



# 10-423/10-623 Generative AI

Machine Learning Department  
School of Computer Science  
Carnegie Mellon University

## Mixture of Experts

Matt Gormley & Pat Virtue

Lecture 16

Mar. 17, 2025

# Reminders

- **Homework 4: Multimodal Foundation Models**
  - **Out: Thu, Mar 13**
  - **Due: Mon, Mar 24 at 11:59pm**

# MoE Timeline

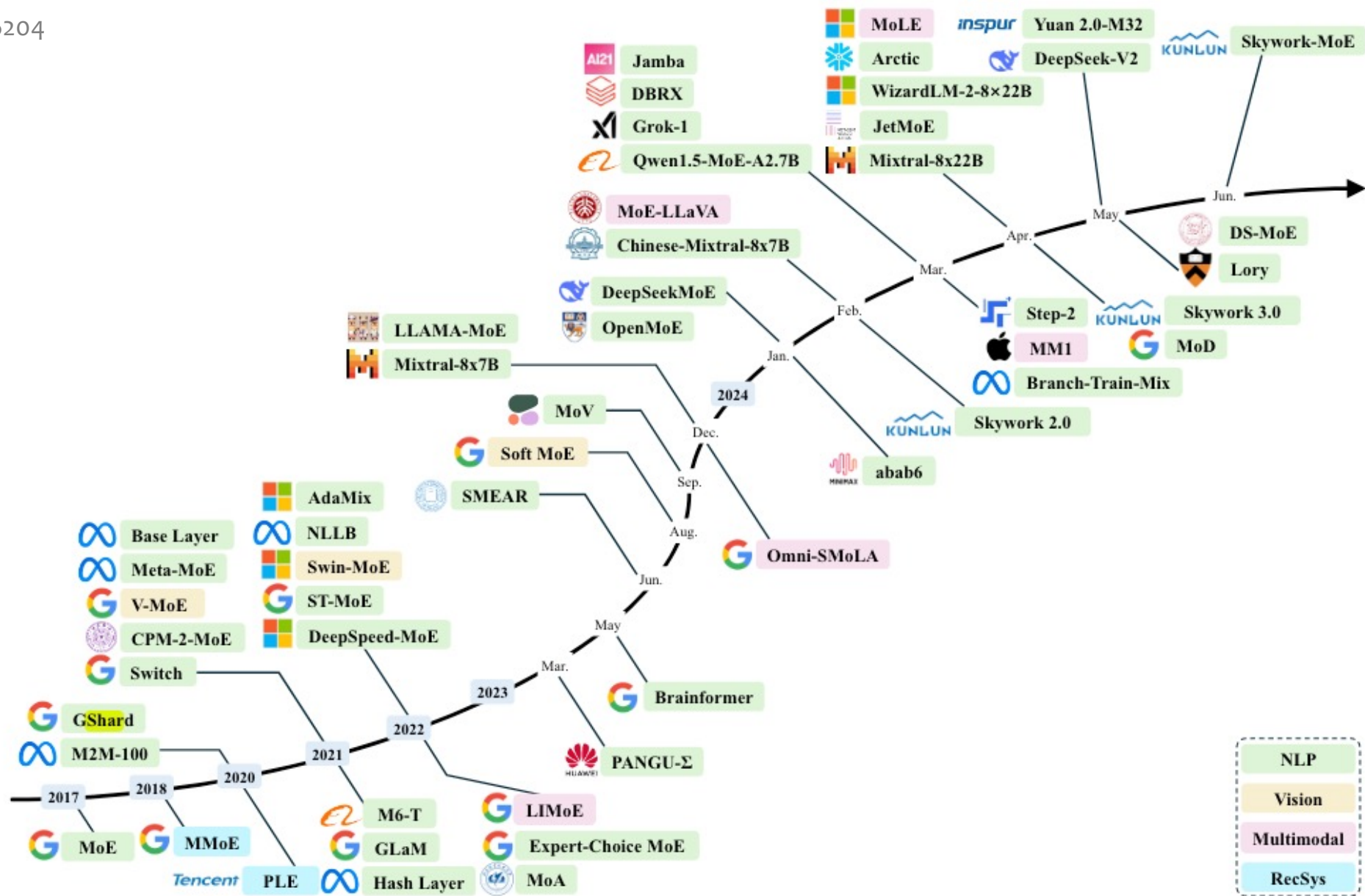
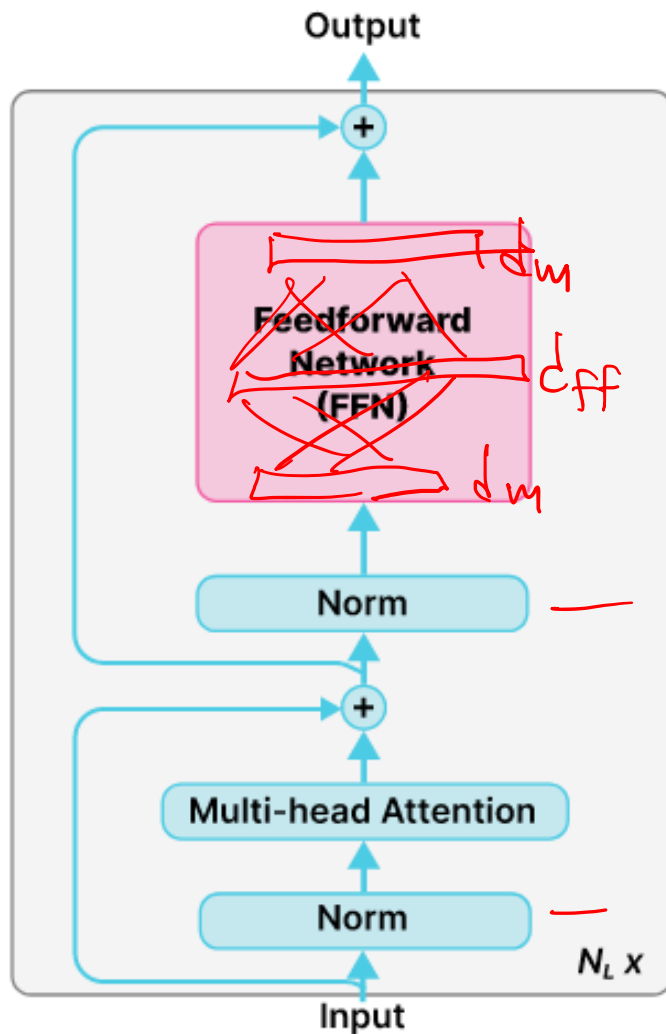


Fig. 1. A chronological overview of several representative mixture-of-experts (MoE) models in recent years. The timeline is primarily structured according to the release dates of the models. MoE models located above the arrow are open-source, while those below the arrow are proprietary and closed-source. MoE models from various domains are marked with distinct colors: Natural Language Processing (NLP) in green, Computer Vision in yellow, Multimodal in pink, and Recommender Systems (RecSys) in cyan.

# The Linear Layer in LLMs

## Dense LMs (OLMo, Llama...)



It is common for more than half of the parameters in a Transformer LLM to reside within the feed-forward neural network layers

	GPT-3
vocab	50,257
d_model	12288
n_heads	96
n_layers	96
d_ff	49152
LayerNorm (B)	0.00
Embedding (B)	0.62
Attention (B)	57.99
Feed-forward (B)	115.97
Total (B)	<b>174.57</b>

$$d_{ff} = 4 \times d_{model}$$

# The Linear Layer in LLMs

**How do you calculate the number of parameters for each layer?**

LayerNorm:

$$\left( \left( (d_m) - 2 \right) \cdot 2 \right) n_{layers}$$

→ Embedding:

$$vocab \times d_{model}$$

Attention:

$$4 \left( (d_m)^2 + d_m \right) n_{layers}$$

Feed-forward:

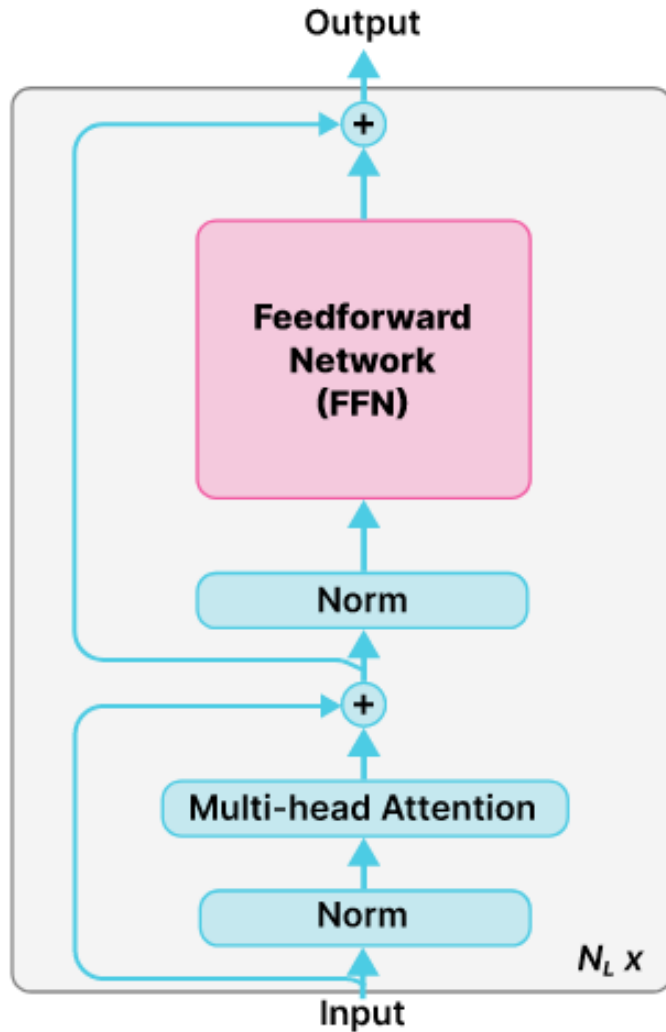
$$\left( 2 \times d_m \times d_{ff} + d_m + d_{ff} \right) n_{layers}$$

It is common for more than half of the parameters in a Transformer LLM to reside within the feed-forward neural network layers

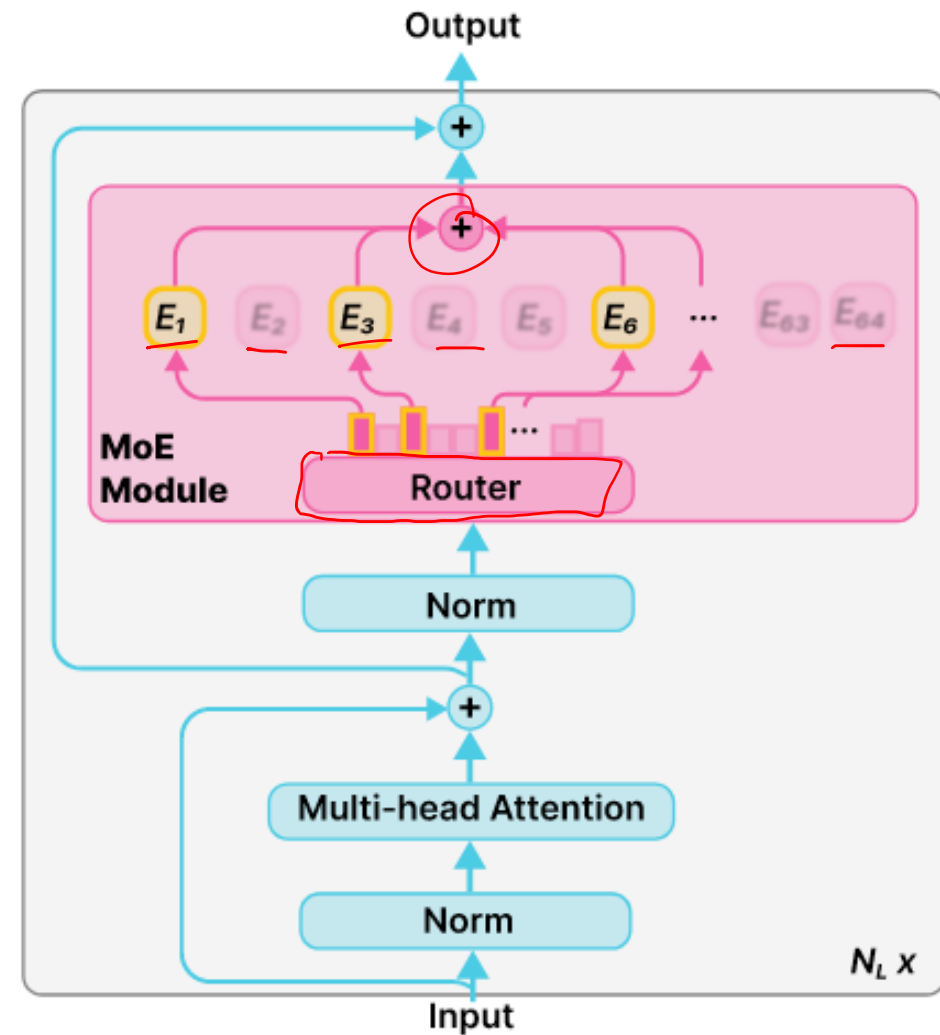
	GPT-3
vocab	50,257
d_model	12288
n_heads	96
n_layers	96
d_ff	49152
LayerNorm (B)	0.00
Embedding (B)	0.62
Attention (B)	57.99
Feed-forward (B)	115.97
Total (B)	174.57

# Sparsely-Gated Mixture of Experts

Dense LMs (OLMo, Llama...)

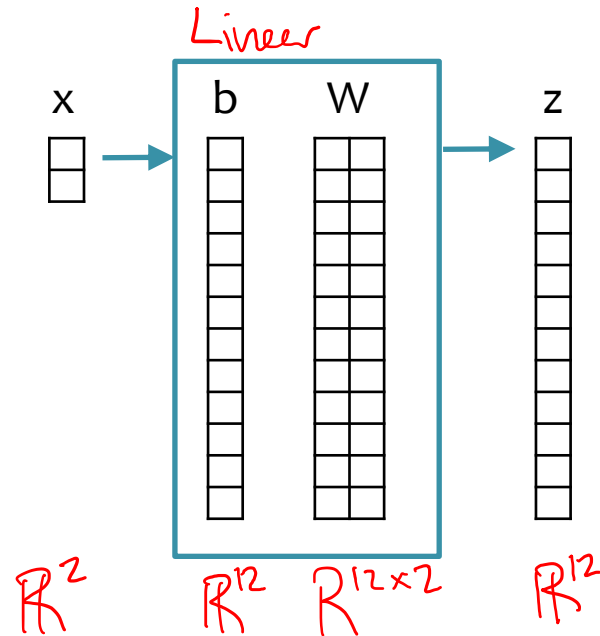


OLMoE



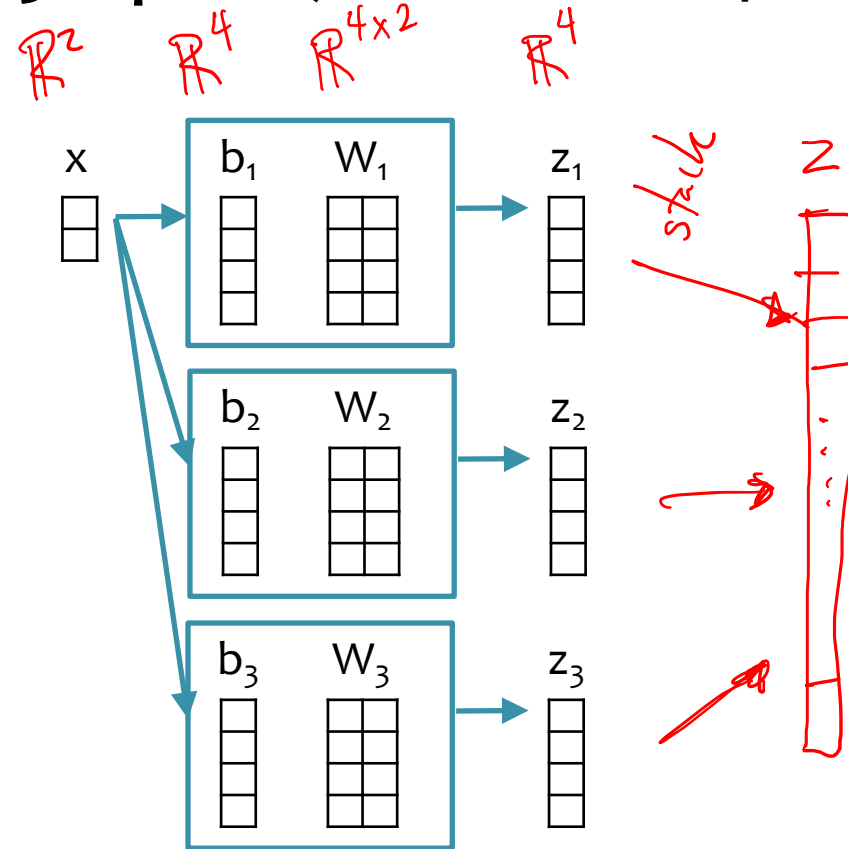
# Breaking a Linear Layer into Experts

## Linear Layer:



$$z = Wx + b$$

## 3 Experts: (with same # of total parameters)

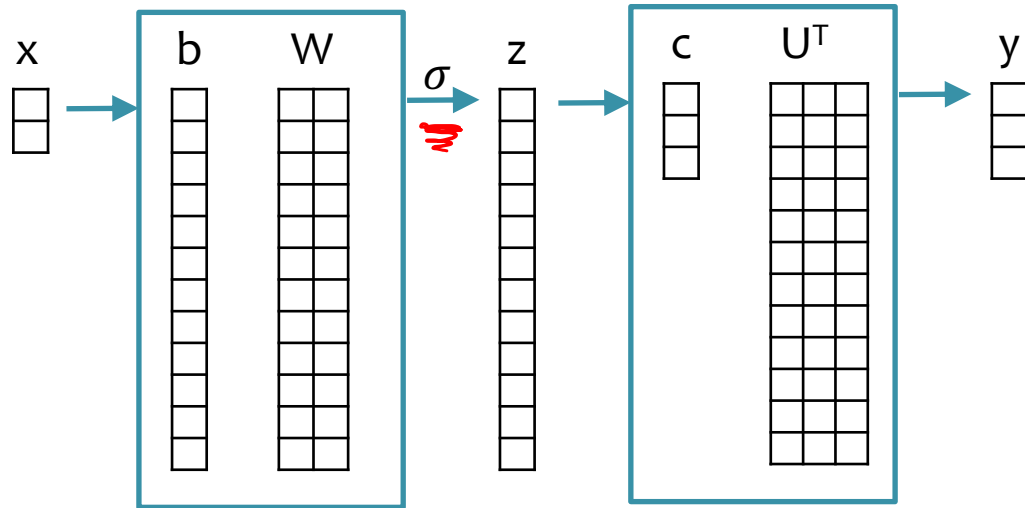


$$z_i = W_i x + b_i$$

Note: In MoE models, each expert is not a *linear* layer, but rather a *feed-forward* layer (e.g., neural network with one hidden layer).

# Breaking a Feed-forward Layer into Experts

Feed-forward Layer:

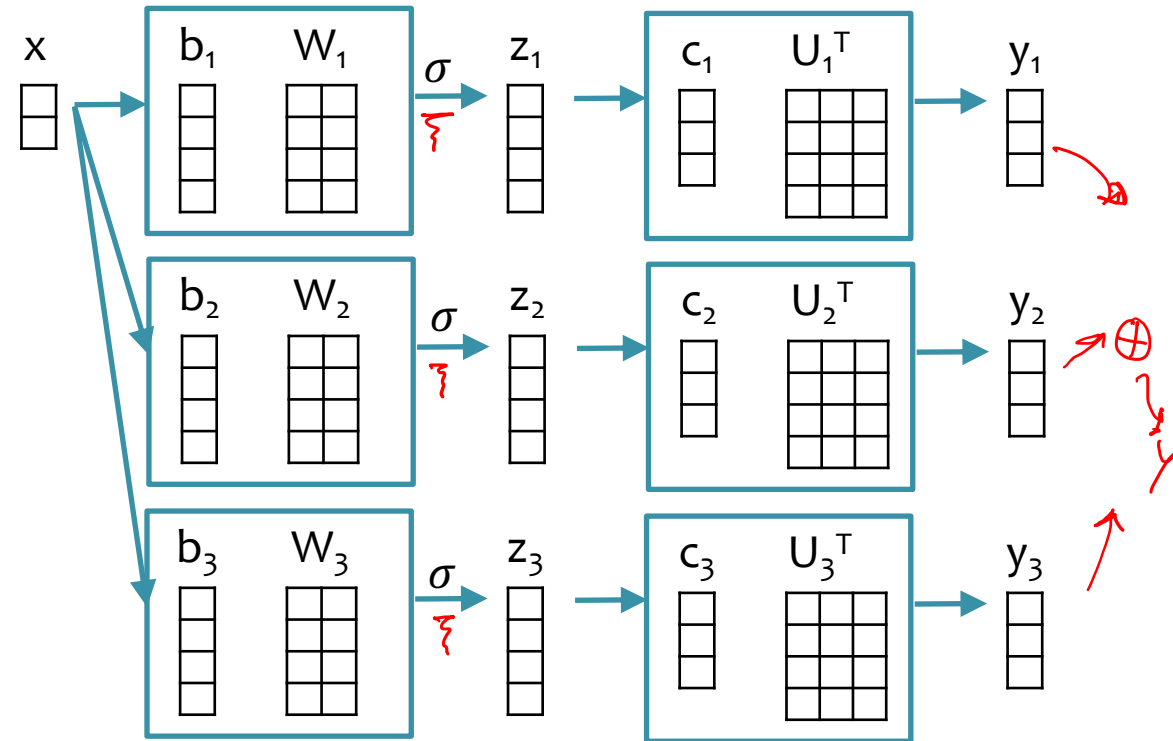


$$z = \sigma(Wx + b)$$

$$y = Uz + c$$

$$y = U \sigma(Wx + b) + c$$

3 Experts: (with same # of total parameters)



$$y_i = U_i \sigma(W_i x + b_i) + c_i$$

$$y_i = U_i z_i + c_i$$

The two computations above are equivalent if  $W = \text{stack}(W_1, W_2, W_3)$  and  $b = \text{stack}(b_1, b_2, b_3)$  and  $U^T = \text{stack}(U_1^T, U_2^T, U_3^T)$  and  $y = \text{stack}(y_1, y_2, y_3)$  and  $c = c_1 + c_2 + c_3$  ?

$$y = y_1 + y_2 + y_3 ?$$



# Dense Mixture of Experts

A dense mixture of experts gives every expert a (non-zero) voice in the output

$$\mathbf{y} = \sum_{i=1}^{N_e} \underbrace{G(\mathbf{x})_i}_{\substack{G(x)_i \in \mathbb{R}}} E_i(\mathbf{x}) \quad E_i(x) \in \mathbb{R}^M$$

## Dense MoE

- Dense Softmax

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

- Each expert is just a feed-forward network

$$E_i(\mathbf{x}) = \text{FFN}_{\text{ReLU}}(\mathbf{x}) = \text{ReLU}(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$$

# Sparsely-Gated Mixture of Experts

Sparsely-Gated MoEs were originally introduced for RNNs, but are generally applicable and now popular for Transformers

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

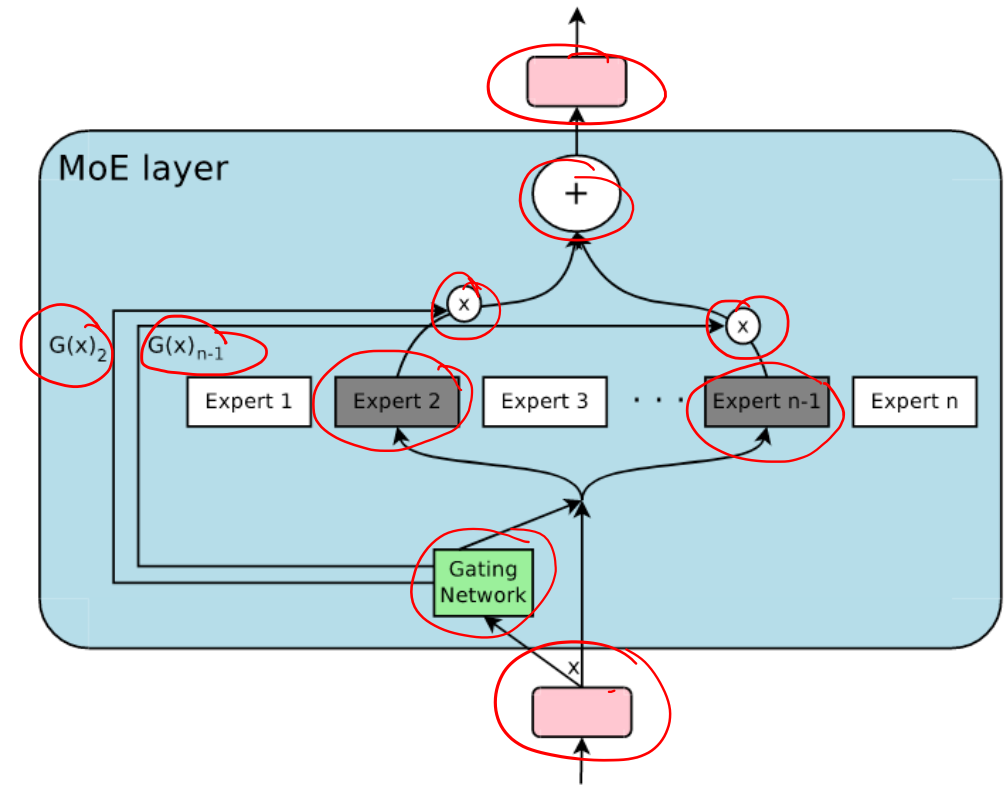
## Sparse MoE

- Softmax over Top-K Gating

$$G(\mathbf{x}) = \text{softmax}(\text{topk}(\mathbf{x} \cdot \mathbf{W}_g + \mathbf{b}_g, k))$$

- Each expert is just a feed-forward network

$$E_i(\mathbf{x}) = \text{FFN}_{\text{ReLU}}(\mathbf{x}) = \text{ReLU}(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$$



# Sparsely-Gated Mixture of Experts

## Definition of top-k:

$$\vec{g} = \text{topk}(\vec{v}, k) \quad \begin{array}{l} \vec{g} \in \mathbb{R}^M \\ \vec{v} \in \mathbb{R}^M \end{array} \quad k \in \mathbb{Z}_+$$

$$g_i = \begin{cases} v_i & \text{if } i \text{ is among the} \\ & \text{top-}k \text{ largest values in } \vec{v} \\ -\infty & \text{otherwise} \end{cases}$$

## Example: Computing the gate

$$\vec{v} = \vec{x}W_g + b_g = [-7, 3, 8, 1]$$

$$\vec{g} = \text{topk}(\vec{v}, 2) = [-\infty, 3, 8, -\infty]$$

$$\vec{G} = \text{softmax}(\vec{g}) = \left[ 0, \frac{\exp(3)}{\exp(3)+\exp(8)}, \frac{\exp(8)}{\exp(3)+\exp(8)}, 0 \right]$$

## Example: Computing the MoE

$$\vec{y} = \begin{array}{l} \cancel{G_1} E_1(\vec{x}) \\ + G_2 E_2(\vec{x}) \\ + G_3 E_3(\vec{x}) \\ + \cancel{G_4} E_4(\vec{x}) \end{array} = G_2 E_2(\vec{x}) + G_3 E_3(\vec{x})$$

# Sparsely-Gated Mixture of Experts

Sparsely-Gated MoEs were originally introduced for RNNs, but are generally applicable and now popular for Transformers

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

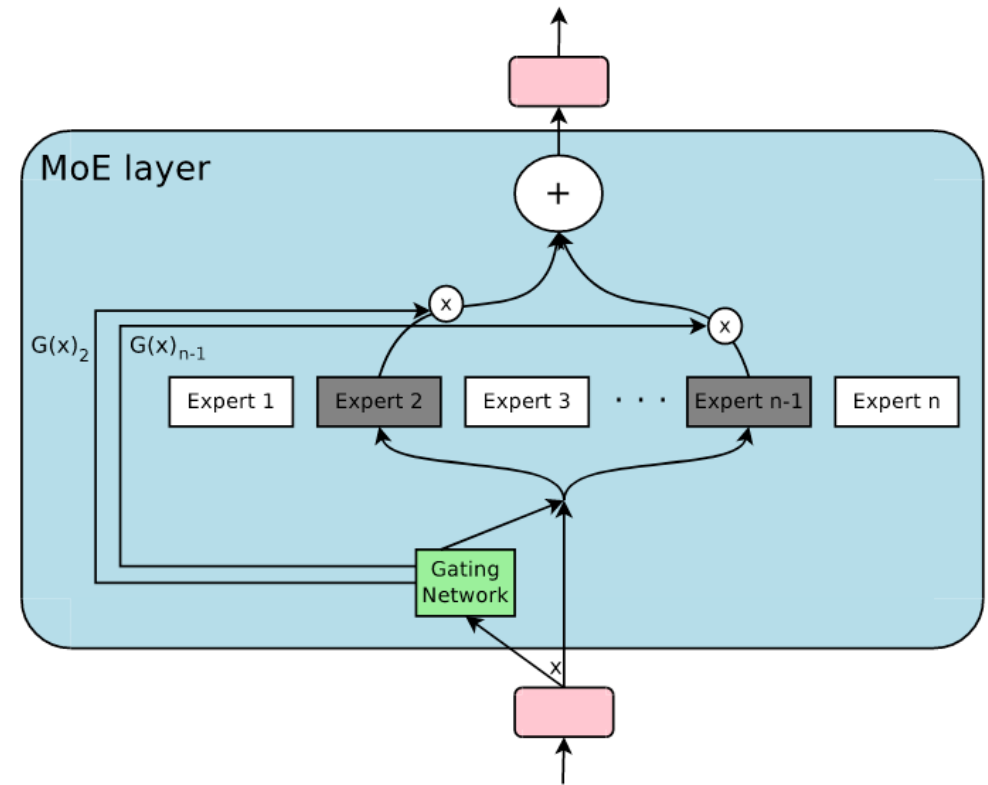
## Sparse MoE

- Softmax over Top-K Gating

$$G(\mathbf{x}) = \text{softmax}(\text{topk}(\mathbf{x} \cdot \mathbf{W}_g + \mathbf{b}_g, k))$$

- Each expert is just a feed-forward network

$$E_i(\mathbf{x}) = \text{FFN}_{\text{ReLU}}(\mathbf{x}) = \text{ReLU}(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$$



# Sparsely-Gated Mixture of Experts

Sparsely-Gated MoEs were originally introduced for RNNs, but are generally applicable and now popular for Transformers

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

## Sparse MoE

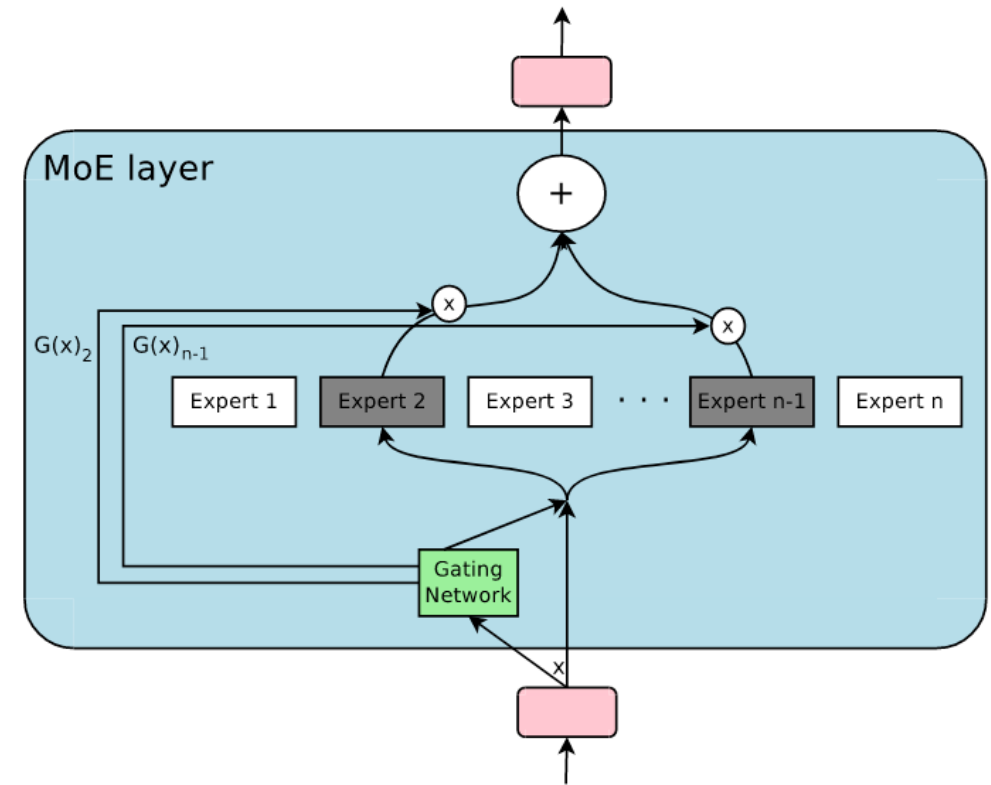
- Softmax over Top-K Gating

$$G(\mathbf{x}) = \text{softmax}(\text{topk}(\mathbf{x} \cdot \mathbf{W}_g + \mathbf{b}_g + \mathbf{r}_{\text{noise}}(\mathbf{x}), k))$$

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

- Each expert is just a feed-forward network

$$E_i(\mathbf{x}) = \text{FFN}_{\text{ReLU}}(\mathbf{x}) = \text{ReLU}(\mathbf{x} \mathbf{W}_1 + \mathbf{b}_1) \mathbf{W}_2 + \mathbf{b}_2$$



# Sparsely-Gated Mixture of Experts

Sparsely-Gated MoEs were originally introduced for RNNs, but are generally applicable and now popular for Transformers

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

## Mixtral

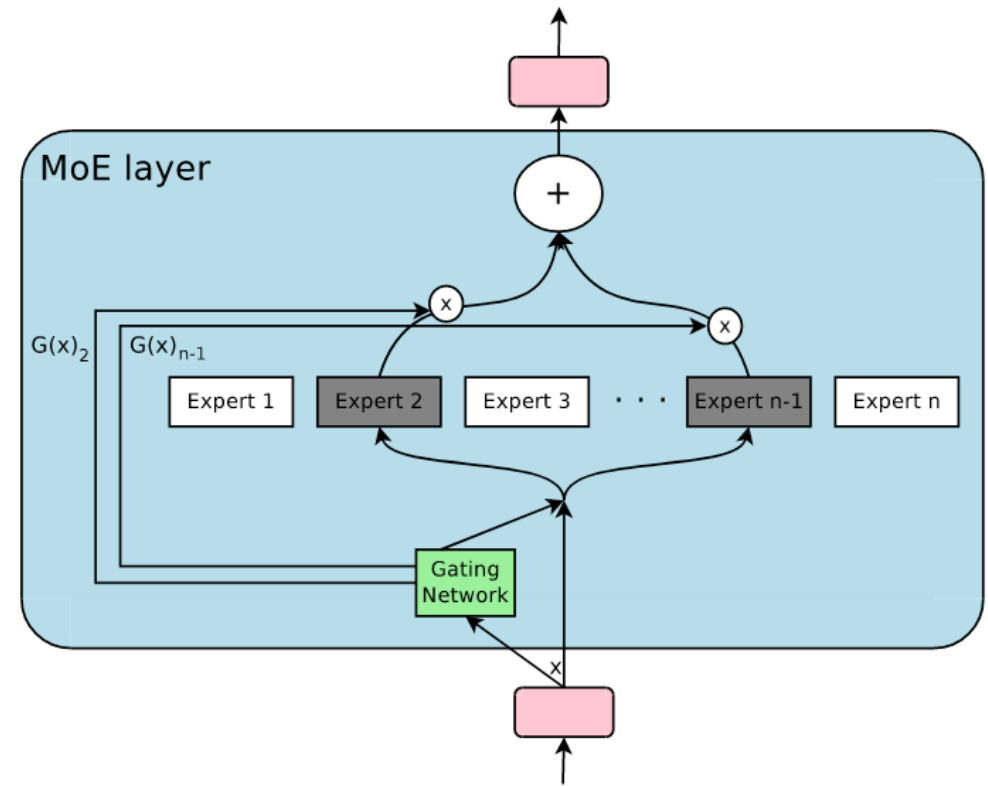
- Softmax over Top-K Gating

$$G(\mathbf{x}) = \text{softmax}(\text{topk}(\mathbf{x} \cdot \mathbf{W}_g + \mathbf{b}_g + \mathbf{r}_{\text{noise}}(\mathbf{x}), k))$$

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

- Each expert is just a feed-forward network

$$E_i(\mathbf{x}) = \text{SwiGLU}(\mathbf{x}) = \underbrace{\text{Swish}(\mathbf{x} \mathbf{W} + \mathbf{b})}_{\text{Hidden}} \odot \underbrace{(\mathbf{x} \mathbf{V} + \mathbf{c})}_{\text{Linear 2}}$$



# Sparsely-Gated Mixture of Experts

Sparsely-Gated MoEs were originally introduced in [1] and are generally applicable and now popular for [2].

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

## Mixtral

- Softmax over Top-K Gating

$$G(\mathbf{x}) = \text{softmax}(\text{topk}(\mathbf{x} \cdot \mathbf{W}_g + \mathbf{b}_g + \mathbf{r}_{\text{noise}}(\mathbf{x}), k))$$

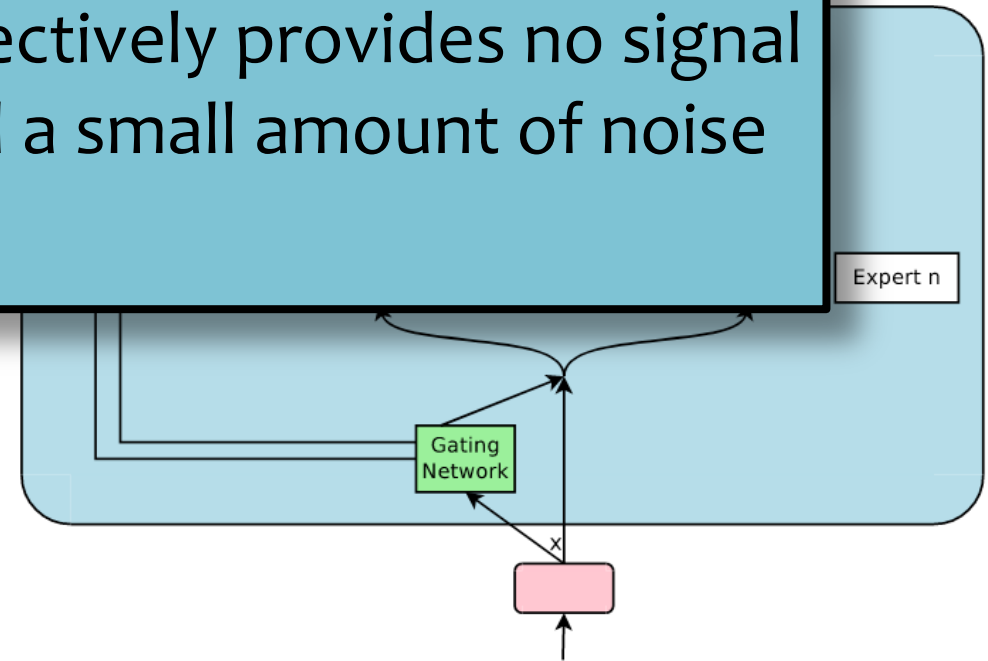
$$\mathbf{r}_{\text{noise}}(\mathbf{x}) = N(\mathbf{0}, \mathbf{I}) \cdot \text{sigma}(\mathbf{x} \mathbf{W}_{\text{noise}} + \mathbf{b}_{\text{noise}})$$

- Each expert is just a feed-forward network

$$E_i(\mathbf{x}) = \text{SwiGLU}(\mathbf{x}) = \text{Swish}(\mathbf{x} \mathbf{W} + \mathbf{b}) \odot (\mathbf{x} \mathbf{V} + \mathbf{c})$$

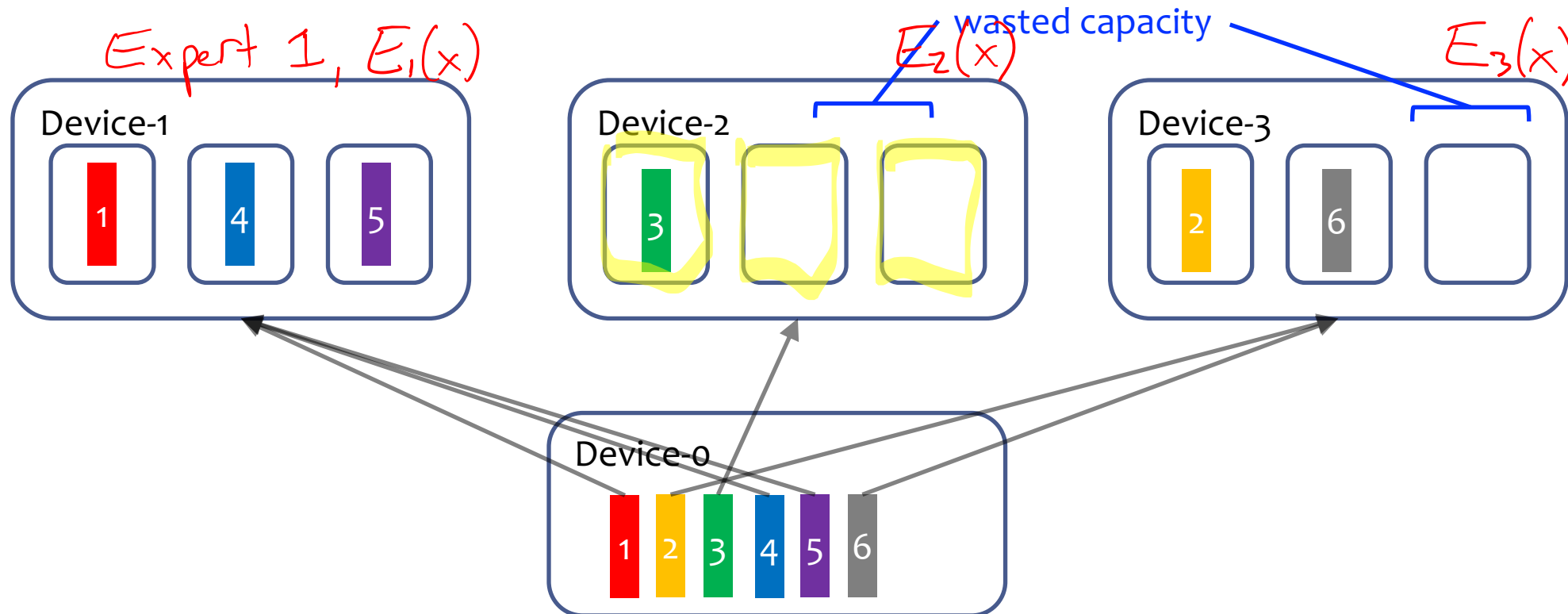
## Initialization:

- We initialize  $\mathbf{W}_g$  and  $\mathbf{W}_{\text{noise}}$  to all zeros
- Effectively provides no signal and a small amount of noise



# Expert Parallelism

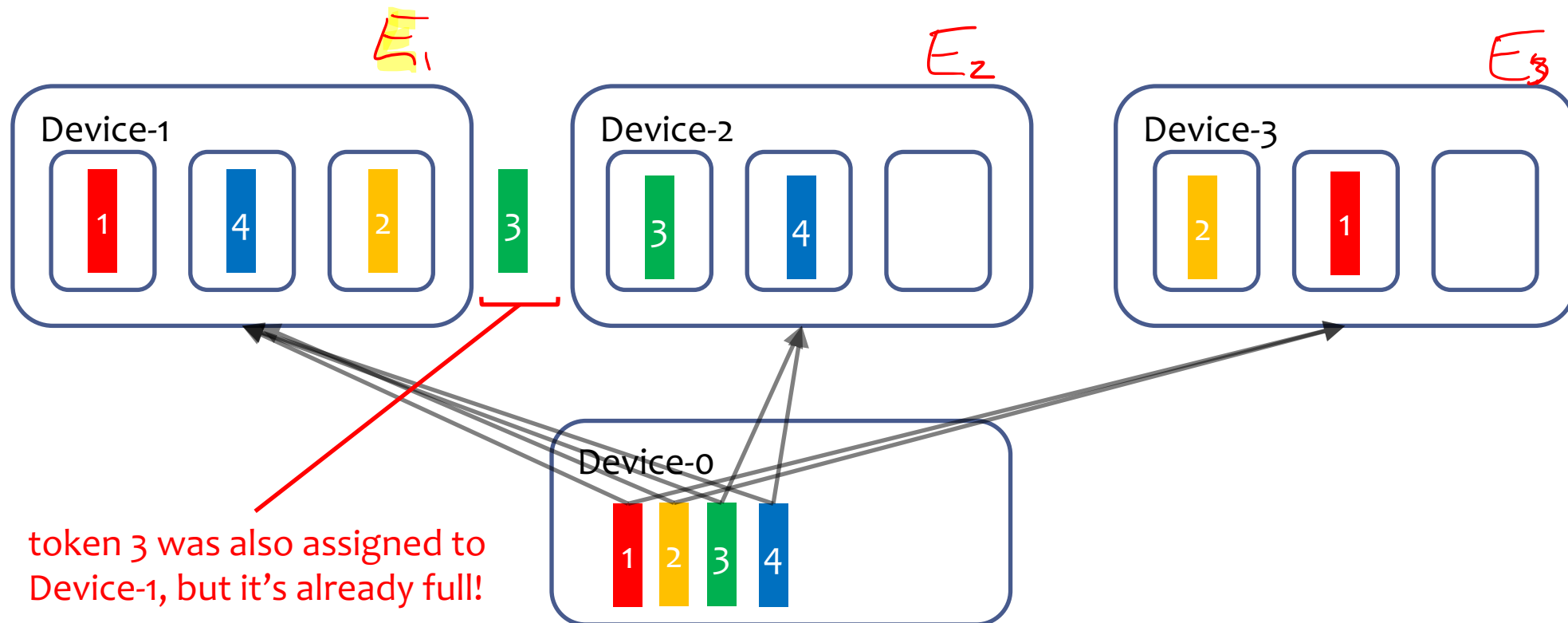
- Each expert can be allocated to a different device (GPU)
- Each token is routed to K experts
- Load balancing issues if too many tokens are routed to the same expert
- **Example:** 3 devices, capacity of 3 tokens/device, 6 tokens, K=1 experts per token





# Expert Parallelism

- Each expert can be allocated to a different device (GPU)
- Each token is routed to K experts
- Load balancing issues if too many tokens are routed to the same expert
- **Example:** 3 devices, capacity of 3 tokens/device, 4 tokens, K=2 experts per token



# Balancing Expert Utilization in an MoE

- **Problem:** left unchecked, the expert gate tends to concentrate on a small number of experts that are popular early in training

- **Solution:**

- Add two regularizers to the loss
- *Load Balance Term:* encourages distributed load over the experts within each batch
- *Router Z-loss:* penalizes large logits to the router to stabilize training

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

$$\mathcal{L}_{LB} = N_e \sum_{i=1}^{N_e} f_i P_i$$

$f_i$  = fraction of tokens routed to expert  $i$   
 $P_i$  = probability to  $E_i$  in current batch

$$\mathcal{L}_{RZ} = \frac{1}{N_b} \sum_{b=1}^{N_b} \left( \log \sum_{d=1}^D \exp(x_d^{(b)}) \right)$$

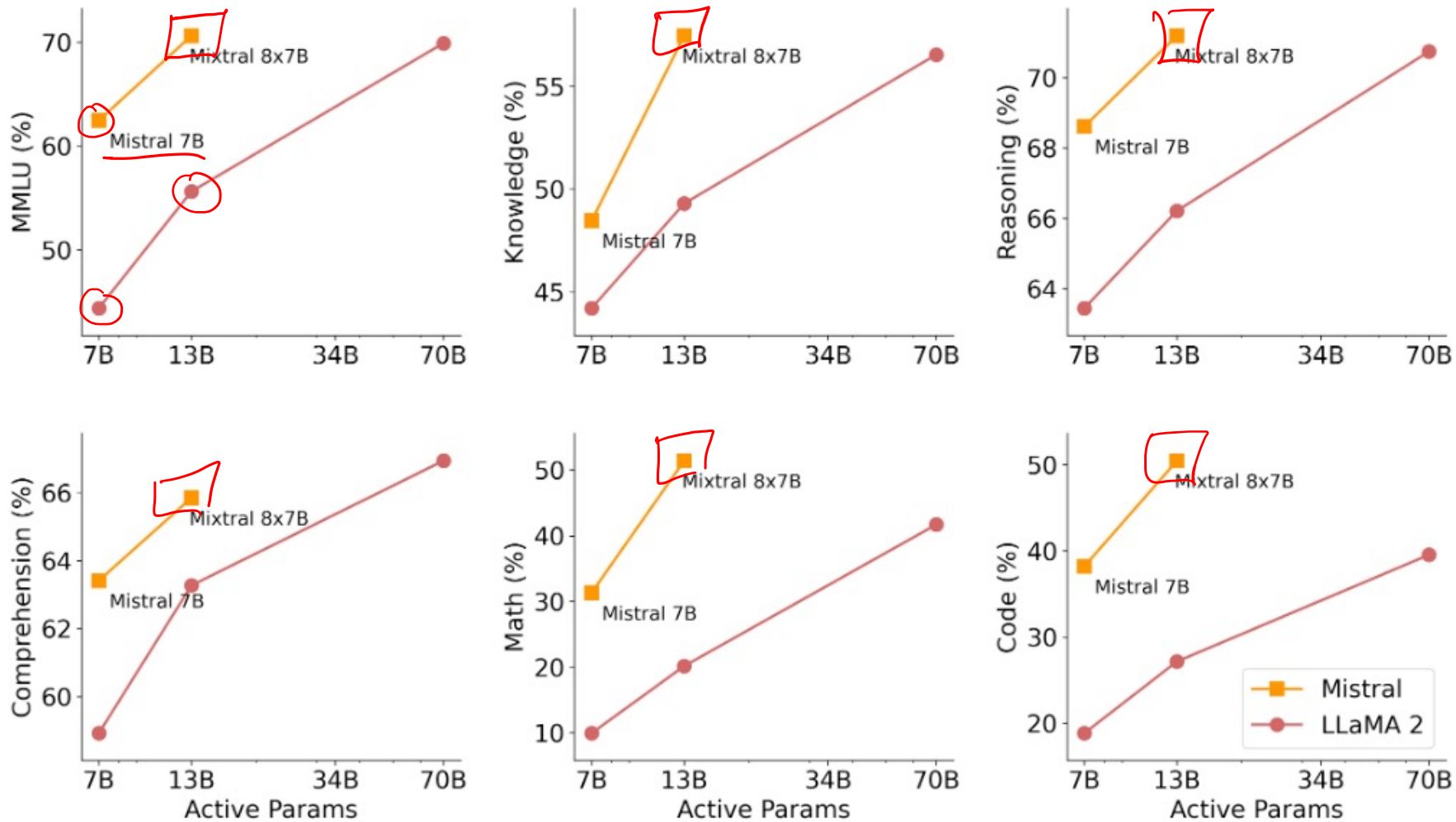
(this is the approach used by O1MoE, but lots of variants exist)

# Active Parameters

- In a transformer with MoE layers, we typically choose the number  $k$  for the top- $k$  to be rather small
  - Mixtral:  $k = 2, N_e = 8$
  - OLMoE:  $k=8, N_e = 64$
- The number of **active parameters** is the count of parameters that are selected for computation by the router
- For an MoE (roughly)
  - GPU memory requirement  $\propto$  # of total parameters
  - FLOPS computation requirement  $\propto$  # of active parameters

# Mixtral vs. Llama-2

$k=2$



# OLMoE Hyperparameters

	<b>OLMoE-1B-7B</b>	<b>JetMoE</b>	<b>OpenMoE</b>	<b>OLMo-1B (0724)</b>
Dimension	2,048	2,048	2,048	2,048
Activation	SwiGLU	SwiGLU	SwiGLU	SwiGLU
<b>FFN dimension</b>	<b>1,024</b>	<b>5,632</b>	<b>8,192</b>	<b>8,192</b>
Vocab size	50,304	32,000	256,384	50,304
Attn heads	16	16	24	16
Num layers	16	24	32	16
<b>Layer norm type</b>	<b>RMSNorm</b>	<b>RMSNorm</b>	<b>RMSNorm</b>	<b>non-parametric</b>
Layer norm eps	1.0E-05	1.0E-05	1.0E-06	1.0E-05
<b>QK-Norm</b>	<b>yes</b>	<b>no</b>	<b>no</b>	<b>no</b>
Pos emb.	RoPE	RoPE	RoPE	RoPE
RoPE $\theta$	10,000	10,000	10,000	10,000
Attention variant	full	MoA	full	full
Biases	-	MLP & Attn	-	-
Weight tying	no	yes	no	no
<b>Init dist</b>	<b>trunc normal</b>	<b>?</b>	<b>?</b>	<b>normal</b>
Init std	0.02	0.02	varies	varies
Init trunc	3×std	-	-	-
MoE layers	Every	Every	Every 6th	-
MoE layer type	dMoE	dMoE	ST-MoE	-
# Experts	64	8	32	1
# Activated	8	2	2	1
# Vocab params	103M	66M	525M	103M
# Active params	1.3B	2.2B	2.6B	1.3B
<b># Total params</b>	<b>6.9B</b>	<b>8.5B</b>	<b>8.7B</b>	<b>1.3B</b>

Sequence length	4,096	4,096	2,048	4,096
<b>Batch size (samples)</b>	<b>1,024</b>	1,024	2,048	512
<b>Batch size (tokens)</b>	<b>~4M</b>	~4M	~4M	~2M
warmup steps	2,000	2,500	10,000	2,000
peak LR	4.0E-04	5.0E-04	0.01	4.0E-04
minimum LR	5.0E-04	5.0E-05	-	5.0E-05
optimizer	AdamW	AdamW	Adafactor	AdamW
weight decay	0.1	0.1	0.0	0.1
beta1	0.9	?	0.9	0.9
beta2	0.95	?	-	0.95
<b>AdamW epsilon</b>	<b>1.0E-08</b>	<b>?</b>	<b>-</b>	<b>1.0E-05</b>
LR schedule	cosine	WSD	Inv Sq Root	cosine
gradient clipping	global 1.0	global 1.0	global 1.0	global 1.0
gradient reduce dtype	FP32	?	?	FP32
optimizer state dtype	FP32	?	?	FP32
<b>LBL weight</b>	<b>0.01</b>	<b>0.01</b>	<b>0.01</b>	<b>-</b>
Router z-loss weight	0.001	0.001	0.0001	-
Pretraining tokens	5,033B	1,000B	1,100B	2,000B
Annealing tokens	100B	250B	-	50B
Annealing schedule	linear	-	-	linear
Annealing min LR	0	-	-	0

Table 10: **Pretraining hyperparameters of OLMoE-1B-7B and comparable models trained from scratch.** We highlight rows where **OLMoE-1B-7B** differs from OLMo-1B. Active params include vocab params. “?” = undisclosed settings, FFN = feed-forward network, Attn = Attention, LR = learning rate, WSD = Weight-Stable-Decay [73], LBL = load balancing loss, Inv Sq Root = Inverse Square Root decay [153], trunc = truncation, std = standard deviation, “varies” = stds that are layer or weight-dependent.

# Performance vs. Cost

MoEs provide a nice tradeoff between performance and FLOPS cost

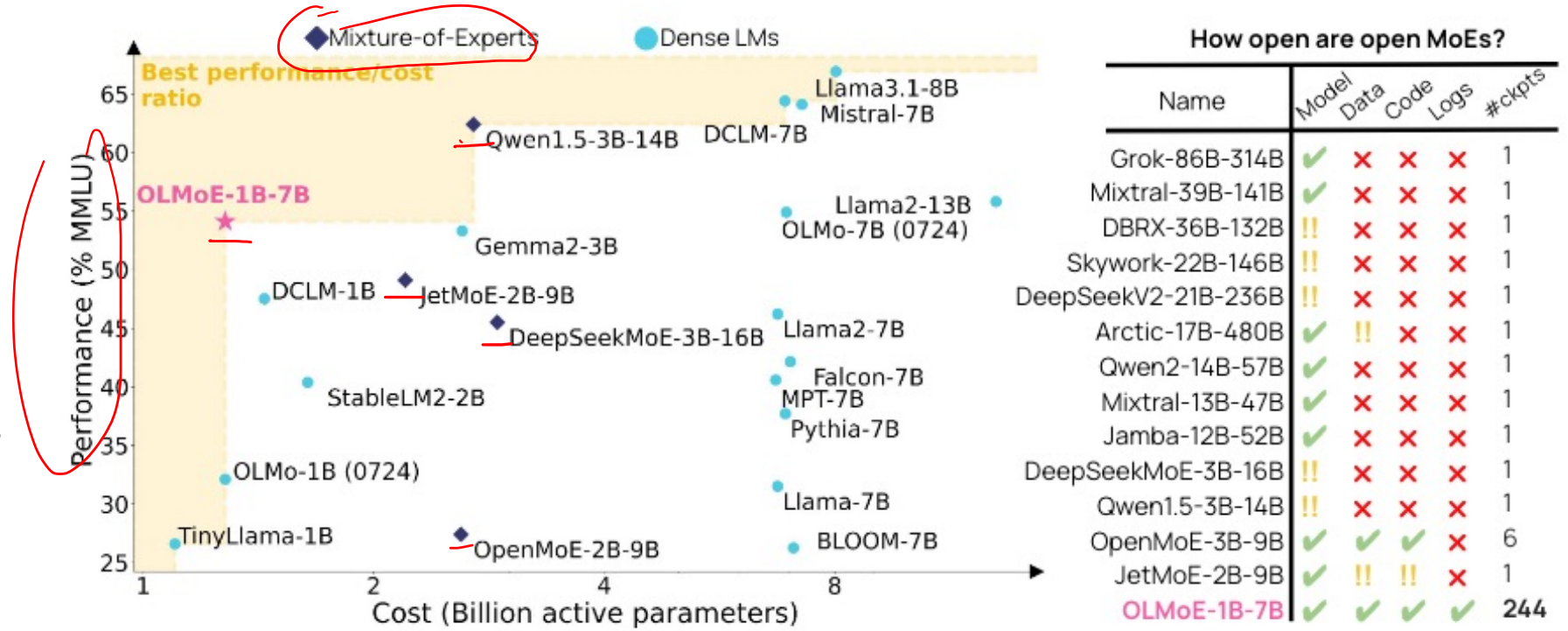


Figure 1: **Performance, cost, and degree of openness of open MoE and dense LMs.** Model names contain rounded parameter counts: model-active-total for MoEs and model-total for dense LMs. #ckpts is the number of intermediate checkpoints available. We highlight MMLU as a summary of overall performance; see §3 for more results. **OLMoE-1B-7B** performs best among models with similar active parameter counts and is the most open MoE.

# Performance vs. Cost

1. Comparing two models with the same active parameters: MoE wins on accuracy
2. Comparing two models with the same total parameters: MoE wins on convergence (measured in Training FLOPS)

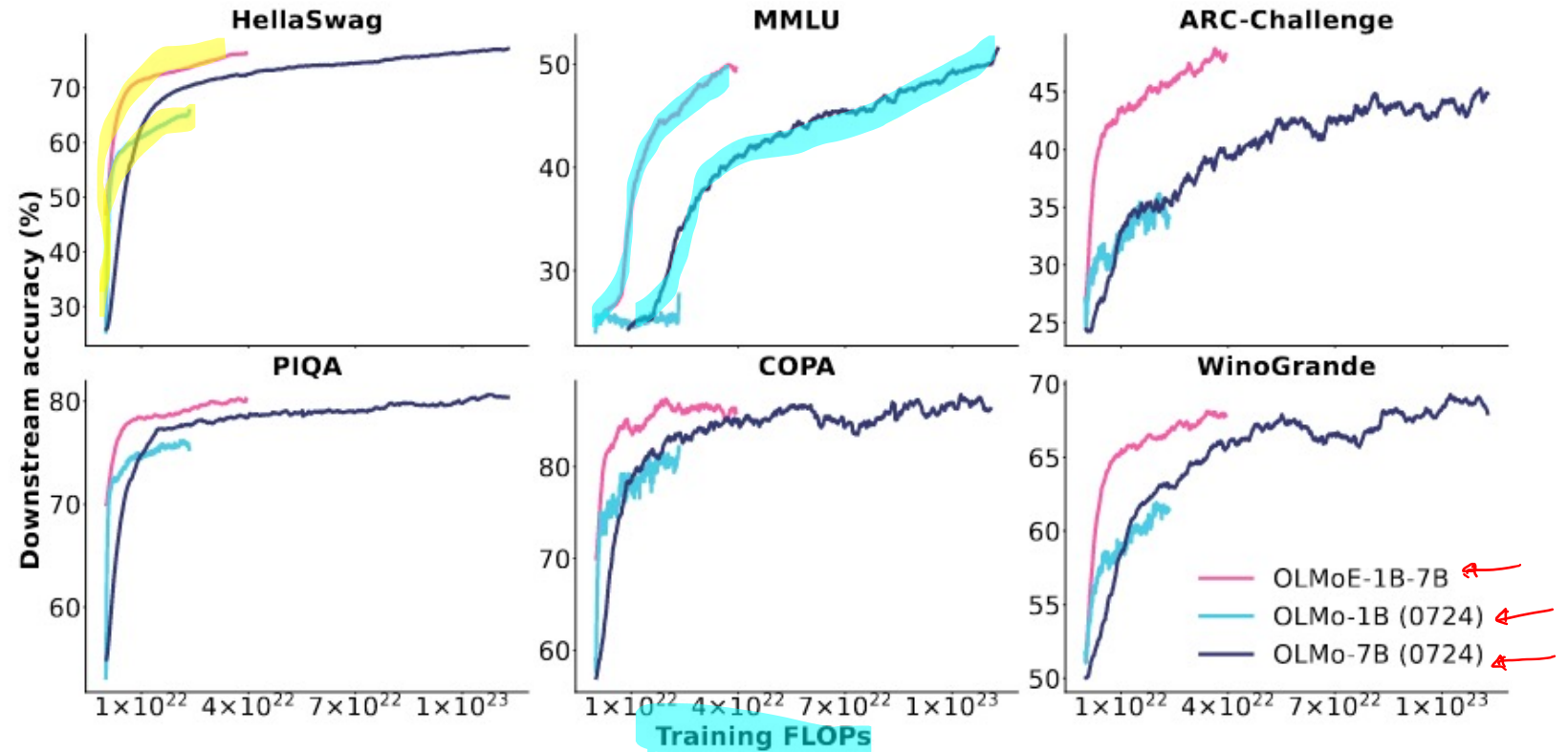
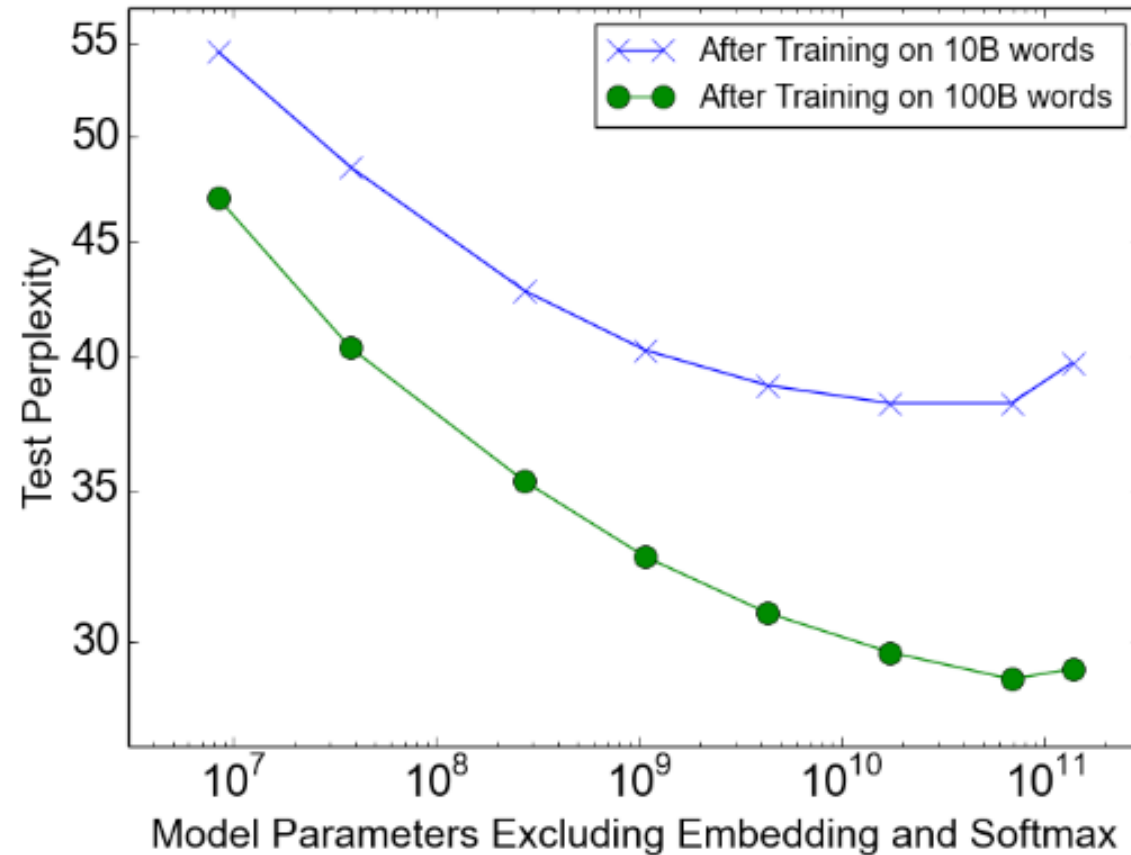


Figure 3: Evaluation of OLMoE-1B-7B and the current best OLMo models during pretraining. OLMoE-1B-7B differs from the OLMo models in its MoE architecture, several training hyperparameters, and its training dataset, see §2. A version of this plot with tokens as the x-axis and markers where annealing starts is in Appendix E. More results, logs, and configurations: <https://wandb.ai/ai2-llm/olmoe/reports/Plot-OLMoE-1B-7B-vs-OLMo-7B-vs-OLMo-1B--Vmlldzo40TcyMjEz>

# How many experts to choose?

Early work with MoEs in LSTM-LMs favored a very large number of experts



32, 256, 1024, 4096, 16384, 65536, and 131072 experts.  
up to 137 billion parameters in the MoE layer.



# How many experts to choose?

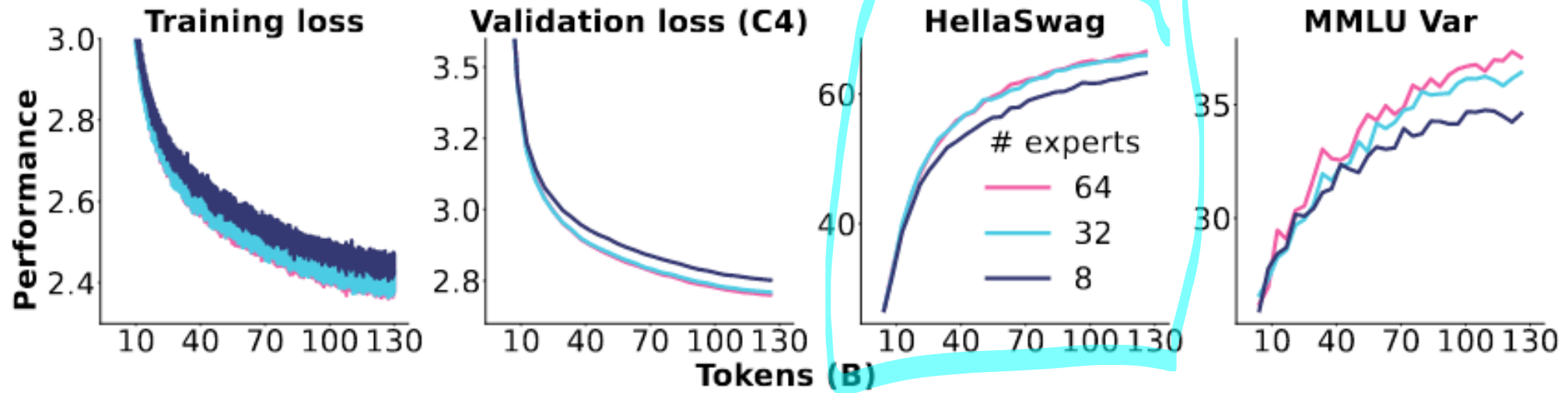


Figure 5: **Expert granularity.** We vary the number of experts in tandem with the FFN dimension to ensure that active and total parameters and thus compute cost remain the same. For example, for 64 experts, the FFN dimension is 1,024 and 8 experts are activated, while for 32 experts it is 2,048 with 4 activated experts. More results, logs, and configurations: <https://wandb.ai/ai2-llm/olmoe/reports/Plot-Granularity--Vmlldzo40TIx0TE4>

More recent work on Transformer LMs has favored a comparatively small number of experts

# Scaling Laws for Routed LMs

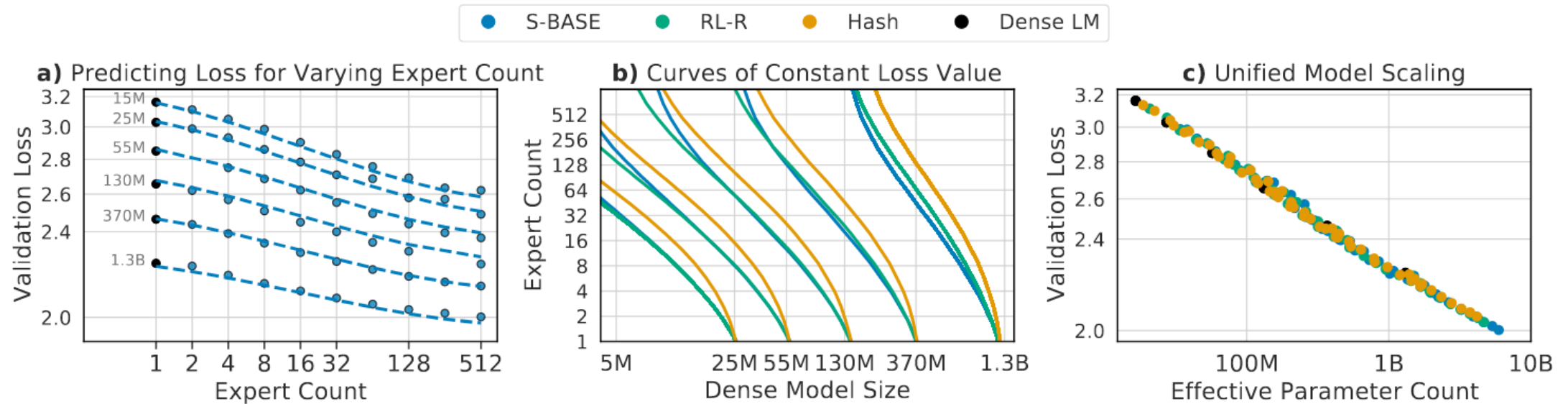


Figure 1. (a) The performance achieved by Routing Networks when varying the number of experts for a fixed dense model size is described by a bilinear function (Eq. 1), (b) whose level curves indicate how to trade model size with expert count to maintain a fixed performance, (c) and which can be manipulated to align dense and routed model performance under a shared power law.