CS11-711 Advanced NLP

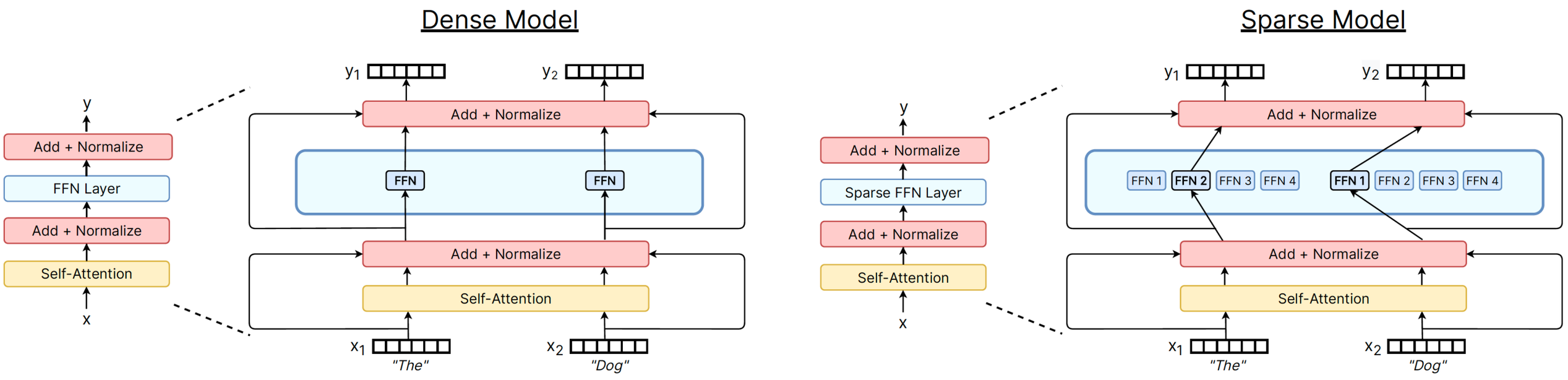
Mixture of Experts

Sean Welleck

Acknowledgements: Stanford CS336 Spring 2025 Lecture 4

Mixture of experts

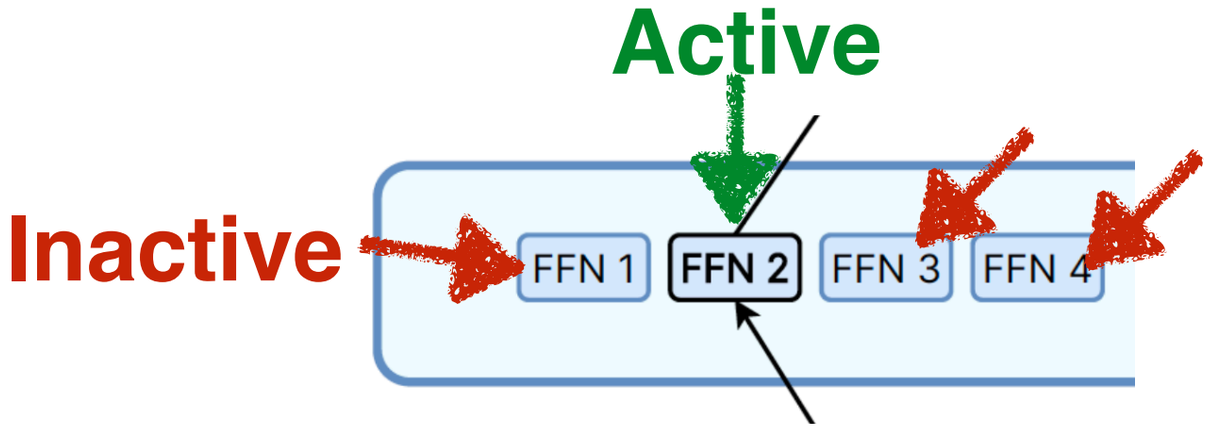
- Key idea: replace Transformer feed-forward network (FFN) layer with a new layer



[Fedus et al 2022]

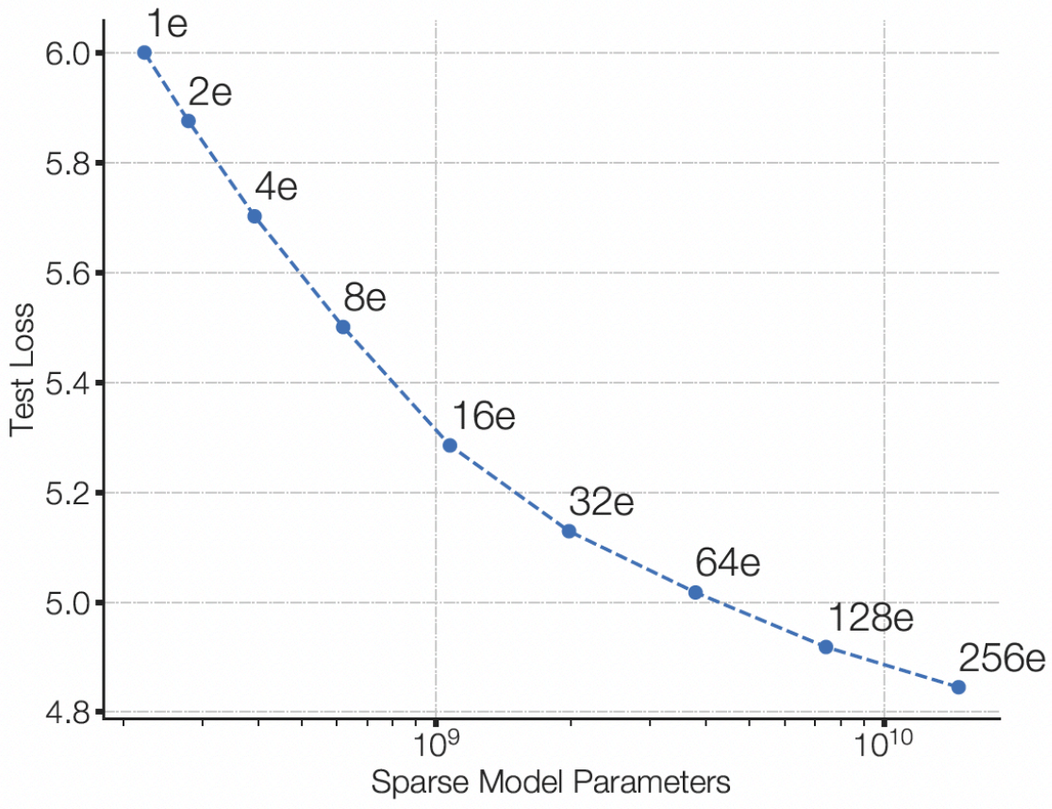
Mixture of experts

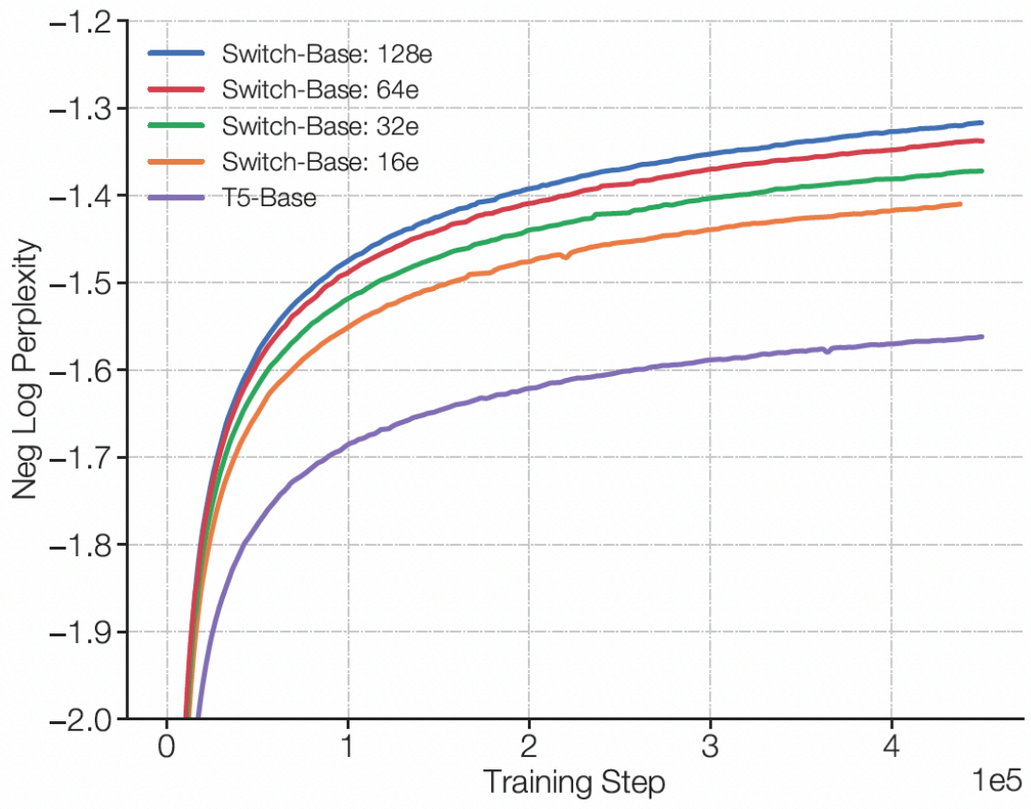
- Key idea: **sparsity**
  - The model has many *total* parameters, but only a relatively smaller number of them are **active** for training and inference



New dimension of scaling

- Scale parameter count while keeping the amount of computation (FLOPs per example) fixed

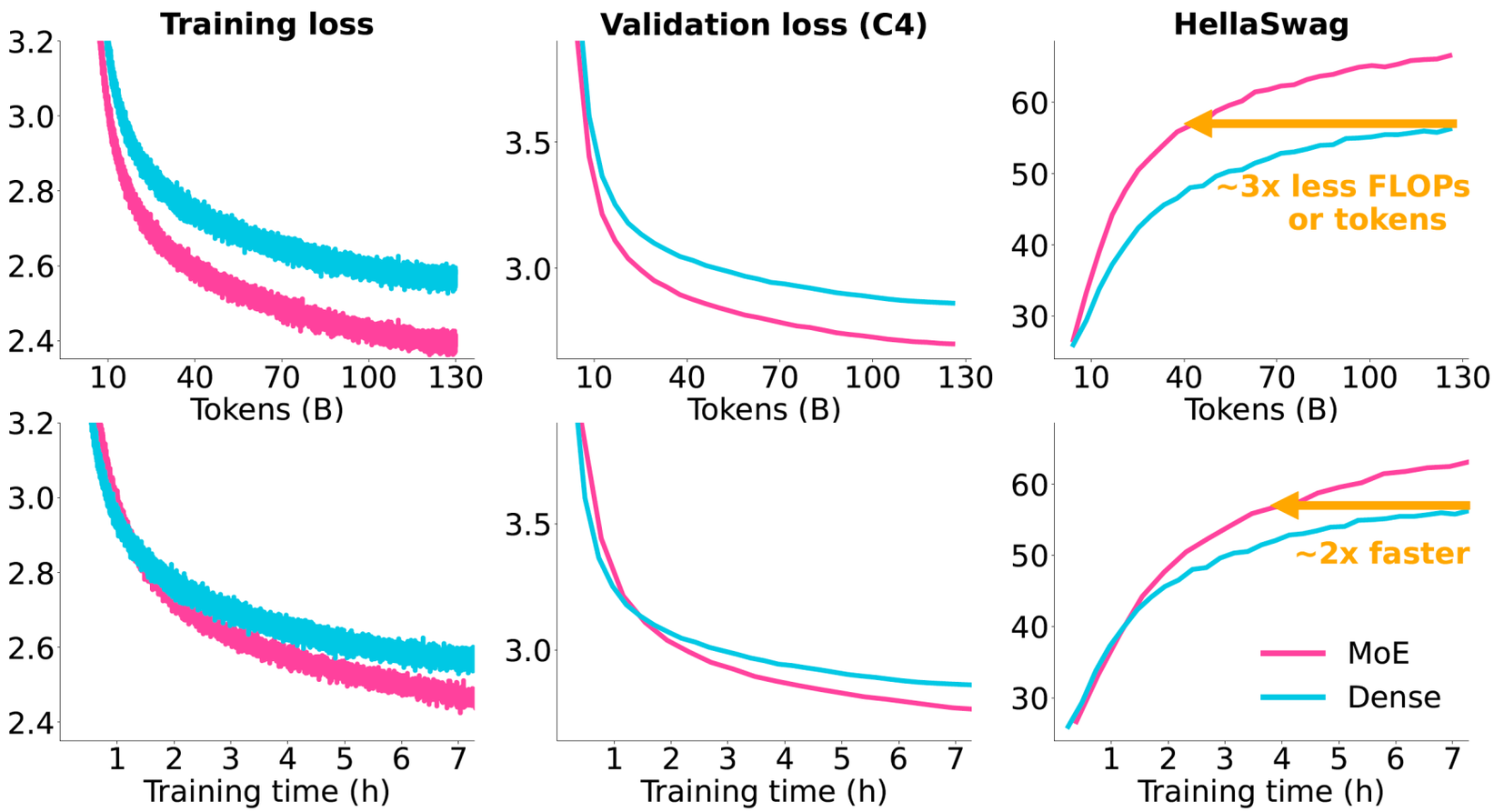




[Fedus et al 2022]

More efficient training

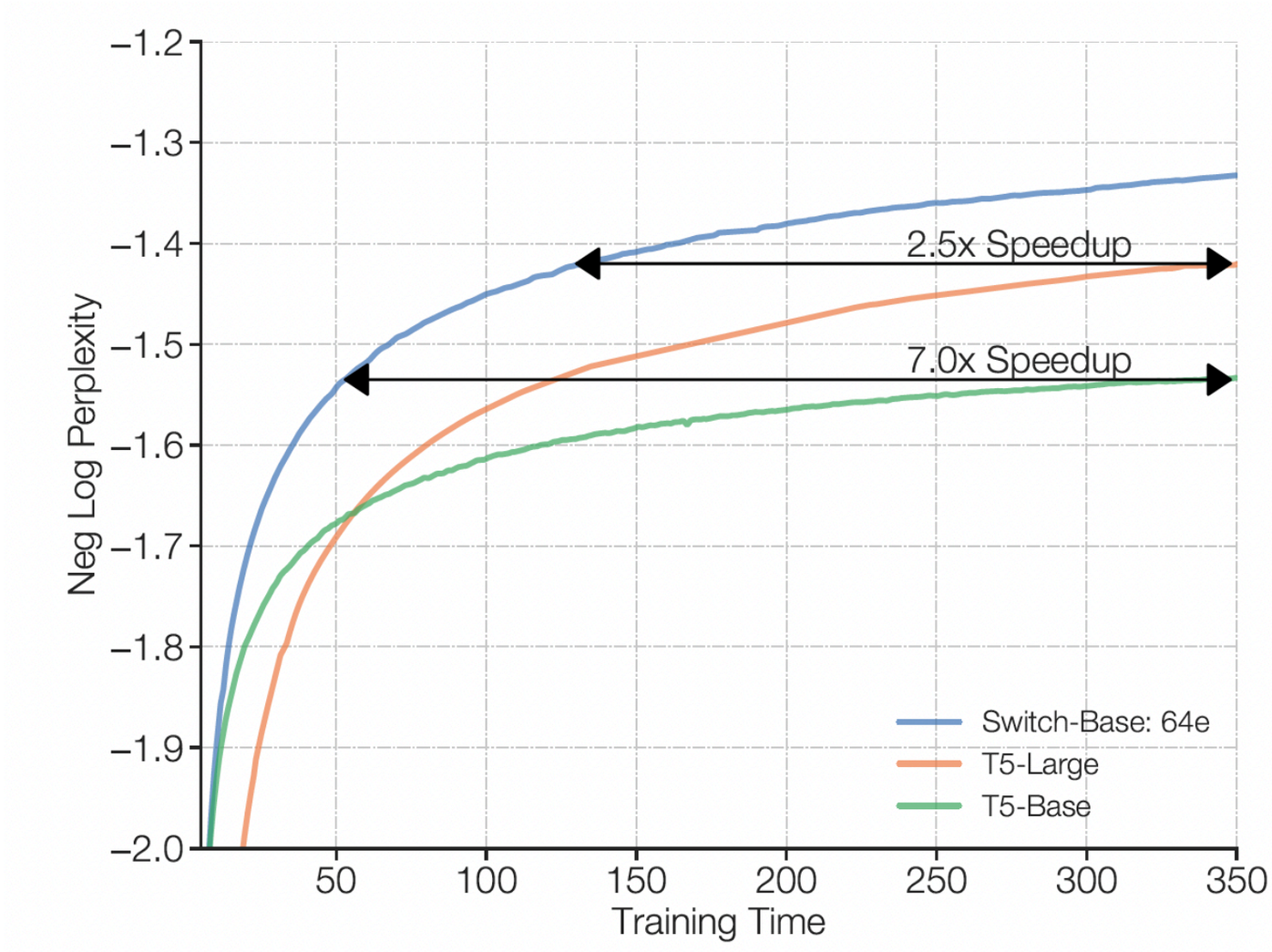
- Fewer FLOPs needed reach a given level of performance compared to the standard dense architecture



[AI2 2025]

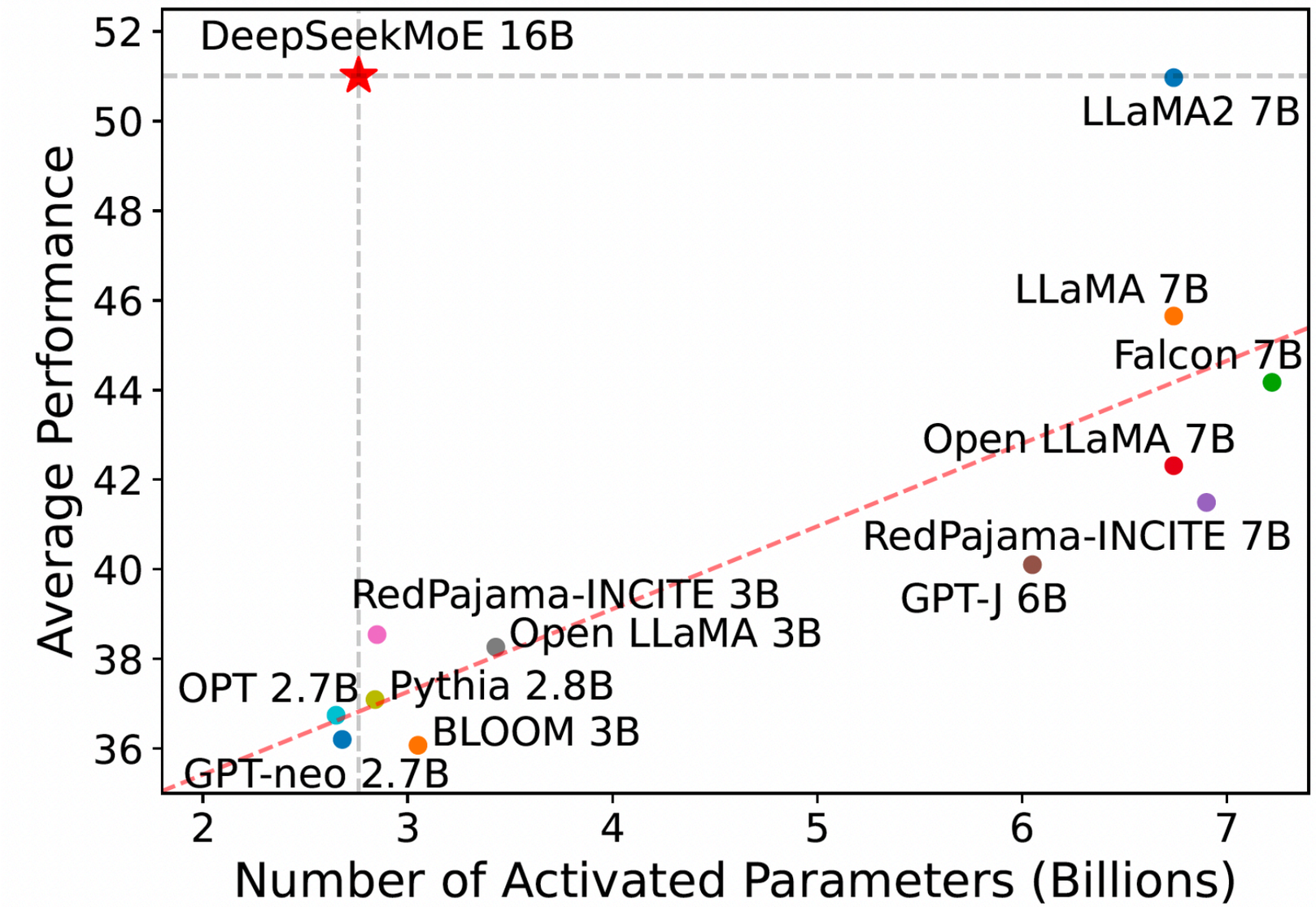
More efficient training

- Faster *wall-clock time* to reach a given level of performance:



[Fedus et al 2022]

Better inference-cost vs. performance tradeoff



[DeepSeek 2024]

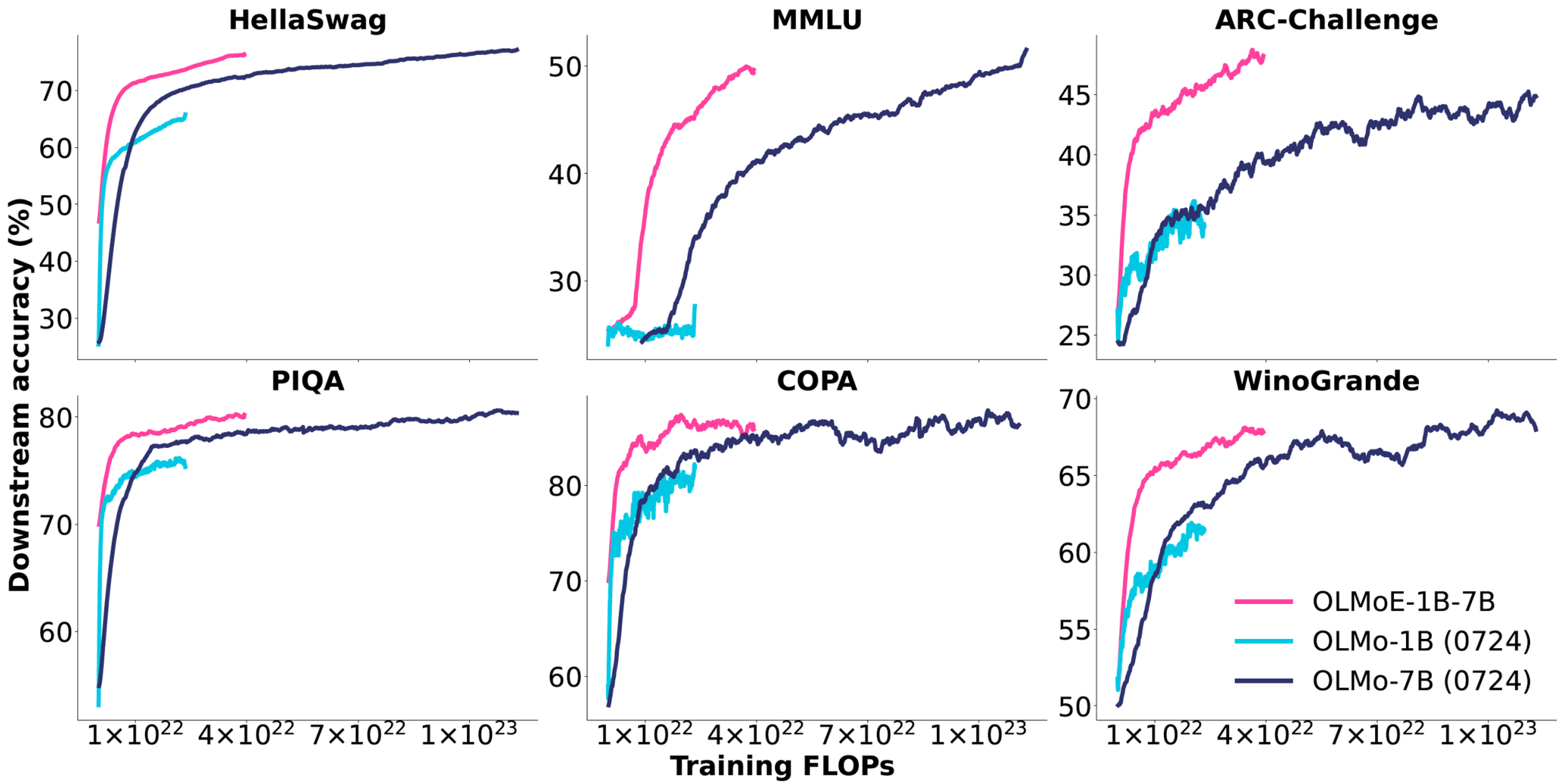
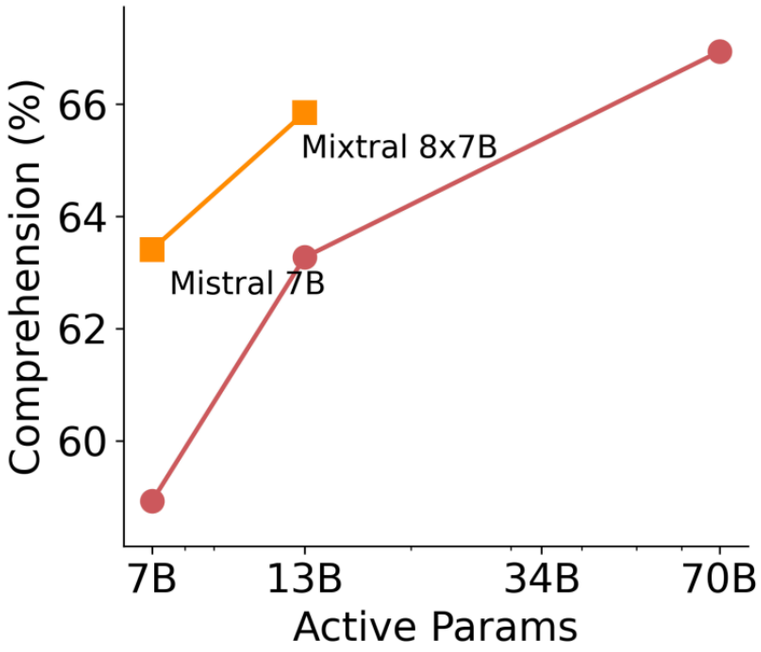
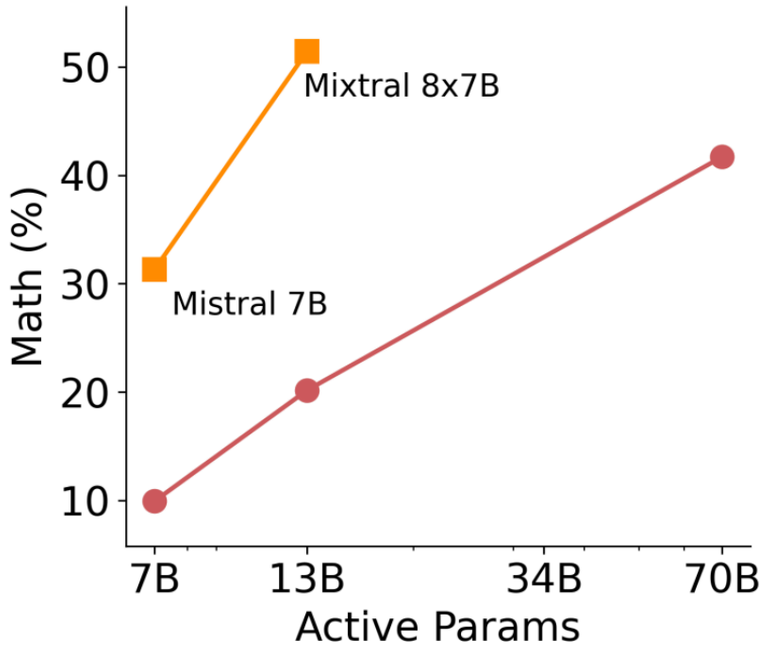
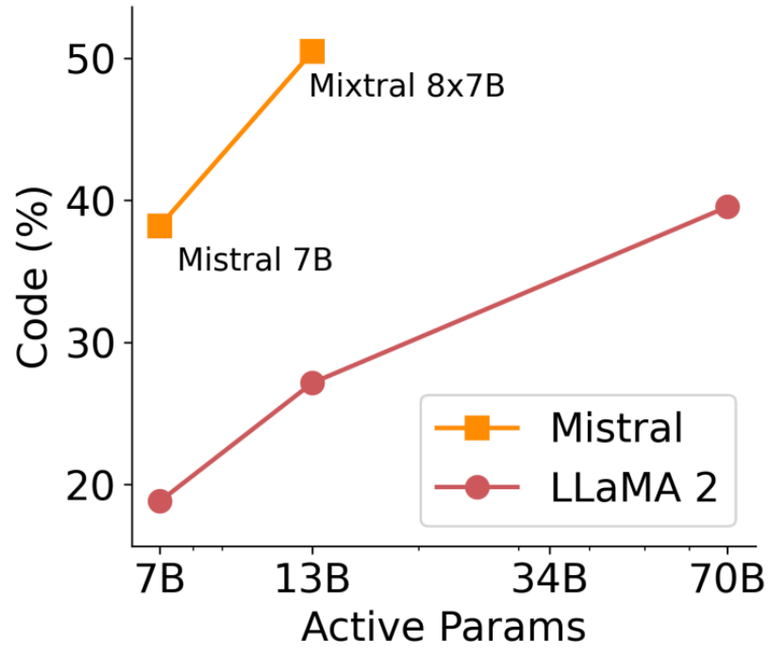


Figure 3: **Evaluation of OLMoE-1B-7B and the current best OLMo models during pretraining.** **OLMoE-1B-7B** differs from the OLMo models in its MoE archi-

[AI2 2025]







[Mistral 2024]

Used in top LLMs

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 model (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.

Gemini 3.0 [11/18/2025]

openai.com/index/introducing-gpt-oss/

Each model is a Transformer which leverages mixture-of-e
(MoE[2]) to reduce the number of active parameters neede

gpt-oss

Qwen3-30B-A3B

💜 Qwen Chat

Qwen3 Highlights

Qwen3 is the latest generation of large language models in Qwen series
comprehensive suite of dense and mixture-of-experts (MoE) models. Bu

Qwen 3

Potential downsides

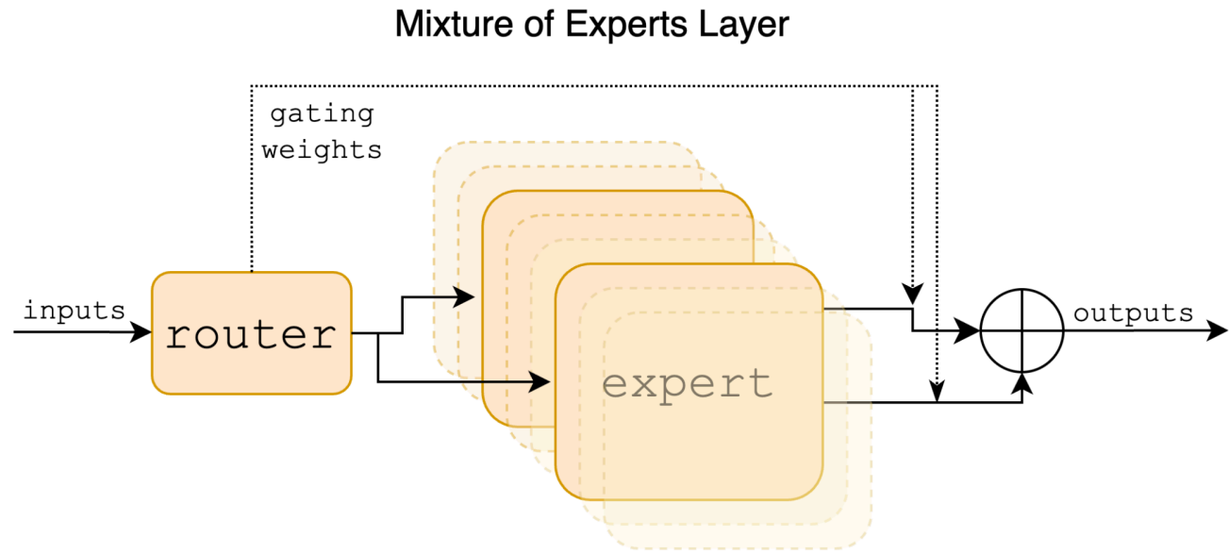
- Increased system complexity
  - Need to coordinate usage of the experts
  - New losses and new hyperparameters
  - Manage a lot of devices to host all of the parameters
- Can be unstable / tricky to train

This lecture

- What are MoEs
- Expert design
- Routing functions
- Loss functions
- Systems
- Case studies

What are MoEs

- A neural network layer that only activates some parameters
- In current state-of-the art LMs, the feedforward layer in a Transformer block



[Mistral 2024]

What are MoEs

- Standard Transformer layer:

$$\tilde{h}_{1:T} = \text{Attn}(h_{1:T}) + h_{1:T}$$
$$h'_{1:T} = \text{FFN}(\tilde{h}_{1:T}) + \tilde{h}_{1:T}$$

- MoE layer: replace $\text{FFN}(h_{1:T})$ with:

$$\tilde{h}_{1:T} = \text{Attn}(h_{1:T}) + h_{1:T}$$
$$h'_{1:T} = \text{MoE}(\tilde{h}_{1:T}) + \tilde{h}_{1:T}$$

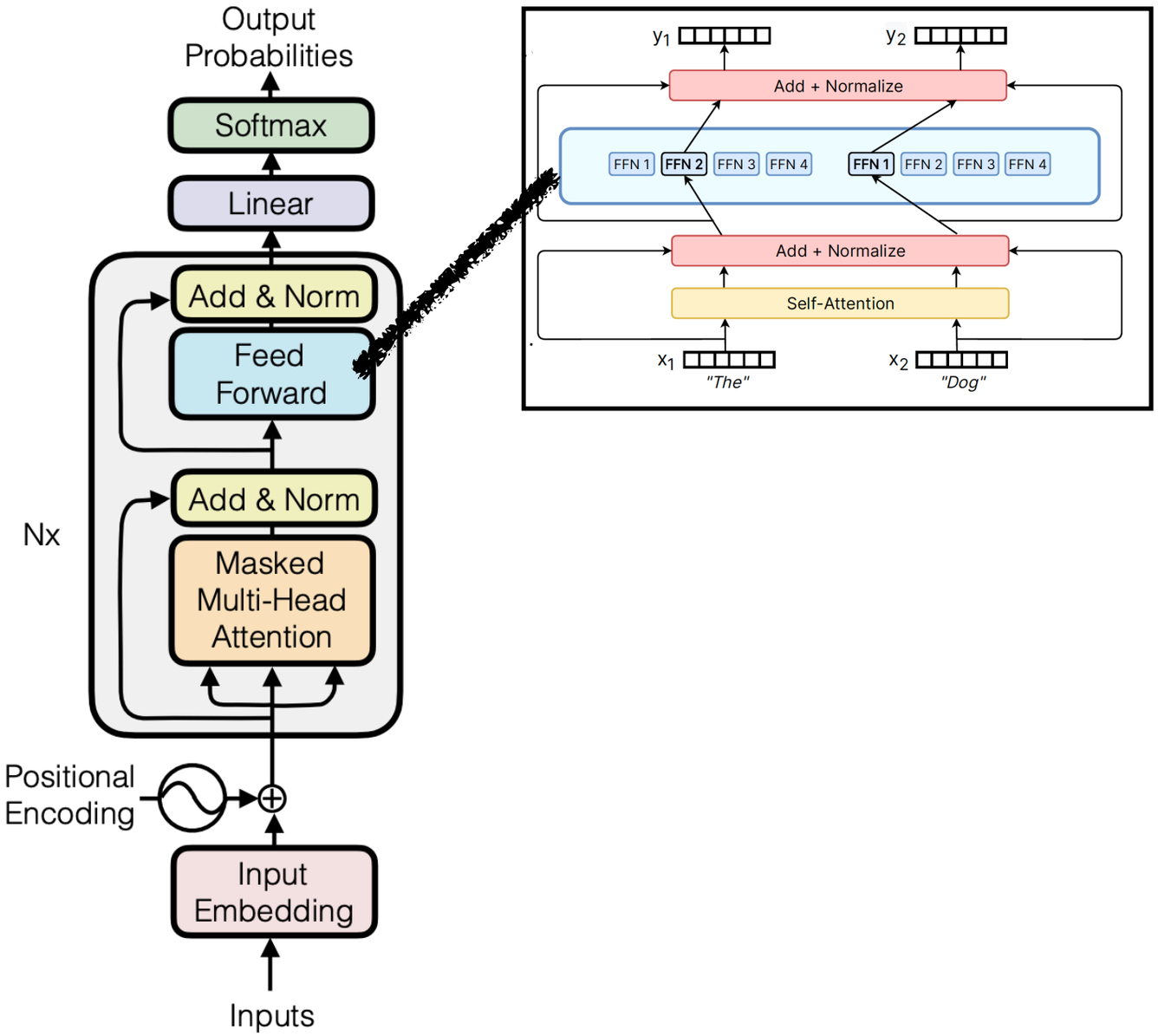
- Internally, the MoE has:
  - Feedforward networks $\text{FFN}_1, \ldots, \text{FFN}_N$ ("experts") with params $\theta_1, \ldots, \theta_N$
  - A parameterized **router** that chooses 1 or more FFNs to use for each input

Expert design

- Considerations
  - Where to put the experts
  - Expert count
  - Size of each expert
  - Number of shared experts

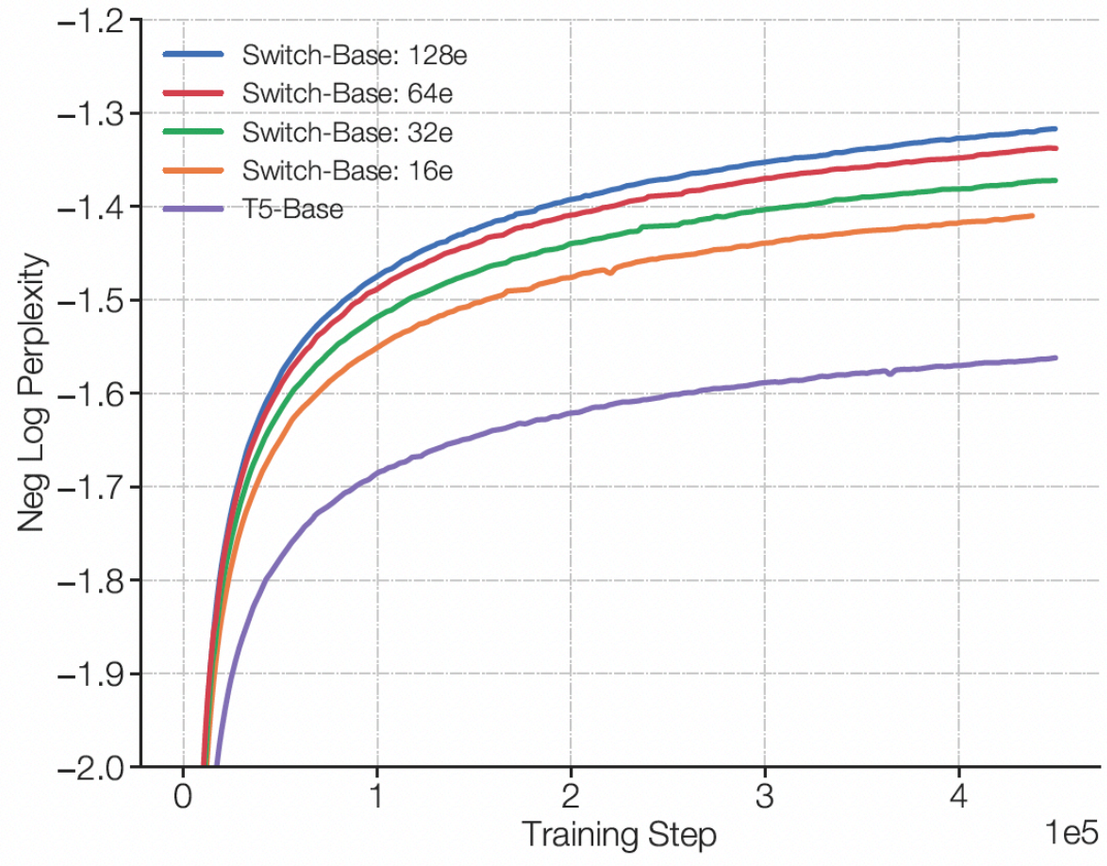
Where to put the experts

- Feed forward layer
- Less common: attention layer
  - Can be unstable
- Nowadays, typically every FFN layer



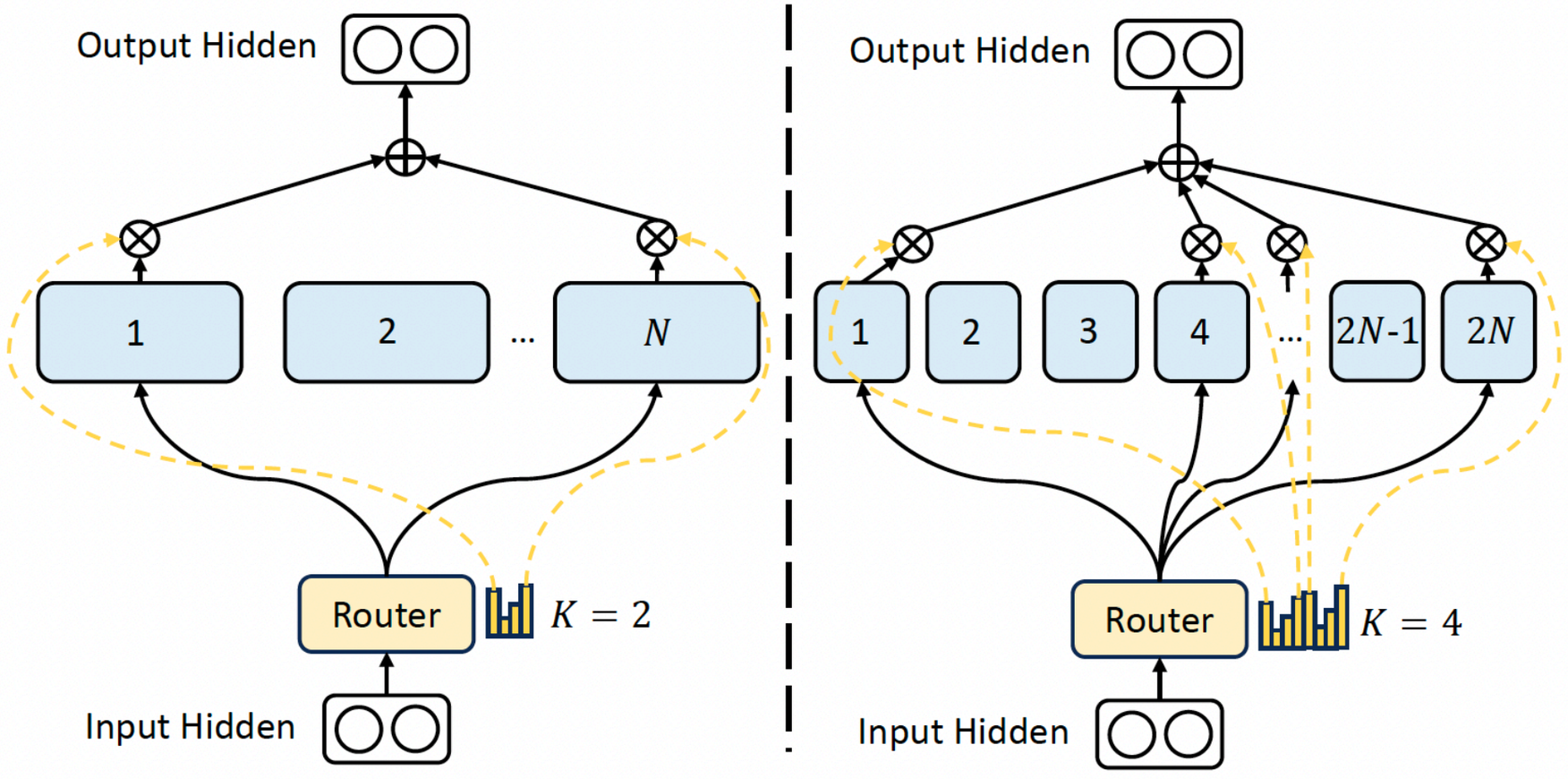
Expert count | increase params

- **Strategy 1:** Increase number of parameters
  - 2N, 4N, 8N, 16N, ….
  - More is better, but with diminishing returns
- More parameters => more computational resources



Expert count | fine-grained

- **Strategy 2**: increase number of experts for a fixed number of parameters by splitting the hidden dimension
  - "Fine-grained expert segmentation" [DeepSeek 2024]



Expert count | fine-grained

- Make more experts of dimension d/n for some integer *n.* Example:
  - **Baseline**: 8-experts per layer of dimension d, 2 activated
  - **Fine-grained**: 16-experts per layer of dimension $d/2$, 4 activated
- Increases flexibility
  - Baseline => 28 combinations per layer
  - Fine-grained => 1,820 combinations per layer

Fine-grained experts

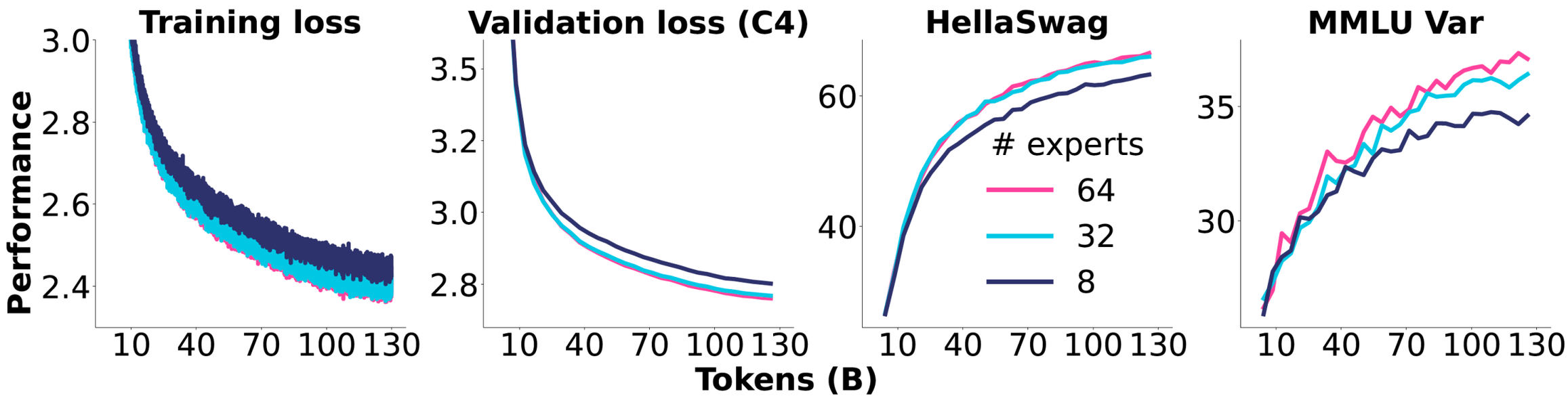
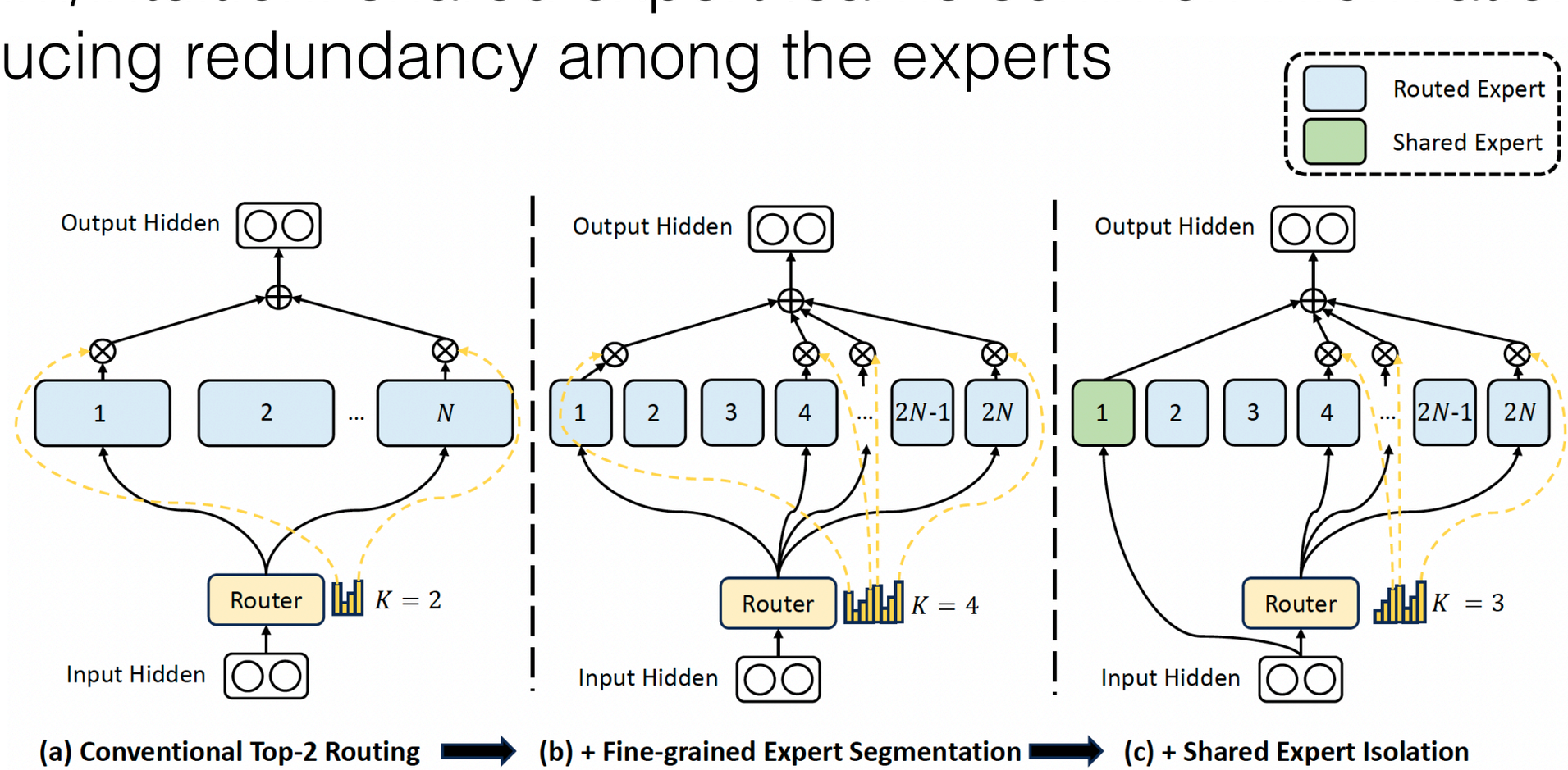


Figure 5: **Expert granularity.** We vary the number of experts in tandem with the FFN dimension to ensure that active and total parameters and thus compute cost remain the same. For example, for 64 experts, the FFN dimension is 1,024 and 8 experts are activated, while for 32 experts it is 2,048 with 4 activated experts. More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-Granularity--Vmlldzo4OTIxOTE4

- Diminishing returns in practice after a certain granularity
- Could lead to more communication costs if not managed well

Shared experts

- Always keep a fixed set of experts active, route to the rest
  - E.g. 1 shared expert, 3 experts selected by routing from the 31 experts
- Claim/intuition: shared expert learns common information, reducing redundancy among the experts



Shared experts

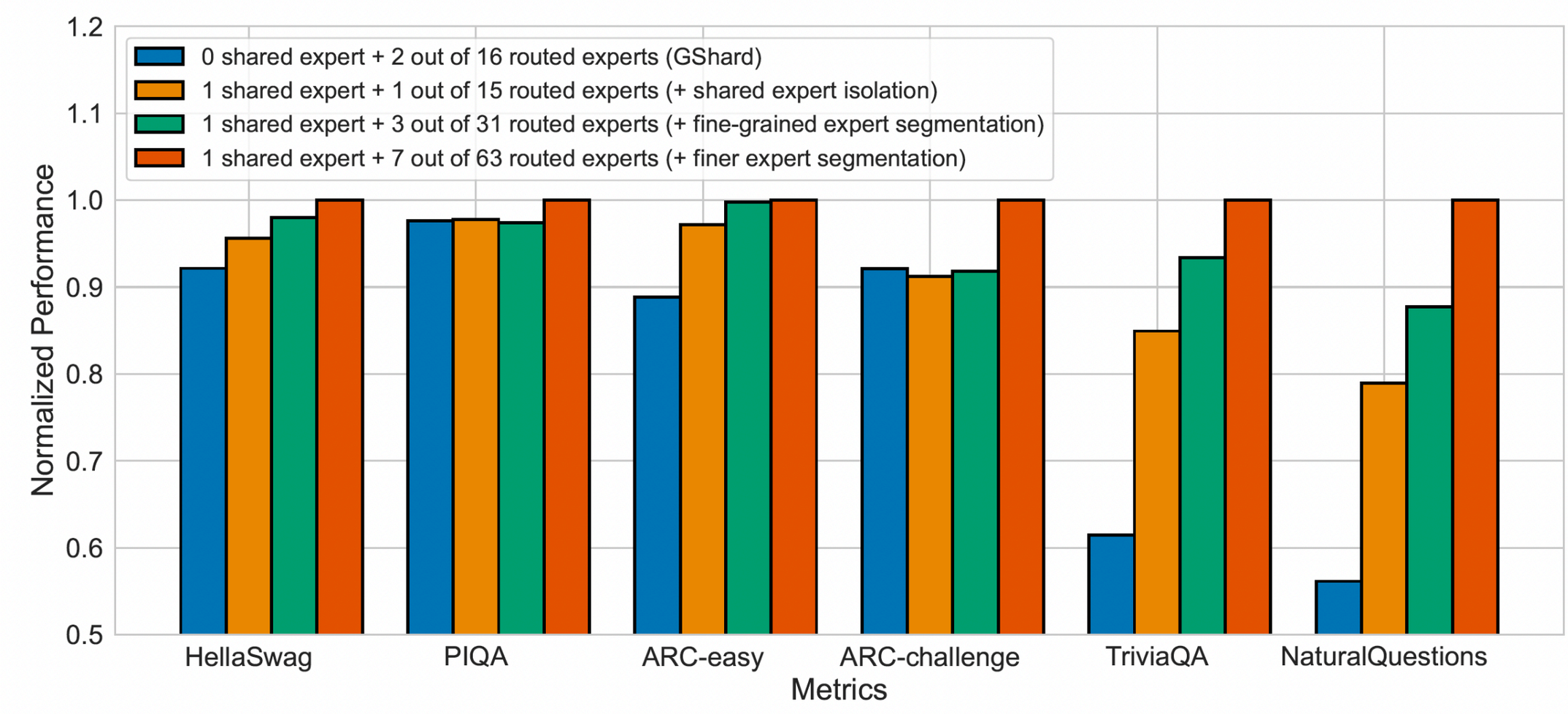


Figure 3 | Ablation studies for DeepSeekMoE. The performance is normalized by the best perfor-

Shared experts

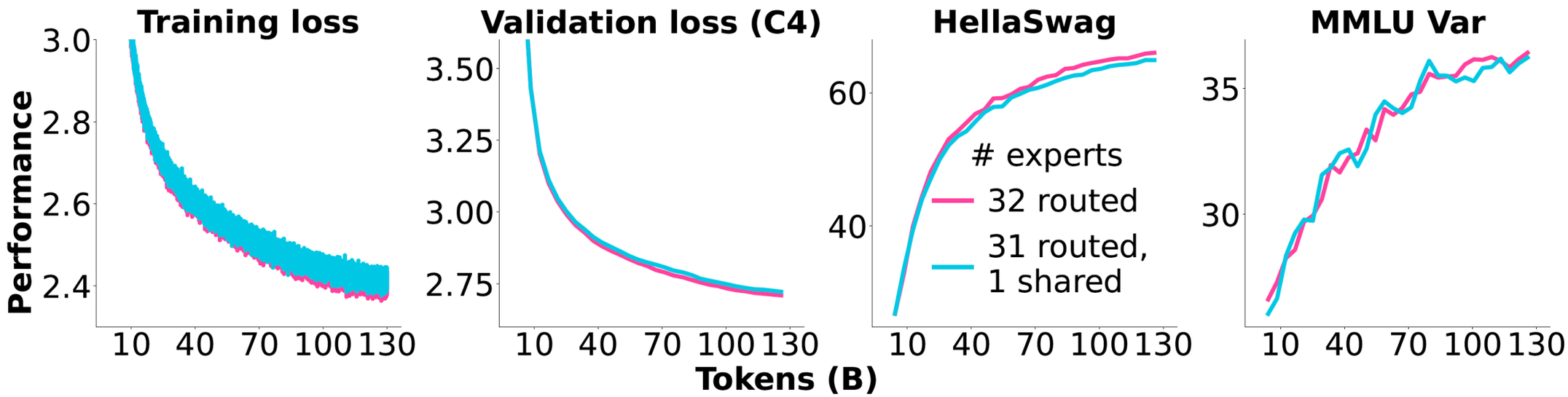


Figure 6: **Shared experts.** Both setups have the same number of active and total parameters and use the same number of FLOPs. 4 of the 32 routed experts are activated, while it is 3 for the 31 routed experts of the other model, as it has 1 always-active shared expert. More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-Expert-sharing--Vmlldzo4OTIyMjQz

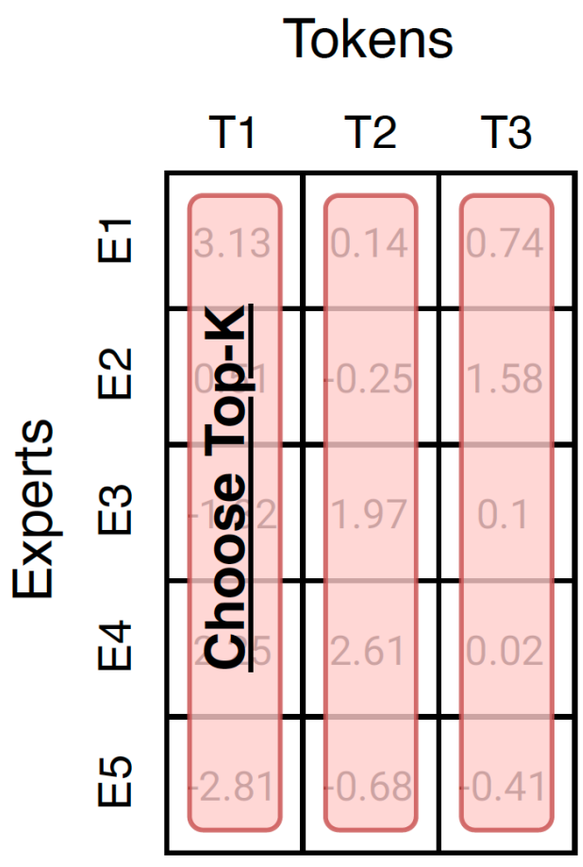
- Mixed results across research studies
- Reduces modeling flexibility

This lecture

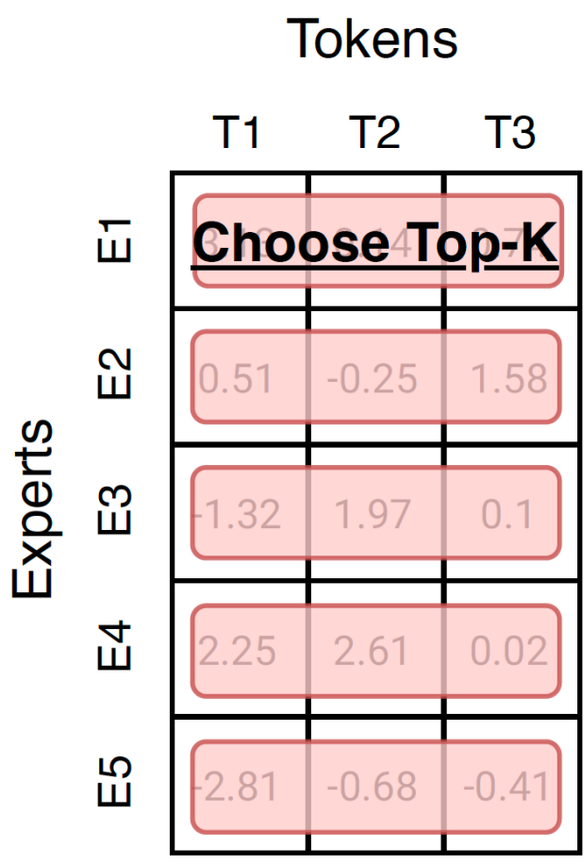
- What are MoEs
- Expert design
- **Routing functions**
- Loss functions
- Systems
- Example

Routing: basic idea

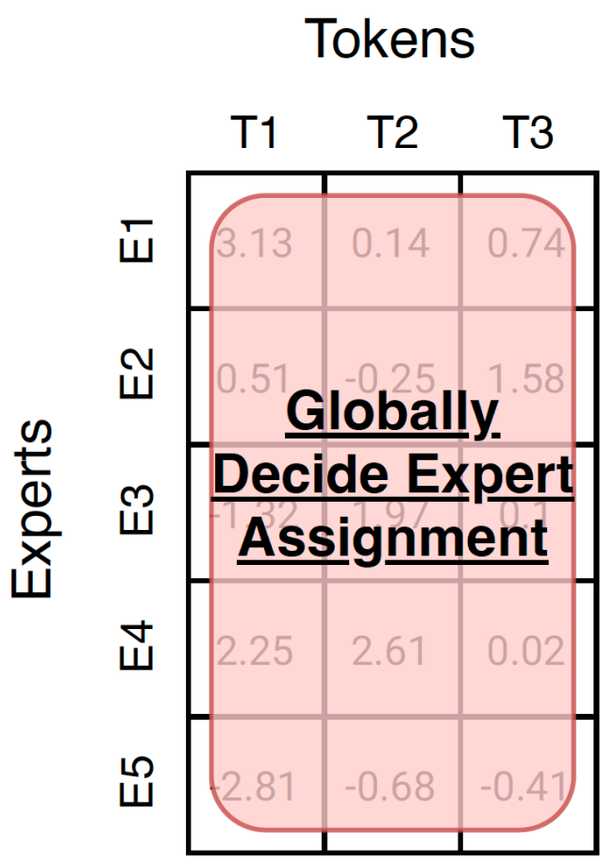
- Compute a score for each (token, expert) pair
- Choose experts according to some strategy



- **Token choice**: Each token chooses top-*k* experts



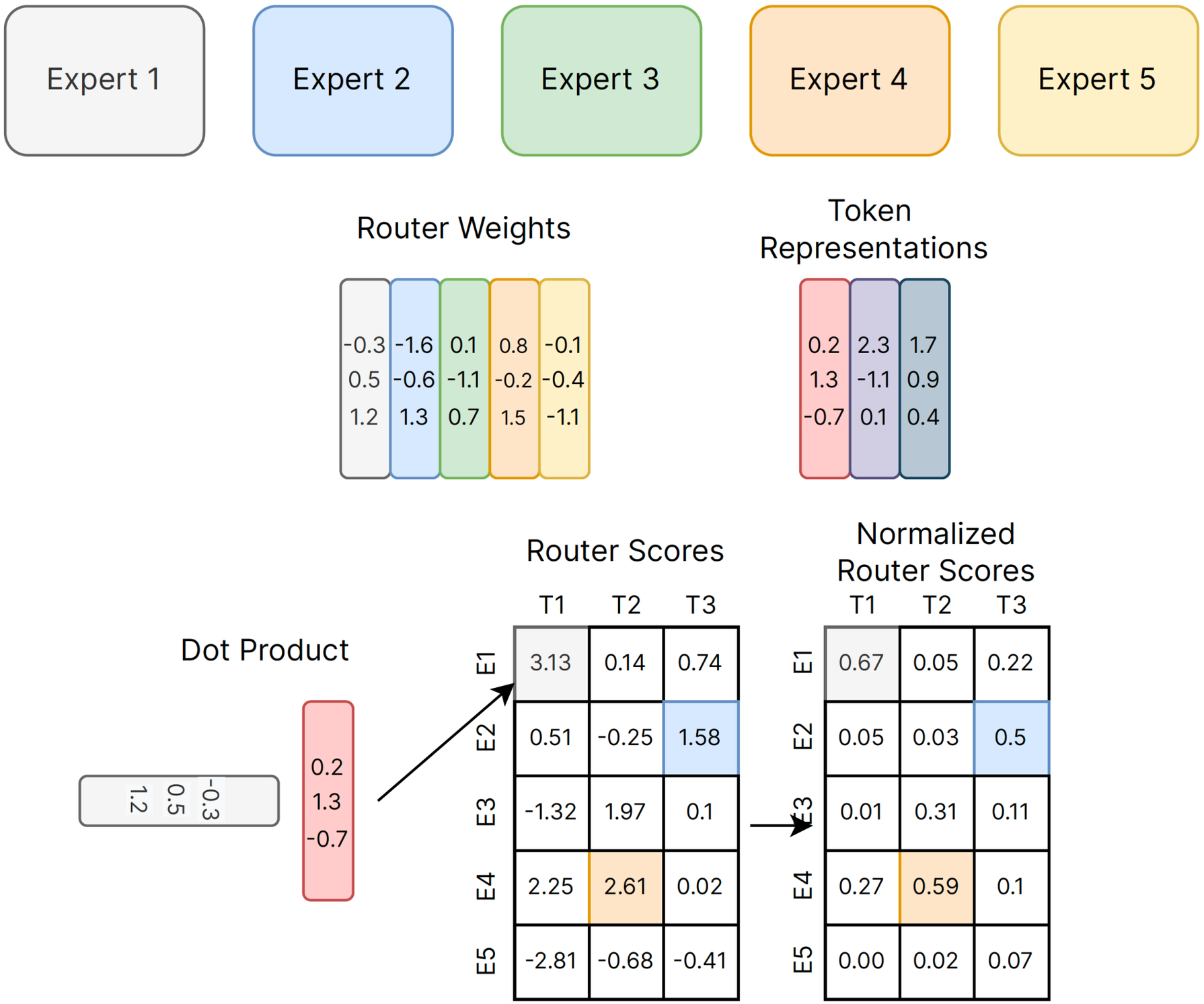
- **Expert choice**: Each expert chooses top-*k* tokens



- **Global choice**: Each expert chooses top-*k* tokens

Computing routing scores

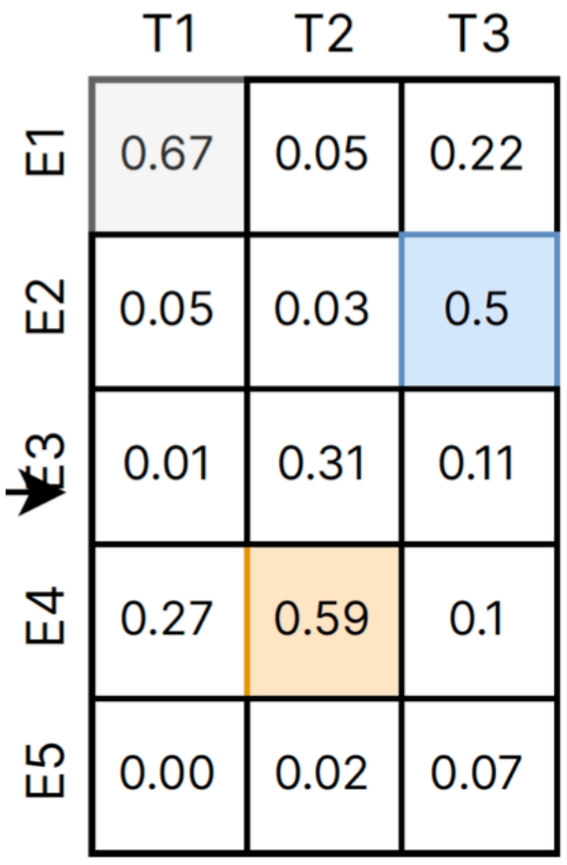
- Surprisingly simple:
  - Introduce a weight vector for each expert
  - Take dot products between each token's hidden state and each expert's weight vector



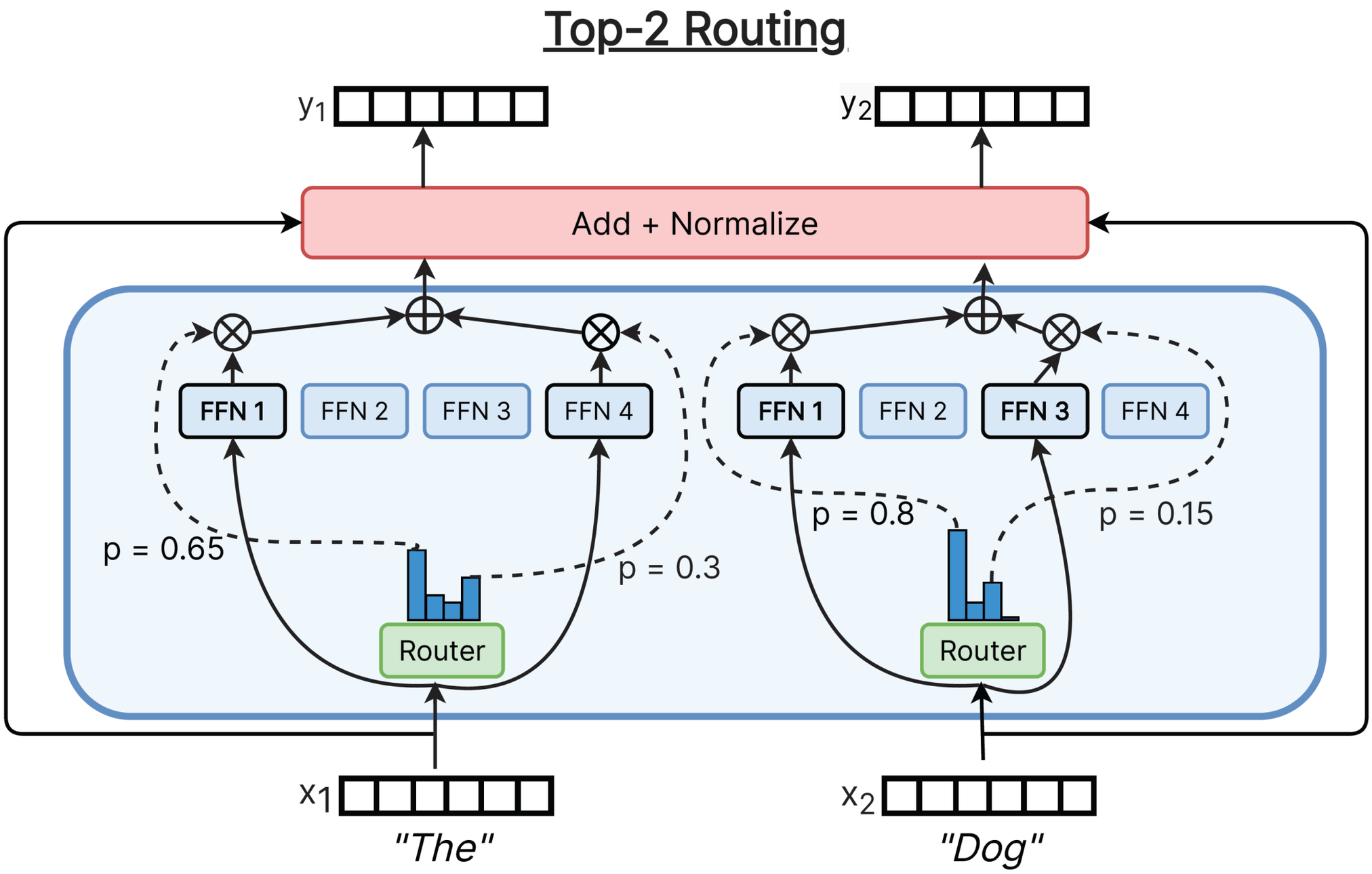
[Fedus et al 2022]

Routing functions

- Given a matrix of scores, now we want to select a set of experts for each token
- I.e., “route” each token to a set of experts



Top-k token-choice routing



Top-k token-choice routing

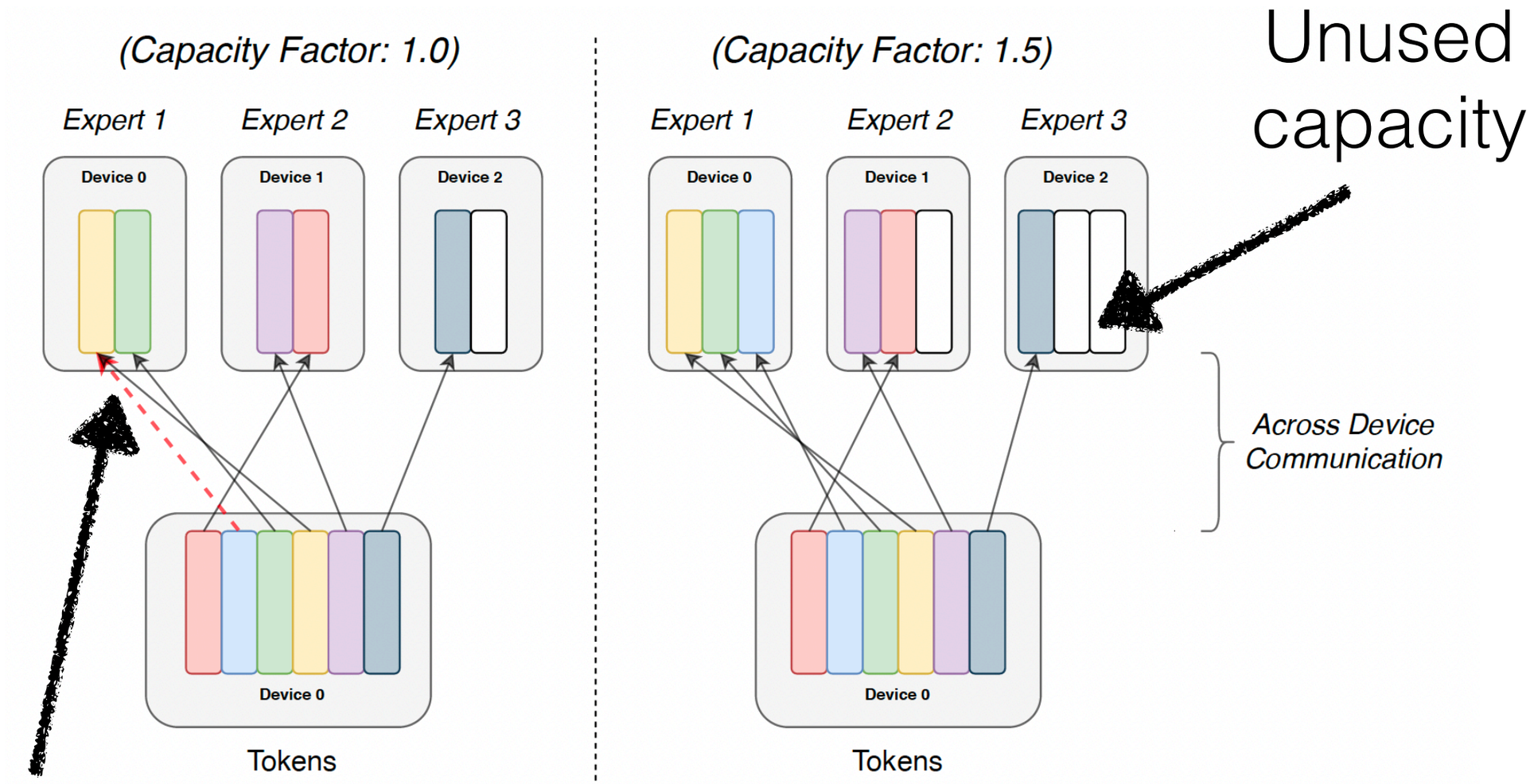
$$s_{t,i} = \text{softmax}_i\left(\tilde{h}_t^T e_i\right)$$

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

$$h'_t = \sum_{i=1}^{N} \left(g_{i,t} \text{FFN}_i(\tilde{h}_t) \right) + \tilde{h}_t$$

Load balancing and capacity

- Failure modes: expert gets too many tokens, expert gets too few tokens



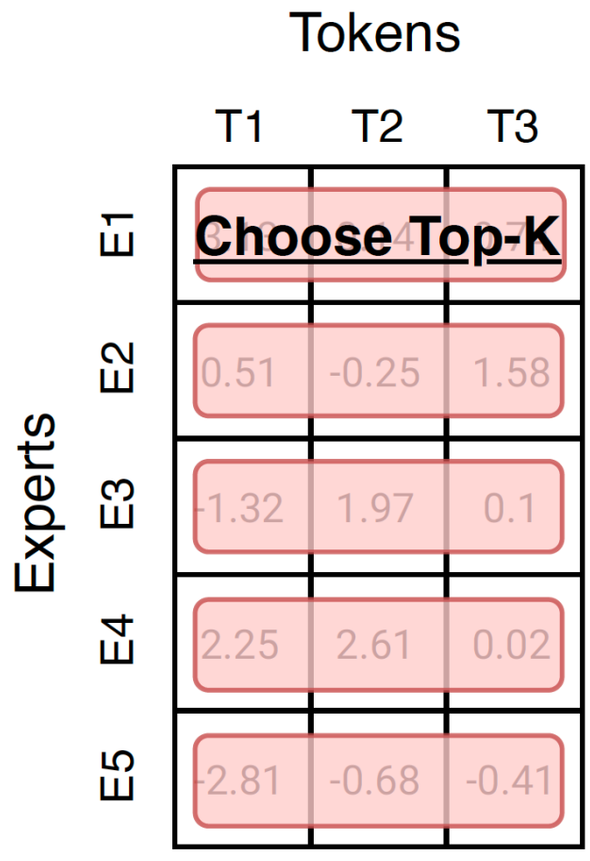
"Token dropping"

Top-k token-choice routing

- Pros
  - Ensures that each token is assigned to experts
  - Relatively simple
- Potential cons
  - **Load balancing** is tricky
    - Example: every token picks the same expert
    - **Token dropping**: when too many tokens are routed to an expert so we have to drop some (e.g., due to memory)
  - Selection operation is not differentiable
- **Keep in mind**: every major MoE LLM nowadays uses top-k token choice routing!

Alternatives: expert choice

- Each expert chooses a fixed number of tokens
- Pros
  - Solves load balancing at training time
  - Model can allocate more capacity to certain tokens
    - I.e., a token may be selected by multiple experts
- Cons
  - At generation time, we only have one token being processed at a time
    - We would need to design a selection mechanism
  - Token dropping: a token is not selected by any experts



Alternatives: expert choice

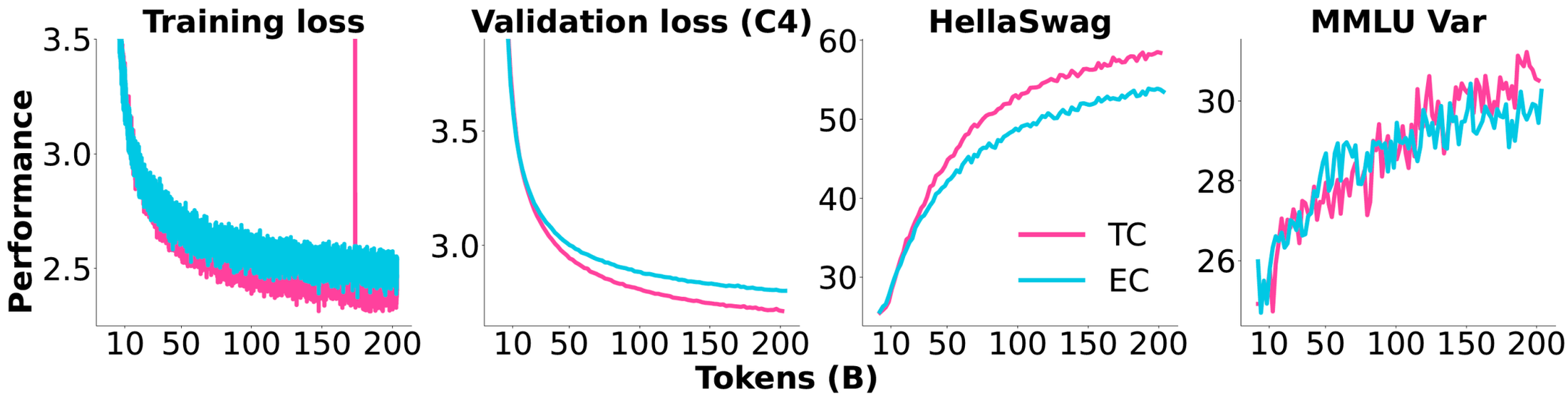
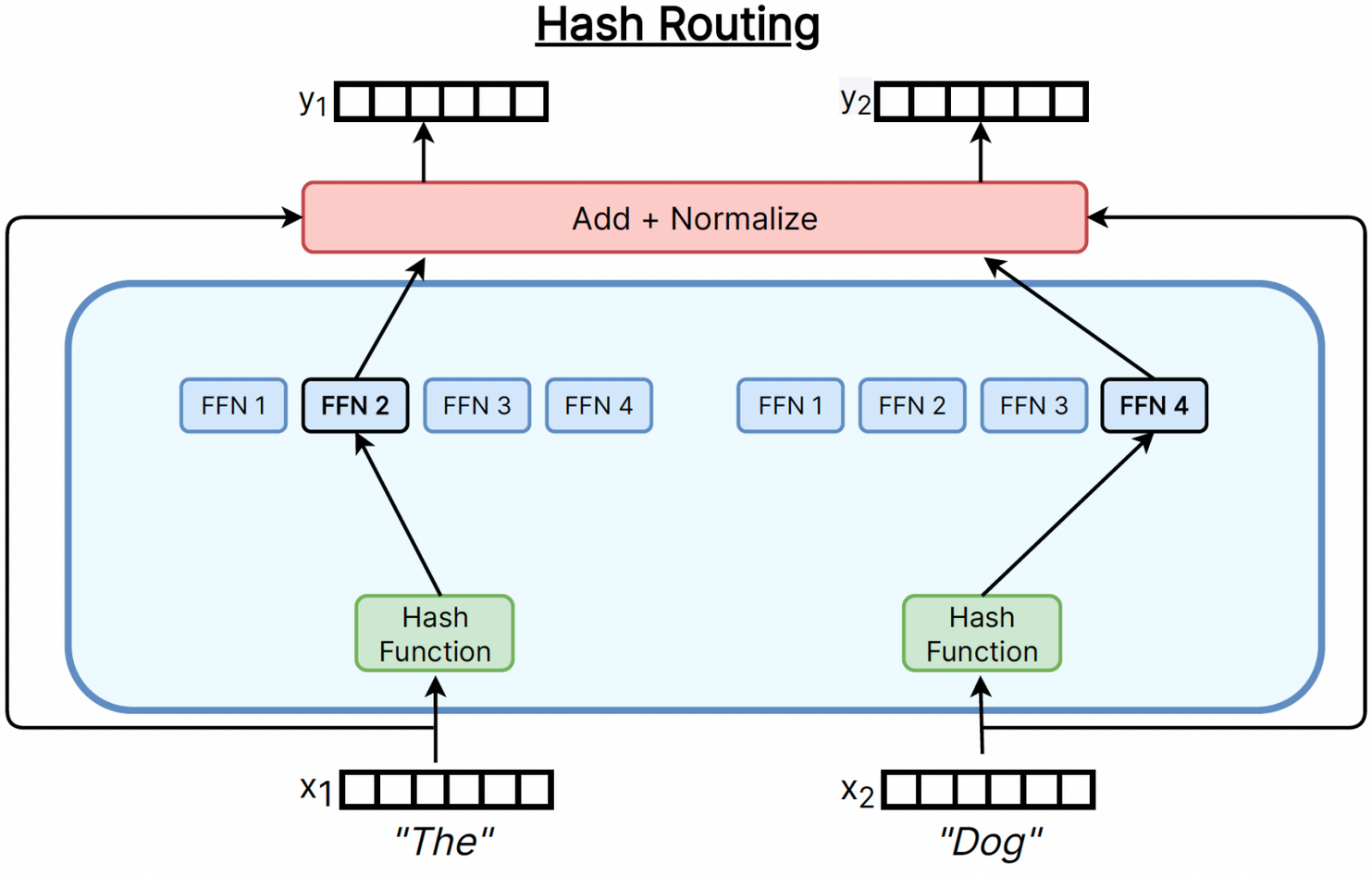


Figure 7: **Expert choice (EC) vs. token choice (TC).** Both models have an 8-expert MoE in every 2nd layer. For TC, 2 experts are activated per token, while for EC the capacity factor is 2. Thus, both models use the same number of active parameters. More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-EC-vs-TC--Vmlldzo4MzkzMDM3

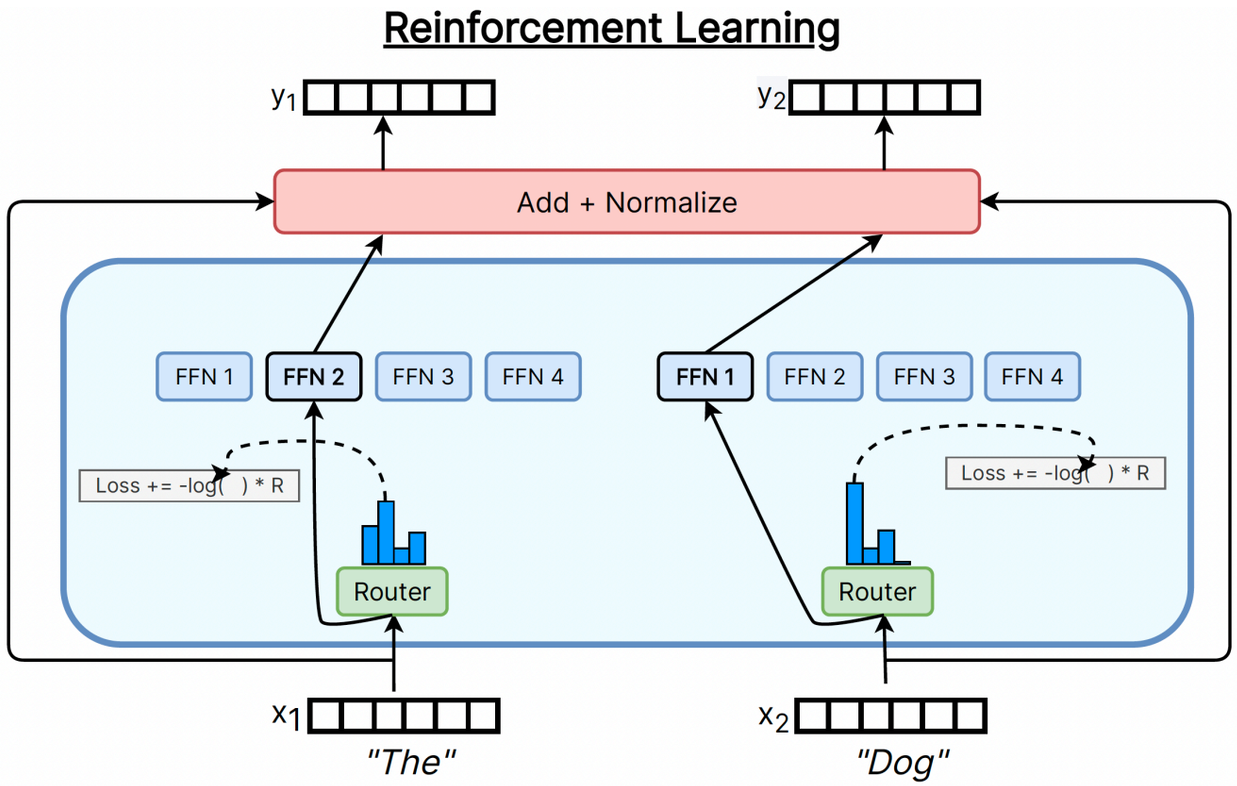
Alternatives: even simpler?

- Hashing [Roller et al 2021]
  - A simple **no-learning** baseline: can perform surprisingly well, but not learning is suboptimal



Alternatives: more complicated?

- Example: reinforcement learning
  - Learn expert selection via RL
- Pros
  - A principled solution for non-differentiability
- Cons
  - Expensive to train, can lead to even more instability
  - Empirically hasn't shown much benefit over simpler methods (e.g. [Clark et al 2022]



Other methods:
linear assignment,
optimal transport…

This lecture

- What are MoEs
- Expert design
- Routing functions
- **Loss functions**
- Systems
- Example

How do we train MoEs?

- Considerations:
  - We want a good model (PPL, task metrics)
  - We want to utilize hardware optimally
    - Non-differentiable routing decision
- RL? [Bengio 2016, Clark 2022]
- Add noise to explore? [Shazeer 2017]
- **Heuristic balancing losses**
  [Fedus 2022 and current methods]

Expert balancing loss

[Zoph et al 2022]

- Intuition: penalize expert if it receives too many tokens and/or gets too much probability mass

$$L = \alpha \sum_{i=1}^{N} f_i P_i$$

f_i : proportion of tokens in sequence routed to expert i

P_i: proportion of routing probability assigned to expert i

Heuristic, but surprisingly effective and widely used

The balancing loss is important

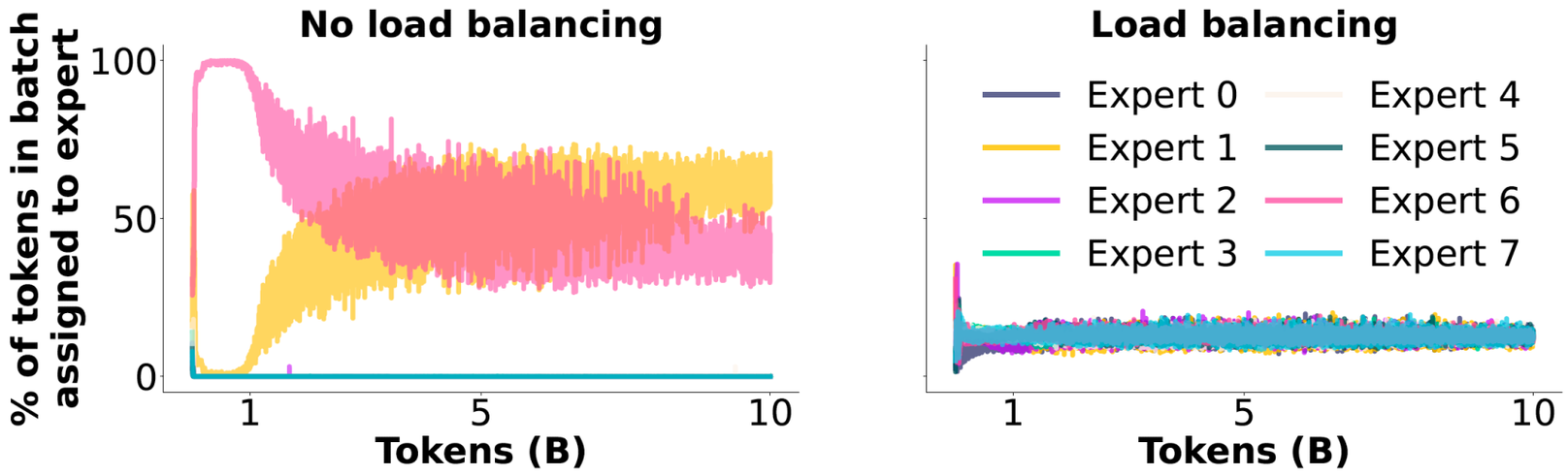


Figure 10: **Expert assignment during training when using or not using a load balancing loss for the first MoE layer.** More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-LBL-vs-No-LBL--Vmlldzo4OTkyNDg4

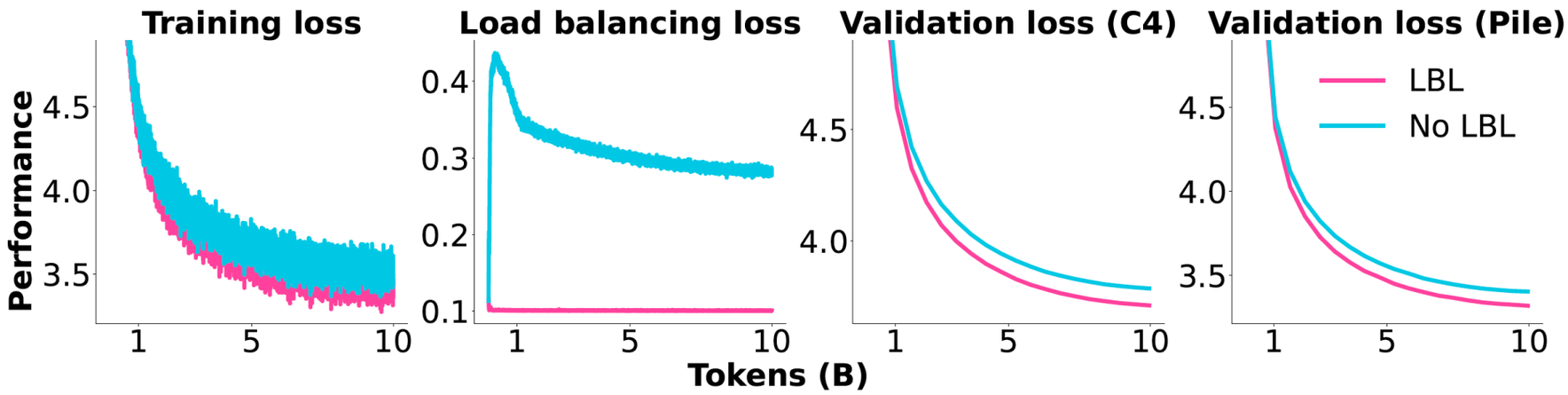


Figure 9: **Impact of applying a load balancing loss (LBL).** The training loss plot excludes the load balancing loss for both models. More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-LBL-vs-No-LBL--Vmlldzo4OTkyNDg4

Variations

- Per-device balancing
  - Put a group of experts on each device
  - Apply loss to groups rather than individual experts

- In practice, used in addition to expert balancing loss

Stability: mixed precision

- Typically float32 is used for the expert routing computations
- But large logits (e.g., strongly preferring an expert) are still an issue…

Stability: z-loss

- Explicitly penalize large logits through a loss

$$L_Z(x) = \frac{1}{B} \sum_{i=1}^{B} \left(\log \sum_{j=1}^{N} e^{x_{i,j}} \right)^2$$

  - B: number of tokens in batch
  - N: number of experts
  - $x \in \mathbb{R}^{B \times N}$ logits used in router

- Basic idea: make the denominator in the softmax smaller

Stability: z-loss

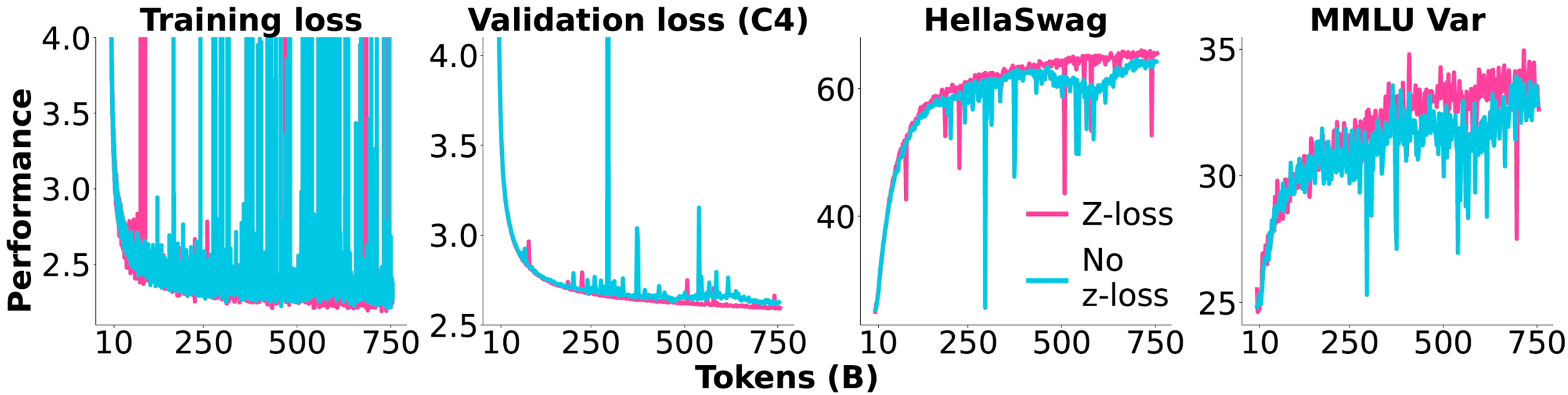


Figure 11: **Router z-loss.** We compare adding router z-loss with a loss weight of 0.001 versus no additional z-loss. More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-Zloss-vs-none--Vmlldzo4NDM4NjUz

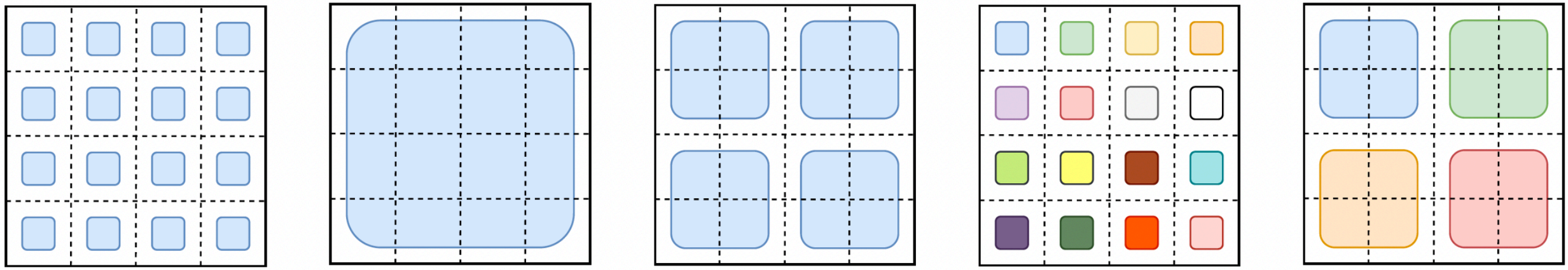
This lecture

- What are MoEs
- Routing functions
- Expert design
- Training objectives
- **Systems**
- Example

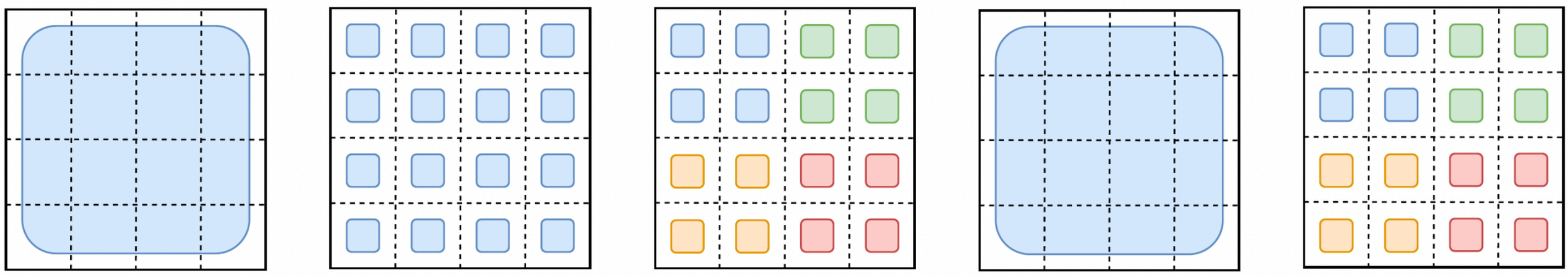
Expert parallelism

- Basic idea: experts can reside on different accelerators. The input data is dynamically dispatched to and fetched from them

How the *model weights* are split over cores



How the *data* is split over cores



[Fedus et al 2022]

Other topics : sparse upcycling

- Turn a dense model into a MoE model
  - Clone the dense MLP multiple times
  - Introduce router parameters
  - Continue pre-training
- Useful if at a fixed number of FLOPs:
  - Dense pretraining + sparse upcycle > sparse pretraining
- Can bias the optimization
  - Intuition: if the pretrained model is already well-trained, it may be difficult for the optimization to find a much different model
- Need to use some of the original model's hyper-parameters

Other topics : sparse upcycling

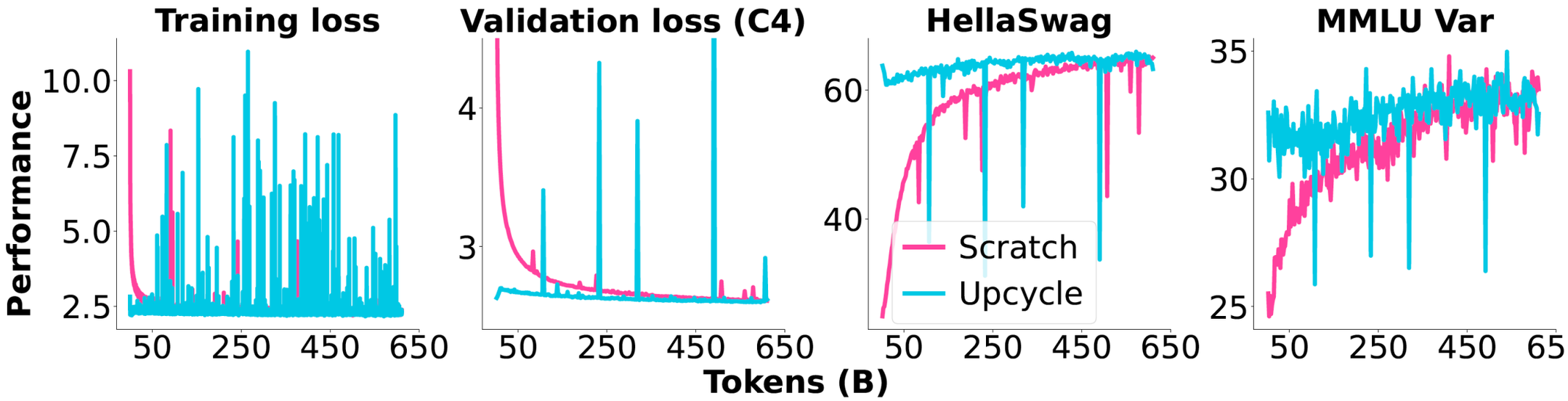
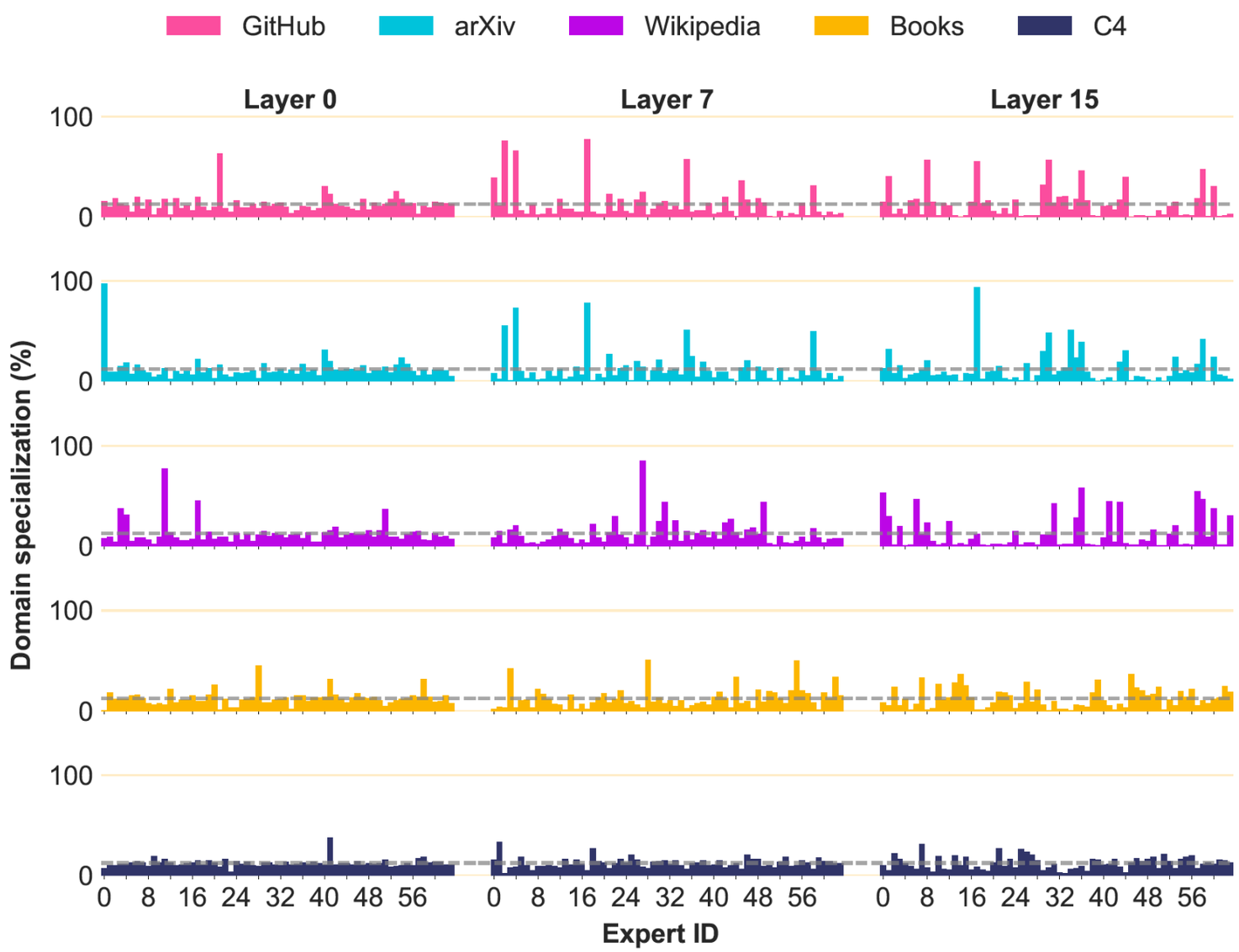


Figure 8: **Sparse upcycling.** We upcycle OLMo-1B (0724) at 2T tokens into an MoE with 8 total experts of which 2 are activated and train it for an additional 610 billion tokens. We compare it to a model trained from scratch for 610 billion tokens. Except for this difference, both models use the same config, which includes some suboptimal settings that contribute to the instability, such as no QK-Norm (§4.2.5) and no truncated normal init (§4.2.2). More results, logs, and configurations: https://wandb.ai/ai2-llm/olmoe/reports/Plot-Scratch-vs-Upcycle--Vmlldzo4NDIyOTc4

- Mixed results: e.g. in OLMOe, sparse pretraining alone catches up quickly (25% of the dense model's budget [AI2 2025])

Other phenomena: domain specialization

- Some experts in some layers may activate frequently in particular domains



This lecture

- What are MoEs
- Routing functions
- Expert design
- Training objectives
- Systems
- **Case studies**

DeepSeek-MoE

- Architecture
  - 16B total parameters, 2.8B active
  - 66 experts with 8 activated
    - 512 dimension
    - 2 shared experts
  - All FFNs use MoE layers
- Routing
  - Per-token
- Pipeline parallelism, no expert parallelism
  - => no token dropping
- Losses
  - Expert balance loss, weight 0.001
  - Device-level balance loss
- Efficiency
  - With 40% of pre-training compute, reaches performance of dense DeepSeek 7B

OLMoE

- Architecture
  - 6.9B total parameters, 1.3B active
  - 64 small experts with 8 activated
    - “Small expert”: 1,024 FFN dimension
    - No shared expert
- Routing
  - Per-token
  - Dropless
- Losses
  - Expert balance loss, weight 0.01
  - Router z-loss, weight 0.001
- Efficiency
  - Pretraining: 23,600 tok/sec/GPU for MoE, 37,500 for dense
  - Reaches performance of dense model 2x faster

Others

- DeepSeek v3
- Mixtral
- Qwen3
- gpt-oss
- …

Thank you