# 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**
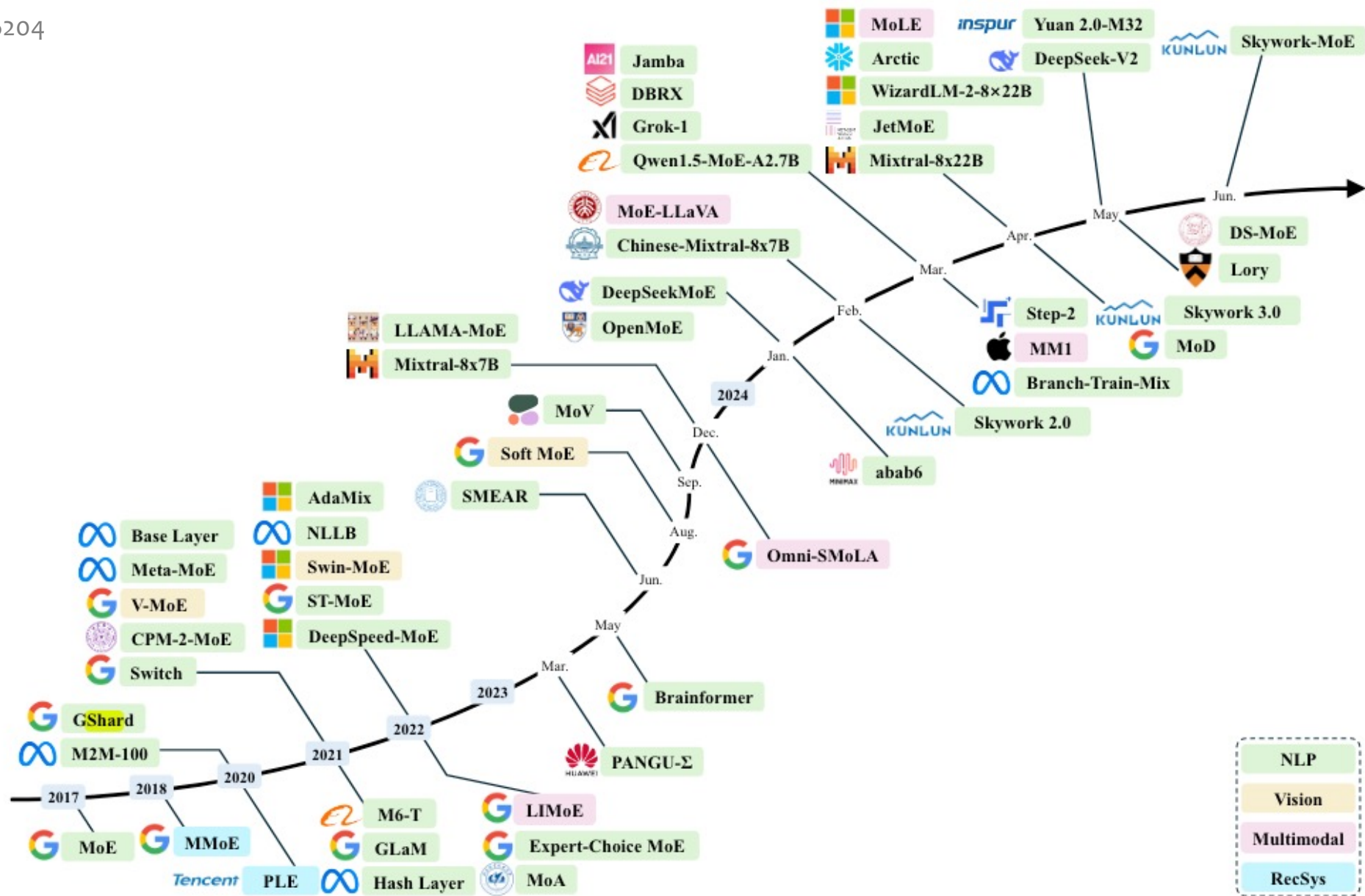
# MoE Timeline



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...)
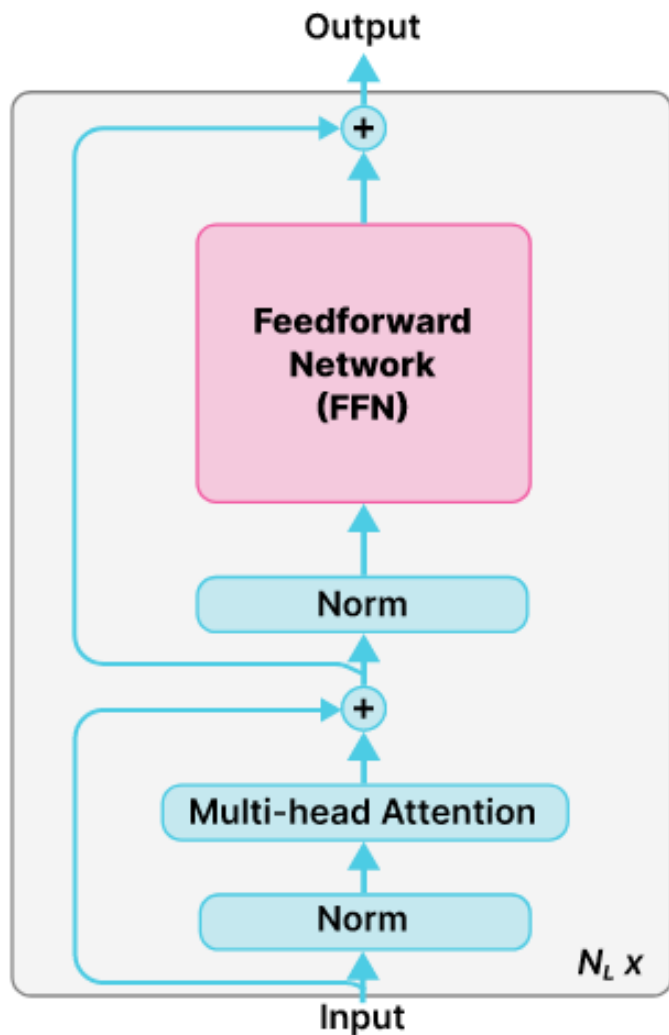
Output



$N_L x$

Input

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** |

Figure from http://arxiv.org/abs/2409.02060

# The Linear Layer in LLMs

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

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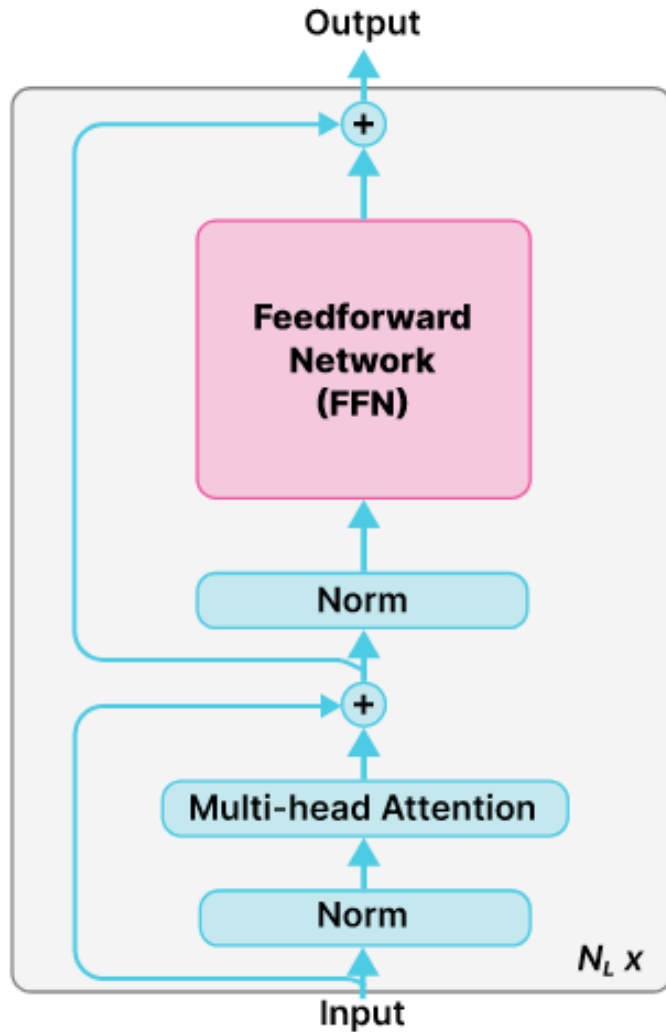
LayerNorm:
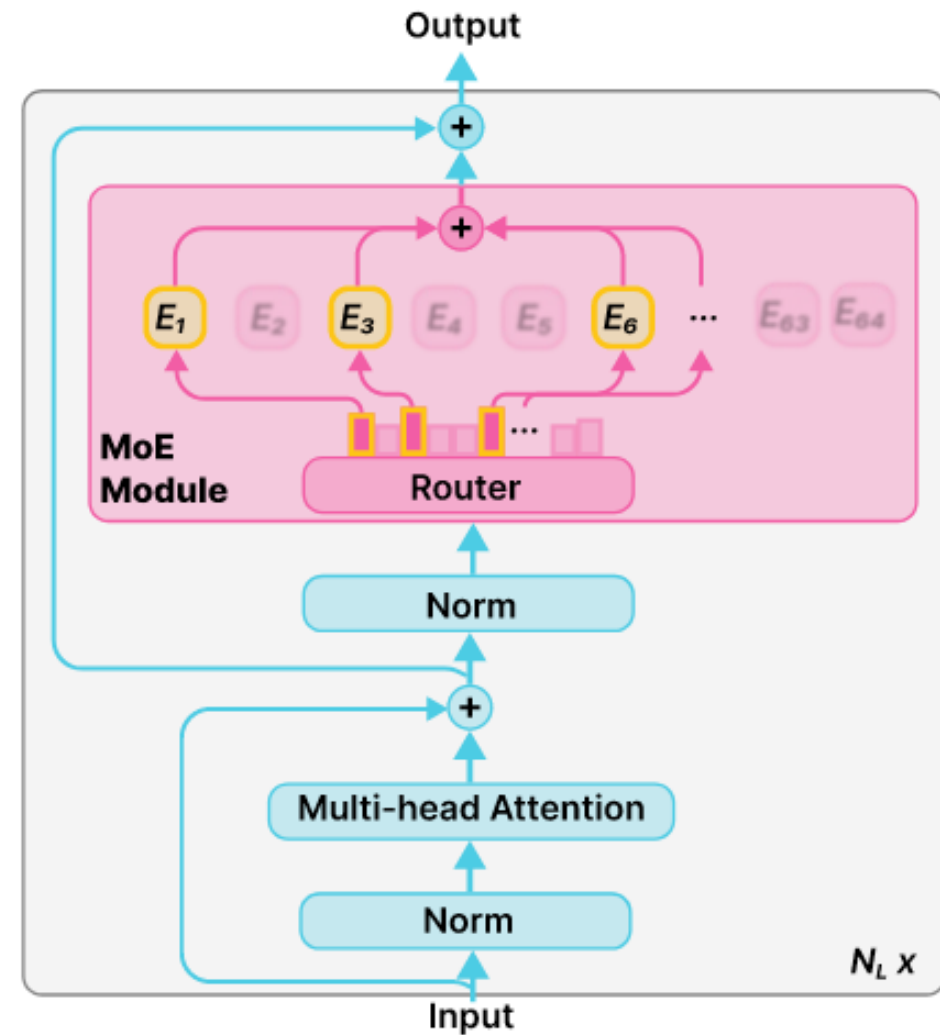
Embedding:

Attention:

Feed-forward:

| | 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

# Breaking a Linear Layer into Experts

**Linear Layer:**
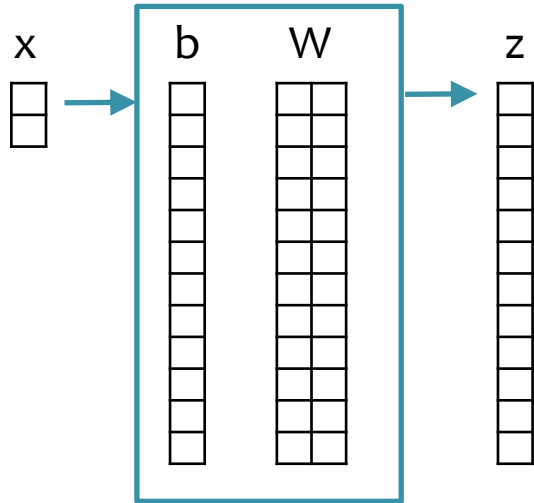
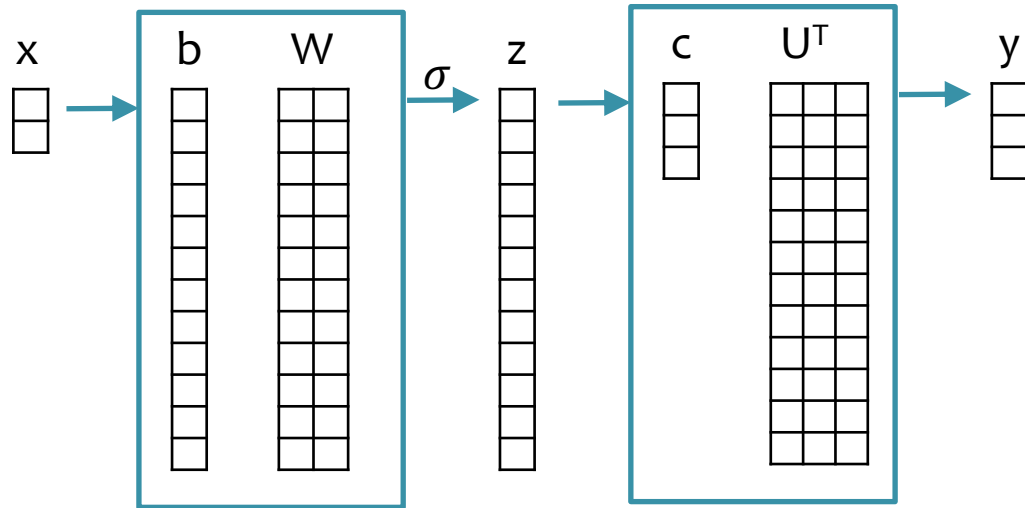**3 Experts:** (with same # of total parameters)



$$z = Wx + b$$
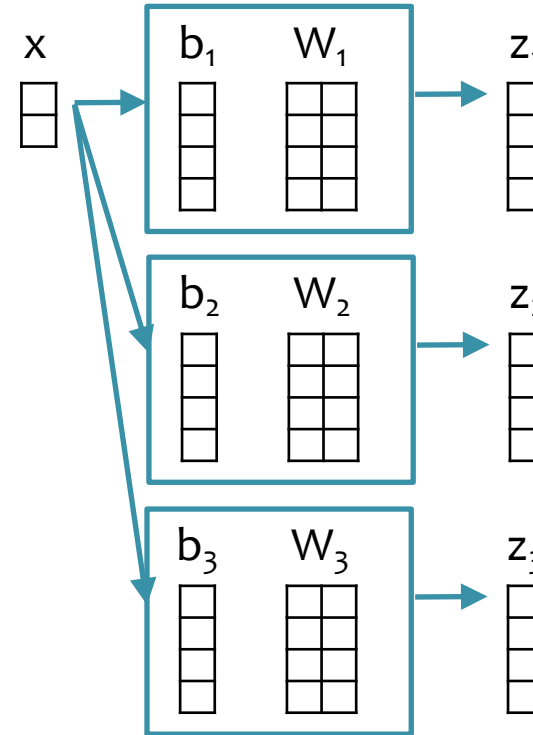
$$z_i = W_i x + b_i$$

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:**

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



$$y = U \, \sigma(Wx + b) + c$$
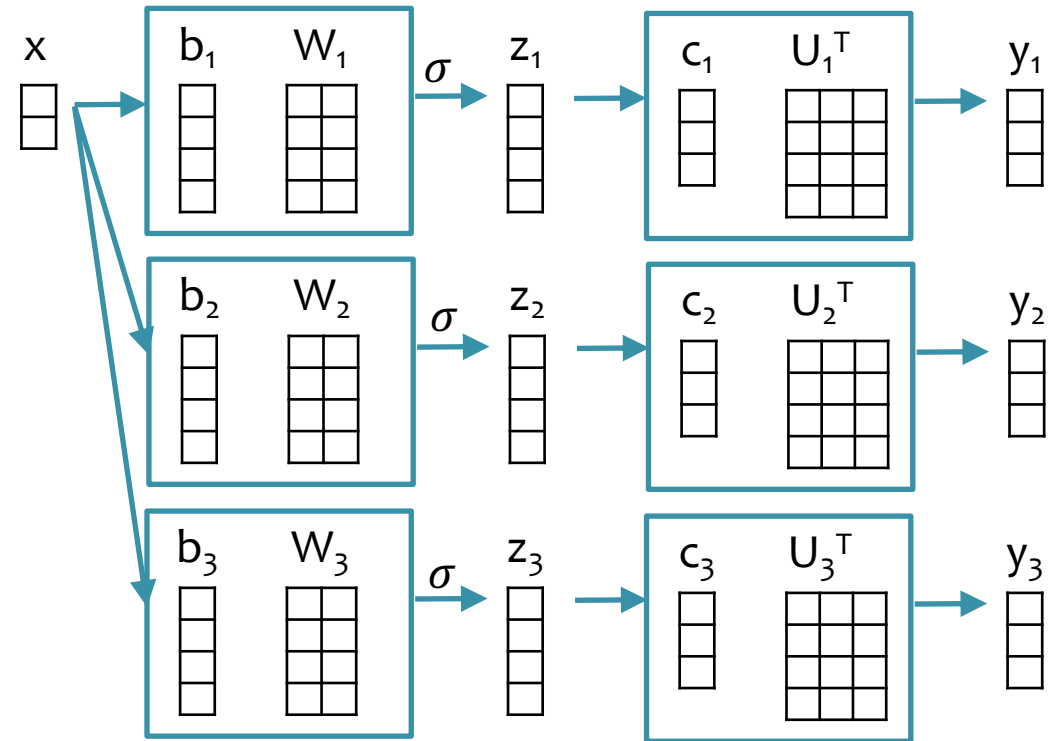
$$y_i = U_i \, \sigma(W_ix + b_i) + c_i$$

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 $c = \underline{\hspace{2cm}}$?

# **Dense** Mixture of Experts

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

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

**Dense MoE**

- Dense Softmax

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

- Each expert is just a feed-forward network

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

Figure from http://arxiv.org/abs/2401.04088

# **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

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



MoE layer

$G(x)_2$ $G(x)_{n-1}$

Expert 1 | Expert 2 | Expert 3 $\cdots$ Expert n-1 | Expert n

Gating Network

# Sparsely-Gated Mixture of Experts

**Definition of top-k:**

**Example: Computing the gate**

**Example: Computing the MoE**

# **Sparsely-Gated** Mixture of Experts

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

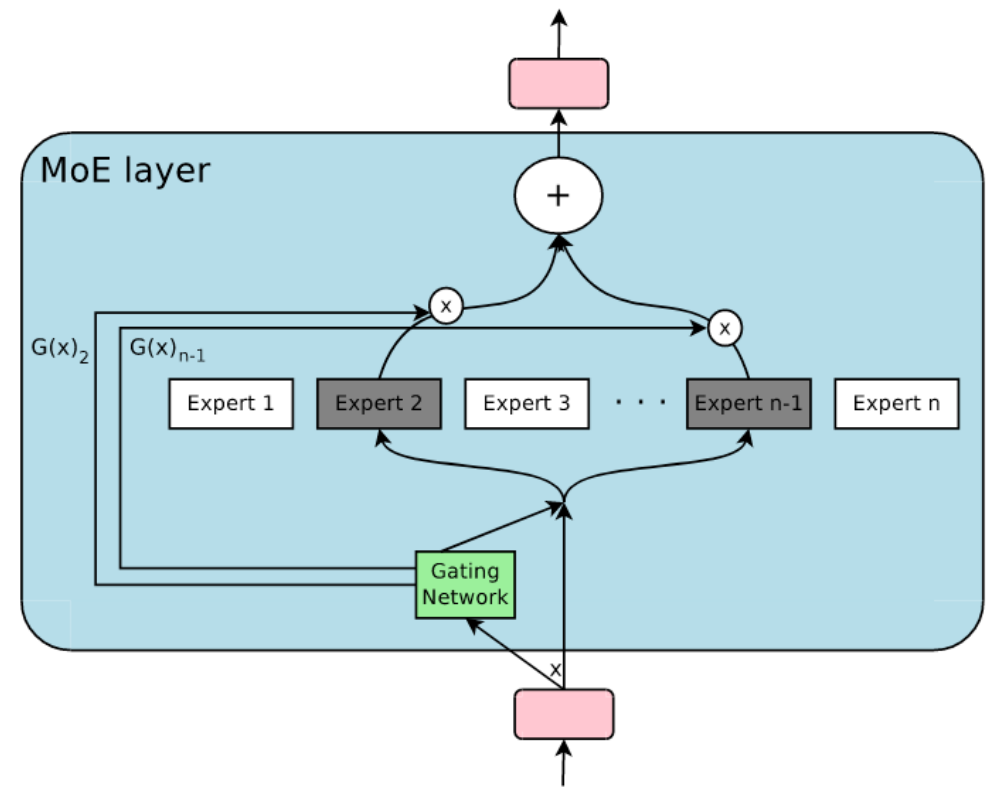$$\mathbf{y} = \sum_{i=1}^{N_e} G(\mathbf{x})_i E_i(\mathbf{x})$$

**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

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



MoE layer

$G(x)_2$  $G(x)_{n-1}$

Expert 1  Expert 2  Expert 3  $\cdots$  Expert n-1  Expert n

Gating Network

Figure from http://arxiv.org/abs/2401.04088

# **Sparsely-Gated** Mixture of Experts

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

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

**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

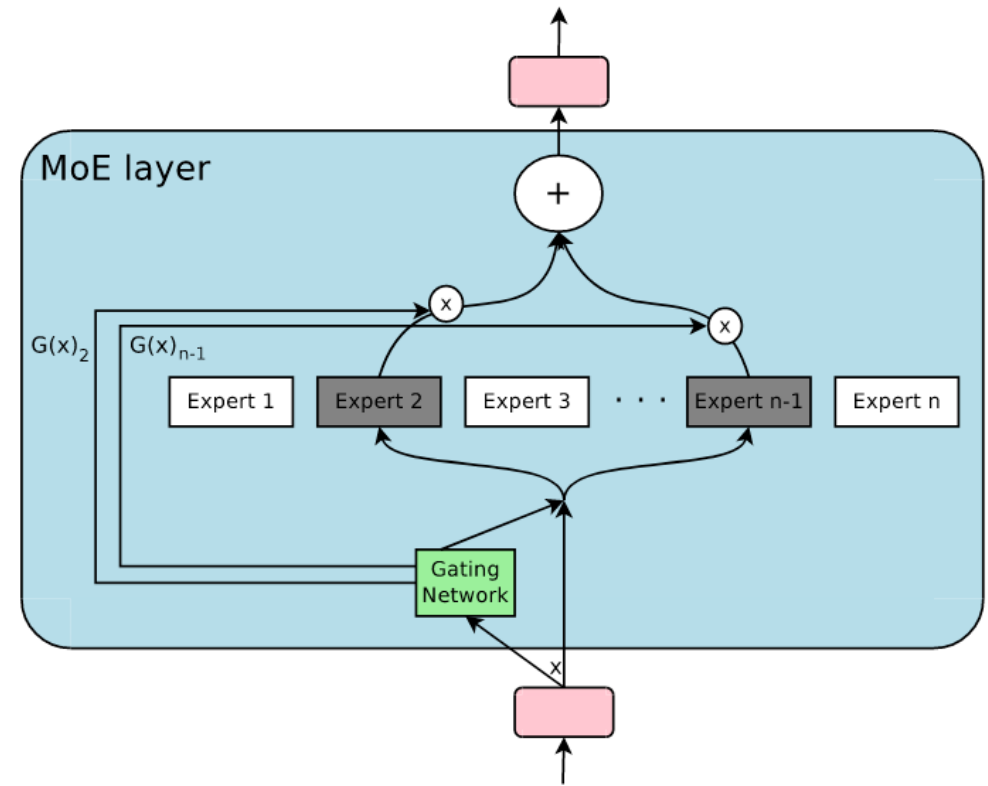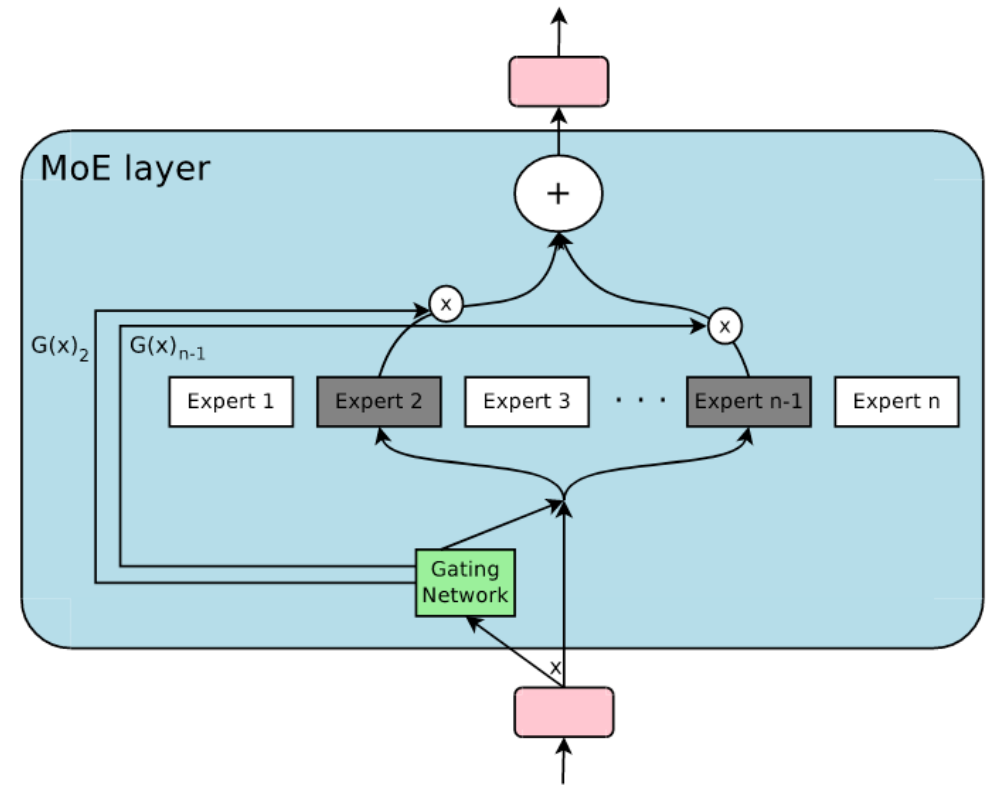$$\mathbf{y} = \sum_{i=1}^{N_e} G(\mathbf{x})_i E_i(\mathbf{x})$$

**Mixtral**

- Softmax over Top-K Gating

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

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

- Each expert is just a feed-forward network

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

# **Sparsely-Gated** Mixture of Experts

Sparsely-Gated MoEs were originally introd
are generally applicable and now popular fo

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

Initialization:

- We initialize $W_g$ and $W_{noise}$ to all zeros
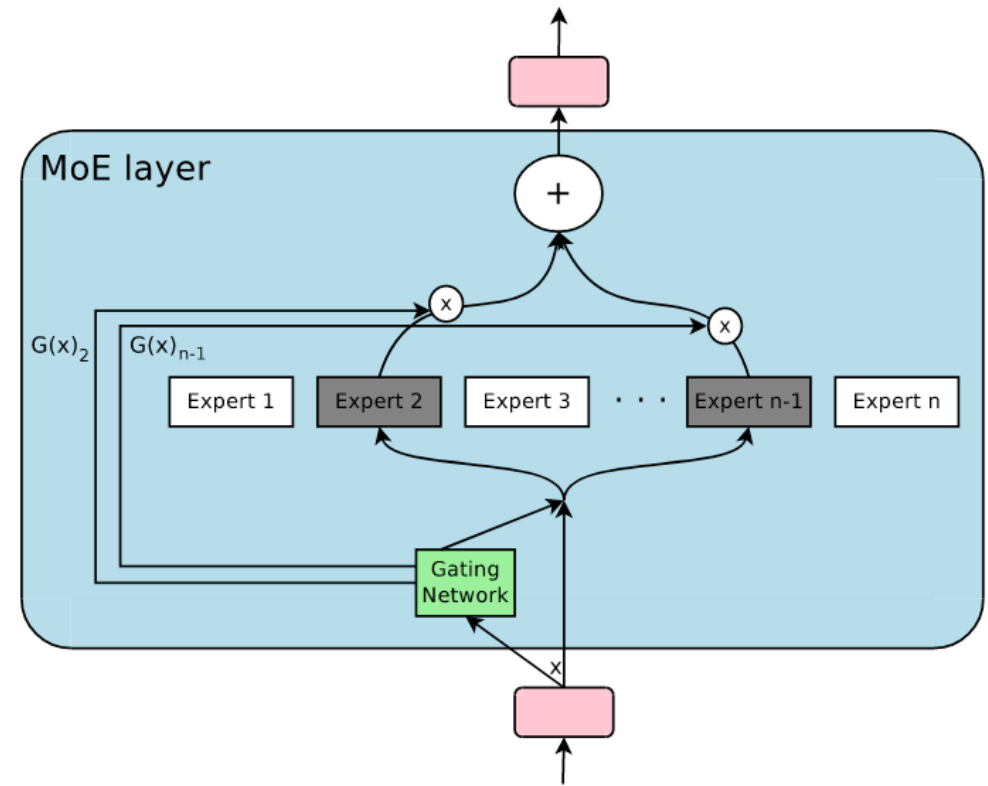- Effectively provides no signal and a small amount of noise

**Mixtral**

- Softmax over Top-K Gating

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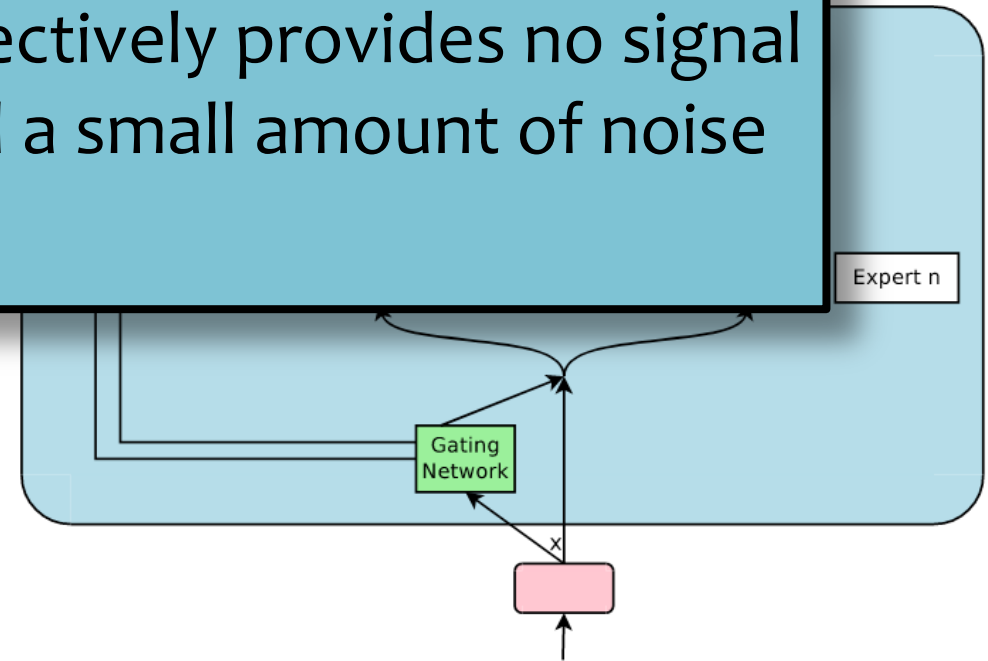
$$\mathbf{r}_{\text{noise}}(\mathbf{x}) = N(\mathbf{0}, \mathbf{I}) \cdot \text{sigma}(\mathbf{x}\mathbf{W}_{\text{noise}} + \mathbf{b}_{\text{noise}})$$

- Each expert is just a feed-forward network

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

Expert n

Gating Network

# 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
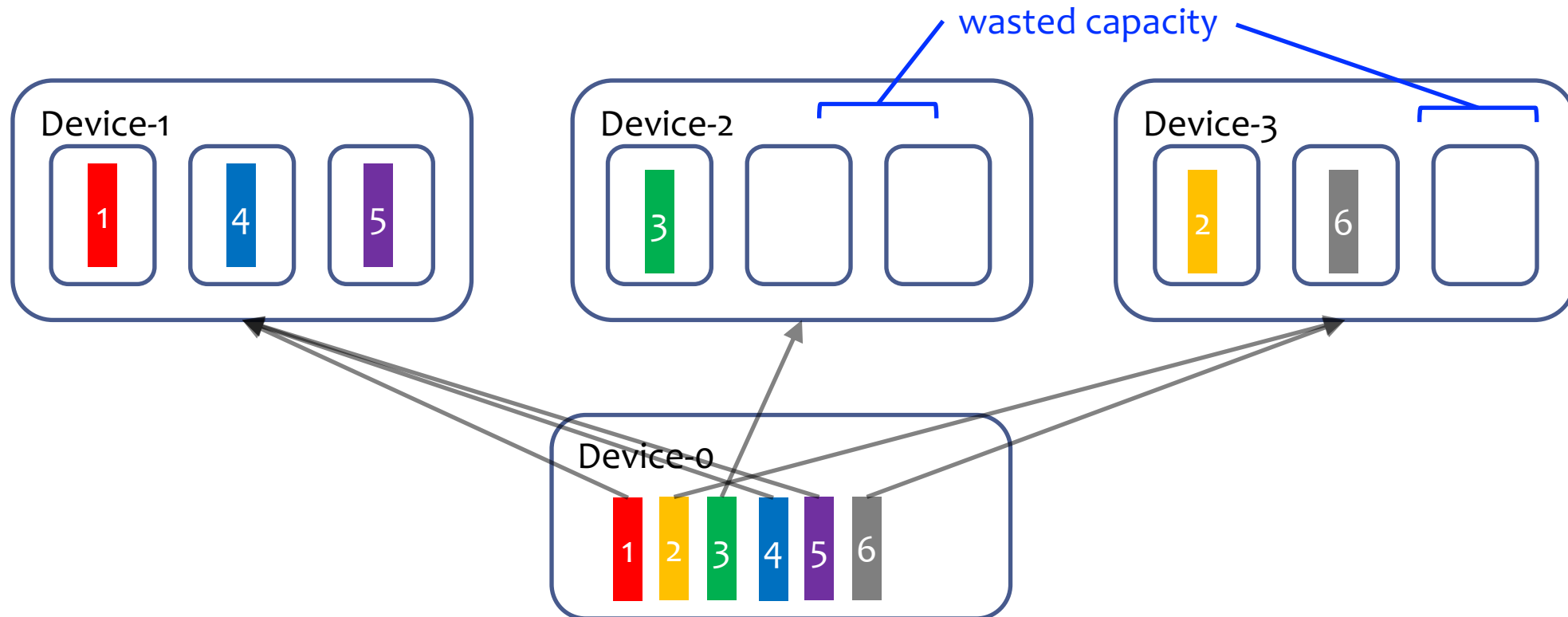
# Expert Parallelism

- 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



token 3 was also assigned to Device-1, but it's already full!
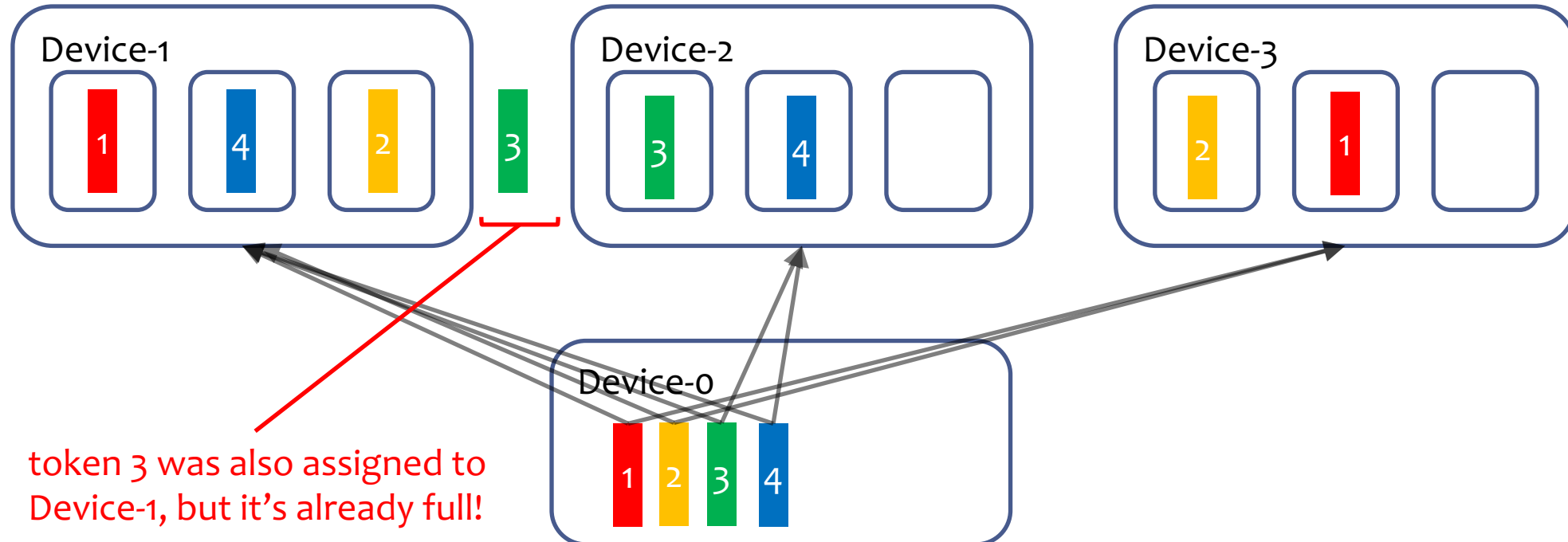
# Balancing Expert Utilization in an MoE

- **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

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

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

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

$f_i =$ fraction of tokens routed to expert $i$

$P_i =$ probability to $E_i$ in current batch

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

# 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

Figure from http://arxiv.org/abs/2401.04088
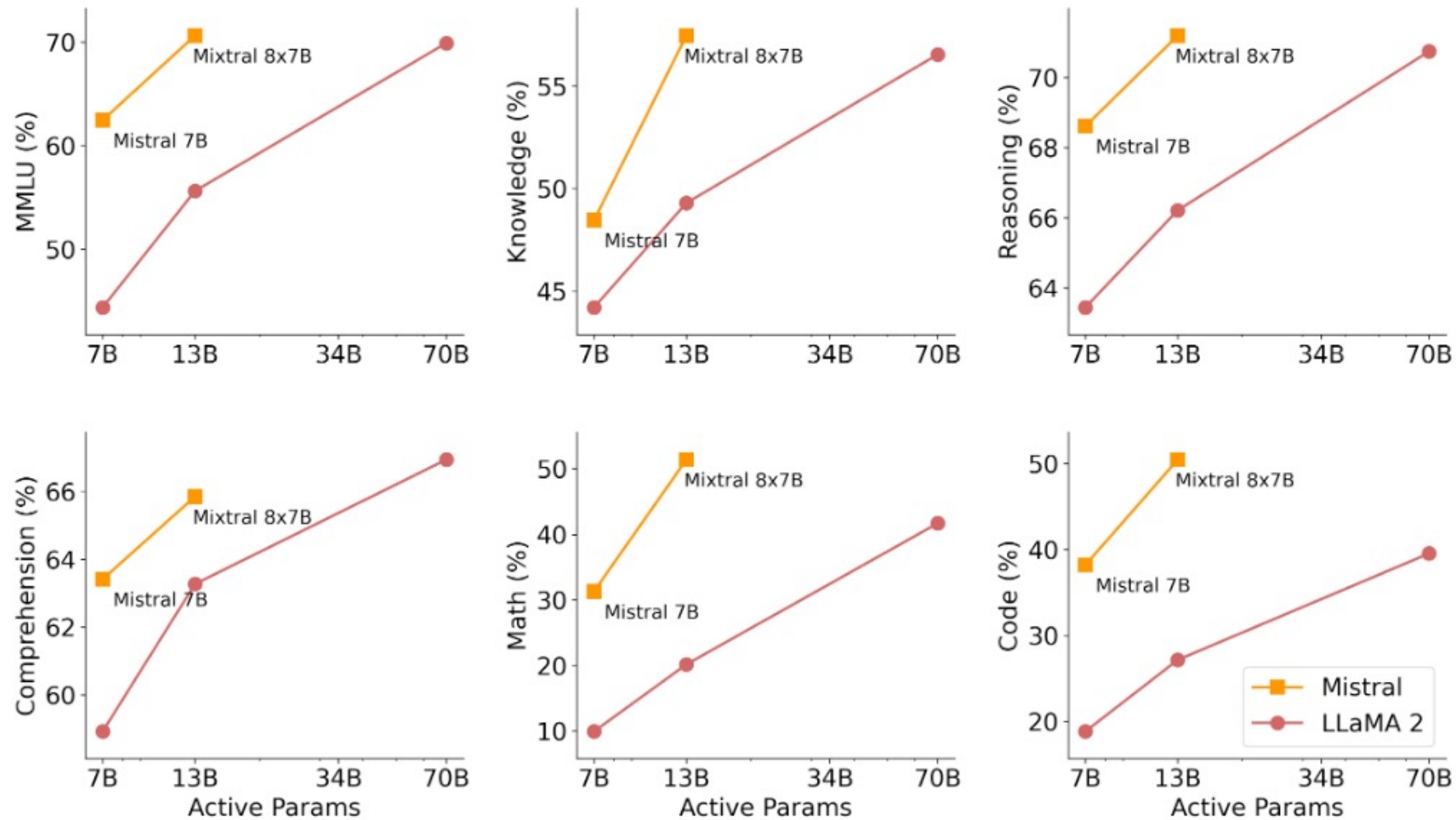
# OlMoE Hyperparameters

| | OLMoE-1B-7B | JetMoE | OpenMoE | OLMo-1B (0724) |
|---|---|---|---|---|
| Dimension | 2,048 | 2,048 | 2,048 | 2,048 |
| Activation | SwiGLU | SwiGLU | SwiGLU | SwiGLU |
| FFN dimension | 1,024 | 5,632 | 8,192 | 8,192 |
| Vocab size | 50,304 | 32,000 | 256,384 | 50,304 |
| Attn heads | 16 | 16 | 24 | 16 |
| Num layers | 16 | 24 | 32 | 16 |
| Layer norm type | RMSNorm | RMSNorm | RMSNorm | non-parametric |
| Layer norm eps | 1.0E-05 | 1.0E-05 | 1.0E-06 | 1.0E-05 |
| QK-Norm | yes | no | no | no |
| 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 |
| Init dist | trunc normal | ? | ? | normal |
| 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 |
| # Total params | 6.9B | 8.5B | 8.7B | 1.3B |

| | | | | |
|---|---|---|---|---|
| Sequence length | 4,096 | 4,096 | 2,048 | 4,096 |
| Batch size (samples) | 1,024 | 1,024 | 2,048 | 512 |
| Batch size (tokens) | ~4M | ~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 |
| AdamW epsilon | 1.0E-08 | ? | - | 1.0E-05 |
| 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 |
| LBL weight | 0.01 | 0.01 | 0.01 | - |
| 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

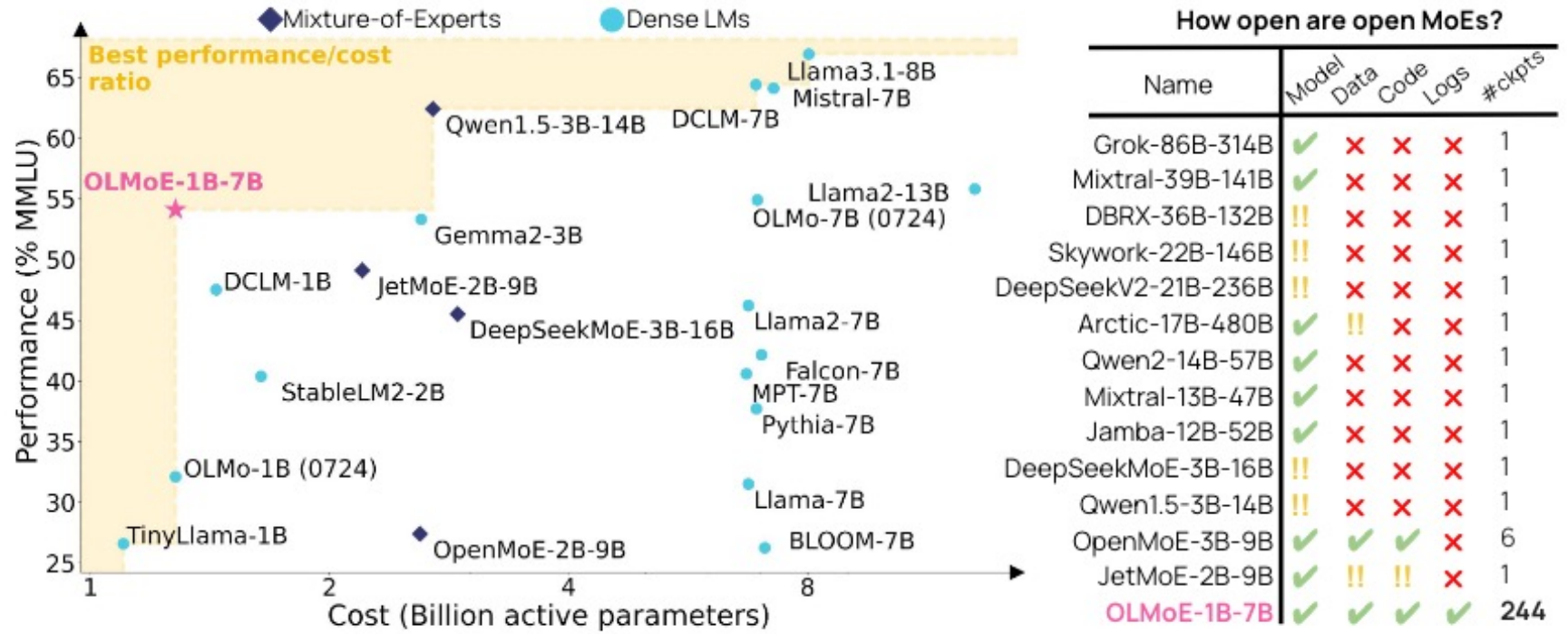MoEs provide a nice tradeoff between performance and FLOPS cost



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.

Figure from http://arxiv.org/abs/2409.02060

# Performance vs. Cost

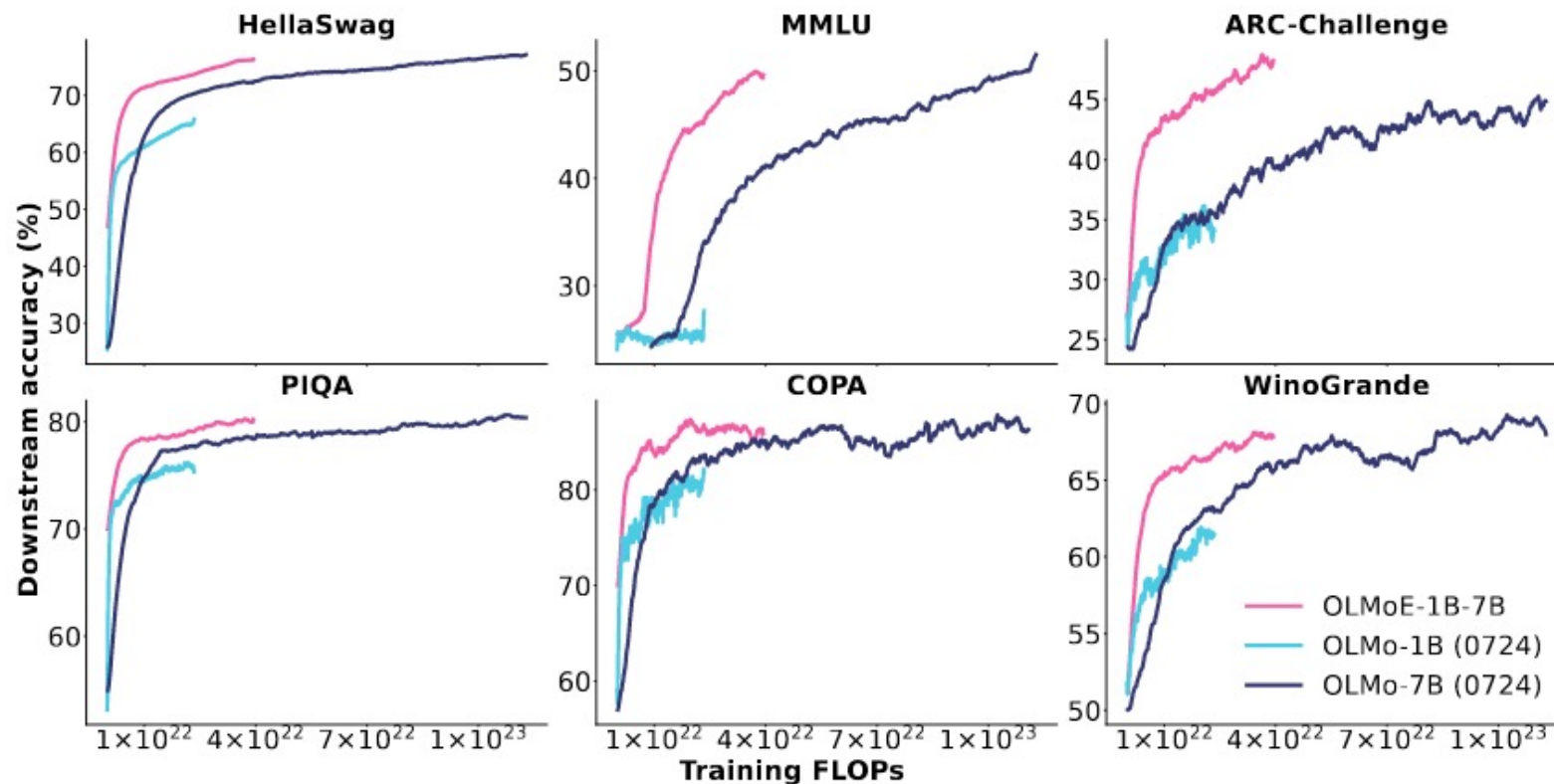MoEs provide a nice tradeoff between performance and FLOPS cost



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--Vmlldzo4OTcyMjEz

Figure from http://arxiv.org/abs/2409.02060

# How many experts to choose?

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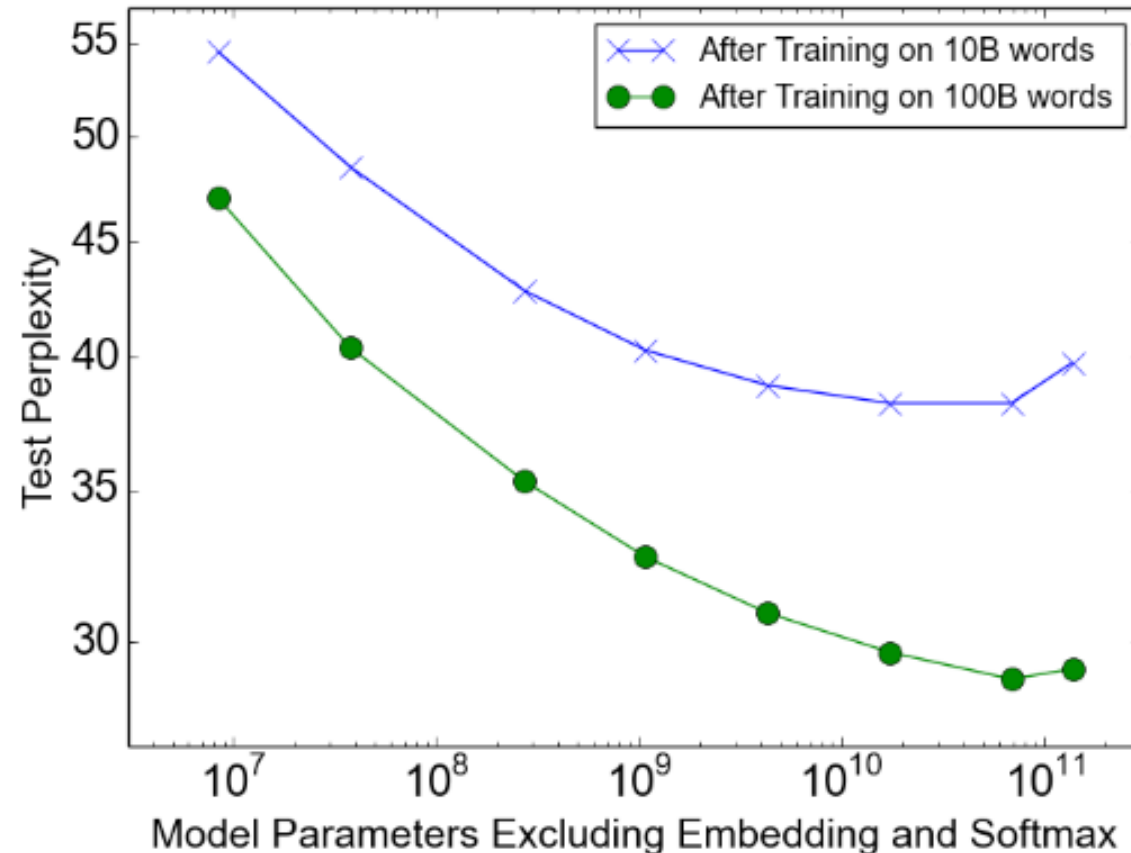


32, 256, 1024, 4096, 16384, 65536, and 131072 experts.

up to 137 billion parameters in the MoE layer.
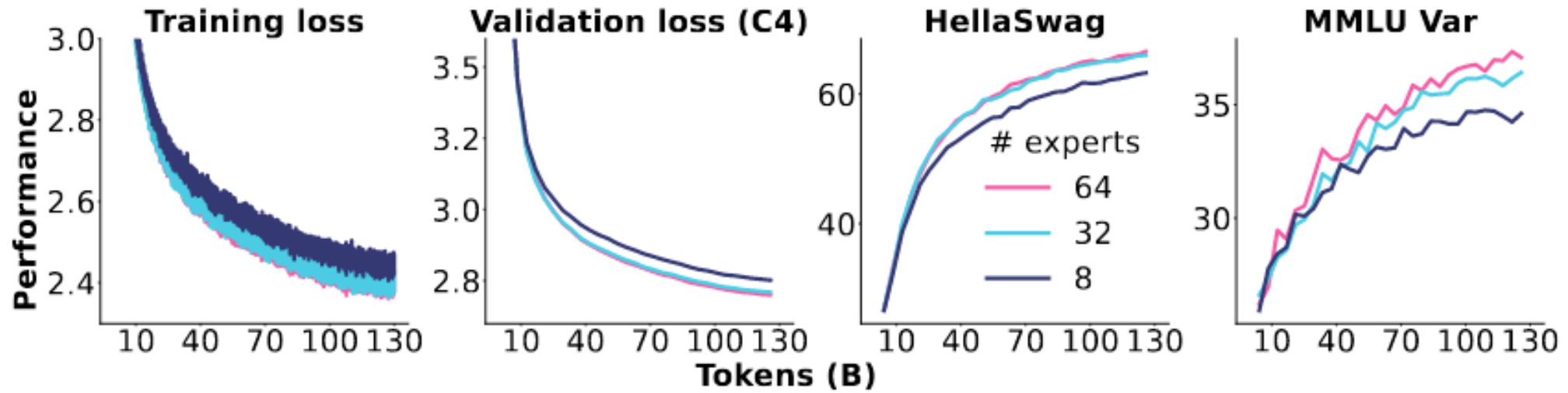
# How many experts to choose?



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--Vmlldzo4OTIxOTE4

More recent work on Transformer LMs has favored a comparatively small number of experts
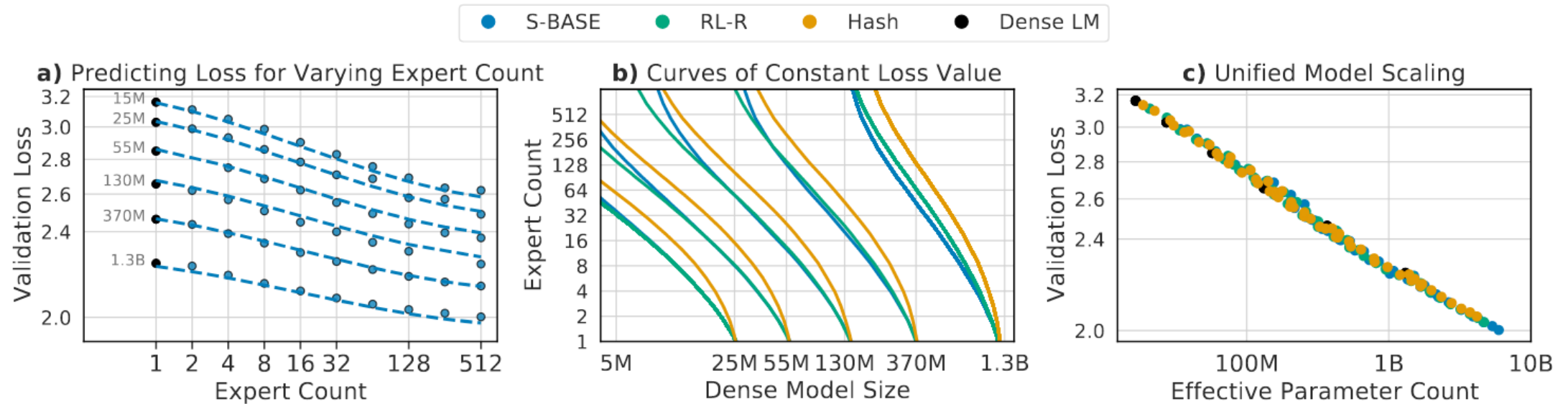
# Scaling Laws for Routed LMs



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.

Figure from https://proceedings.mlr.press/v162/clark22a.html