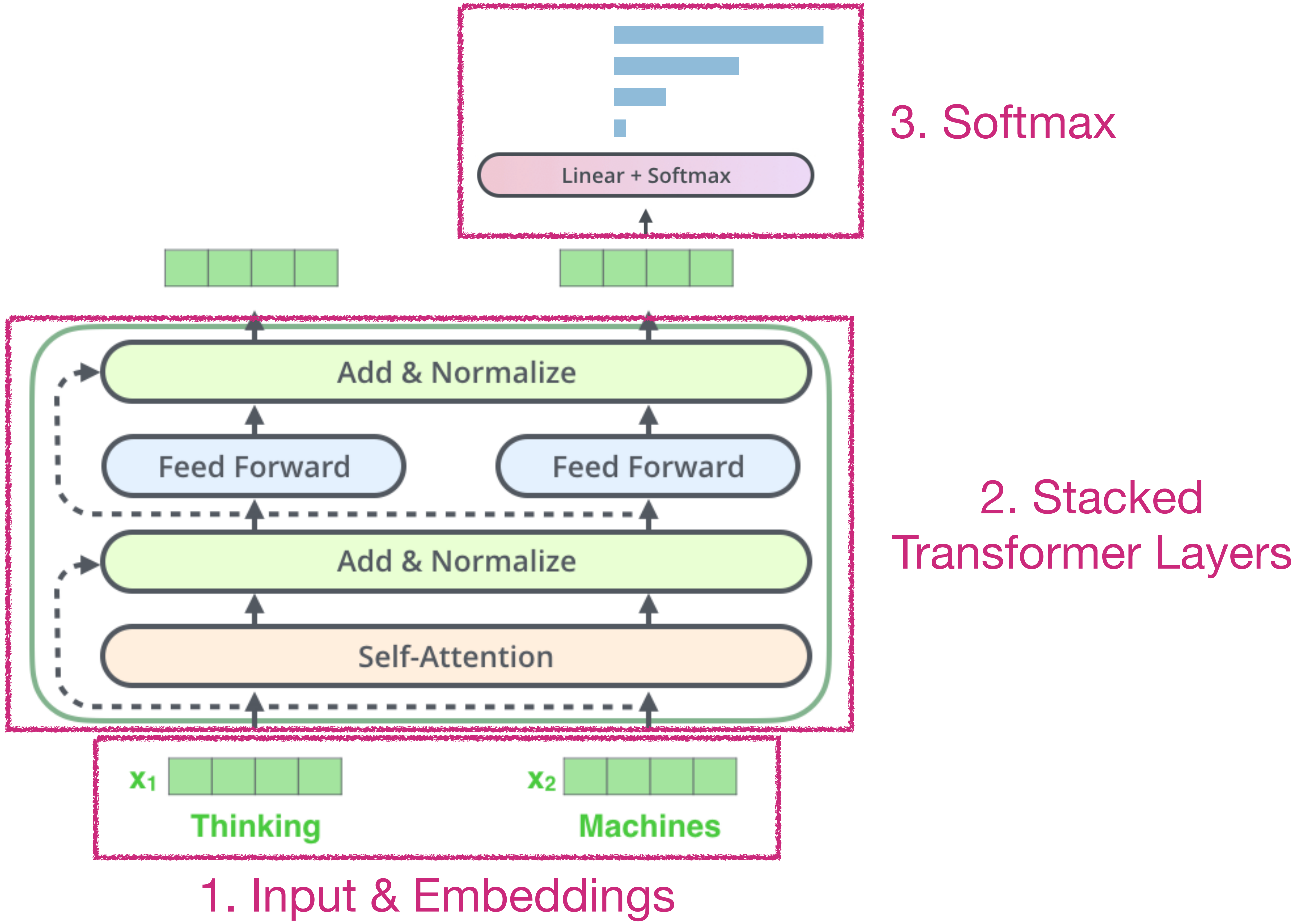
Advanced Architectures

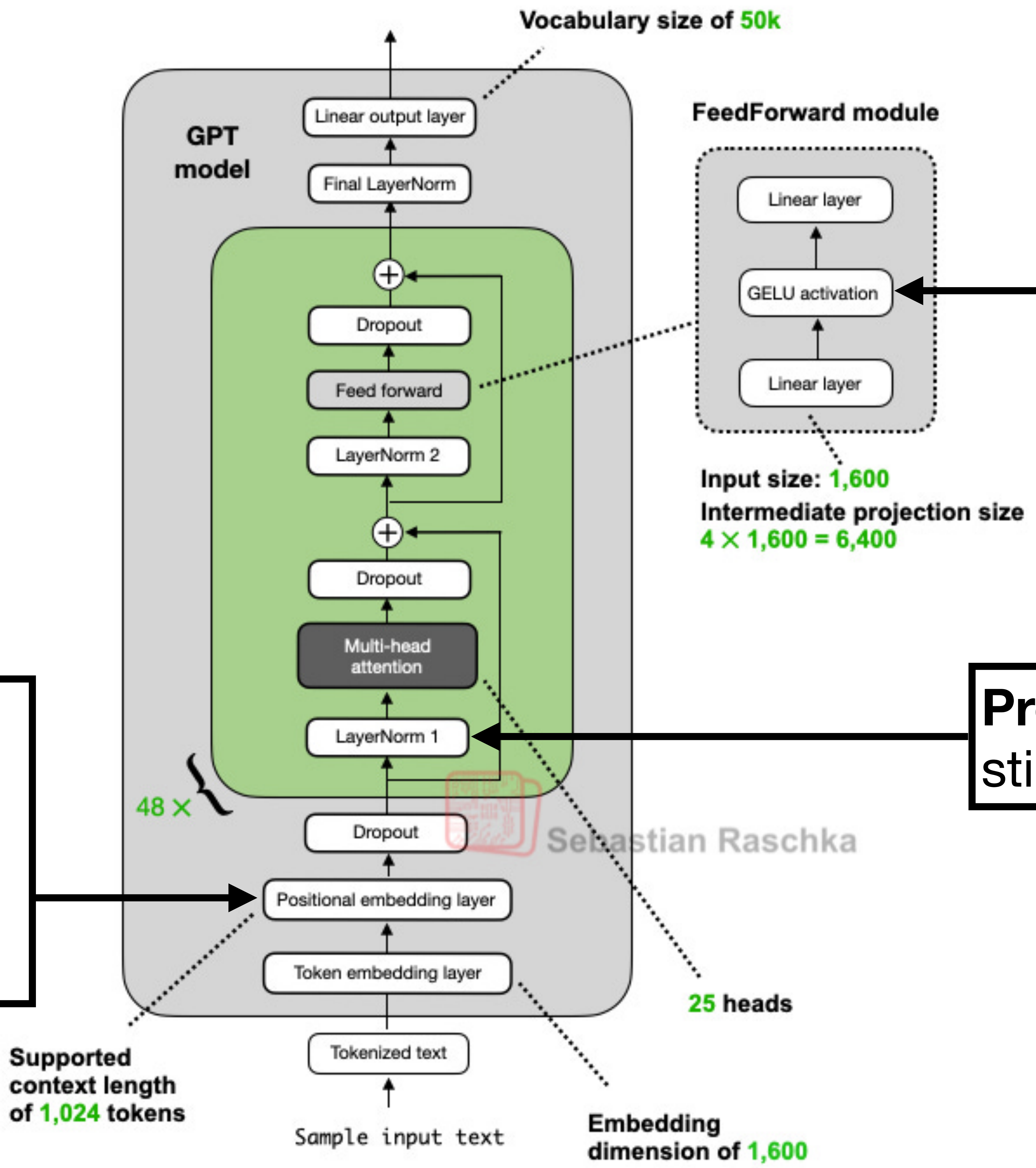
CS 288 Spring 2026
UC Berkeley
cal-cs288.github.io/sp26

Berkeley EECS BAIR

Case study 1: Transformers (2017)



Case study 2: GPT-2 XL 1.5B (2019)



GELU instead of ReLU

Pre-norm instead of Post-norm; still using LayerNorm

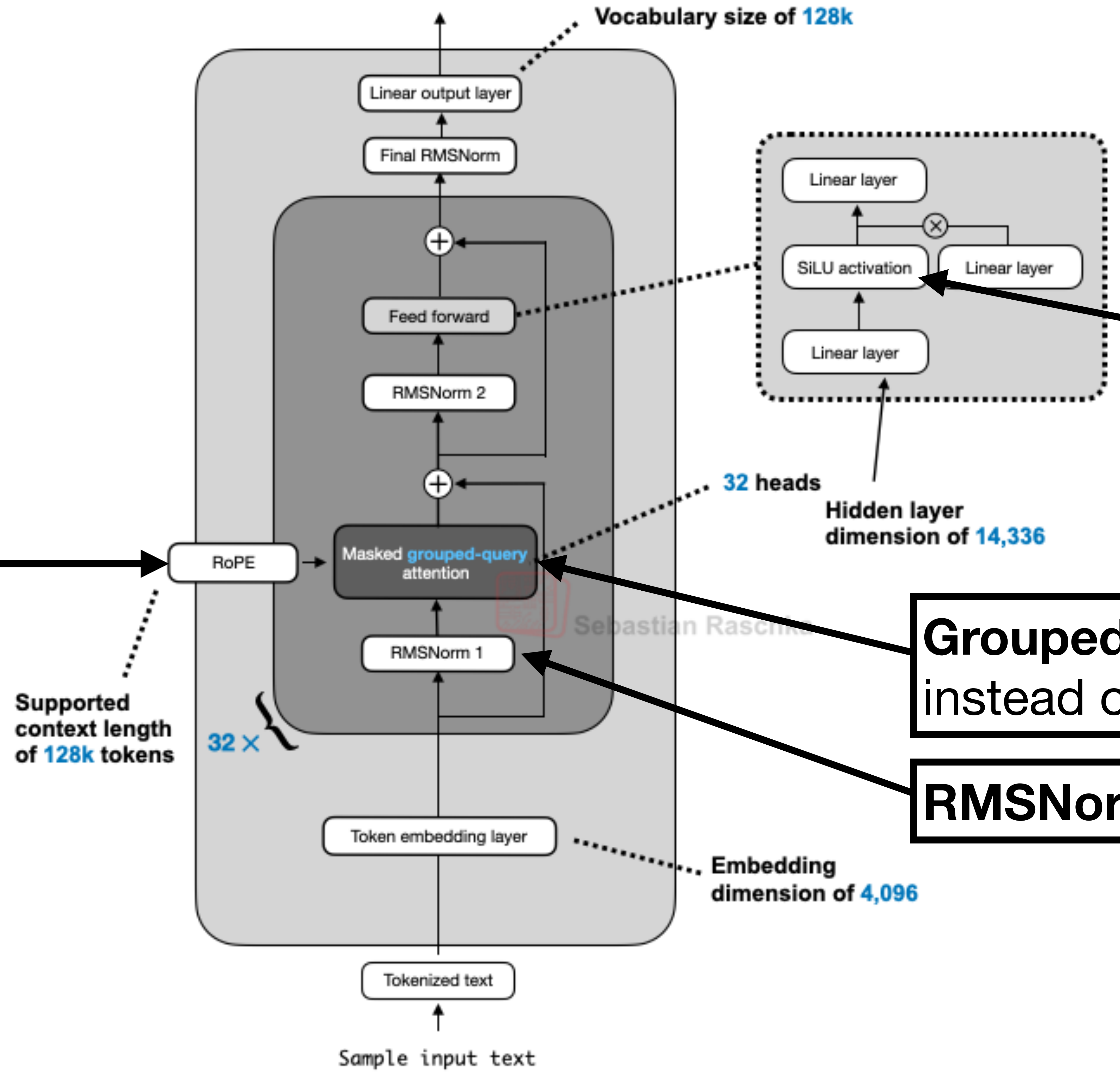
Absolute embeddings instead of Sinusoidal Encoding (still in the input layers): add a position vector to the embedding; supported 1,024 tokens

Image source: https://sebastianraschka.com/blog/

Case Study 3: Llama 3 8B (2024)

RoPE

- Add position encoding in the **attention**
- Model the **relative** positional information
- Now support 128K tokens



SwiGLU (adding gating to SiLU activation)

Grouped-query attention instead of multi-head attention

RMSNorm instead of LayerNorm

Image source: https://sebastianraschka.com/blog/

So many variants in 2026: What matters?

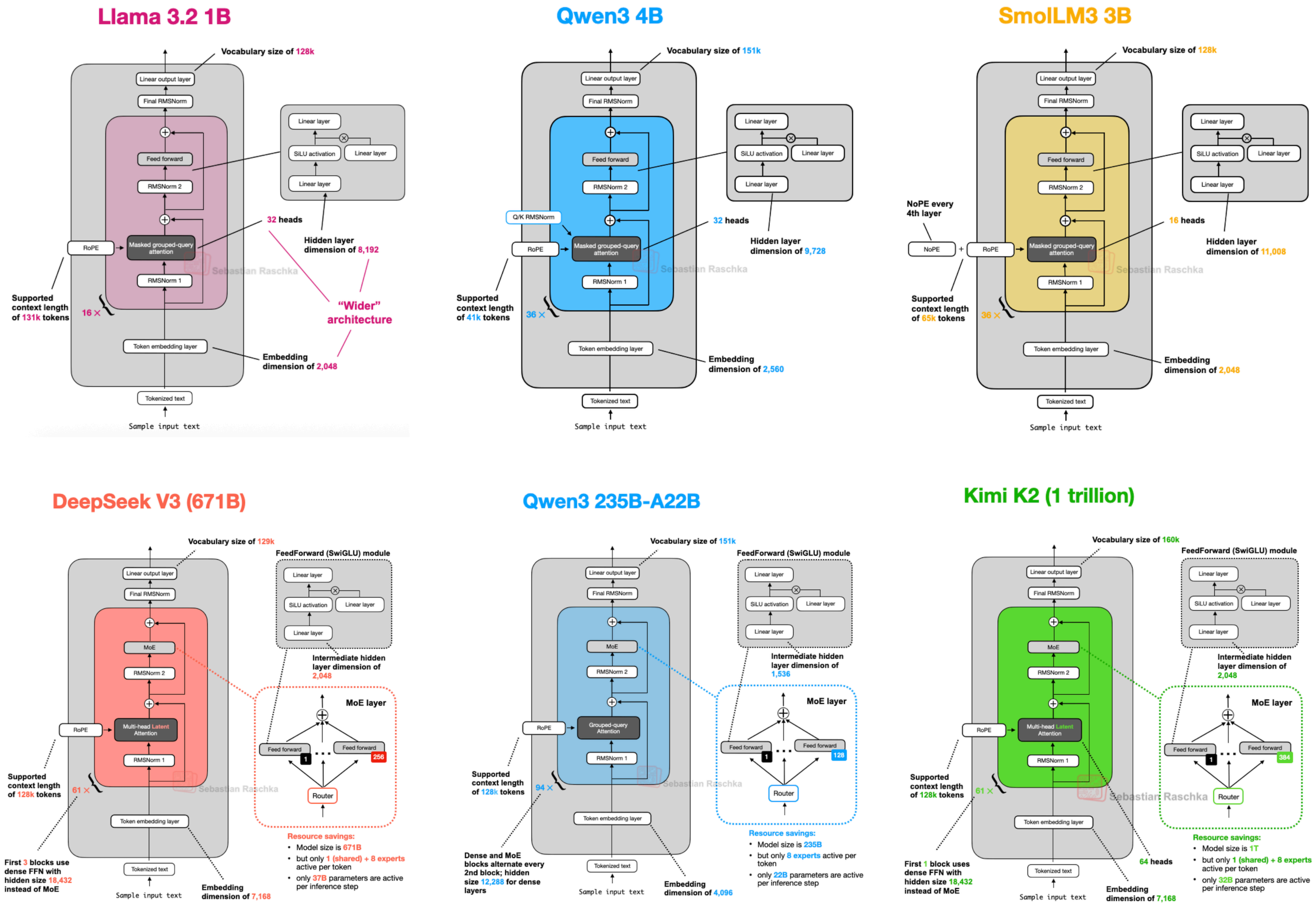
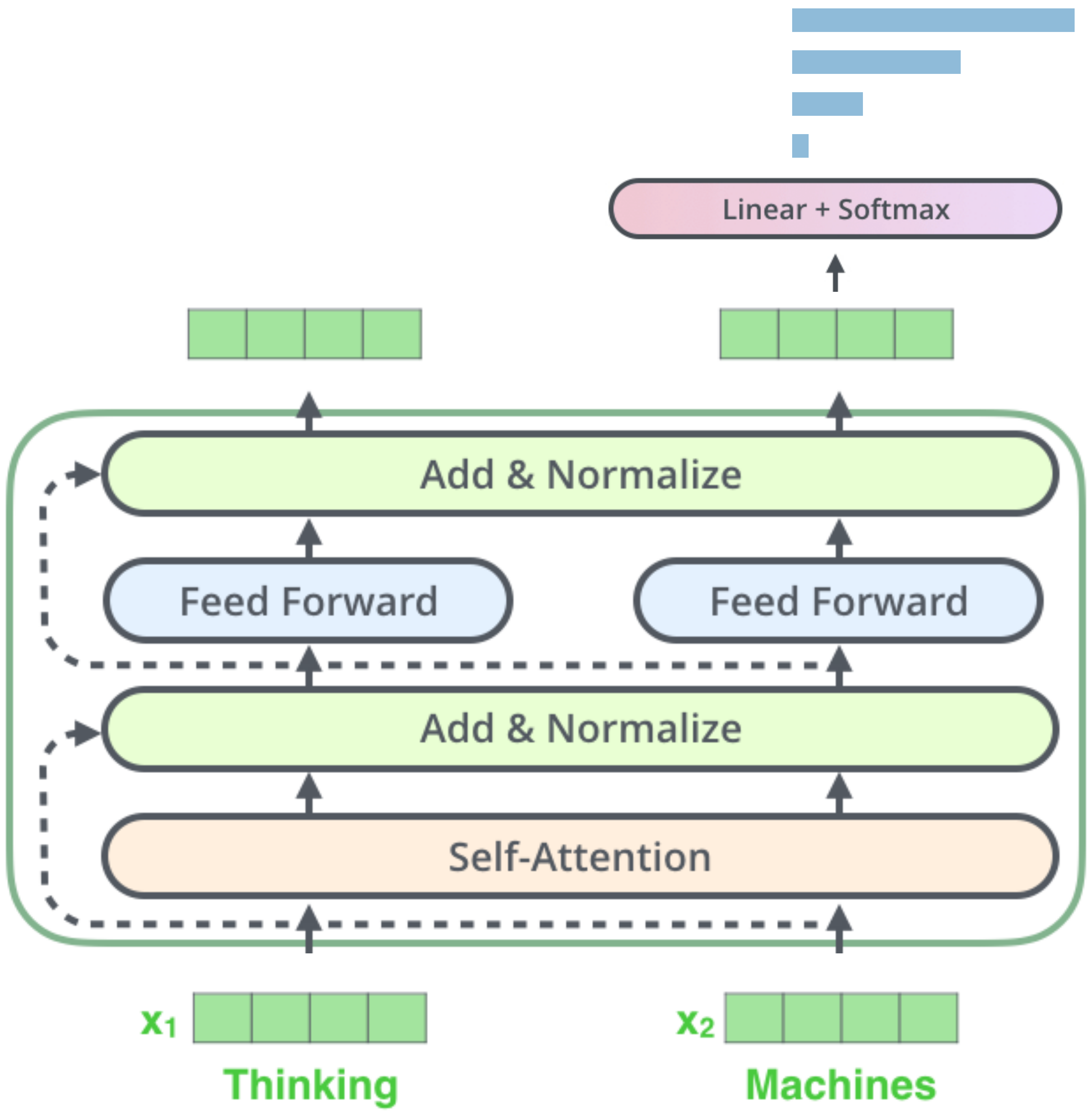


Image source: https://sebastianraschka.com/blog/

Today's topic: Advanced architectures



- We already covered the architecture ("Transformers") on 02/10 (a month ago!), ending with the Llama variants (2024).
- How do 2026 LLM architectures look like?
    - ~~Scaling~~ More efficient scaling → Mixture-of-Experts (MoE)
    - Scaling the "context window" → Attention variants

Mixture-of-Experts (MoE)

Mixture-of-Experts in ~2021

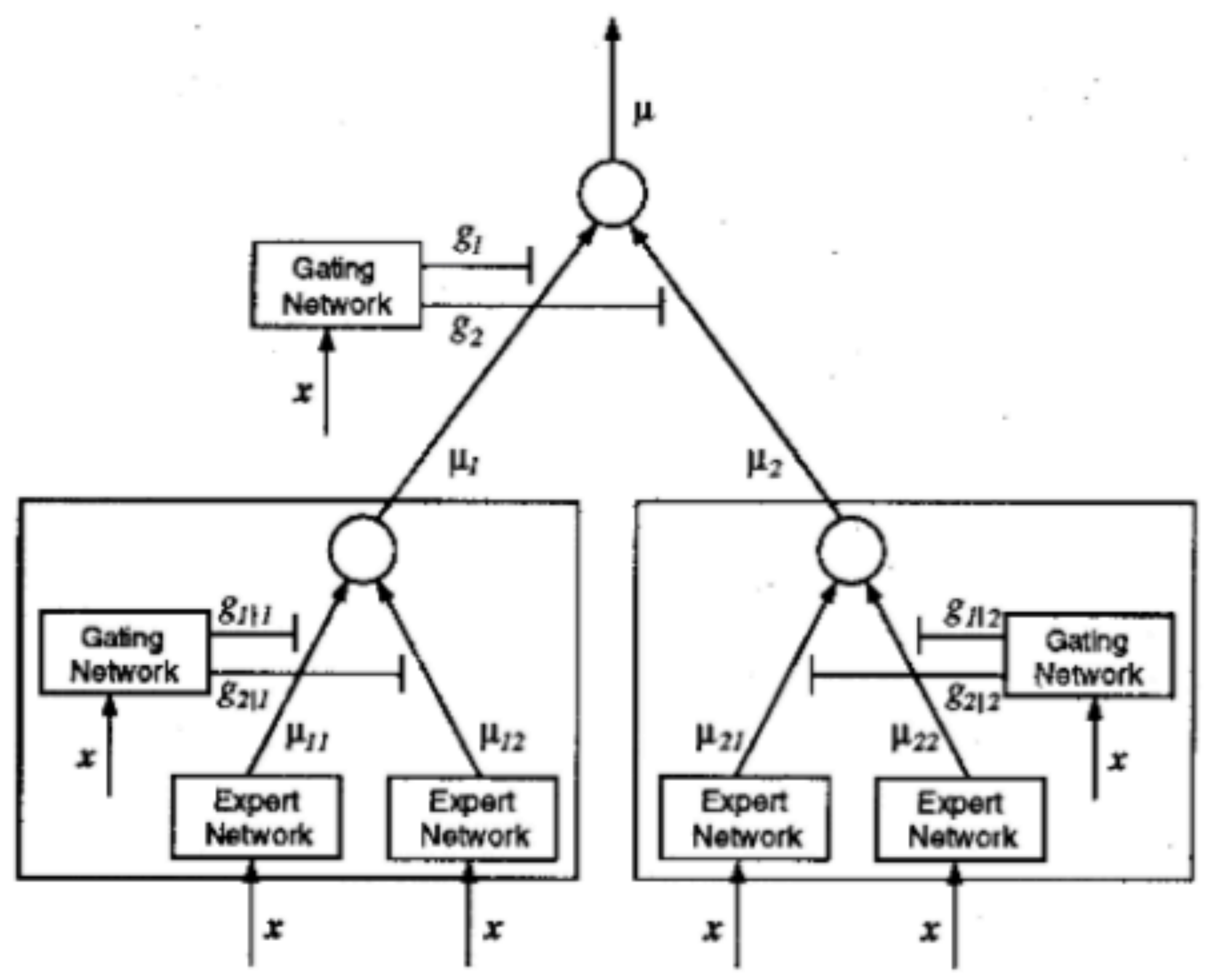


Figure 1: A two-level hierarchical mixture of experts.

Hierarchical Mixtures of Experts for the EM Algorithm, 1993

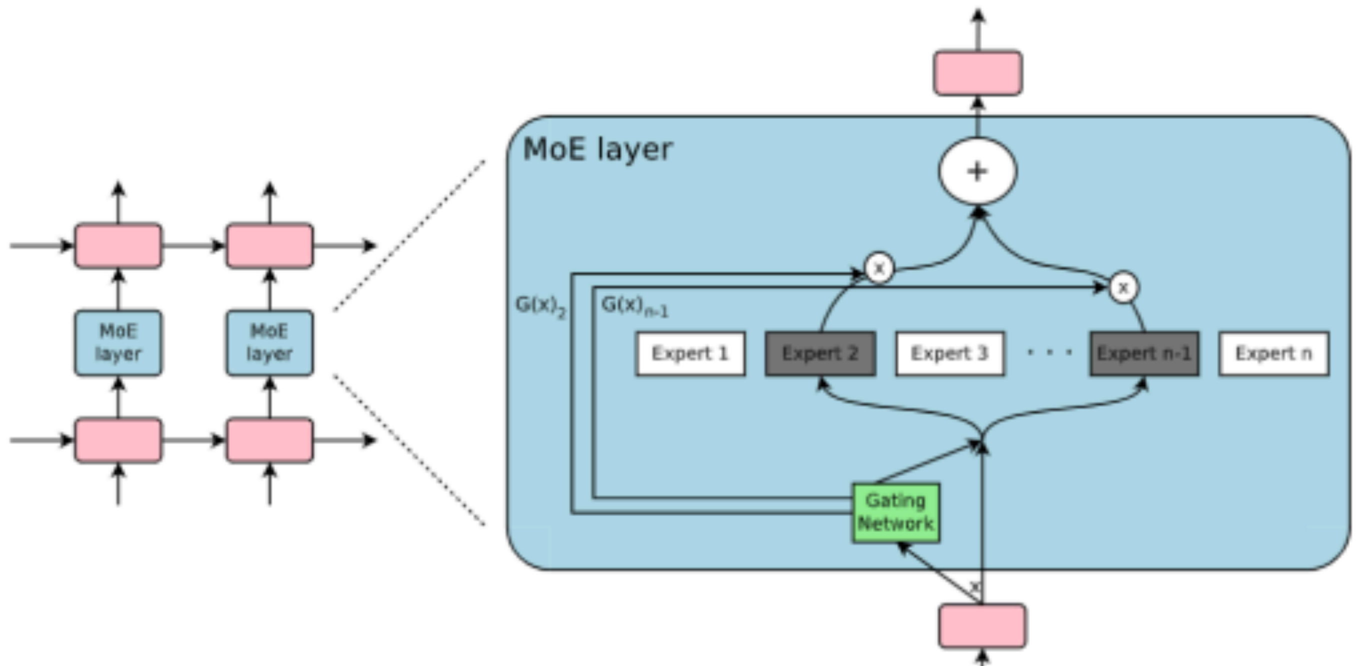
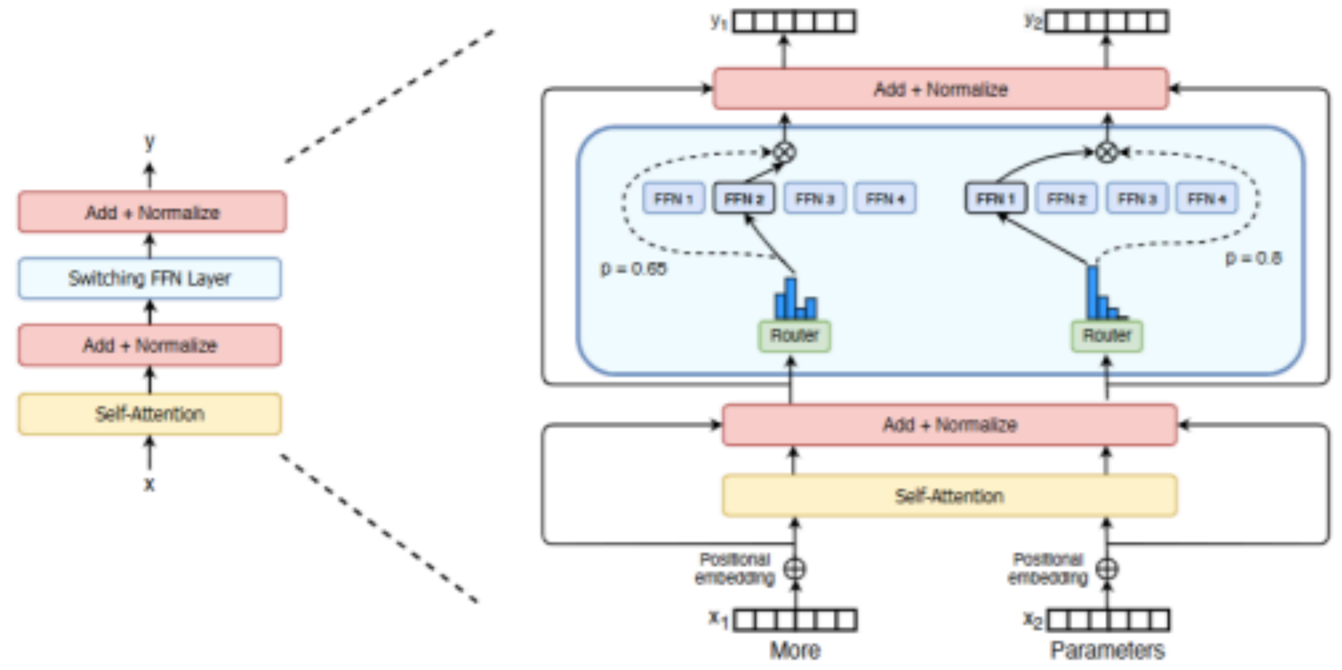


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

Sparse-Gated Mixture of Experts in LSTM, 2017



Sparse-Gated Mixture of Experts in Transformer, 2021

Source: ICML 2024 MoE Tutorial

Mixture-of-Experts in ~2021

GSHARD: SCALING GIANT MODELS WITH CONDITIONAL COMPUTATION AND AUTOMATIC SHARDING

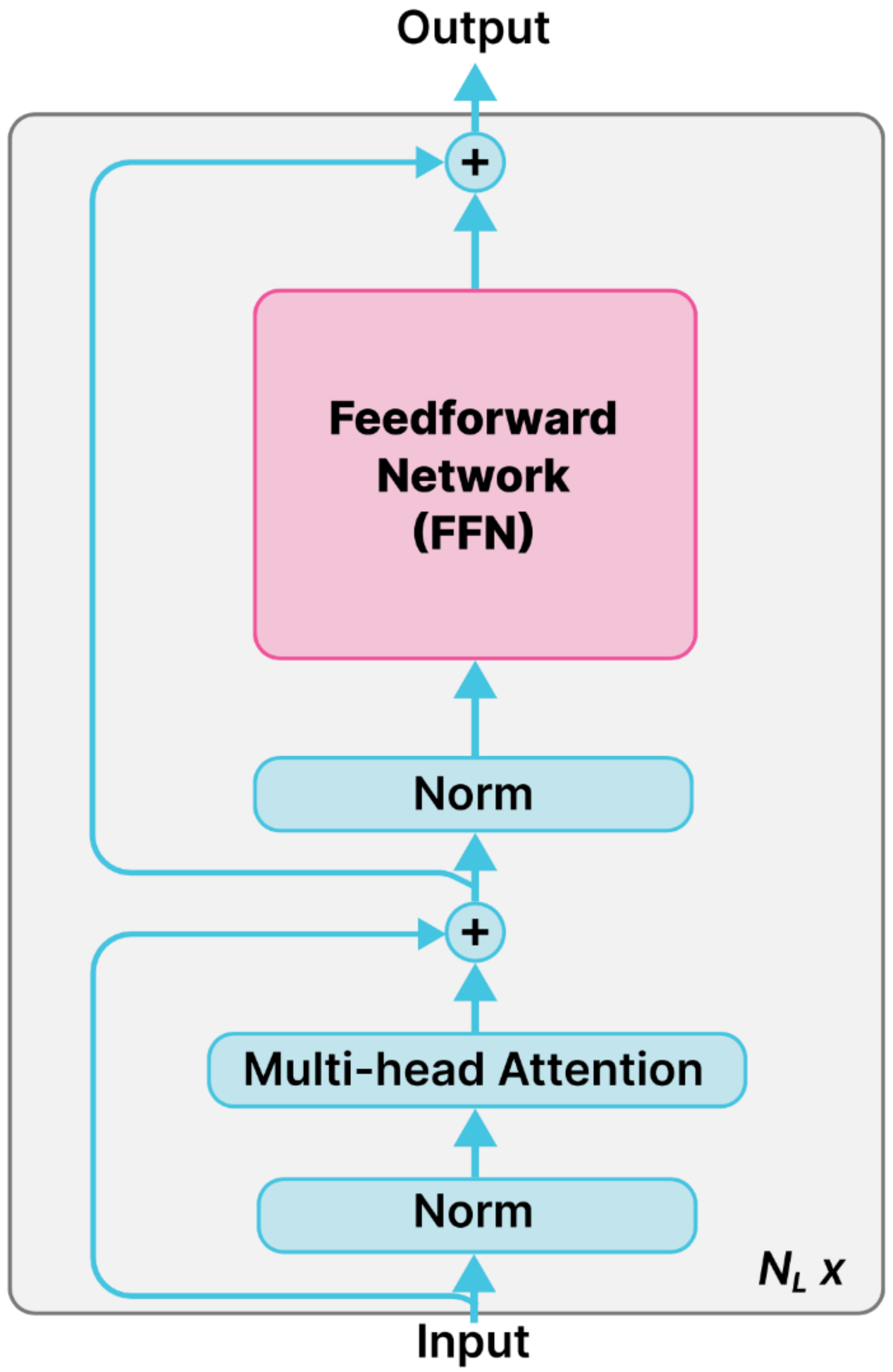
Dmitry Lepikhin lepikhin@google.com
HyoukJoong Lee hyouklee@google.com
Yuanzhong Xu yuanzx@google.com
Dehao Chen dehao@google.com
Orhan Firat orhanf@google.com
Yanping Huang huangyp@google.com
Maxim Krikun krikun@google.com
Noam Shazeer noam@google.com
Zhifeng Chen zhifengc@google.com

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
LIAMFEDUS@GOOGLE.COM
Barret Zoph*
BARRETZOPH@GOOGLE.COM
Noam Shazeer
NOAM@GOOGLE.COM
Google, Mountain View, CA 94043, USA

- Google published influential MoE Transformers papers, e.g., GShard (ICLR 2021) and Switch Transformers (JMLR 2022)
- Extremely infrastructure-heavy — only a handful of organizations could train MoEs
- Major 2023 milestone: Open-source community successfully reproduced MoE training

Mixture-of-Experts 101



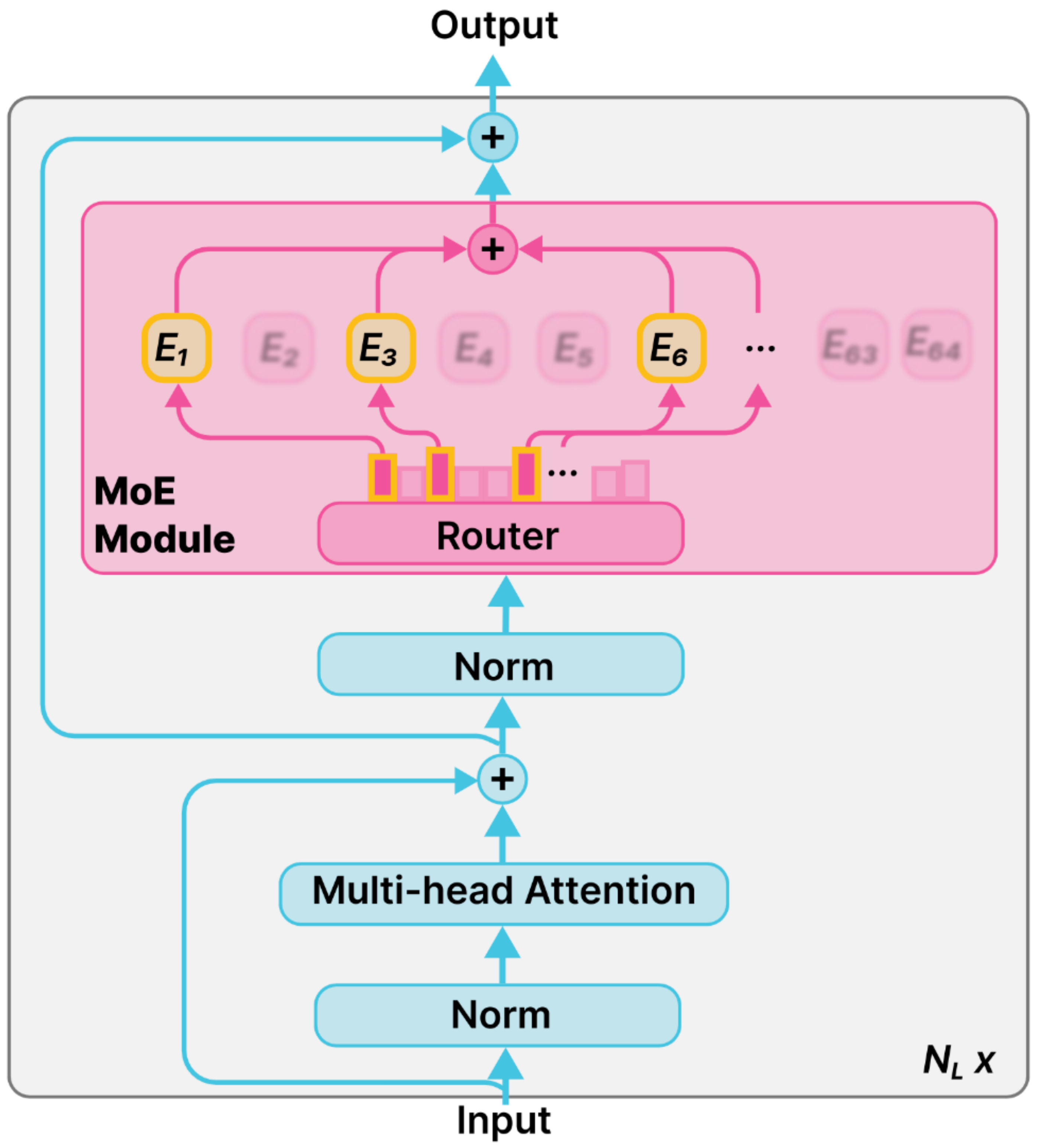
Describing the l-th layer; T: the sequence length; Layer normalization omitted for brevity

$$\mathbf{u}_{1:T}^{l} = \text{Self-Att}\left(\mathbf{h}_{1:T}^{l-1}\right) + \mathbf{h}_{1:T}^{l-1},$$

$$\mathbf{h}_{t}^{l} = \text{FFN}\left(\mathbf{u}_{t}^{l}\right) + \mathbf{u}_{t}^{l},$$

Fig from OLMoE paper, formulas from DeepSeekMoE paper

Mixture-of-Experts 101



Describing the l-th layer; T: the sequence length; Layer normalization omitted for brevity

$$\mathbf{u}_{1:T}^{l} = \text{Self-Att}\left(\mathbf{h}_{1:T}^{l-1}\right) + \mathbf{h}_{1:T}^{l-1},$$

$$\cancel{\mathbf{h}_t^l = \text{FFN}\left(\mathbf{u}_t^l\right) + \mathbf{u}_t^l,}$$

$$s_{i,t} = \text{Softmax}_i\left({\mathbf{u}_t^l}^T \mathbf{e}_i^l\right),$$

← denotes the token-to-expert affinity

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N\}, K), \\ 0, & \text{otherwise}, \end{cases}$$

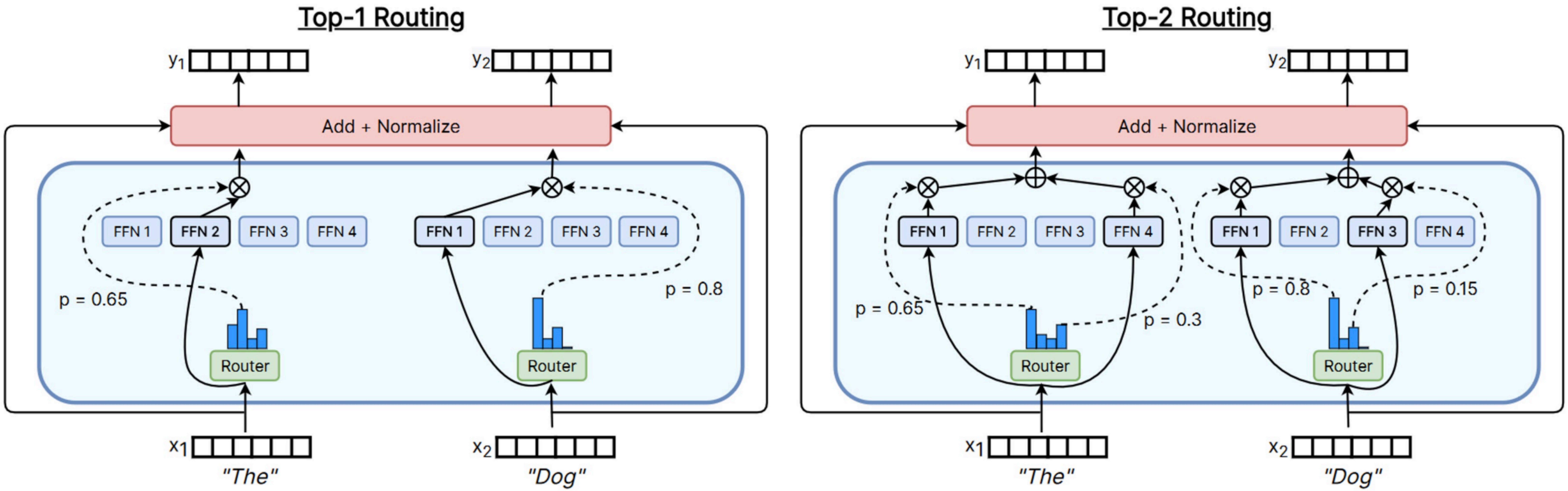
← $\text{Top}k(\cdot, K)$ denotes the set comprising K highest affinity scores (Thus, $g_{i,t}$ is sparse, indicating only K out of N values are non-zero.)

$$\mathbf{h}_t^l = \sum_{i=1}^{N} \left(g_{i,t} \, \text{FFN}_i\left(\mathbf{u}_t^l\right) \right) + \mathbf{u}_t^l,$$

Sparsity ensures computational efficiency, e.g., each token will be assigned to and computed in only K experts

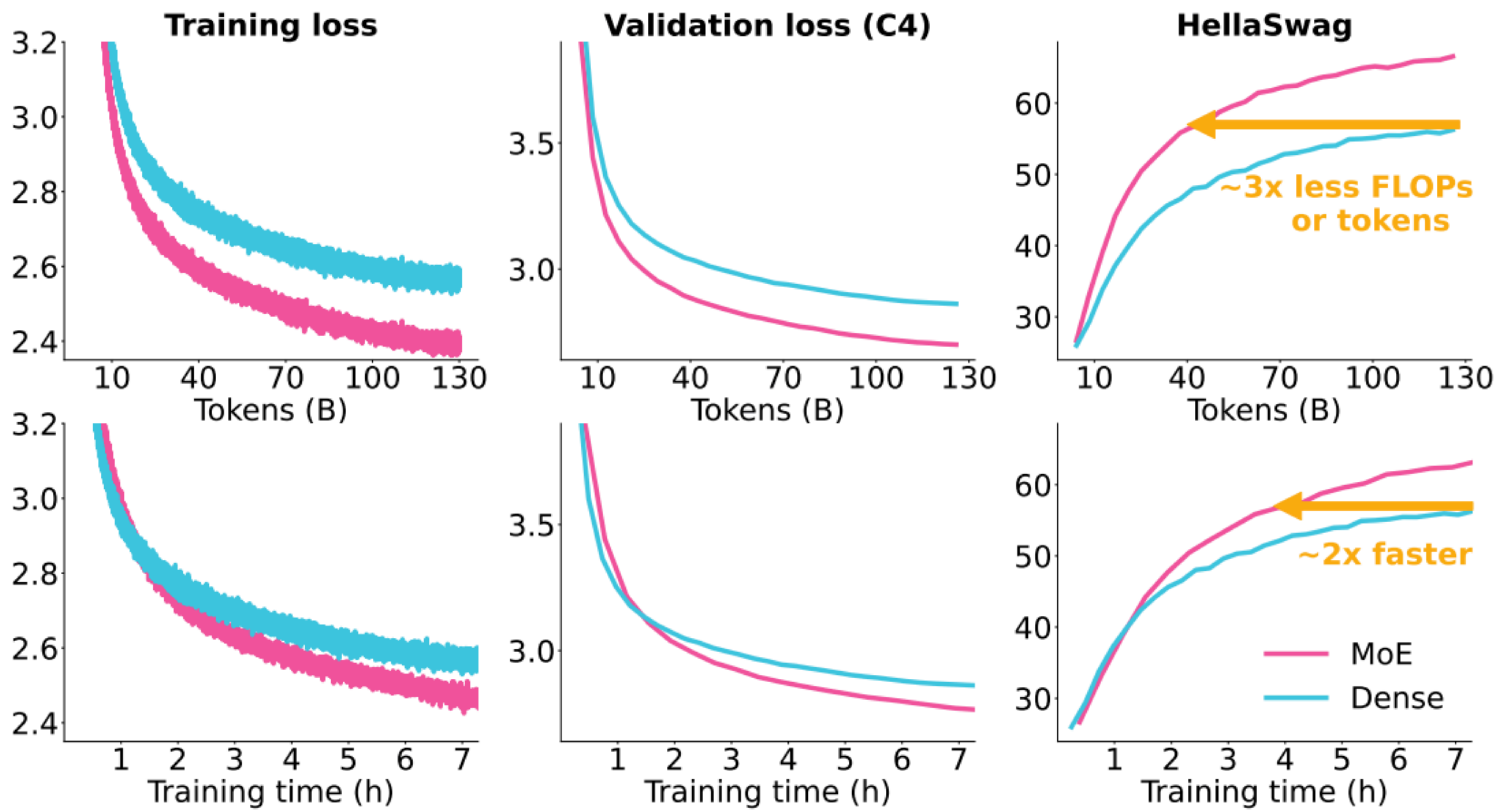
Fig from OLMoE paper, formulas from DeepSeekMoE paper

Mixture-of-Experts 101



Source: ICML 2024 MoE Tutorial

MoE vs. Dense results (1/2)

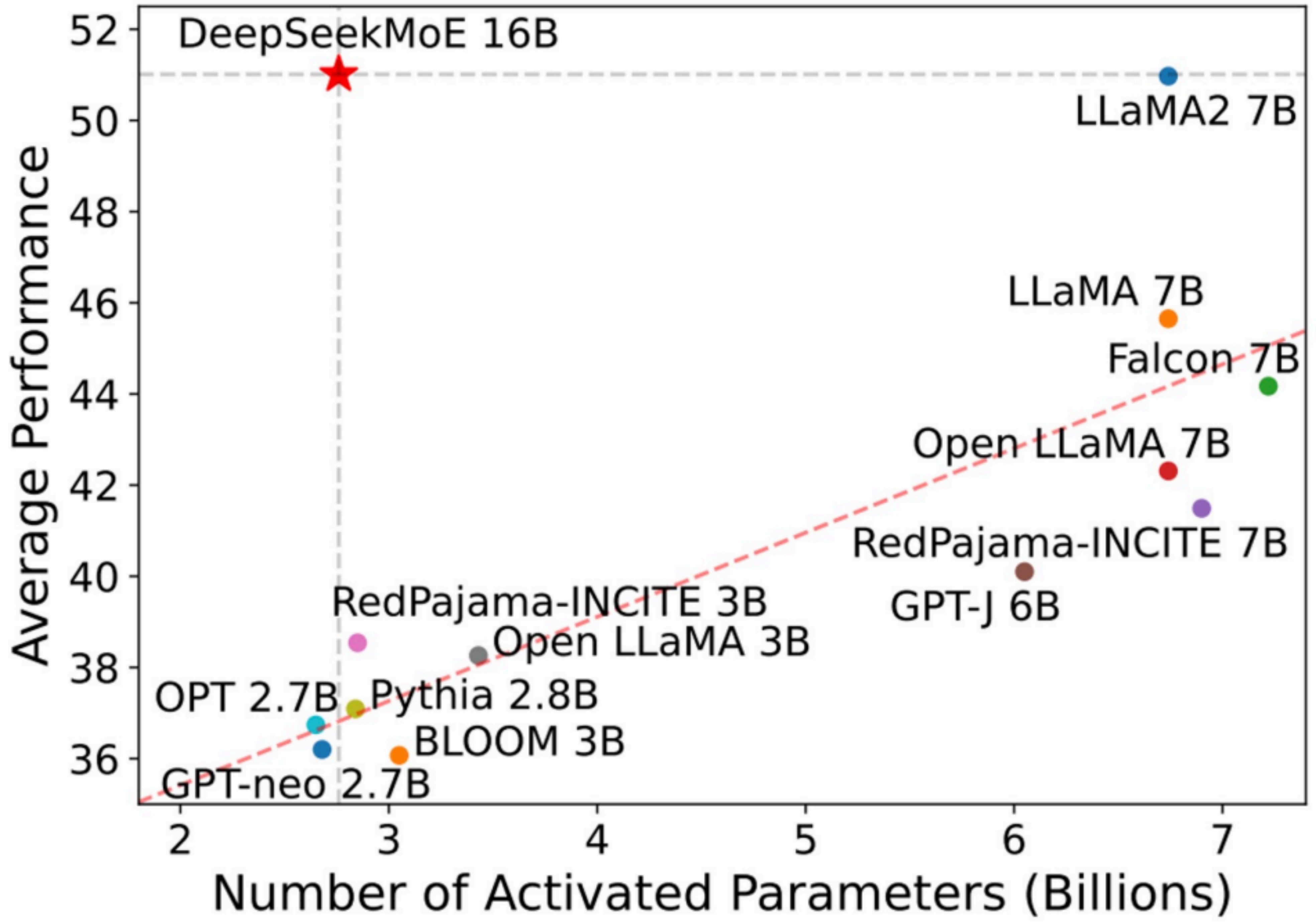


An MoE with 1B active, 7B total parameters is

- Similar cost to a 1B dense model, since typically the number of active experts dominates the cost
- Much more performant than a 1B dense

Source: OLMoE paper

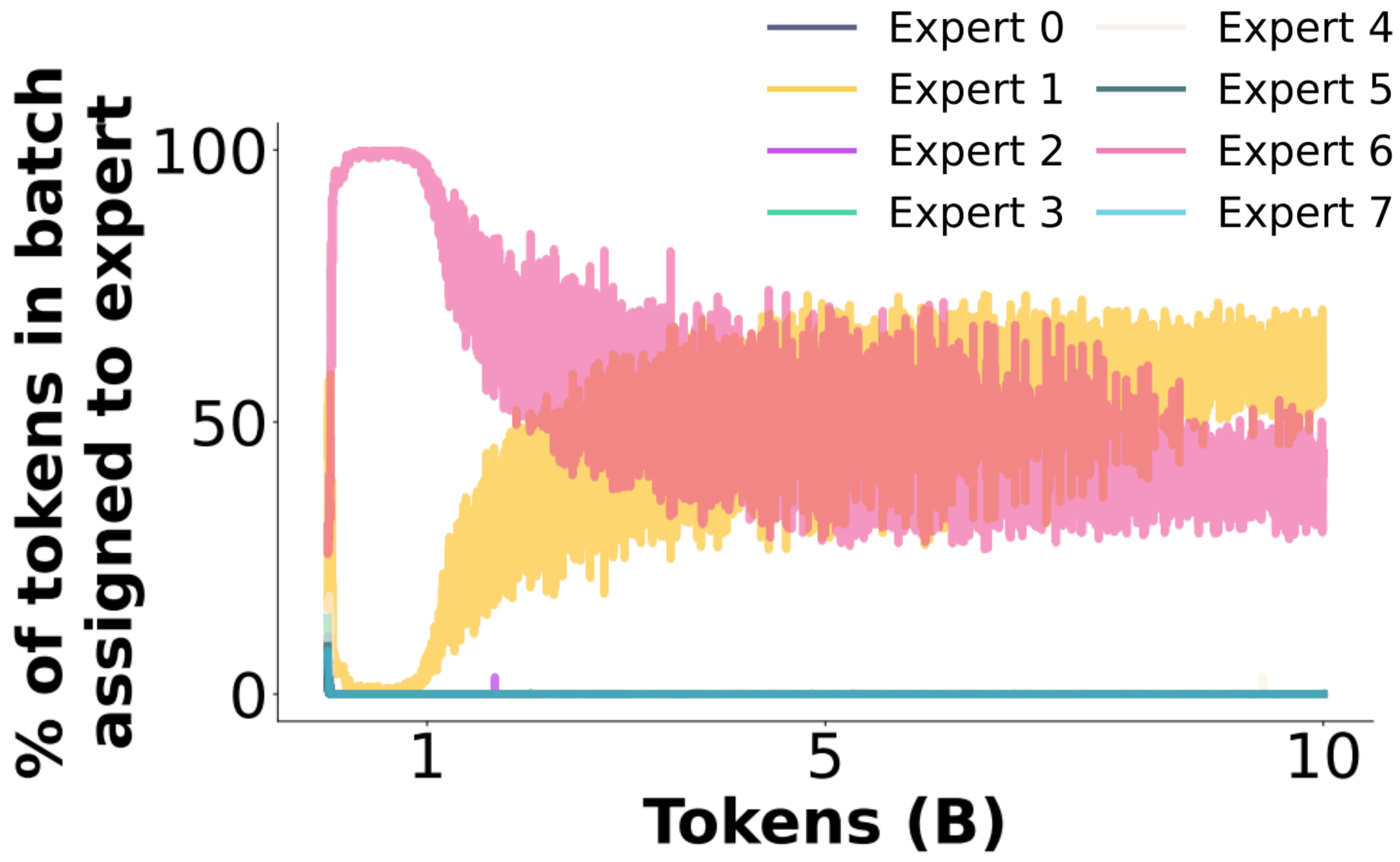
MoE vs. Dense results (2/2)



Metric	# Shot	DeepSeek 7B (Dense)	DeepSeekMoE 16B
# Total Params	N/A	6.9B	16.4B
# Activated Params	N/A	6.9B	2.8B
FLOPs per 4K Tokens	N/A	183.5T	74.4T
# Training Tokens	N/A	2T	2T
Pile (BPB)	N/A	0.75	**0.74**
HellaSwag (Acc.)	0-shot	75.4	**77.1**
PIQA (Acc.)	0-shot	79.2	**80.2**
ARC-easy (Acc.)	0-shot	**67.9**	**68.1**
ARC-challenge (Acc.)	0-shot	48.1	**49.8**
RACE-middle (Acc.)	5-shot	**63.2**	61.9
RACE-high (Acc.)	5-shot	**46.5**	**46.4**
DROP (EM)	1-shot	**34.9**	32.9
GSM8K (EM)	8-shot	17.4	**18.8**
MATH (EM)	4-shot	3.3	**4.3**
HumanEval (Pass@1)	0-shot	26.2	**26.8**
MBPP (Pass@1)	3-shot	**39.0**	**39.2**
TriviaQA (EM)	5-shot	59.7	**64.8**
NaturalQuestions (EM)	5-shot	22.2	**25.5**
MMLU (Acc.)	5-shot	**48.2**	45.0
WinoGrande (Acc.)	0-shot	**70.5**	**70.2**
CLUEWSC (EM)	5-shot	**73.1**	72.1
CEval (Acc.)	5-shot	**45.0**	40.6
CMMLU (Acc.)	5-shot	**47.2**	42.5
CHID (Acc.)	0-shot	**89.3**	**89.4**

Source: DeepSeekMoE paper

Problem: Load Balancing



Expert-Level Balance Loss. In order to mitigate the risk of routing collapse, we also employ an expert-level balance loss. The computation of the balance loss is as follows:

$$\mathcal{L}_{\mathrm{ExpBal}} = \alpha_1 \sum_{i=1}^{N'} f_i P_i, \tag{12}$$

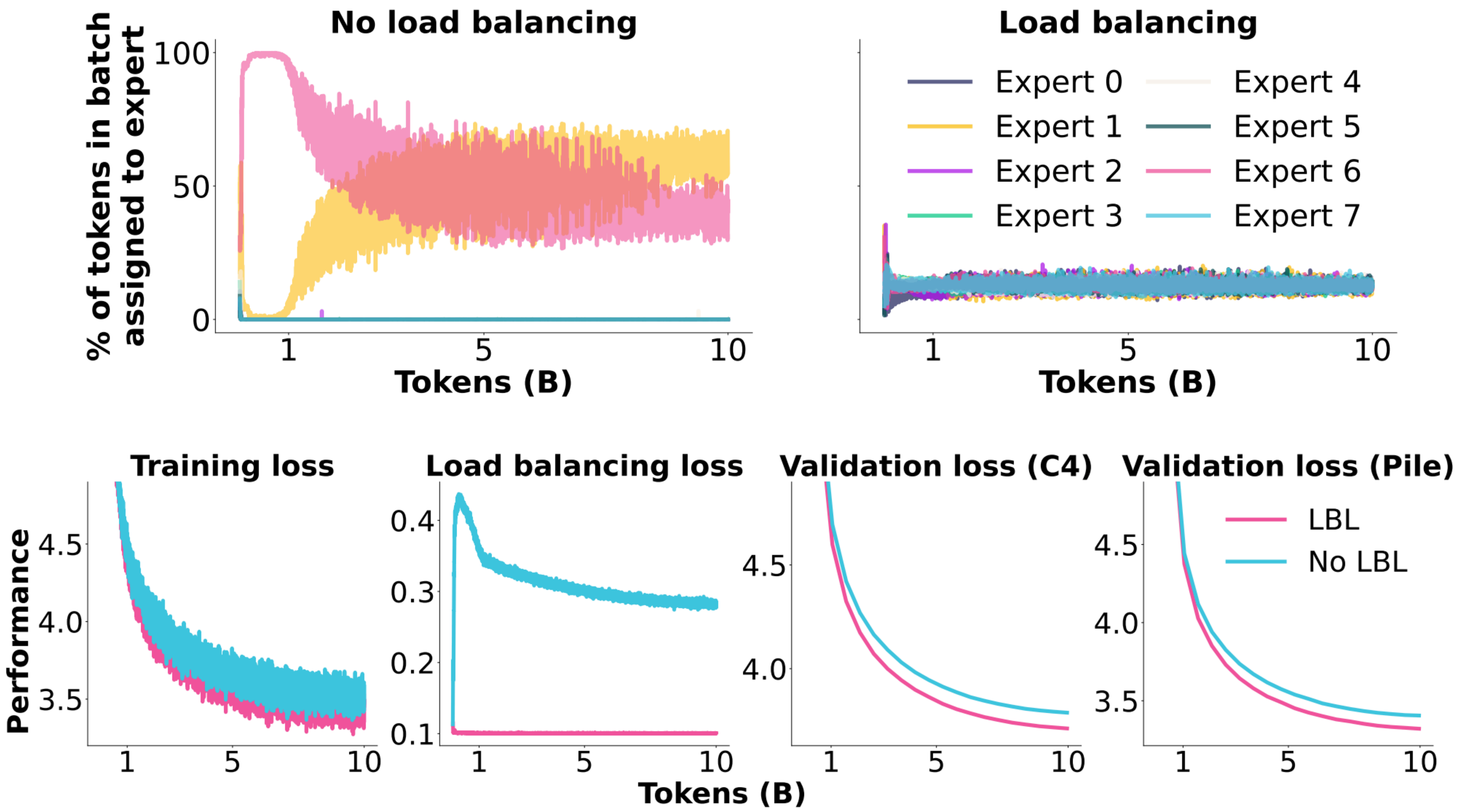
$$f_i = \frac{N'}{K'T} \sum_{t=1}^{T} \mathbb{1}(\text{Token } t \text{ selects Expert } i), \tag{13}$$

$$P_i = \frac{1}{T} \sum_{t=1}^{T} s_{i,t}, \tag{14}$$

where α_1 is a hyper-parameter called expert-level balance factor, N' is equal to $(mN - K_s)$ and K' is equal to $(mK - K_s)$ for brevity. $\mathbb{1}(\cdot)$ denotes the indicator function.

Fig from OLMoE paper, formulas from DeepSeekMoE paper

After adding a load balancing loss



Source: OLMoE paper

Different ways of ensuring LB

Auxiliary-Loss-Free Load Balancing. For MoE models, an unbalanced expert load will lead to routing collapse (Shazeer et al., 2017) and diminish computational efficiency in scenarios with expert parallelism. Conventional solutions usually rely on the auxiliary loss (Fedus et al., 2021; Lepikhin et al., 2021) to avoid unbalanced load. However, too large an auxiliary loss will impair the model performance (Wang et al., 2024a). To achieve a better trade-off between load balance and model performance, we pioneer an auxiliary-loss-free load balancing strategy (Wang et al., 2024a) to ensure load balance. To be specific, we introduce a bias term b_i for each expert and add it to the corresponding affinity scores $s_{i,t}$ to determine the top-K routing:

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \operatorname{Topk}(\{s_{j,t} + b_j | 1 \leqslant j \leqslant N_r\}, K_r), \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

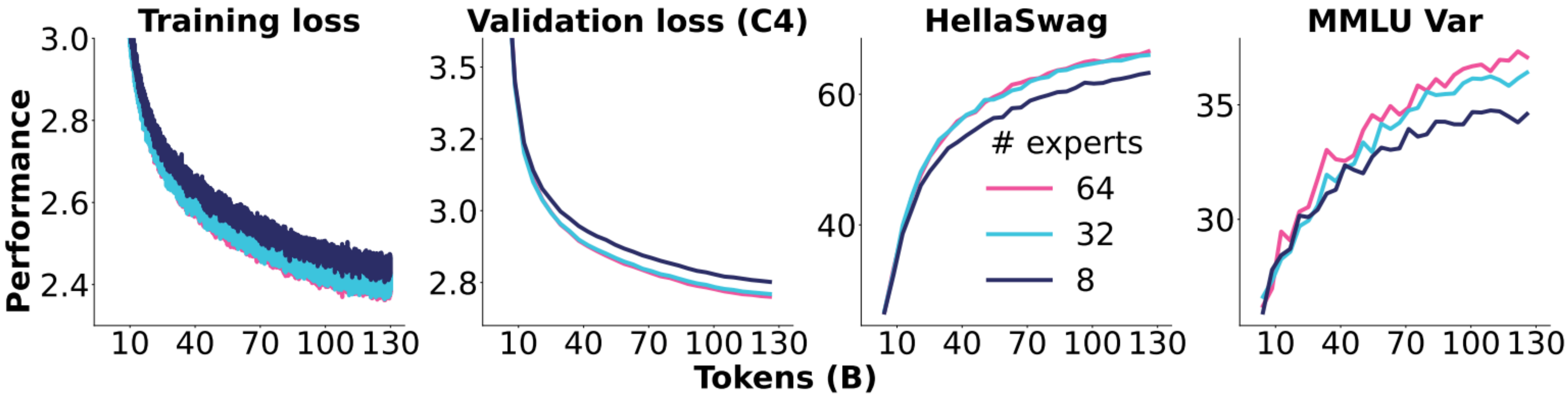
Note that the bias term is only used for routing. The gating value, which will be multiplied with the FFN output, is still derived from the original affinity score $s_{i,t}$. During training, we keep monitoring the expert load on the whole batch of each training step. At the end of each step, we will decrease the bias term by γ if its corresponding expert is overloaded, and increase it by γ if its corresponding expert is underloaded, where γ is a hyper-parameter called bias update speed. Through the dynamic adjustment, DeepSeek-V3 keeps balanced expert load during training, and achieves better performance than models that encourage load balance through pure auxiliary losses.

TL;DR: DeepSeek V3 came up with a technique that ensures LB without an auxiliary loss

Source: DeepSeek V3 Technical Report

MoE design choice 1: Expert granularity

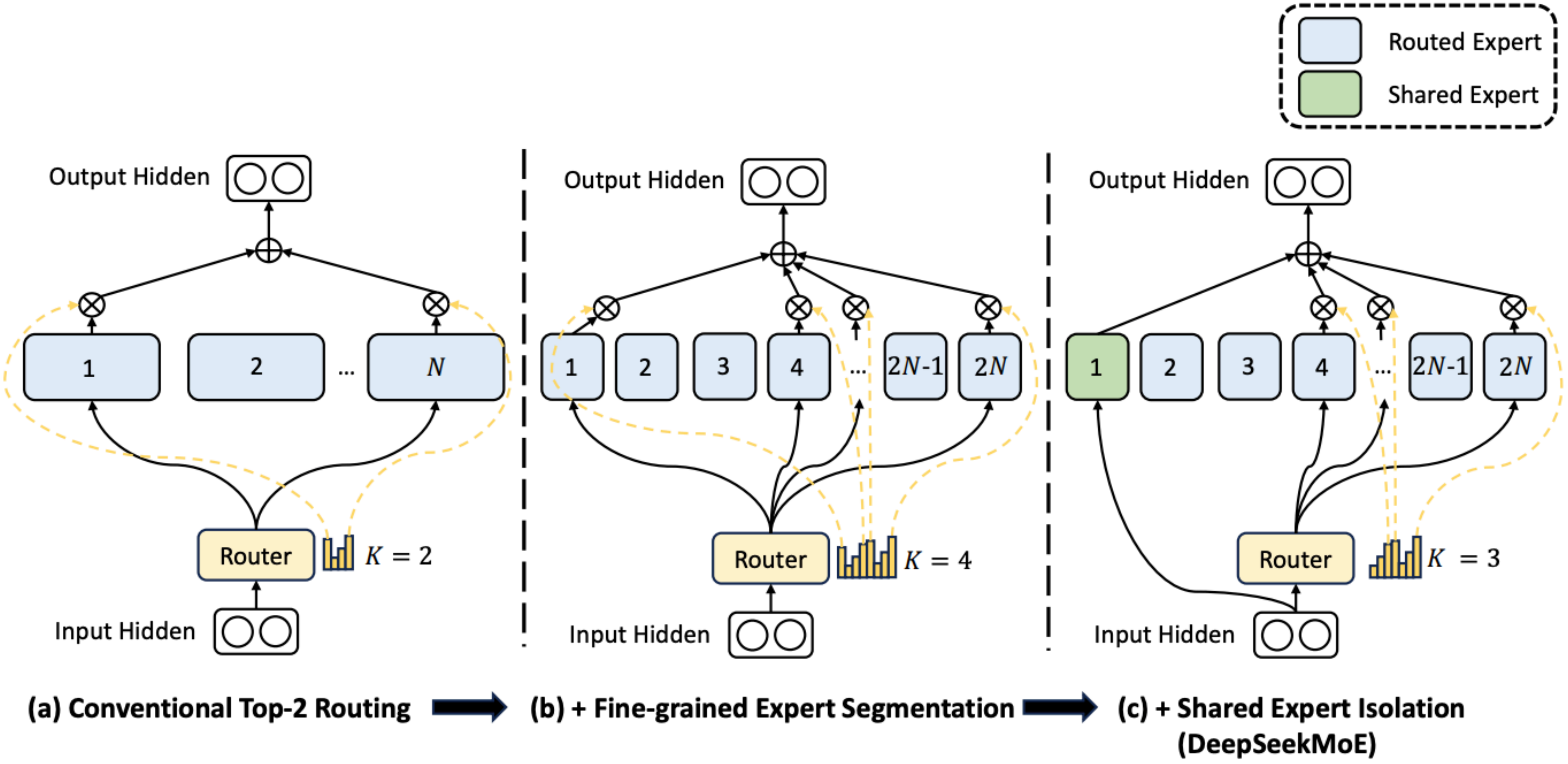
- Early MoEs mostly used coarse-grained MoEs (e.g., activating 2 out of 8 experts)
- Nowadays, most MoEs are fine-grained, e.g., activating 4 or 8 out of 64 experts, with DeepSeekMoE one of the earlier works



Source: OLMoE paper

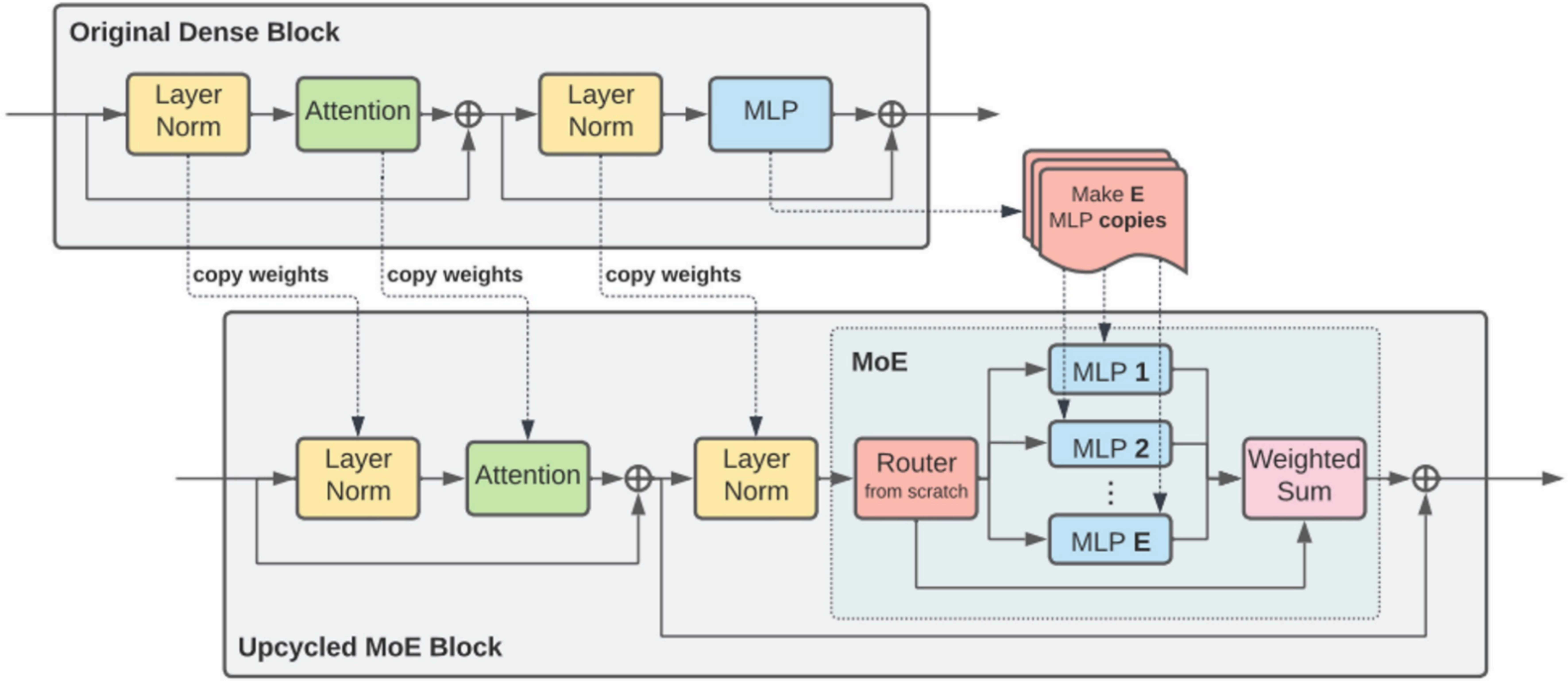
MoE design choice 2: Shared experts

- Early MoEs did not have shared experts
- Nowadays, most MoEs have shared experts, with DeepSeekMoE one of the earlier works



Source: DeepSeekMoE paper

MoE design choice 3: Sparse upcycling



Source: ICML 2024 MoE Tutorial

MoE design choice 3: Sparse upcycling

Example 1： Mixtral 8×22B(7B) （April, 2024）

Total 141B parameters, 39B activate parameters, (8 experts and 2 experts are selected)

CC-BY-NC license

Model	Active parameters	Common sense and reasoning					Knowledge	
		MMLU	HellaS	WinoG	Arc C (5)	Arc C (25)	TriQA	NaturalQS
LLaMA 2 70B	*70B*	69.9%	87.1%	83.2%	86.0%	85.1%	77.57%	35.5%
Command R	*35B*	68.2%	87.0%	81.5%	-	66.5%	-	-
Command R+	*104B*	75.7%	**88.6%**	**85.4%**	-	71.0%	-	-
Mistral 7B	*7B*	62.47%	83.1%	78.0%	77.2%	78.1%	68.8%	28.1%
Mixtral 8x7B	*12.9B*	70.63%	86.6%	81.2%	85.8%	85.9%	78.4%	36.5%
Mixtral 8x22B	*39B*	**77.75%**	**88.5%**	84.7%	**91.3%**	**91.3%**	**82.2%**	**40.1%**

Less widely used these days

Source: ICML 2024 MoE Tutorial

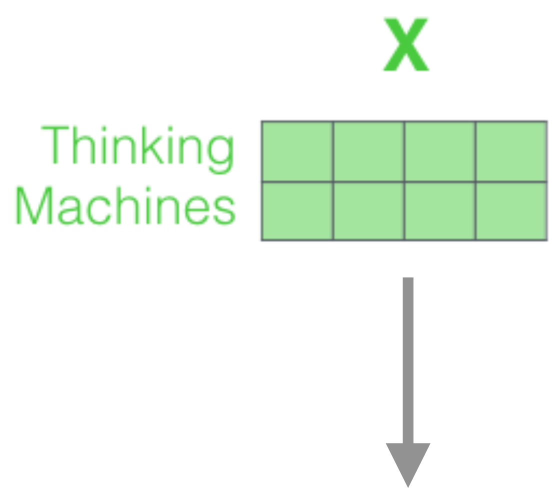
Case studies

	Parameters (Active/Total)	Experts (Active/Total)	Other info
DeepSeekMoE (January 2024)	2.8B Active, 16.4B Total	8 Active, 66 Total (2 Shared + 6 out of 64)	First layer is dense
OLMoE (September 2024)	1.3B Active, 6.9B Total	8 Active, 64 Total (No shared expert)	
DeepSeek-V3 (December 2024)	37B Active, 671B Total	9 Active, 257 Total (1 Shared + 8 out of 256)	First 3 layers are dense
GPT-OSS (August 2025)	GPT-OSS 120B: 5.1B Active, 120B Total GPT-OSS 20B: 3.6B Active, 20B Total	4 Active, 128 Total (No shared expert) 4 Active, 32 Total (No shared expert)	
Qwen3-235B-A22B (August 2025)	22B Active, 235B Total	8 Active, 128 Total (No shared expert)	
Nemotron-3 (December 2025)	3.2B to 3.5B Active, 30B Total	5 to 6 Active, 130 Total (2 Shared + 4 out of 128)	Hybrid Mamba-Transformer
GLM-5 (February 2026)	40B to 44B Active, 744B Total	8 Active, 256 Total (1 Shared expert)	First 3 layers are dense

Attention Variants

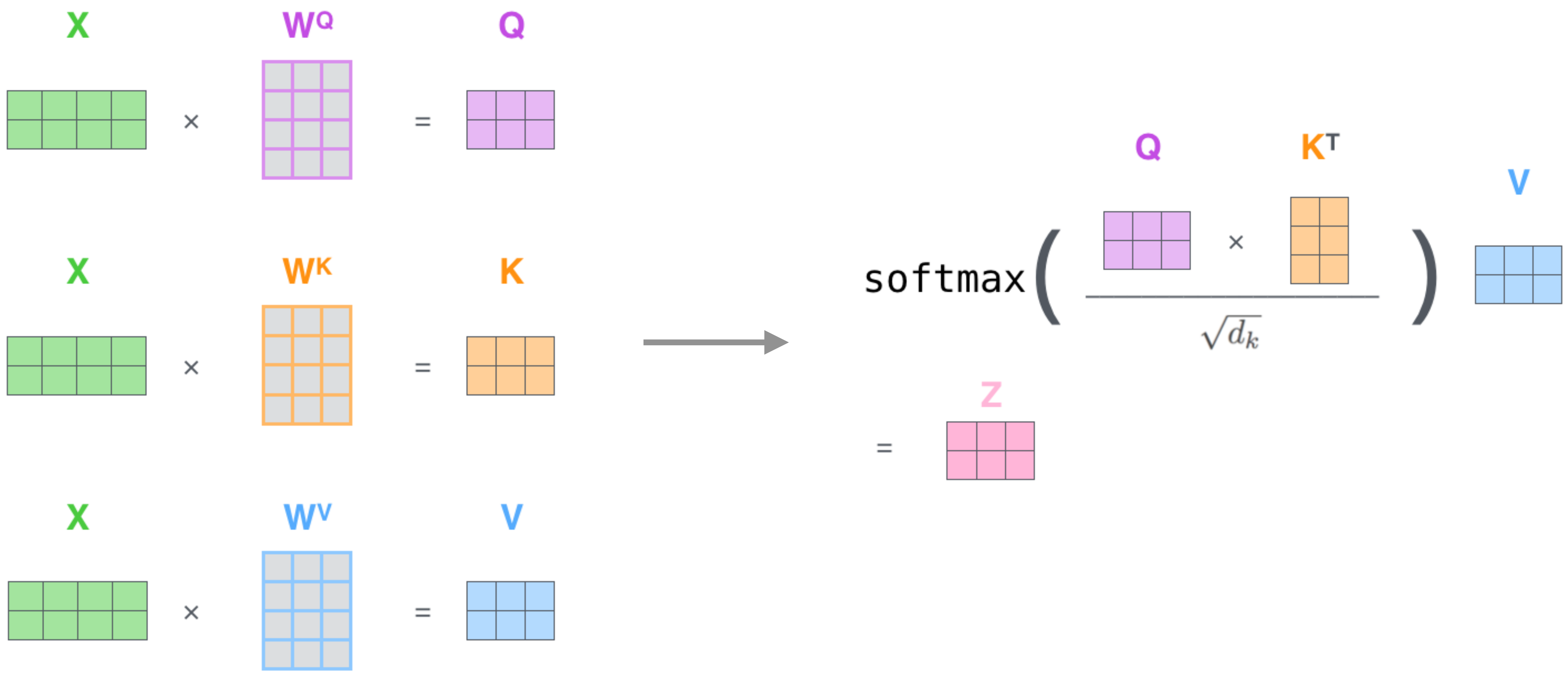
Recap: Self-attention

Goal: map a sequence of input vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^{d_1}$ to a sequence of n vectors $\mathbf{z}_1, \cdots, \mathbf{z}_n \in \mathbb{R}^{d_2}$



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d}}\right)V$$

(scales quadratically with the sequence length (n), because Q and K are $\mathbb{R}^{n \times d}$)



Recap: Self-attention

Goal: map a sequence of input vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^{d_1}$ to a sequence of n vectors $\mathbf{z}_1, \cdots, \mathbf{z}_n \in \mathbb{R}^{d_2}$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d}}\right)V$$

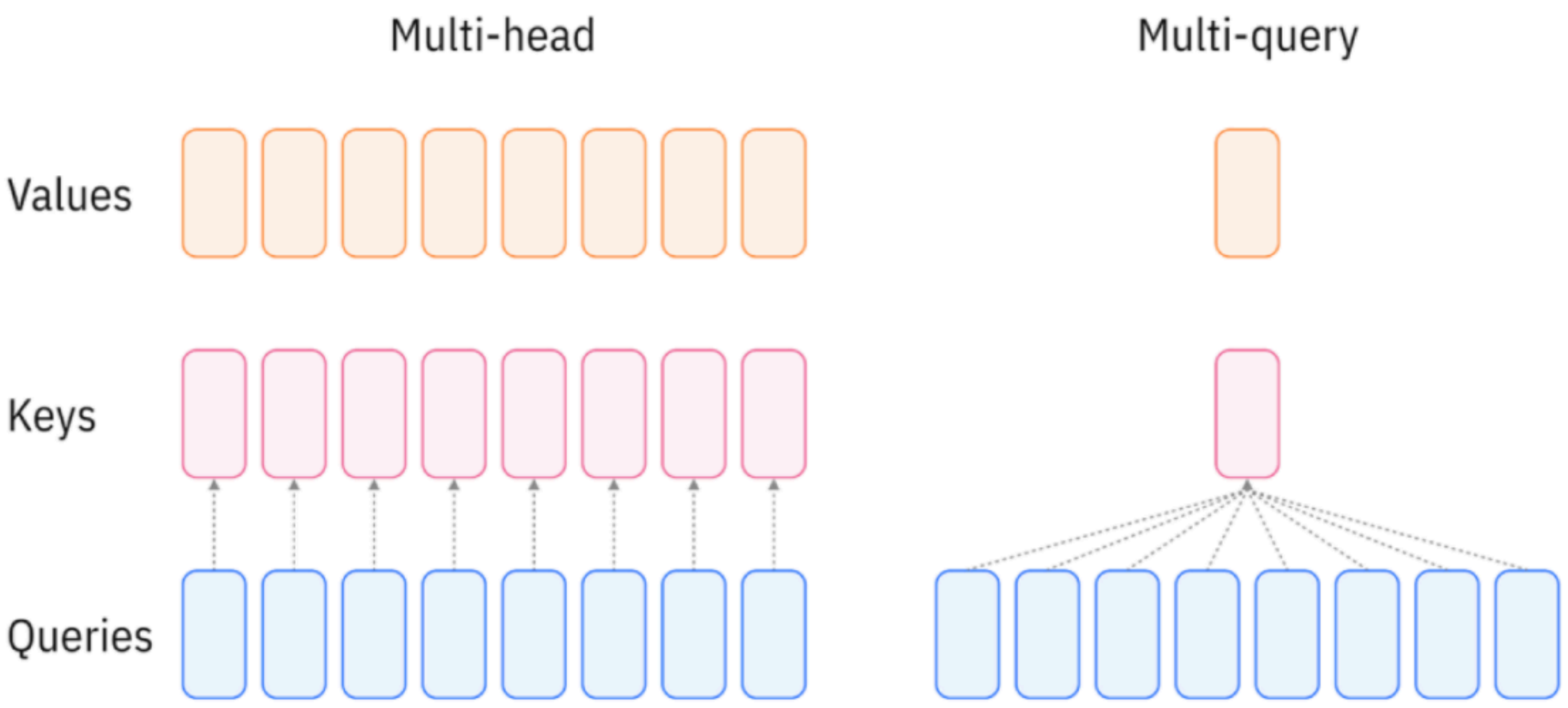
(scales quadratically with the sequence length (n), because Q and K are $\mathbb{R}^{n \times d}$)

KV caching (Key-Value caching)

- When generating a new token, the model must attend to all previous tokens, thus need to recompute K and V for all previous tokens every step
- Let's store previously computed Keys and Values in GPU memory!
- Compute K and V only for the new token; reuse cached vectors for previous tokens → Much faster generation
- Tradeoff: Turns the system memory-bound
  - Cache size grows linearly with sequence length
  - Becomes a major GPU memory bottleneck

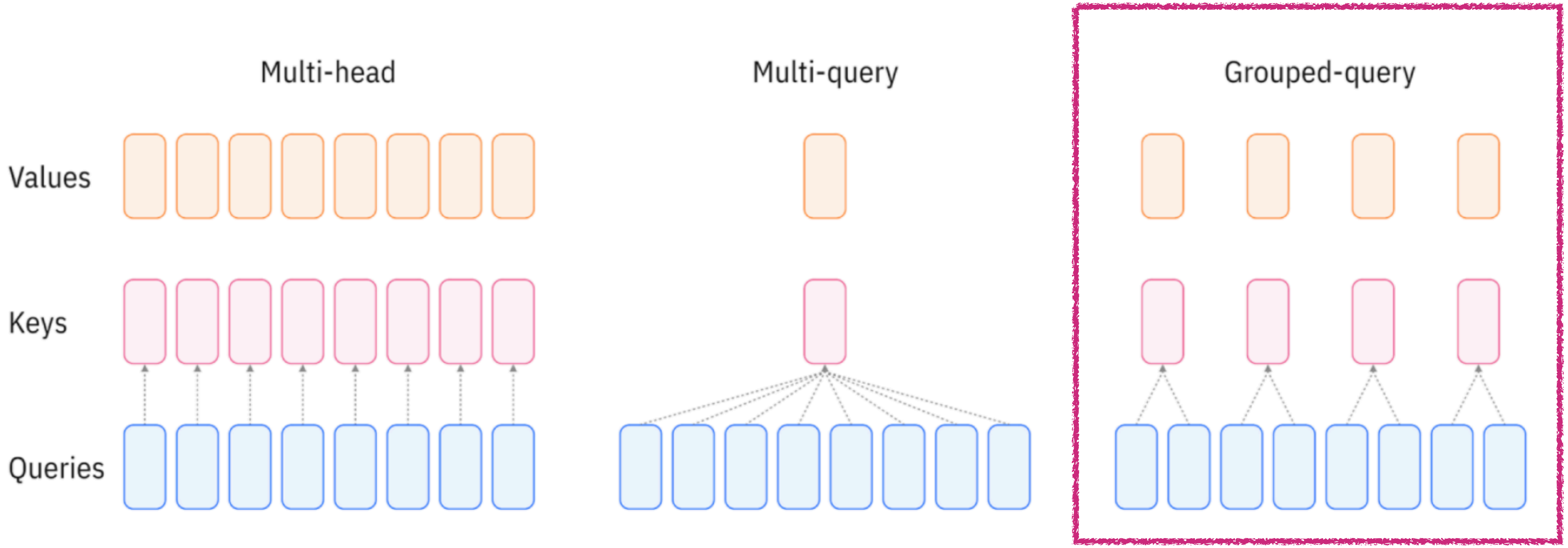
→ Motivation for efficient attention

Recap: Multi-Query Attention



Shazeer. 2019. "Fast Transformer Decoding: One Write-Head is All You Need"

Recap: Grouped-Query Attention



- Shares key and value heads for each ***group*** of query heads
- Saves memory, which leads to faster inference
- Popularized by **Llama 2/3**

Shazeer. 2019. "Fast Transformer Decoding: One Write-Head is All You Need"

Multi-Head Latent (MLA) Attention

- Used in DeepSeek V2, V3, and R1, offering a different memory-saving strategy
- Instead of sharing key and value heads like GQA, MLA compresses the key and value tensors into a lower-dimensional space before storing them in the KV cache
- At inference time, these compressed tensors are projected back to their original size before being used. This adds an extra matrix multiplication but reduces memory usage.

Benchmark (Metric)	# Shots	Small MoE w/ MHA	Small MoE w/ MLA	Large MoE w/ MHA	Large MoE w/ MLA
# Activated Params	-	2.5B	2.4B	25.0B	21.5B
# Total Params	-	15.8B	15.7B	250.8B	247.4B
KV Cache per Token (# Element)	-	110.6K	15.6K	860.2K	34.6K
BBH (EM)	3-shot	37.9	**39.0**	46.6	**50.7**
MMLU (Acc.)	5-shot	48.7	**50.0**	57.5	**59.0**
C-Eval (Acc.)	5-shot	**51.6**	50.9	57.9	**59.2**
CMMLU (Acc.)	5-shot	52.3	**53.4**	60.7	**62.5**

Table 9 | Comparison between MLA and MHA on hard benchmarks. DeepSeek-V2 shows better performance than MHA, but requires a significantly smaller amount of KV cache.

The memory requirements for MLA are much lower than for MHA

MLA performs better than MHA (here tested on Mixture-of-Experts architectures)

Accorinding to DeepSeek V2

- MQA and GQA underperform MHA (contradicting with some previous work)
- MLA is on par or better than MHA

Results from DeepSeek V2
Image source: https://sebastianraschka.com/blog/

Multi-Head Latent (MLA) Attention: Results

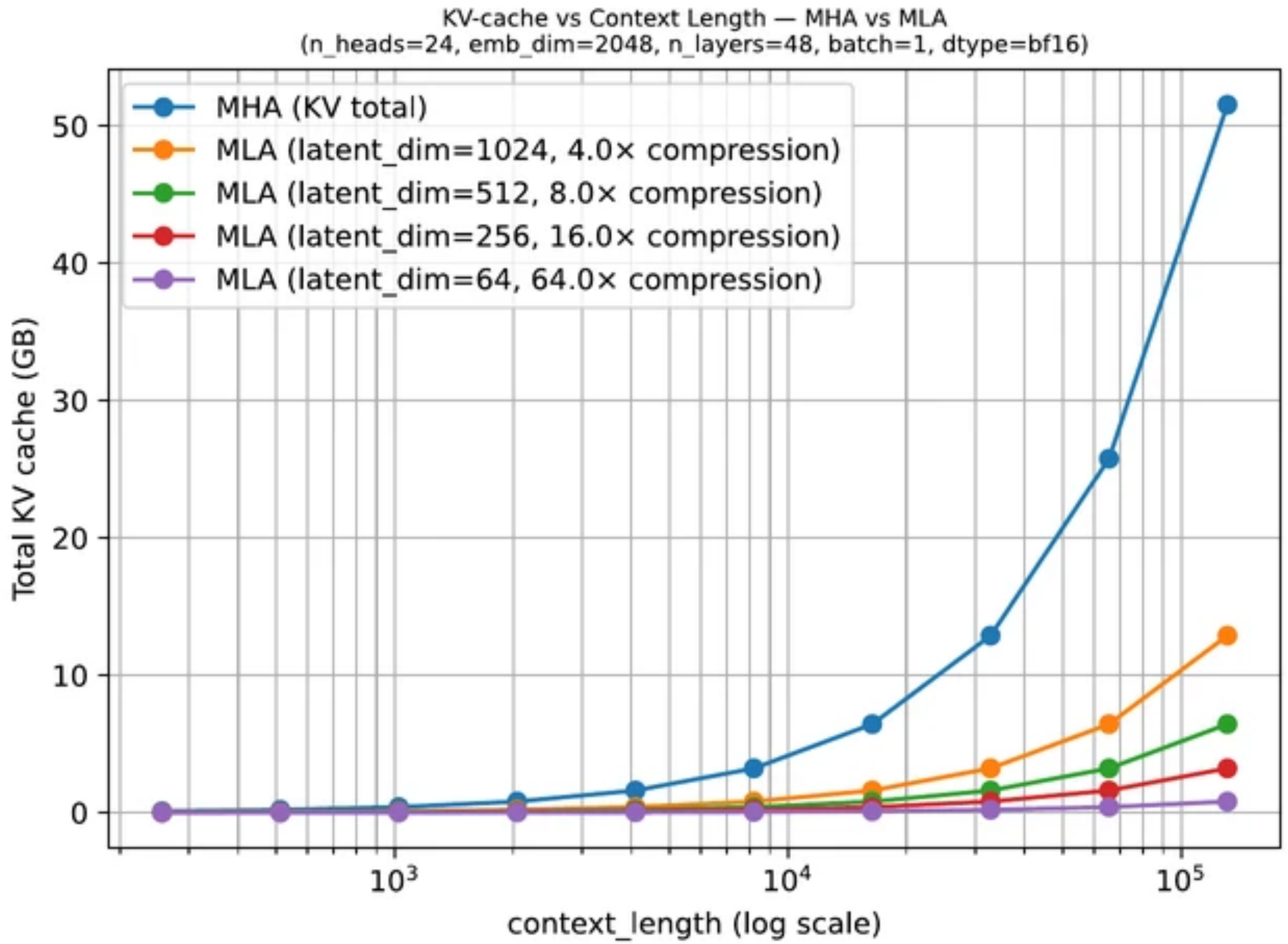


Image source: https://sebastianraschka.com/blog/

Sliding window attention

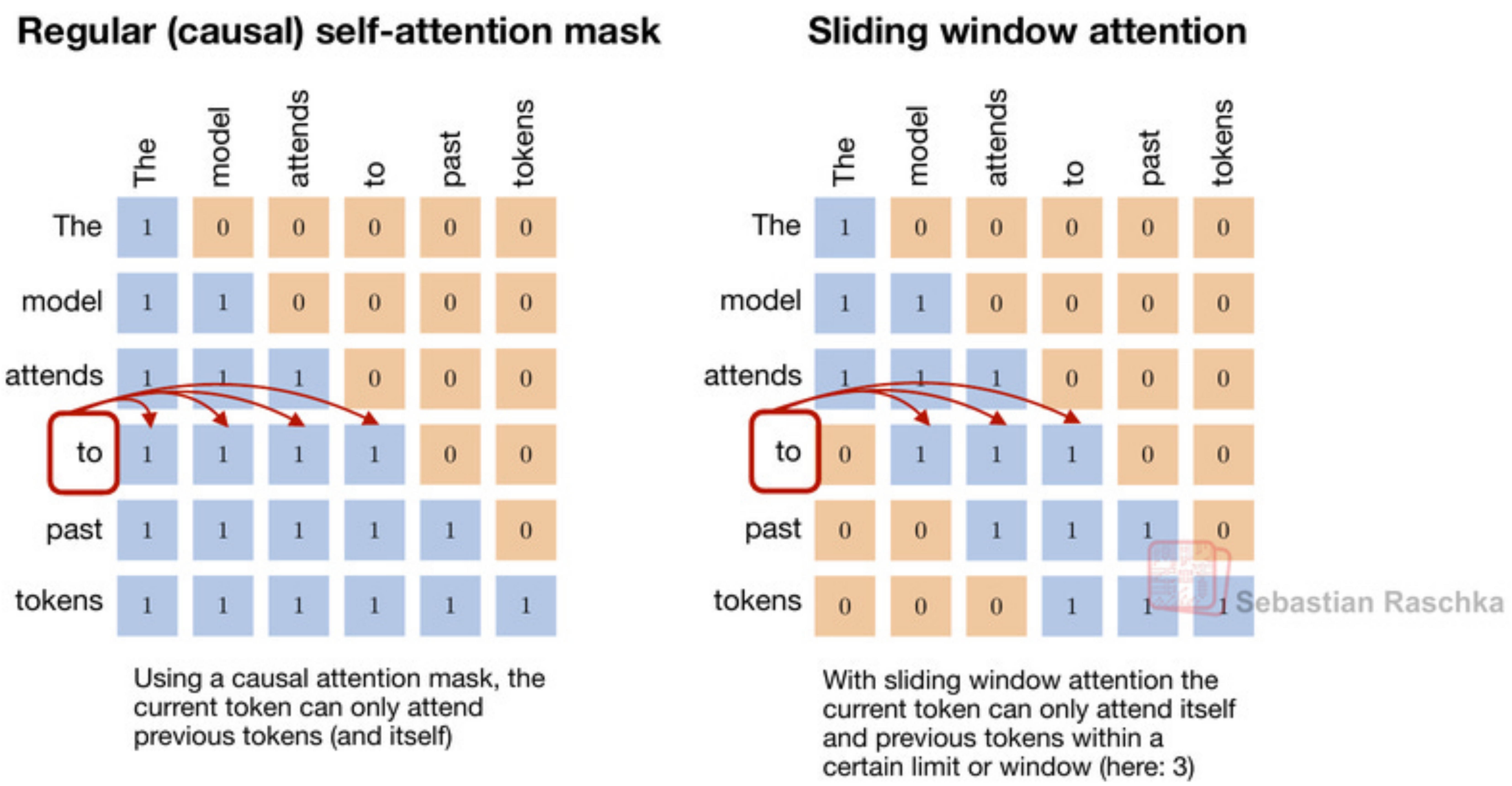
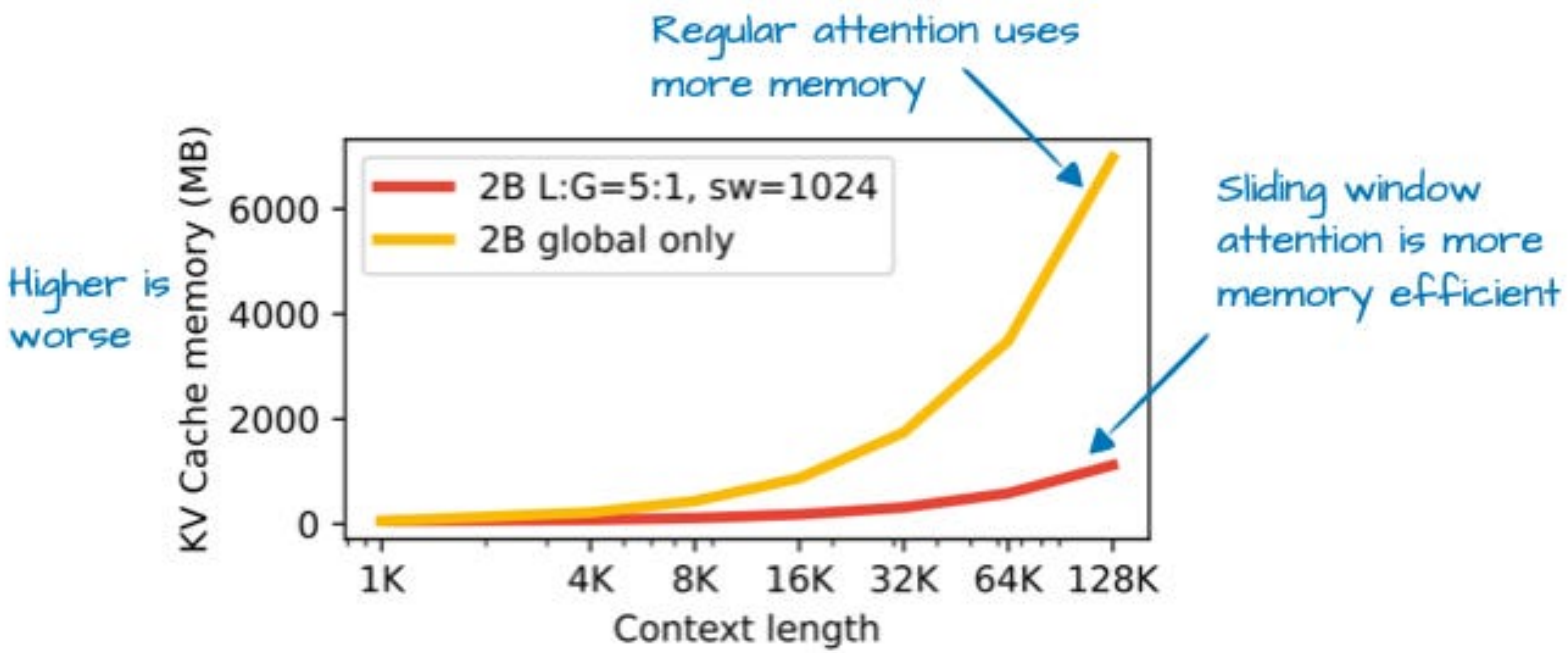


Image source: https://sebastianraschka.com/blog/

Sliding window attention results

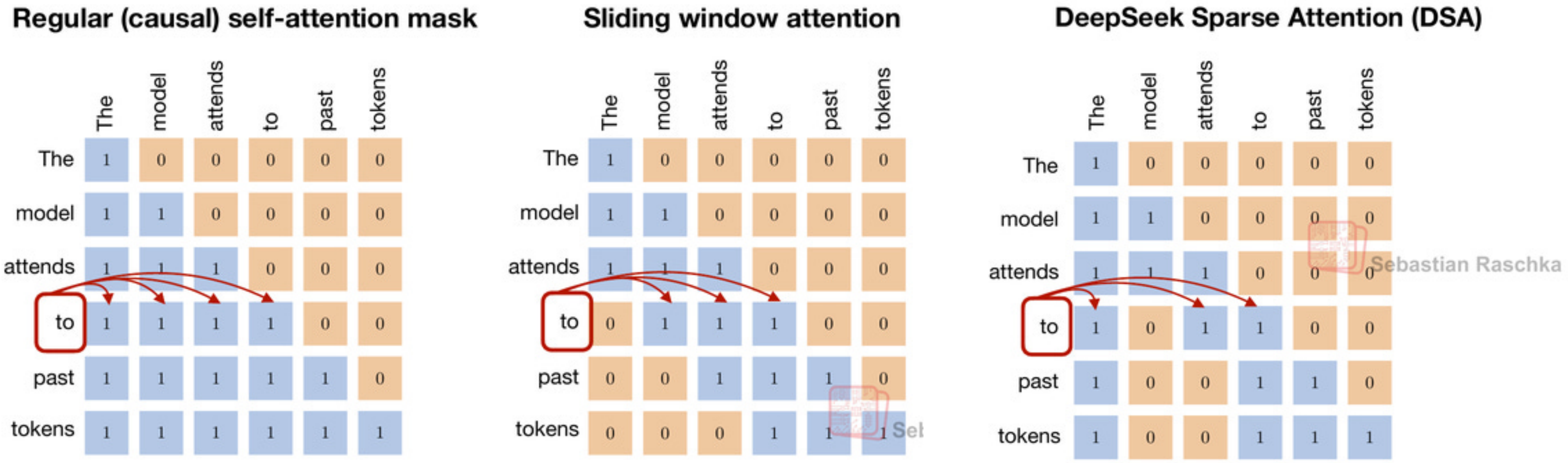


- First introduced by LongFormer (Ai2, 2020), and used in many flagship models, e.g., Mistral 7B (2023), BigBird (Google, 2020), Gemma 1/2/3 (Google, 2024-25), Qwen 3 (2025)
- However, likely fundamentally limited in long-range dependencies

Image source: https://sebastianraschka.com/blog/

Sparse Attention

- Key idea: Instead of a fixed window, attend only to a top-K subset
- Idea existed since 2019; most popularized by DeepSeek Sparse Attention (DSA)



- The Lightning Indexer: Performs a very fast, low-precision check (FP8, d=32, fewer heads) to see which tokens are actually relevant to the current query
- Main Attention: Once the indexer picks the top-K tokens (e.g., 2,048 out of 128,000), performs the "real" attention, using the full-rank & high-precision computation

Can We Use Different Attention Mechanisms?:
Linear Attention Hybrids

What's linear attention?

RNNs: Process text sequentially by maintaining a compressed hidden state $h_t = f(h_{t-1} + x_t)$.

- O(1) inference per token, but suffering from gradient vanishing, no parallelize training.

Transformers: Softmax-based Self-Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d}}\right)V$$

- Parallelization and direct modeling of long-range dependencies, but O(N^2) complexity.

Linear attention: Attempts to recover the efficiency of RNNs by bypassing the Softmax bottleneck, effectively treating the attention as a linear map that can be computed in O(N) time.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\intercal}{\sqrt{d}}\right)V \approx \phi(Q)\big(\phi(K)^\intercal V\big)$$

- Here, ϕ is a kernel feature function.
- This lets attention be computed like RNN: $S_t = S_{t-1} + \phi(k_t)v_t^\intercal$ then $y_t = \phi(q_t)^\intercal S_t$

Linear attention: Background

- Linear attention variants have been around for a long time
  - Example: "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention"
- (Arguably) Didn't gain traction until 2025, as they typically degraded the model accuracy
- Since fall 2025, variants of linear attention *hybrids* started to get attention
  - Examples: MiniMax-M1, Qwen3-Next, Kimi Linear

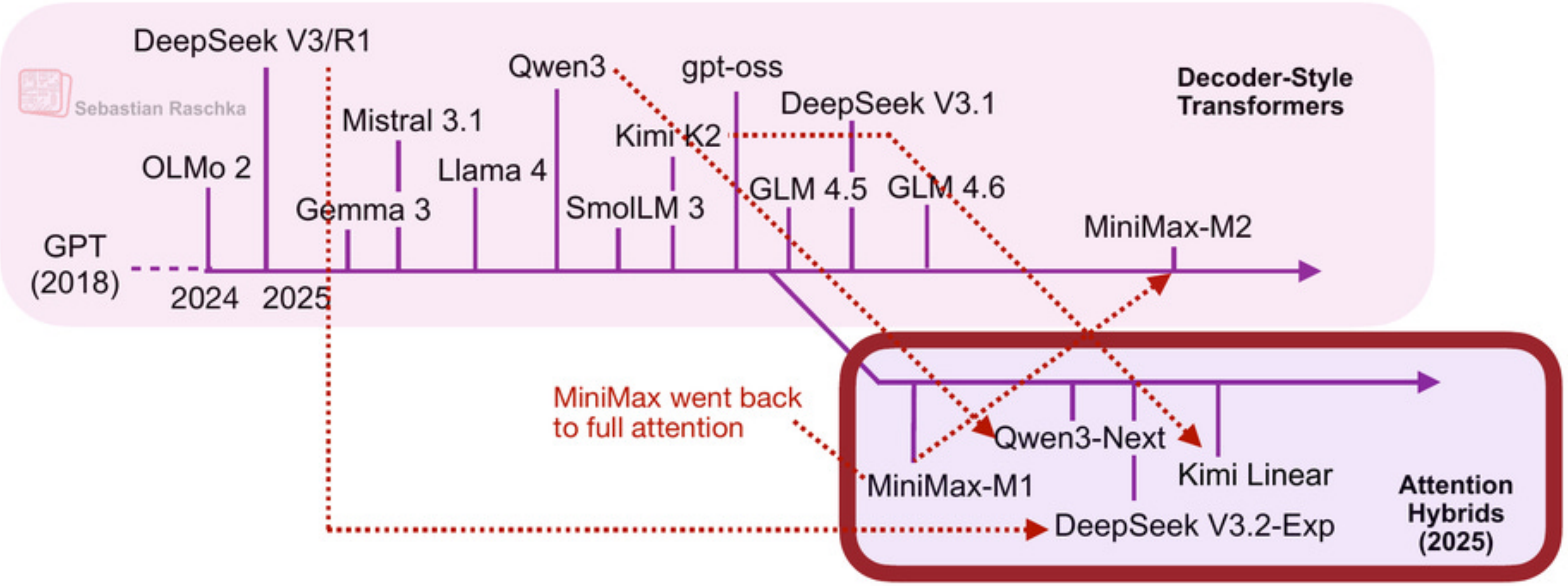
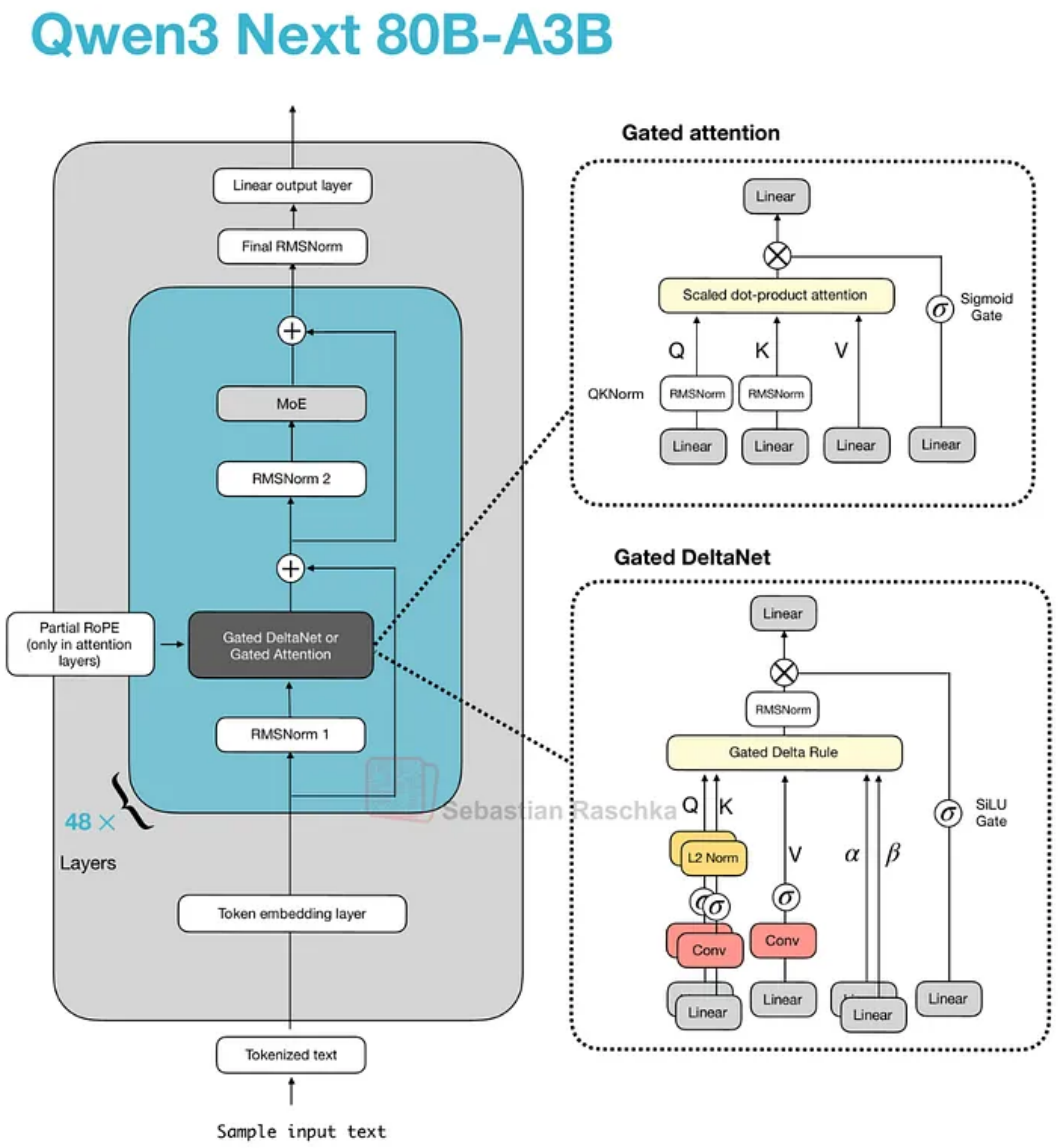


Image source: https://sebastianraschka.com/blog/

Case study: Qwen3-Next 80B-A3B (September 2025)



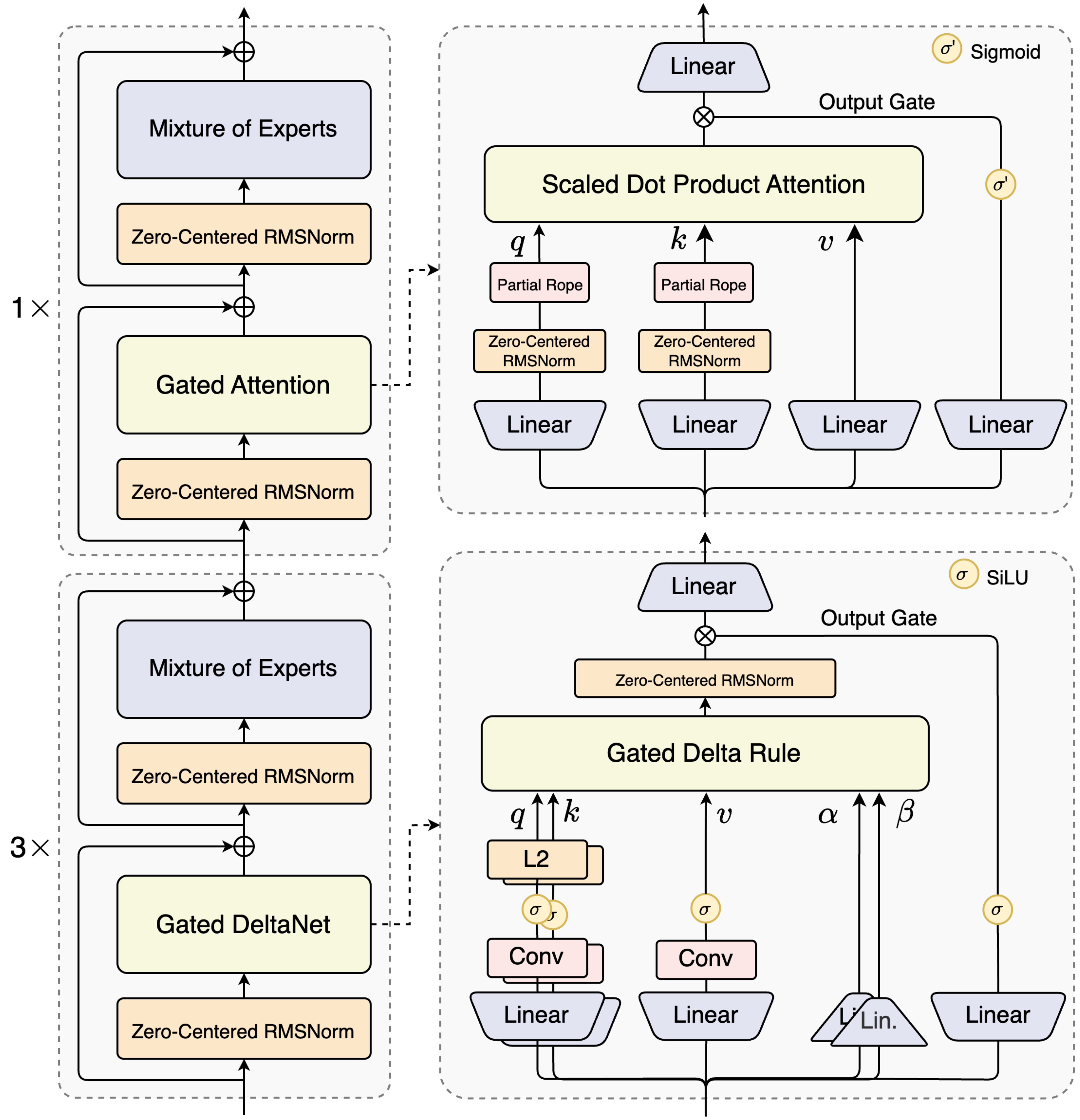
- An **MoE** model (3B Active, 80B Total)
- Replaced the regular attention mechanism by a **Gated DeltaNet** + **Gated Attention** hybrid (**3**:**1** Ratio)

```
Layer 1   : Linear attention → MoE
Layer 2   : Linear attention → MoE
Layer 3   : Linear attention → MoE
Layer 4   : Full attention   → MoE

Layer 5   : Linear attention → MoE
Layer 6   : Linear attention → MoE
Layer 7   : Linear attention → MoE
Layer 8   : Full attention   → MoE

...
```

- Enabled the native 262k token context length

Image source: https://sebastianraschka.com/blog/

Qwen3-Next 80B-A3B (1): Gated Attention



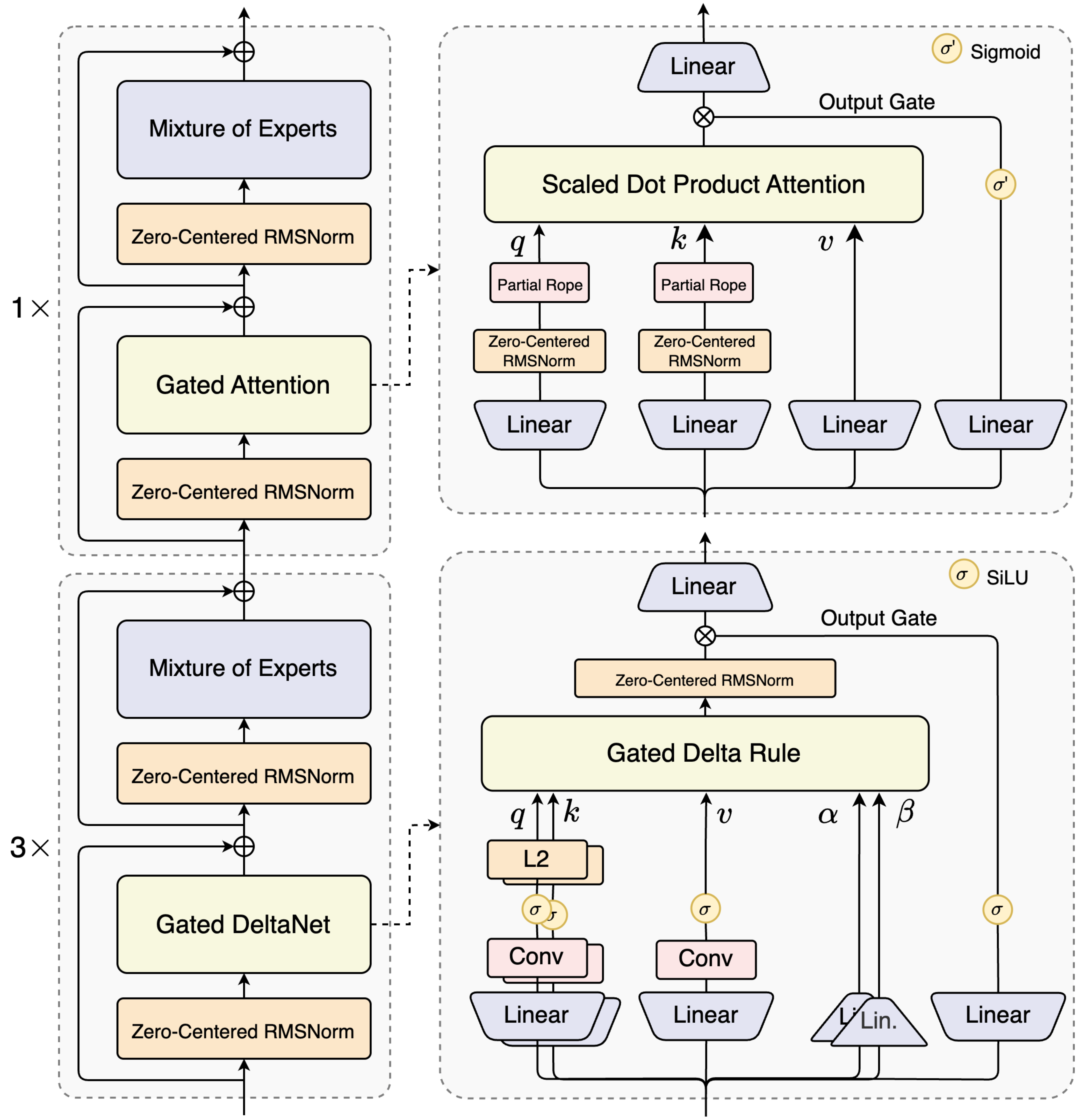
- Gated Attention: essentially regular full attention with an additional sigmoid gate.
  - "[…] the attention output gating mechanism helps eliminate issues like Attention Sink and Massive Activation, ensuring numerical stability across the model."


Gated Attention for Large Language Models: Non-linearity, Sparsity, and Attention-Sink-Free

Zihan Qiu*[1], Zekun Wang*[1], Bo Zheng*[1], Zeyu Huang*[2], Kaiyue Wen[3], Songlin Yang[4], Rui Men[1], Le Yu[1], Fei Huang[1], Suozhi Huang[5], Dayiheng Liu[✉1], Jingren Zhou[1], Junyang Lin[✉1]

[1]Qwen Team, Alibaba Group [2]University of Edinburgh [3]Stanford University [4]MIT [5]Tsinghua University

Qwen3-Next 80B-A3B (1): Gated DeltaNet



- Gated DeltaNet replaces the attention mechanism itself with a recurrent delta-rule memory update.
  - DeltaNet: Linear-time alternative to transformer attention
    - Maintains a shared memory state updated at each token, using the delta rule from online learning
    - Originally proposed as an improved version of Mamba/state-space models (an alternative to transformers— a separate big topic)
  - Adding a gating mechanism for adaptive memory control

GATED DELTA NETWORKS:
IMPROVING MAMBA2 WITH DELTA RULE

Songlin Yang *
MIT CSAIL
yangsl66@mit.edu

Jan Kautz
NVIDIA
jkautz@nvidia.com

Ali Hatamizadeh *
NVIDIA
ahatamizadeh@nvidia.com

Memory consumption comparison

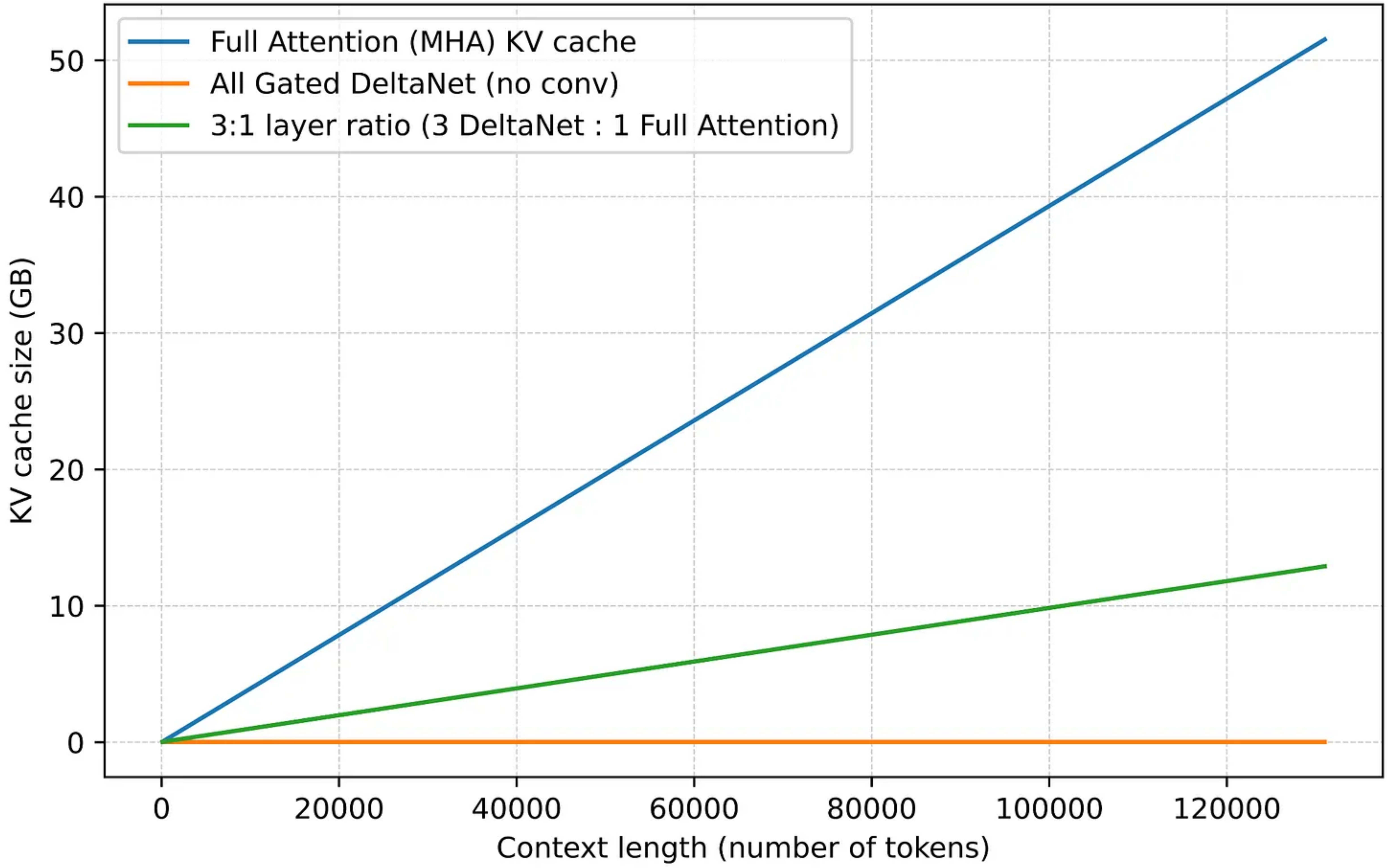
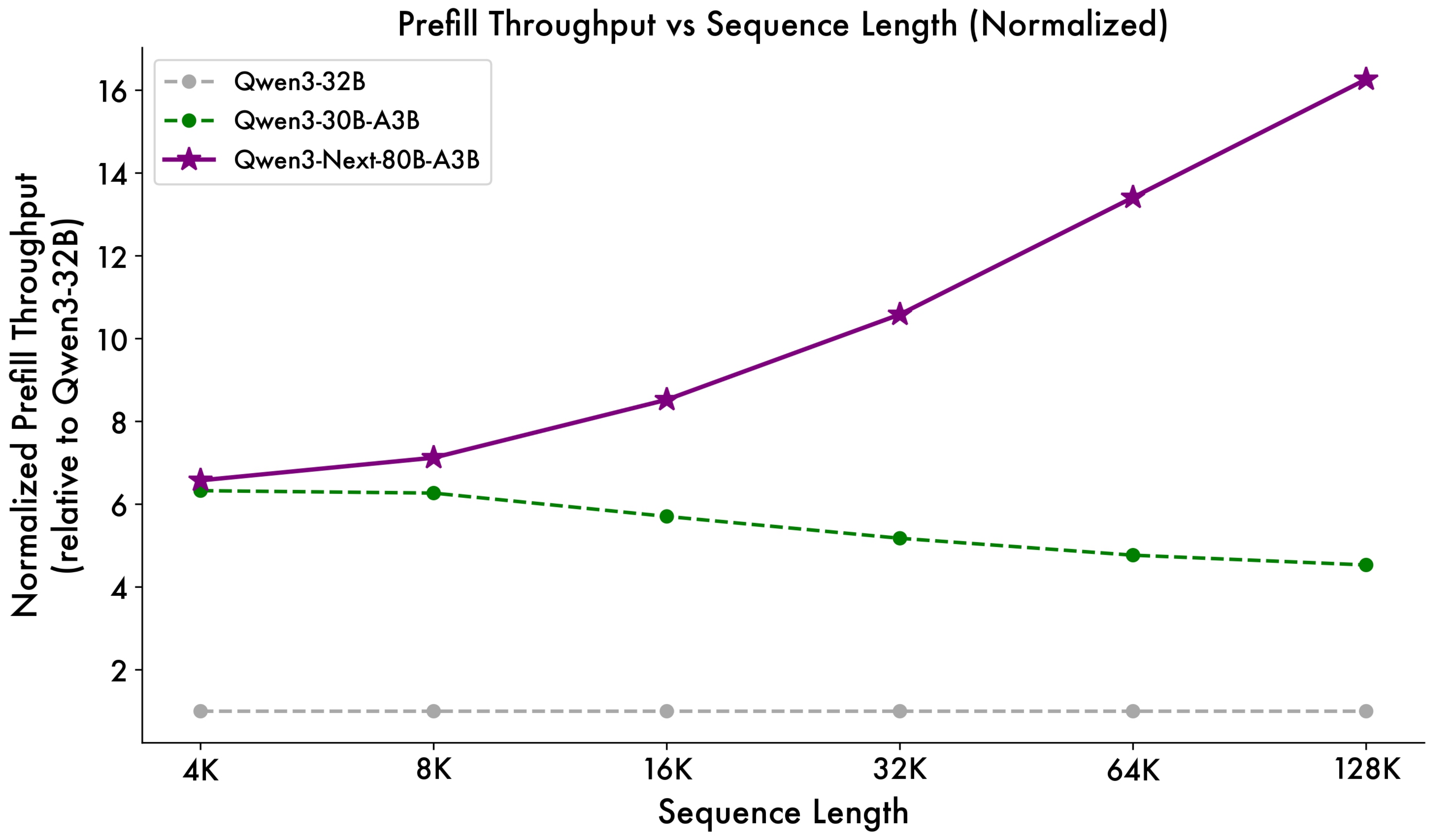
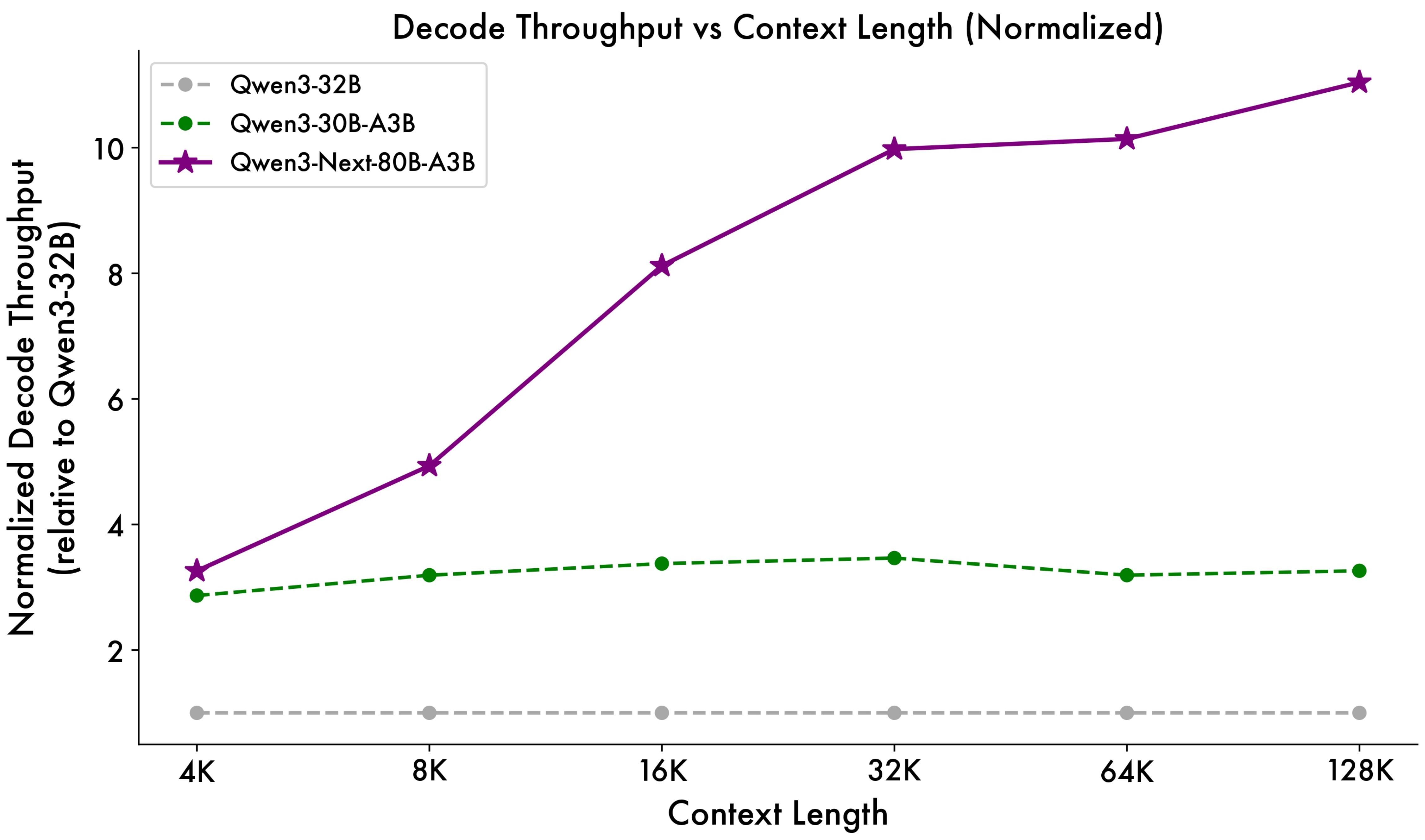


Image source: https://sebastianraschka.com/blog/

Qwen3-Next results





Model	RULER															
	Avg.	4K	8K	16K	32K	64K	96k	128K	192k	256k	384k	512k	640k	768k	896k	1M
Qwen3-30B-A3B-Instruct-2507	86.8	98.0	96.7	96.9	97.2	93.4	91.0	89.1	89.8	82.5	83.6	78.4	79.7	77.6	75.7	72.8
Qwen3-235B-A22B-Instruct-2507	92.5	98.5	97.6	96.9	97.3	95.8	94.9	93.9	94.5	91.0	92.2	90.9	87.8	84.8	86.5	84.5
Qwen3-Next-80B-A3B-Instruct	91.8	98.5	99.0	98.0	98.7	97.6	95.0	96.0	94.0	93.5	91.7	86.9	85.5	81.7	80.3	80.3

Image source: Qwen blog

Case study: Kimi Linear 48B-A3B (October 2025)

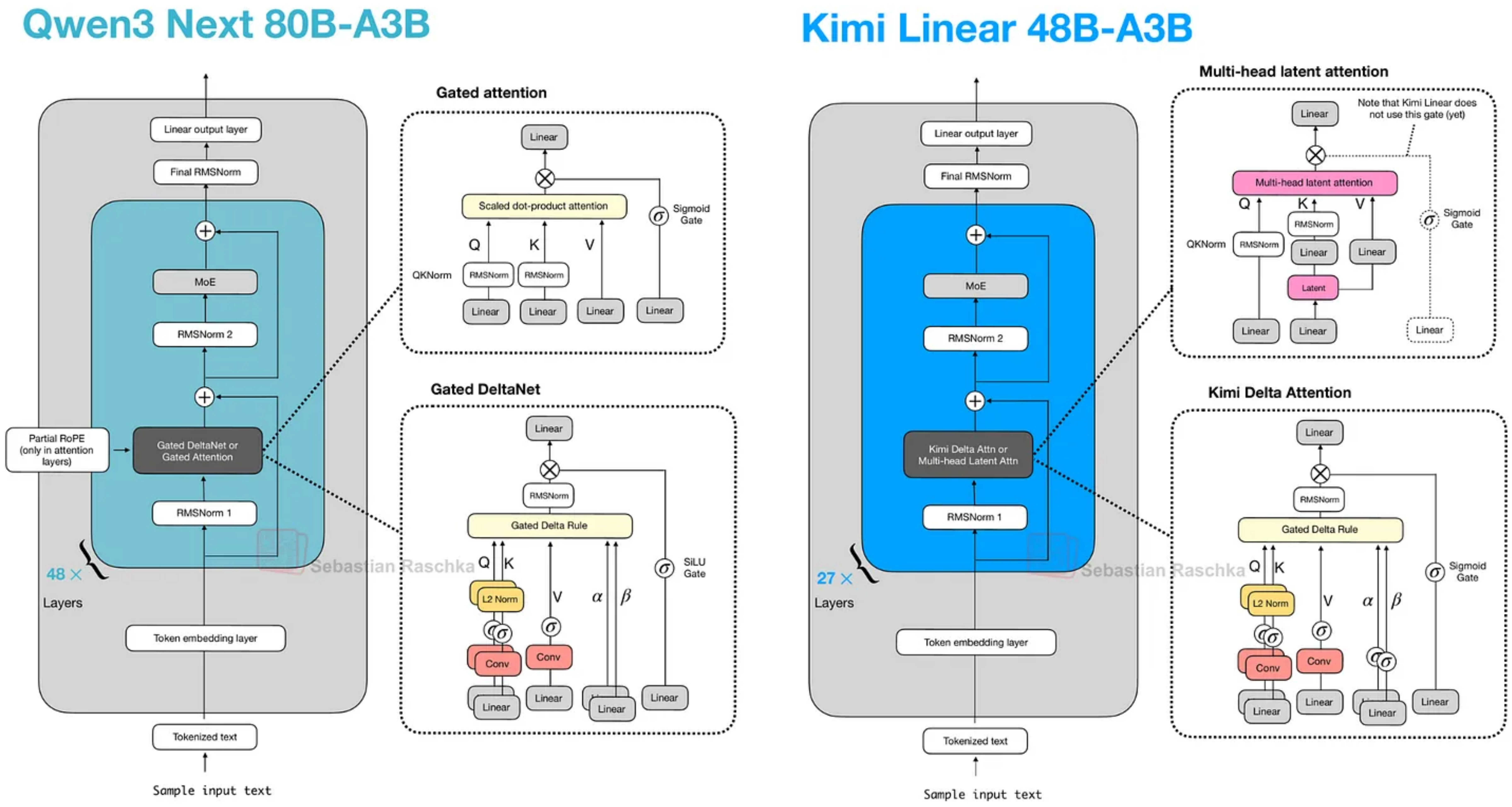


Image source: https://sebastianraschka.com/blog/

Advanced architectures: Summary

- Rapid ongoing changes, even in 2025/2026!
- Key ideas
  - ~~Scaling~~ **More efficient scaling → Mixture-of-Experts (MoE)**
    - Established as the de-facto architecture since 2021 in industry, and widely used in frontier open-source models since 2024
    - Trends: higher sparsity, more fine-grained experts, shared experts (a bit mixed)
  - **Scaling the "context window" → Attention variants**
    - KV organization: Grouped-Query Attention (GQA) vs. Multi-head Latent Attention (MLA)
    - Attention patterns: Sliding window vs. Sparse attention (e.g., DSA)
    - New mechanisms: Linear Attention Hybrids started to appear in large frontier models in Fall 2025

Questions?

Acknowledgement

Stanford CS224N NLP with Deep Learning by Diyi Yang and Tatsunori Hashimoto
(and older versions by Chris Manning)