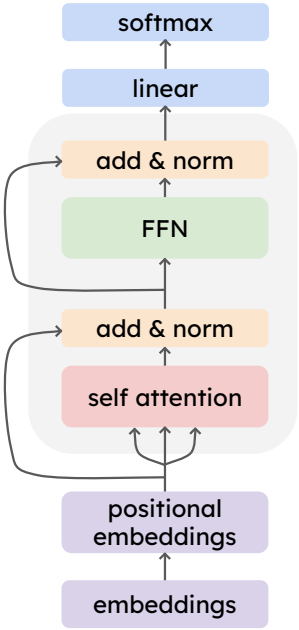
Long Contexts & Experts

Today's agenda

Two model "upgrades":

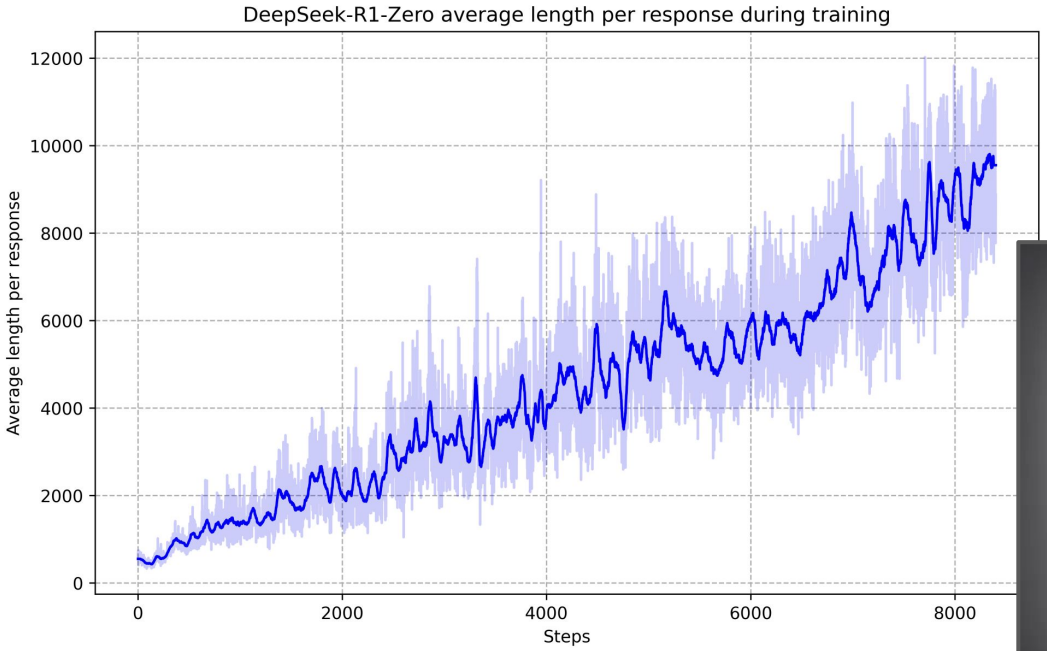
- Scaling sequence length
- Mixture of experts



a transformer

Scaling Sequence Length

Our previous lecture said …



DeepSeek-R1 pa

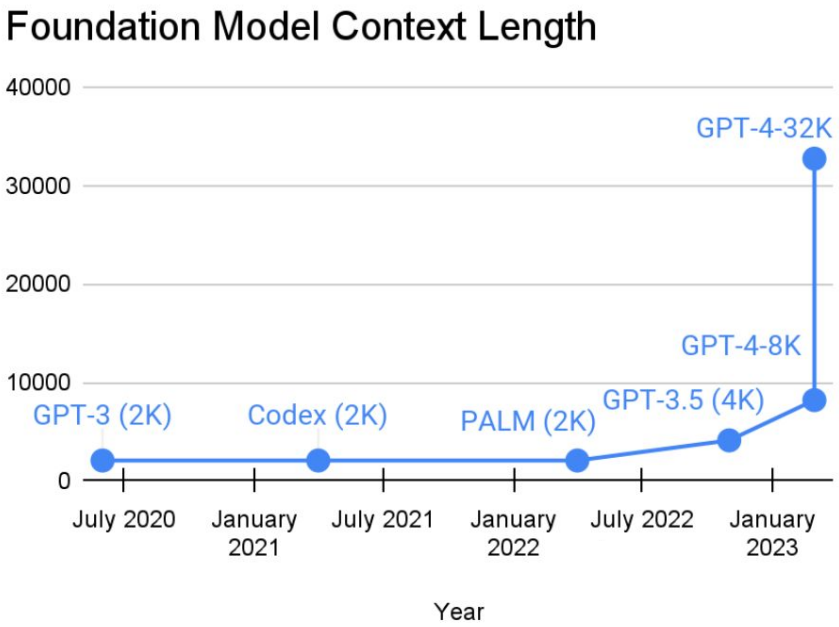


Context lengths

- One sentence: ~20 tokens
- One document: 100-10k tokens
- One book: 50k-300k tokens
- One video: 1.5k-1M tokens (~300/sec)
- One codebase: 20k-1B tokens
- One genome: 3B nucleotides

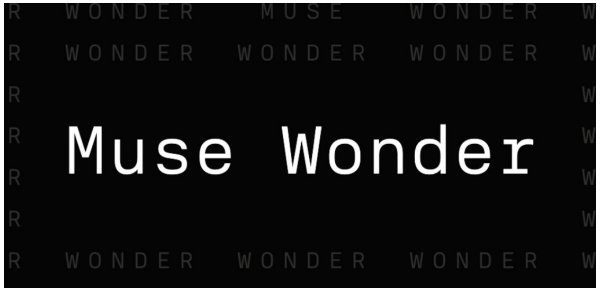


Context lengths



[From Hazy Research's blog post]

Meta



1 million context length released Oct 12, 2026

Why is this challenging?

Generalization: models trained on shorter sequences don't generalize well to longer sequences

Data: most internet documents aren't long enough to support pre-training with extremely long inputs & outputs.

Compute/memory: standard attention requires quadratic computation

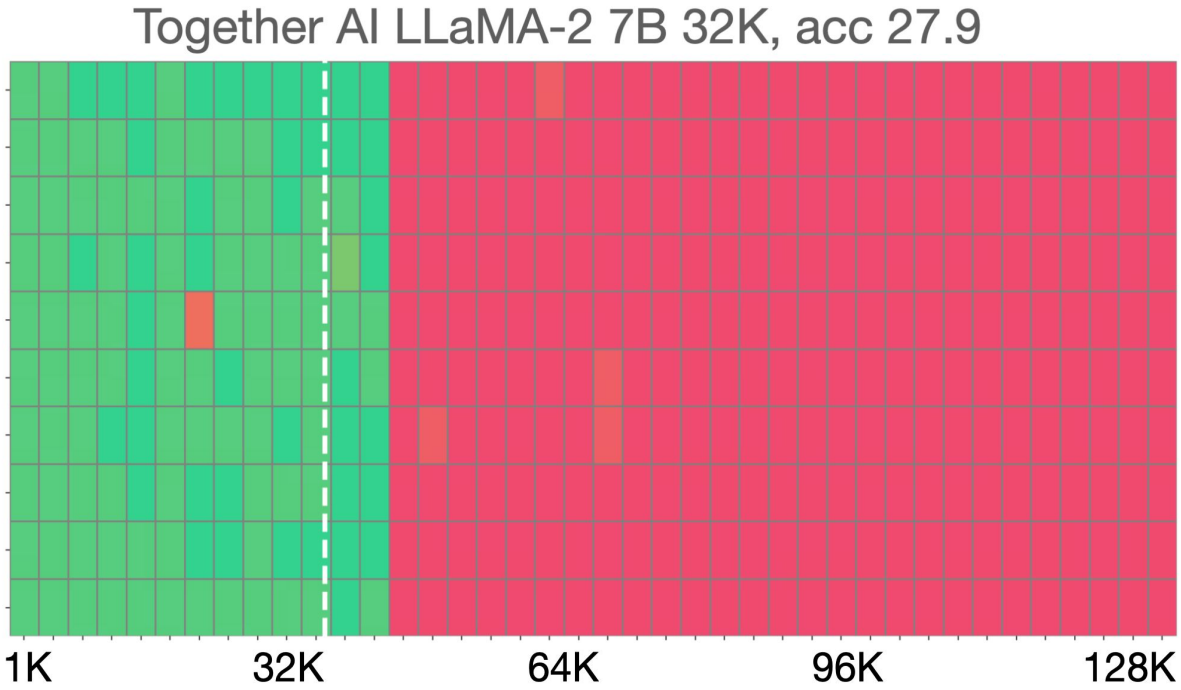
Scaling sequence length

- Three methods:
  - Staged training w/ long documents
  - Position encoding modifications
  - Attention modifications
- Tasks and benchmarks for analyzing performance

Scaling sequence length

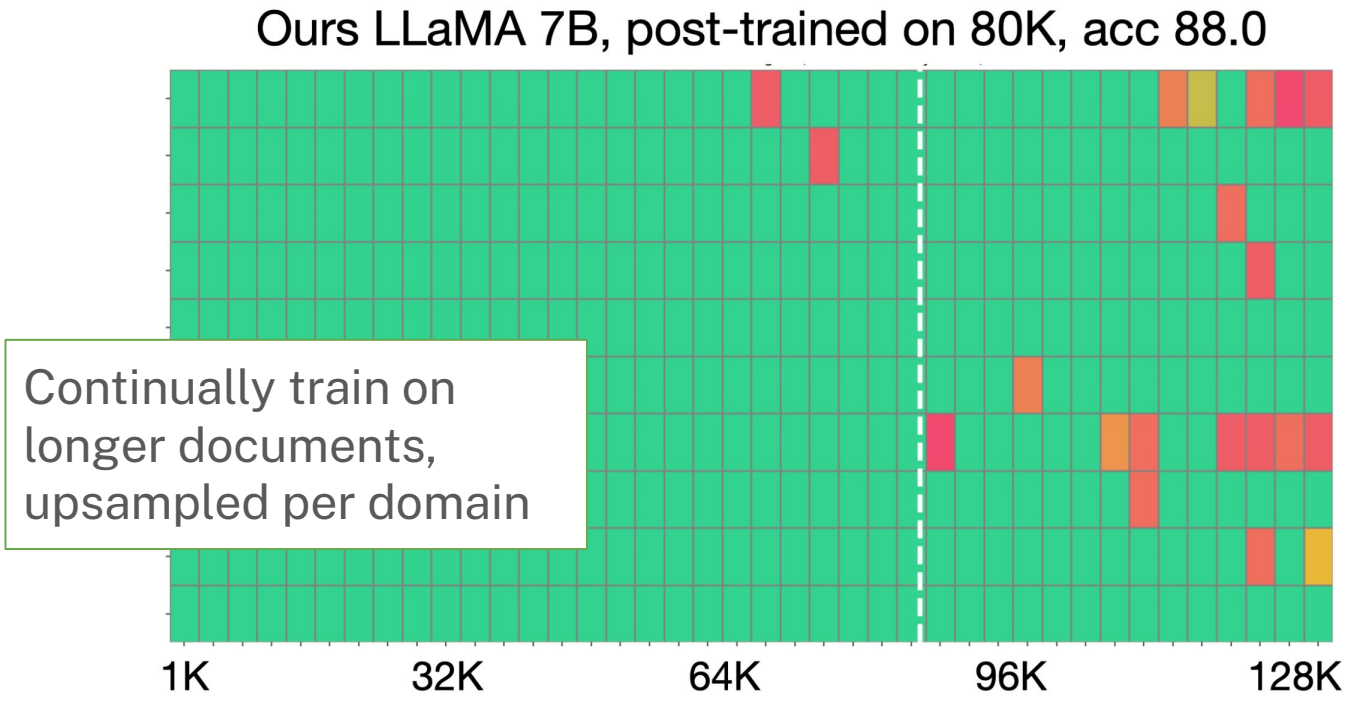
- Three methods:
  - **Staged training w/ long documents**
  - Position encoding modifications
  - Attention modifications
- Tasks and benchmarks for analyzing performance

A starting point



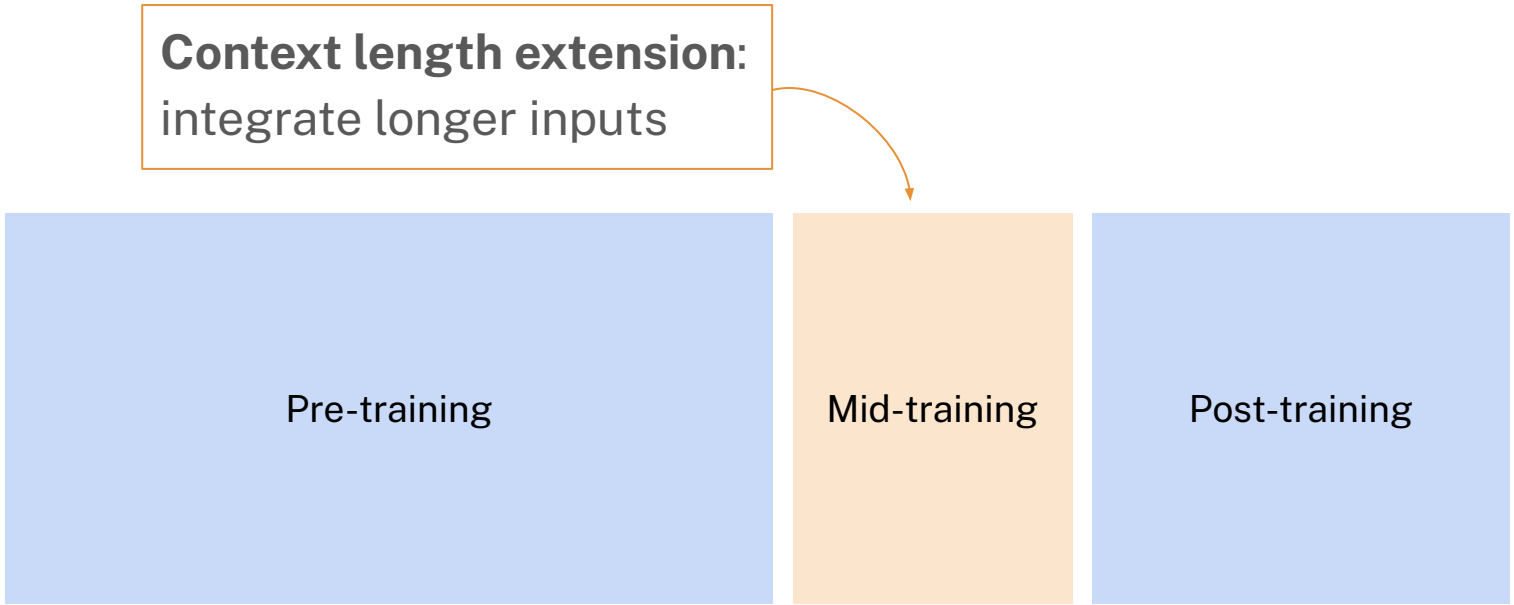
Fu et al. 2024

What we'd rather have



Fu et al. 2024

Mid-training



Staged training

Recipe 👨‍🍳:

1. Start from a base model, e.g. 4k-8k context
2. Apply position encoding modification
3. **Continual pretraining** on long documents (e.g. billions of tokens), w/ progressively increasing sequence lengths (e.g. 8k → 32k → 64k → 128k).
4. **Finetune** on long-context instruction data

Don’t need full pretraining budget!

Just 0.1-1% of original pretraining tokens is sufficient.

Data for continual pretraining

Source Type	Typical Length, in Tokens
Common Crawl (web text)	600 - 1,200
Wikipedia	500 - 1,500
Scientific papers (arXiv)	5,000 - 10,000+
Books (Project Gutenberg)	50,000 - 100,000+
Github (code)	100 - 5,000

Data for continual pretraining

Most of the pretraining data we have on hand is short.

Source Type	Typical Length, in Tokens
Common Crawl (web text)	600 - 1,200
Wikipedia	500 - 1,500
Scientific papers (arXiv)	5,000 - 10,000+
Books (Project Gutenberg)	50,000 - 100,000+
Github (code)	100 - 5,000

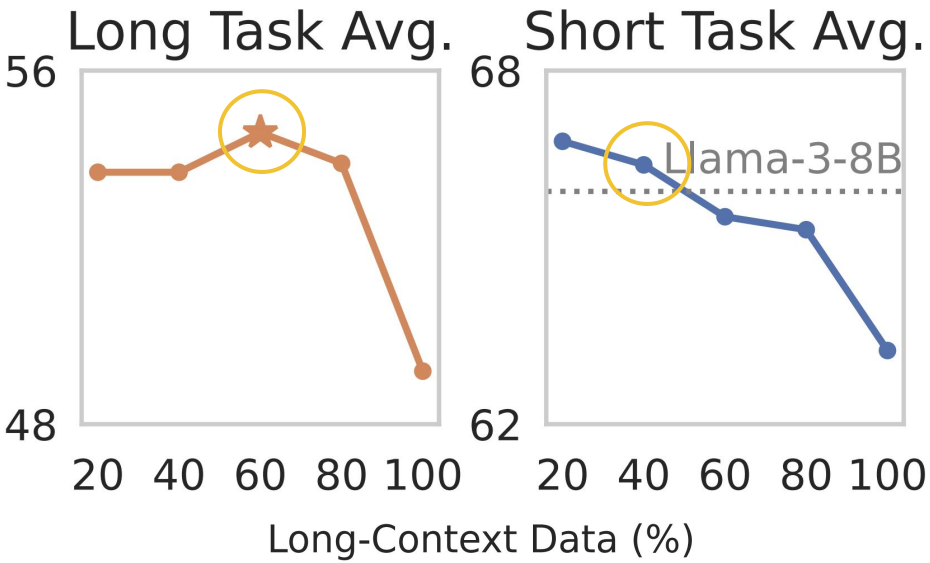
Data for continual pretraining

We'll want to upsample these, or concatenate related documents.

Source Type	Typical Length, in Tokens
Common Crawl (web text)	600 - 1,200
Wikipedia	500 - 1,500
Scientific papers (arXiv)	5,000 - 10,000+
Books (Project Gutenberg)	50,000 - 100,000+
Github (code)	100 - 5,000

Data for continual pretraining

Careful, though... Gao et al. 2025 recommends mixing in some short context data to avoid regressing on performance for short context tasks.



Data for SFT

Generate synthetic long-context tasks, e.g.

Needle-in-a-haystack retrieval

```
Dictionary [1] {122: 765, 4548:
1475, 4818: 4782}
...
Dictionary [85] {344: 1579, 116:
617, 330: 411}

Above is a list of dictionaries
... Report the value of key 2931
and the dictionary it is in.
```

Xiong et al. 2024

Long-range QA

Synthesize QA:

```
Given the following snippet of a
book, ask a relevant question and
provide the answer.
```

And then include the entire book in the actual task example.

Gao et al. 2025

Staged training

Recipe 👨‍🍳:

1. Start from a base model, e.g. 4k-8k context
2. Apply position encoding modification
3. **Continual pretraining** on long documents (e.g. billions of tokens), w/ progressively increasing sequence lengths (e.g. 8k → 32k → 64k → 128k).
4. **Finetune** on long-context instruction data

Scaling sequence length

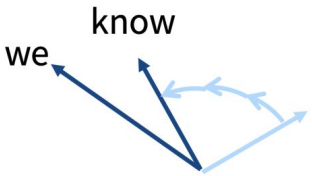
- Three methods:
  - Staged training w/ long documents
  - **Position encoding modifications**
  - Attention modifications
- Tasks and benchmarks for analyzing performance

Recap: RoPE

We encode relative positions using **rotary positional encodings**.

Multiply key and query vectors by:

$$R_t = \begin{pmatrix} \cos t\theta_1 & -\sin t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin t\theta_1 & \cos t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos t\theta_2 & -\sin t\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin t\theta_2 & \cos t\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos t\theta_{d/2} & -\sin t\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin t\theta_{d/2} & \cos t\theta_{d/2} \end{pmatrix}$$



Embedding
“of course we know”

Rotate we by ‘2 positions’
Rotate know by ‘3 positions’

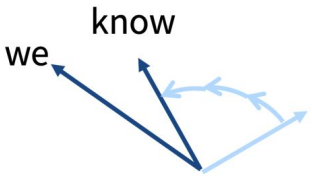
k or *q*

RoPE, a closer look

Position is encoded by rotating pairs of dimensions.

The **first** dimension pair for position *t* is rotated by $t \times \theta_1$.

$$R_t = \begin{pmatrix} \cos t\theta_1 & -\sin t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin t\theta_1 & \cos t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos t\theta_2 & -\sin t\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin t\theta_2 & \cos t\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos t\theta_{d/2} & -\sin t\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin t\theta_{d/2} & \cos t\theta_{d/2} \end{pmatrix}$$



Embedding
“of course we know”

Rotate we by ‘2 positions’
Rotate know by ‘3 positions’

k or *q*

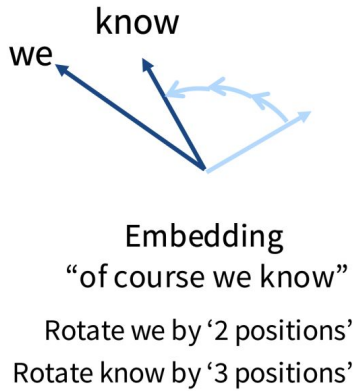


RoPE, a closer look

Position is encoded by rotating pairs of dimensions.

The **second** dimension pair for position *t* is rotated by $t \times \theta_2$.

$$R_t = \begin{pmatrix} \cos t\theta_1 & -\sin t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin t\theta_1 & \cos t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos t\theta_2 & -\sin t\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin t\theta_2 & \cos t\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos t\theta_{d/2} & -\sin t\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin t\theta_{d/2} & \cos t\theta_{d/2} \end{pmatrix}$$



k or *q*



RoPE, a closer look

Position is encoded by rotating pairs of dimensions.

The **second** dimension pair for position *t* is rotated by *t* x θ_2.

$$R_t = \begin{pmatrix} \cos t\theta_1 & -\sin t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin t\theta_1 & \cos t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos t\theta_2 & -\sin t\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin t\theta_2 & \cos t\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos t\theta_{d/2} & -\sin t\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin t\theta_{d/2} & \cos t\theta_{d/2} \end{pmatrix}$$

k or *q*



For this dim pair:

At *t = 1*, we rotate by 1 x θ_2

At *t = 2*, we rotate by 2 x θ_2

... and so on. Each position in the sequence adds another θ_2 of rotation!

RoPE, a closer look

Position is encoded by rotating pairs of dimensions.

The **second** dimension pair for position *t* is rotated by *t* x θ_2.

$$R_t = \begin{pmatrix} \cos t\theta_1 & -\sin t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin t\theta_1 & \cos t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos t\theta_2 & -\sin t\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin t\theta_2 & \cos t\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos t\theta_{d/2} & -\sin t\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin t\theta_{d/2} & \cos t\theta_{d/2} \end{pmatrix}$$

Period = how many positions *t* it takes to complete a full circle or rotation.

Bigger θ = faster rotation and shorter period.

k or *q*



RoPE, a closer look

Each dimension pair has a periodic structure.

$\theta_j = b^{-\frac{2j}{d}}$, where *b* is a base of 10000 and *d* is total # of dimensions.

$$R_t = \begin{pmatrix} \cos t\theta_1 & -\sin t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin t\theta_1 & \cos t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos t\theta_2 & -\sin t\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin t\theta_2 & \cos t\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos t\theta_{d/2} & -\sin t\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin t\theta_{d/2} & \cos t\theta_{d/2} \end{pmatrix}$$

Period = how many positions *t* it takes to complete a full circle or rotation.

Bigger θ = faster rotation and shorter period.

RoPE, a closer look

Each dimension pair has a periodic structure.

$$\theta_1 = b^{-\frac{2}{d}} \qquad \theta_2 = b^{-\frac{4}{d}} \qquad \theta_{d/2} = b^{-1}$$

$$R_t = \begin{pmatrix} \cos t\theta_1 & -\sin t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin t\theta_1 & \cos t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos t\theta_2 & -\sin t\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin t\theta_2 & \cos t\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos t\theta_{d/2} & -\sin t\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin t\theta_{d/2} & \cos t\theta_{d/2} \end{pmatrix}$$

Period = how many positions *t* it takes to complete a full circle or rotation.

Bigger θ = faster rotation and shorter period.

RoPE, a closer look

Each dimension pair has a periodic structure.

$$\theta_1 = b^{-\frac{2}{d}} \qquad \theta_2 = b^{-\frac{4}{d}} \qquad \theta_{d/2} = b^{-1}$$

$$R_t = \begin{pmatrix} \cos t\theta_1 & -\sin t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin t\theta_1 & \cos t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos t\theta_2 & -\sin t\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin t\theta_2 & \cos t\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos t\theta_{d/2} & -\sin t\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin t\theta_{d/2} & \cos t\theta_{d/2} \end{pmatrix}$$

Takeaway: initial dimension pairs rotate fast (high frequency) as *t* increases, later pairs rotate slower.

$$\theta_1 > \theta_2 > \ldots > \theta_{d/2}$$

RoPE with attention

During **attention**, we compute the dot product of a query at position *m* and a key at position *n*.

Imagine our query and key are 2-dimensional, so we have just one *θ*.

$$R_t = \begin{pmatrix} \cos t\theta & -\sin t\theta \\ \sin t\theta & \cos t\theta \end{pmatrix}$$

Rotate by an angle of *t* x *θ*

So, the query is rotated by *mθ* and the key is rotated by *nθ*.

$$(R_m q)^\top (R_n k) = q^\top R_{(n-m)} k \qquad R_{(n-m)} = \begin{pmatrix} \cos (n-m)\theta & -\sin (n-m)\theta \\ \sin (n-m)\theta & \cos (n-m)\theta \end{pmatrix}$$

During training, our model learns to handle **relative positions**, or rotations by angles of (*n* - *m*)*θ*.

RoPE w/ longer sequences

With different θ, rotating by $(n-m)\theta$ leads to different angles that cycle at different speeds as $n-m$ grows bigger.

What happens w/ long sequences and **long relative distances**?

- **High frequency pairs**: as relative distances grow, their angles cycle through the period. The model has seen every angle a query and a key can produce for these dimensions during training. ✔
- **Low frequency pairs**: during pretraining with short contexts, the model only sweeps part of the circle! If two tokens are very far from another at test time, the model has never seen their relative angle. ✗

RoPE scaling

Solution 1: Position interpolation squeezes long sequences into the original context window.

Scale positions *t* down by *s = L'/L*, for new max sequence length *L'* and original length *L*, so the model sees familiar relative positions:

$$R_t = \begin{pmatrix} \cos\frac{t}{s}\theta_1 & -\sin\frac{t}{s}\theta_1 & \cdots & 0 & 0 \\ \sin\frac{t}{s}\theta_1 & \cos\frac{t}{s}\theta_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cos\frac{t}{s}\theta_{d/2} & -\sin\frac{t}{s}\theta_{d/2} \\ 0 & 0 & \cdots & \sin\frac{t}{s}\theta_{d/2} & \cos\frac{t}{s}\theta_{d/2} \end{pmatrix}$$

Then, finetune on longer data.

RoPE scaling

Position interpolation fixes low frequency dim pairs so that their angles fit into the original context window. ✔

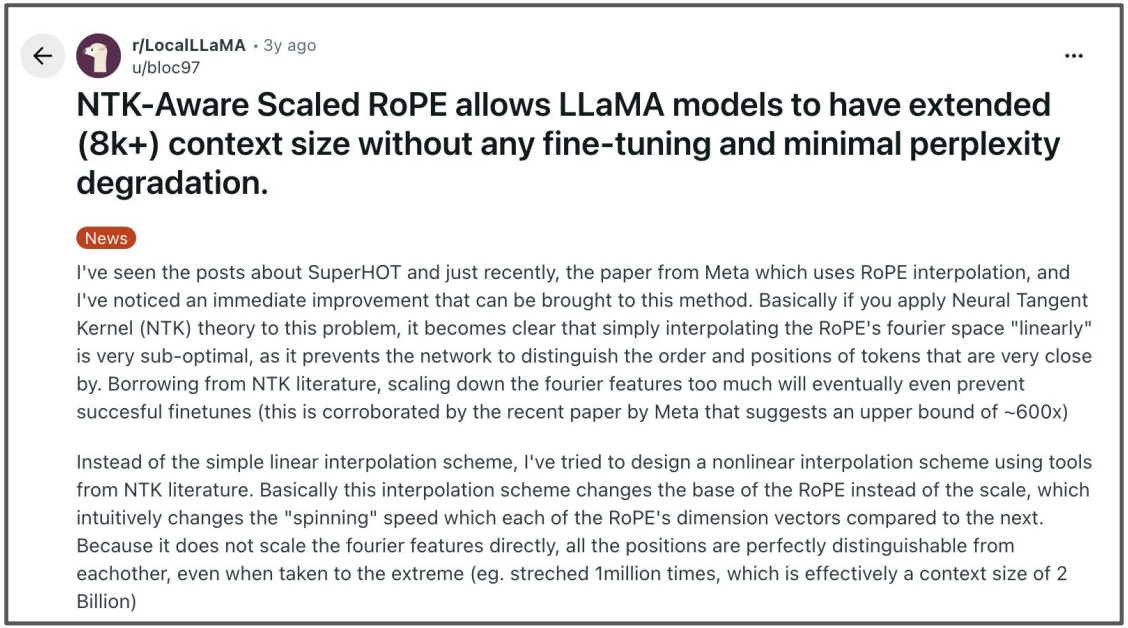
But it also slows the faster pairs unnecessarily, dulling the model's sense of local relative positions! 🧐

Solution 2: NTK-aware interpolation

Let's adjust the rotary base instead!

RoPE scaling

Solution 2: NTK-aware interpolation



r/LocalLLaMA • 3y ago
u/bloc97

NTK-Aware Scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

News

I've seen the posts about SuperHOT and just recently, the paper from Meta which uses RoPE interpolation, and I've noticed an immediate improvement that can be brought to this method. Basically if you apply Neural Tangent Kernel (NTK) theory to this problem, it becomes clear that simply interpolating the RoPE's fourier space "linearly" is very sub-optimal, as it prevents the network to distinguish the order and positions of tokens that are very close by. Borrowing from NTK literature, scaling down the fourier features too much will eventually even prevent succesful finetunes (this is corroborated by the recent paper by Meta that suggests an upper bound of ~600x)

Instead of the simple linear interpolation scheme, I've tried to design a nonlinear interpolation scheme using tools from NTK literature. Basically this interpolation scheme changes the base of the RoPE instead of the scale, which intuitively changes the "spinning" speed which each of the RoPE's dimension vectors compared to the next. Because it does not scale the fourier features directly, all the positions are perfectly distinguishable from eachother, even when taken to the extreme (eg. streched 1million times, which is effectively a context size of 2 Billion)

RoPE scaling

Solution 2: NTK-aware interpolation

Adjust the rotary base.

Compress positions less in high-freq pairs, and more for low-freq pairs:

Before

$$\theta_j = b^{-\frac{2j}{d}}$$

After

$$\theta_j = b'^{-\frac{2j}{d}} \qquad b' = b \cdot s^{\frac{d}{d-2}} \qquad s = \frac{L'}{L}$$

YaRN, a SOTA approach, builds upon this method, using *NTK-by-parts*.

Scaling sequence length

- Three methods:
  - Staged training w/ long documents
  - Position encoding modifications
  - **Attention modifications**
- Tasks and benchmarks for analyzing performance

Recap: attention computation

$Q \in \mathbb{R}^{n \times d}$ (query) $K \in \mathbb{R}^{n \times d}$ (keys) $V \in \mathbb{R}^{n \times d}$ (values)

$$A = \text{softmax}\left(Q(K)^\top / \sqrt{d}\right) \in \mathbb{R}^{n \times n}$$

$$O = AV \in \mathbb{R}^{n \times d}$$

Notation note: in our transformer lecture slides, we used the letters *Q*, *K*, *V* to represent the weight matrices that we apply onto an embedding matrix *X*. Here, *Q*, *K*, *V* represent **the output** of that calculation, or the resultant queries, keys, and values.

Attention compute & memory

$Q \in \mathbb{R}^{n \times d}$ (query) $K \in \mathbb{R}^{n \times d}$ (keys) $V \in \mathbb{R}^{n \times d}$ (values)

$$A = \text{softmax}\left(Q(K)^\top / \sqrt{d}\right) \in \mathbb{R}^{n \times n}$$

$$O = AV \in \mathbb{R}^{n \times d}$$

Compute: number of output entries × length of each dot product
Memory: number output entries

Compute: number of output entries × length of each dot product
Memory: number output entries

Attention compute & memory

$Q \in \mathbb{R}^{n \times d}$ (query) $K \in \mathbb{R}^{n \times d}$ (keys) $V \in \mathbb{R}^{n \times d}$ (values)

$$A = \text{softmax}\left(Q(K)^\top / \sqrt{d}\right) \in \mathbb{R}^{n \times n}$$

$$O = AV \in \mathbb{R}^{n \times d}$$

Compute: $n^2 \times d$
Memory: n^2

Compute: $nd \times n$
Memory: nd

Don't forget batch size!

Attention compute & memory

$Q \in \mathbb{R}^{n \times d}$ (query) $K \in \mathbb{R}^{n \times d}$ (keys) $V \in \mathbb{R}^{n \times d}$ (values)

$$A = \text{softmax}\left(Q(K)^\top / \sqrt{d}\right) \in \mathbb{R}^{n \times n}$$

$$O = AV \in \mathbb{R}^{n \times d}$$

Compute: O(bn^2d)
Memory: O(bn^2)

Compute: O(bn^2d)
Memory: O(bnd)

Where b = batch size

Attention compute & memory

$Q \in \mathbb{R}^{n \times d}$ (query) $K \in \mathbb{R}^{n \times d}$ (keys) $V \in \mathbb{R}^{n \times d}$ (values)

$$A = \text{softmax}\left(Q(K)^\top / \sqrt{d}\right) \in \mathbb{R}^{n \times n}$$

$$O = AV \in \mathbb{R}^{n \times d}$$

Compute: O(bn^2d)
Memory: O(bn^2)

Compute: O(bn^2d)
Memory: O(bnd)

Takeaway: attention is quadratic in sequence length *n*, in both memory and computation!

Multi-head attention compute & memory

Recap: multi-head attention splits attention across *h* heads.

$$A = \text{softmax}\left(Q_i (K_i)^\top / \sqrt{d}\right) \in \mathbb{R}^{n \times n}$$

Compute: number of output entries × length of each dot product × number of heads
Memory: number output entries × number of heads

$$O = AV_i \in \mathbb{R}^{n \times d}$$

Compute: number of output entries × length of each dot product × number of heads
Memory: number output entries × number of heads

Multi-head attention compute & memory

Recap: multi-head attention splits attention across *h* heads.

$$A = \text{softmax}\left(Q_i (K_i)^\top / \sqrt{d}\right) \in \mathbb{R}^{n \times n}$$

Compute: $n^2 \times d/h \times h = n^2 d$
Memory: $n^2 \times h = n^2 h$

$$O = AV_i \in \mathbb{R}^{n \times d}$$

Compute: $n(d/h) \times n \times h = n^2 d$
Memory: $n(d/h) \times h = nd$

Multi-head attention compute & memory

Recap: multi-head attention splits attention across *h* heads.

No effect on compute complexity, but **increases memory**!

$$A = \text{softmax}\left(Q_i (K_i)^\top / \sqrt{d}\right) \in \mathbb{R}^{n \times n}$$

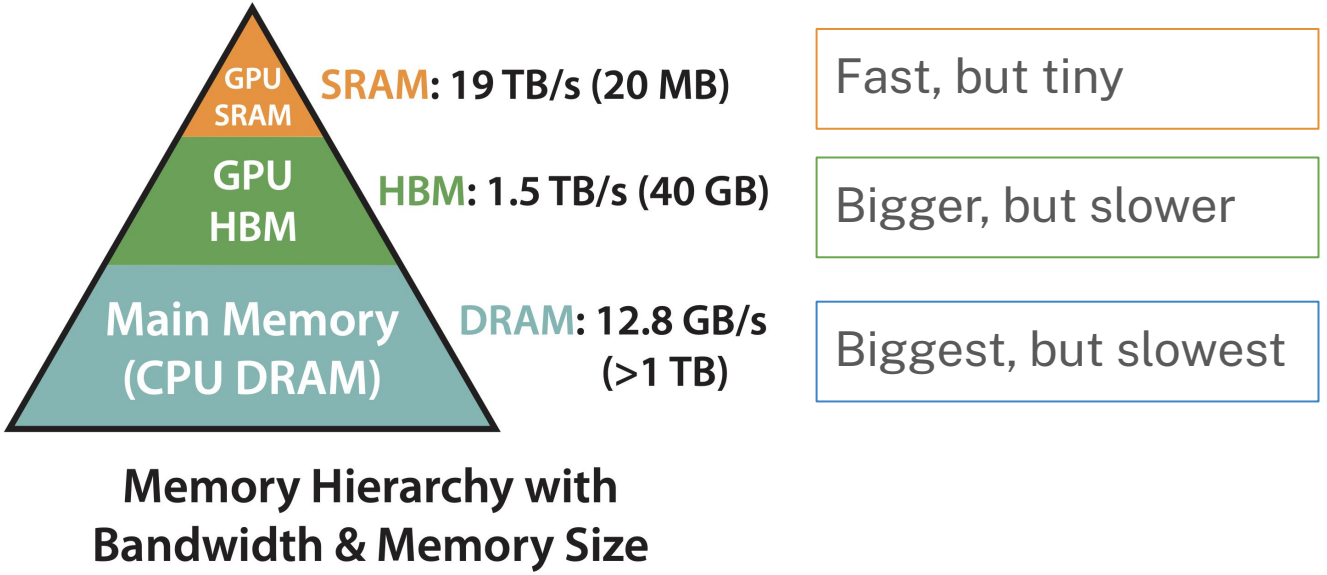
Compute: O(bn^2d)
Memory: O($bn^2\mathbf{h}$)

$$O = AV_i \in \mathbb{R}^{n \times d}$$

Compute: O(bn^2d)
Memory: O(bnd)

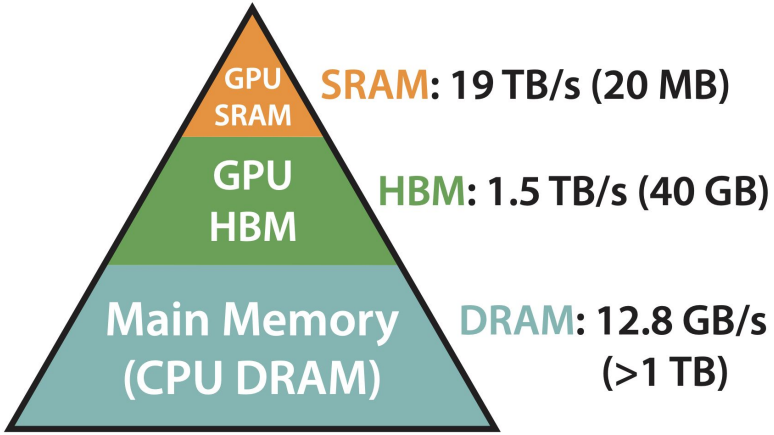
Memory bottlenecks

Accelerators (e.g. a NVIDIA A100 GPU) have limited memory capacity and bandwidth.



Memory bottlenecks

Accelerators (e.g. a NVIDIA A100 GPU) have limited memory capacity and bandwidth.



Memory Hierarchy with Bandwidth & Memory Size

[FlashAttention paper]

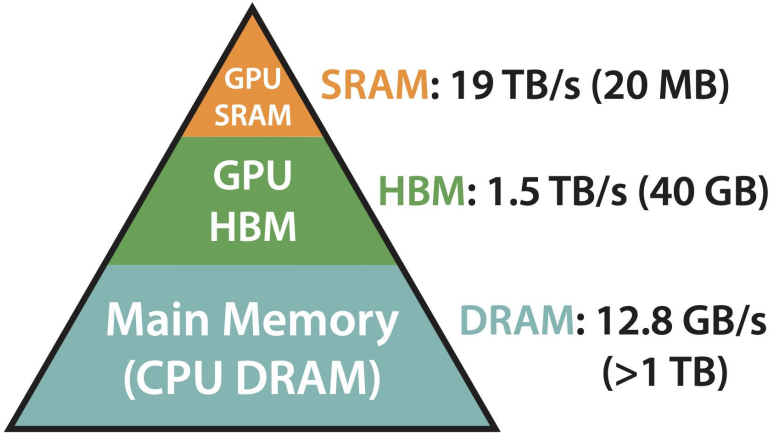
On chip memory, very accessible to compute cores executing ops

GPU memory for model weights, activations, Q, K, V tensors

CPU memory. Slow for GPU to reach (illustrated here mostly for context).

Memory bottlenecks

Accelerators (e.g. a NVIDIA A100 GPU) have limited memory capacity and bandwidth.



Memory Hierarchy with Bandwidth & Memory Size

FlashAttention paper

During attention computation, $O(bn^2h)$ means lots of slow transfers!

Long contexts (big n) make it expensive to pretrain & generate!
Expensive = slow and OOM

Memory-efficient computation

Mitigation: You can compute softmax “online” to avoid keeping n^2 matrices in memory.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(Q(K)^\top / \sqrt{d}\right)V$$

Rearrange this in fraction form!

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

Memory-efficient computation

Mitigation: You can compute softmax “online” to avoid keeping n^2 matrices in memory.

		memory
$\text{Attention}(Q,K,V) = \frac{\exp\left(\frac{QK^\top}{\sqrt{d}}\right)V}{\sum \exp\left(\frac{QK^\top}{\sqrt{d}}\right)}$	produces a $n \times d/h$ matrix	$b \times \mathbf{n} \times \mathbf{d/h} \times h$
	produces n scalars	$b \times \mathbf{n} \times h$

Memory-efficient computation

Mitigation: You can compute softmax "online" to avoid keeping n^2 matrices in memory.

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

memory
O(*bnd*)
O(*bnh*)

The numerator is the more expensive part, since $d > h$. But importantly, **no n^2** !

Instead of computing attention all at once and **suffer quadratically** …

Let's handle each query and key pair one at a time!

And add each pair's contributions to the multiplication in the numerator and denominator separately. (Rabe & Staats 2021)

Memory-efficient computation

Mitigation: You can compute softmax "online" to avoid keeping n^2 matrices in memory.

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

memory
O(*bnd*)
O(*bnh*)

The numerator is the more expensive part, since $d > h$. But importantly, **no n^2** !

Just changing the order in which we compute things!

Analogy: To find the average of a million numbers, maybe we don't store all all of the numbers. We could keep a running total and a running count, and divide at the end.

Memory-efficient computation

For each token's query *q*:

$$v^* \leftarrow 0,\ s^* \leftarrow 0$$

For *i* = 1, ..., *n*:

$$w_i' = \mathrm{dot}\,(q, k_i)$$

$$v^* \leftarrow v^* + v_i \exp\,(w_i')$$

$$s^* \leftarrow s^* + \exp\,(w_i')$$

$$\mathrm{Attention}\,(q) = v^*/s^*$$

Memory-efficient computation

Memory requirements during inference (similar improvements for training):

Sequence length	$n = 2^8$	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}
Size of inputs and outputs	160KB	640KB	2.5MB	10MB	40MB	160MB	640MB
Memory overhead of standard attention	270KB	4.0MB	64MB	1GB	OOM	OOM	OOM
Memory overhead of memory-eff. attn.	270KB	4.0MB	16MB	17MB	21MB	64MB	256MB

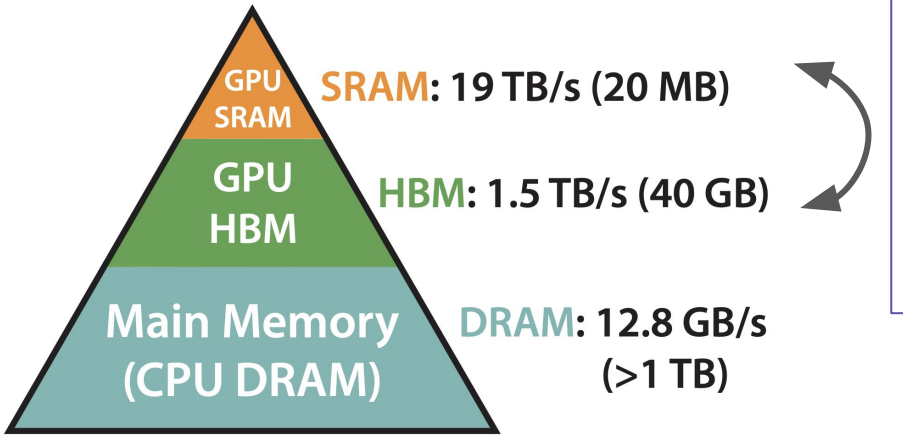
Rabe & Staats 2021

In FlashAttention, we split *Q*, *K*, V into blocks, or "tiles", that are small enough to fit in on-chip memory.

We combine each tiles' calculations to the running softmax.

Memory-efficient computation

A **kernel** is a function that runs on GPU. (🌽 != GPU kernel != a Linux kernel != jupyter notebook kernel)



Memory Hierarchy with Bandwidth & Memory Size

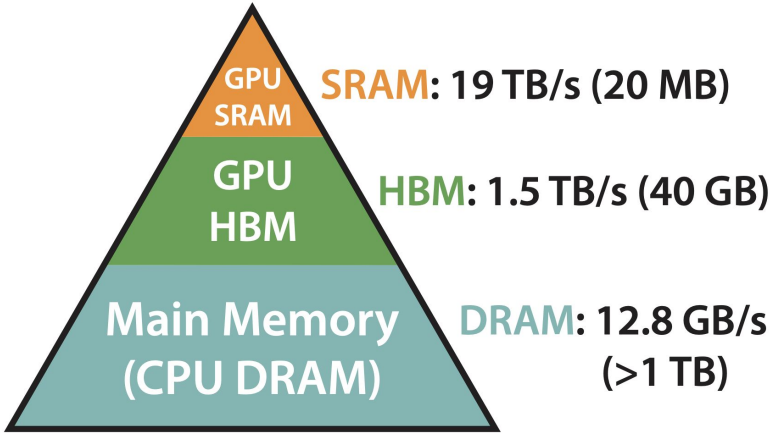
FlashAttention paper

Normally, each step of attention (scores, softmax, multiplying by V) is its own kernel.

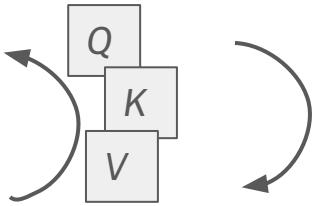
Every handoff b/t kernels requires a back and forth.

FlashAttention

A **kernel** is a function that runs on GPU. (🌽 != GPU kernel != a Linux kernel != jupyter notebook kernel)



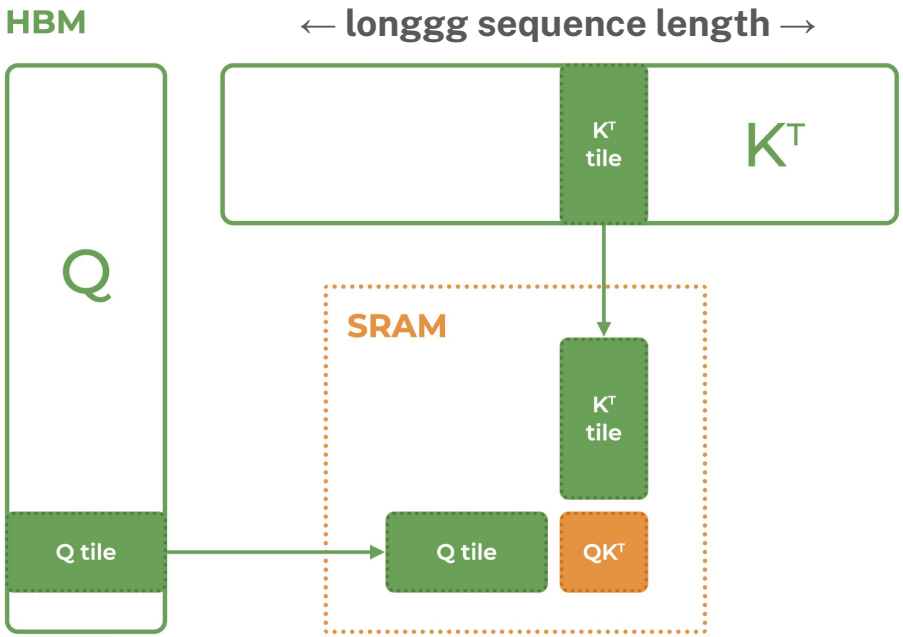
Memory Hierarchy with Bandwidth & Memory Size



FlashAttention does all of the steps in **one kernel**!

FlashAttention paper

FlashAttention



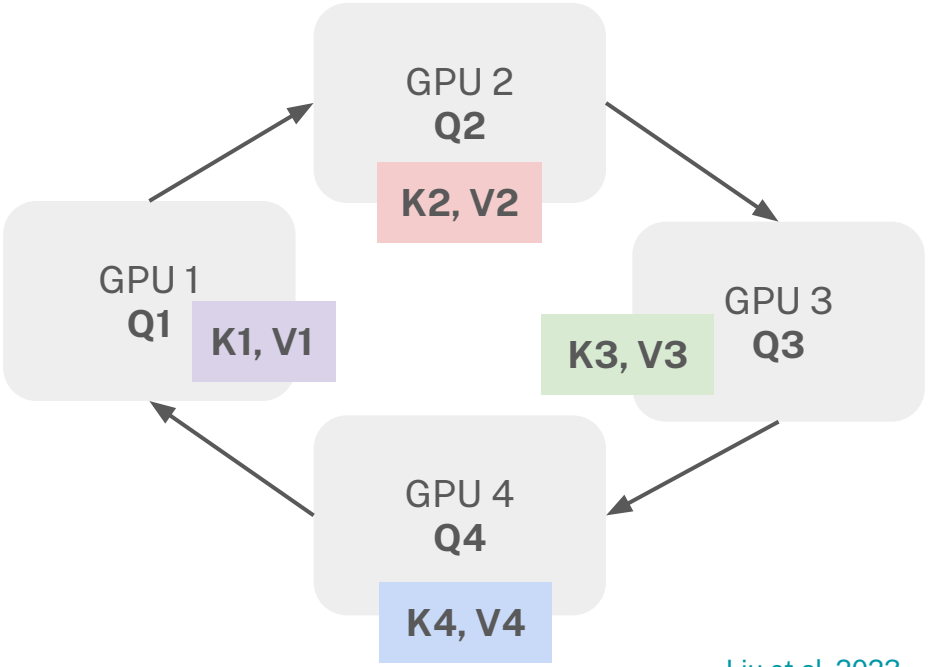
Ring attention

FlashAttention works inside one GPU.

What can we do with **multiple GPUs**?

Split our longgg sequence over devices, or **context parallelism**.

Each device holds one query block, and key-value blocks traverse through a ring.



Liu et al. 2023

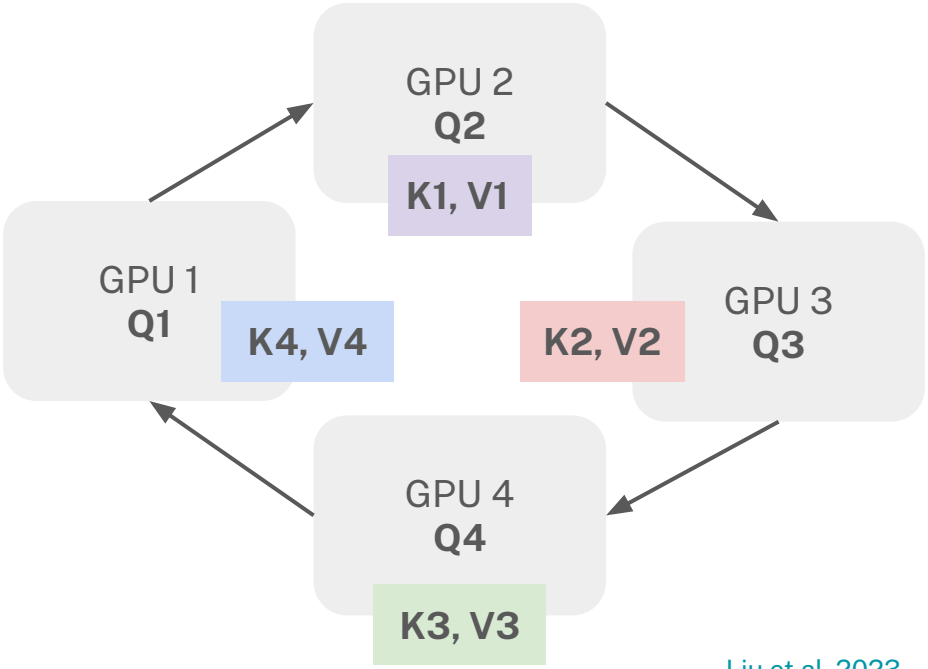
Ring attention

FlashAttention works inside one GPU.

What can we do with **multiple GPUs**?

Split our longgg sequence over devices, or **context parallelism**.

Each device holds one query block, and key-value blocks traverse through a ring.



Liu et al. 2023

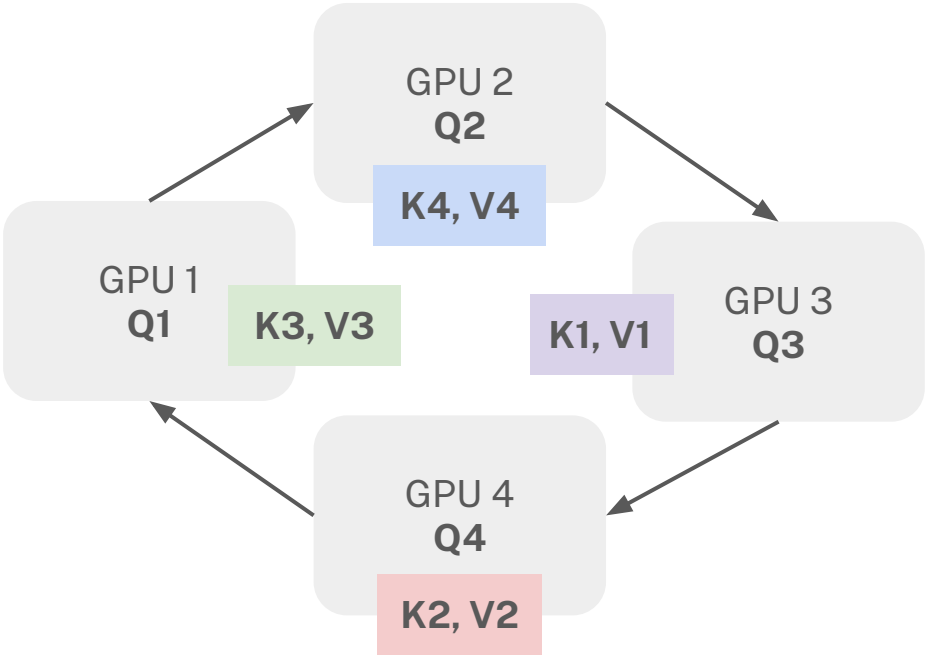
Ring attention

FlashAttention works inside one GPU.

What can we do with **multiple GPUs**?

Split our longgg sequence over devices, or **context parallelism**.

Each device holds one query block, and key-value blocks traverse through a ring.



Liu et al. 2023

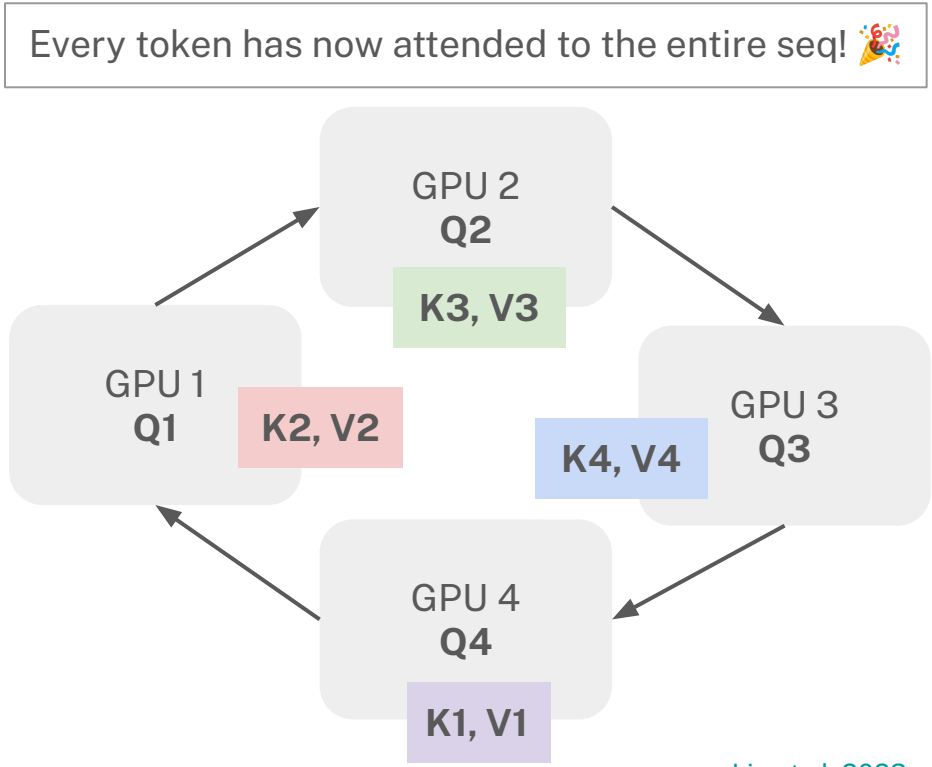
Ring attention

FlashAttention works inside one GPU.

What can we do with **multiple GPUs**?

Split our longgg sequence over devices, or **context parallelism**.

Each device holds one query block, and key-value blocks traverse through a ring.



Liu et al. 2023

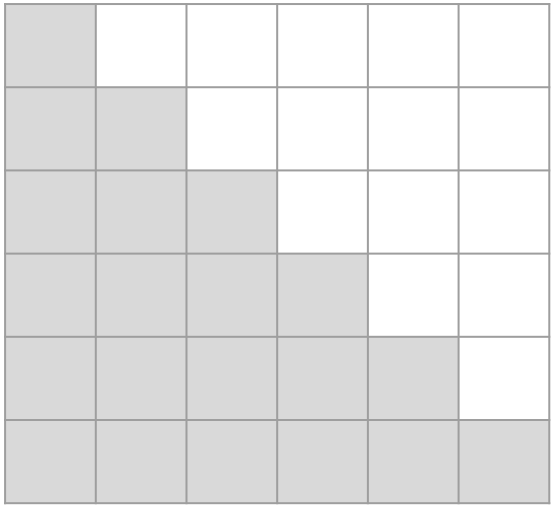
Recap: causal attention

	<s>	the	chicken	crossed
<s>	1.00	0.00	0.00	0.00
the	0.48	0.52	0.00	0.00
chicken	0.31	0.35	0.34	0.00
crossed	0.25	0.26	0.23	0.26

Compute:
$O(n^2)$

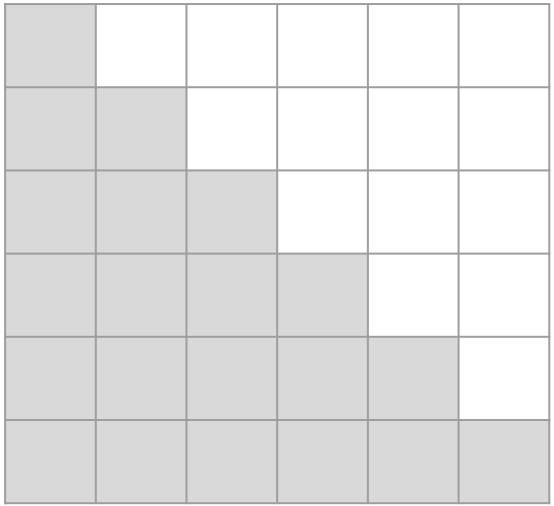
Sparse or sliding window attention

What if we didn’t compute attention between every pair of tokens?

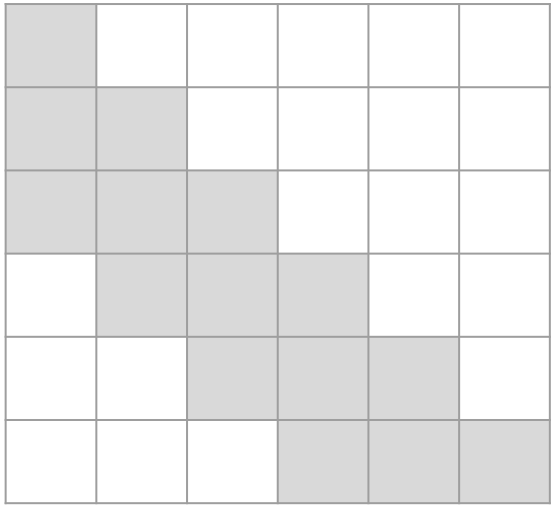


Sparse or sliding window attention

regular causal attention



sliding window attention



Compute: O(*nw*), and $w < n$

Some LMs, e.g. DeepSeek-V3.2, *learn* sparse attention patterns per token.

Putting it all together in an example LM

Example model: **Olmo 3**

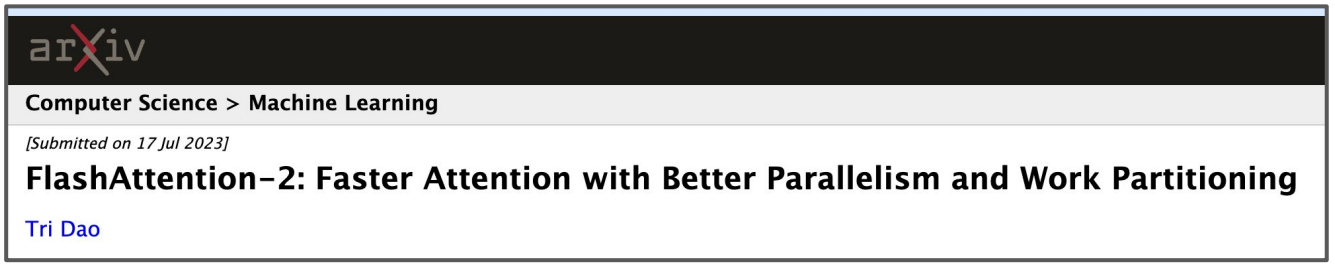
RoPE extension OLMO 3 uses RoPE (Su et al., 2024) to encode positional information within the transformer architecture. We experiment with several methods for extending RoPE beyond the original pretraining context length, including adjusted base frequency scaling (Xiong et al., 2023; Rozière et al., 2024), position interpolation (Chen et al., 2023), and YaRN (Peng et al., 2023). Each approach is applied either to all RoPE instances or is restricted to RoPE used in full attention layers. We find that applying YaRN only to full attention layers yields the best overall performance.

Generation pipeline The main challenge in generating synthetic data for long-context understanding is the bootstrap problem: how can we create effective data without having access to models that can process long context? Our pipeline uses document statistics to identify the most important terms and then extracts snippets containing those terms. Those snippets are subsequently provided to a language model to create aggregation tasks. In detail:

Putting it all together in an example LM

Example model: **Olmo 3**

Training OLMO 3 BASE is trained using the OLMo-core[4] codebase. With this stack, we train the 7B model at 7700 tokens per second per GPU and the 32B model at 1960 tokens per second per GPU at a sequence length of 8192, using `bfloat16` precision throughout. This corresponds to roughly 43% and 41% MFU, respectively. We achieve this performance by combining PyTorch's built-in `torch.compile()`, custom kernels for operations such as attention (Dao, 2024) and the language modeling head (Hsu et al., 2025), asynchronous and batched gathering of metrics, and asynchronous checkpoint writing, among other optimizations.



Putting it all together in an example LM

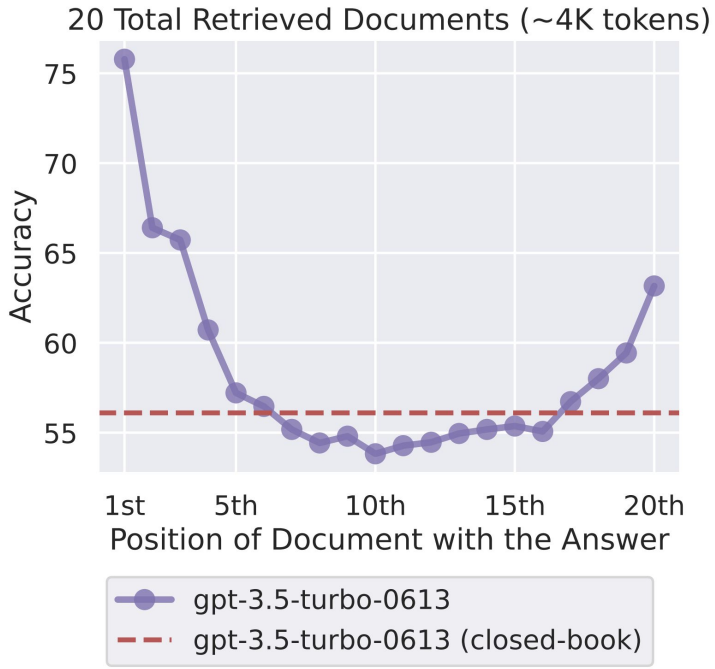
Example model: **Olmo 3**

- To support scalable pretraining at longer sequence lengths, and to keep inference costs manageable, we introduce a sliding window attention (SWA) pattern (Beltagy et al., 2020) in which each token can attend to previous tokens in a window of size 4096. We add SWA at three out of every four layers, and ensure that the last layer always uses full attention.

Scaling sequence length

- Three methods:
  - Staged training w/ long documents
  - Position encoding modifications
  - Attention modifications
- **Tasks and benchmarks for analyzing performance**

Lost in the Middle



Input Context

```
Write a high-quality answer for the given question
using only the provided search results (some of
which might be irrelevant).

Document [1](Title: List of Nobel laureates in
Physics) ...
Document [2](Title: Asian Americans in science and
technology) ...
Document [3](Title: Scientist) ...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Liu et al. 2024

RULER

Needle-in-a-haystack

```
......
One of the special magic
numbers for {word} is:
12345.
One of the special magic
numbers for {word} is:
54321.
......
```
What are all the special magic numbers for {word} mentioned in the provided text?
```
Answer: 12345 54321
```

Common words extraction

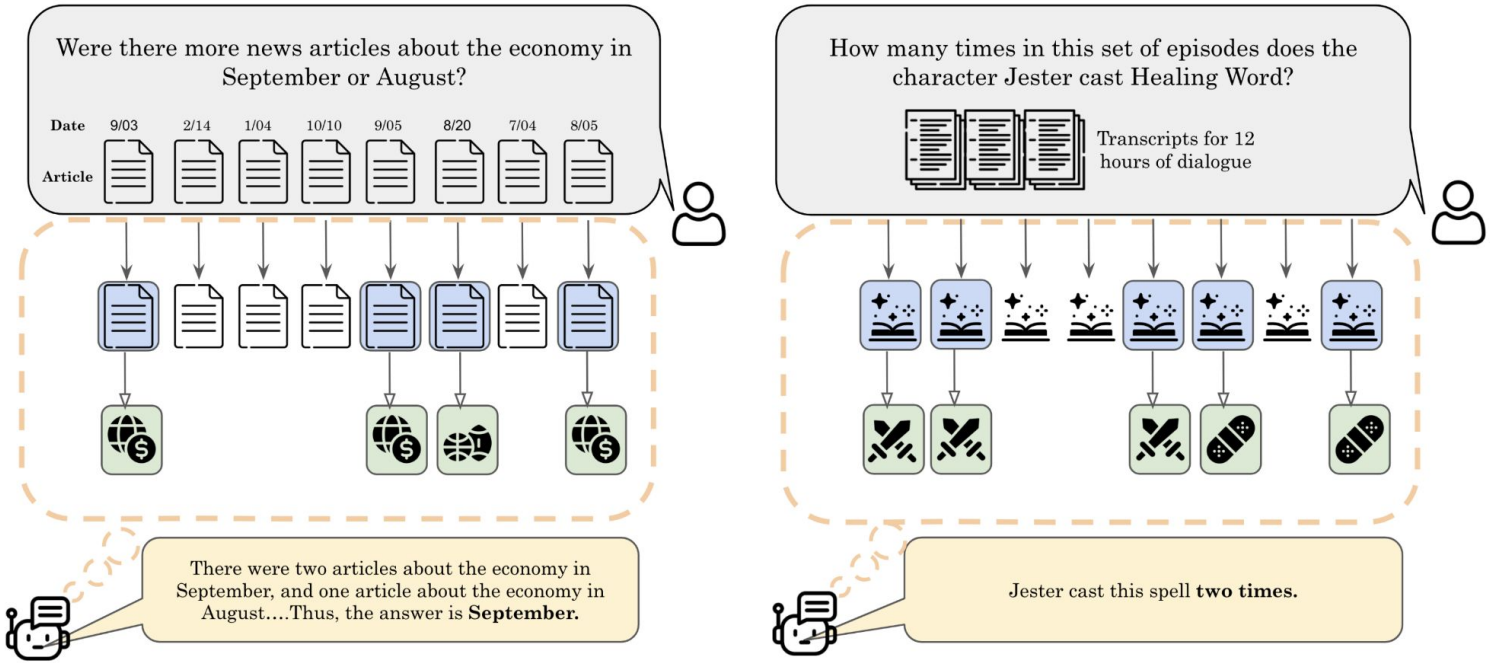
```
aaa bbb ccc aaa ddd eee
ccc fff ggg hhh iii iii
......
```
What are the 10 most common words in the above list?
```
Answer: aaa ccc iii ......
```

Variable tracking

```
VAR X1 = 12345 ......
VAR Y1 = 54321 ......
VAR X2 = X1 ......
VAR Y2 = Y1 ......
VAR X3 = X2 ......
VAR Y3 = Y2 ......
```
Find all variables that are assigned the value 12345.
```
Answer: X1 X2 X3
```

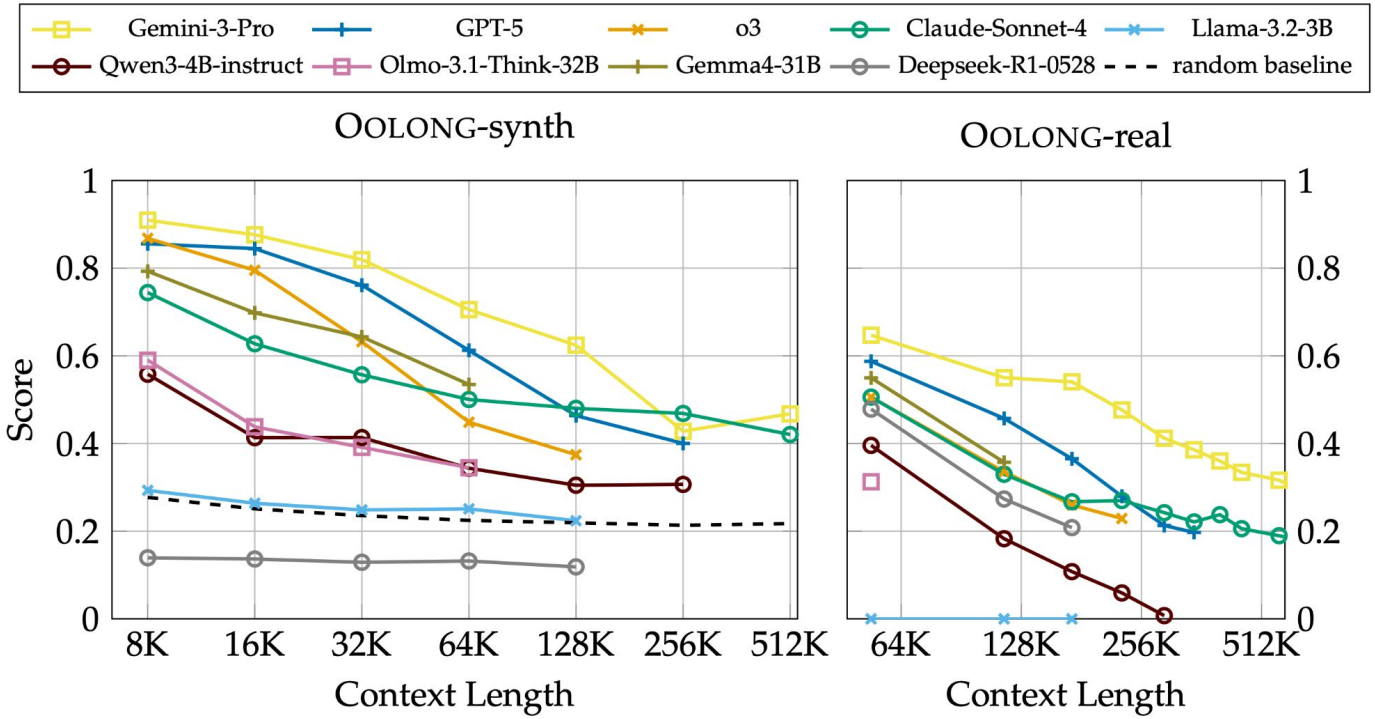
Hsieh et al. 2024

Oolong



[Bertsch et al. 2026]

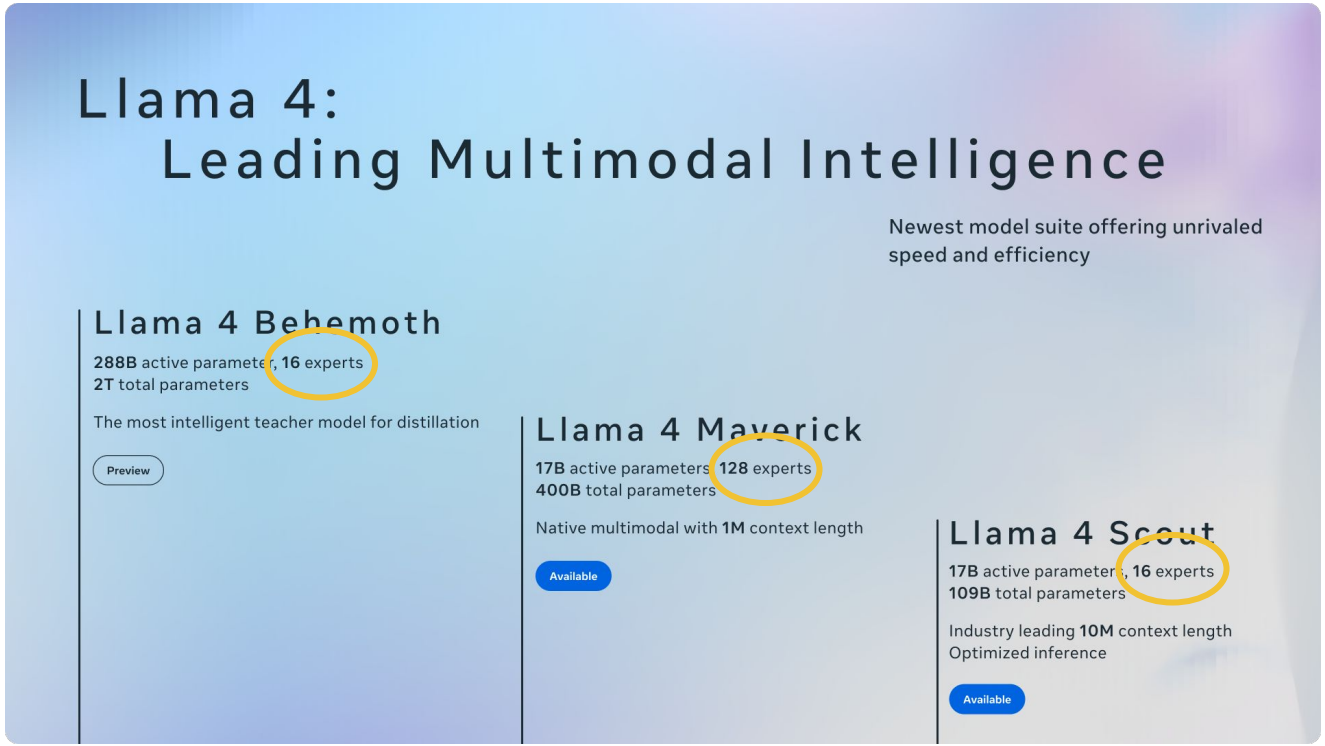
Oolong



Bertsch et al. 2026

Mixture-of-Experts

Mixture of Experts

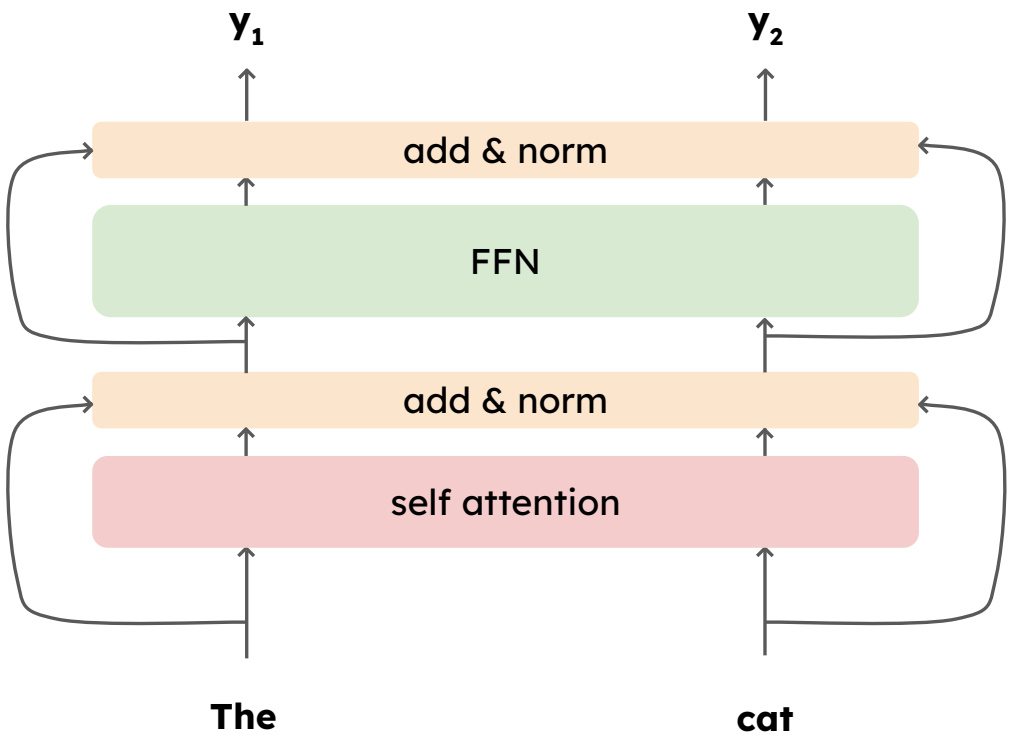


Mixture of Experts

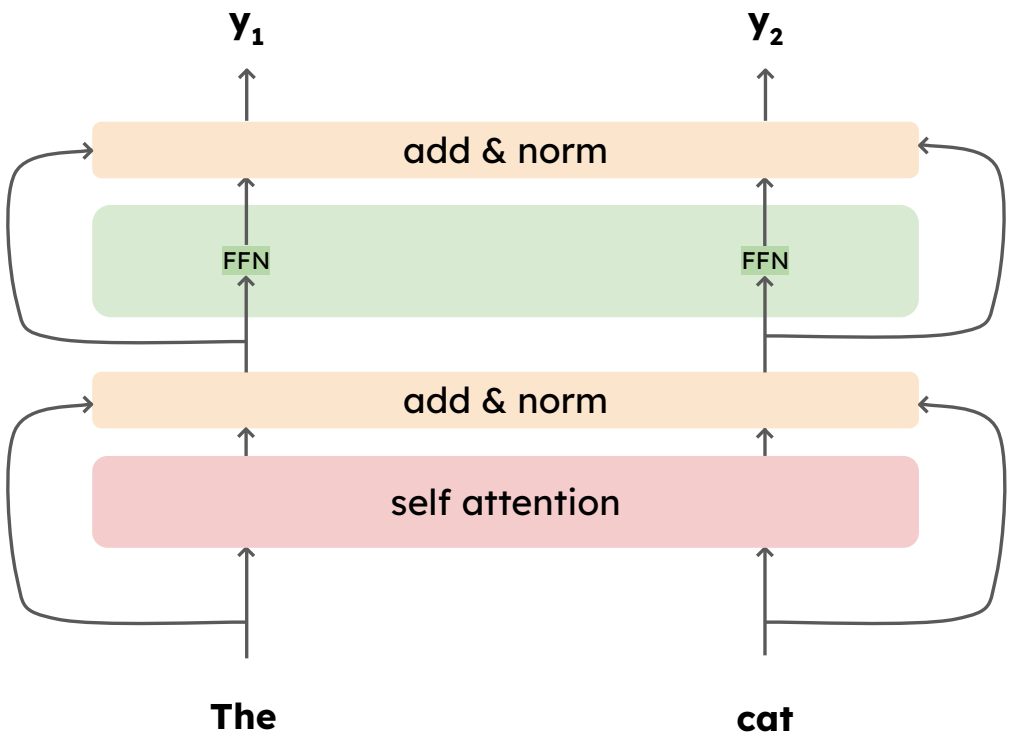
Gemini 3 Pro
Model Card

Architecture : Gemini 3 Pro is a sparse mixture-of-experts (MoE) (Clark et al., 2022; Du et al., 2021; Fedus et al., 2021; Jiang et al., 2024, Lepikhin et al., 2020; Riquelme et al., 2021; Roller et al., 2021; Shazeer et al., 2017) transformer-based models (Vaswani et al., 2017) with native multimodal support for text, vision, and audio inputs. Sparse MoE models activate a subset of model parameters per input token by learning to dynamically route tokens to a subset of parameters (experts); this allows them to decouple total model capacity from computation and serving cost per token. Developments to the model architecture contribute to the significantly improved performance from previous model families.

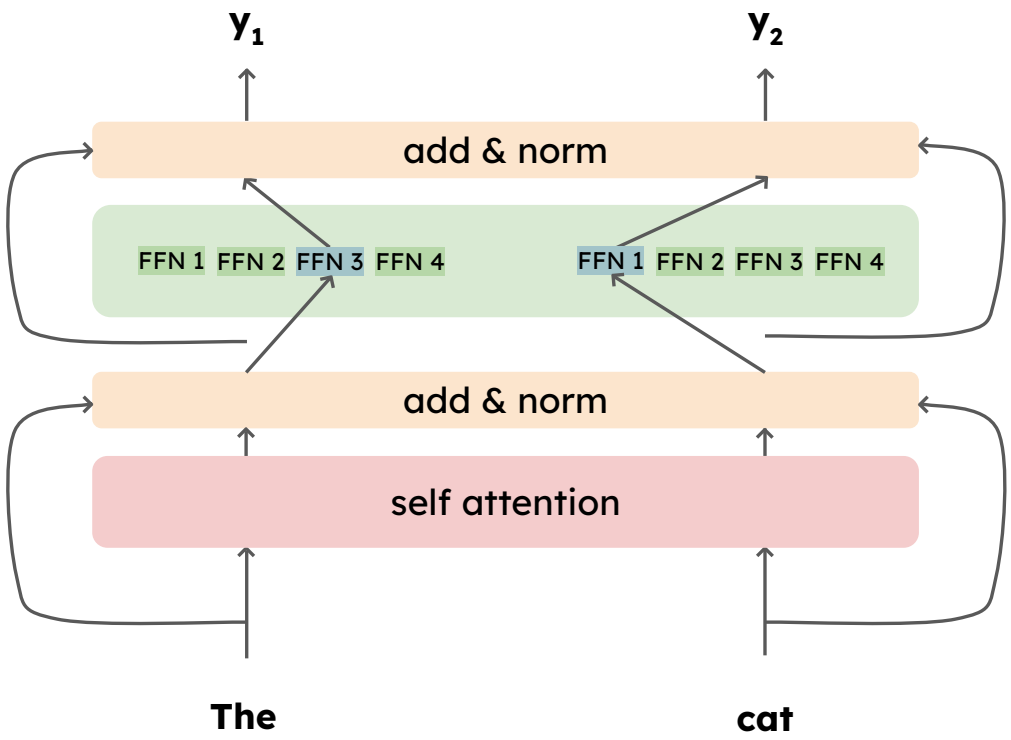
A dense model



A dense model



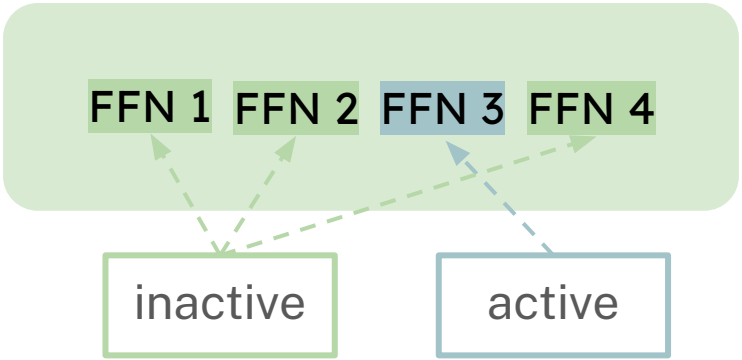
Mixture of experts



Mixture of experts

Key idea: **sparsity**

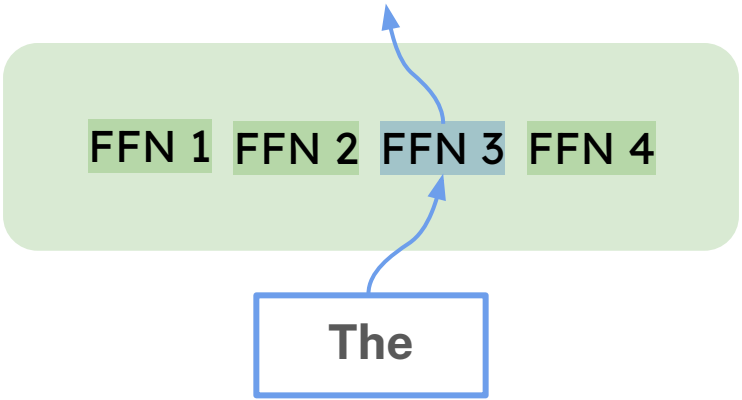
Our total number of parameters is high, but only a relatively smaller number of them are active during training and inference



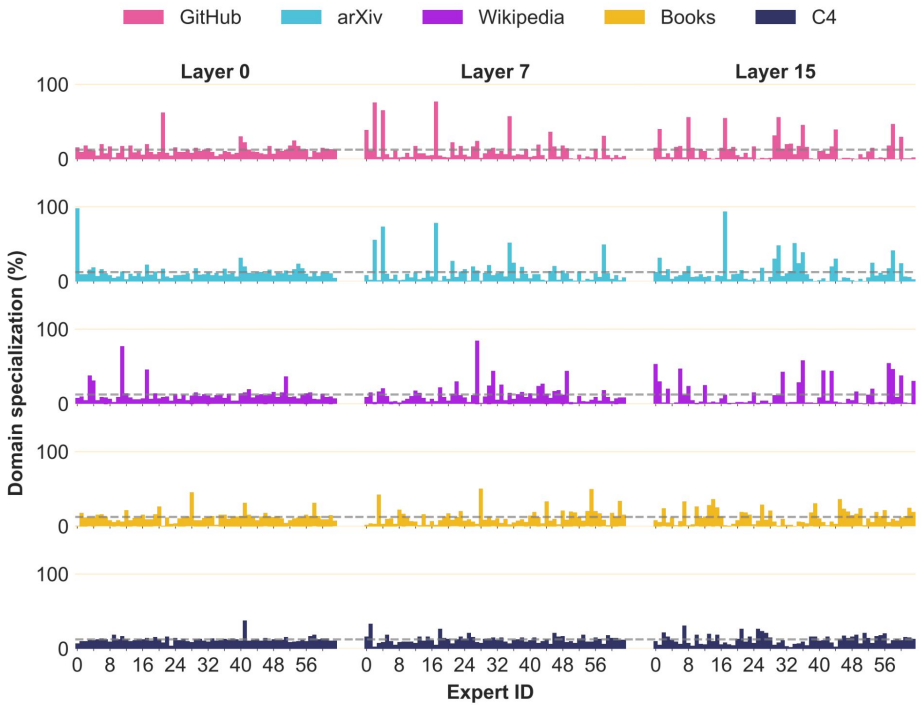
Mixture of experts

These FNN are “**experts**” that specialize in specific types of inputs or patterns.

A parameterized **router** chooses 1 or more FNNs to use for each token.



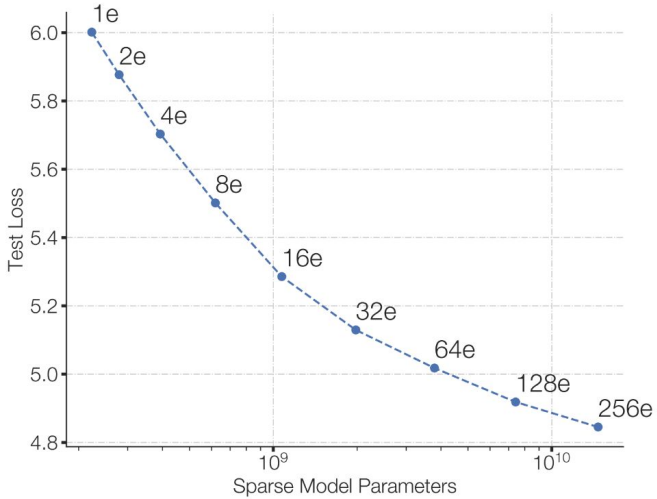
Domain specialization

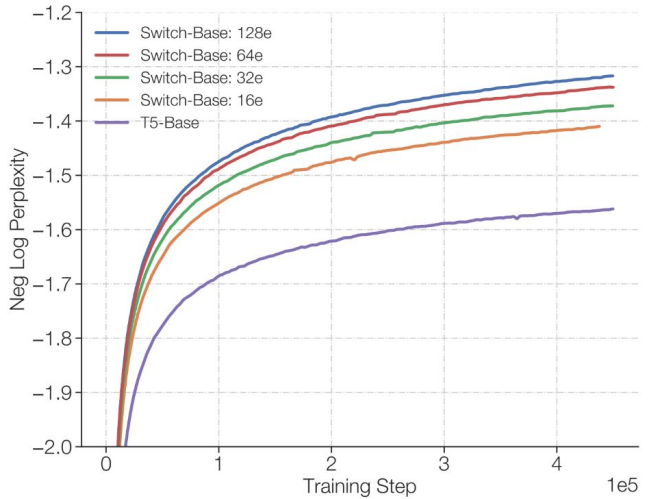


OLMoE paper

New dimension of scaling

Scale # of parameters, while keeping the amount of FLOPs per example fixed.

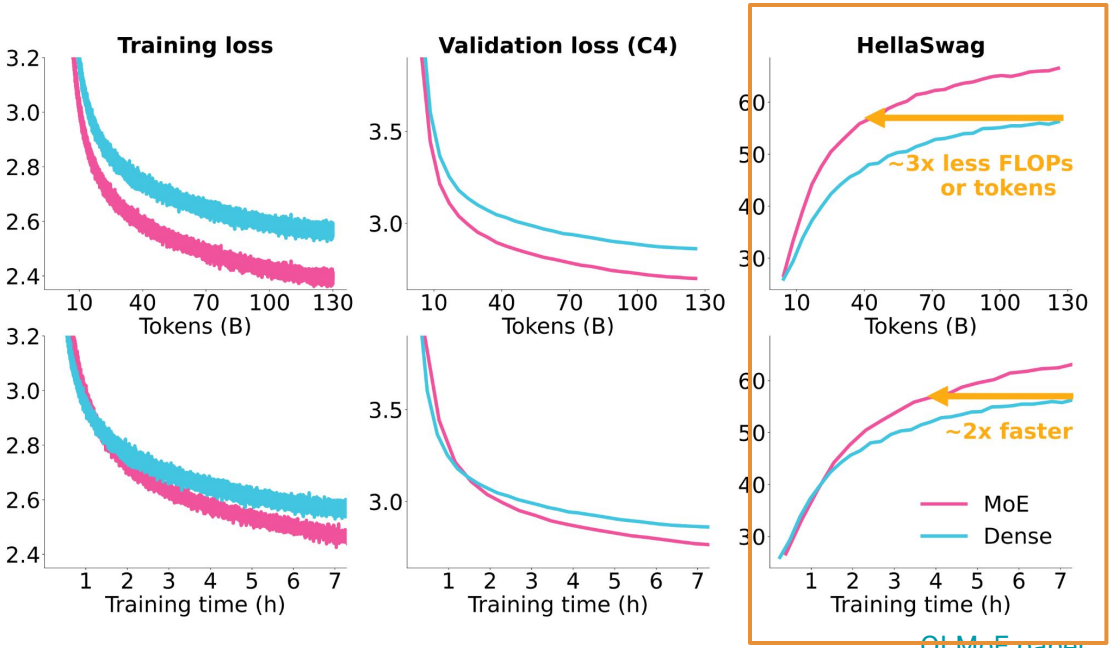




Fedus et al. 2022

More efficient training

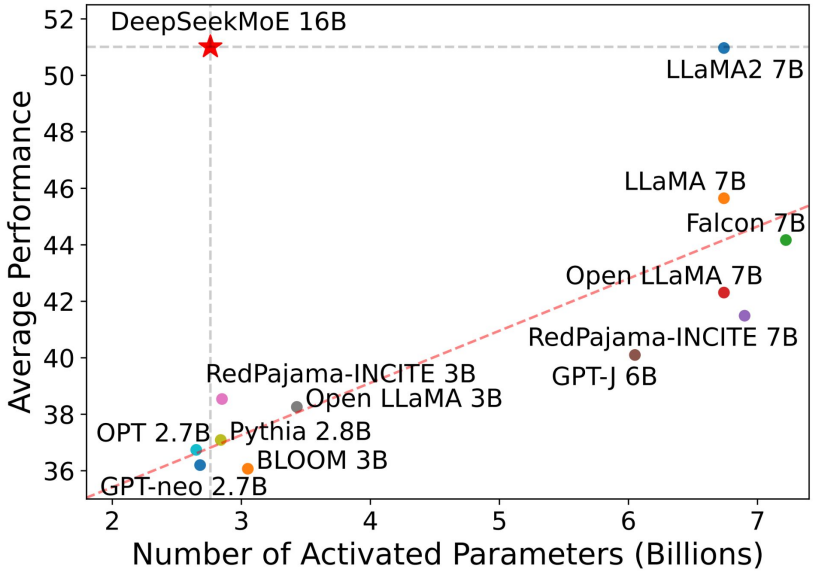
Fewer FLOPs and less time needed to reach a given level of performance.



OLMoE paper

More efficient inference

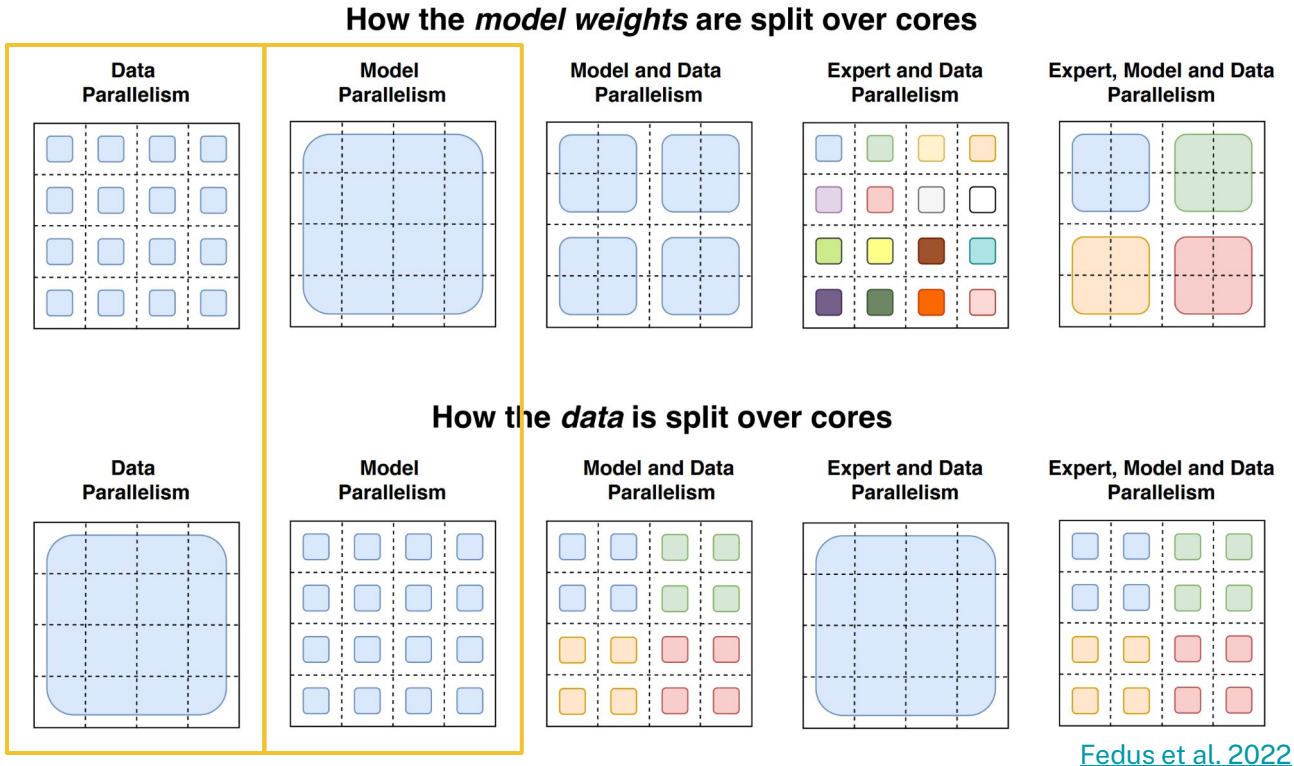
Better inference-cost vs. performance tradeoff.



DeepSeekMoE paper

Enables all sorts of parallelism

16 GPUs

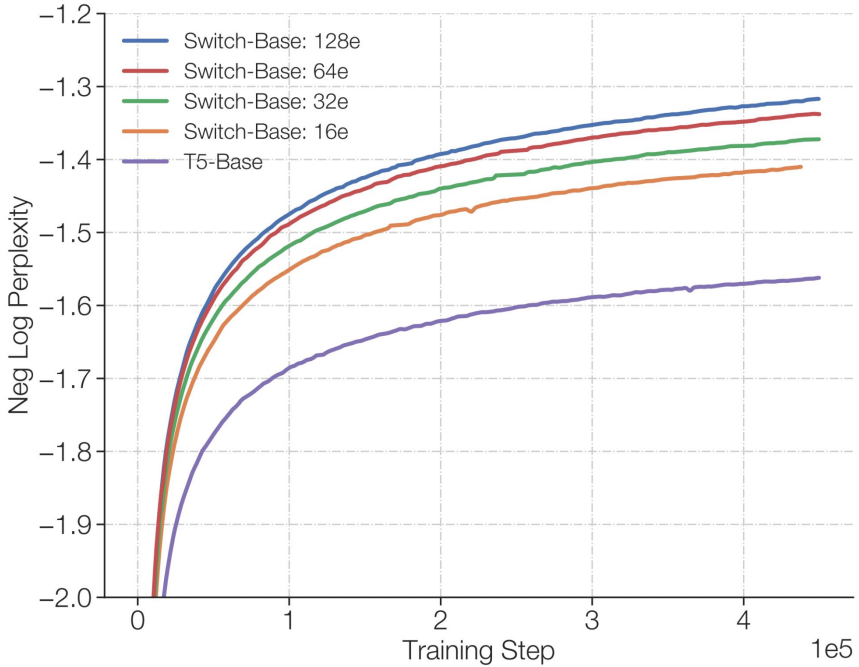


Fedus et al. 2022

Design decision: how many experts?

Strategy 1: Add more experts of the same size, increasing the number of parameters.

More is better, but with diminishing returns plus extra compute cost.

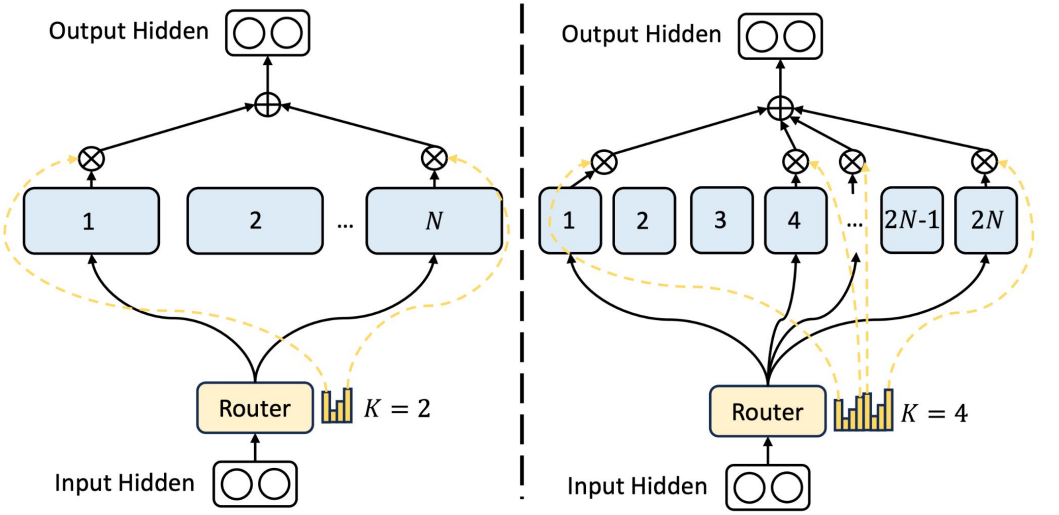


Fedus et al. 2022

Design decision: how many experts?

Strategy 2: Increase the number of experts, but fix the total number of parameters. Instead, split the hidden dimension among experts.

8 experts per layer of dimension *d*, **2 activated** per token.



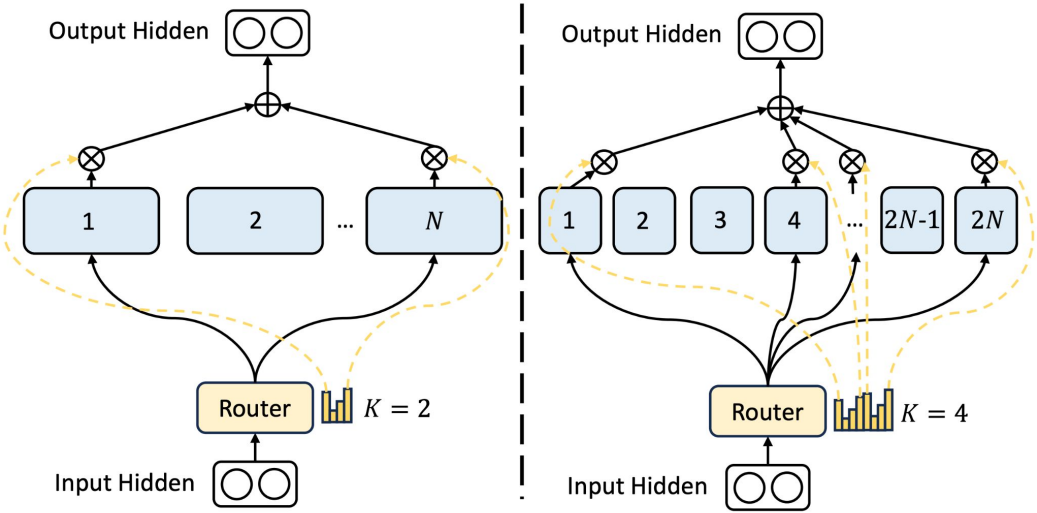
16 experts per layer of dimension *d*/2, **4 activated** per token.

DeepSeekMoE paper

Design decision: how many experts?

Strategy 2: Increase the number of experts, but fix the total number of parameters. Instead, split the hidden dimension among experts.

$\binom{8}{2}$



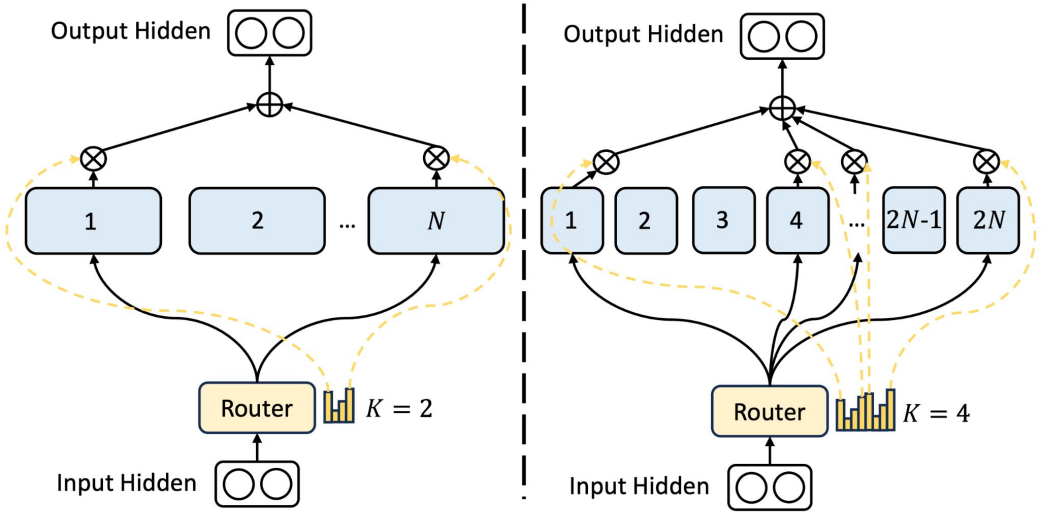
$\binom{16}{4}$

DeepSeekMoE paper

Design decision: how many experts?

Strategy 2: Increase the number of experts, but fix the total number of parameters. Instead, split the hidden dimension among experts.

28 possible combinations of experts per layer



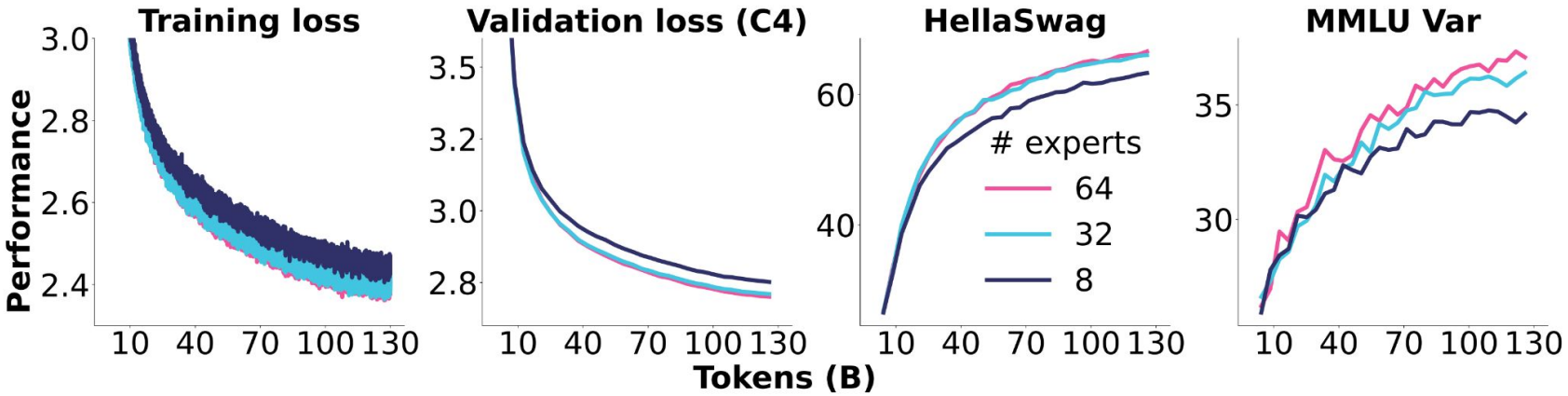
1,820 possible combinations of experts per layer.

More flexibility in expert mixes

DeepSeekMoE paper

Design decision: how many experts?

Diminishing returns in practice after a certain granularity



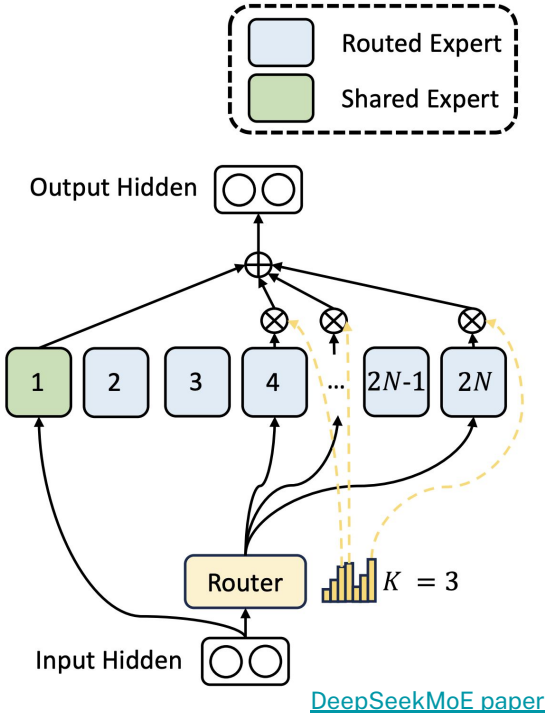
OLMoE paper

Design decision: do we share experts?

Shared experts are a fixed set of experts that are always active

- Example: 32 experts total, 1 shared and 3 selected by routing from the other 31.

Intention: shared expert learns common information, reducing redundancy among the experts.

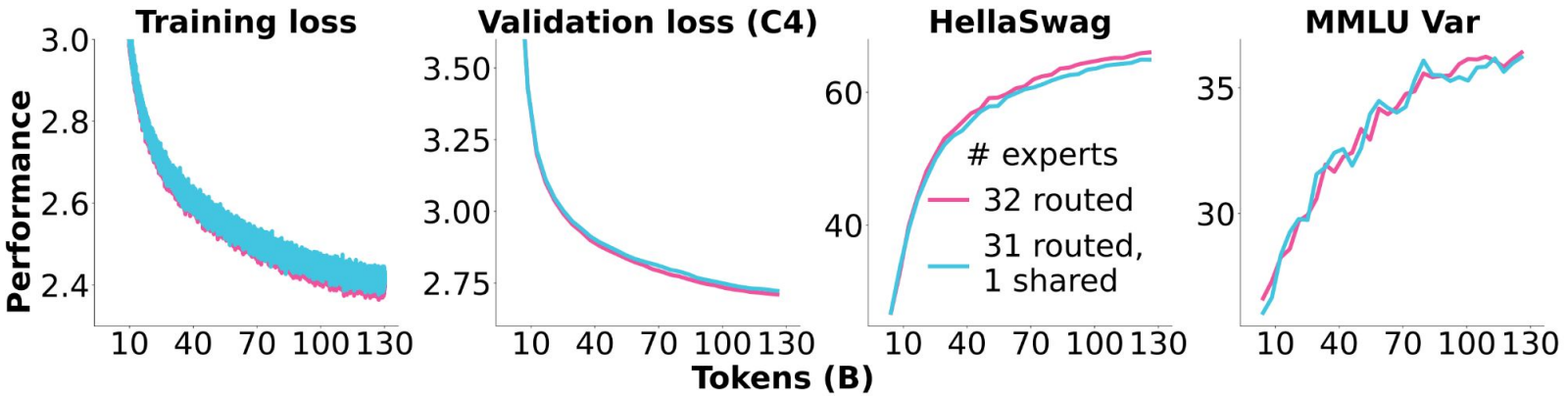


DeepSeekMoE paper

Design decision: do we share experts?

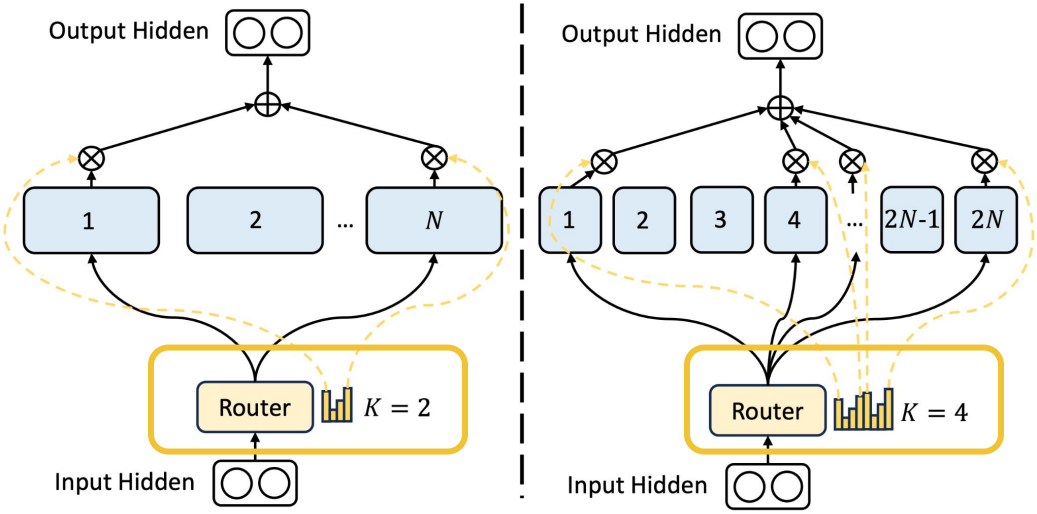
In practice, mixed results…

Turning one expert into a shared one eliminates ~90% of expert combos!



OLMoE paper

Routing

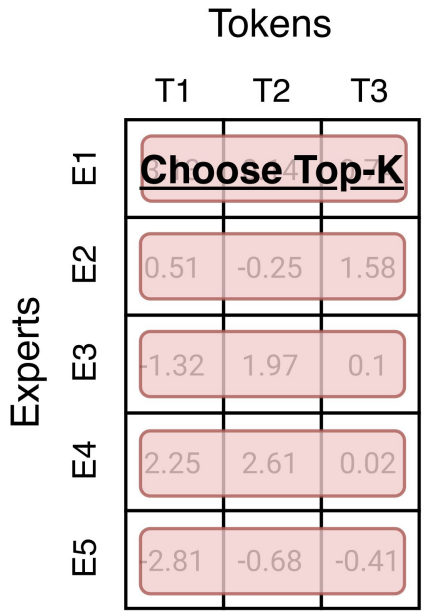

Output Hidden
1
2
...
N
Router
$K = 2$
Input Hidden
Output Hidden
1
2
3
4
...
2N-1
2N
Router
$K = 4$
Input Hidden

Routing

Compute a score for each (token, expert) pair:

- Introduce a weight vector for each expert
- Take dot products b/t each token's hidden state and each expert's weight vector.

Then, choose experts according to some strategy.



Each **expert** (row) chooses top-*k* tokens. Awk for generation...

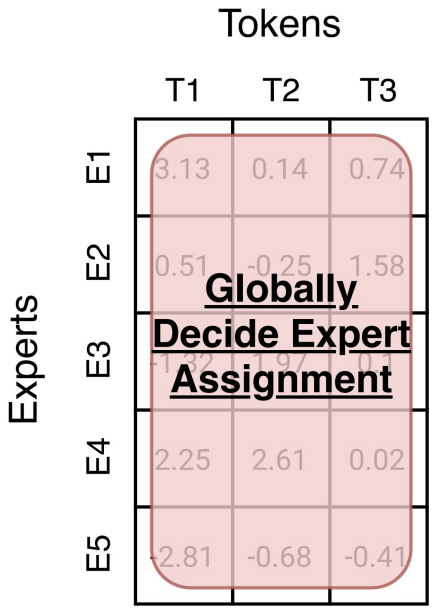
Fedus et al. 2022

Routing

Compute a score for each (token, expert) pair:

- Introduce a weight vector for each expert
- Take dot products b/t each token's hidden state and each expert's weight vector.

Then, choose experts according to some strategy.



Non-greedy assignment approaches. Requires extra compute ...

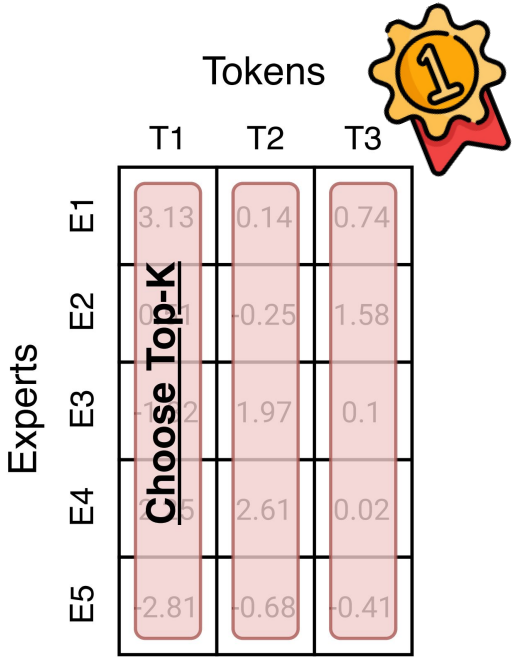
Fedus et al. 2022

Routing

Compute a score for each (token, expert) pair:

- Introduce a weight vector for each expert
- Take dot products b/t each token's hidden state and each expert's weight vector.

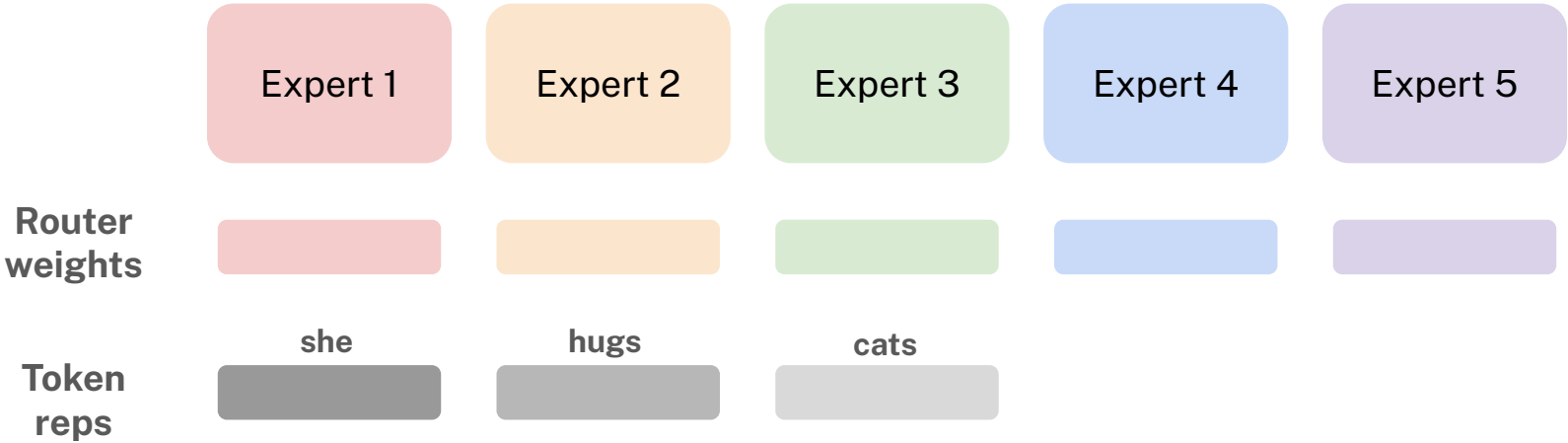
Then, choose experts according to some strategy.



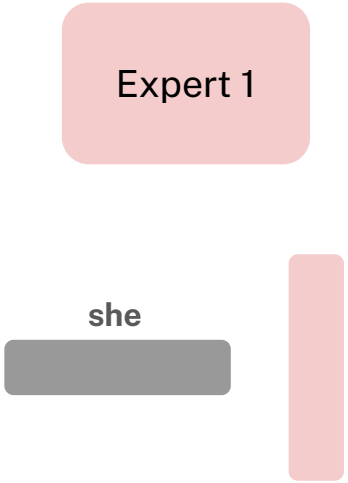
Each **token** (column) chooses top-*k* experts.

Fedus et al. 2022

Routing, illustrated

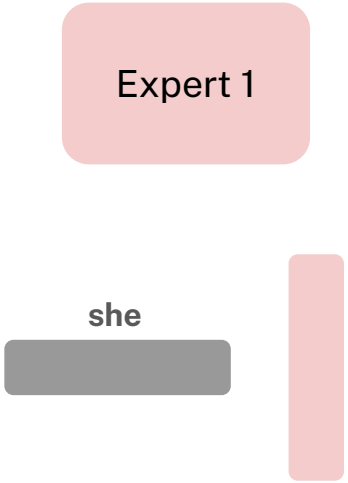


Routing, illustrated



	she	hugs	cats
E1	3.13		
E2			
E3			
E4			
E5			

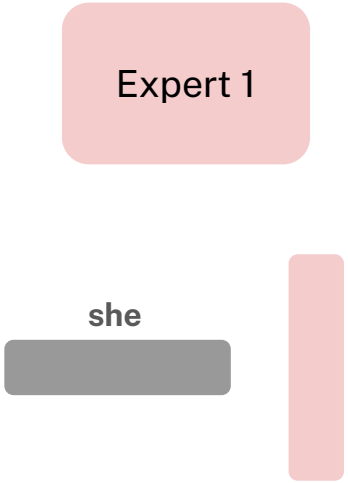
Routing, illustrated



	she	hugs	cats
E1	3.13	0.14	0.74
E2	0.51	-0.25	1.58
E3	-1.32	1.97	0.10
E4	2.25	2.61	0.02
E5	-2.81	-0.68	-0.41

Numbers from Fedus et al. 2022

Routing, illustrated



	she	hugs	cats
E1	0.67	0.05	0.22
E2	0.05	0.03	0.5
E3	0.01	0.31	0.11
E4	0.27	0.59	0.10
E5	0.00	0.02	0.07

Softmax (normalize) each column!

Numbers from Fedus et al. 2022

Routing, illustrated

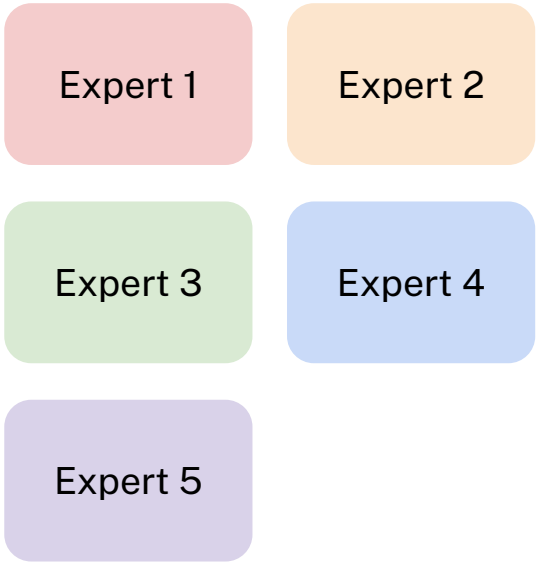


	she	hugs	cats
E1	0.67	0.05	0.22
E2	0.05	0.03	0.5
E3	0.01	0.31	0.11
E4	0.27	0.59	0.10
E5	0.00	0.02	0.07

Token choice
top-1 routing

Numbers from Fedus et al. 2022

Routing, illustrated



If E1 gets too many, we risk token dropping!

	she	hugs	cats
E1	0.67	0.05	0.22
E2	0.05	0.03	0.5
E3	0.01	0.31	0.11
E4	0.27	0.59	0.10
E5	0.00	0.02	0.07

Token choice **top-2** routing

Our allocation is a bit unbalanced! 🤔

Unused capacity

Numbers from [Fedus et al. 2022]

How do we train MoEs?

Three losses:

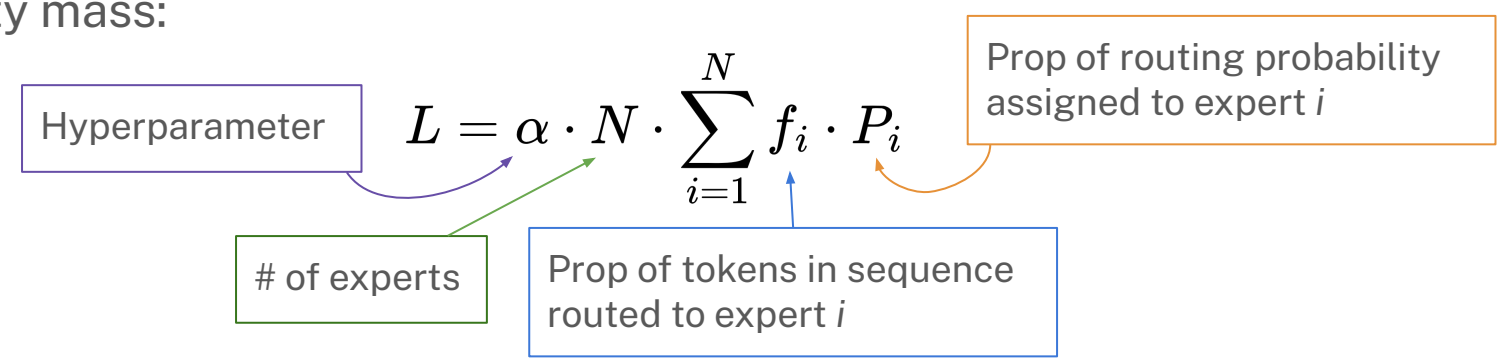
- Typical cross-entropy language modeling loss
- Heuristic balancing loss
- Router z-loss for stability

Heuristic balancing loss

The router only gets gradients from the experts it actually picked.

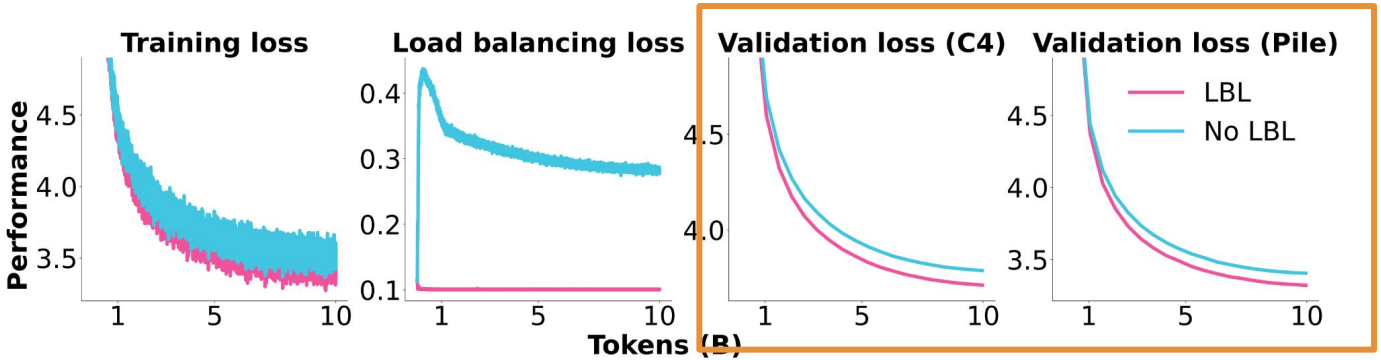
This can lead to a rich-get-richer unevenness: a few experts get chosen early, improve, get chosen more, and the rest are idle!

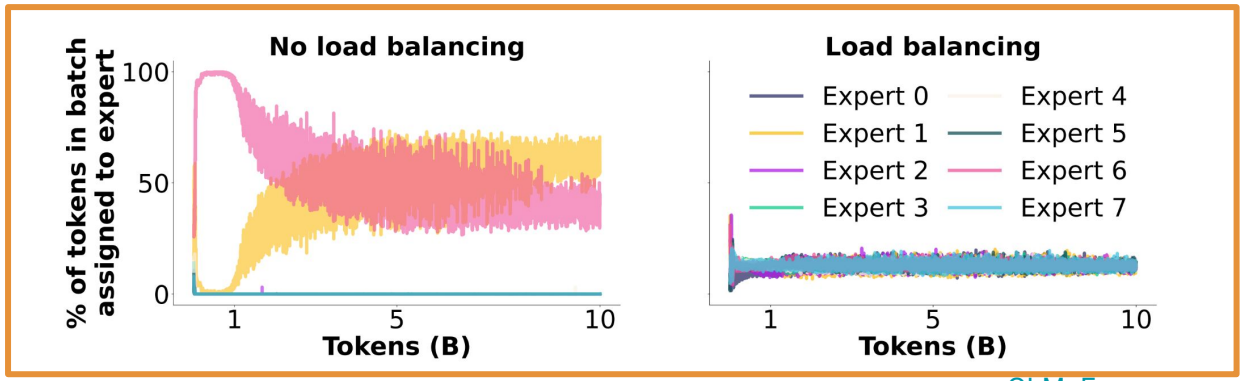
Penalize an expert if it receives too many tokens and/or gets too much probability mass:



Fedus et al. 2022

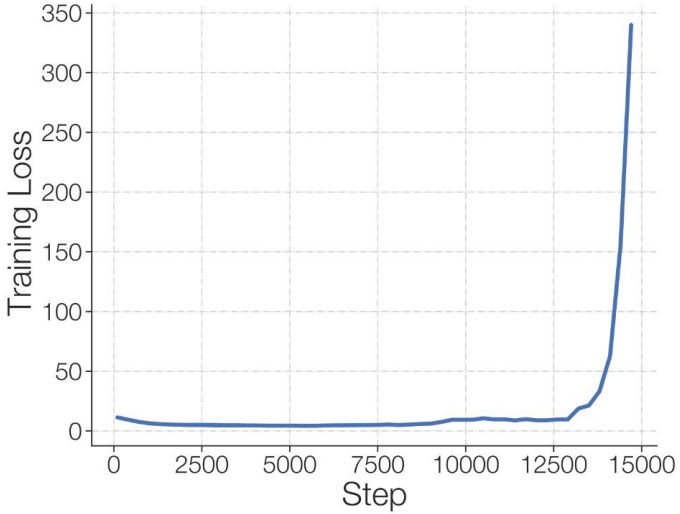
Heuristic balancing loss

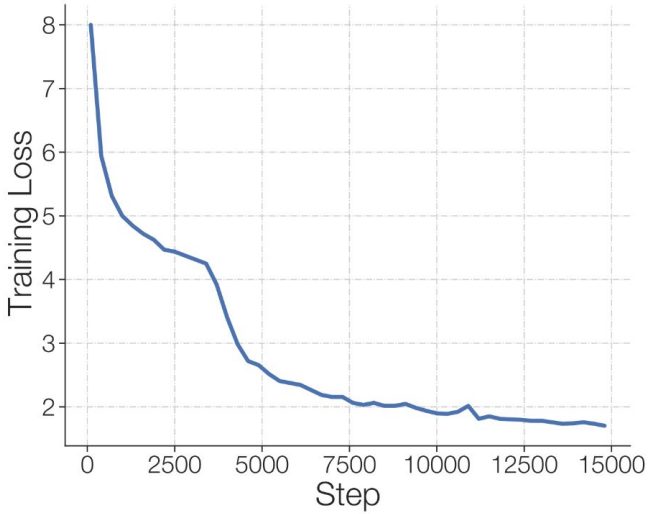




OLMoE paper

Training MoEs can be unstable



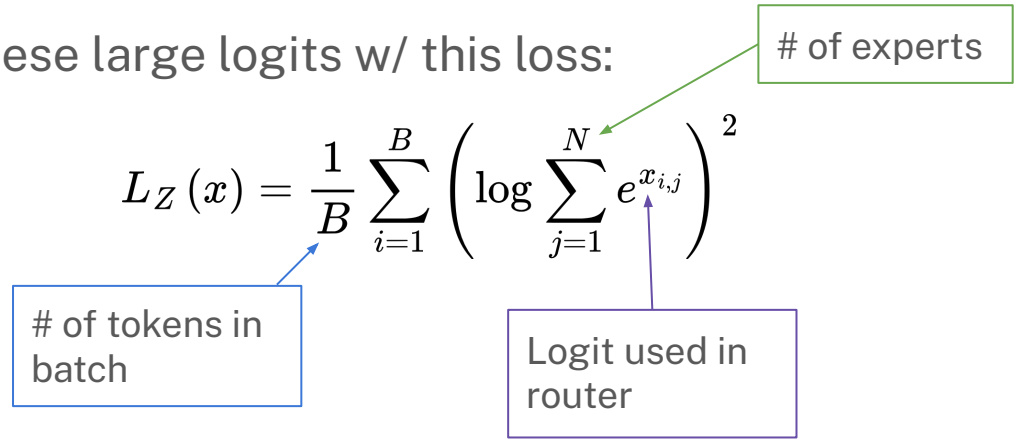


Zoph et al. 2022

Router z-loss for stability

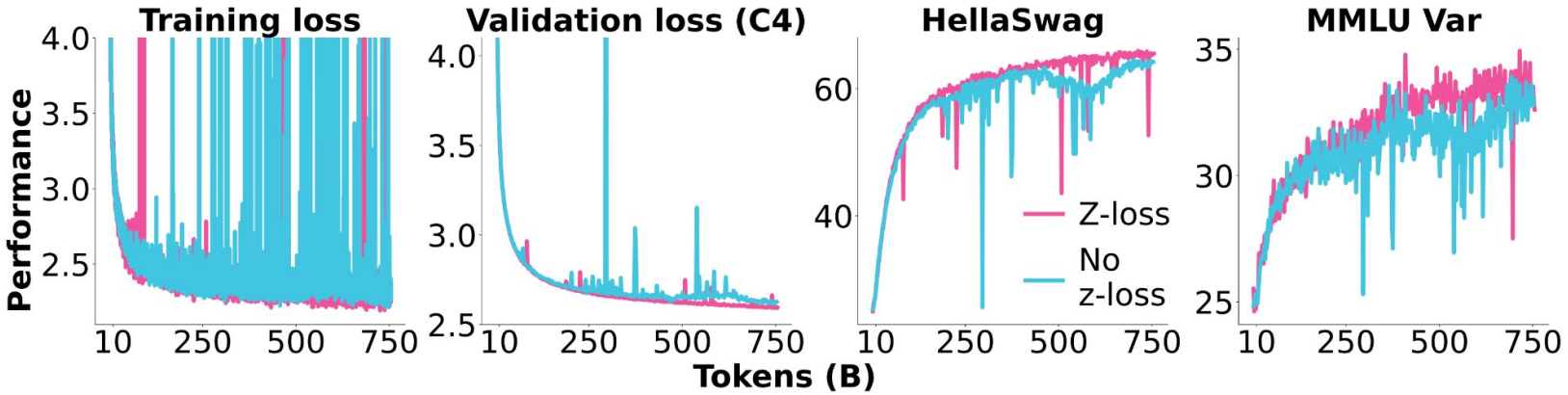
Large logits (e.g. strongly preferring an expert) can make training unstable, cause numerical precision issues, and leave the router stuck in overconfident decisions.

We can penalize these large logits w/ this loss:



Zoph et al. 2022

Router z-loss for stability



OLMoE paper

Example of MoEs in use: FlexOlmo

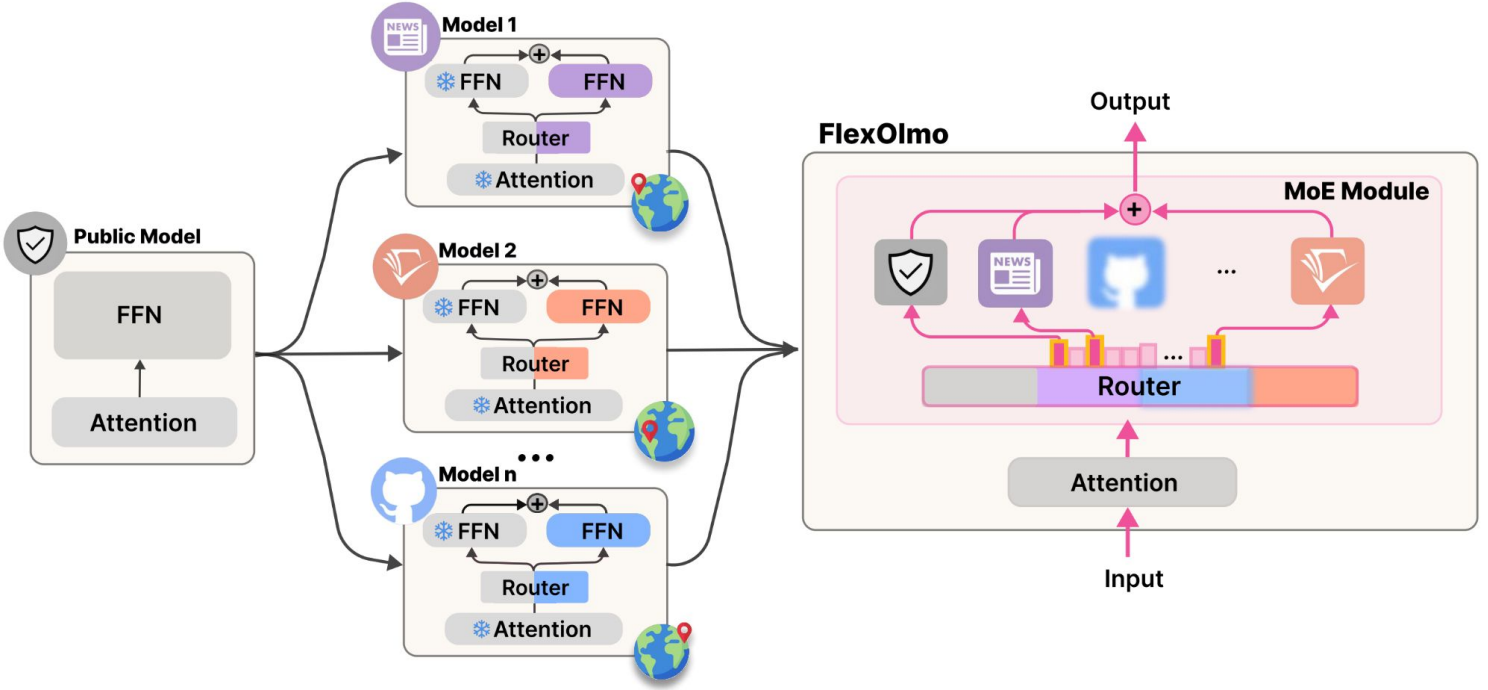
Motivation: Data owners are frequently hesitant or unable to use their data for model training. They want flexibility, control, and attribution.

How can MoEs enable data owners to **contribute to an LM without directly sharing their raw data**?

FlexOlmo allows a data owner to:

- locally branch from a shared public model
- add an expert trained on their data locally
- contribute their expert back to the shared model

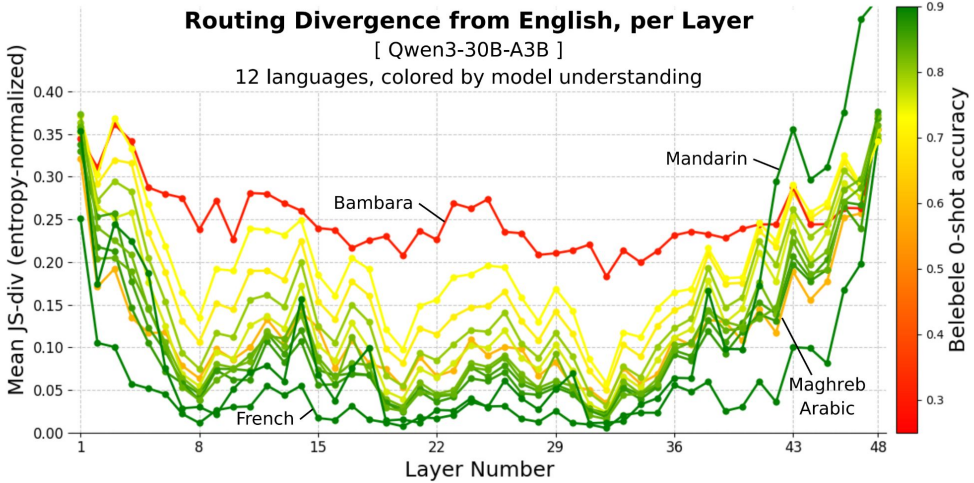
Example of MoEs in use: FlexOlmo



FlexOlmo paper

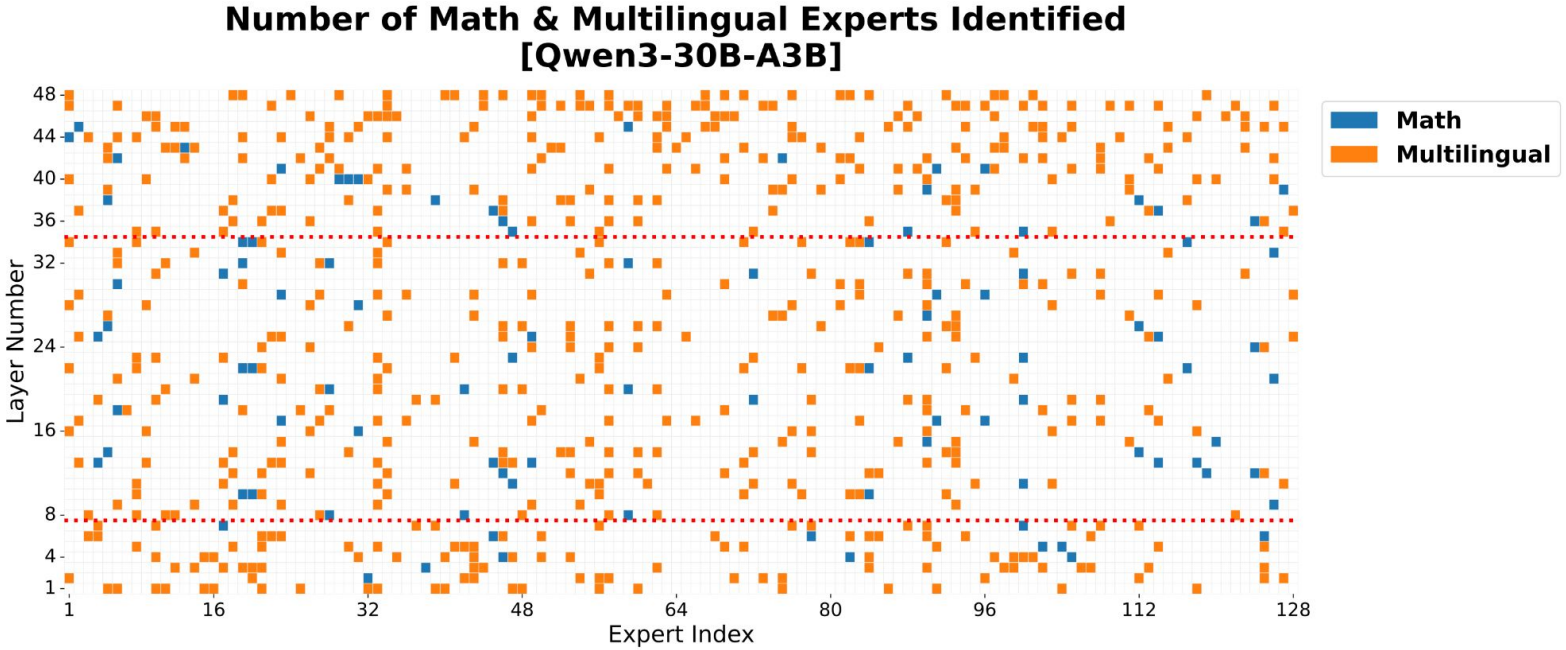
MoEs & multilinguality

MoE models route tokens in language-specific ways in early and late layers, and route cross-lingually in middle layers.



[Bandarkar et al. 2026]

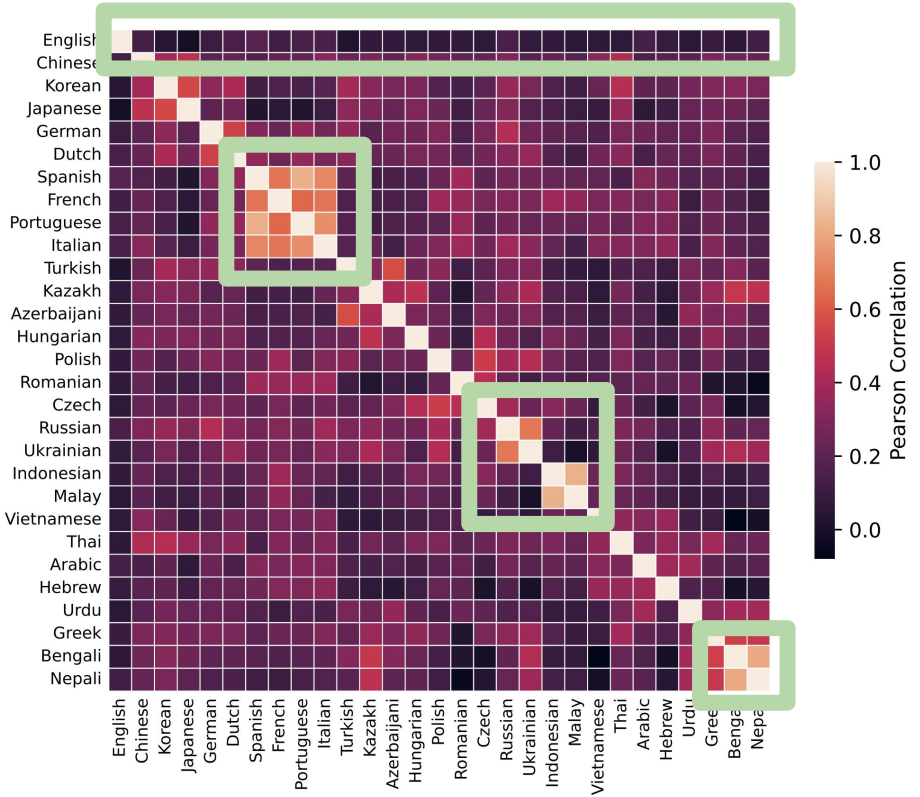
MoEs & multilinguality



Bandarkar et al. 2026

MoEs & multilinguality

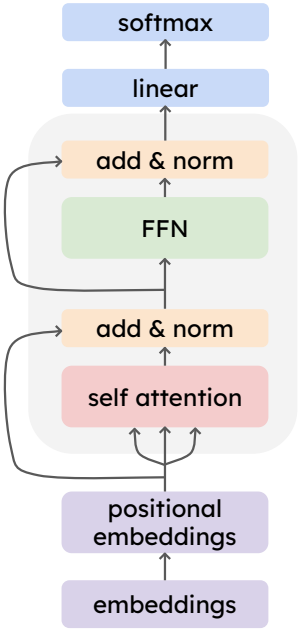
MoE expert activation patterns also relate to language similarity. (Jiang et al. 2026)



Today's agenda ✔

Two model "upgrades":

- Scaling sequence length
- Mixture of experts



a transformer

Logistics

Midterm presentations due 11:59 pm *before* Monday lecture!

Another round of peer feedback after presentations.

Possibly helpful for your final report: https://aclrollingreview.org/reviewerguidelines#paper-issues

🆕 4. Check for common problems in NLP papers

As a counter to common problems with reviews, there are also common problems with NLP papers. We do ask you to watch out for these and point them out when you see them. Above all else, published research should at least be technically sound and appropriately scoped. As with the above reviewer issues, it's a case-by-case evaluation: some things in the list below may be appropriate in a given context, given that they are sufficiently motivated. We provide codes for different types of issues (e.g. M1, G2...) that can be referenced in discussions.

Issues with methodology (M)	
M1. LLM-only evaluation without validation	If LLMs are used as automated evaluators, is their reliability *in this context* sufficiently validated?
M2. Reproducibility issues	Are there enough details to reproduce the experiments, including hyperparameter selection? Is it clear whether the code/data/etc will be publicly available? If not, is that sufficiently motivated? If so, are they at least provided for reviewer evaluation? If you can't find this in the paper, check the authors' answers for the Responsible NLP checklist. You are not obligated to run the code, especially if it's computationally expensive, but you are welcome to do this to substantiate your assessment of the paper. Note: reproducibility score refers to *reproducibility* (being able to run the same experiment with roughly the same results) rather than *replicability* (getting similar results after reimplementation or on different models/data). Since reviewers are not obligated to run anything, the scope of ARR reproducibility score is to at least assess whether enough details/code is provided for reproducibility.
M3. Undisclosed data quality issues	The papers that provide resources may be accompanied by full datasets or samples. You are *not* obligated to check this data. But you are welcome to, and if you notice issues that are not disclosed/discussed in the paper — this is a serious problem to bring up.
M4. Unmotivated selection	Any paper will have a limited number of models and benchmarks. But it should be clear why this sample was selected, and that motivation should be directly linked to the scope of the claimed contributions. You *may* feel that you would have selected a different sample, but that's ok, as long as the sample that the authors selected is appropriate for their claims.