Mixture-of-experts Models

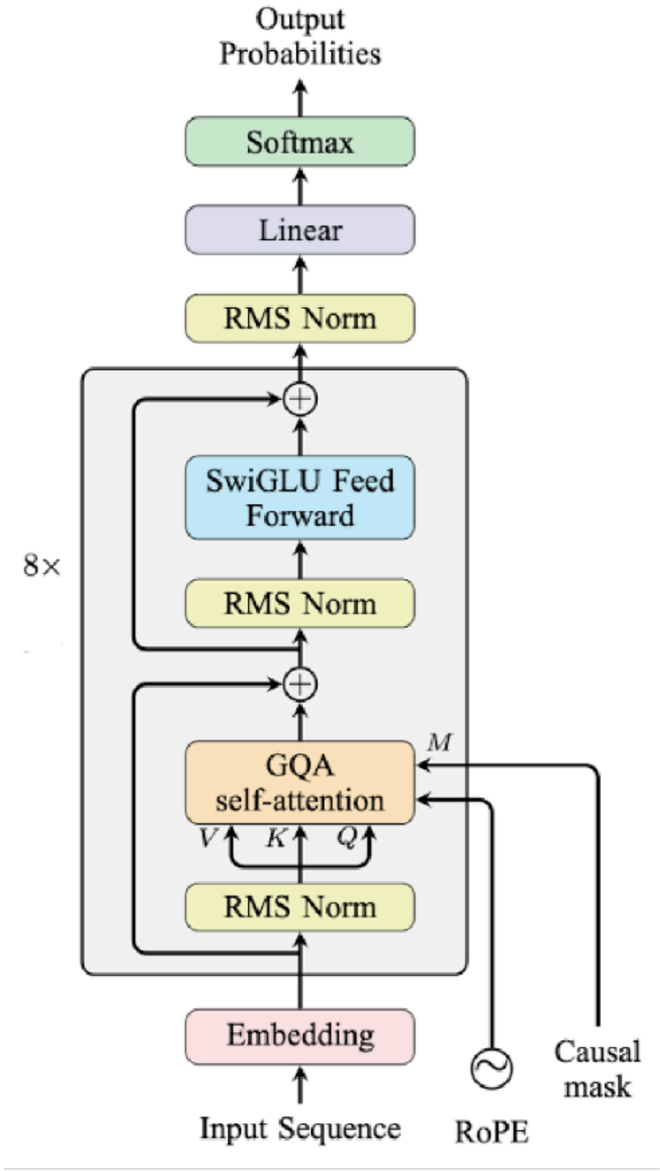
Aaron Mueller
CS599 B1: Advanced Natural Language Processing
Oct. 20, 2026

Administrivia

- The final project proposal is due **today** at 11:59pm!
- I will aim to get your quiz grades to you by the end of the week.
- I will prioritize grading the project proposals over HW2.
- Our request for more disk space on the SCC was approved! We now have 5TB (≈161 GB per student).
    - (Bigger group sizes mean more space per project)
- Heads-up: I will be away next week (Oct. 26 - Oct. 30) for EMNLP
    - No office hours
    - Pre-recorded lecture for Tue.
    - No content/pre-recorded lecture for Thu. Enjoy the small break!
    - Feel free to use Piazza for questions!
    - I *might* be able to do ad hoc meetings if needed, but it may be at early hours ET

Review: The Transformer Block

- Transformers are quite efficient when it comes to training time, but less so when it comes to memory
- Maybe there are more efficient ways to use or distribute computation?
- Maybe we can use more parameters during training than we do during inference?



Efficiency

- As researchers, engineers, and users, we would like to be able to use the best possible models despite not having OpenAI money.

- *How can we get the best model for our hardware?*

- We'll explore two strategies this week:
  - Today: mixture-of-experts models
  - Thursday: quantization

The Feed-forward Layer

It's common for more than half the parameters in a model to be in the feed-forward layers of a Transformer.

- Embedding:

$$\text{vocab_size} \times d_{model}$$

- Attention:

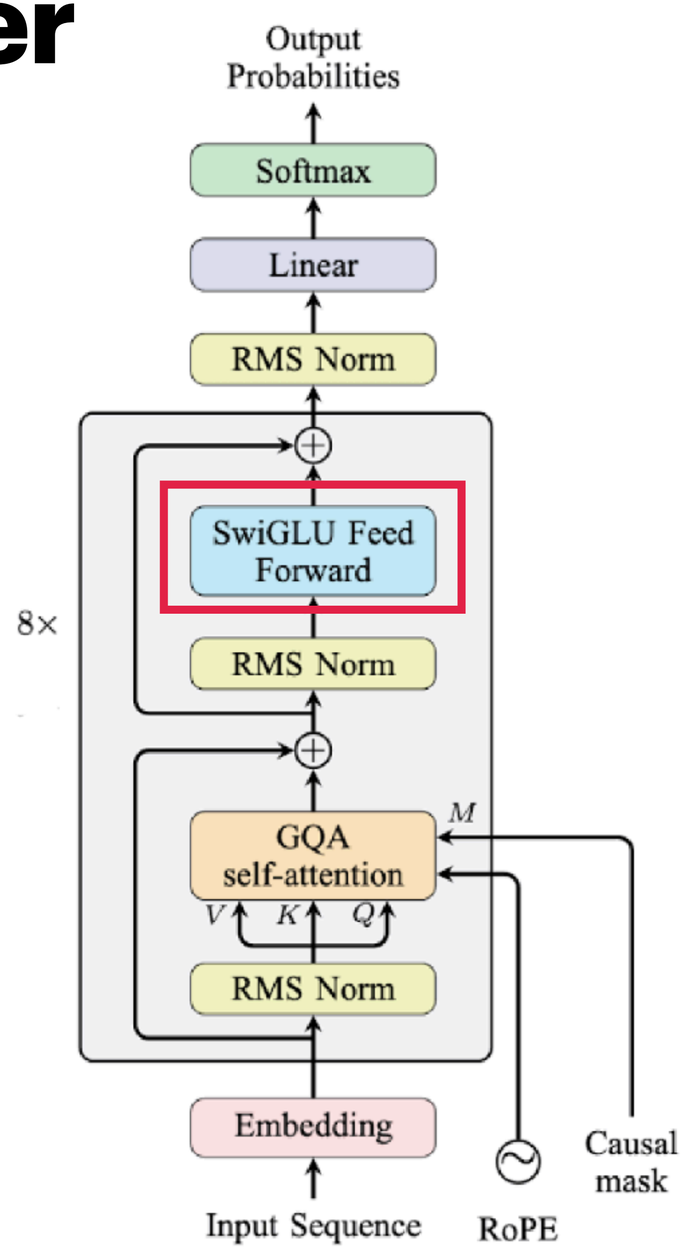
$$4((d_{model})^2 + d_{model}) \times n_{layers}$$

- Feed-forward:

$$(3d_{model} \times d_{ff} + d_{model} + d_{ff}) \times n_{layers}$$

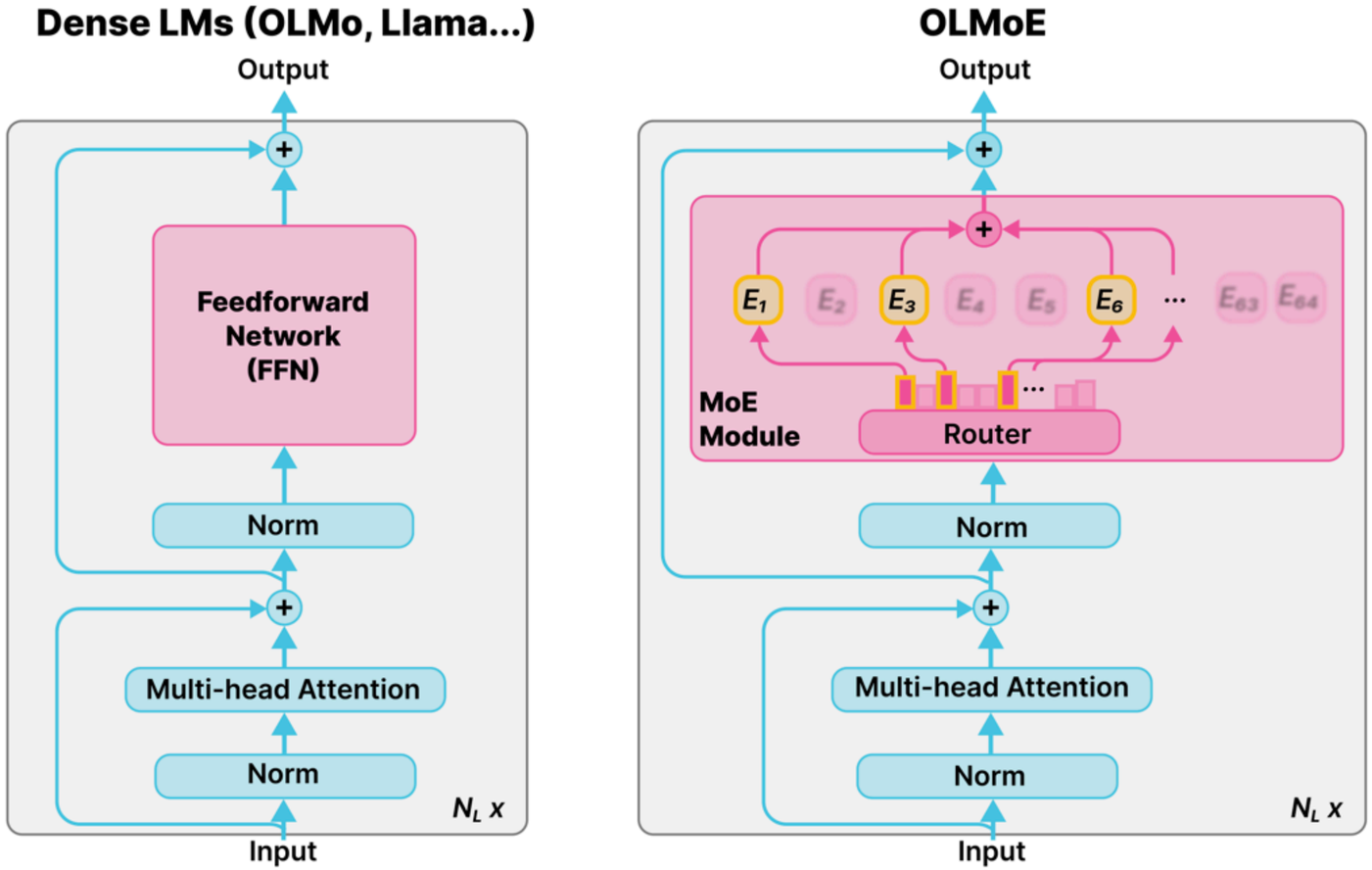
Qwen3.5-27B

d_{model}	5,120
d_{ff}	23,040
n_{layers}	48
vocab_size	248,320



Mixture-of-experts (MoE) Models

- Idea: what if we distribute the parameters across a set of specialized feed-forward layers?
- We can use a **mixture-of-experts (MoE)** architecture
- Contains a **router** that decides which expert(s) to use for a given input
- Each **expert** is a feed-forward layer



[Muenninghoff et al., 2025]

Dense MoE

In **dense MoE**, every expert gets some possibly non-zero weight from the router:

Layer Output

$$\mathbf{y} = \sum_{i=1}^{N} G(\mathbf{x})_i E_i(\mathbf{x}) \qquad G(\mathbf{x})_i \in \mathbb{R} \qquad E_i(\mathbf{x}) \in \mathbb{R}^{d_{model}}$$

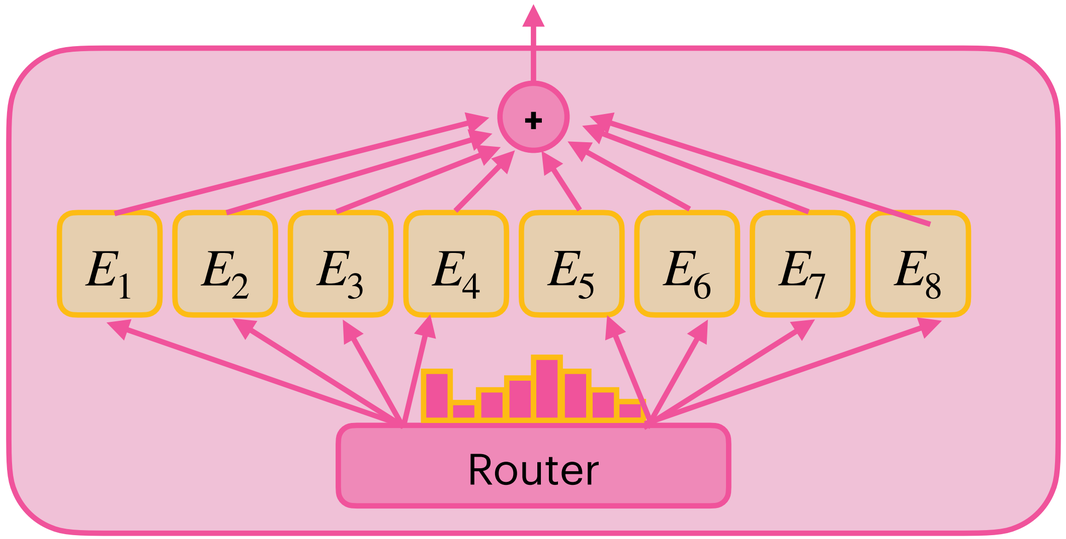
Router

$$G(\mathbf{x}) = \text{softmax}(\mathbf{x} \cdot W_g)$$

Expert

$$E_i(\mathbf{x}) = \text{SwiGLU}(\mathbf{x})$$

$$= W_3(\text{Swish}(W_1\mathbf{x}) \odot W_2\mathbf{x})$$



Sparsely Gated MoE

In **sparsely gated MoE**, we only take the top-k experts:

Layer Output

$$\mathbf{y} = \sum_{i=1}^{N} G(\mathbf{x})_i E_i(\mathbf{x}) \qquad G(\mathbf{x})_i \in \mathbb{R} \qquad E_i(\mathbf{x}) \in \mathbb{R}^{d_{model}}$$

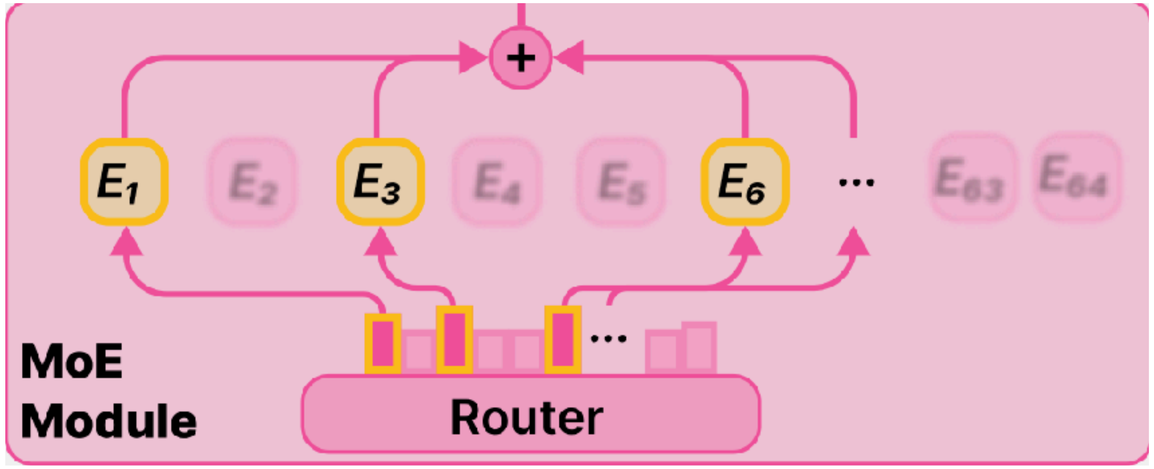
Router

$$G(\mathbf{x}) = \text{softmax}(\text{top-}k(\mathbf{x} \cdot W_g + \mathbf{b}_g))$$

Expert

$$E_i(\mathbf{x}) = \text{SwiGLU}(\mathbf{x})$$

$$= W_3(\text{Swish}(W_1\mathbf{x}) \odot W_2\mathbf{x})$$



[Muenninghoff et al., 2025]

Router Noise

- Current sparsely gated MoE models also tend to introduce a noise term into the router to prevent overreliance on a small number of experts early in training:

$$\mathbf{y} = \sum_{i=1}^{N} G(\mathbf{x})_i E_i(\mathbf{x})$$

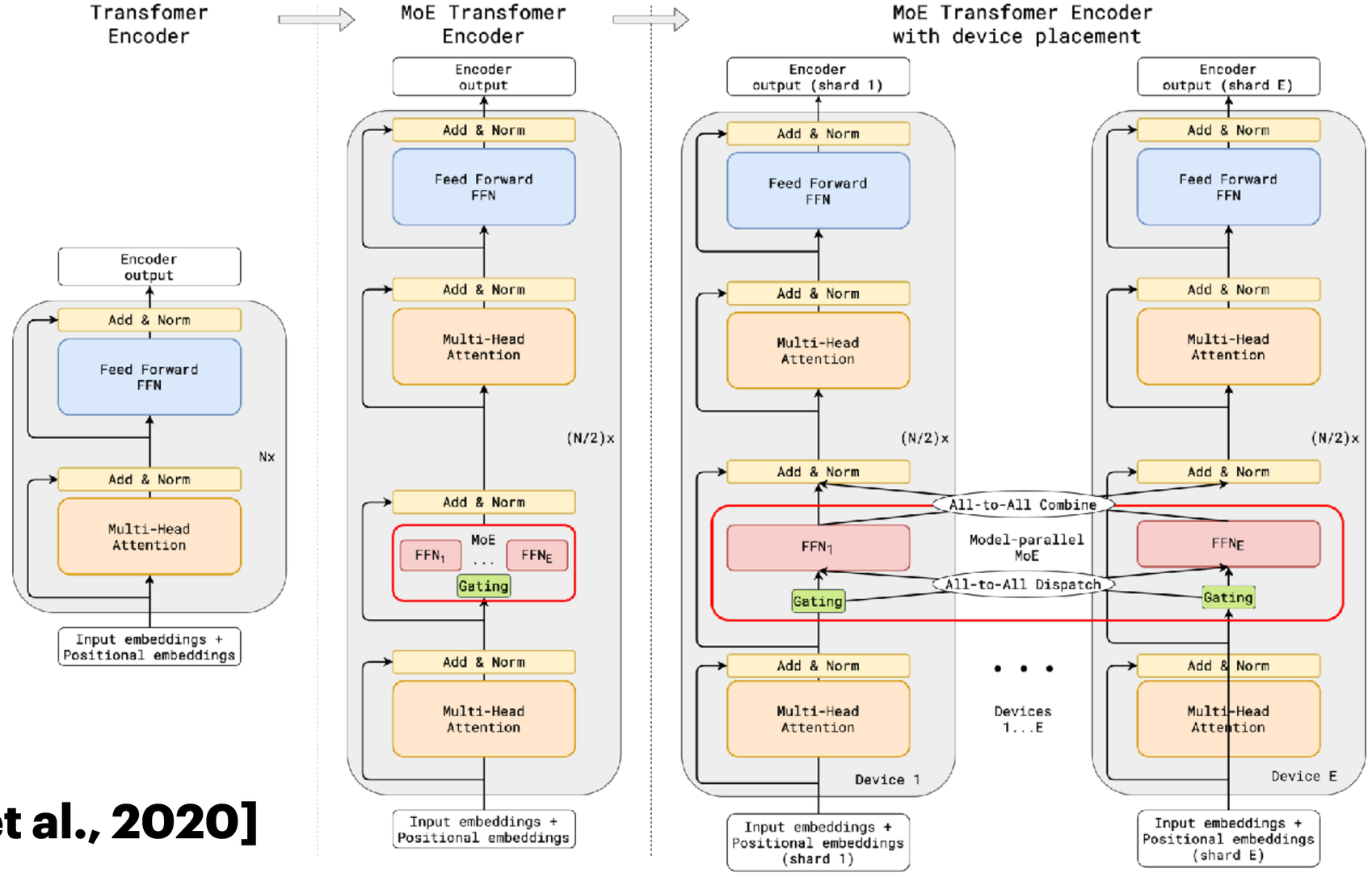
W_g and W_{noise} are initialized to be near zero. Provides basically no signal and a small amount of noise.

$$G(\mathbf{x}) = \text{softmax}(\text{top-}k(\mathbf{x} \cdot W_g + \mathbf{b}_g + \mathbf{r}_{\text{noise}}(\mathbf{x})))$$

$$\mathbf{r}_{\text{noise}}(\mathbf{x}) = \mathcal{N}(\mathbf{0}, I) \cdot \sigma(\mathbf{x} W_{\text{noise}} + \mathbf{b}_{\text{noise}})$$

Expert Parallelism

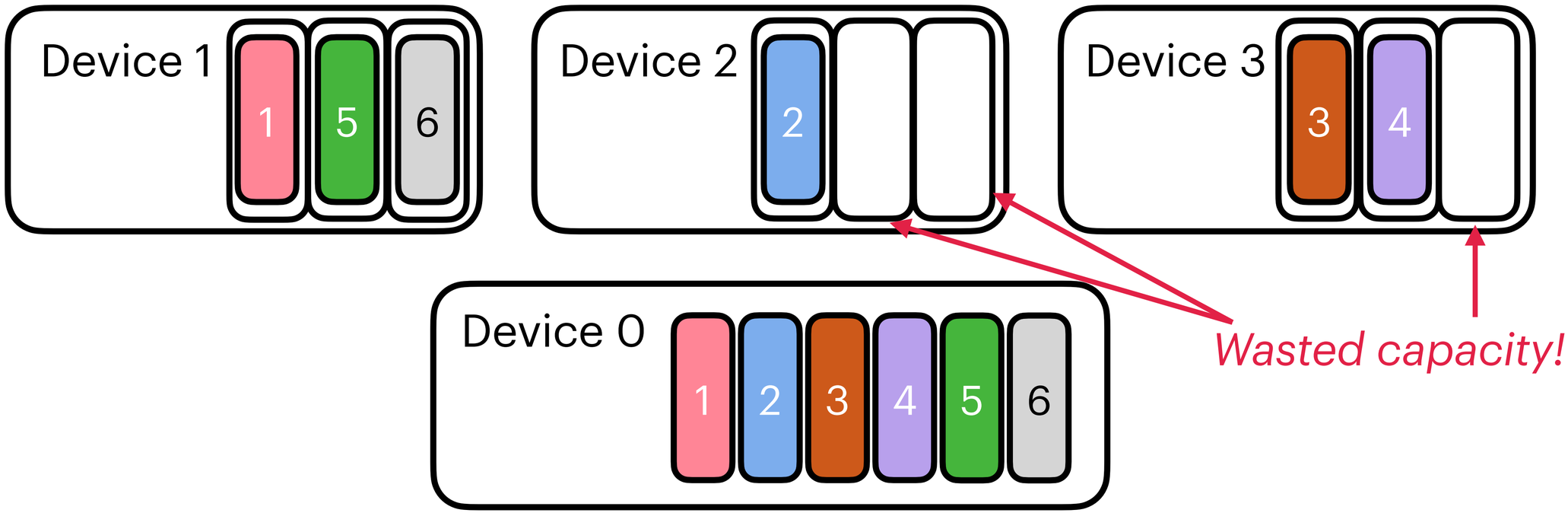
MoE models parallelize nicely: each expert can fit onto a separate device



[Lepikhin et al., 2020]

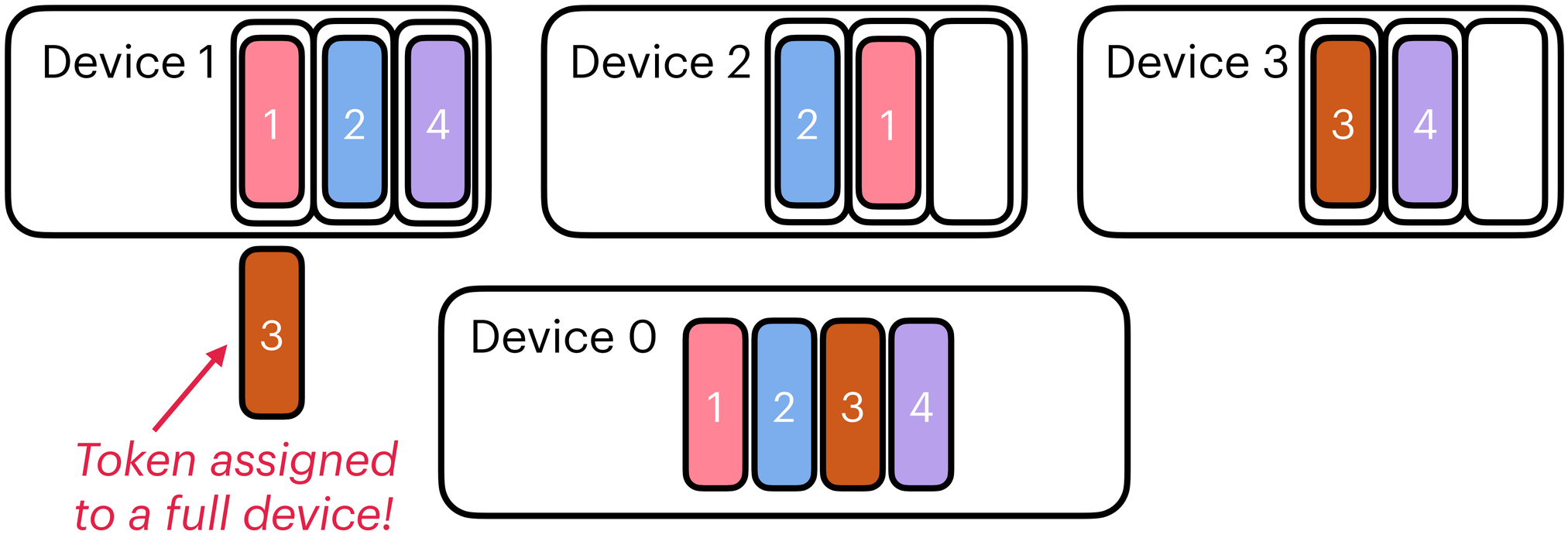
Expert Parallelism

- Each expert can be placed on a different GPU
  - The dense part of the model needs to be shared across all GPUs
- We can face *load balancing* issues if too many tokens are routed to a given expert
- Example: 3 devices, 3 tokens per device, 6 tokens total, 1 expert per token



Expert Parallelism

- Each expert can be placed on a different GPU
  - The dense part of the model needs to be shared across all GPUs
- We can face *load balancing* issues if too many tokens are routed to a given expert
- Example: 3 devices, 3 tokens per device, 4 tokens total, 2 experts per token



Expert Regularization

- **Problem:** the router tends to concentrate all of the signal on a small number of well performing experts early in training
  - Inefficient parameter utilization; poor load balancing
- **Solution:** Add a *load balance (LB) term* and a *router Z-loss (RZ) term* to the loss

$$L = L_{CE} + \alpha L_{LB} + \beta L_{RZ}$$

$$L_{LB} = N \sum_{i=1}^{N} f_i P_i$$

$$L_{RZ} = \frac{1}{T} \sum_{t=1}^{T} \left(\log \sum_{i=1}^{N} \exp(x_i^{(t)}) \right)^2$$

f_i: fraction of tokens routed to expert i

P_i: mean probability to E_i in current batch

$x_i^{(t)}$: router logit

Active Parameters

- In a sparsely gated MoE transformer, we typically choose a small k
  - Mixtral: $k = 2, N = 8$
  - OlMoE: $k = 8, N = 64$
- The number of **active parameters** is the count of parameters selected for computation by the router
- For an MoE model:
  - GPU memory is proportional to the number of *total* parameters (N)
  - FLOPs is proportional to number of *active* parameters (k)

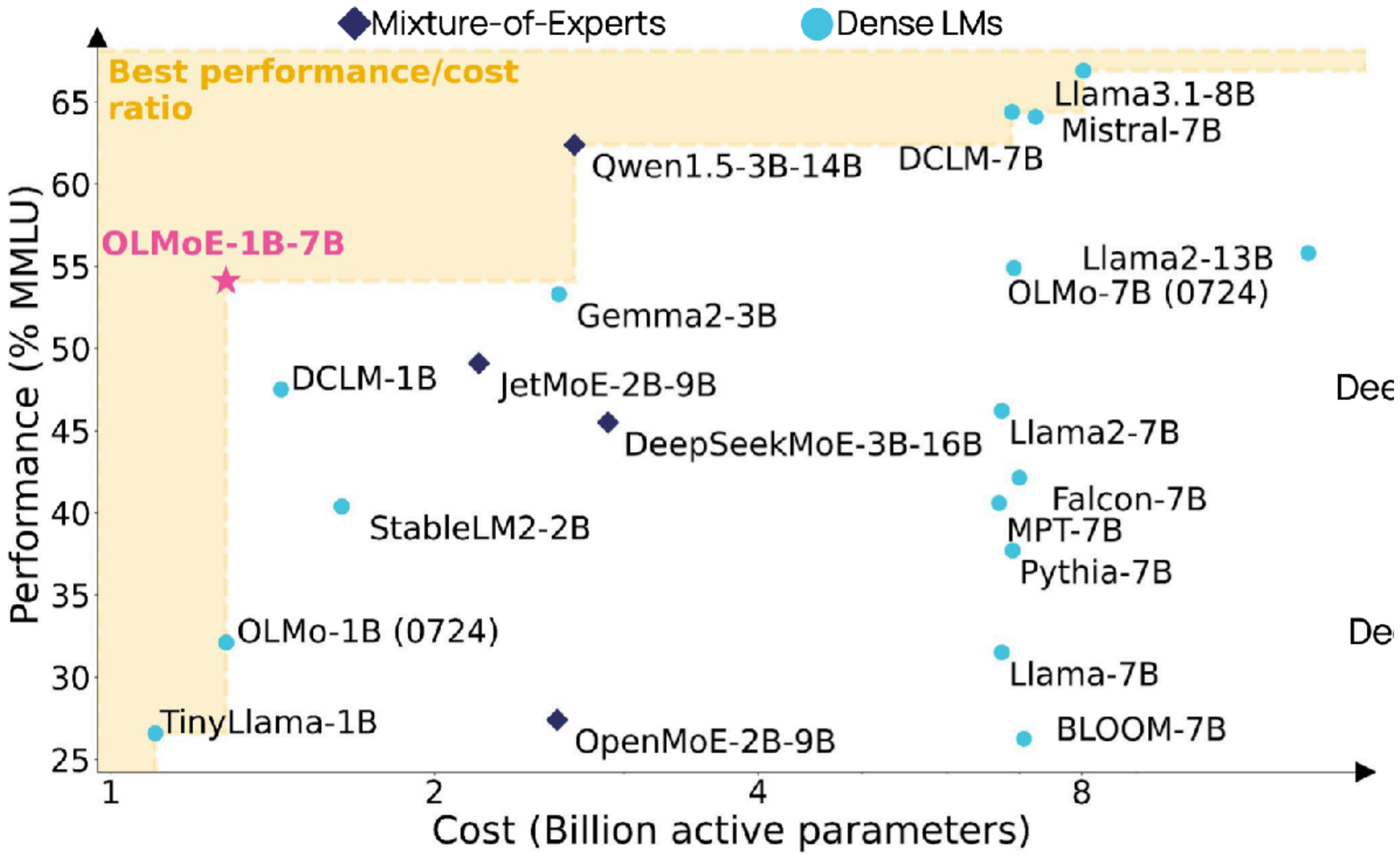
Active Parameters

- We have a model with n_{layers} MoE layers, N experts per layer, k active experts per token, P_E parameters per expert, and P_S shared parameters. Ignoring router parameters:

$$P_{total} = P_S + n_{layers} N P_E \qquad P_{active} = P_S + n_{layers} k P_E$$

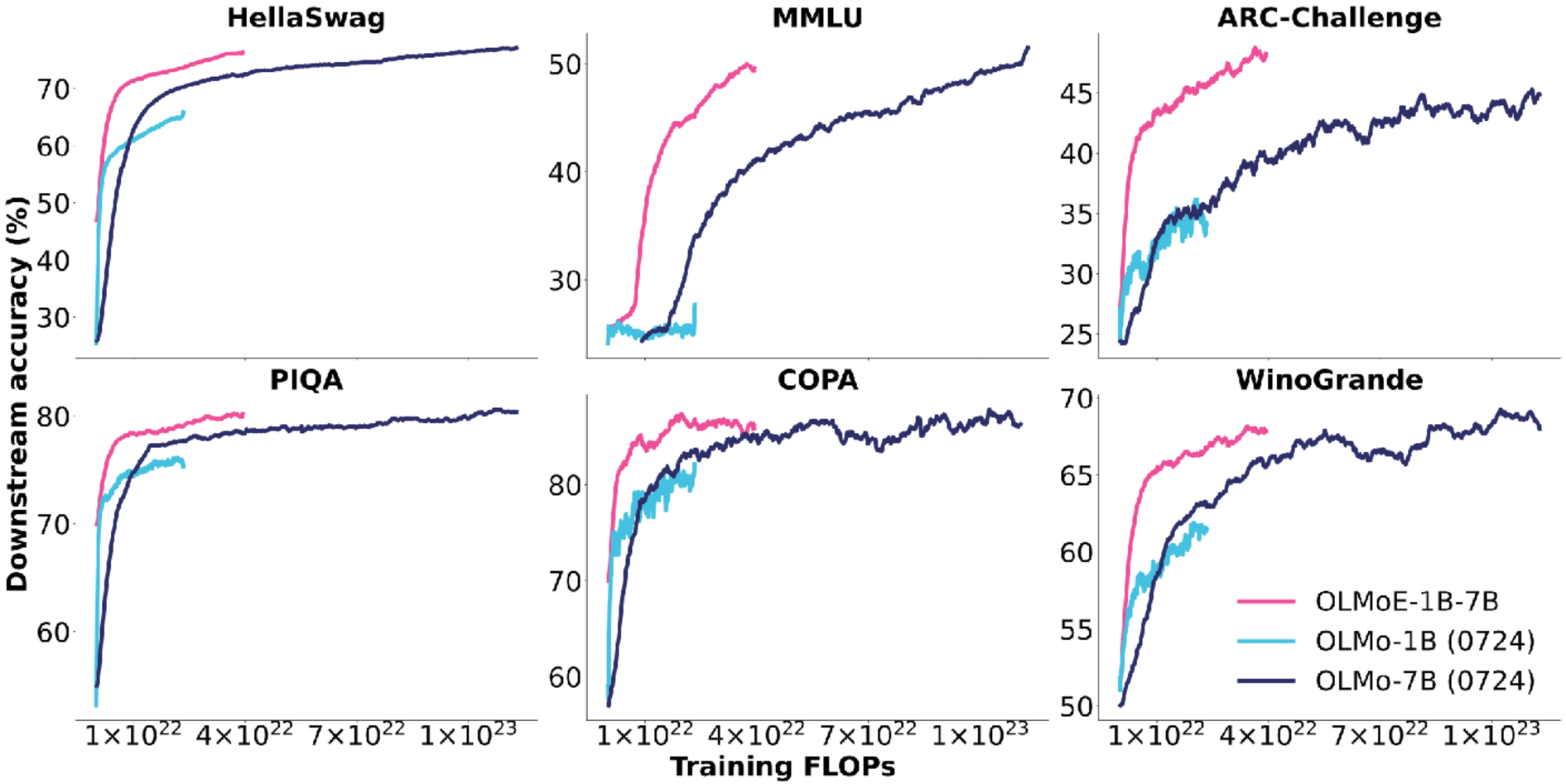
- So if we have $P_S = 16\text{B}$, $P_E = 1\text{B}$, $n_{layers} = 4$, $k = 2$, $N = 8$, we have **48B total parameters** and **24B active parameters**
- What if we increase N from 8 to 16, but keep $k = 2$?
- What if we increase k from 2 to 4, but keep $N = 8$?

Performance vs. Cost



[Muennighoff et al., 2025]

Performance vs. Cost



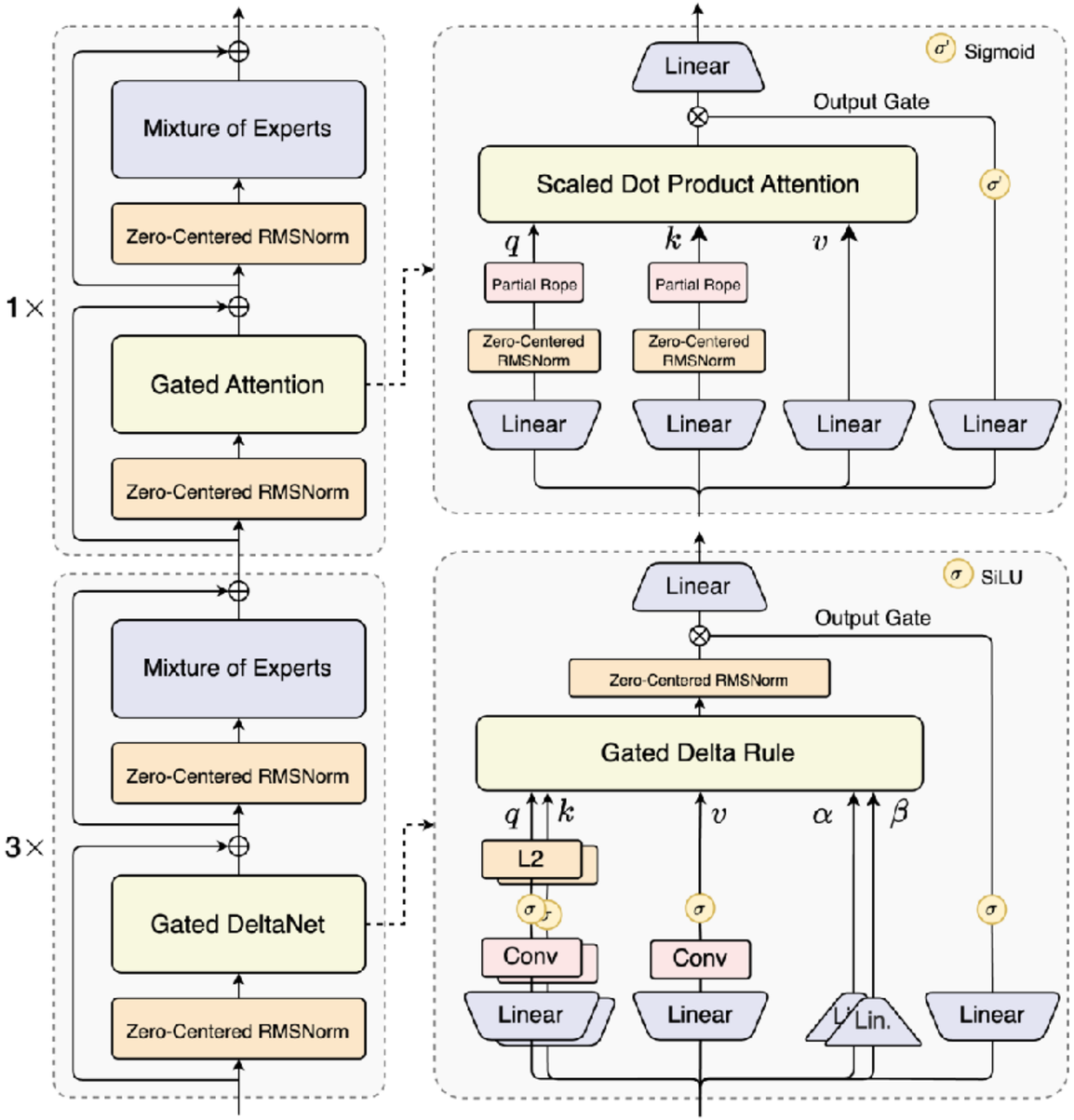
Significant training efficiency benefits!

[Muennighoff et al., 2025]

Qwen3.5

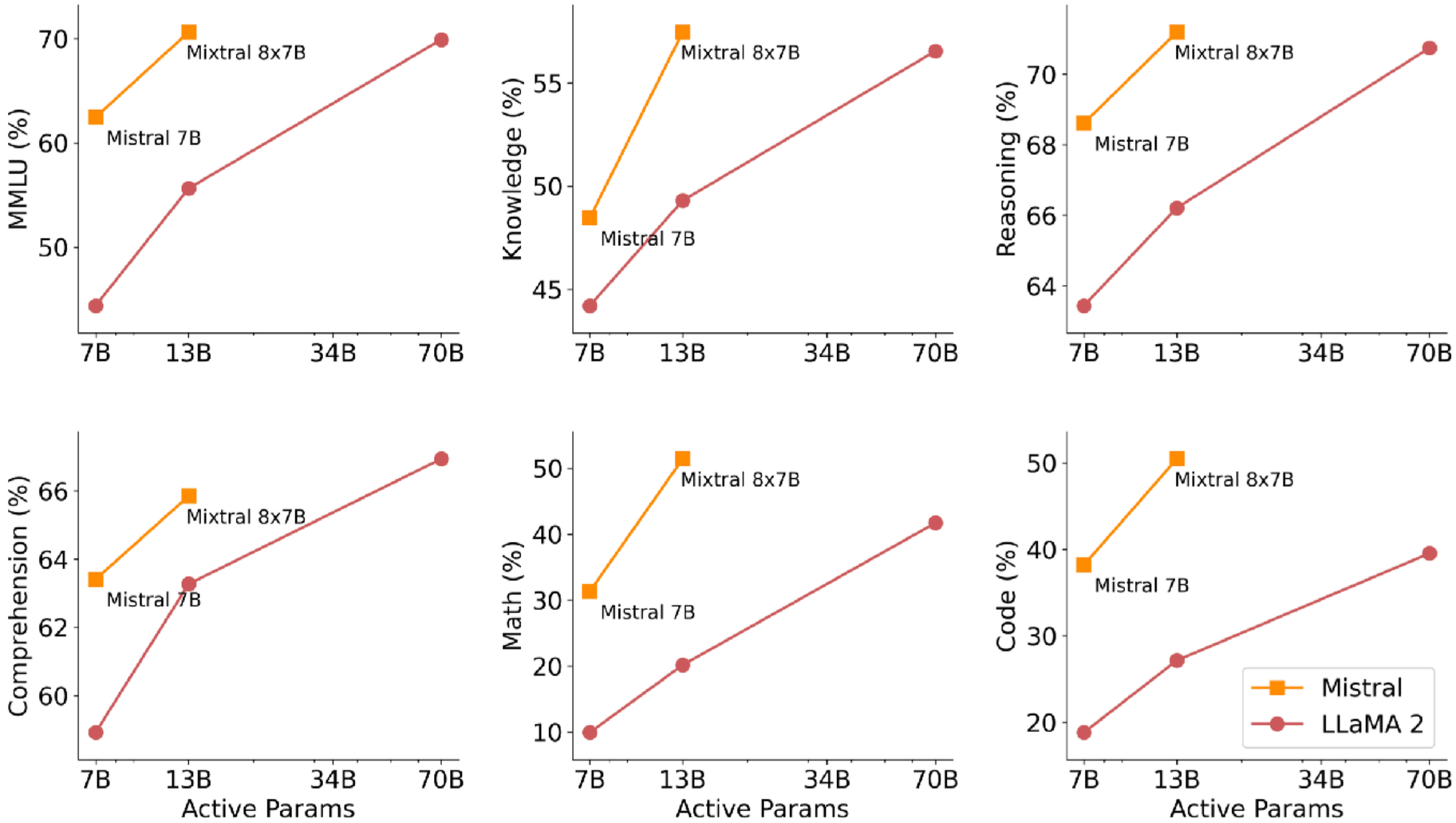
- Puts many efficiency tricks together:
  - Sparsely gated MoE
  - Linear attention layers

Benchmark	Qwen3.5	GPT-5.2	Claude 4.5 Opus	Gemini 3 Pro	K2.5
IFBench	**76.5**	75.4	58.0	70.4	70.2
MultiChallenge	**67.6**	57.9	54.2	64.2	62.7
AIME 2026	91.3	**96.7**	93.3	90.6	93.3
GPQA Diamond	88.4	**92.4**	87.0	91.9	87.6
MMLU-Pro	87.8	87.4	89.5	**89.8**	87.1
LiveCodeBench v6	83.6	87.7	84.8	**90.7**	85.0



Mixtral

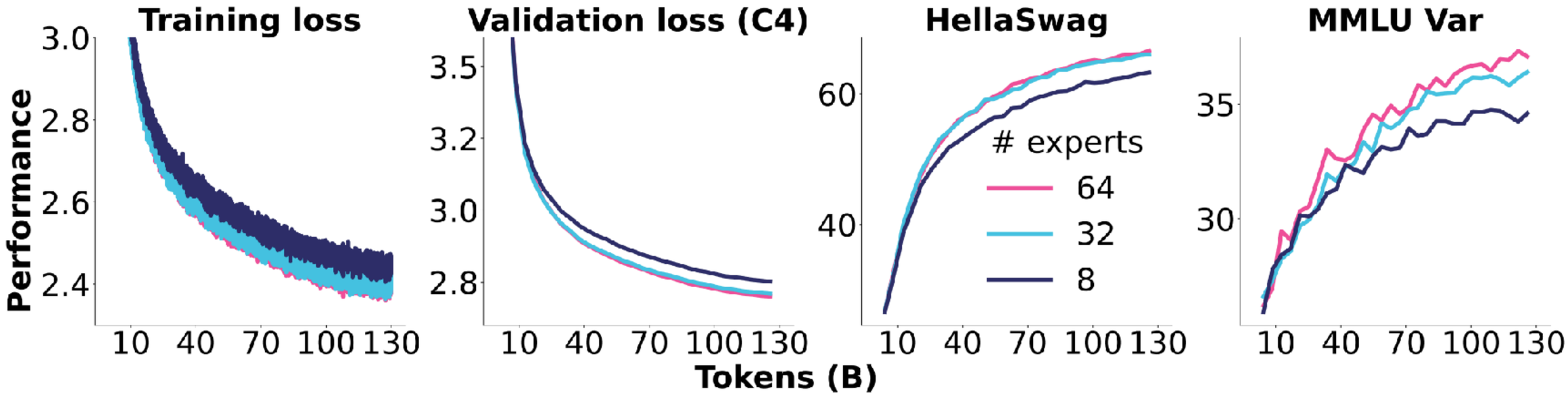
[Jiang et al., 2024]



Performance seems to scale more with total parameters, meaning we can get good performance with fewer active parameters.

How many experts?

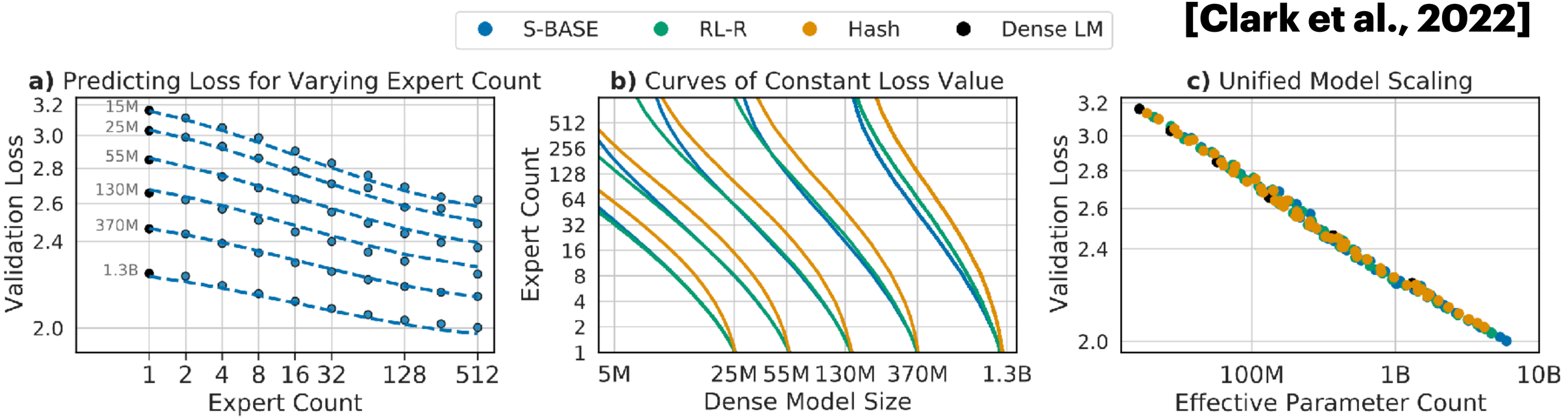
[Muenninghoff et al., 2025]



Recent work suggests that a relatively small number of experts is sufficient to get most performance/efficiency gains

MoE Scaling Laws

[Clark et al., 2022]



- As we increase the number of experts, we need fewer parameters in the dense part of the model
- Ultimately, loss scales predictably with effective parameter count
- (Note: effective parameter count ≠ active parameter count!)

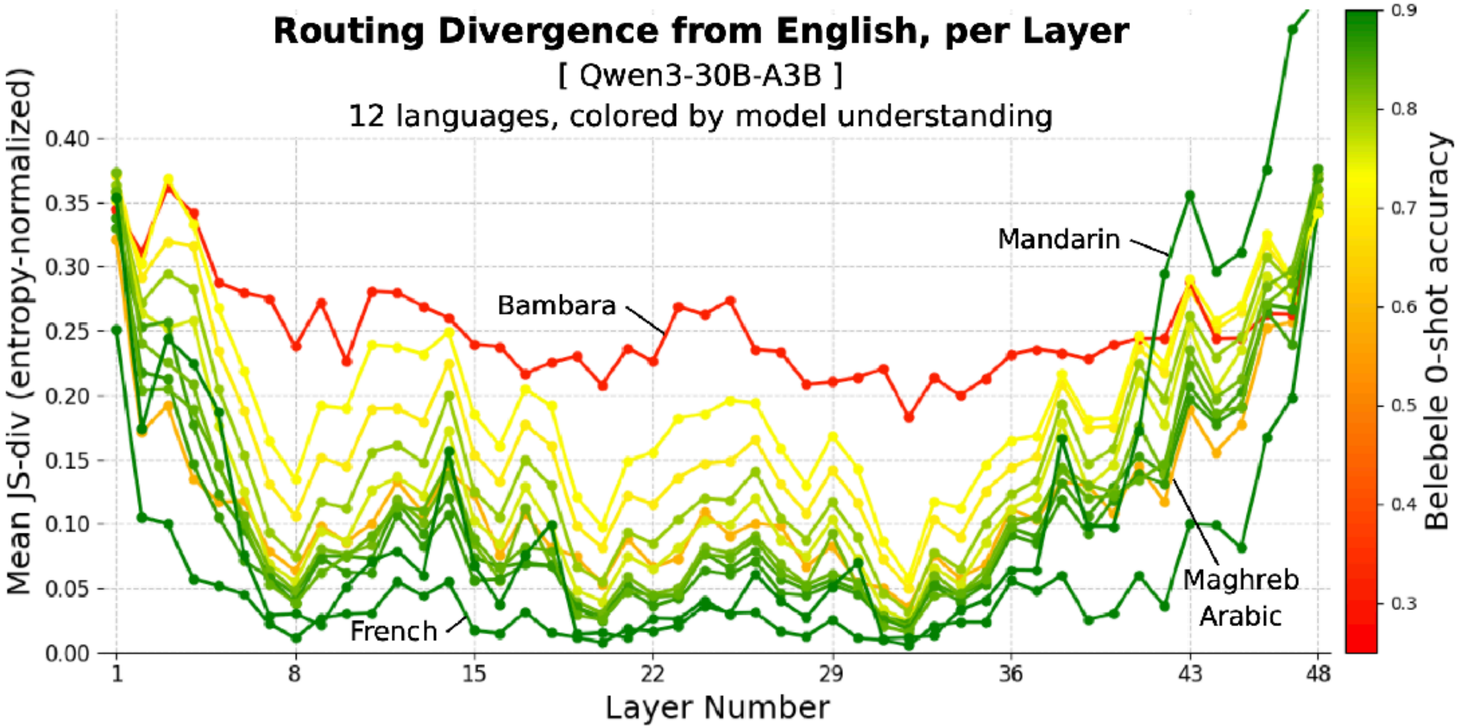
Do experts specialize? [Jiang et al., 2024]

Are LMs grouping certain kinds of computations into certain experts, or are they uniformly distributing their computations across experts?



Do experts specialize?

- Experts are not assigned roles such as “math” or “history”
- Prior work has investigated routing patterns by domain, language (see figure on right), syntax, and layer
- The previous figure from Mixtral observed patterns associated with token identity and syntax, but not as much to do with topic



[Bandarkar et al., 2026]

Expert Pruning

- Large MoE models contain a great deal of expert redundancy and imbalance, and not all experts contribute equally
- **Idea:** identify and remove low-impact experts
- **Lasby et al. [2025]** propose an expert pruning procedure (REAP) based on 2 factors:
    - How often and how strongly does the router choose the expert?
    - When chosen, how much does the expert change the output of its layer?

router value

$$S_j = \frac{1}{|X_j|} \sum_{x \in X_j} g_j(x) \cdot \|f_j(x)\|_2$$

Inputs where expert j is active

output of expert j

Expert Pruning

Lasby et al. [2025]

Model	Compression	Number of Parameters	Method	Coding		
				EvalPlus	LiveCodeBench	Code Avg.
Qwen3-30B-A3B	Baseline	30.5B	-	85.9	30.2	58.1
	25%	23.2B	M-SMoE	82.2	29.3	55.8
			HC-SMoE	80.0	25.8	52.9
			REAP	**84.3**	**30.8**	**57.5**
	50%	16.0B	M-SMoE	62.1	20.5	41.3
			HC-SMoE	57.4	18.5	37.9
			REAP	**82.1**	**29.3**	**55.7**
GLM-4.5-Air	Baseline	106.9B	-	82.0	37.4	59.7
	25%	81.9B	M-SMoE	78.1	33.0	55.5
			HC-SMoE	79.3	36.3	57.8
			REAP	**79.4**	**39.0**	**59.2**
	50%	57.0B	M-SMoE	49.3	9.9	29.6
			HC-SMoE	66.2	22.0	44.1
			REAP	**75.5**	**35.2**	**55.3**

Pros and Cons of MoE

- Higher parameter capacity at similar active expert compute
  - More weights to store
- Experts can be distributed across devices
  - Communication and load balancing challenges
- Different inputs can use different parameters
  - Useful specializations are learned, but not guaranteed