CS639 Deep Learning for NLP

Mixture of Experts in LLMs

Junjie Hu

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Slides adapted from ICML 2024 Tutorial, Tatsu

https://junjiehu.github.io/cs639-spring26/

Goal for Today

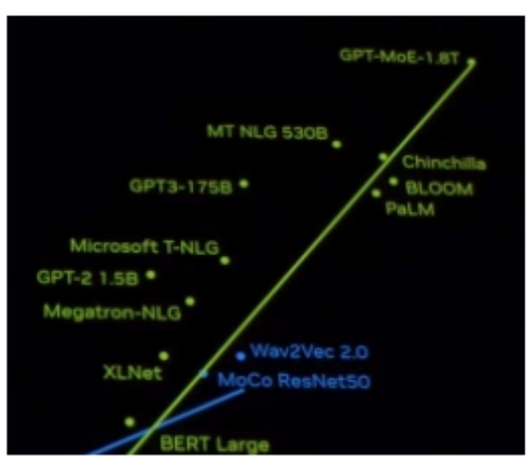
1. Overview
2. MoE Design Varies
3. Summary

Why scale LLMs?

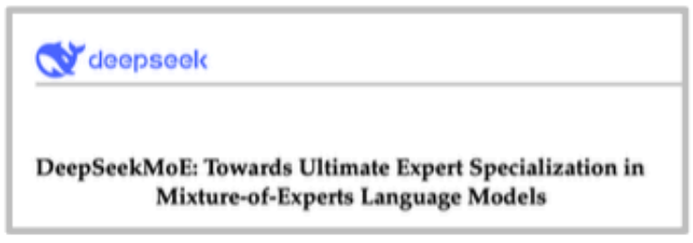
- Compute-performance scaling laws
  - Empirical relationships between the amount of **computation** and the resulting **model performance.**
- Problem: training & inference cost grows with larger models

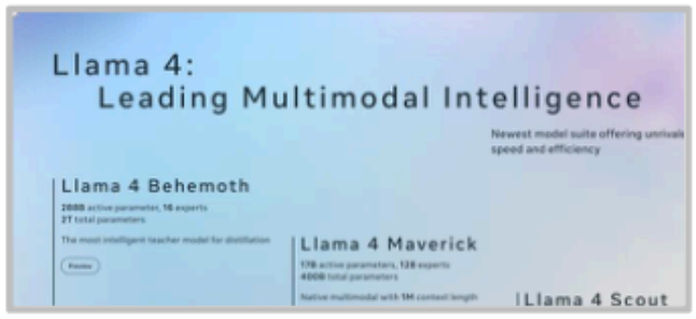
What is Mixture-of-Experts?

- Key Idea
  - Not all parameters need to be active per input
  - Dynamically route tokens to specialized sub-networks (experts)



GPT4 (?)





What is Mixture-of-Experts?

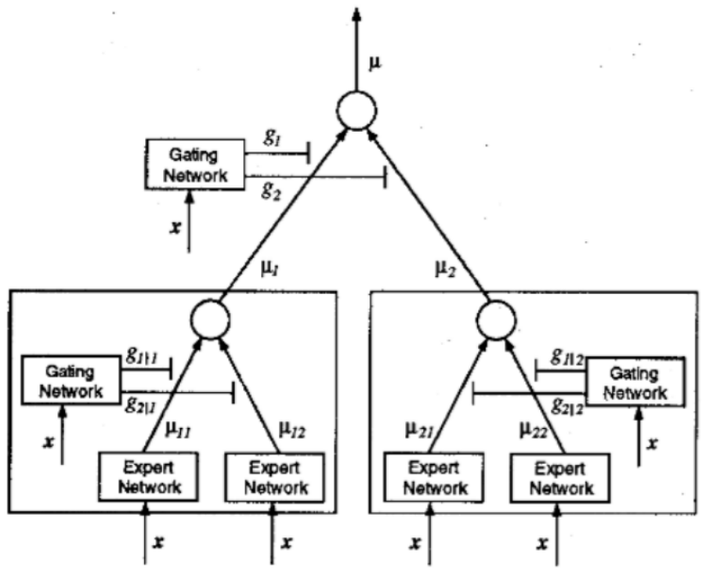


Figure 1: A two-level hierarchical mixture of experts.

Hierarchical Mixtures of Experts for the EM Algorithm, 1993

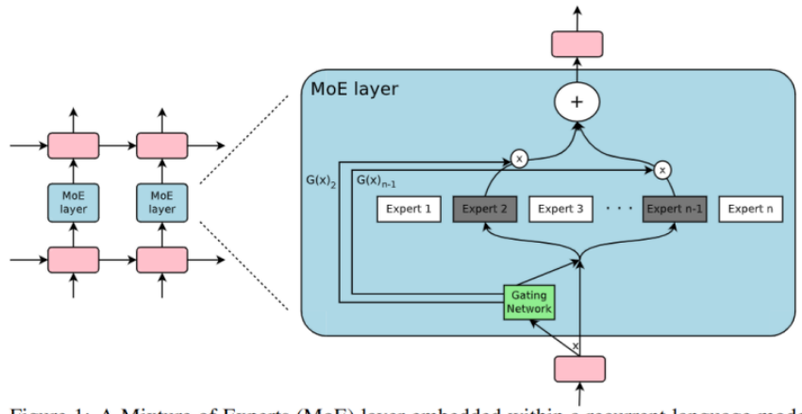
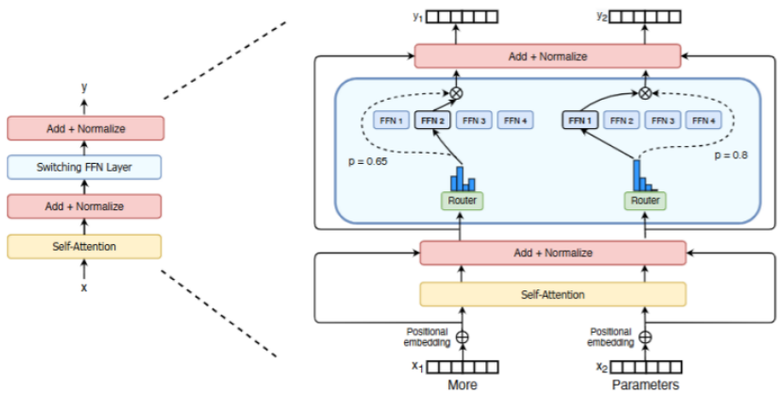


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

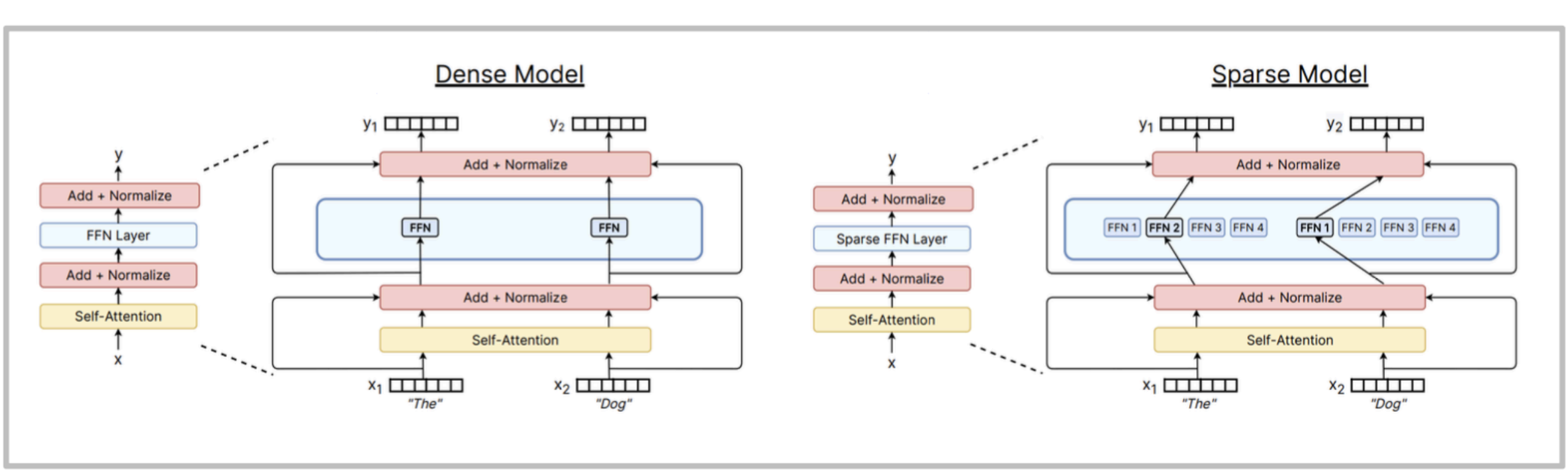
Sparse-Gated Mixture of Experts in LSTM, 2017



Sparse-Gated Mixture of Experts in Transformer, 2021

What's a MoE?

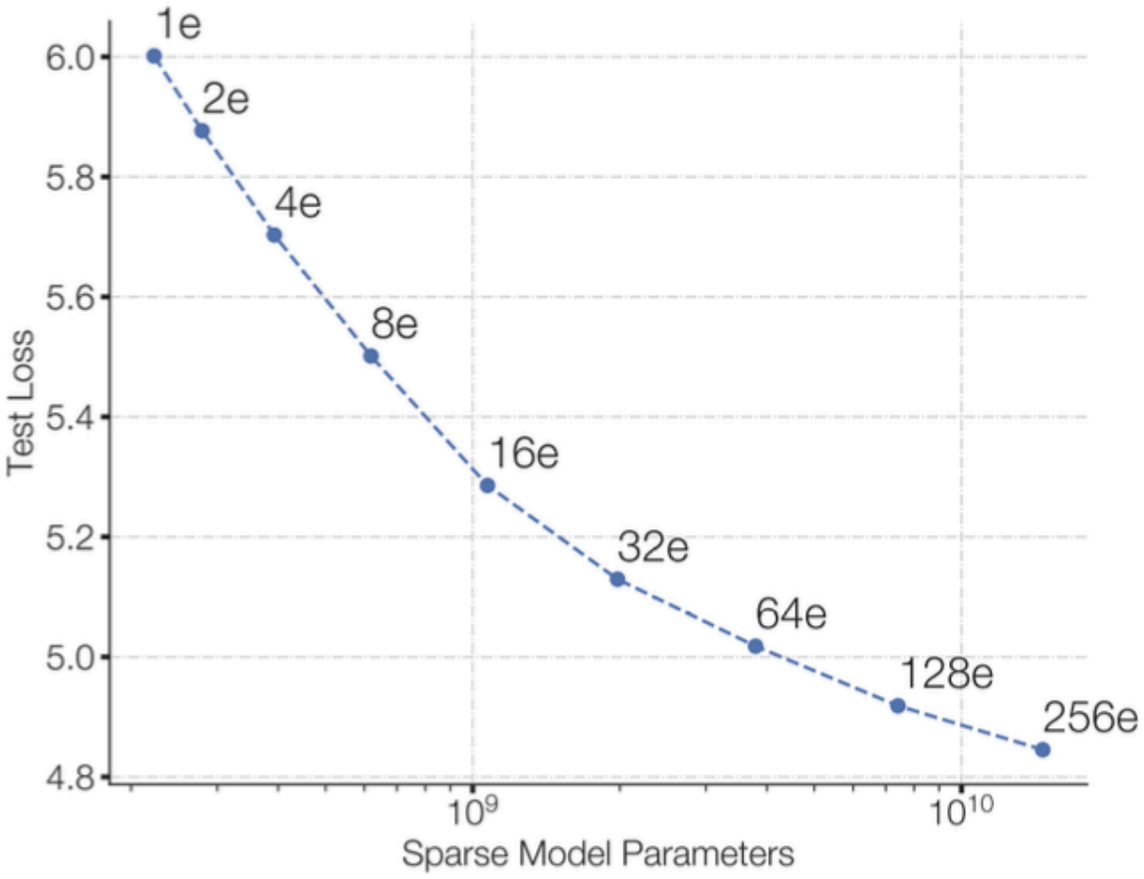
- Replace big feedforward with (many) big feedforward networks and a selector layer
- You can increase the # experts without affecting FLOP

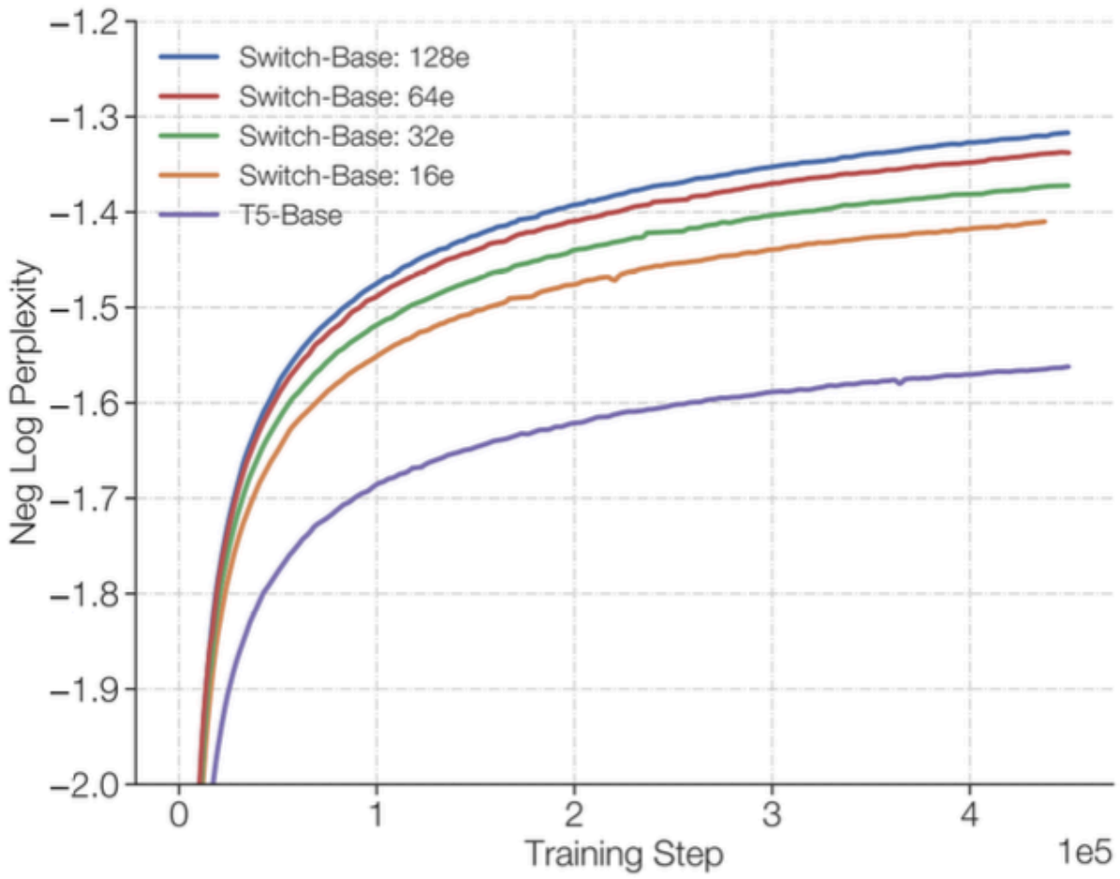


[Fedus et al 2022]

Why are MoEs getting popular?

- Same FLOP, more parameters do better

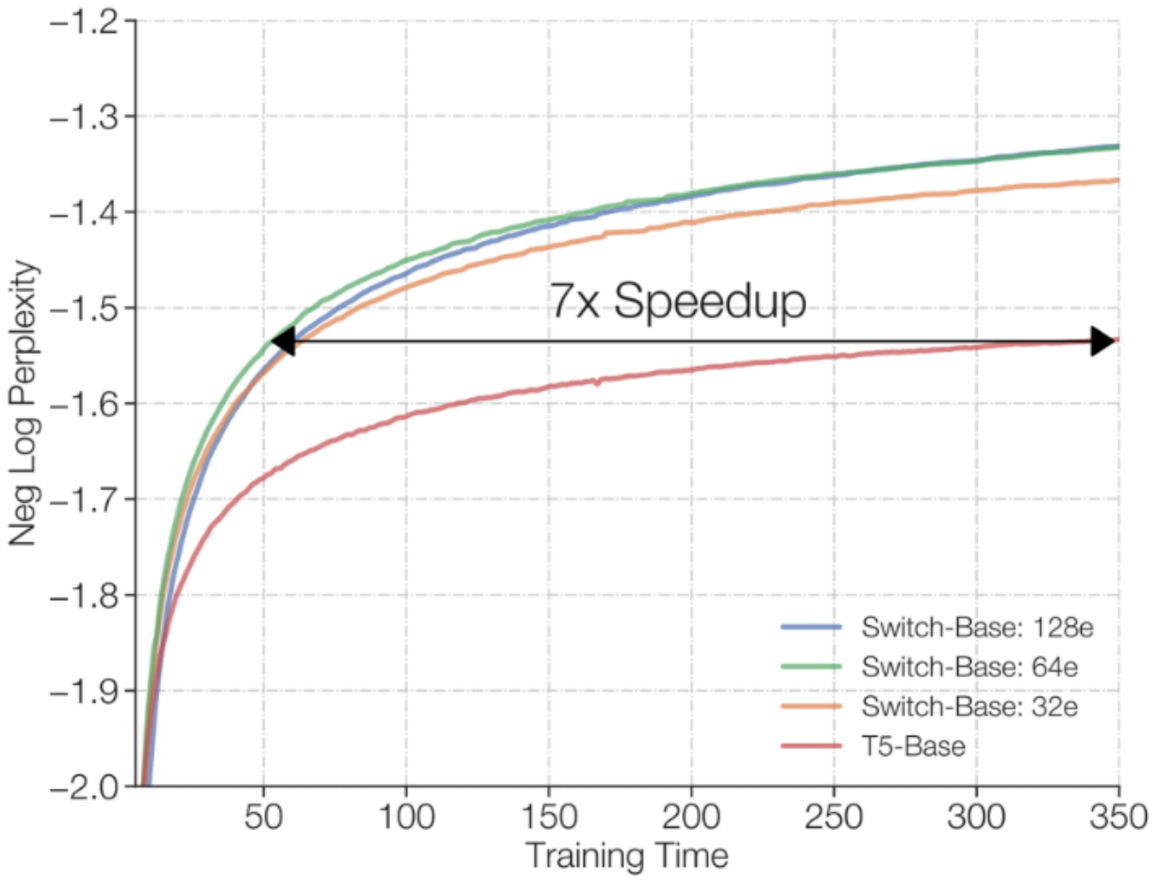




[Fedus et al 2022]

Why are MoEs getting popular?

- Faster to train MoEs, confirmed by AI2's OlMoE



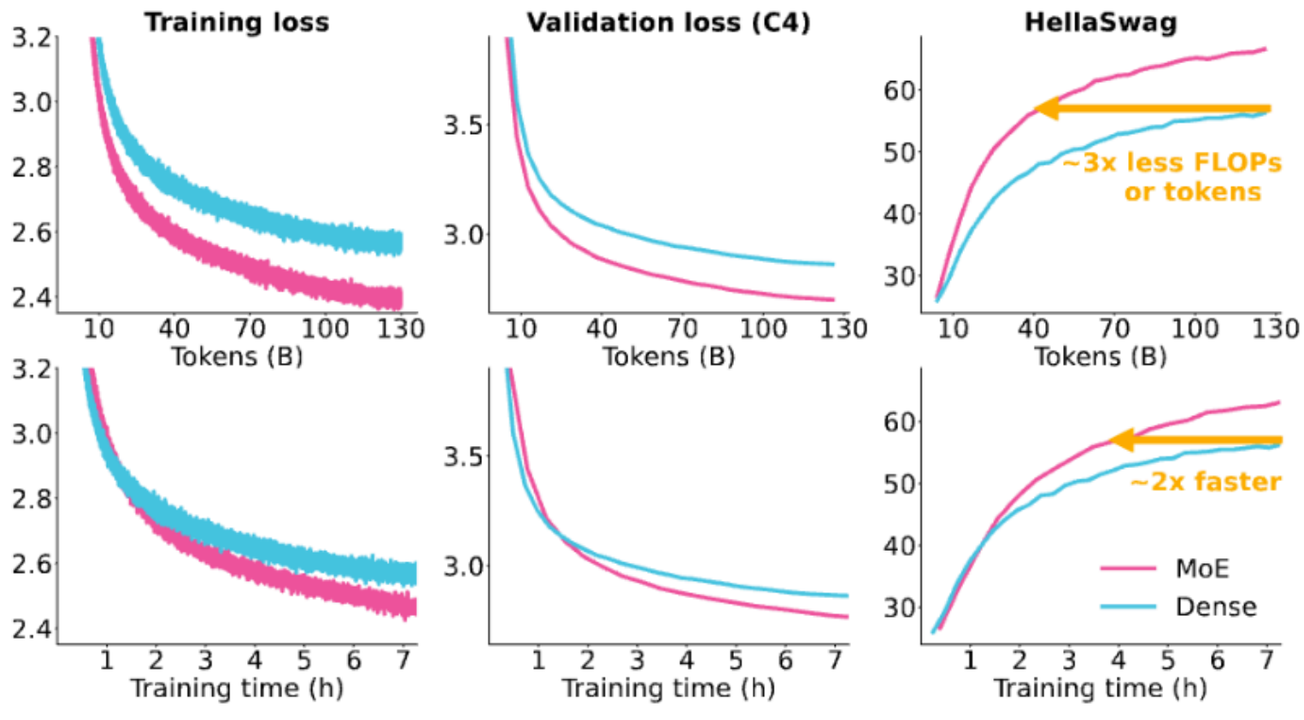
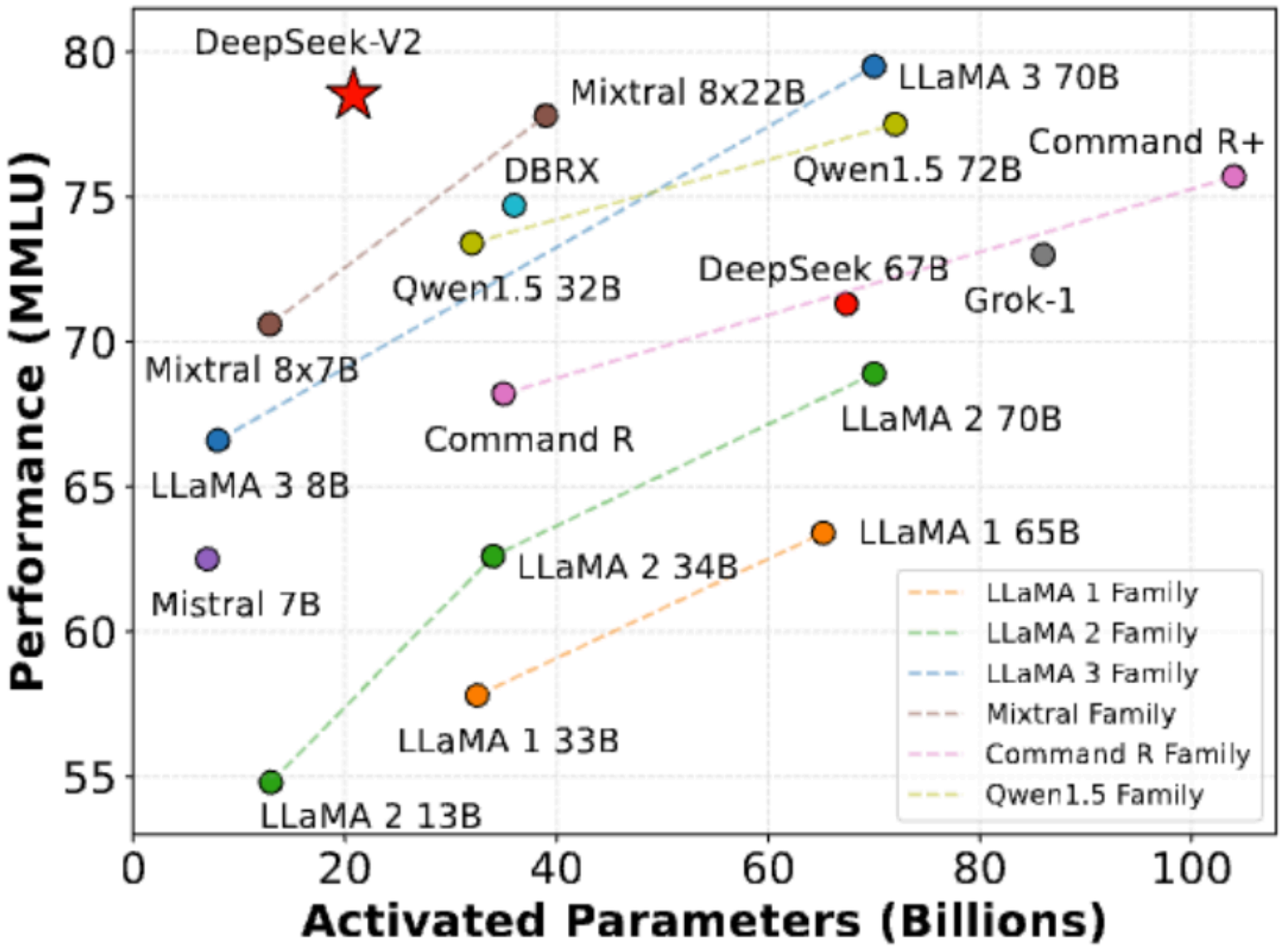


Figure 4: **MoE vs. Dense.** We train a 1.3B parameter dense model and a 1.3B active, 6.9B total parameter MoE model, each on 128 H100 GPUs. Apart from MoE-related changes, we train both

[OlMoE]

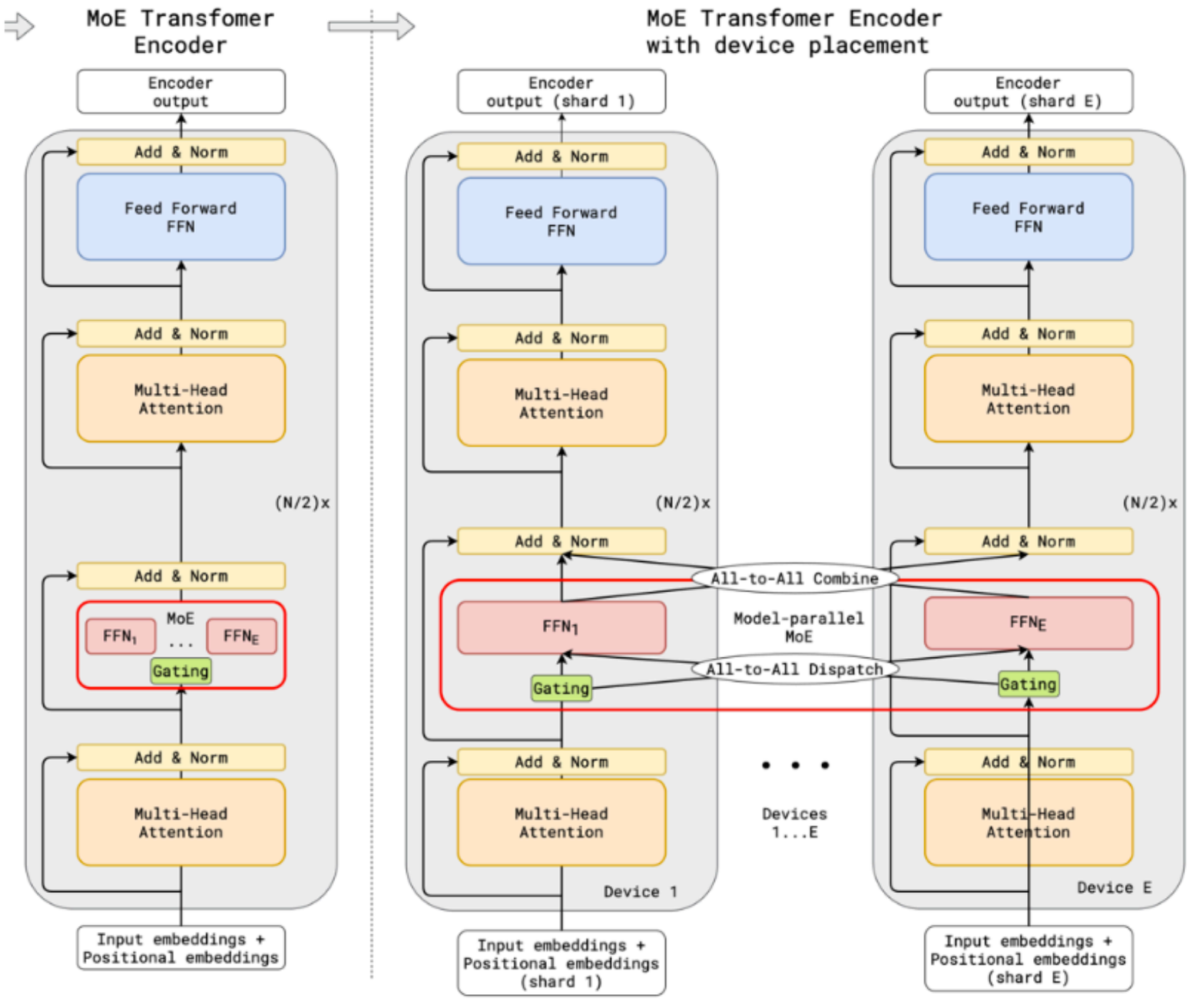
Why are MoEs getting popular?

- **Highly competitive** vs dense equivalents, confirmed by DeepSeek V2, while using **fewer activated parameters**



Why are MoEs getting popular?

- Parallelizable to many devices (GPUs or even machines)



Some MoE Results — from the west

- MoEs are most of the **high-performance** open models, and are quite **fast for model inference**

Category Benchmark	Llama 4 Maverick	Gemini 2.0 Flash	DeepSeek v3.1	GPT-4o
Inference Cost Cost per 1M input & output tokens (3:1 blended)	$0.19-$0.49[5]	**$0.17**	$0.48	$4.38
Image Reasoning MMMU	**73.4**	71.7	No multimodal support	69.1
MathVista	**73.7**	73.1		63.8
Image Understanding ChartQA	**90.0**	88.3		85.7
DocVQA (test)	**94.4**	—		92.8
Coding LiveCodeBench (10/01/2024-02/01/2025)	43.4	34.5	**45.8/49.2**[3]	32.3[3]
Reasoning & Knowledge MMLU Pro	80.5	77.6	**81.2**	—
GPQA Diamond	**69.8**	60.1	68.4	53.6

Earlier MoE results from Chinese groups – Qwen

- Chinese LLM companies are doing quite a bit of MoE work

Model	MMLU	GSM8K	HumanEval	Multilingual	MT-Bench
Mistral-7B	64.1	47.5	27.4	40.0	7.60
Gemma-7B	64.6	50.9	32.3	-	-
Qwen1.5-7B	61.0	62.5	36.0	45.2	7.60
DeepSeekMoE 16B	45.0	18.8	26.8	-	6.93
Qwen1.5-MoE-A2.7B	62.5	61.5	34.2	40.8	7.17

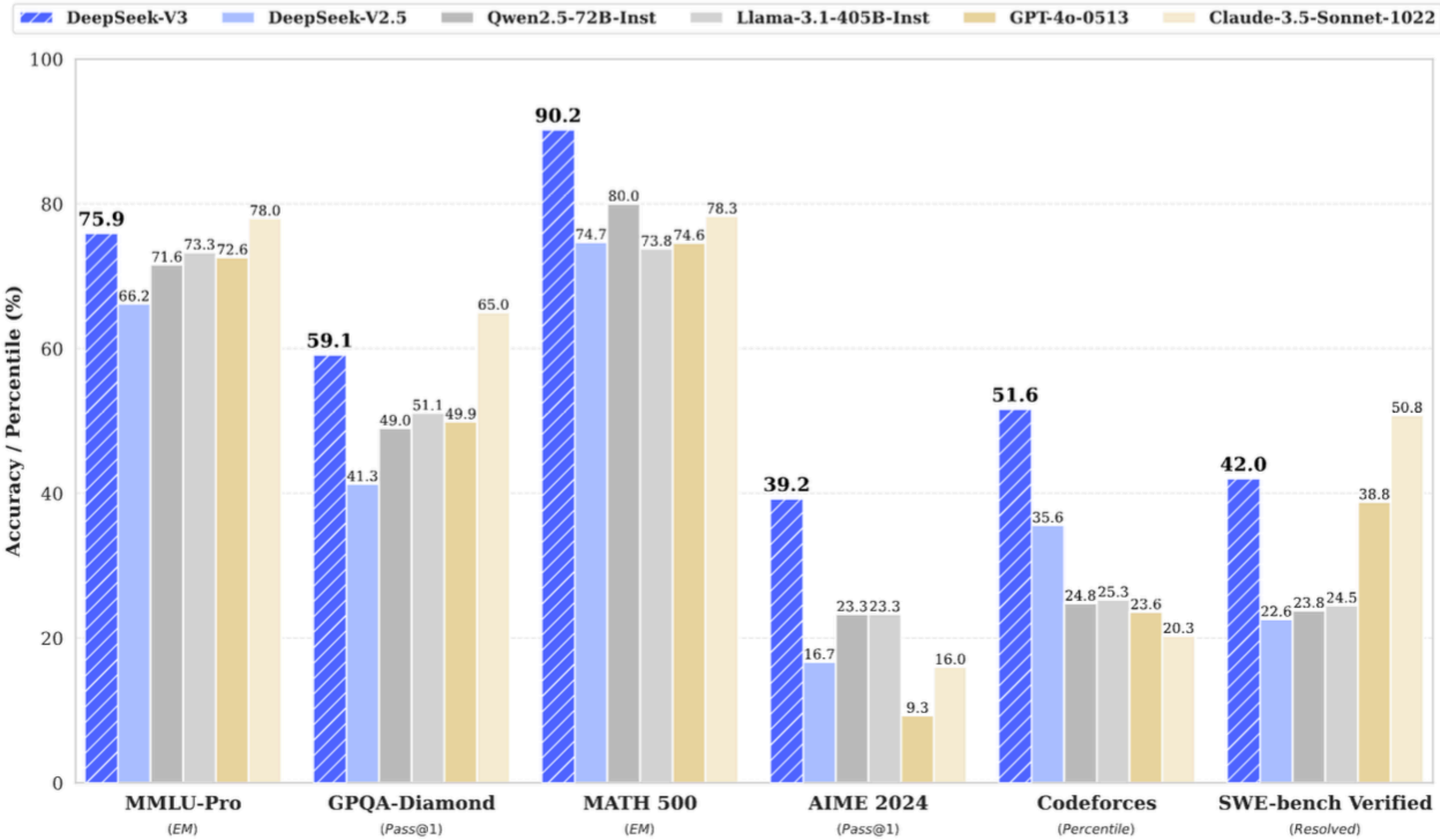
Model	#Parameters	#(Activated) Parameters
Mistral-7B	7.2	7.2
Qwen1.5-7B	7.7	7.7
Gemma-7B	8.5	7.8
DeepSeekMoE 16B	16.4	2.8
Qwen1.5-MoE-A2.7B	14.3	2.7

Earlier MoE results from Chinese groups – DeepSeek

- There's also some good recent ablation work on MoEs showing they're generally good

Metric	# Shot	Dense	Hash Layer	Switch
# Total Params	N/A	0.2B	2.0B	2.0B
# Activated Params	N/A	0.2B	0.2B	0.2B
FLOPs per 2K Tokens	N/A	2.9T	2.9T	2.9T
# Training Tokens	N/A	100B	100B	100B
Pile (Loss)	N/A	2.060	1.932	1.881
HellaSwag (Acc.)	0-shot	38.8	46.2	49.1
PIQA (Acc.)	0-shot	66.8	68.4	70.5
ARC-easy (Acc.)	0-shot	41.0	45.3	45.9
ARC-challenge (Acc.)	0-shot	26.0	28.2	30.2
RACE-middle (Acc.)	5-shot	38.8	38.8	43.6
RACE-high (Acc.)	5-shot	29.0	30.0	30.9
HumanEval (Pass@1)	0-shot	0.0	1.2	2.4
MBPP (Pass@1)	3-shot	0.2	0.6	0.4
TriviaQA (EM)	5-shot	4.9	6.5	8.9
NaturalQuestions (EM)	5-shot	1.4	1.4	2.5

Earlier MoE results from Chinese groups – DeepSeek



Why haven't MoEs been more popular?

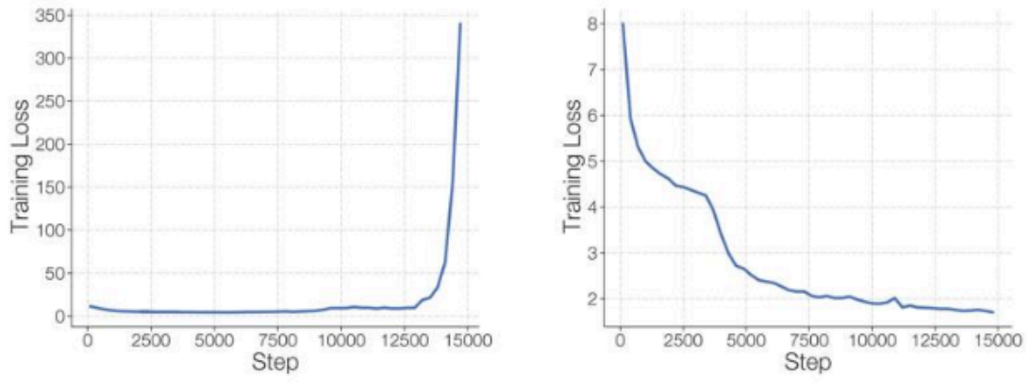
- Infrastructure is complex / advantages on multiple node

At a high level, sparsity is good when you have many accelerators (e.g. GPU/TPU) to host all the additional parameters that comes when using sparsity. Typically models are trained using data-parallelism where different machines will get different slices of the training/inference data. The machines used for operating on the different slices of data can now be used to host many more model parameters. Therefore, sparse models are good when training with data parallelism and/or have high throughput while serving: training/serving on many machines which can host all of the parameters.

[Fedus et al 2022]

- Training objectives are somewhat heuristics (and sometime unstable)

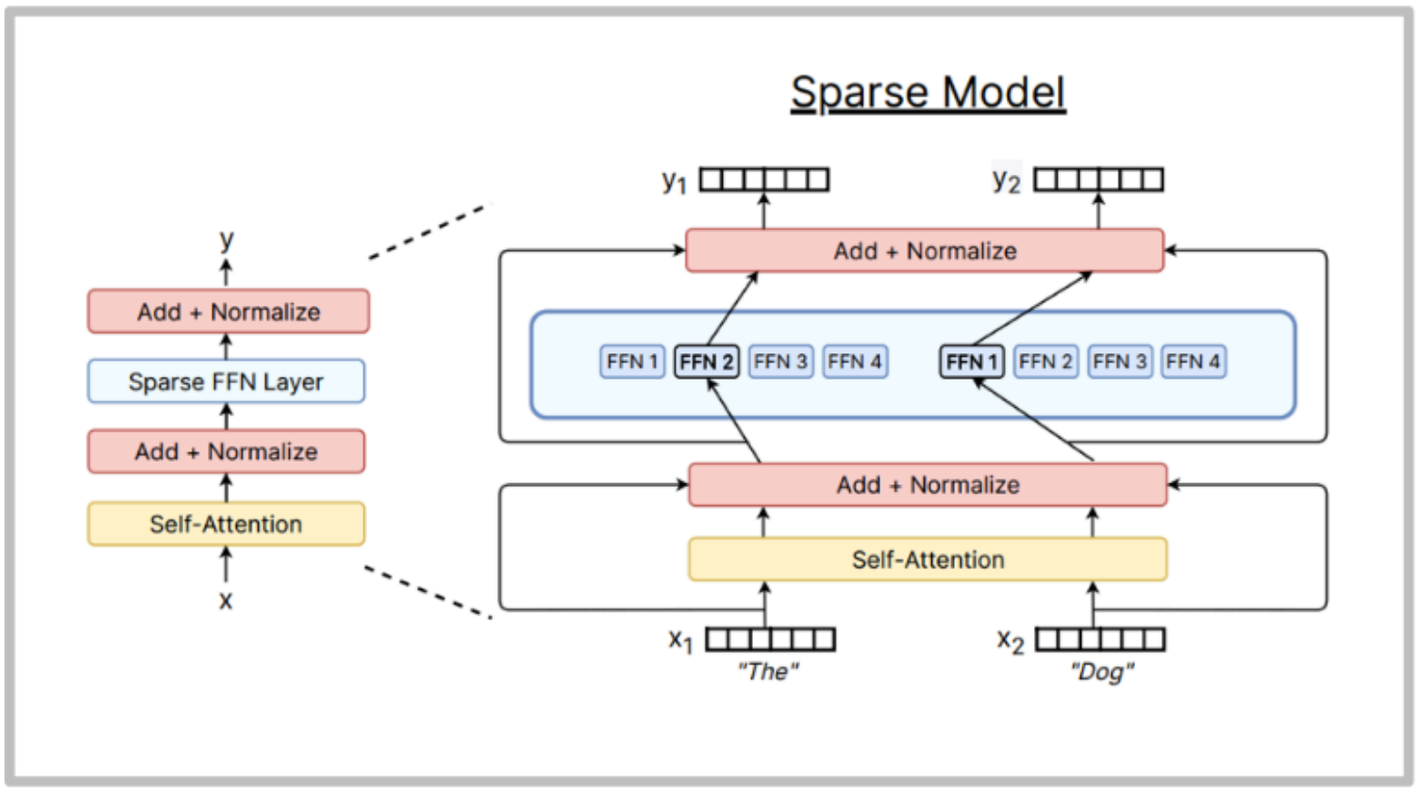
Sparse models often suffer from training instabilities (Figure 1) worse than those observed in standard densely-activated Transformers.



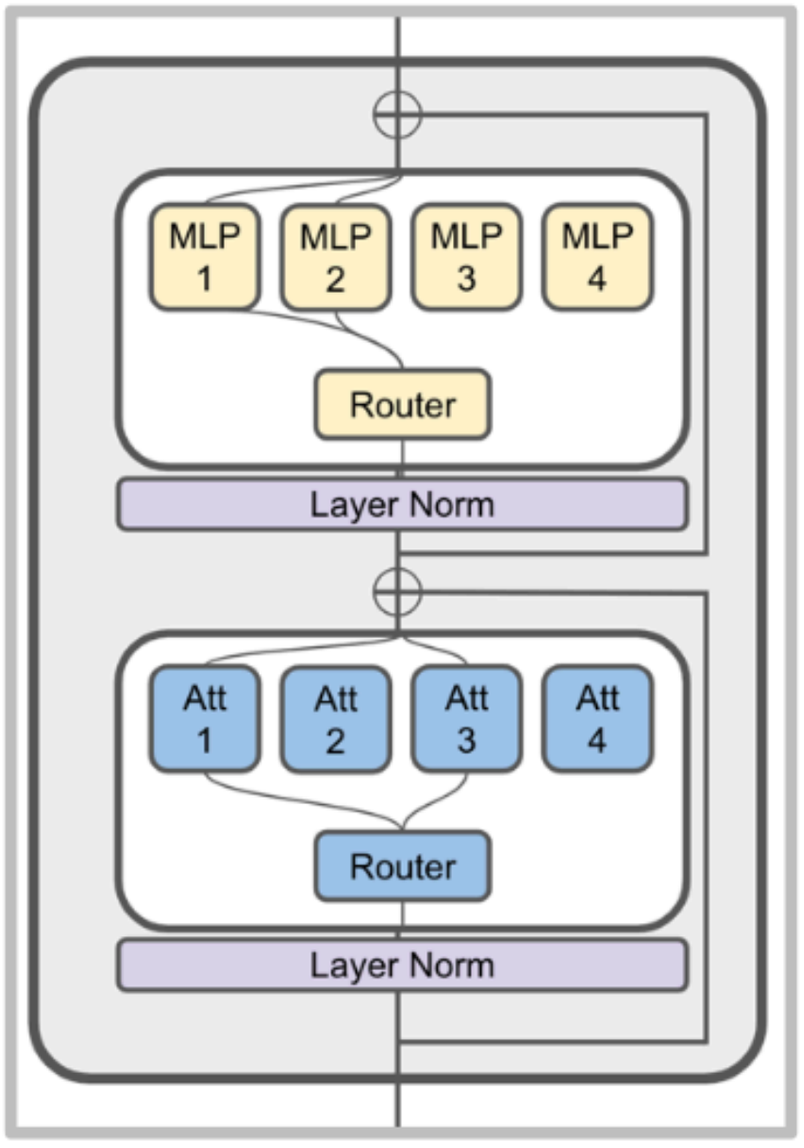
[Zoph et al 2022]

What MoEs generally look like?

Typical: replace MLP with MoE layer



Less common: MoE for attention heads

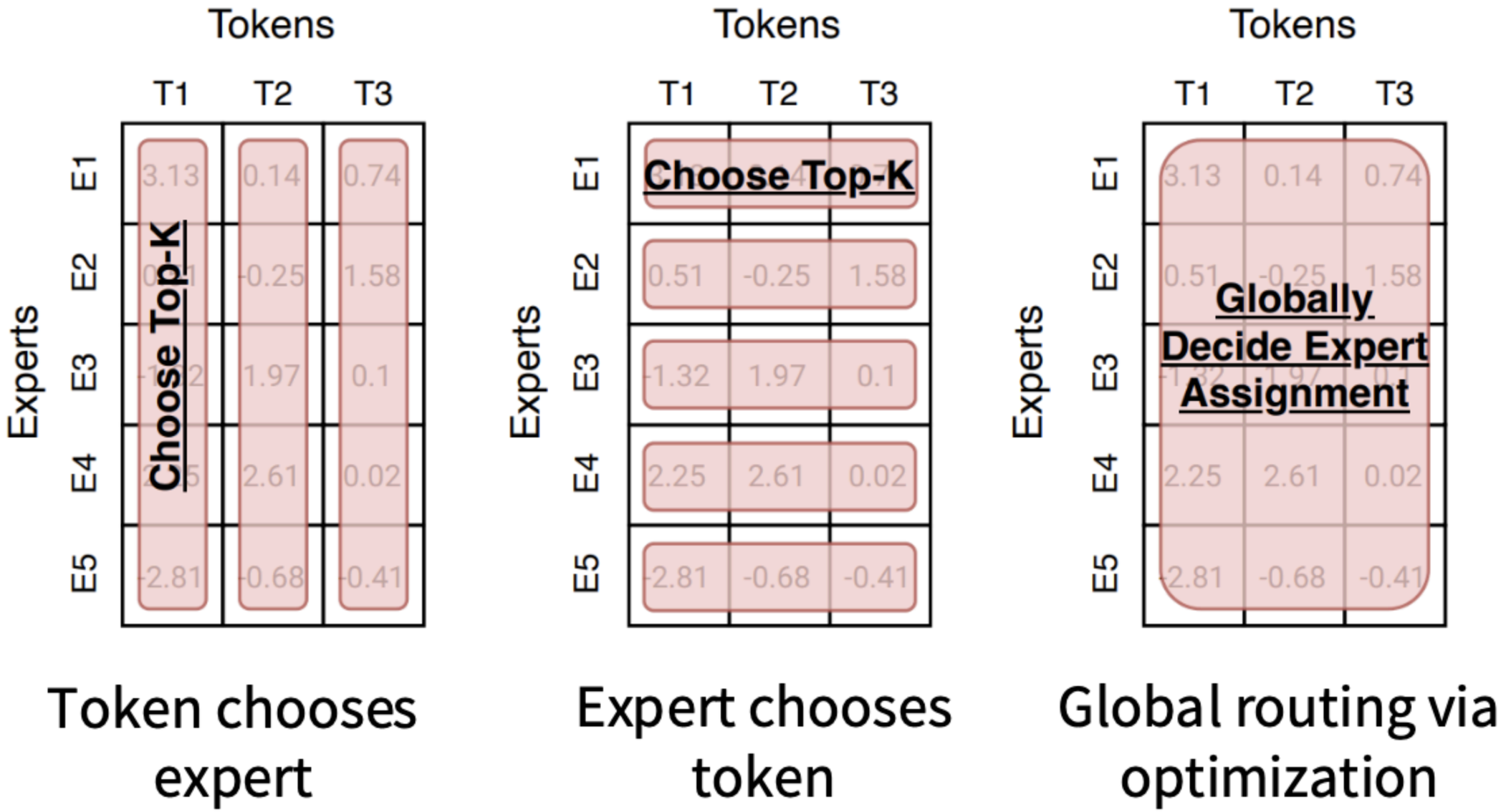


MoE - Design Varies

- Routing function
- Expert sizes
- Training objectives

Routing function - overview

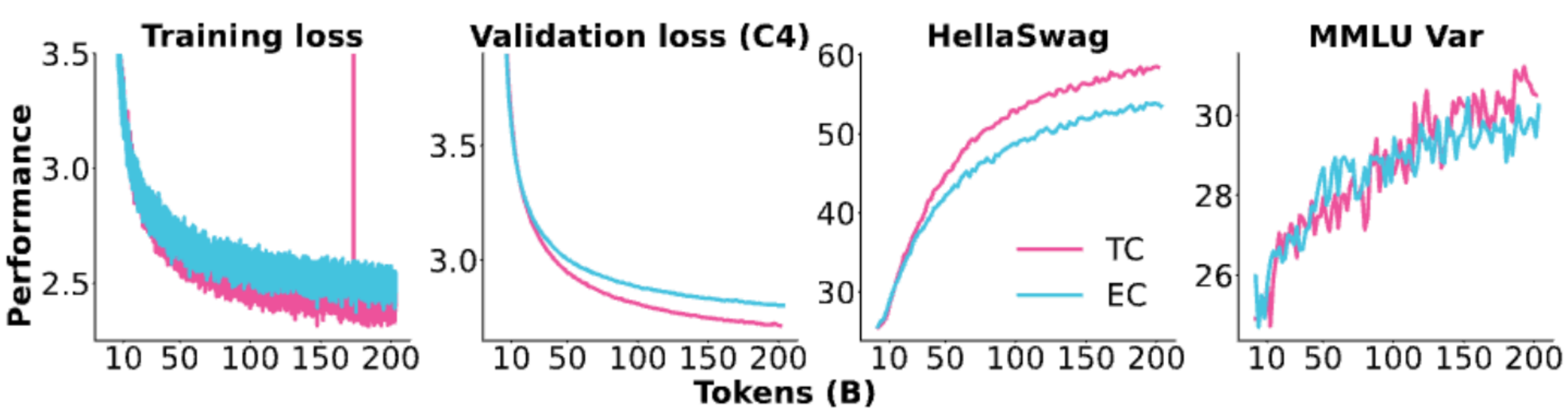
- Many of the routing algorithms boil down to ‘choose top k’ (k usually is small, e.g., 2)
  - Token chooses: each token choose the top-K experts
  - Expert chooses: each expert choose the top-K tokens



[Fedus et al 2022]

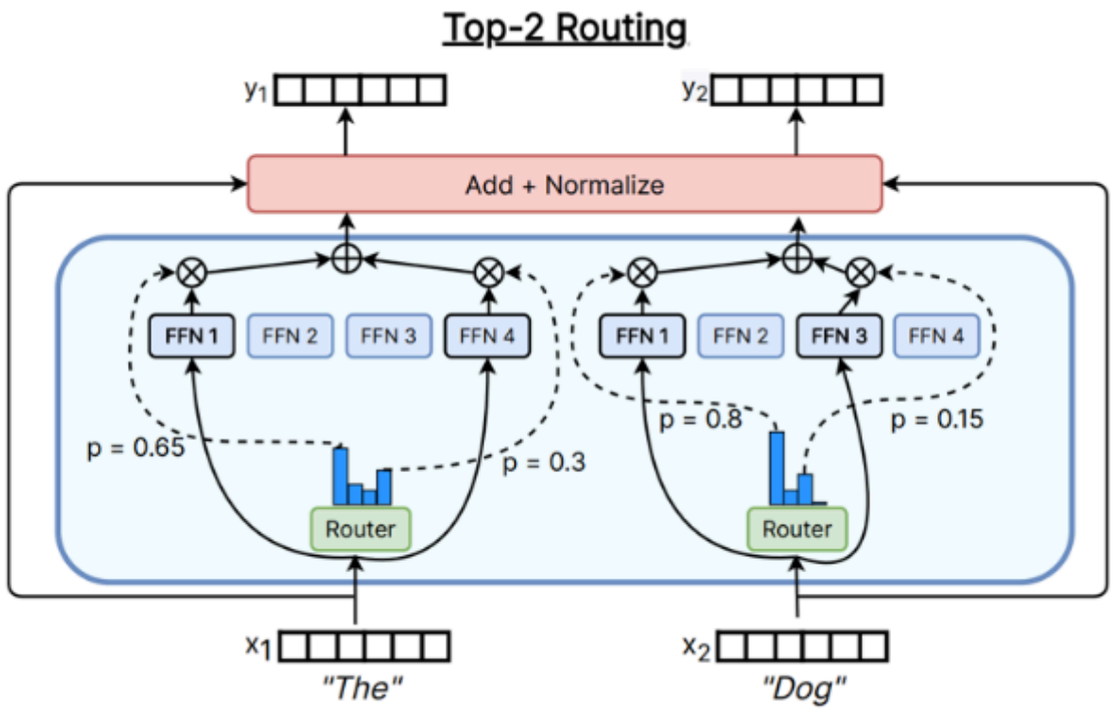
Routing Type

- Almost all the MoEs do a standard ‘token choice topk’ routing. Some recent ablations



Common routing variants in detail

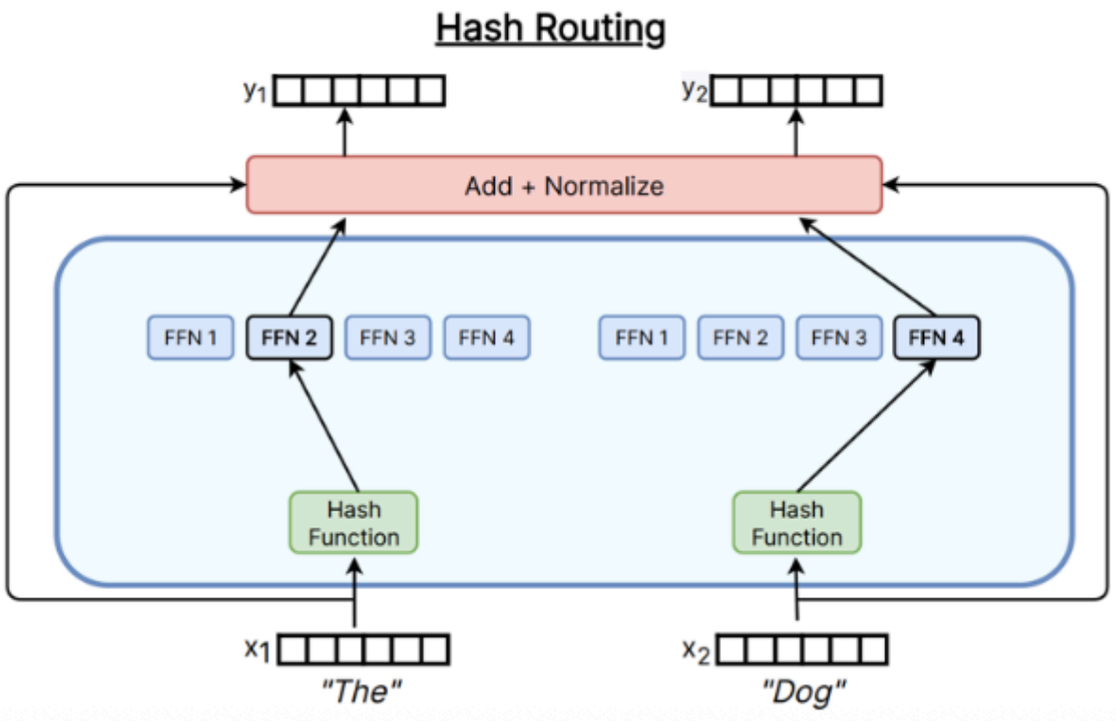
Top-k



Top-K: Used in most MoEs

Switch Transformer (k=1), Gshard (k=2), Grok (2), Mixtral (2), Qwen (4), DBRX (4), DeepSeek (7)

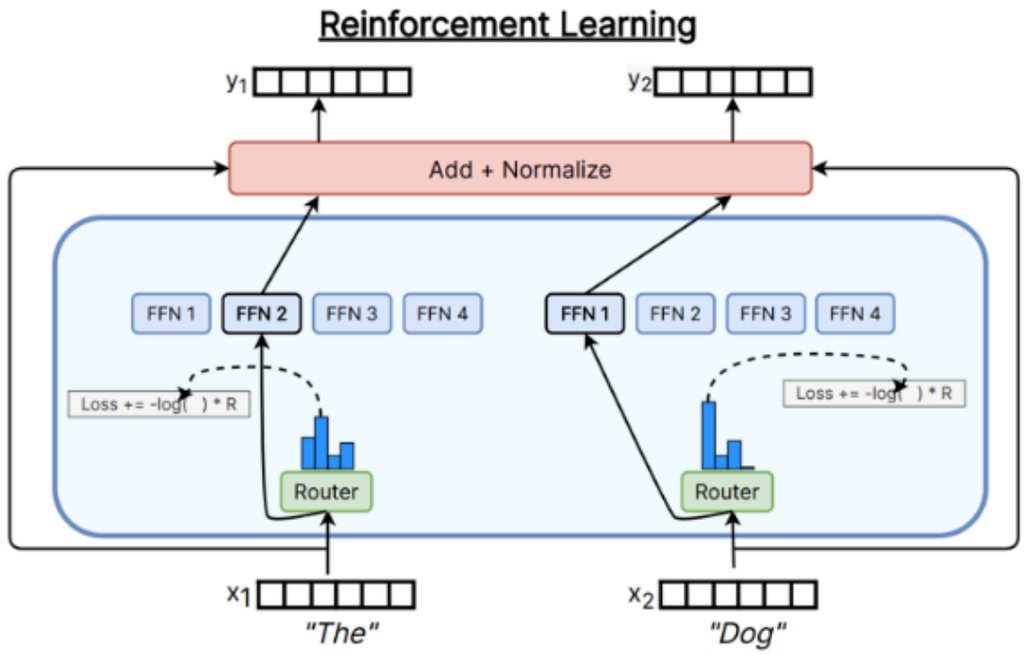
Hashing



Common baseline

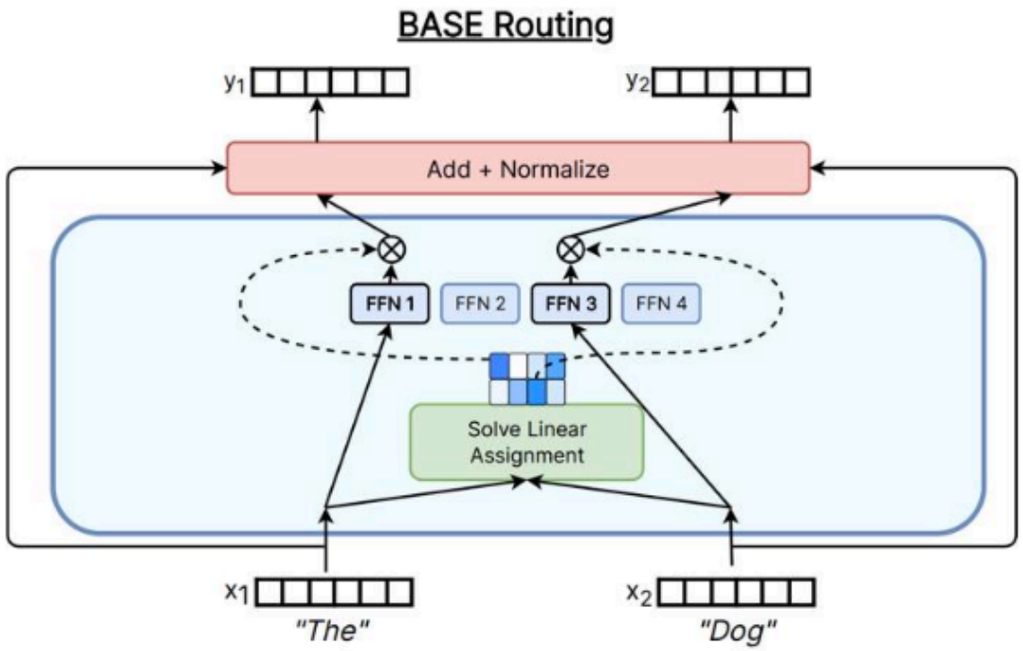
Other routing methods

RL to learn routes



Used in some of the earliest work (Bengio 2013), not common now

Solve a matching problem



Linear assignment for routing
Used in various papers like Clark ‘22

Top-K routing in detail.

- Most papers do the old and classic top-k routing. How does this work?

Gating

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

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

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

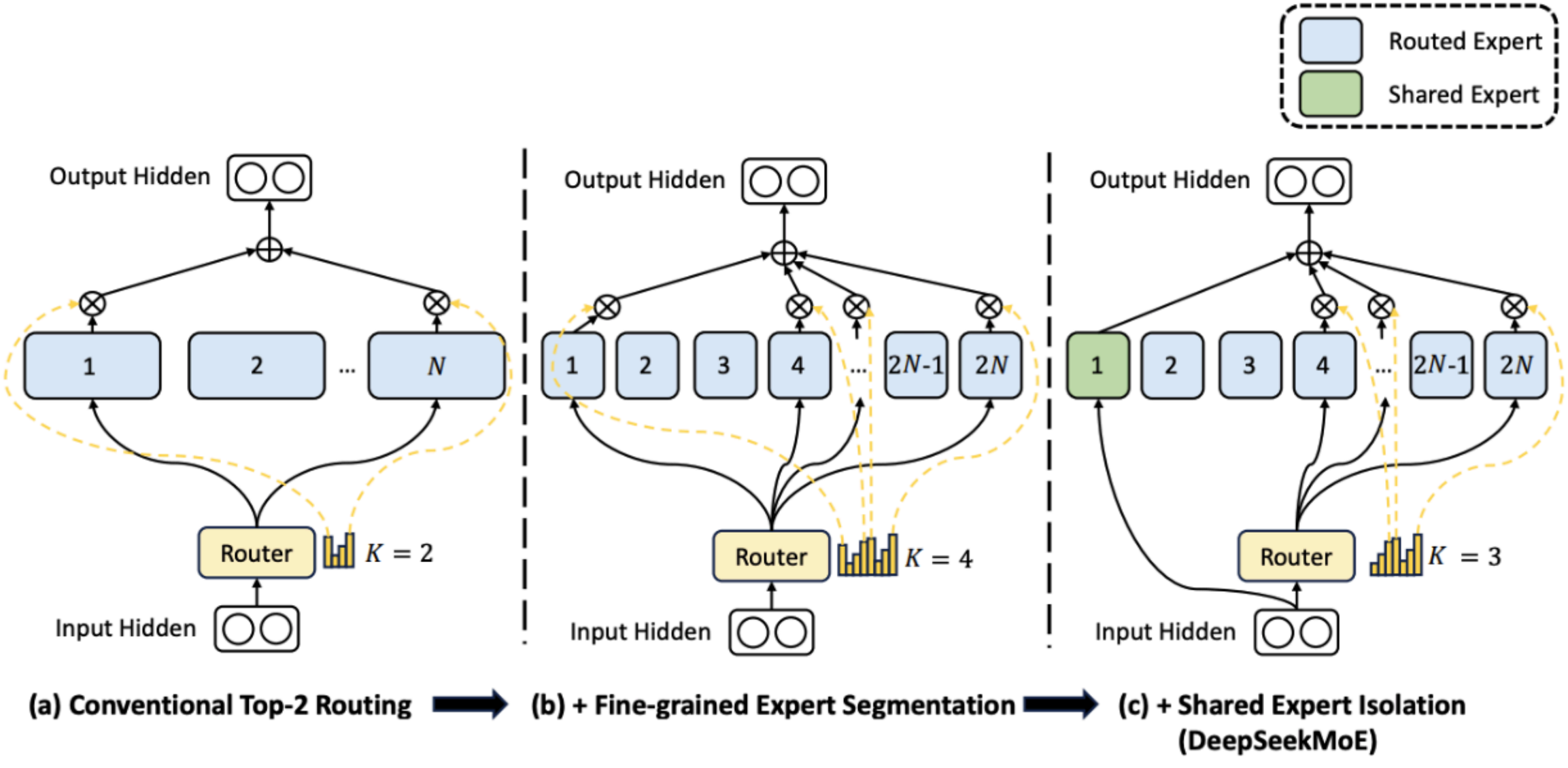
Gates selected by a logistic regressor

This is the DeepSeek (V1-2) router (Grok, Qwen do this too)

Mixtral, DBRX, DeepSeek v3 softmaxes after the TopK

Recent variations from DeepSeek and other Chinese LMs

- Smaller, larger number of experts + a few shared experts that are always on.
- Used in DeepSeek / Qwen, originally from DeepSpeed MoE



(a) Conventional Top-2 Routing → **(b) + Fine-grained Expert Segmentation** → **(c) + Shared Expert Isolation (DeepSeekMoE)**

Various ablations from the DeepSeek paper

- More experts, shared experts all seem to generally help

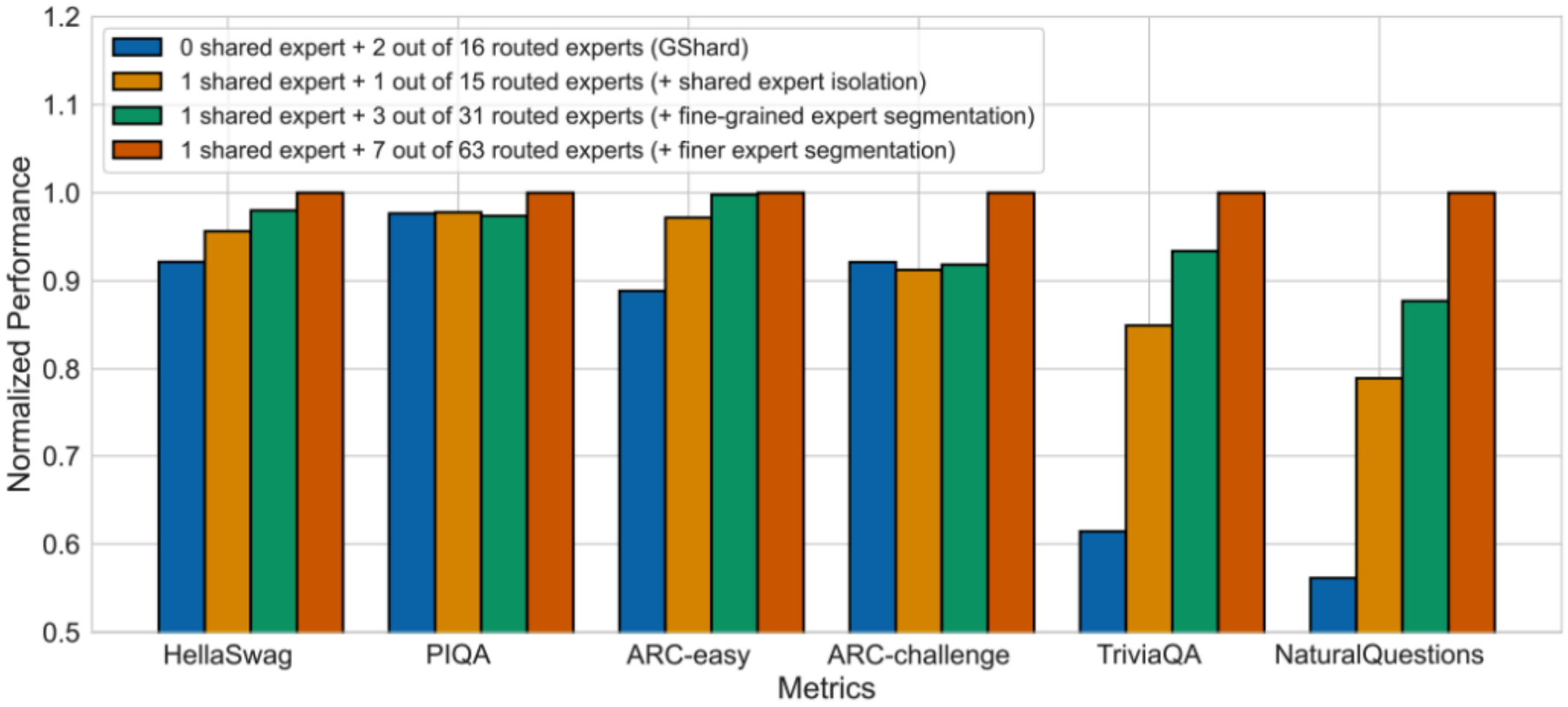
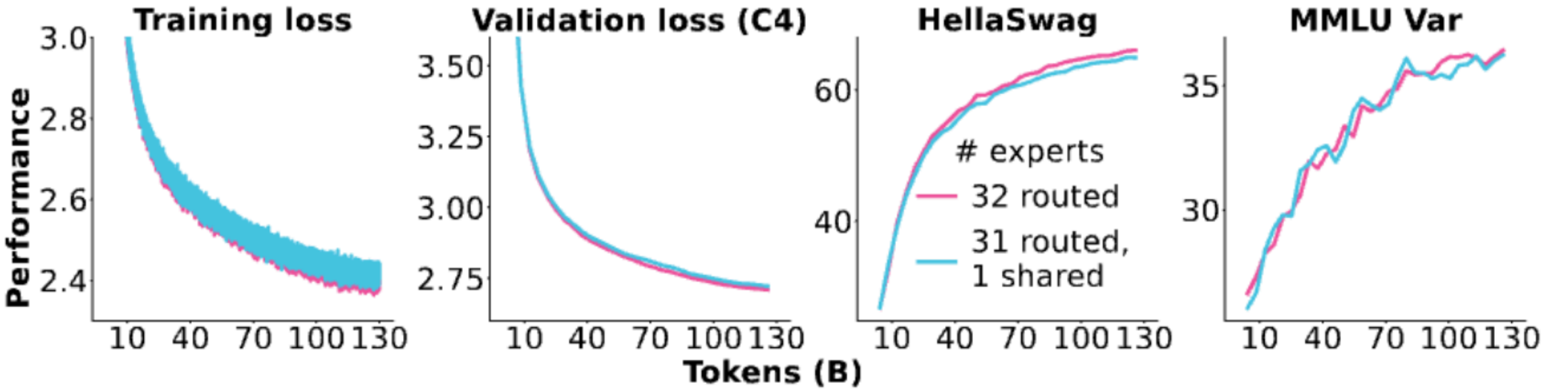
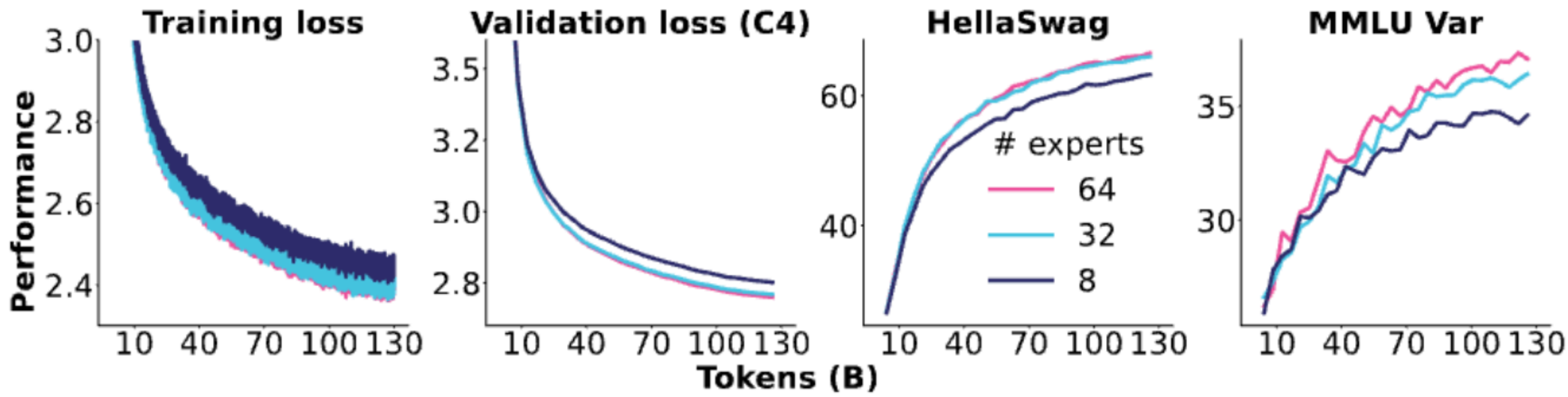


Figure 3 | Ablation studies for DeepSeekMoE. The performance is normalized by the best performance for clarity in presentation. All compared models have the same number of parameters and activated parameters. We can find that fine-grained expert segmentation and shared expert isolation both contribute to stronger overall performance.

Ablations from OlMoE

- AI2's OlMoE: **gains from fine-grained experts**, **no gains from shared experts**





Expert routing setups for recent MoEs

Model	Routed	Active	Shared	Fine-grained ratio
GShard	2048	2	0	
Switch Transformer	64	1	0	
ST-MOE	64	2	0	
Mixtral	8	2	0	
DBRX	16	4	0	
Grok	8	2	0	
DeepSeek v1	64	6	2	1/4
Qwen 1.5	60	4	4	1/8
DeepSeek v3	256	8	1	1/14
OlMoE	64	8	0	1/8
MiniMax	32	2	0	~1/4
Llama 4 (maverick)	128	1	1	1/2

How do we train MoEs?

- Major challenge: we need sparsity for training-time efficiency…But sparse gating decisions are not differentiable!

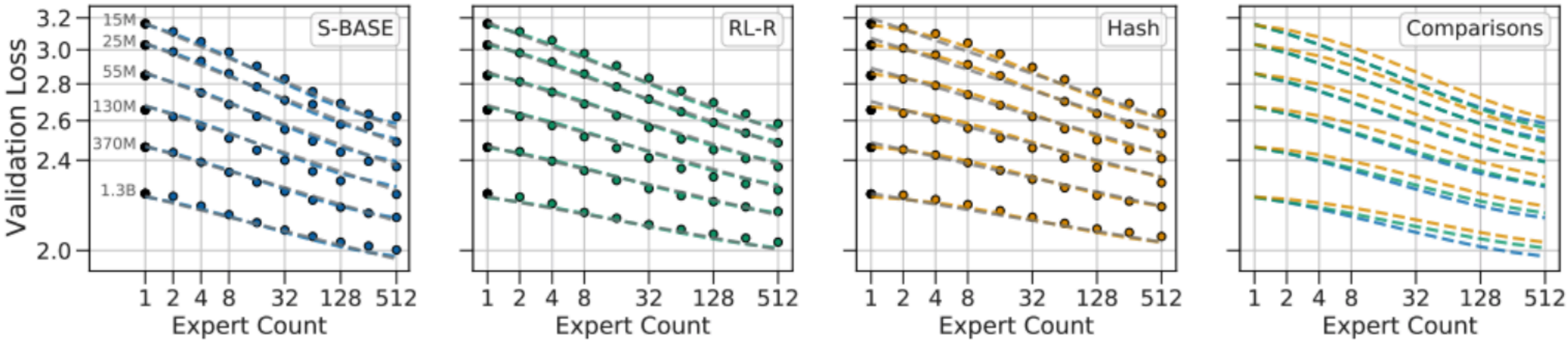
- Solutions?

1. Reinforcment learning to optimize gating policies
2. Stochastic perturbations
3. Heuristic ‘balancing’ losses.

Guess which one people use in practice?

RL for MoEs

- RL via REINFORCE does work, but not so much better that it's a clear win
- RL is the 'right solution' but it has high gradient variances and complexity, which means it's not widely used



(REINFORCE baseline approach, Clark et al 2020)

Stochastic approximations

- From Shazeer et al 2017 – **routing decisions are stochastic with gaussian perturbations.**
  1. This naturally leads to experts that are a bit more robust.
  2. The softmax means that the model learns how to rank K experts

$$G(x) = Softmax(KeepTopK(H(x), k))$$

$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i)$$

$$KeepTopK(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

This has been explored for a while, but gradually been abandoned

Heuristic balancing losses

- Another key issue – systems efficiency requires that we use experts evenly..

For each Switch layer, this auxiliary loss is added to the total model loss during training. Given N experts indexed by $i = 1$ to N and a batch $\mathcal{B}$ with T tokens, the auxiliary loss is computed as the scaled dot-product between vectors f and P,

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i \tag{4}$$

where f_i is the fraction of tokens dispatched to expert i,

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax}\, p(x) = i\} \tag{5}$$

and P_i is the fraction of the router probability allocated for expert i, [2]

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x). \tag{6}$$

Example from deepseek (v1-2)

Per-expert balancing – same as the switch transformer

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

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

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

Per-device balancing – the objective above, but aggregated by device.

$$\mathcal{L}_{\text{DevBal}} = \alpha_2 \sum_{i=1}^{D} f'_i P'_i, \tag{15}$$

$$f'_i = \frac{1}{|\mathcal{E}_i|} \sum_{j \in \mathcal{E}_i} f_j, \tag{16}$$

$$P'_i = \sum_{j \in \mathcal{E}_i} P_j, \tag{17}$$

What happens when removing load balancing losses?

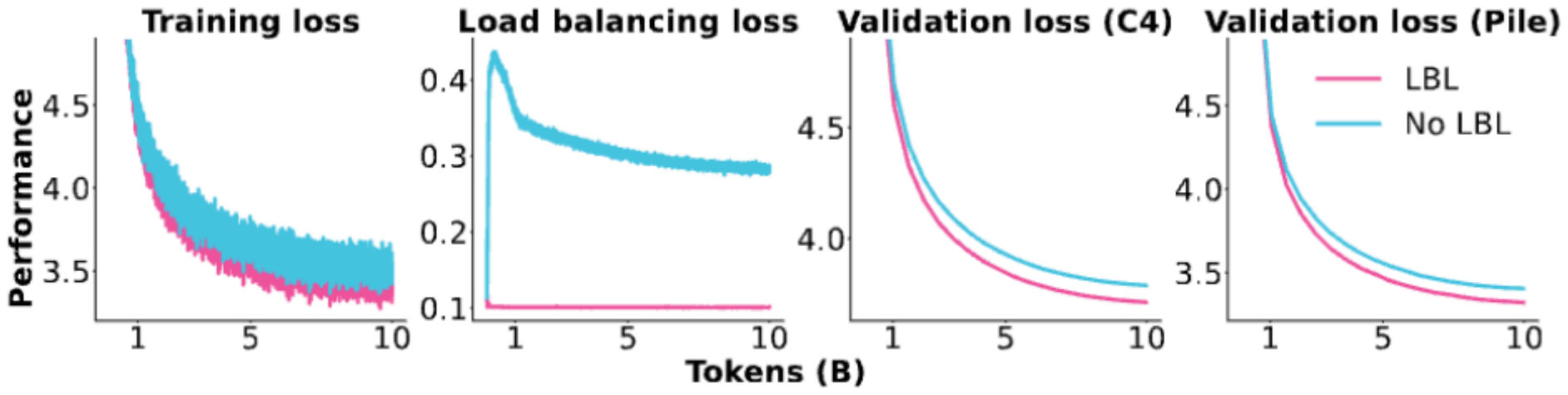


Figure 9: **Impact of applying a load balancing loss (LBL).** The training loss plot excludes the load balancing loss for both models. More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-LBL-vs-No-LBL--Vmlldzo4OTkyNDg4

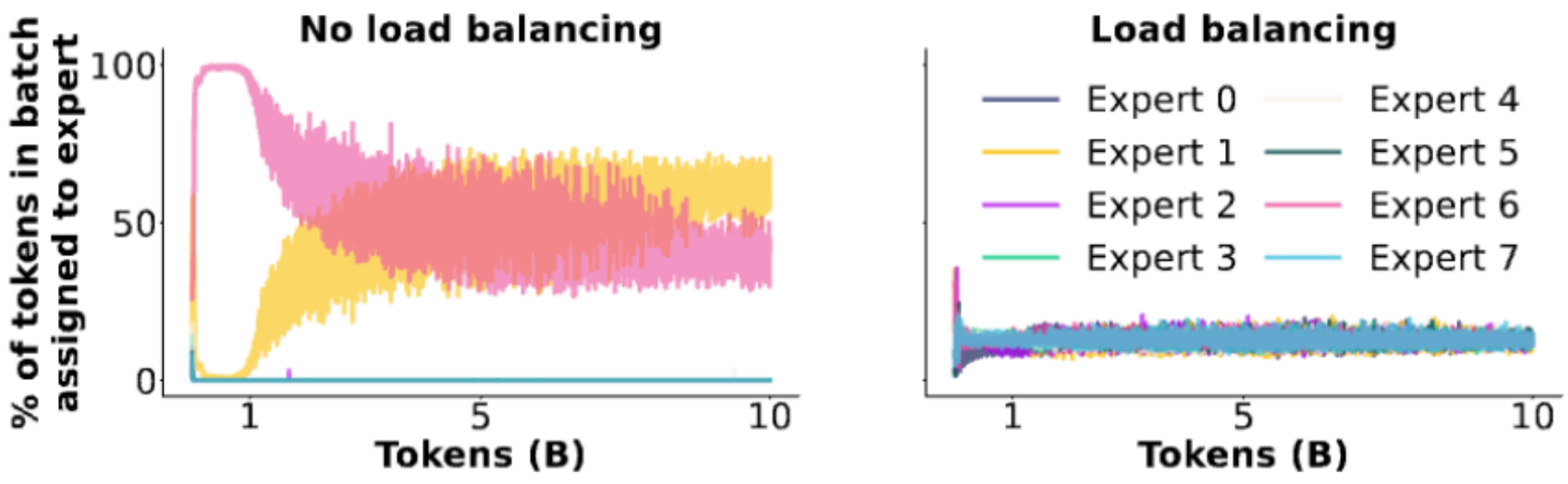
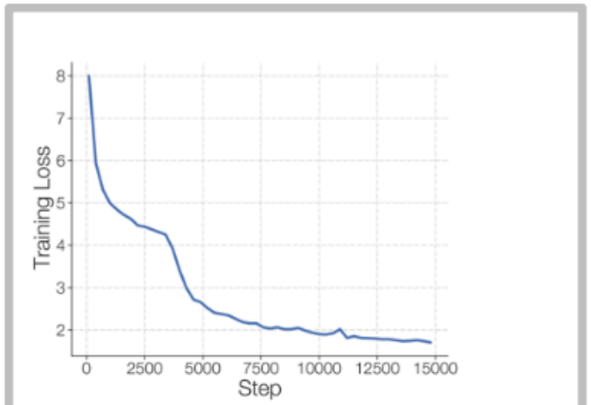


Figure 10: **Expert assignment during training when using or not using a load balancing loss for the first MoE layer.** More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-LBL-vs-No-LBL--Vmlldzo4OTkyNDg4

Issues with MoEs - stability



[7]Exponential functions have the property that a small input perturbation can lead to a large difference in the output. As an example, consider inputting 10 logits to a softmax function with values of 128 and one logit with a value 128.5. A roundoff error of 0.5 in `bfloat16` will alter the softmax output by 36% and incorrectly make all logits equal. The calculation goes from $\frac{\exp(0)}{\exp(0)+10\cdot\exp(-0.5)} \approx 0.142$ to $\frac{\exp(0)}{\exp(0)+10\cdot\exp(0)} \approx 0.091$. This occurs because the max is subtracted from all logits (for numerical stability) in softmax operations and the roundoff error changes the number from 128.5 to 128. This example was in `bfloat16`, but analogous situations occur in `float32` with larger logit values.

[Zoph 2022]

Solution: Use Float 32 just for the expert router (sometimes with an aux z-loss)

$$L_z(x) = \frac{1}{B}\sum_{i=1}^{B}\left(\log\sum_{j=1}^{N} e^{x_j^{(i)}}\right)^2 \tag{5}$$

Z-loss stability for the router

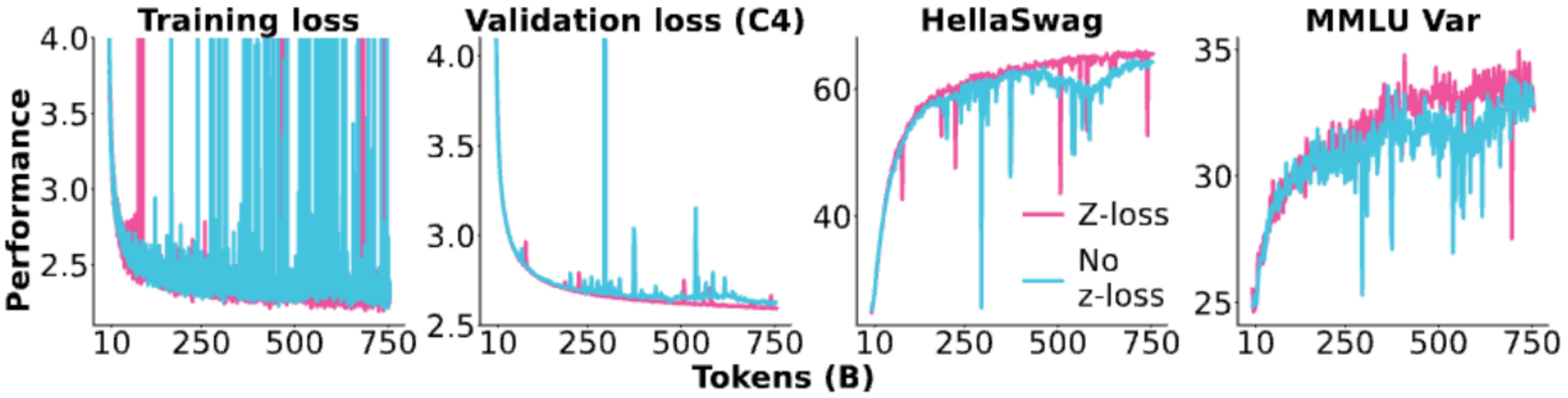
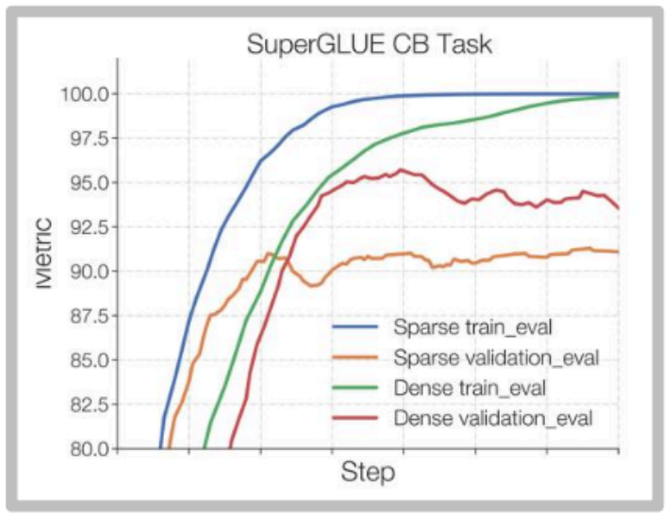


Figure 11: **Router z-loss.** We compare adding router z-loss with a loss weight of 0.001 versus no additional z-loss. More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-Zloss-vs-none--Vmlldzo4NDM4NjUz

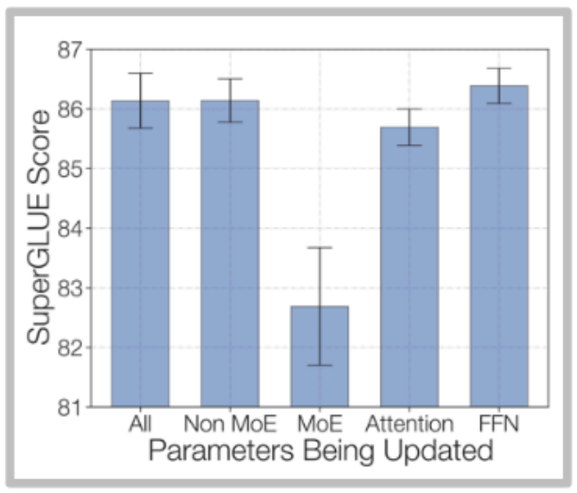
What happens when we remove the z-loss?

Issues with MoEs – fine-tuning

Sparse MoEs can overfit on smaller fine-tuning data



Zoph et al solution – finetune non-MoE MLPs



DeepSeek solution – use lots of data 1.4M SFT

Training Data. For training the chat model, we conduct supervised fine-tuning (SFT) on our in-house curated data, comprising 1.4M training examples. This dataset spans a broad range of categories including math, code, writing, question answering, reasoning, summarization, and more. The majority of our SFT training data is in English and Chinese, rendering the chat model versatile and applicable in bilingual scenarios.

Summary

- MoEs take advantage of sparsity – not all inputs need the full model
    - Discrete routing is hard, but top-k heuristics seem to work
- Lots of empirical evidence now that MoEs work, and are cost-effective

Questions?