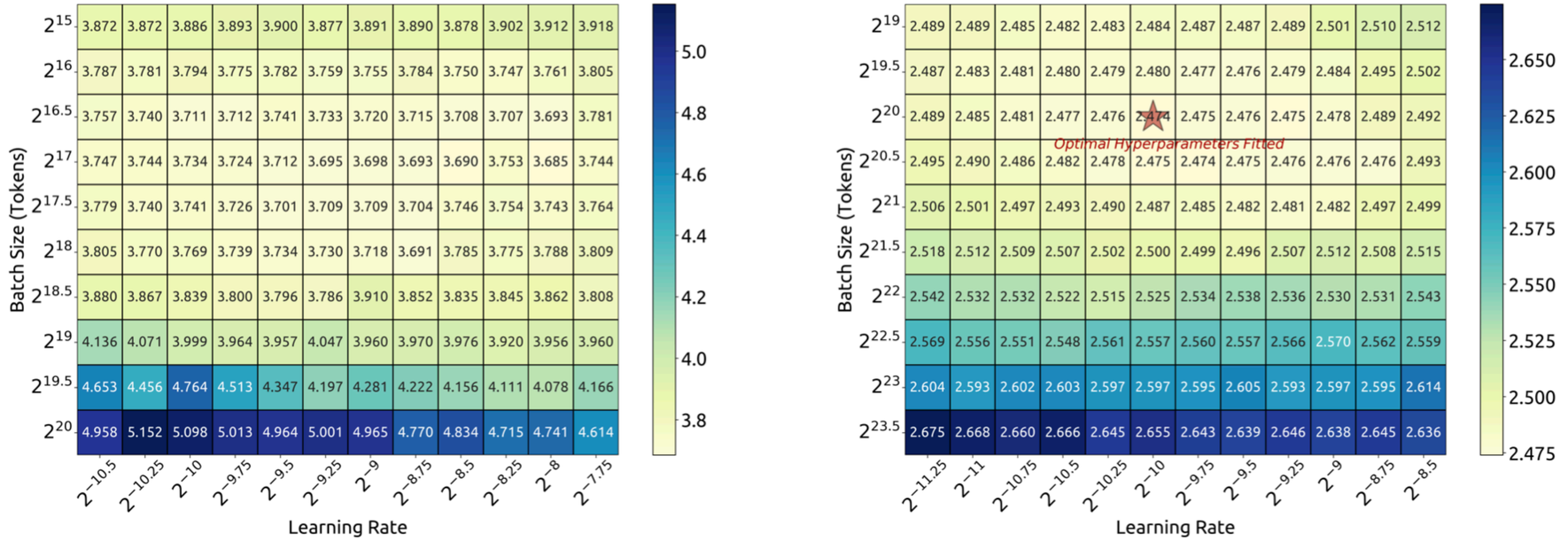


Figure 3: Loss vs Learning Rate. After applying for the Tensor Program, the learning rate shift becomes minimal.

μP

Side Note: Without μP?

- How to find optimal learning rate and batch size without μP?
  - DeepSeek v1 [Bi+ 2024] estimate these using scaling laws



(a) 1e17 FLOPs (177M FLOPs/token)

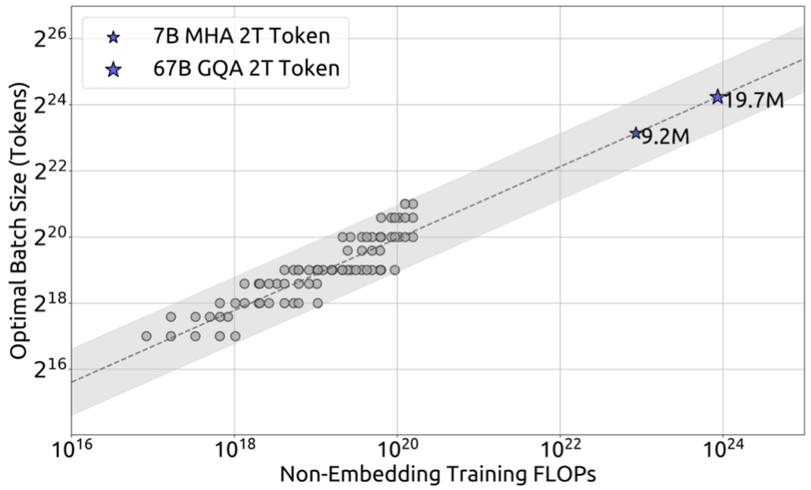
(b) 1e20 FLOPs (2.94B FLOPs/token)

Figure 2 | Training loss w.r.t. batch size and learning rate with 1e17 and 1e20 FLOPs.

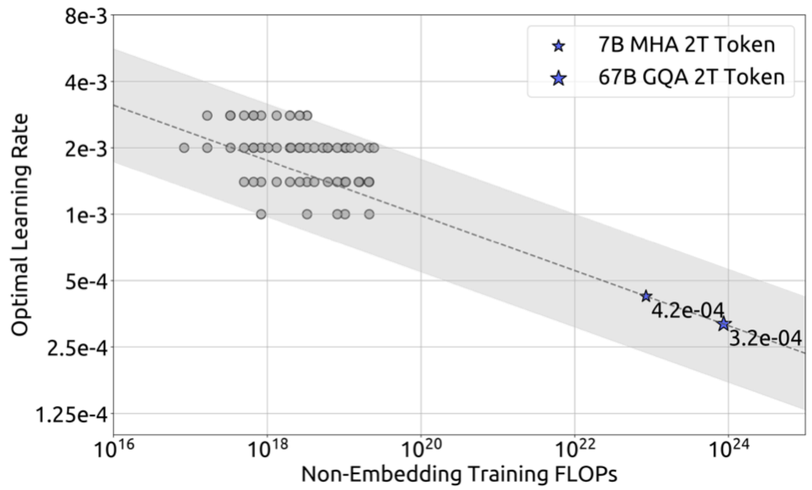
Side Note: Without μP?

- How to find optimal learning rate and batch size without μP?
  - DeepSeek v1 [Bi+ 2024] estimate these using scaling laws

$$\eta_{\text{opt}} = 0.3118 \cdot C^{-0.1250}$$
$$B_{\text{opt}} = 0.2920 \cdot C^{0.3271} \quad (1)$$



(a) Batch size scaling curve

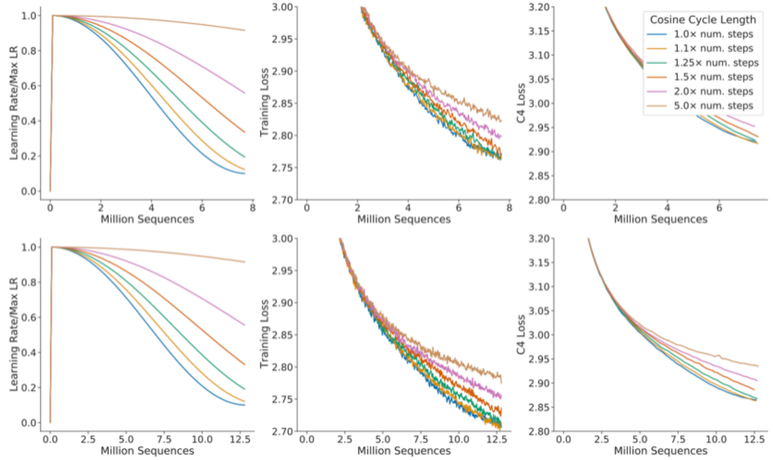


(b) Learning rate scaling curve

Recall: Chinchilla Scaling Law

What Makes the Difference?

- Improperly configured cosine learning rate decay
- To get optimal loss, we should set **`T_max`** to actual end-training steps
  - But they used fixed scheduler so LR does not drop appropriately

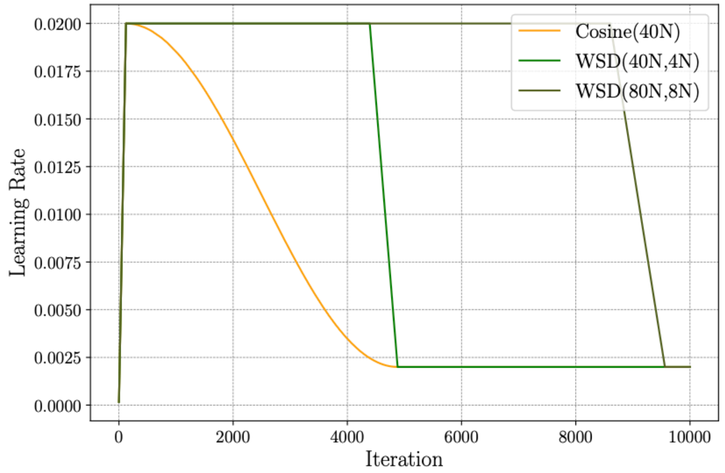


[Hoffman+ 2022]

- From chinchilla
  - To fit a scaling law, we need to train from scratch, not just early stop
  - This turns the cost of fitting a scaling law from $O(N)$ to $O(N^2)$...

Solution? WSD LR Scheduler [Hu+ 2024]

- Warmup-Stable-Decay (WSD) LR scheduler
  - Split learning rate into warmup, stable, and decay phases

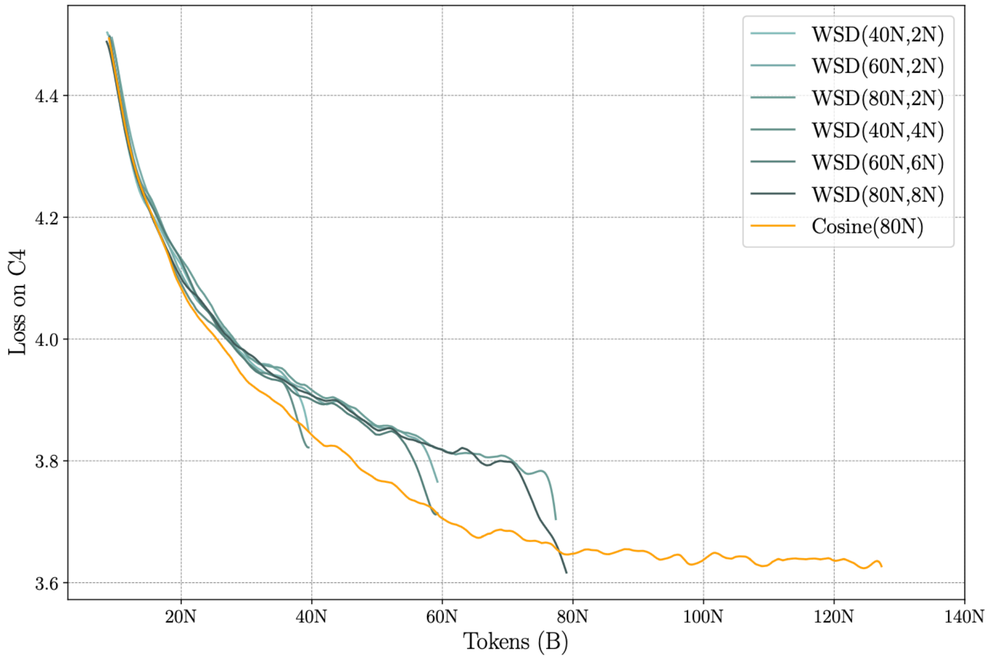


$$WSD(T;s) = \begin{cases} \frac{s}{W}\eta, & s < W \text{ warmup} \\ \eta, & W < s < T \text{ stable} \\ f(s-T)\eta, & T < s < S \text{ decay} \end{cases}$$

portion: ~ 10%

- 👍 Any stable checkpoint → add decay → valid model
- 👍 Chinchilla analysis: we can restart the run at the end of the stable phase
  - Scaling law measurement at $O(N)$ cost, not $O(N^2)$

Solution? WSD LR Scheduler [Hu+ 2024]



- During stable phase: large weight updates, minimal loss reduction
- During decay phase: small updates, dramatic loss drop
  - Decay phase = only 10% of total training → sufficient convergence

Model Comparison: 2023 - 2024 Q1

Model	Type	Activation	Norm	Attention	# Tokens
LLaMA 1 (23Q1)	Dense	SwiGLU	RMSNorm	MHA	1.0 - 1.4T
LLaMA 2 (23Q3)	Dense	SwiGLU	RMSNorm	MHA / GQA	2.0T
Mistral 7B (23Q4)	Dense	SwiGLU	RMSNorm	SWA + GQA	-
Mixtral 8x7B (24Q1)	MoE	SwiGLU	-	GQA	-
Gemma 1 (24Q1)	Dense	GeGLU	RMSNorm	MQA / MHA	3 - 6T
OLMo (24Q1)	Dense	SwiGLU	NP LayerNorm	MHA	2 - 2.5T
DeepSeekMoE (24Q1)	MoE	SwiGLU	RMSNorm	MHA	2T
MiniCPM (24Q2)	Dense / MoE	SwiGLU	RMSNorm	MHA / GQA	1.1T

Key Takeaway: 2023 - 2024 Q1

- LLaMA recipe remains dominant, but with growing variations
  - Dominant SwiGLU, RMSNorm, RoPE

- Rising of mixture-of-experts
  - 8 experts, top-2 (Mixtral) → 64+ experts, top-8 (DeepSeekMoE)
  - All routed (Mixtral) → shared expert isolation(DeepSeekMoE)

- Cosine LR scheduling → WSD (MiniCPM)

- More vocab size (32k; LLaMA → 256K; Gemma) and data scale (1T → 6T)
  - But still, very small context length

Part 2: 2024 Q2 - 2024 Q4

Qwen 2 [Yang+ 2024]

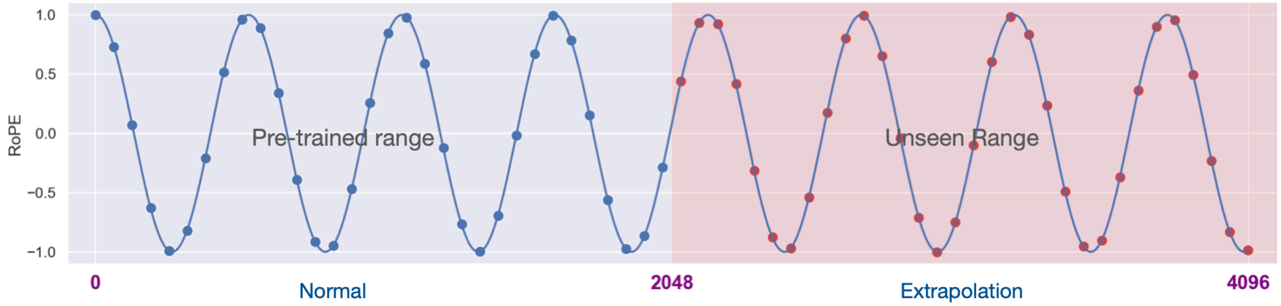
- Standard recipe: RMSNorm, Pre-norm, RoPE, SwiGLU, GQA
- MoE: Qwen2-57B-A14B (64 experts, 8 active, upcycled from Qwen-7B)
  - 64 experts + shared experts, top-8 routing

- Training: 7T tokens (dense), 4.5T (MoE), 12T (0.5B)

- Long context support
  - QKV bias; adds bias to Q, K, V projections [Su+ 2023]
  - Yet another RoPE extensioN (YaRN) [Peng+ 2023]
  - Dual chunk attention (DCA) [An+ 2024]

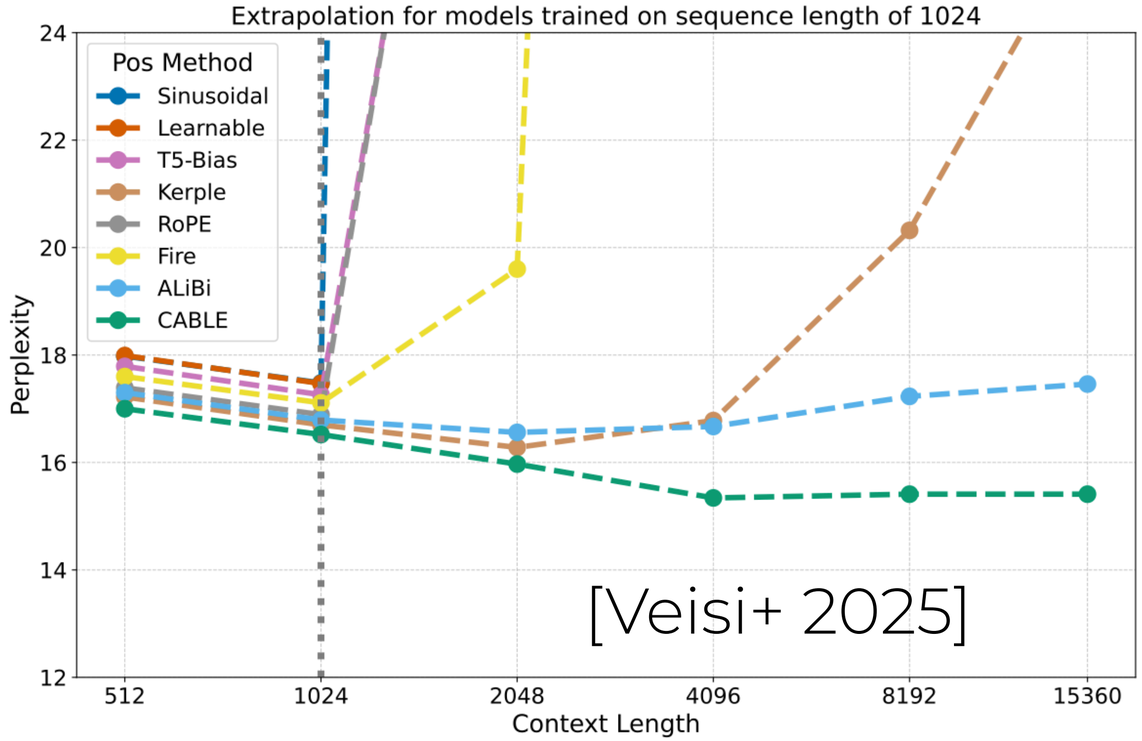
Challenge: Long Context Support

- What happens if the model gets a seq. longer than its training length?
  - Out-of-distribution of sequence length
  - aka length extrapolation (generalization) problem



[Chen+ 2023]

e.g. unseen θ



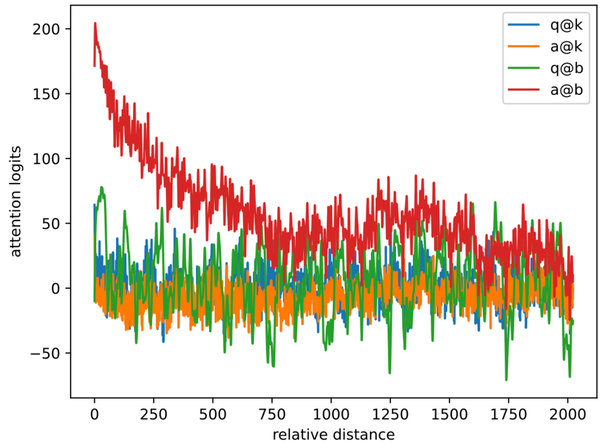
[Veisi+ 2025]

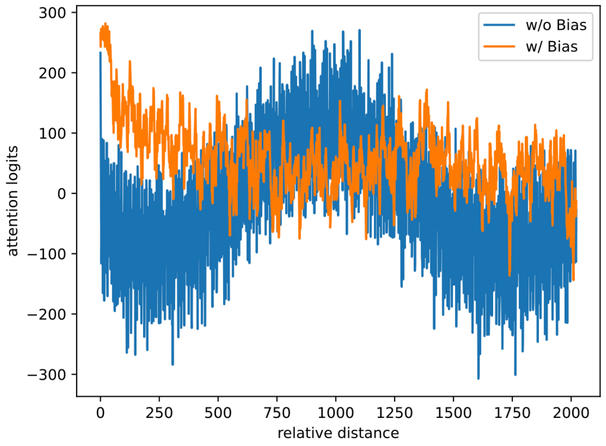
QKV Bias

$$O_i = Softmax\left(\frac{\overline{RoPE(XW_Q^i)\ RoPE(XW_K^i)^\top}}{\sqrt{d_k}} + M\right)V_i$$

- First, let's rewrite dot product of QK (m, n: positional index)
  - $(q_m R_m)(k_n R_n)^\top = q_m R_m R_n^\top k_n^\top = q_m \underline{R_{m-n}} k_n^\top$
  - RoPE represents relative distance $(m-n)$ between Q and K

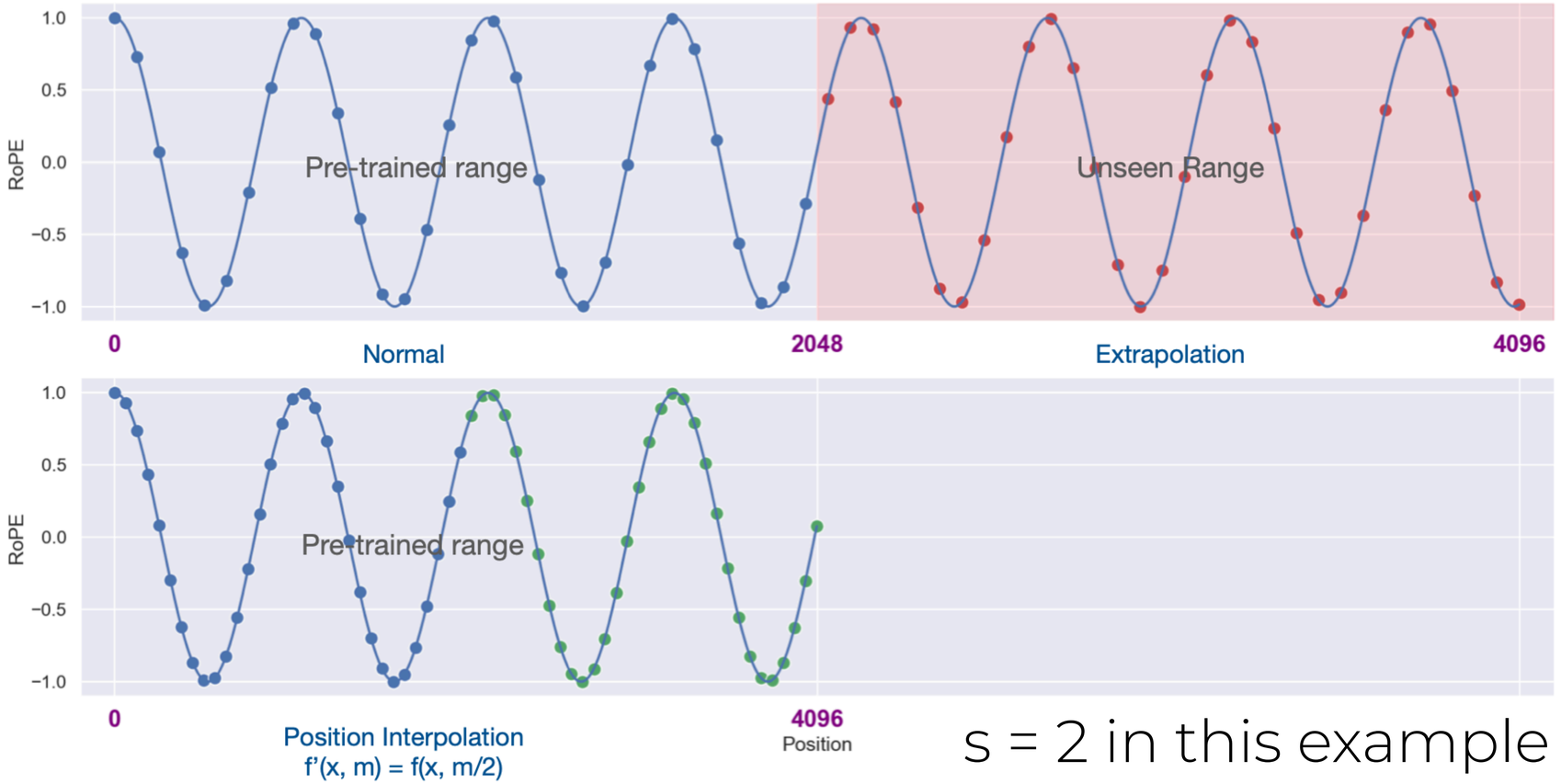
- With QK bias: $(q_m + a)\, R_{m-n}\, (k_n + b)^\top = q_m R_{m-n} k_n^\top + a R_{m-n} k_n^\top + q_m R_{m-n} b^\top + a R_{m-n} b^\top$
  - Term 1: original RoPE score without bias
  - Term 2-3: expectation is nearly zero (no effects)
  - Term 4: acts as a distance penalty (decays attention logits for long-distance tokens, reducing OOD spikes)





Position Interpolation [Chen+ 2023]

- Squeeze RoPE range!



s = 2 in this example

RoPE: $\theta_i = b^{-2i/d}$

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 \\ \sin m\theta_1 & \cos m\theta_1 \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

+ PI: $cos\left(\frac{m}{s}\theta_i\right)$

$sin\left(\frac{m}{s}\theta_i\right)$

"Base" in RoPE

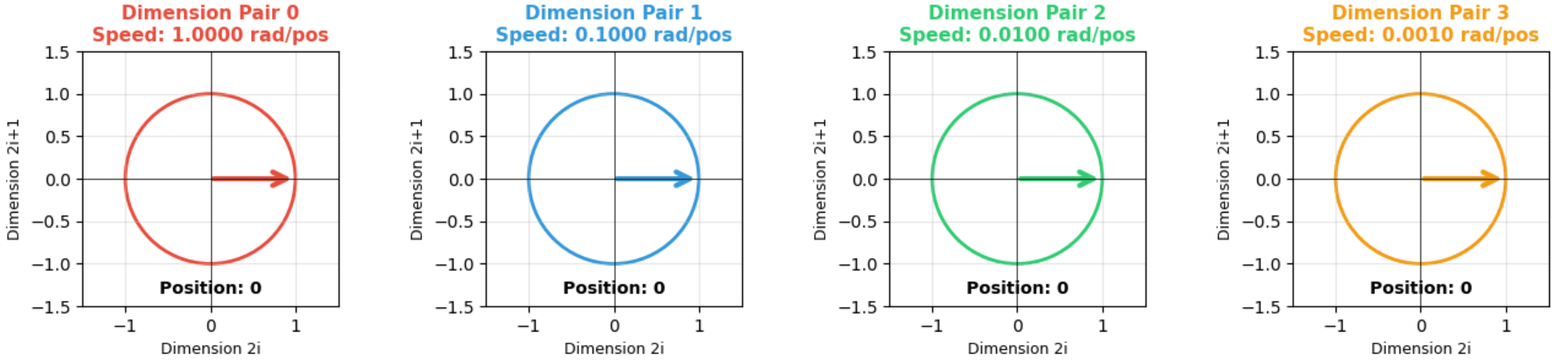
- Recall: Base frequency of RoPE is $\theta_i = b^{-2i/d}$
  - "Base" b determines rotation speed of RoPE as index m changes
  - In addition, it is linked to RoPE's wavelength, and it determines the maximum context length of the model!

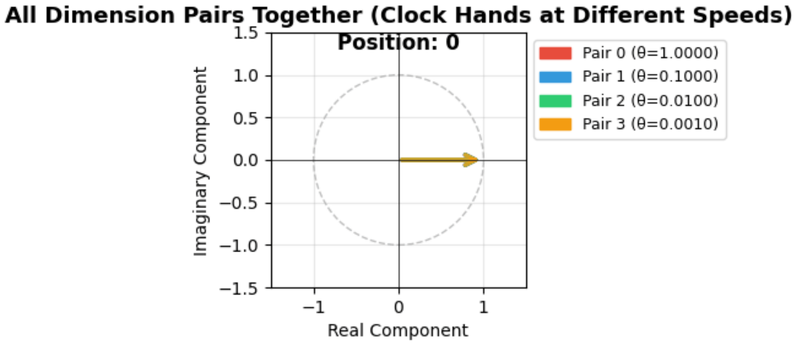
- Wavelength: # tokens (distance) for a full rotation (2π) $\lambda_i = \dfrac{2\pi}{\theta_i} = 2\pi b^{2i/d}$
  - High-freq $(i \approx 0)$: short-wavelength
    - Captures subtle changes between adjacent tokens
  - Low-freq. $(i \approx d/2)$: long-wavelength
    - Captures global information by seeing very long distance tokens

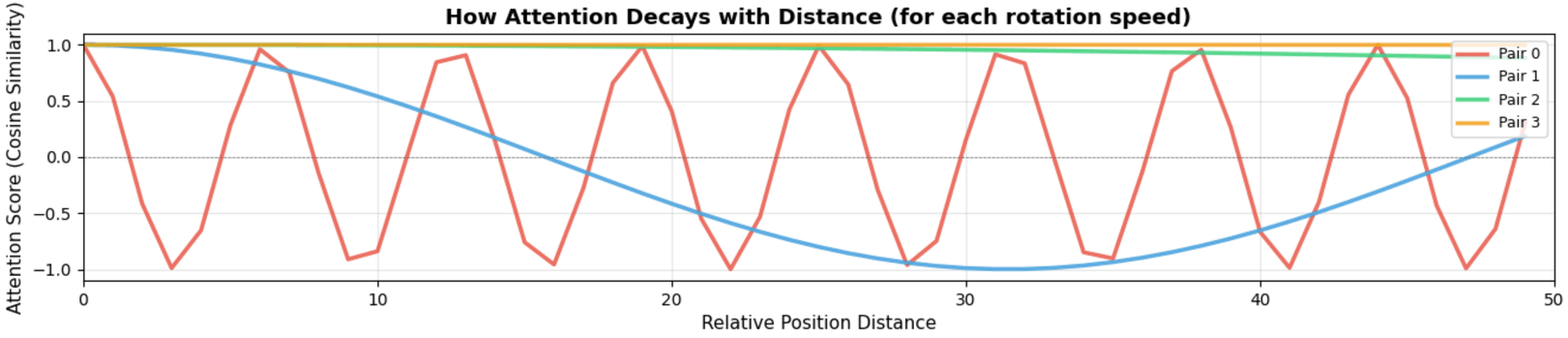
"Ba

- Rec
  - "E
  - Ir
    th

- Wav
  - H
    -
  - L
    -







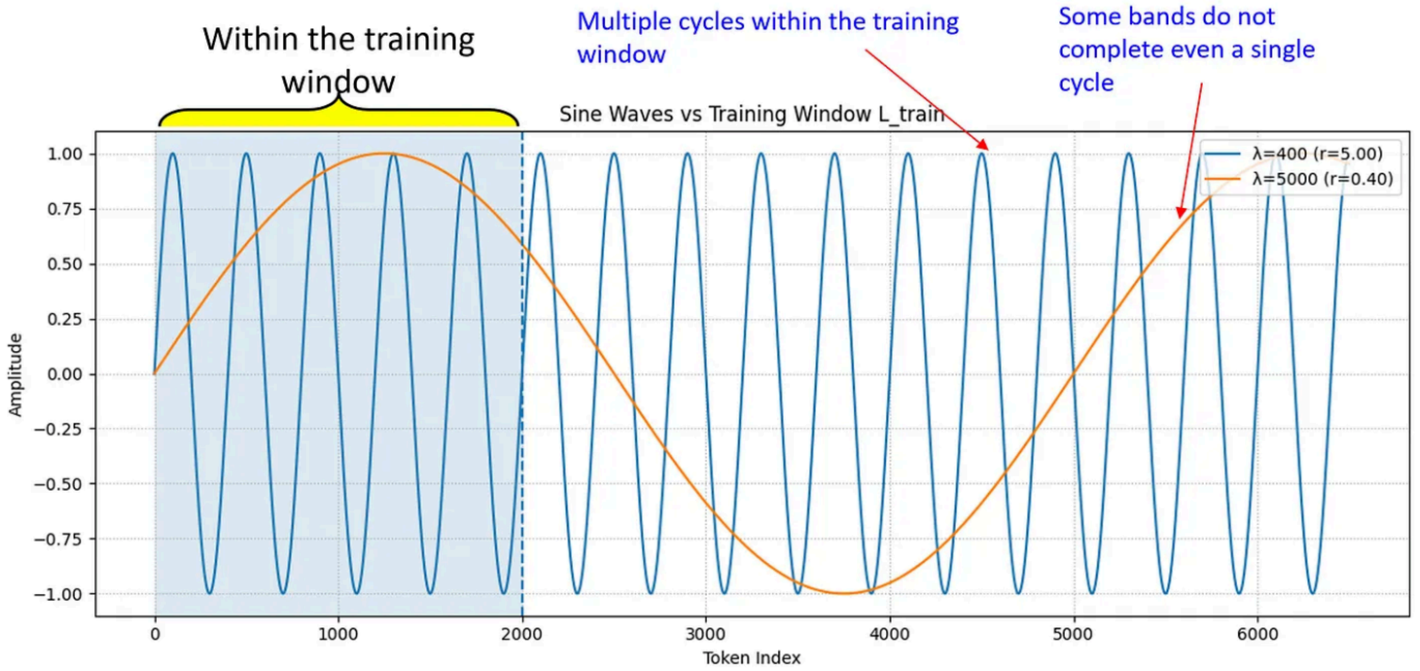
$\gamma^{2i/d}$

ens

Image credit: https://medium.com/@saeed.mehrang/understanding-rotary-position-embeddings-rope-a-visual-guide-ef8319353ddb

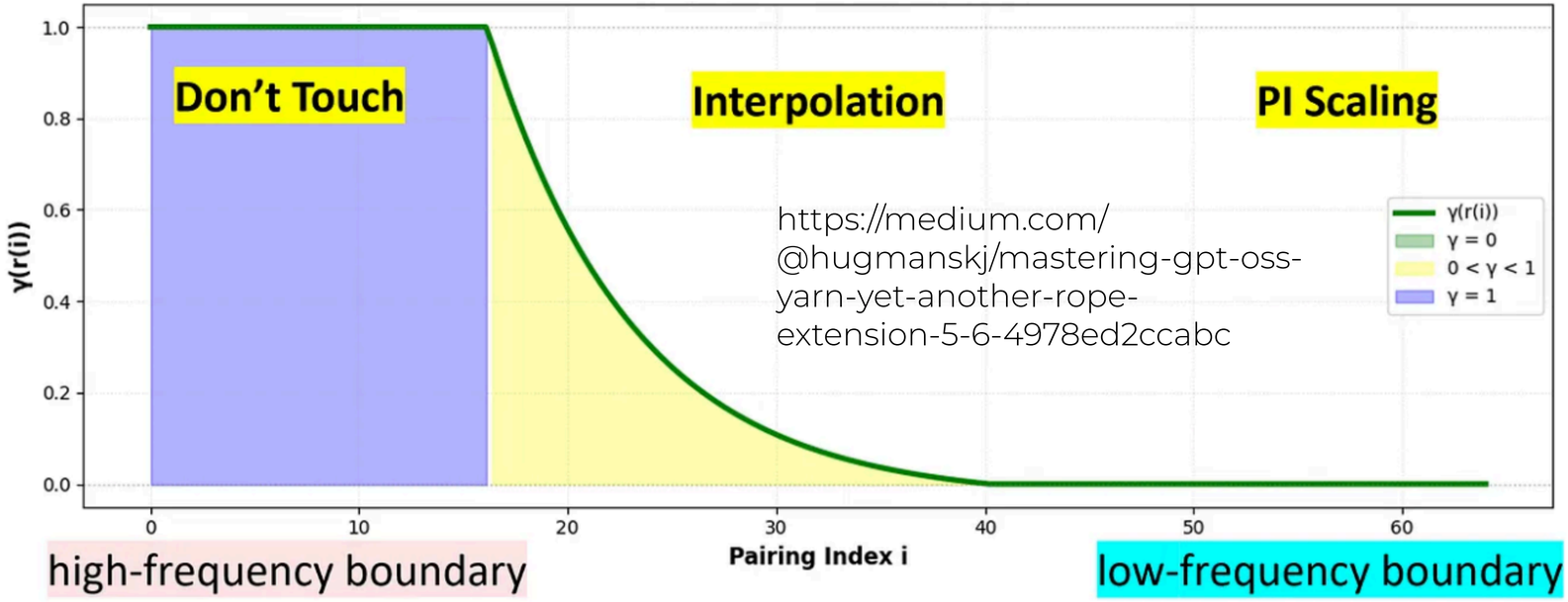
Yet Another RoPE ExtensioN (YaRN) [Peng+ 2023]

- Position interpolation treats all dimensions equally
  - This over-compresses rotation angles in high-freq. dimensions
  - Hence, local context between nearer tokens is degraded
- Actually, we don't need to squeeze the high-freq. dimensions



Yet Another RoPE ExtensioN (YaRN) [Peng+ 2023]

- NTK-based PI and YaRN: split and do band-wise scaling



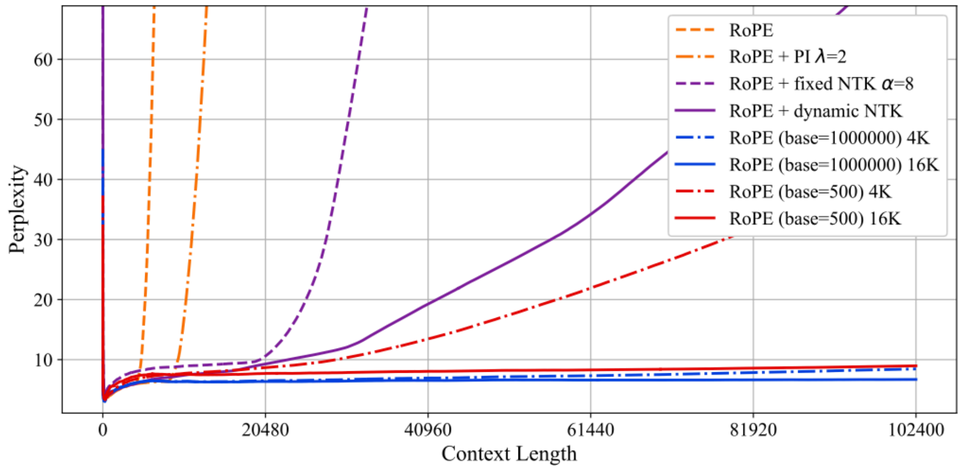
- + temperature on attention softmax

$$\text{softmax}\left(\frac{\boldsymbol{q}_m^T \boldsymbol{k}_n}{t\sqrt{|D|}}\right) \quad \sqrt{\frac{1}{t}} = 0.1\ln(s) + 1. \quad \text{s = context extension ratio}$$

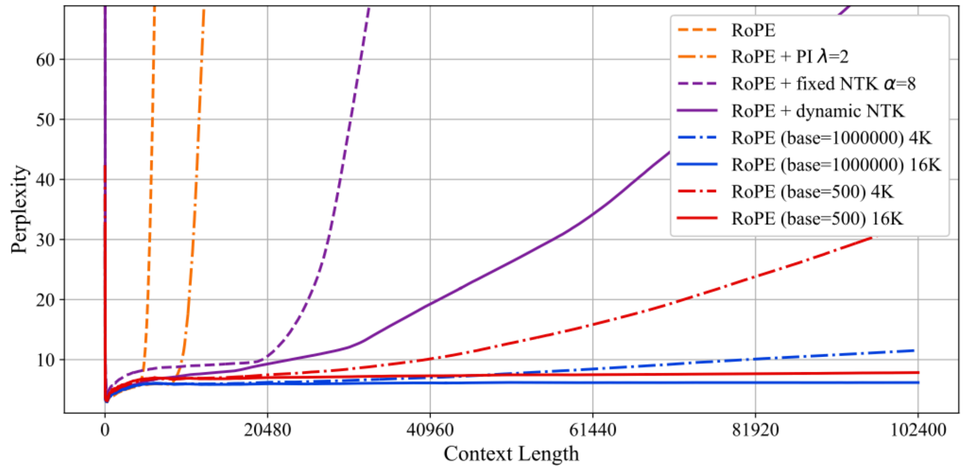
Scaling Laws of RoPE [Liu+ 2023]

- Early models use base b = 10,000 (LLaMA) $\theta_i = b^{-2i/d}$
- But modern models use a very large base e.g. 1,000,000 for Qwen 2

- "Scaling Laws of RoPE-based Extrapolation" [Liu+ 2023]
  - Previous base (10,000) is bad; smaller or larger bases are good



(a) Results on LLaMA2 7B



(b) Results on LLaMA2 13B

Scaling Laws of RoPE [Liu+ 2023]

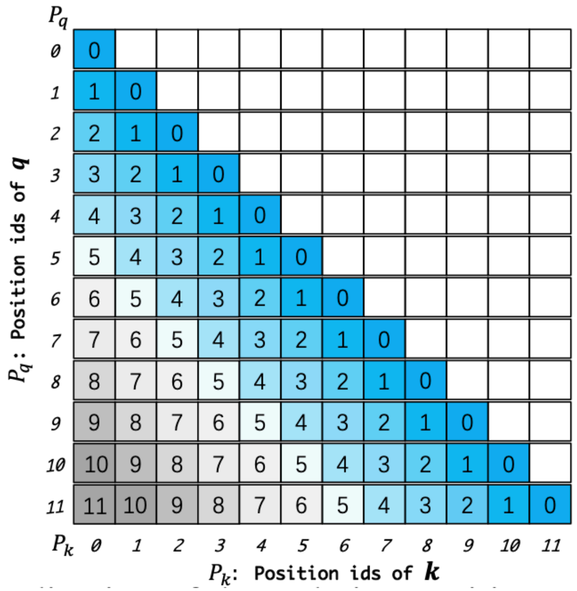
- Smaller base (e.g. 500) → shorter wavelength
  - High dims fully rotate within short seq. length (4K) when training
    - No OOD rotation angle when seq. length is large
  - → Unbounded extrapolation: graceful degradation in PPL without catastrophic failure

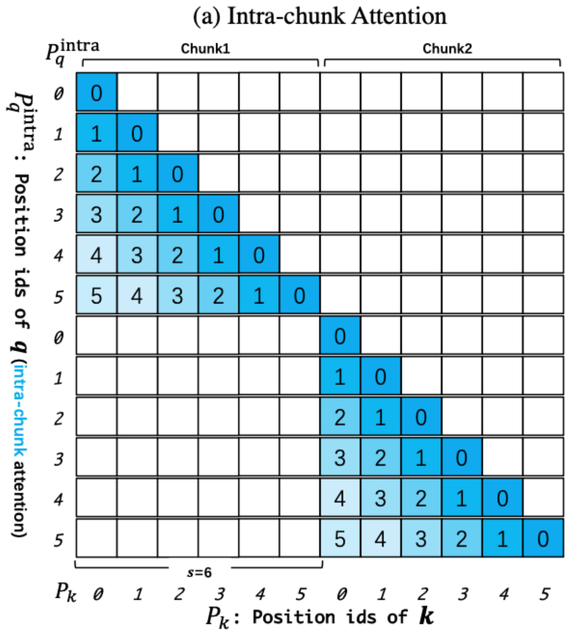
- Larger base (e.g. 1,000,000) → longer wavelength
  - Minimal angle shifts between tokens → Very high-res phase space
  - Extended seq. remain within the seen range (interpolation)
  - → Bounded but high-performance
    - But with YARN or other methods, we can further extend bound

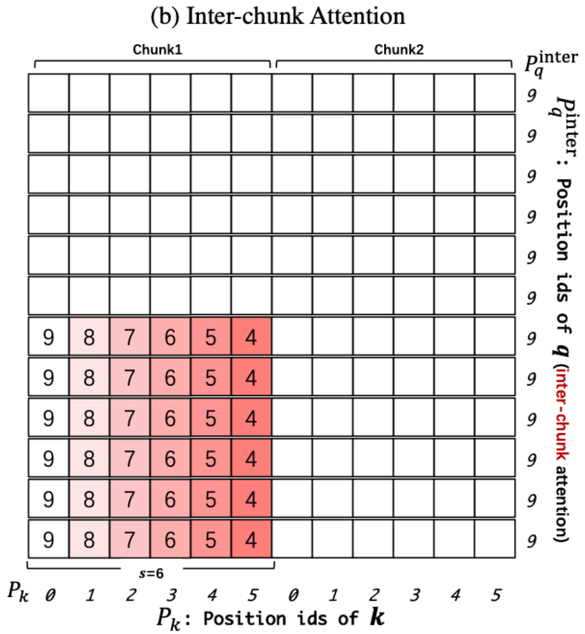
Dual Chunk Attention (DCA) [An+ 2024]

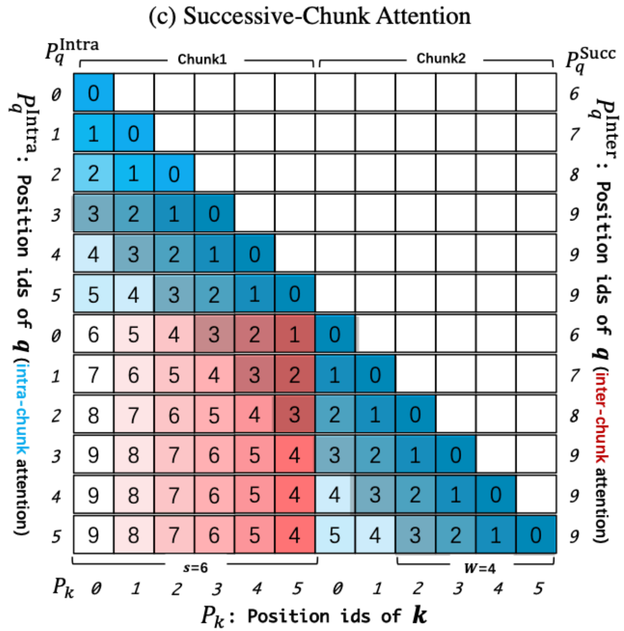
- Dual chunk attention splits sequence into multiple chunks
  - Chunk size must be smaller than the pre-trained context length
  - Intra-chunk attention
    - Captures local dependencies within each individual chunk
  - Inter-chunk attention
    - Captures long-range dependencies between different chunks
  - Successive-chunk attention
    - Rectifies the discontinuity of relative positional information at chunk boundaries

Dual Chunk Attention (DCA) [An+ 2024]









context window (c) = 10
chunk size (s) = 6

qk distance within the chunk

[c-1, c-s] distance

Alleviates the sudden index spike between neighboring tokens

combining these:

$$M[i][j] = \begin{cases} P_{\mathbf{q}}^{\text{Intra}}[i] - P_{\mathbf{k}}[j] & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 0 \\ P_{\mathbf{q}}^{\text{Succ}}[i] - P_{\mathbf{k}}[j] & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 1 \\ P_{\mathbf{q}}^{\text{Inter}}[i] - P_{\mathbf{k}}[j] & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor > 1. \end{cases}$$

relative distance mask

Why DCA Works? [An+ 2024]

- It rearranges positional indices of the seq
  - with intra-chunk, inter-chunk, successive-chunk

$$M[i][j] = \begin{cases} P_{\mathbf{q}}^{\text{Intra}}[i] - P_{\mathbf{k}}[j] & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 0 \\ P_{\mathbf{q}}^{\text{Succ}}[i] - P_{\mathbf{k}}[j] & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor = 1 \\ P_{\mathbf{q}}^{\text{Inter}}[i] - P_{\mathbf{k}}[j] & \text{if } \lfloor i/s \rfloor - \lfloor j/s \rfloor > 1. \end{cases}$$

- As a result, the relative distances are restricted within the range that RoPE has seen during pre-training
  - And it's training-free!

Model	Evaluation Context Window 4096	8192	16384	32768	65536
Finetuned					
Longlora-32k 7B	**8.14**	**7.85**	**7.70**	7.80	91.79
Together-32k 7B	8.21	7.95	7.76	**7.64**	$>10^2$
CodeLlama-16k 7B	8.93	8.64	8.44	8.36	**8.65**
CLEX-16k 7B	8.84	7.66	7.43	7.57	8.73
Training-free					
Llama2 7B	**7.87**	$>10^2$	$>10^2$	$>10^2$	$>10^2$
Llama2-ReRoPE 7B	7.94	7.75	OOM	OOM	OOM
Llama2-PI 7B	7.87	9.19	15.11	$>10^2$	$>10^2$
Llama2-PI-Yarn 7B	7.87	8.80	11.75	42.42	$>10^2$
Llama2-NTK 7B	7.87	11.98	26.12	58.91	$>10^2$
Llama2-NTK-Yarn 7B	7.87	8.06	9.82	11.74	41.57
CHUNKLLAMA2 7B	7.87	7.67	7.64	7.89	15.87
CHUNKLLAMA2 13B	7.15	6.95	6.99	7.90	15.14
CHUNKLLAMA2 70B	**5.24**	**5.18**	**5.21**	**5.30**	**5.59**
Llama3					
Llama3 8B	9.04	8.71	78.88	$>10^2$	$>10^2$
Llama3 70B	5.36	5.16	$>10^2$	$>10^2$	$>10^2$
CHUNKLLAMA3 8B	9.04	8.71	8.61	8.62	8.95
CHUNKLLAMA3 70B	**5.36**	**5.16**	**5.14**	**5.14**	**5.21**

ChunkLLaMA = DCA + LLaMA

Qwen 2: Hyperparameters [Yang+ 2024]

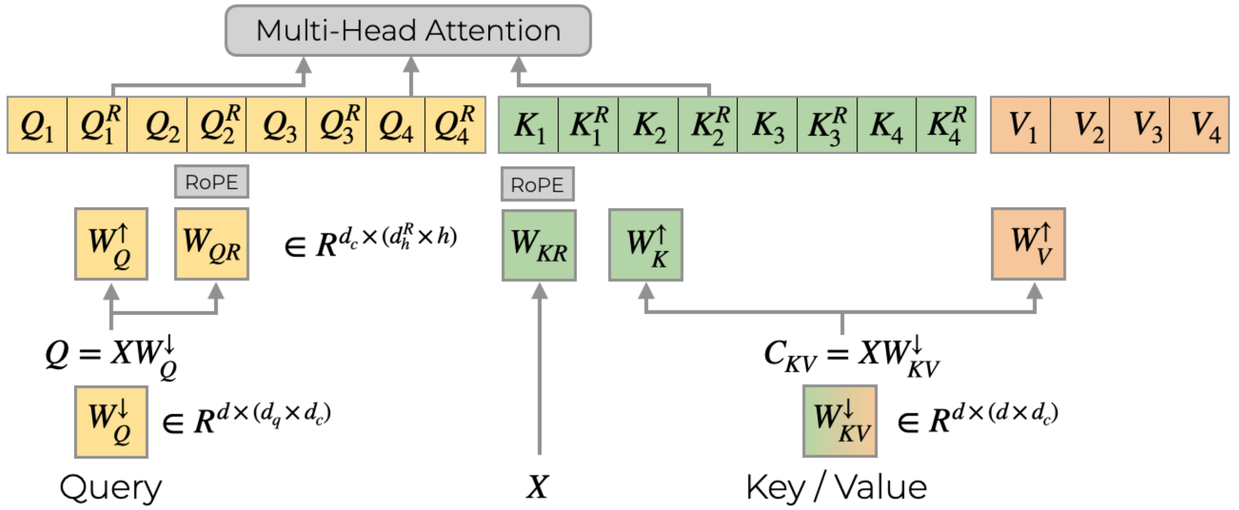
Model	# Params	Layers	d_model	Q Heads	KV Heads	d_ff	Context	Vocab
0.5B	0.5B	24	896	14	2	4864	32K	151,936
1.5B	1.5B	28	1536	12	2	8960	32K	151,936
7B	7B	28	3584	28	4	18944	32K	152,064
72B	72B	80	8192	64	8	29568	32K	152,064
57B-A14B	57B(14B)	28	3584	28	4	2560/expert	32K	152,064

Long Context Support: Takeaway

- Long context support: A major challenge, yet essential
  - Computation: KV cache memory, complexity of full attention
    - Solutions: GQA, MLA, interleaved local-global attentions
  - Positional encoding: RoPE struggles with unseen long contexts
    - Training with long context? Highly resource-intensive
    - Solutions: RoPE variants (PI, YaRN, base frequency scaling, …)

DeepSeek-V2 [DeepSeek-AI 2024]

- Multi-head latent attention (MLA)
  - With decoupled RoPE
- Fine-grained MoE:
  - 160 experts, 2 shared experts,
  - 6 / 2 activated experts
- Training: 8.1T tokens
- YaRN for long context (4K → 128K)



Attention Mechanism	KV Cache per Token (# Element)	Capability
Multi-Head Attention (MHA)	$2n_h d_h l$	Strong
Grouped-Query Attention (GQA)	$2n_g d_h l$	Moderate
Multi-Query Attention (MQA)	$2d_h l$	Weak
MLA (Ours)	$(d_c + d_h^R)l \approx \frac{9}{2} d_h l$	Stronger

Routing and Load Balancing

- For each token, select M GPUs that have experts with high score
- Then, perform top-K selection among experts on these M devices

- Communication balance loss
  - Makes the communication of each device balanced

$$\mathcal{L}_{\text{CommBal}} = \alpha_3 \sum_{i=1}^{D} f_i'' P_i'',$$

$$f_i'' = \frac{D}{MT} \sum_{t=1}^{T} \mathbb{1}(\text{Token } t \text{ is sent to Device } i),$$

$$P_i'' = \sum_{j \in \mathcal{E}_i} P_j,$$

$$\mathcal{L}_{\text{DevBal}} = \alpha_2 \sum_{i=1}^{D} f_i' P_i',$$

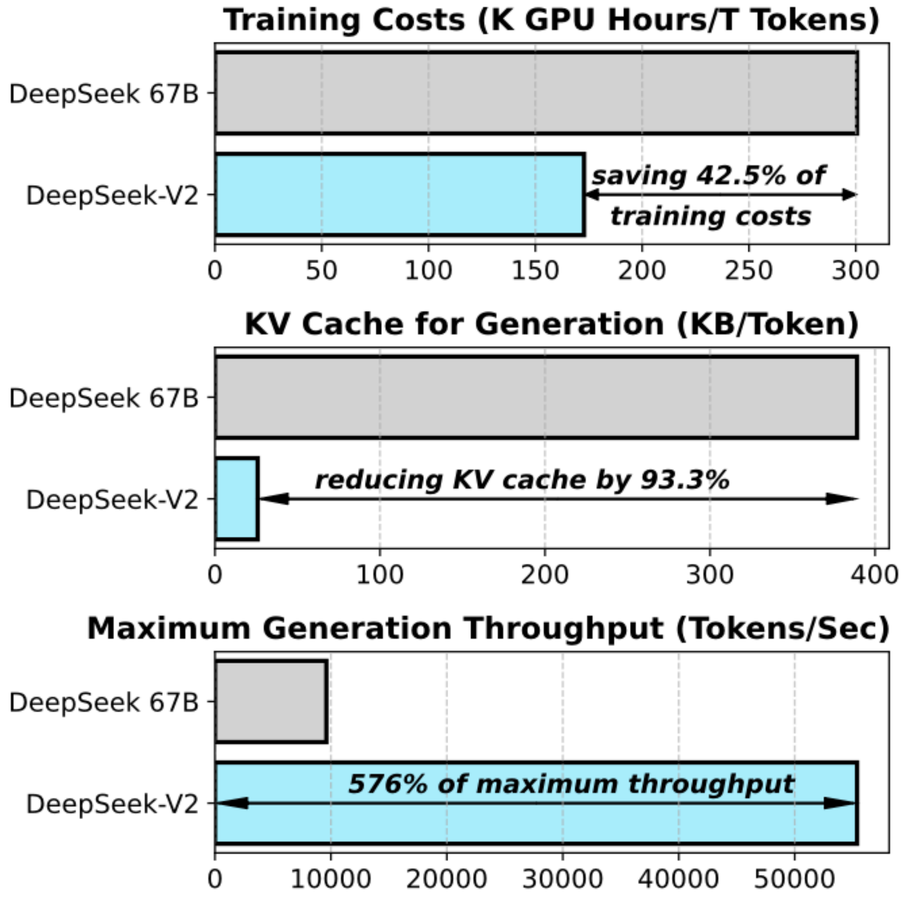
$$f_i' = \frac{1}{|\mathcal{E}_i|} \sum_{j \in \mathcal{E}_i} f_j,$$

$$P_i' = \sum_{j \in \mathcal{E}_i} P_j,$$

- + Expert-level, device-level balance losses

DeepSeek-V2: Hyperparameters [DeepSeek-AI 2024]

	DeepSeek-V2
Total Params	236B
Active Params	21B
Layers	60
d_model	5120
Attention Heads	128
KV Comp. Dim (d_c)	512
Query Comp. Dim	1536
Decoupled RoPE Dim	64
Total Experts	2 shared + 160 routed
Active Experts	2 shared + 6 routed
Tokens	8.1T
Context	4K → 128K



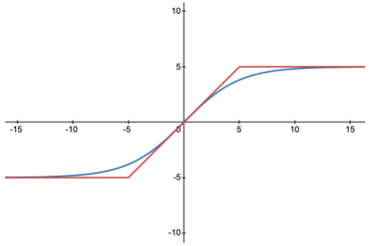
Gemma 2 [Gemma Team 2024]

- Key components:
  - Knowledge distillation in pre-training
  - Local/global attention interleaving
  - Logit soft-capping for attention stability
- Other ingredients:
  - GQA
  - Pre + post norm: RMSNorm on both input/output

- Soft-capping the logits to some maximum value via Tanh
  - Softcap:

$$\text{softcap}(x) = t * \tanh(\frac{x}{t})$$

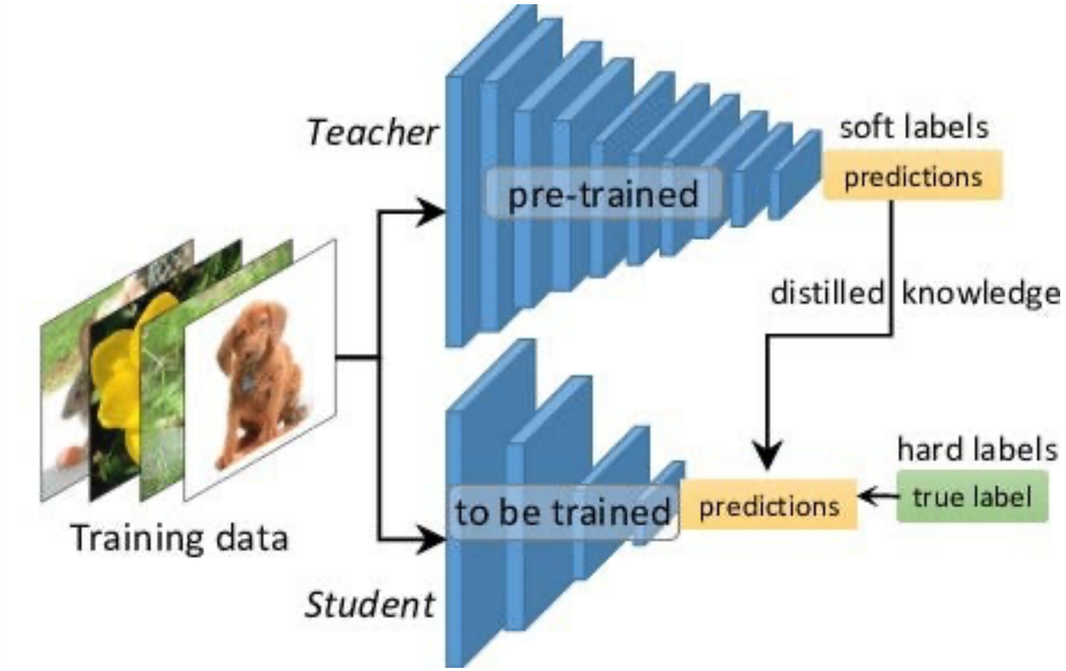
  - Hardcap: clip x in [-t, t]
- Gemma 2 uses t = 30.0 for output softmax, and t = 50.0 for attention softmax



Gemma 2: Hyperparameters [Gemma Team 2024]

Model	Params	Layers	d_model	Heads	KV Heads	Context	Tokens
2B	~2.6B	26	2304	8	4	8K	2T
9B	~9.2B	42	3584	16	8	8K	8T
27B	~27.2B	46	4608	32	16	8K	13T

Knowledge Distillation [Gemma Team 2024]

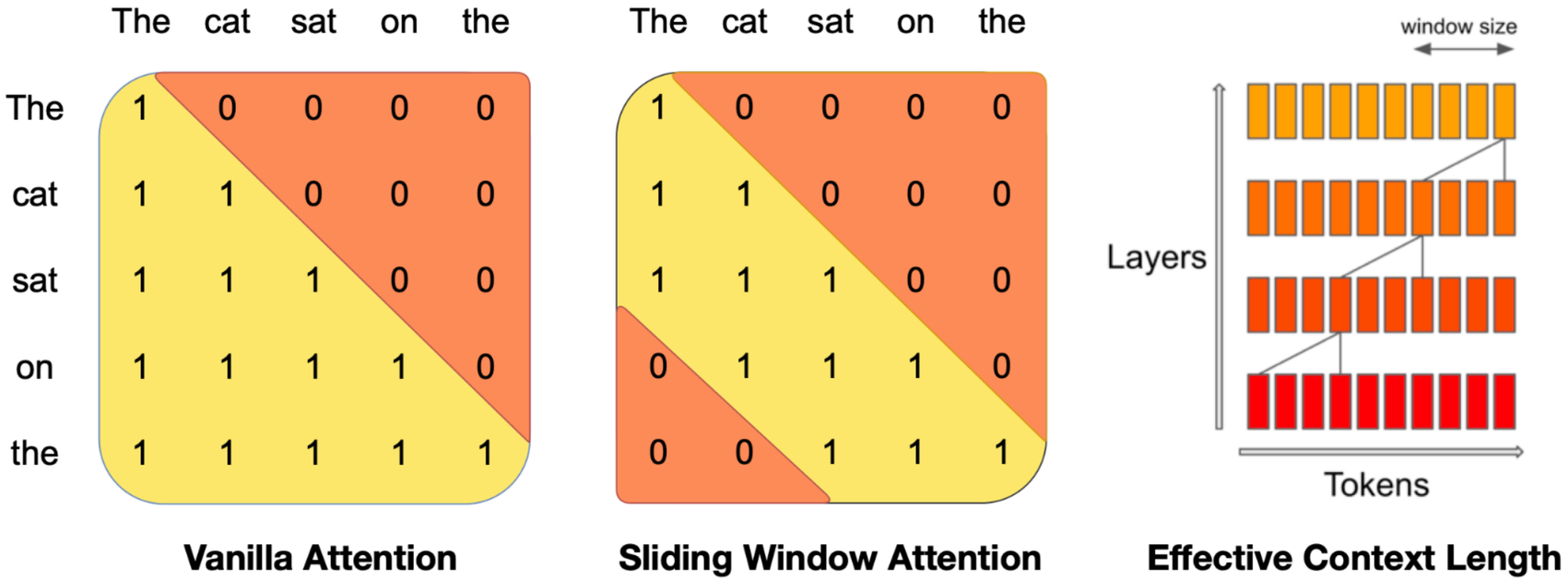


$$\min_{P_S} \sum_{x} -P_T(x \mid x_c) \log P_S(x \mid x_c),$$

where P_S is the parameterized probability of the student. Note that knowledge distillation was also used in Gemini 1.5 (Gemini Team, 2024).

Attention Interleaving [Gemma Team 2024]

- Use local sliding window attention and GQA-based global attention
- Attention interleaving
  - Even layers: SWA (4096 tokens) / odd layers: GQA (8192 tokens)



Qwen 2.5 [Qwen Team 2024]

- Architecture: same as Qwen2 (GQA, SwiGLU, RoPE, QKV bias)
  - DCA+YARN, DeepSeekMoE-style MoE
- Key change: massive data scaling, 7T → 18T tokens

- Control tokens: 3 → 22 (expanded for tool-use)
- Scaling law used to predict optimal HPs (MiniCPM influence)

Models	Layers	Heads (Q / KV)	Tie Embedding	Context / Generation Length	License
0.5B	24	14 / 2	Yes	32K / 8K	Apache 2.0
1.5B	28	12 / 2	Yes	32K / 8K	Apache 2.0
3B	36	16 / 2	Yes	32K / 8K	Qwen Research
7B	28	28 / 4	No	128K / 8K	Apache 2.0
14B	48	40 / 8	No	128K / 8K	Apache 2.0
32B	64	40 / 8	No	128K / 8K	Apache 2.0
72B	80	64 / 8	No	128K / 8K	Qwen

DeepSeek-V3 [DeepSeek-AI 2024]

- Key components:
  - Base: DeepSeekV2 (MLA + MoE)
  - Auxiliary-loss-free load balancing
  - Multi-token prediction (MTP)
  - FP8 mixed precision training

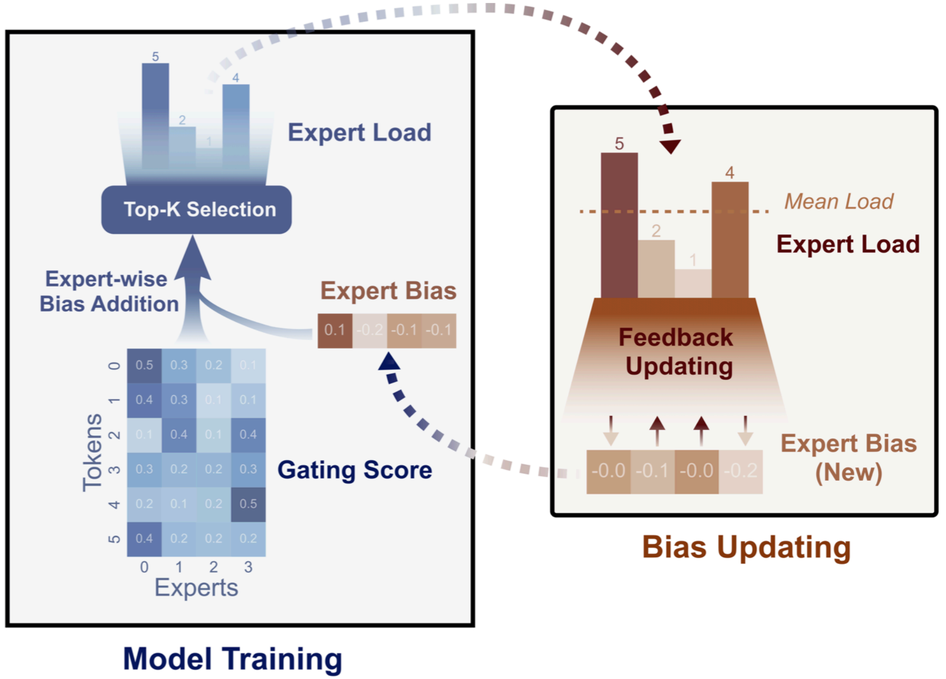
	DeepSeek-V3
Total Params	671B
Active Params	37B
Layers	61
d_model	7168
Attention Heads	128
KV Compression (d_c)	512
RoPE Dim (d^R_h)	64
Total Experts	1 shared + 256 routed
Active Experts	1 shared + 8 routed
Expert d_ff	2048
MTP Depth	1
Tokens	14.8T
Context	4K → 128K
Gating	Sigmoid

Auxiliary-Loss-Free LB [Wang+ 2024]

- Auxiliary LB losses are effective, but it can harm model performance
- Auxiliary-loss-free LB

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \text{Topk}(\{s_{j,t} + b_j | 1 \leqslant j \leqslant N_r\}, K_r), \\ 0, & \text{otherwise.} \end{cases}$$

  - Heuristic bias update



$$\mathbf{h}'_t = \mathbf{u}_t + \sum_{i=1}^{N_s} \text{FFN}_i^{(s)}(\mathbf{u}_t) + \sum_{i=1}^{N_r} g_{i,t} \text{FFN}_i^{(r)}(\mathbf{u}_t),$$

$$g_{i,t} = \frac{g'_{i,t}}{\sum_{j=1}^{N_r} g'_{j,t}},$$

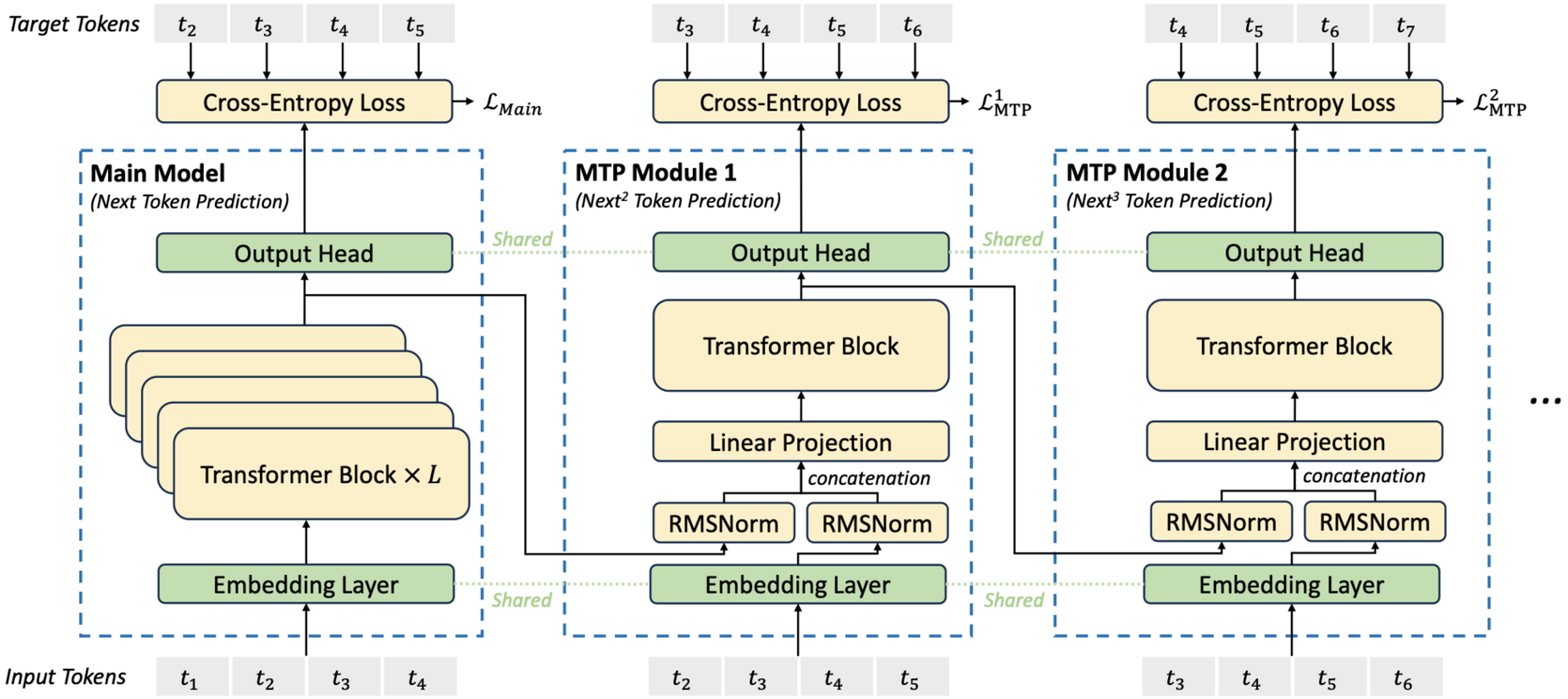
$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N_r\}, K_r), \\ 0, & \text{otherwise,} \end{cases}$$

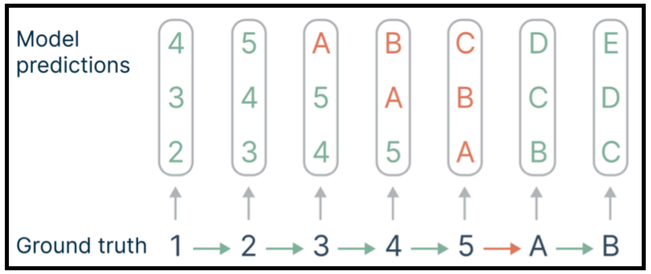
$$s_{i,t} = \text{Sigmoid}(\mathbf{u}_t^T \mathbf{e}_i),$$

Note: actually, DSv3 also uses seq.-wise aux loss to prevent extreme imbalance within a sequence

Multi-Token Prediction [Gloeckle+ 2024, DeepSeek-AI 2024]

- Extends the predict scope to multiple future tokens at each position
  - Better model performance + can be used for speculative decoding
  - Why it works? speculation: lookahead can reinforce choice points





[Gloeckle+ 2024]

DeepSeek-V3 [DeepSeek-AI 2024]

- 👍 Very stable training (no loss spikes, no rollback during training)

- 👍 DeepSeek-V3 achieves frontier performance at ~10x lower cost
  - e.g. LLaMA 3 405B used 3.8 × 10^25 FLOPs on 16K H100s (~30.8M H100-hours, estimated)

Stage	H800 hrs	Cost (@ $2/h)
Pre-training	2,664K	$5.328M
Context extension	119K	$0.238M
Post-training	5K	$0.01M
Total	2,788K	$5.576M

MiniMax-01 [MiniMax 2024]

- Key components:
  - Lightning attention: linear ($O(td^2)$) in sequence length
  - Post-norm with DeepNorm [Wang+ 2022]

```
def deepnorm(x):
    return LayerNorm(x * α + f(x))

def deepnorm_init(w):
    if w is ['ffn', 'v_proj', 'out_proj']:
        nn.init.xavier_normal_(w, gain=β)
    elif w is ['q_proj', 'k_proj']:
        nn.init.xavier_normal_(w, gain=1)
```

Architectures	Encoder		Decoder	
	α	β	α	β
Encoder-only (e.g., BERT)	$(2N)^{\frac{1}{4}}$	$(8N)^{-\frac{1}{4}}$	-	-
Decoder-only (e.g., GPT)	-	-	$(2M)^{\frac{1}{4}}$	$(8M)^{-\frac{1}{4}}$
Encoder-decoder (e.g., NMT, T5)	$0.81(N^4M)^{\frac{1}{16}}$	$0.87(N^4M)^{-\frac{1}{16}}$	$(3M)^{\frac{1}{4}}$	$(12M)^{-\frac{1}{4}}$

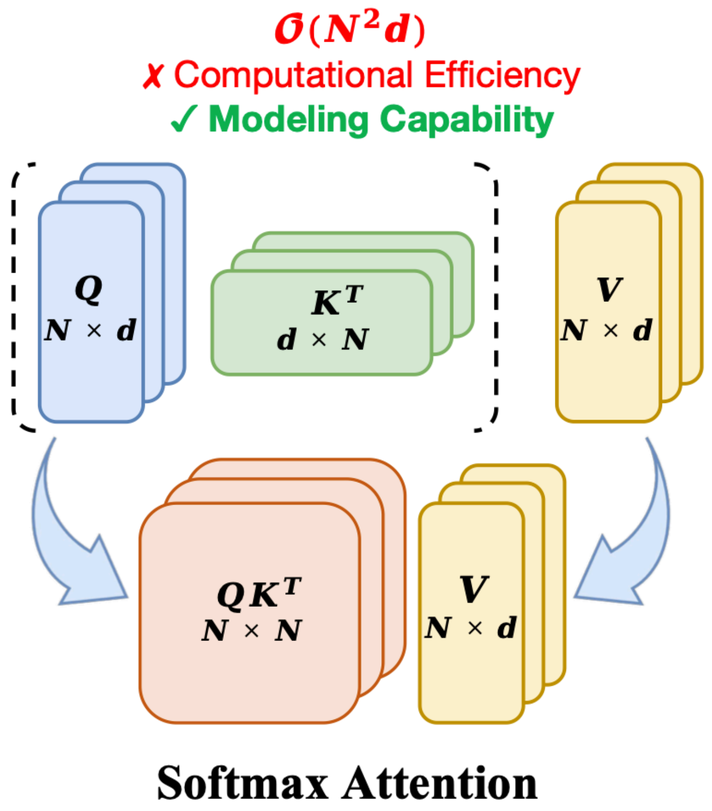
Figure 2: (a) Pseudocode for DEEPNORM. We take Xavier initialization (Glorot and Bengio, 2010) as an example, and it can be replaced with other standard initialization. Notice that α is a constant. (b) Parameters of DEEPNORM for different architectures (N-layer encoder, M-layer decoder).

- Recipes: GQA (on standard attention) + partial RoPE (50% of head dim)

Attention Complexity

- Context length: 2K (LLaMA; 2023) → 1M - 10M (2025)
  - We saw that recent models use interleaved SWA + full attention
  - But still attention itself is a very expensive op
  - Quadratic on the input sequence!
  - (and linear KV cache)

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2.d)$	$O(1)$	$O(1)$
Recurrent	$O(n.d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k.n.d^2)$	$O(1)$	$O(\log_k(n))$
Recurrent	$O(r.n.d)$	$O(1)$	$O(n/r)$

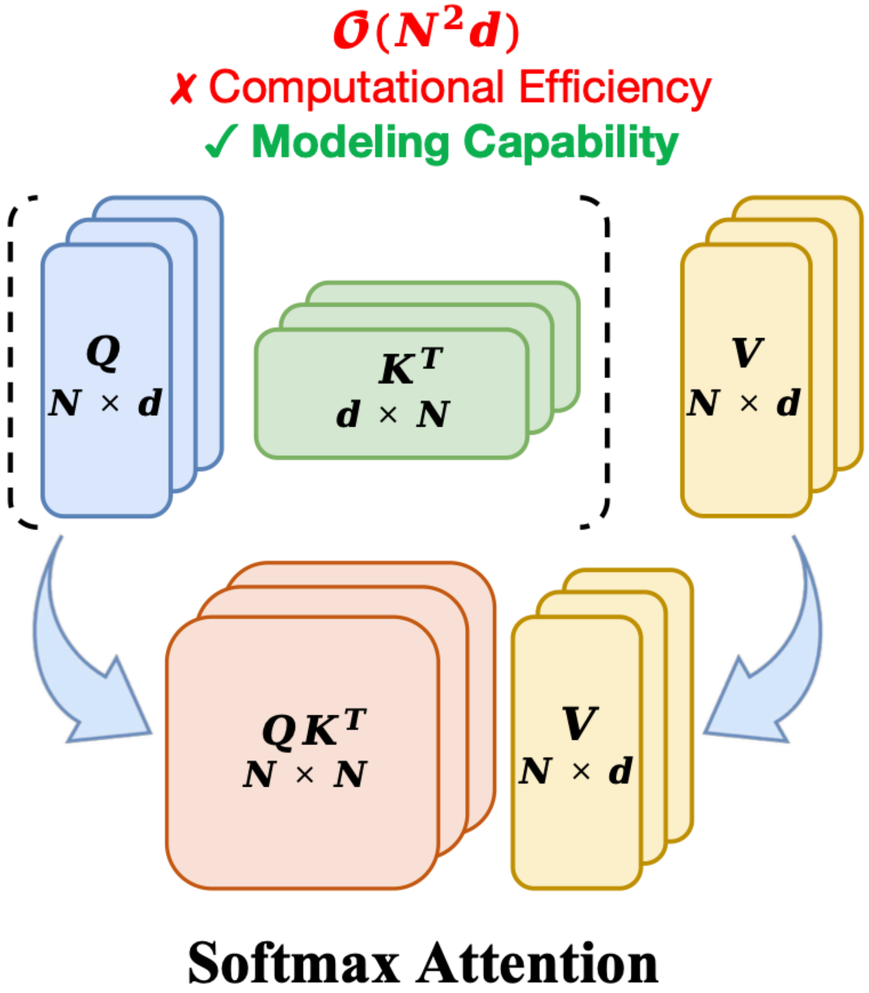


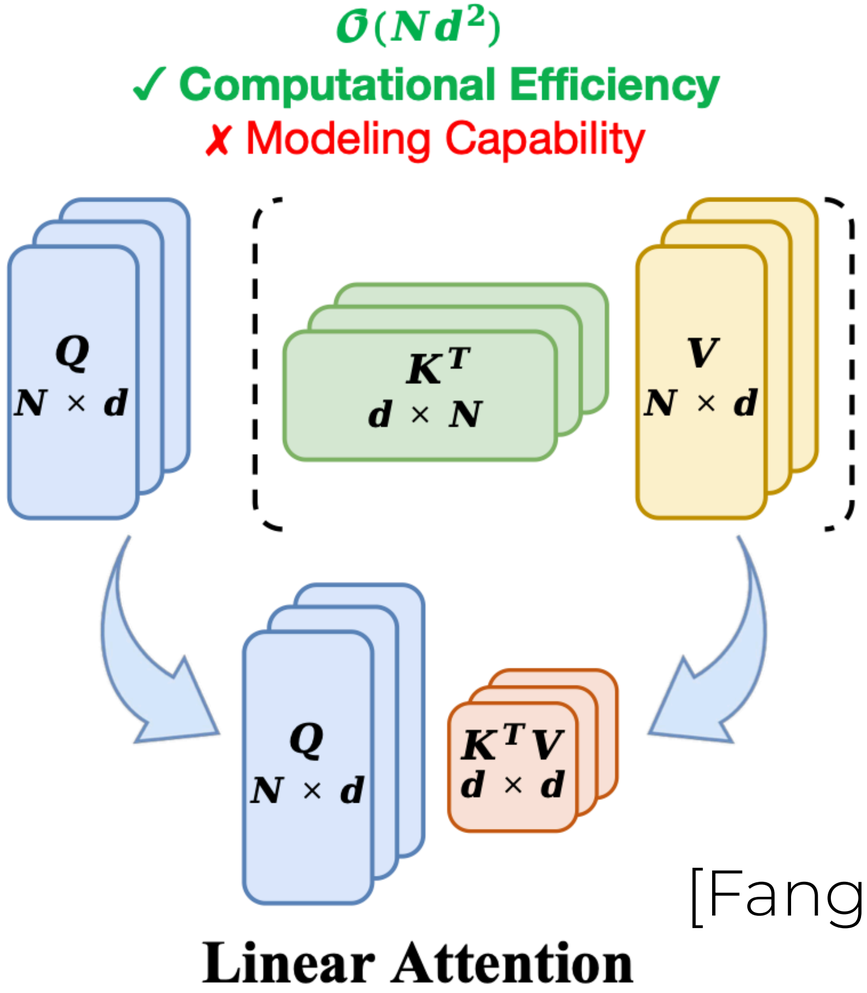
Linear Attention [Katharopoulos+ 2020]

- Can we modify attention to be linear complexity?
- Recall Softmax attention: $o_t = \sum_{i=1}^{t} \frac{\exp(q_t \cdot k_i^\top)}{\sum_{j=1}^{t} \exp(q_t \cdot k_j^\top)} v_i$ t = seq. length
- Linear attention: Softmax to kernel feature map $\exp(q_t \cdot k_i) \approx \phi(q_t)\phi(k_i)^\top$
  - Since there is no Softmax, we can replace as

$$o_t = \sum_{i=1}^{t} \phi(q_t) \cdot \phi(k_i)^\top v_i \quad = \phi(q_t) \sum_{i=1}^{t} \phi(k_i)^\top v_i$$

Linear Attention [Katharopoulos+ 2020]





[Fang+ 2025]

Linear Attention == RNN [Katharopoulos+ 2020]

- Define "state" as: $S = \sum_{i=1}^{t} k_i^\top v_i \quad o_t = q_t S.$ (omit kernel feature map)
- Due to the autoregressive (causal) characteristic, we can rewrite as:

$$S_t = S_{t-1} + k_t^\top v_t, \quad S_0 = \mathbf{0},$$
$$o_t = q_t S_t.$$

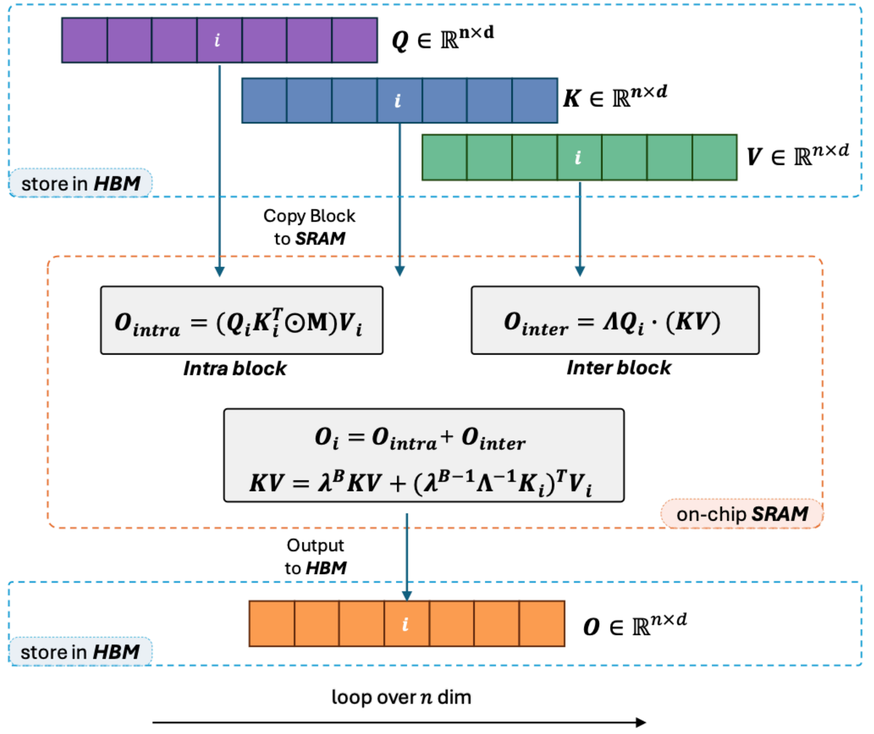
This is RNN!

	Softmax Attention	Linear Attention
Training	O(N^2)	O(N)
Inference per token	O(N)	O(1)
Inference state	KV cache: O(N)	Fixed state: O(1)
Strength	Good performance	Linear complexity
Weakness	Quadratic complexity	Memory overflow (RNN!)

Lightning Attention [Qin+ 2024]

- Linear attention is O(N), but not really fast in practice (in GPU)
  - Since state update is cumulative sum (and sequentially)
  - This makes massive SRAM ↔ HBM communication cost

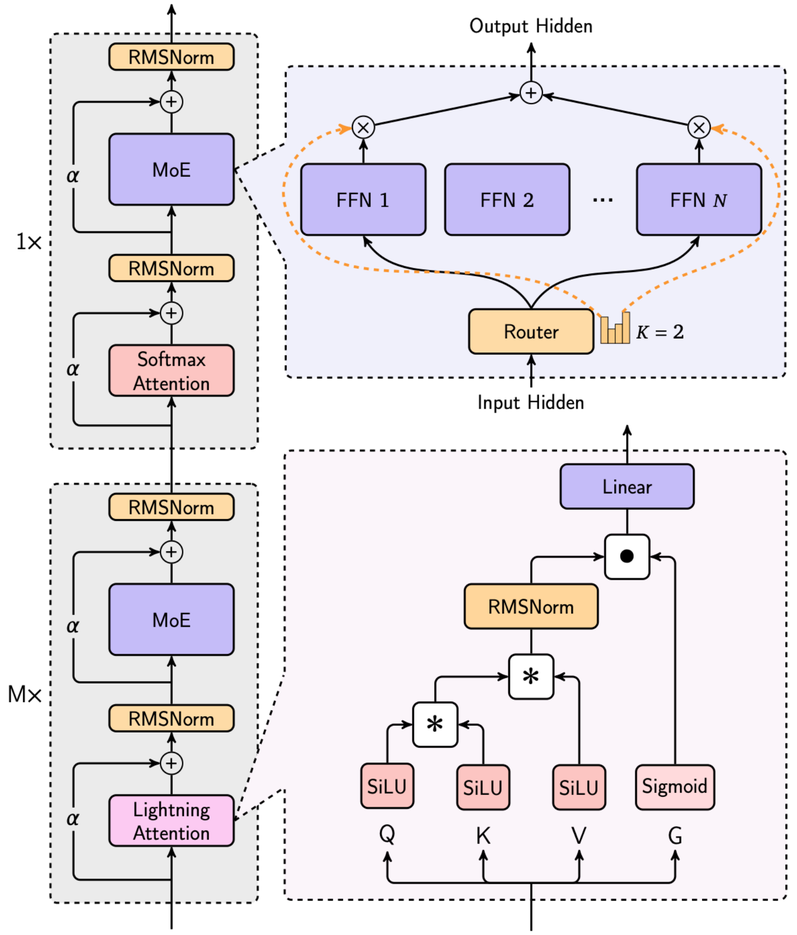
- Lightning attention: split seqs into multiple blocks (tiling)
  - Intra-block: Softmax attention (parallel)
  - Inter-block: linear attention (sequential)



Lightning Attention [MiniMax 2024]

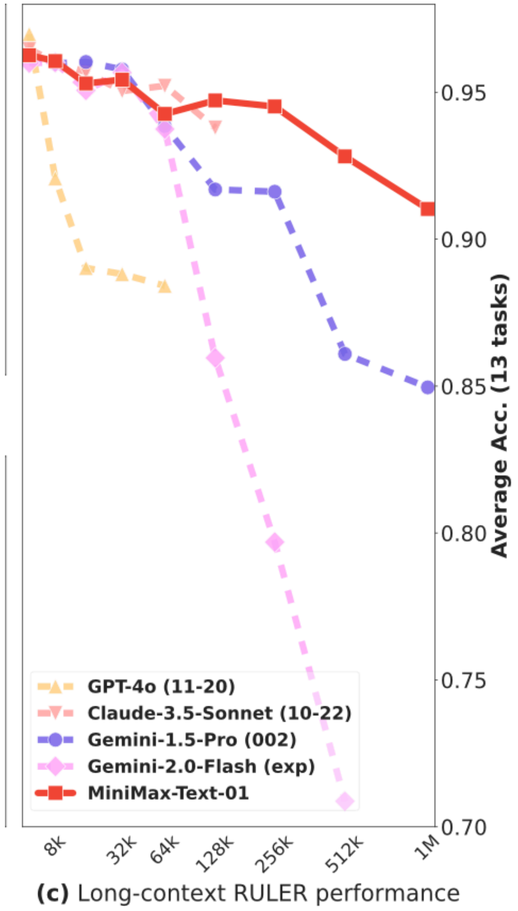
- Pure linear attention is weak at retrieval tasks

- Hybrid linear attention
  - 7:1 ratio
  - (7 linear / 1 Softmax)



MiniMax-01 [MiniMax 2024]

- Training 1M context is done with 3-stage context extension
  - 1) 128K: RoPE base 5M, 300B tokens
  - 2) 512K: RoPE base 10M, 32B tokens
  - 3) 1M: RoPE base 10M, 26B tokens
    - RoPE rotation angle: $\theta_i = \theta_{base}^{-\frac{2i}{d}}$
  - Total pre-training: ~11.8T tokens
  - Inference: extrapolates to 4M tokens



(c) Long-context RULER performance

Model Comparison: 2024 Q2 - 2024 Q4

Model	Params	Active	Attention	Experts	Tokens	Context	Cost
Qwen2-72B	72B	72B	GQA	Dense	7T	128K	N/A
DeepSeek-V2	236B	21B	MLA	2s+160r	8.1T	128K	N/A
Gemma 2-27B	27B	27B	GQA	Dense	13T	8K	N/A
Qwen2.5-72B	72B	72B	GQA	Dense	18T	128K	N/A
DeepSeek-V3	671B	37B	MLA	1s+256r	14.8T	128K	$5.576M
MiniMax-01	456B	45.9B	Lightning+Softmax	32	11.8T	1M	N/A

Key Takeaway: 2024 Q2 - 2024 Q4

- Attention evolution
  - Still many models use GQA, but MLA is rising
  - Local / global-level attention with interleaving
  - Linear-complexity attention (e.g. Lightning attention)

- New setups: fine-grained MoE, aux-loss-free LBL, …
  - Training: WSD, Multi-token prediction, mixed-precision (e.g. FP8)…

- Data scaling, long context
  - ~20T training tokens (Qwen2.5) vs. 2T in 2023 (LLaMA 2)
  - ~1M context length (MiniMax-01) vs. 32K in 2023 (Mixtral)

Key Takeaway: 2024 Q2 - 2024 Q4

Component	Early 2024	Mid 2024	Late 2024
Attention	GQA	+ MLA (DeepSeek-V2)	+ Lightning Attn. (MiniMax-01)
KV Cache	~25% (GQA-8)	~7% (MLA)	~7% (MLA)
MoE Routing	Softmax + aux loss	Softmax + aux loss	Sigmoid, aux-loss-free (DeepSeek V3)
Experts	64+2 (DeepSeekMoE)	160+2 (DeepSeek V2)	256+1 (DeepSeek V3)
Training	BF16, cosine/WSD	-	+ FP8, MTP (DeepSeek V3)
Data	7T (Qwen2)	13T (Gemma 2)	18T (Qwen2.5)
Distillation	-	KD in pre-training (Gemma 2)	-

Key Takeaway: 2024 Q2 - 2024 Q4

Metric	2023	2024
Max Params	70B (LLaMA 2)	671B (DeepSeek-V3)
Training Data	2T (LLaMA 2)	18T (Qwen2.5)
Context Length	32K (Mixtral)	1M (MiniMax-01)
Max Experts	8 (Mixtral)	256 (DeepSeek-V3)
KV Cache	MHA (100%)	MLA (~7%)

2023 vs. 2024

Part 3: 2025 -

OLMo 2 [Groeneveld+ 2025]

OLMo-0424	OLMo 2
$\boldsymbol{h} := \boldsymbol{x} + \mathrm{Attention}(\mathrm{LN}(\boldsymbol{x}))$	$\boldsymbol{h} := \boldsymbol{x} + \mathrm{RMSNorm}(\mathrm{Attention}(\boldsymbol{x}))$
$\boldsymbol{h}_{\mathrm{out}} := \boldsymbol{h} + \mathrm{MLP}(\mathrm{LN}(\boldsymbol{h}))$	$\boldsymbol{h}_{\mathrm{out}} := \boldsymbol{h} + \mathrm{RMSNorm}(\mathrm{MLP}(\boldsymbol{h}))$

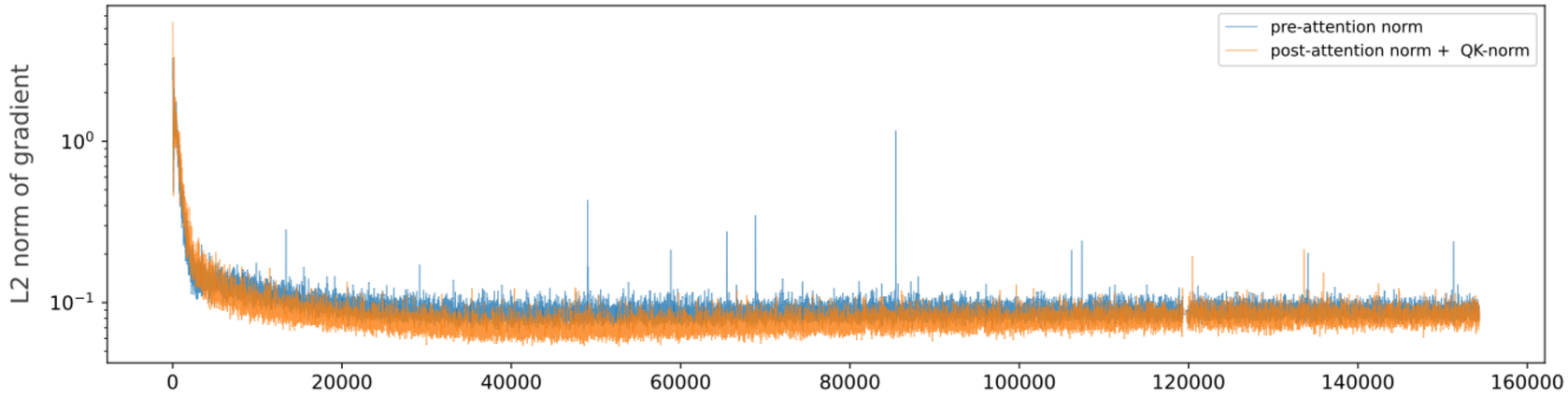
- OLMo 2 uses post-norm
  - SwinTF v2 [Liu+ 2021] introduced it
  - Pre-norm: 👍 stability / 👎 representation collapse in deep layers
  - Post-norm: 👍 deep layers 👎 vanishing grads in early layers
    - + QK-norm mitigates downside of post-norm

vanilla

$$\mathbf{softmax}(\frac{QK^T}{\sqrt{d}})V$$

qk-norm

$$\mathbf{softmax}(g * \hat{Q}\hat{K}^T)V$$



OLMo 2 [Groeneveld+ 2025]

Model	Params	Layers	Hidden	Heads	KV Heads	Context	Tokens
7B	7B	32	4096	32	32 (MHA)	4K	~4T
13B	13B	40	5120	40	40 (MHA)	4K	~5T

- Other recipes
  - Uses MHA (not GQA)
  - SwiGLU
  - z-loss

- Fully open: data (Dolma 1.7), code, 500+ checkpoints, training logs

Gemma 3 [Gemma Team 2025]

- Logit soft-capping → replaced by QK-Norm (simpler, more stable)
- Local-global ratio 1:1 → 5:1 interleaving (5 local layers per 1 global layer)
  - SWA window: 4096 → 1024 tokens
  - KV overhead: 60% → 15% (at 32K context)
- RoPE: base frequency 10K → 1M (for long context)
- All models trained via knowledge distillation

Model	Params (non-embed)	Context	Tokens	Vision
1B	698M	32K	2T	No
4B	3,209M	128K	4T	Yes (417M)
12B	10,759M	128K	12T	Yes (417M)
27B	25,600M	128K	14T	Yes (417M)

Qwen 3 [Qwen Team 2025]

- Most setups are based on Qwen2
  - But, QKV bias is removed (Qwen2 used it as a distinctive feature)
  - QK-Norm is used instead (replaces QKV bias for stability)

- MoE: No shared experts (Qwen2.5 used shared experts)
  - 28 routed experts, top-8 activation
  - Note: DeepSeek-V3, GLM-4.5, Qwen3-Next (later version of Qwen model) use shared experts; some ongoing debate…

- Data: 36T tokens (2x Qwen2.5's 18T, largest ever at the time)

Qwen 3: Hyperparameters [Qwen Team 2025]

Model	Params	Layers	Q Heads	KV Heads	Context	Vocab
0.6B	0.6B	28	16	8	32K	151,936
4B	4B	36	32	8	128K	151,936
8B	8B	36	32	8	128K	151,936
14B	14B	40	40	8	128K	151,936
32B	32B	64	64	8	128K	151,936

Model	Total	Active	Layers	Experts	Active Exp	Context
30B-A3B	30B	3B	48	128	8	128K
235B-A22B	235B	22B	94	128	8	128K

GLM-4.5 [Zhipu AI 2025]

- Deep and narrow: 92 layers with d_ff = 5120 (DS-V3: 61 layers, 7168d)
  - Aux-loss-free LBL
  - MTP layer (1 layer, MoE-based)
  - QK-Norm (for GLM-4.5 only)
- Training: 23T tokens
- Muon optimizer
  - We will see later in this class

	GLM-4.5	GLM-4.5-Air
Total Params	355B	106B
Active Params	32B	12B
Layers	92 (3 dense + 89 MoE) + 1 MTP	46 (1 dense + 45 MoE) + 1 MTP
Hidden	5120	4096
Attention	96	96
KV Heads	8	8
Total Experts	160	128
Active Experts	8	8
Shared Expert	1	1
Context	128K	128K

GPT-OSS [OpenAI 2025]

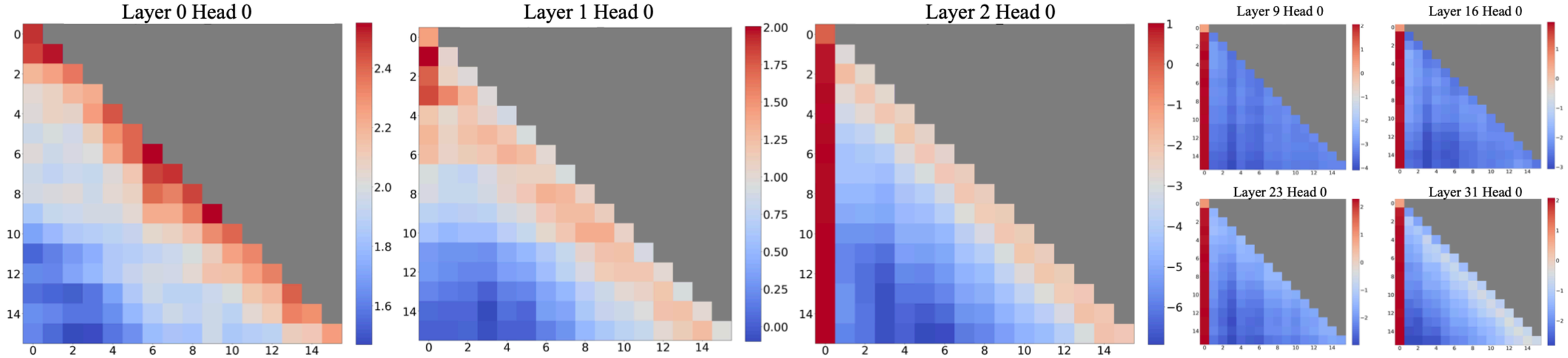
- Wide-and-shallow: hidden=2880 but only 36 / 24 layers
  - GLM-4.5: hidden=5120, 93 layers (narrow-and-deep)
  - Why? wider models → better GPU parallelism → faster inference

- Attention: Interleaved attention (SWA + full) + attention sink
- MoE, YaRN for long context
- MXFP4 quantization for memory reduction

	gpt-oss-120b	gpt-oss-20b
Total Params	116.8B	20.9B
Active Params	5.13B	3.61B
Layers	36	24
Hidden	2880	2880
Q Heads	64	64
KV Heads	8	8
Head Dim	64	64
Experts	128	32
Top-K	4	4
Context	131K	131K
Vocab	201K	201K

Attention Sink [Xiao+ 2024]

Average attention logits of LLaMA 2-7B



first two layers capture the local pattern

and then, the model heavily attends to the initial tokens

- But why? initial tokens are visible to all subsequent tokens
  - So, the model exploits initial tokens as "attention sinks", capturing unnecessary attention (e.g. to make attention prob. sum to 1)

Attention Sink [Xiao+ 2024]

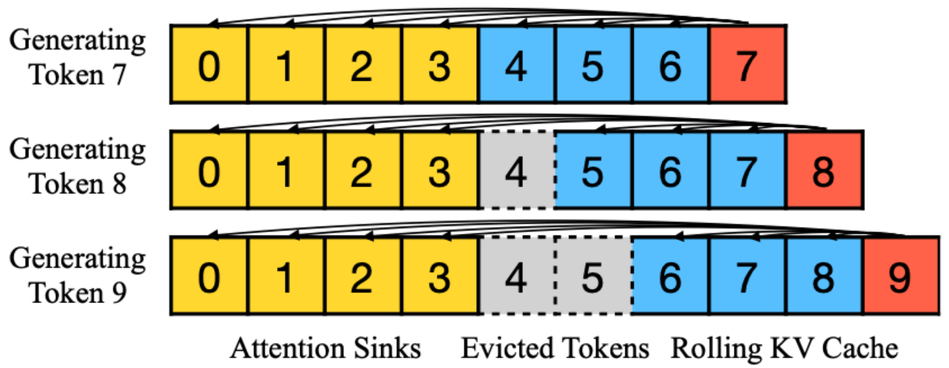
- Why does this matter?
  - Under a restricted context window (e.g. SWA) or beyond the pre-trained length, the attention sinks disappear

  - → It removes a considerable portion of the denominator of softmax

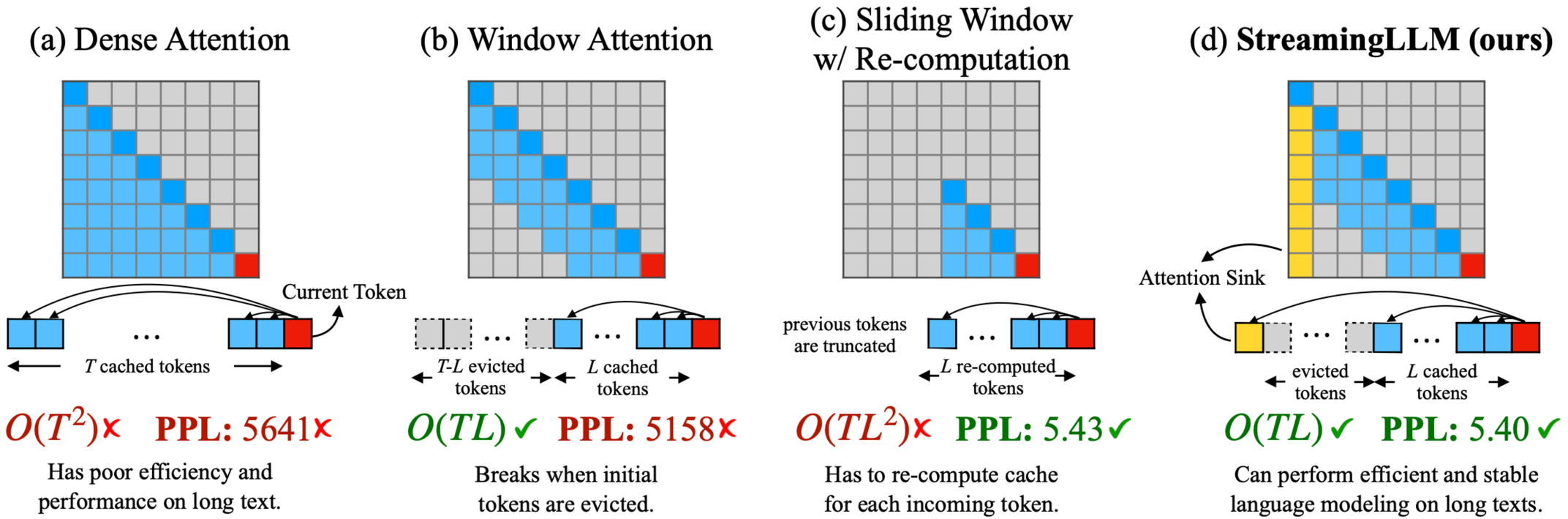
$$\mathbf{SoftMax}(x)_i = \frac{e^{x_i}}{e^{x_1} + \sum_{j=2}^{N} e^{x_j}}, \quad x_1 \gg x_j, j \in 2, \ldots, N$$

distribution shift

- Solution: Rolling KV cache
  - No fine-tuning is required



Attention Sink [Xiao+ 2024]



Kimi K2 [Kimi Team 2025]

- MuonClip optimizer
  - Muon + weight decay + QK clip
- Sparsity scaling law
  - With fixed activated params, does adding more (smaller) experts help?
- MLA, WSD
- Trained on 15.5T tokens

Item	Kimi K2	DeepSeek-V3
Total Params	1.04T	671B
Active Params	32.6B	37B
Layers	61	61
Hidden	7168	7168
Attention Heads	64	128
Routed Experts	384	256
Shared Experts	1	1
Top-K	8	8
Dense Layers	1	3
Expert Grouping	No	Yes
Sparsity	48	32

Muon Optimizer [Jordan 2024]

- What is wrong with Adam/AdamW?
  - We need to store 1st/2nd moments
  - Params. updates are dominated by a few directions with large grads
    - In a large matrix, only small dims change meaningfully per step
    - Most of the weight matrix's capacity sits idle

- Can we lift up the rare (with small grads) directions?
  - Orthogonalization: SVD the 1st moment $M = U\Sigma V^{\mathsf{T}}$ (not use 2nd)
  - Then, drop Σ and use $M = UV^{\mathsf{T}}$
  - All singular values become 1, so every direction gets equal scale

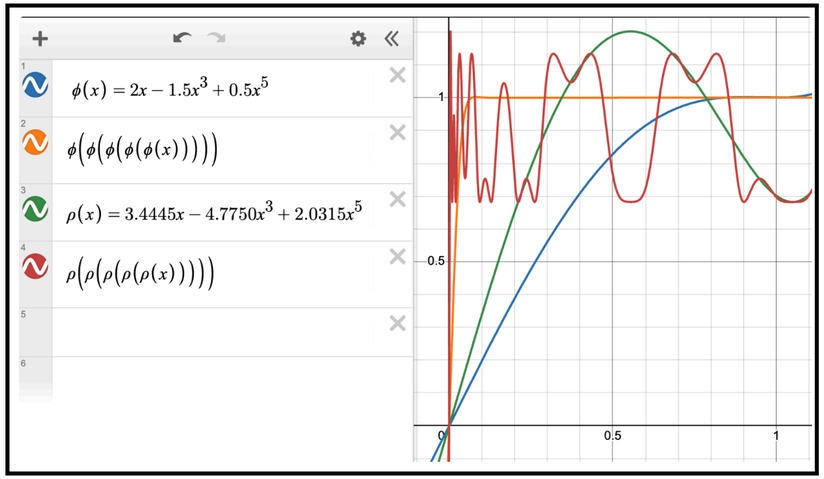
Muon Optimizer [Jordan 2024]

- But, SVD is very expensive; so use Newton-Schulz (NS) iteration instead
  - Approximately orthogonalize the matrix with a few iterations

$$\mathrm{Ortho}(G) = \arg\min_O \{\|O - G\|_F : \text{either } O^\top O = I \text{ or } OO^\top = I\}$$

To understand why the NS iteration orthogonalizes the update, let $G = USV^\top$ be the SVD of the update matrix produced by SGD-momentum. Then running one step of the NS iteration with coefficients (a, b, c) yields the following output:

$$\begin{aligned} G' &:= aG + b(GG^\top)G + c(GG^\top)^2 G \\ &= (aI + b(GG^\top) + c(GG^\top)^2)G \\ &= (aI + bUS^2U^\top + cUS^4U^\top)USV^\top \\ &= U(aS + bS^3 + cS^5)V^\top \end{aligned}$$



- Now the matrix is a semi-orthogonal matrix (singular values are 1)

Muon Optimizer [Jordan 2024]

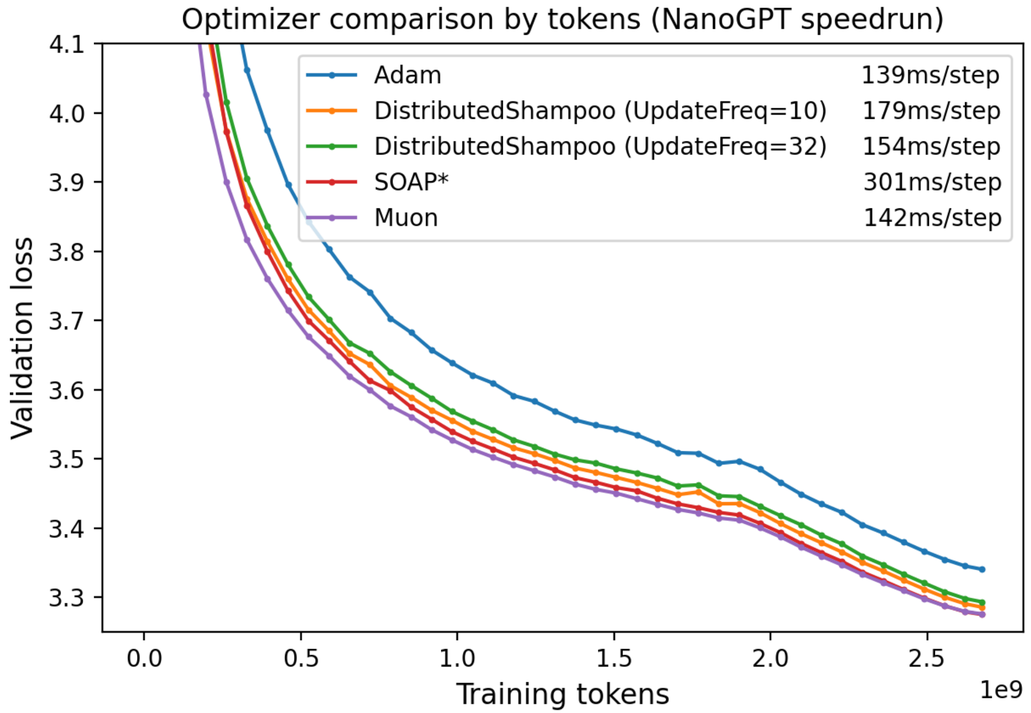
Algorithm 2 Muon

Require: Learning rate η, momentum μ

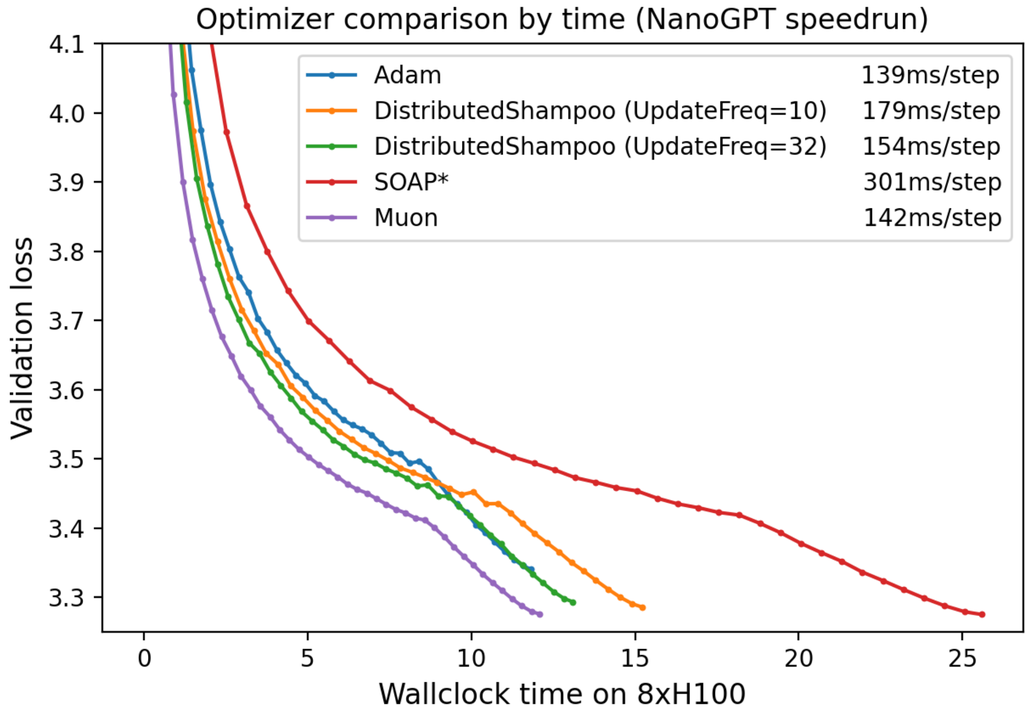
Initialize $B_0 \leftarrow 0$
for $t = 1, \ldots$ **do**
    Compute gradient $G_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta_{t-1})$
    $B_t \leftarrow \mu B_{t-1} + G_t$
    $O_t \leftarrow \text{NewtonSchulz5}(B_t)$
    Update parameters $\theta_t \leftarrow \theta_{t-1} - \eta O_t$
end for
return θ_t

Muon Optimizer [Jordan 2024]

- Note: this is NanoGPT (very small scale compared to frontiers')
  - But GLM-4.5 [Zhipu AI 2025] adopts Muon!



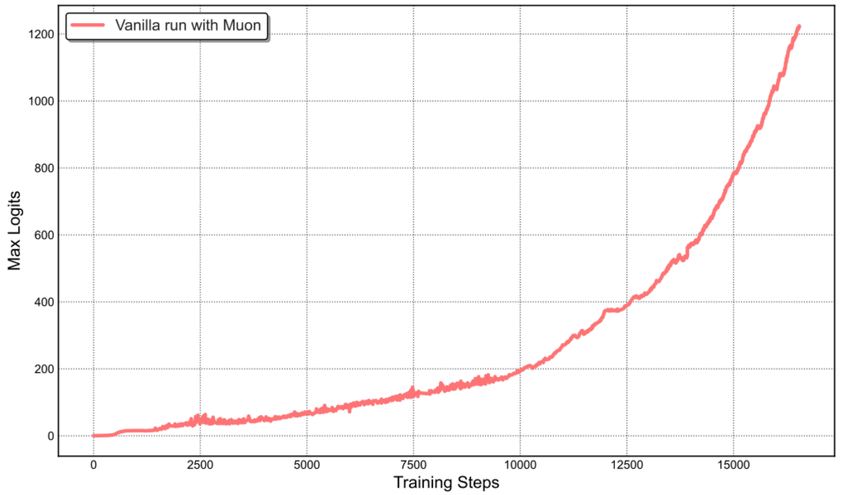
*SOAP is under active development. Future versions will significantly improve the wallclock overhead.



*SOAP is under active development. Future versions will significantly improve the wallclock overhead.

MuonClip [Kimi Team 2025]

- Muon is good at small-scale, but has problem at large-scale models
  - Attention logits rapidly grow (exploding attention logits)
  - This is occurs less with AdamW



- Logit soft-cap clips the attention logits but, QK dot products still grow

- QK-norm is not applicable when using MLA
  - Since K is not materialized

$$O = Softmax\left(\frac{X\ W_{QK}\ C_{KV}^{\top}}{\sqrt{d_k}} + M\right) V_t$$

MuonClip [Kimi Team 2025]

- Taming Muon with QK-clip

Algorithm 1 MuonClip Optimizer

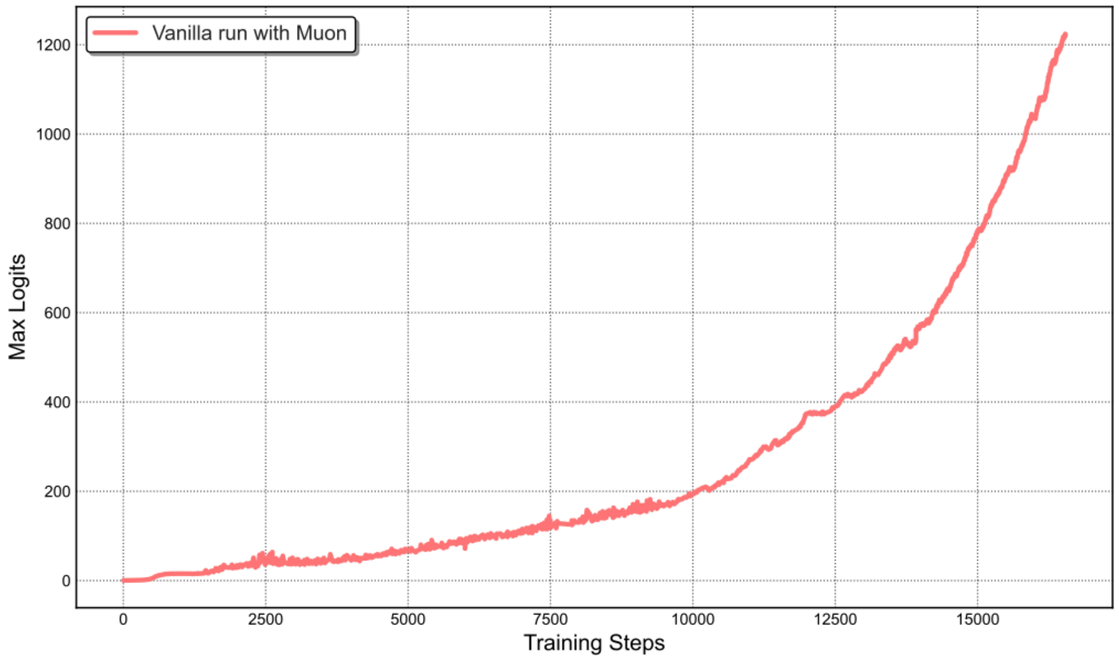
for each training step t **do**
  // 1. Muon optimizer step
  for each weight $\mathbf{W} \in \mathbb{R}^{n \times m}$ **do**
    $\mathbf{M}_t = \mu \mathbf{M}_{t-1} + \mathbf{G}_t$ ▷ $\mathbf{M}_0 = \mathbf{0}$, $\mathbf{G}_t$ is the grad of $\mathbf{W}_t$, μ is momentum
    $\mathbf{O}_t = \text{Newton-Schulz}(\mathbf{M}_t) \cdot \sqrt{\max(n, m)} \cdot 0.2$ ▷ Match Adam RMS
    $\mathbf{W}_t = \mathbf{W}_{t-1} - \eta \left(\mathbf{O}_t + \lambda \mathbf{W}_{t-1} \right)$ ▷ learning rate η, weight decay λ
  end for
  // 2. QK-Clip
  for each attention head h in every attention layer of the model **do**
    Obtain $S^h_{\max}$ already computed during forward
    if $S^h_{\max} > \tau$ **then**
      $\gamma \leftarrow \tau / S^h_{\max}$
      $\mathbf{W}^h_{qc} \leftarrow \mathbf{W}^h_{qc} \cdot \sqrt{\gamma}$
      $\mathbf{W}^h_{kc} \leftarrow \mathbf{W}^h_{kc} \cdot \sqrt{\gamma}$
      $\mathbf{W}^h_{qr} \leftarrow \mathbf{W}^h_{qr} \cdot \gamma$
    end if
  end for
end for

$$S^h_{\max} = \frac{1}{\sqrt{d}} \max_{\mathbf{X} \in B} \max_{i,j} \mathbf{Q}^h_i \mathbf{K}^{h\top}_j$$

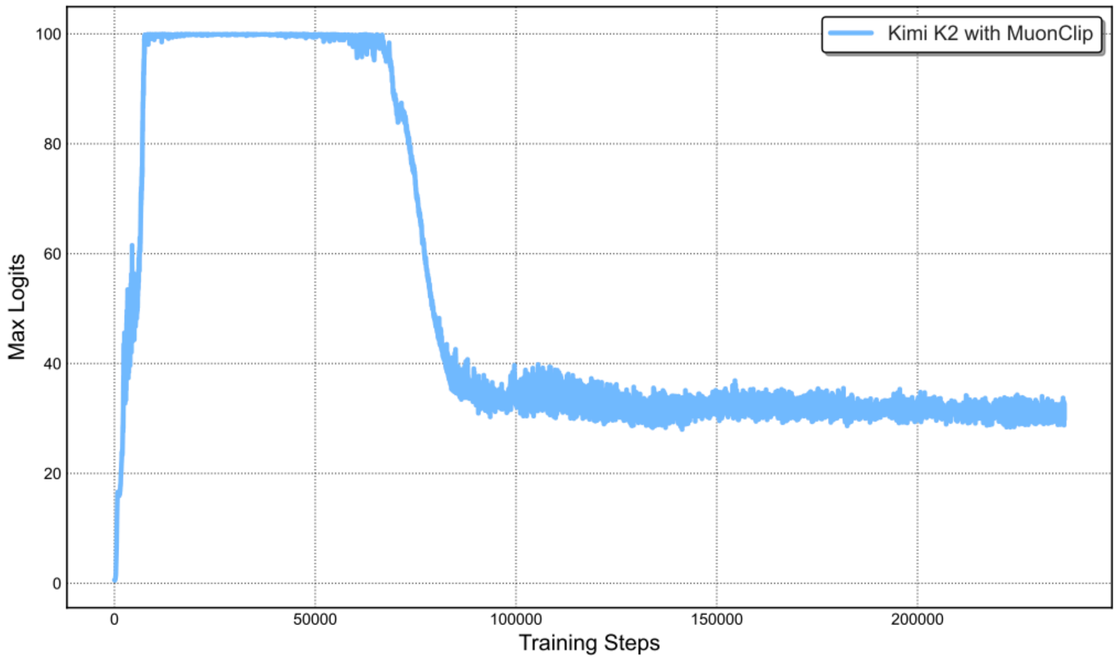
rescale QK matrices when QK explode

MuonClip [Kimi Team 2025]

- Taming Muon with QK-clip



Kimi K2 uses $\tau = 100$



MuonClip [Kimi Team 2025]

- Other techniques to tame Muon [Liu+ 2025]
  - + Weight decay

> **Weight Decay** While Muon performs significantly better than AdamW on a small scale as shown by K. Jordan et al. 2024, we found the performance gains diminish when we scale up to train a larger model with more tokens. We observed that both the weight and the layer output's RMS keep growing to a large scale, exceeding the high-precision range of bf16, which might hurt the model's performance. To resolve this issue, we introduced the standard AdamW (Loshchilov et al. 2019) weight decay mechanism into Muon[2].

  - Hybrid update with Muon + AdamW
    - Muon cannot be used for 1D params (orthogonality is not defined)
    - Empirically, AdamW is better for embedding layers

MuonClip [Kimi Team 2025]

- Other techniques to tame Muon [Liu+ 2025]
  - Consistent update RMS
    - AdamW maintains update RMS as 1 (0.2-0.4 in practice)
    - But update RMS of Muon is $\sqrt{1/max(A,B)}$ for [A, B]-shaped matrix

> - When $\max(A, B)$ is too large, e.g. the dense MLP matrix, the updates become too small, thus limiting the model's representational capacity and leading to suboptimal performances;
> - When $\max(A, B)$ is too small, e.g. treating each KV head in GQA (Shazeer 2019) or MLA (DeepSeek-AI et al. 2024) as a separate parameter, the updates become too large, thus causing training instabilities and leading to suboptimal performances as well.

    - So, match RMS of Muon and AdamW as:

$$\mathbf{W}_t = \mathbf{W}_{t-1} - \eta_t(0.2 \cdot \mathbf{O}_t \cdot \sqrt{\max(A, B)} + \lambda \mathbf{W}_{t-1})$$

Sparsity Scaling Law [Moonshot AI 2025]

- With fixed activated params, does adding more (smaller) experts help?
  - Yes!
  - Kimi K2 uses 384 routed experts (48 sparsity)

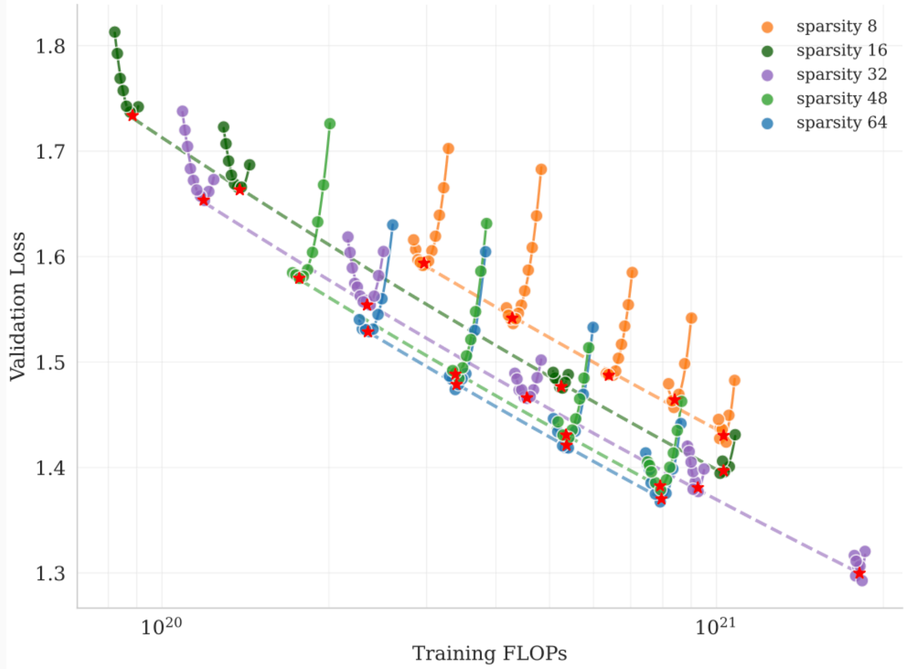
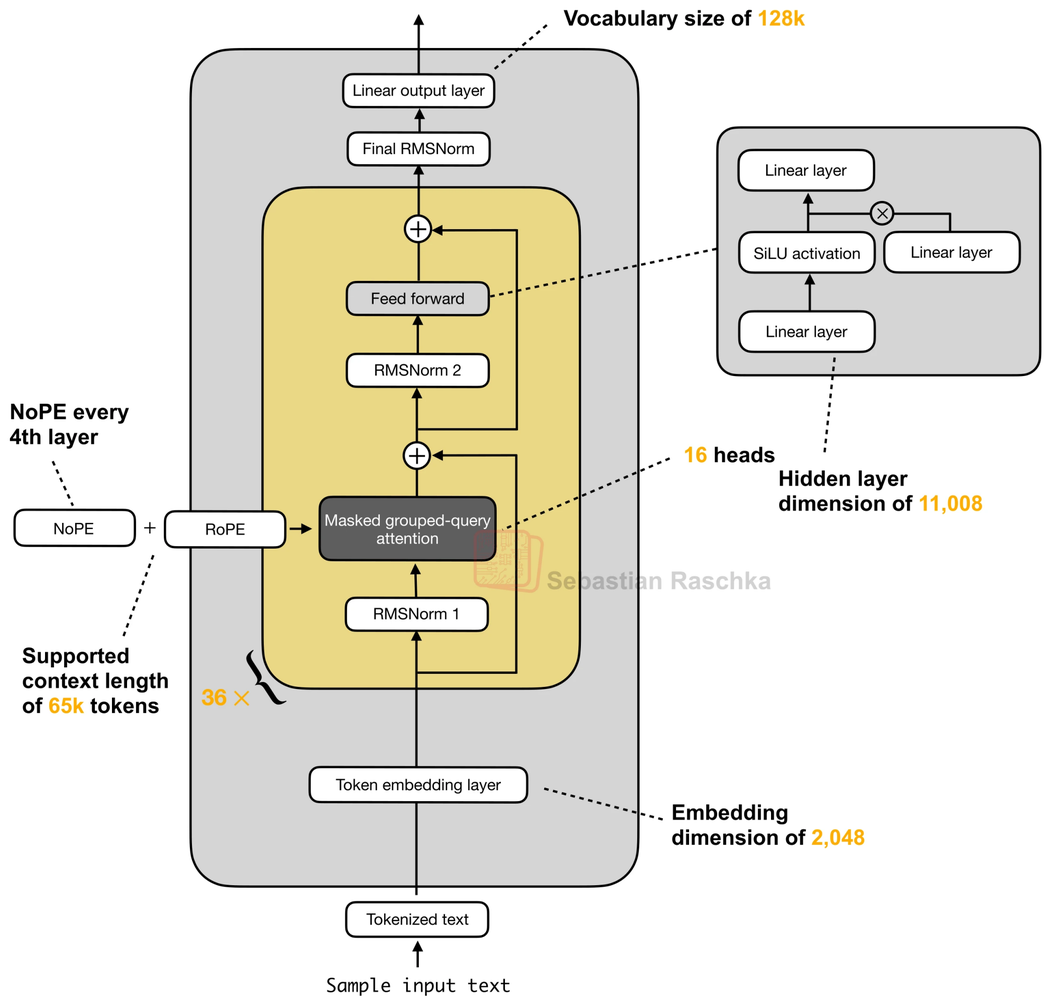


Figure 5: Sparsity Scaling Law. Increasing sparsity leads to improved model performance. We fixed the number of activated experts to 8 and the number of shared experts to 1, and varied the total number of experts, resulting in models with different sparsity levels.

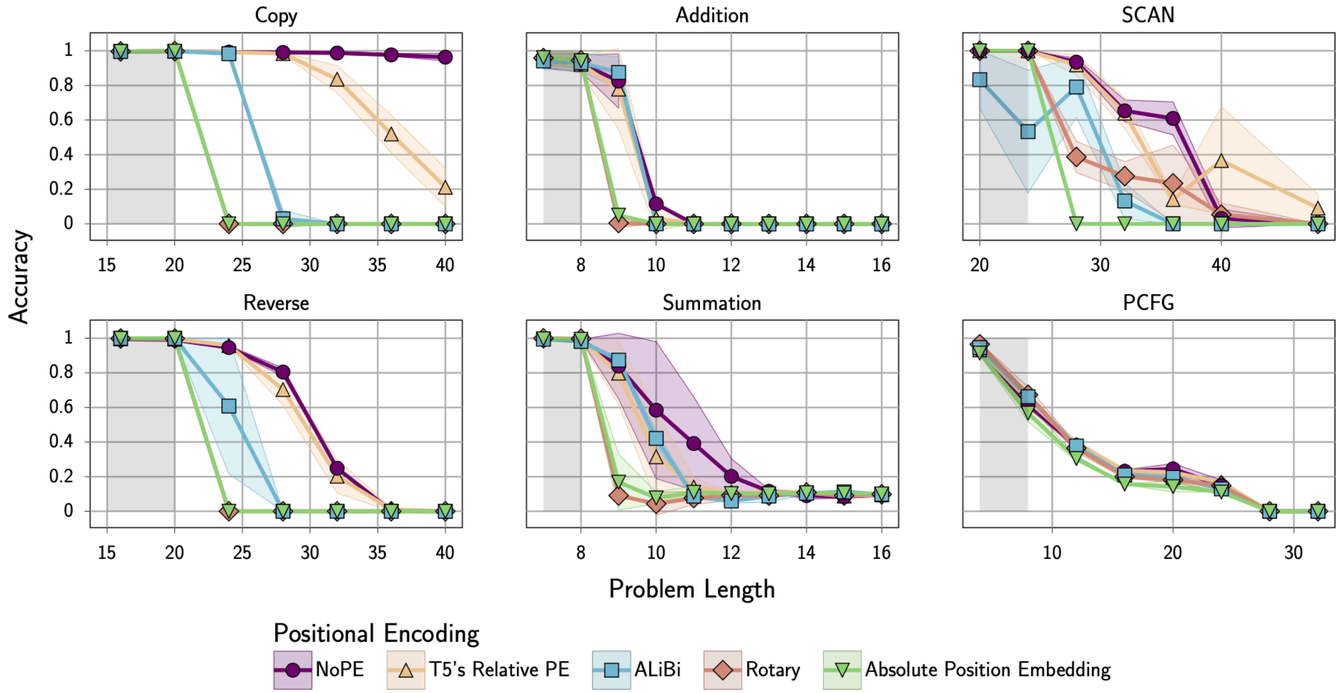
SmolLM3 [HuggingFace 2025]

- ~3B parameters, GQA, SwiGLU, RMSNorm (Pre-norm)
- Positional Encoding:
  - RoPE + NoPE



NoPE: No Positional Encoding [Kazemnejad+ 2023]

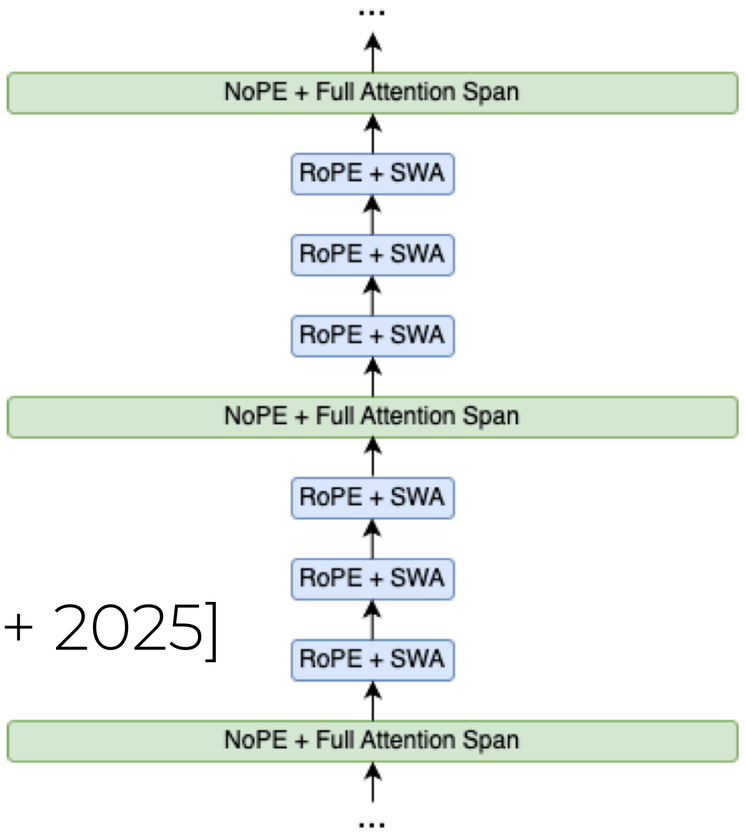
- Actually, the model learns token position implicitly
  - Why? with casual generation, it still imposes an ordering structure
  - Causal mask encodes relative ordering (token attends only to past)
- And RoPE is bad at context generalization



RoPE + NoPE [HuggingFace 2025, Yang+ 2025, Meta AI 2025]

- A practical version is mixing RoPE and NoPE
  - RoPE: cover local context
  - NoPE: cover global context

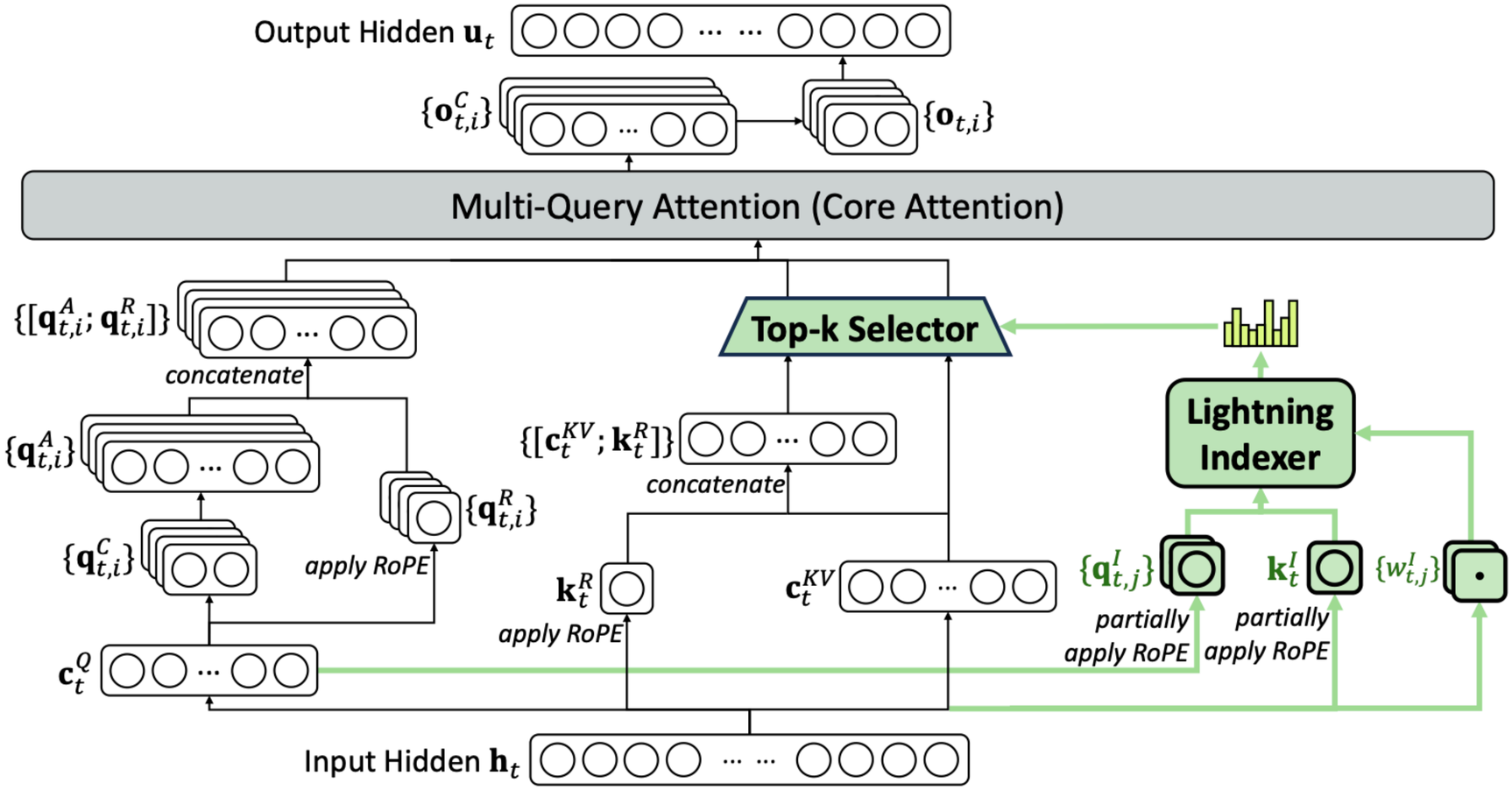
- Recent hybrid attention models often keep RoPE in local SWA layers and use NoPE in global attention layers



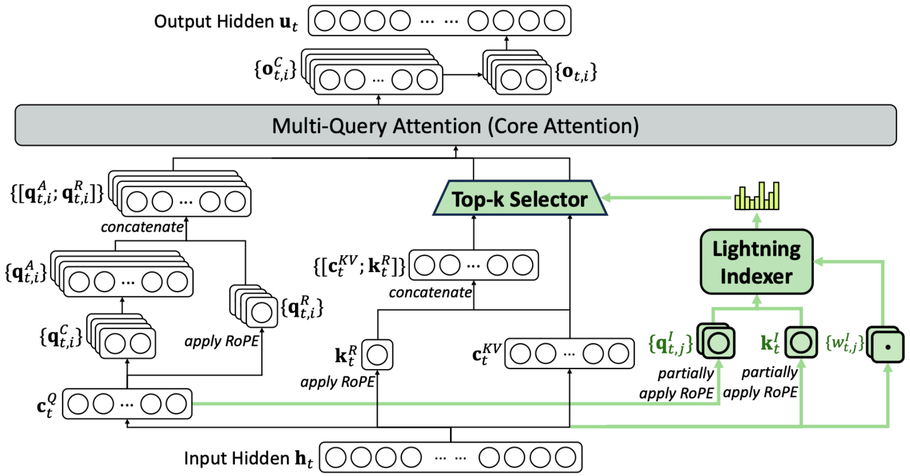
[Yang+ 2025]

DeepSeek-V3.2 [DeepSeek-AI 2025]

- DeepSeek sparse attention (DSA) + MLA



DeepSeek-V3.2 [DeepSeek-AI 2025]



- Lightning indexer
  - Compute index score between the Q and K

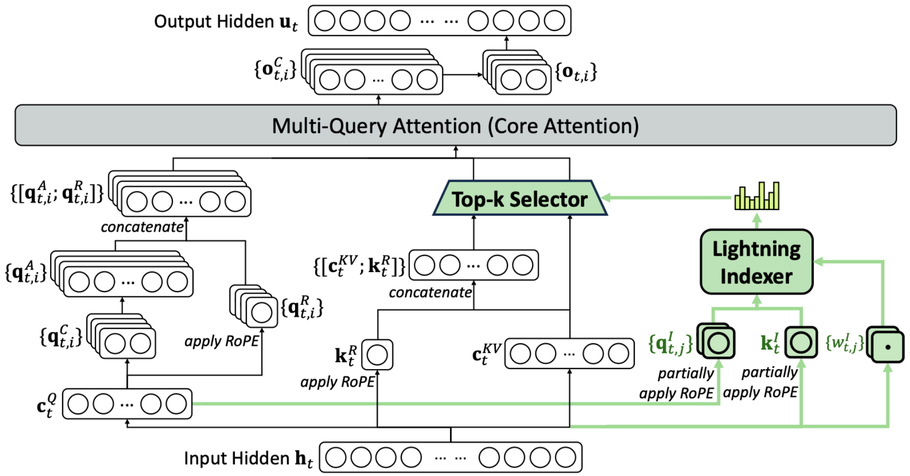
$$I_{t,s} = \sum_{j=1}^{H^I} w^I_{t,j} \cdot \mathrm{ReLU}\left(\mathbf{q}^I_{t,j} \cdot \mathbf{k}^I_s\right),$$

  - This is lightning compared to the standard attention because
    - (1) No softmax (2) fewer heads (3) implemented in FP8

- Top-k selector
  - It retrieves top-k selected KV entries
  - Attention output is computed between the Q and selected KVs

$$\mathbf{u}_t = \mathrm{Attn}\left(\mathbf{h}_t, \left\{\mathbf{c}_s \,\middle|\, I_{t,s} \in \text{Top-k}\left(I_{t,:}\right)\right\}\right).$$

DeepSeek-V3.2 [DeepSeek-AI 2025]



- How to train DSA models?
  - From the DS-V3.1 checkpoint,
    - (1) Dense warm-up stage $\mathcal{L}^I = \sum_t \mathbb{D}_{\mathrm{KL}}\left(p_{t,:} \,\|\, \mathrm{Softmax}\left(I_{t,:}\right)\right).$
      - Use dense attention, freeze params except lightning indexer
      - Train indexer to follow attention scores of main attention heads
    - (2) Sparse training stage $\mathcal{L}^I = \sum_t \mathbb{D}_{\mathrm{KL}}\left(p_{t,\mathcal{S}_t} \,\|\, \mathrm{Softmax}\left(I_{t,\mathcal{S}_t}\right)\right).$
      - Train the main models

- → Full attention: $O(L^2)$ / DSA attention $O(Lk)$ (k=2048)
  - Indexer is still $O(L^2)$, but much cheaper than full MLA attention

Key Takeaway: 2025

- What hasn't changed since 2023 - 2025?
  - Decoder-only architecture + autoregressive left-to-right generation
  - RoPE(though hybrid RoPE + NoPE is used)
  - SwiGLU for FFN activation
  - BPE-based tokenization
  - Pre-norm placement

Key Takeaway: 2025

- The winning formula in 2025
  - Norm: RMSNorm + QK-Norm (mostly pre-norm except OLMo 2)
  - Attention: GQA with QK-Norm or MLA (+DSA for DeepSeek)
    - Hybrid linear attentions
  - FFN: SwiGLU - unchanged since LLaMA
  - PE: RoPE is dominant, but NoPE is emerging as alternatives
  - MoE: Fine-grained
  - Training: WSD or cosine decay, MTP, FP8
  - Optimizer: Still AdamW but Muon family emerging

Model Scaling Trends: 2025

Metric	2023	2024	2025
Max Total Params	70B (LLaMA 2)	671B (DSv3)	1.04T (Kimi K2)
Max Training Data	2T (LLaMA 2)	18T (Qwen2.5)	36T (Qwen3)
Max Context	32K (Mixtral)	1M (MiniMax-01)	10M (LLaMA 4 Scout)
Max Experts	8 (Mixtral)	256 (DSv3)	384 (Kimi K2)

Korean Models: K-EXAONE [LG AI Research 2026]

- Muon optimizer + WSD scheduler
- MTP for self-drafting (~1.5x decoding speedup)
- Interleaved SWA + (GQA) QK norm attention (RoPE in SWA only)
- Sparse MoE (with 1 shared)

Block	Configuration	Value
Main Block	Layers (Total/SWA/GA)	48 / 36 / 12
	Sliding Window Size	128
	Attention Heads (Q/KV)	64 / 8
	Head Dimensions	128
	Experts (Total/Shared/Activated)	128 / 1 / 8
	Parameters (Total/Activated)	236B / 23B
MTP Block	Attention Heads (Q/KV)	64 / 8
	Head Dimensions	128
	Parameters	0.52B

256K context

3 training stages:

1) 8K length train
2) → 32K
3) → 256K

Korean Models: Solar Open [Upstage Solar Team 2026]

- Similar to GPT-OSS, Qwen3, GLM-4.5, DSv3 (with careful HP search)
- Dataset: 4.5T synthetic data (Korean only 0.8% of FineWeb)
- Training: low-to-high quality curriculum
  - Phase 1: Diverse data, 10% synthetic
  - Phase 2: Progressively filtered (32% → 64% synthetic)
  - Phase 3: 1.5T high-quality Korean culture, math, code
  - Total: ~19.7T pre-training + 1.15T mid-training

Hyperparameter	Value
Total Parameters	102.6B
Active Parameters per Token	12B
Context Length	131,072
Vocabulary Size	196,608
Num Layers	48
Hidden Size	4,096
Intermediate Size	10,240
MoE Intermediate Size	1,280
Num Attention Heads	64
Num Key-Value Heads	8
Head Dimension	128
Num Dense Layers	0
Num Experts (Total)	129
Num Experts (Routed)	128
Num Shared Experts	1
Num Experts per Token (Top-k)	8
Activation Function	SiLU (Elfwing et al., 2018)
Positional Embedding	RoPE (θ: 10^6) (Su et al., 2024)

480 B200

Korean Models: HCX 32B [Naver Cloud 2026]

- Base recipes
  - RMSNorm, SwiGLU, RoPE (with 500,000 base), GQA-8

- Pre-training curriculum
  - Pre-training: general web data
  - Mid-training: high-quality data + context length extension
    - RoPE base 500,000 to 5M

Domain	Stage 1	Stage 2	Stage 3	Stage 4
General	79.4%	61.6%	59.1%	17.0%
Code	12.0%	20.1%	20.0%	25.2%
Math	8.6%	18.2%	20.0%	25.3%
Instruction tuning	0.0%	0.1%	1.0%	32.5%

Summary

Component	2023	2024	2025
Attention	MHA → GQA	+ MLA, Linear Attn.	GQA(standard), MLA+DSA, Linear hybrids
Pos Encoding	RoPE	RoPE	RoPE or RoPE + NoPE
MoE Experts	8 (Mixtral)	64 → 256	128 → 384
MoE Routing	Aux loss	+ Aux-loss-free	Both used
Stabilization	—	Logit Soft-Cap	→ QK-Norm
Optimizer	AdamW	AdamW + WSD	Still AdamW but Muon?
Data Scale	1-2T	up to 18T	up to 36T