EE-608: Deep Learning For Natural Language Processing:

Diffusion

James Henderson

Idiap Research Institute

DLNLP, Lecture 10

Outline

Mixture of Experts

Other Methods for Faster Inference

Outline

The Linear Layer in LLMs

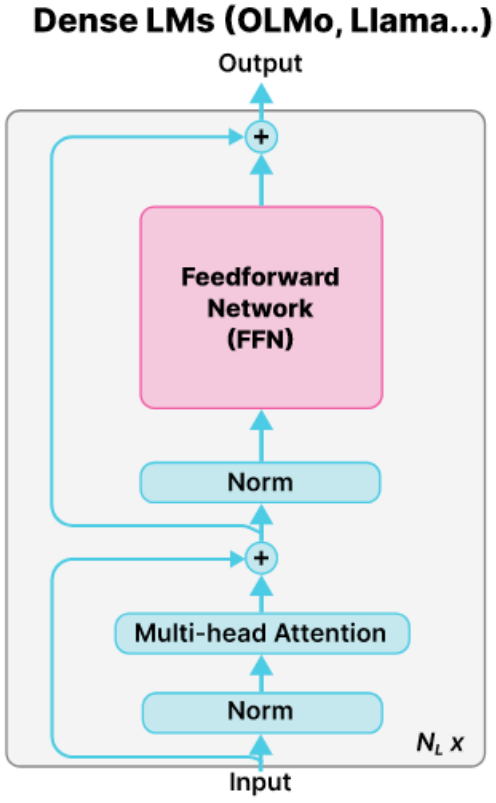


Figure from http://arxiv.org/abs/2409.02060

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**

Slide from Aran Nayebi & Matt Gormley

The Linear Layer in LLMs

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

LayerNorm:

$$\left(\left(\left(d_{\text{model}}\right) \times 2\right) \times 2\right) n_{\text{layers}}$$

Embedding:

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

Attention:

$$4\left(\left(d_{\text{model}}\right)^2 + d_{\text{model}}\right) n_{\text{layers}}$$

Feed-forward:

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

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

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

	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**

Slide from Aran Nayebi & Matt Gormley

Sparsely-Gated Mixture of Experts

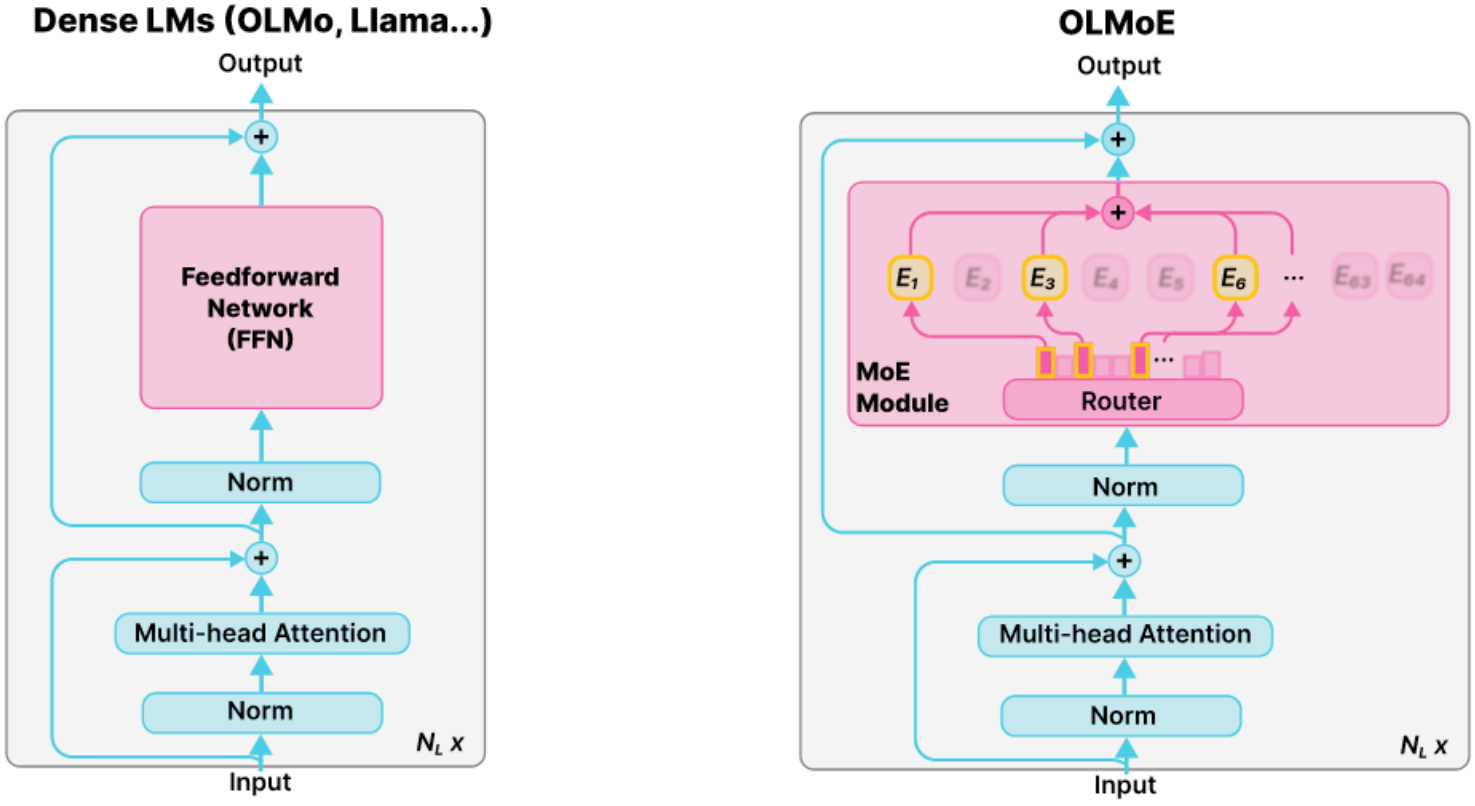
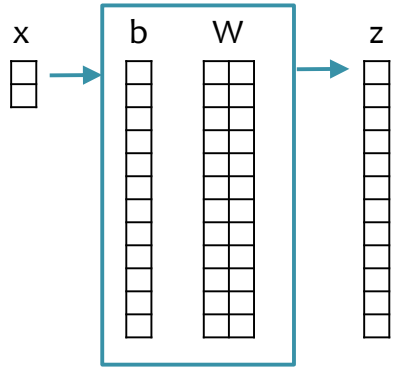


Figure from http://arxiv.org/abs/2409.02060

Slide from Aran Nayebi & Matt Gormley

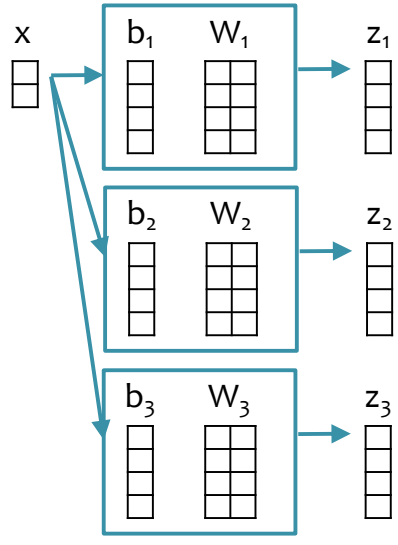
Breaking a Linear Layer into Experts

Linear Layer:



$z = Wx + b$

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



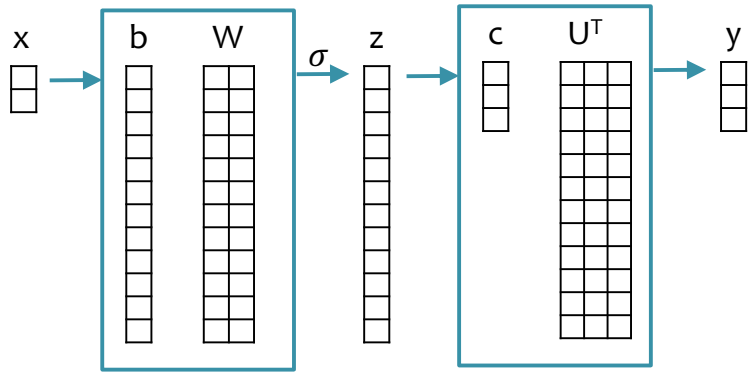
$z_i = W_i x + b_i$

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).

Slide from Aran Nayebi & Matt Gormley

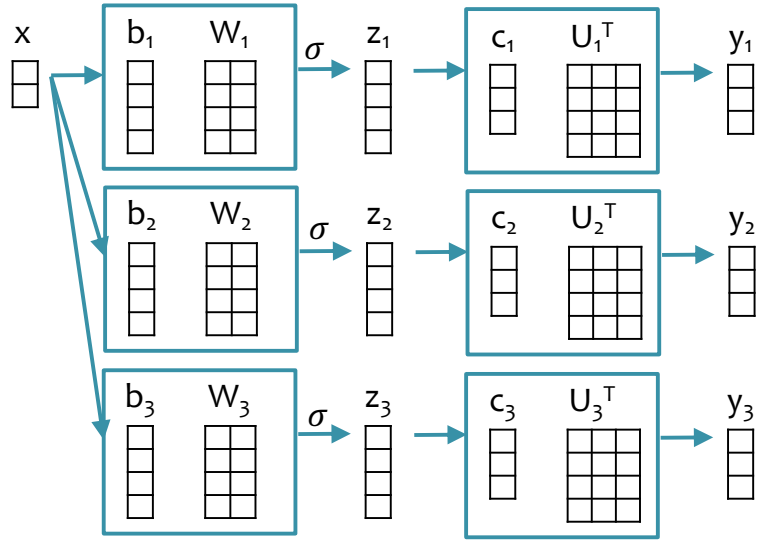
Breaking a Feed-forward Layer into Experts

Feed-forward Layer:



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

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



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

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 $y = \underline{y_1 + y_2 + y_3}$? and $c = \underline{c_1 + c_2 + c_3}$?

Slide from Aran Nayebi & Matt Gormley

Dense Mixture of Experts

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

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

Dense MoE

- Dense Softmax

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

- Each expert is just a feed-forward network

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

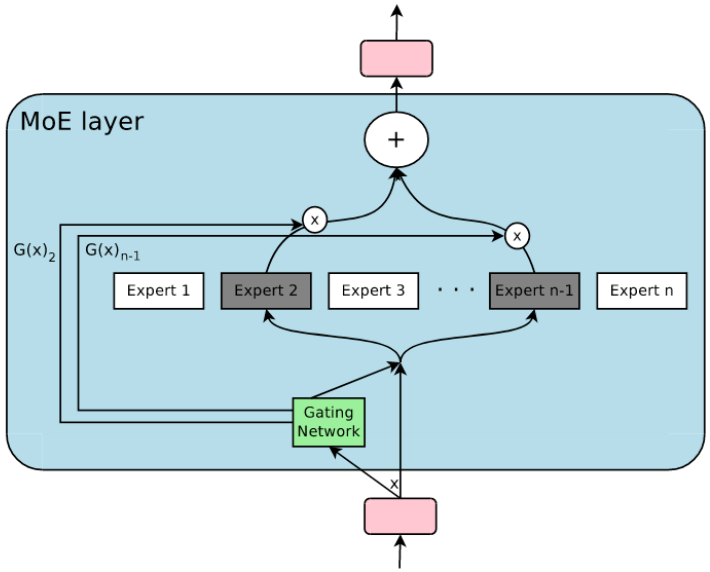
Figure from http://arxiv.org/abs/2401.04088

Slide from Aran Nayebi & Matt Gormley

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

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

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

Slide from Aran Nayebi & Matt Gormley

Sparsely-Gated Mixture of Experts

Definition of top-k:

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

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

Example: Computing the gate

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

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

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

Example: Computing the MoE

$$\vec{y} = G_1 E_1(\vec{x}) + G_2 E_2(\vec{x}) + G_3 E_3(\vec{x}) + G_4 E_4(\vec{x})$$

$$= 0 \cdot E_1(\vec{x}) + \frac{e^3}{e^3 + e^8} E_2(\vec{x}) + \frac{e^8}{e^3 + e^8} E_3(\vec{x}) + 0 \cdot E_4(\vec{x})$$

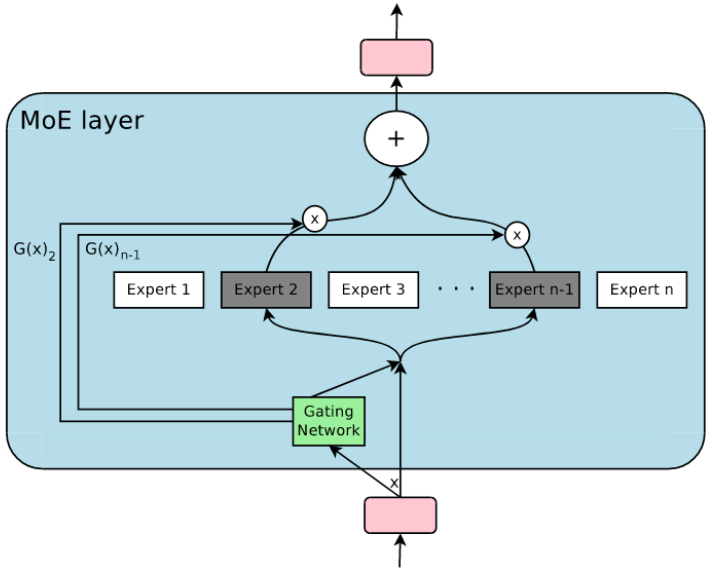
$$= \frac{e^3}{e^3 + e^8} E_2(\vec{x}) + \frac{e^8}{e^3 + e^8} E_3(\vec{x}).$$

Slide from Aran Nayebi & Matt Gormley

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

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

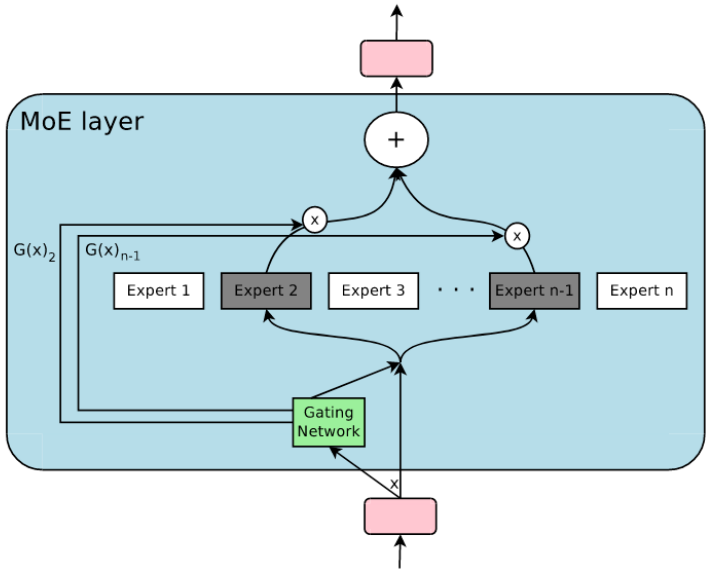
Figure from http://arxiv.org/abs/2401.04088

Slide from Aran Nayebi & Matt Gormley

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

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

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

- Each expert is just a feed-forward network

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

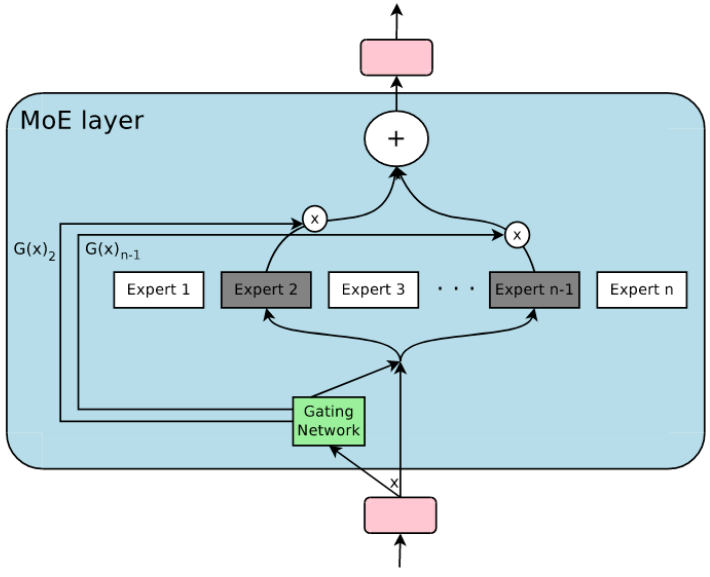
Figure from http://arxiv.org/abs/2401.04088

Slide from Aran Nayebi & Matt Gormley

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

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

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

- Each expert is **still** just a feed-forward network

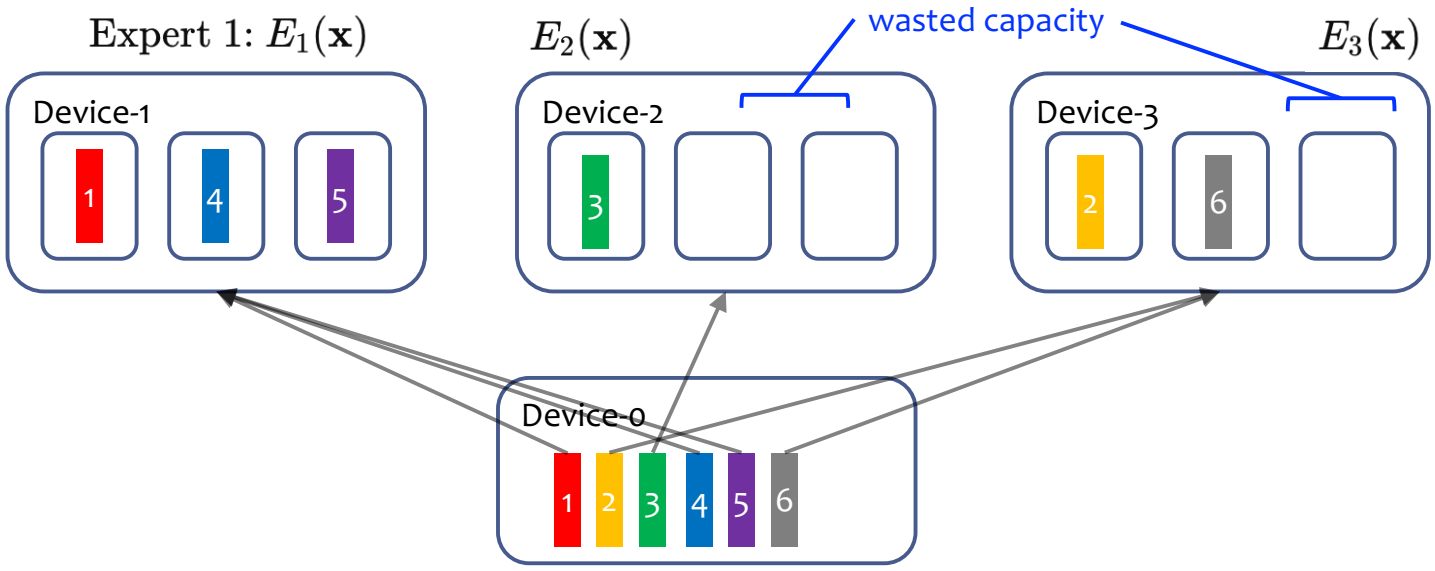
$$E_i(\mathbf{x}) = \text{SwiGLU}(\mathbf{x}) = \text{Swish}(\mathbf{x}\mathbf{W} + \mathbf{b}) \odot (\mathbf{x}\mathbf{V} + \mathbf{c})$$

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

Slide from Aran Nayebi & Matt Gormley

Expert Parallelism

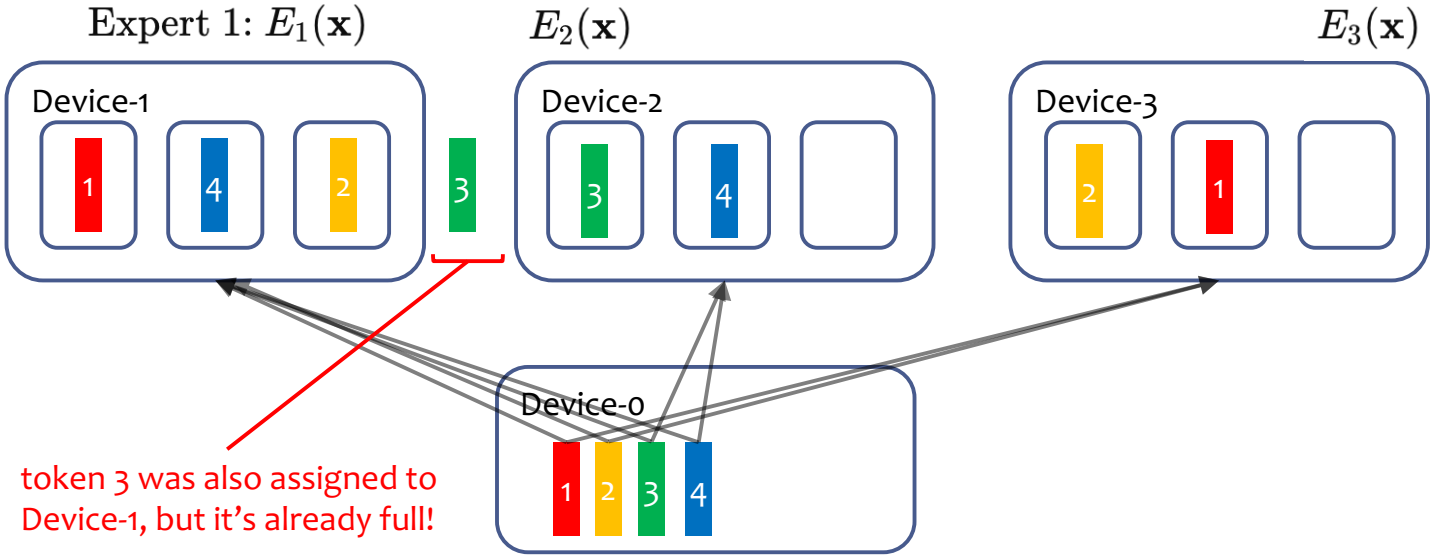
- Each expert can be allocated to a different device (GPU)
- Each token is routed to K experts
- Load balancing issues if too many tokens are routed to the same expert
- **Example:** 3 devices, capacity of 3 tokens/device, 6 tokens, K=1 experts per token



Slide from Aran Nayebi & Matt Gormley

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



Slide from Aran Nayebi & Matt Gormley

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

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

Slide from Aran Nayebi & Matt Gormley

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

Slide from Aran Nayebi & Matt Gormley

Mixtral vs. Llama-2

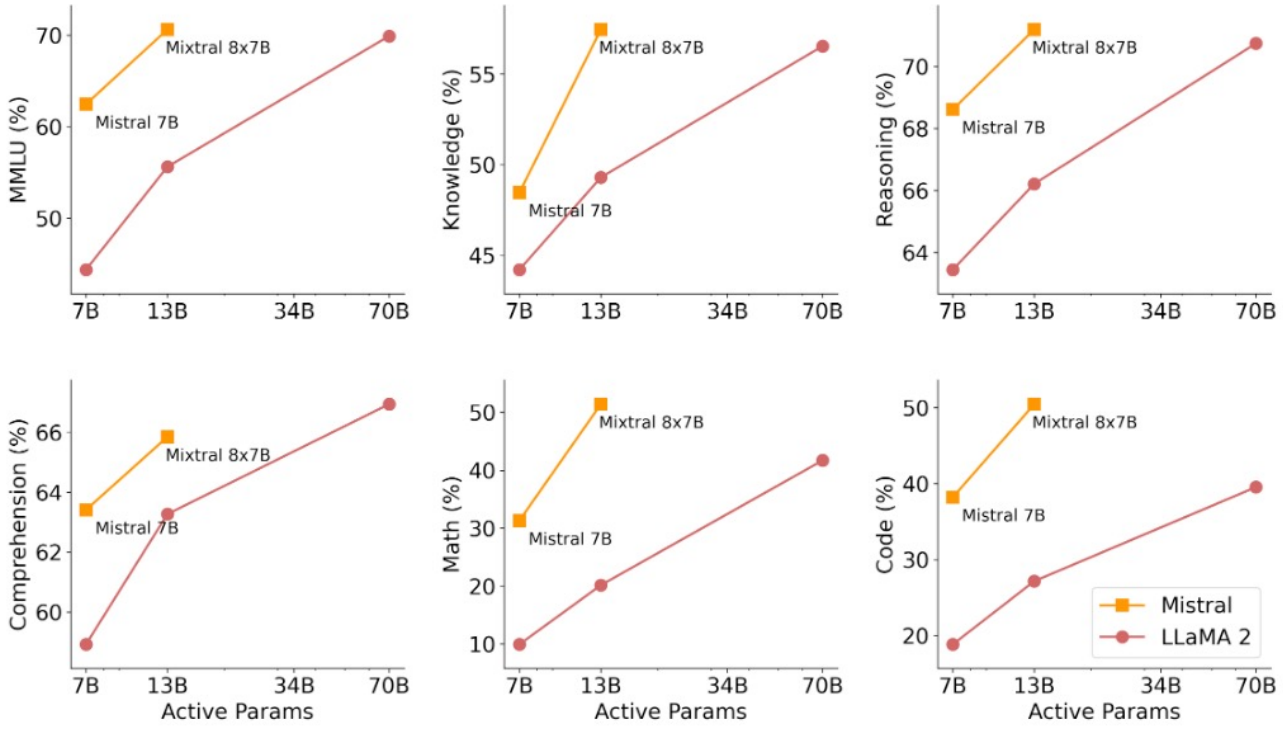


Figure from http://arxiv.org/abs/2401.04088

Slide from Aran Nayebi & Matt Gormley

OlMoE Hyperparameters

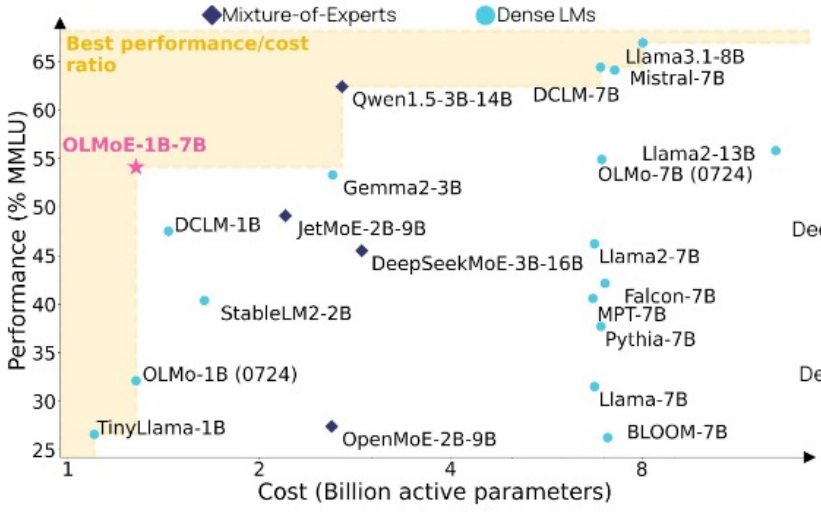
	OLMoE-1B-7B	JetMoE	OpenMoE	OLMo-1B (0724)
Dimension	2,048	2,048	2,048	2,048
Activation	SwiGLU	SwiGLU	SwiGLU	SwiGLU
FFN dimension	1,024	5,632	8,192	8,192
Vocab size	50,304	32,000	256,384	50,304
Attn heads	16	16	24	16
Num layers	16	24	32	16
Layer norm type	RMSNorm	RMSNorm	RMSNorm	non-parametric
Layer norm 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 θ	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.

Slide from Aran Nayebi & Matt Gormley

Performance vs. Cost

MoEs provide a nice tradeoff between performance and FLOPS cost



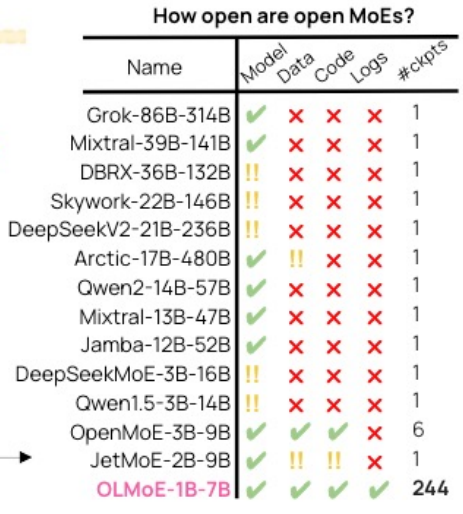


Figure 1: **Performance, cost, and degree of openness of open MoE and dense LMs.** Model names contain rounded parameter counts: `model-active-total` for MoEs and `model-total` for dense LMs. `#ckpts` is the number of intermediate checkpoints available. We highlight MMLU as a summary of overall performance; 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

Slide from Aran Nayebi & Matt Gormley

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 & sometimes accuracy too! (measured in Training FLOPS)

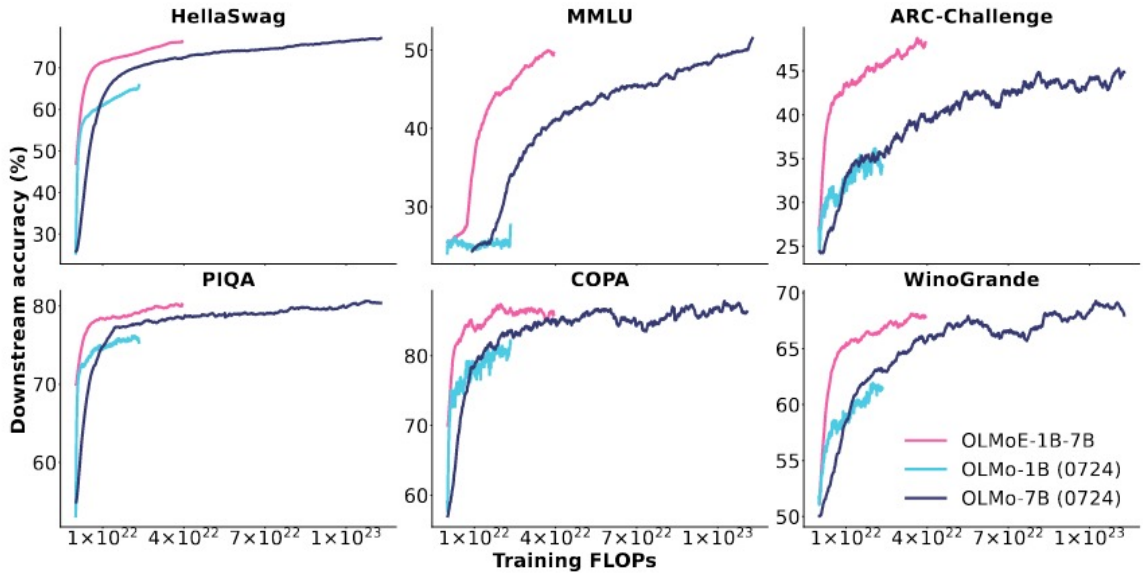


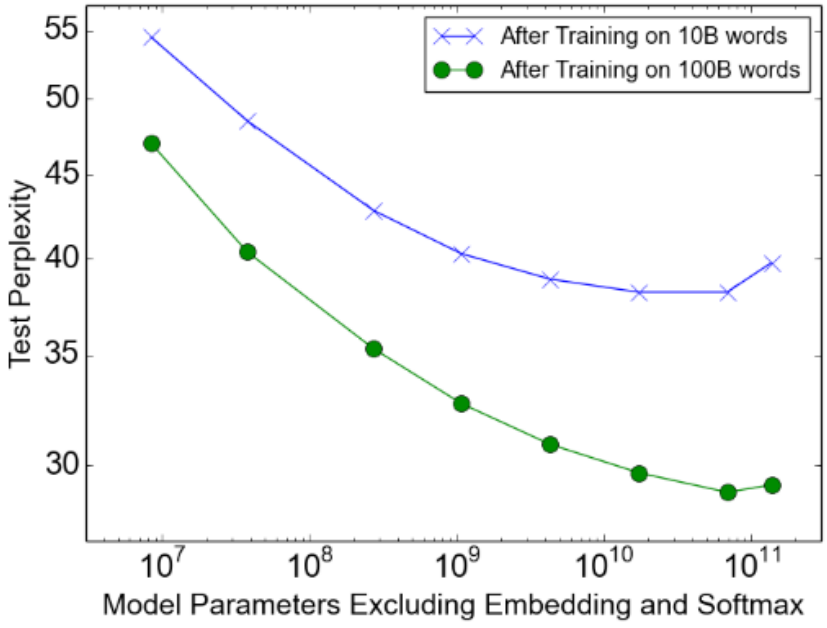
Figure 3: **Evaluation of OLMoE-1B-7B and the current best OLMo models during pretraining.** **OLMoE-1B-7B** differs from the OLMo models in its MoE architecture, several training hyperparameters, and its training dataset, see §2. A version of this plot with tokens as the x-axis and markers where annealing 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--Vmlldzo4OTcyMjEz

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

Slide from Aran Nayebi & Matt Gormley

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.

Slide from Aran Nayebi & Matt Gormley

How many experts to choose?

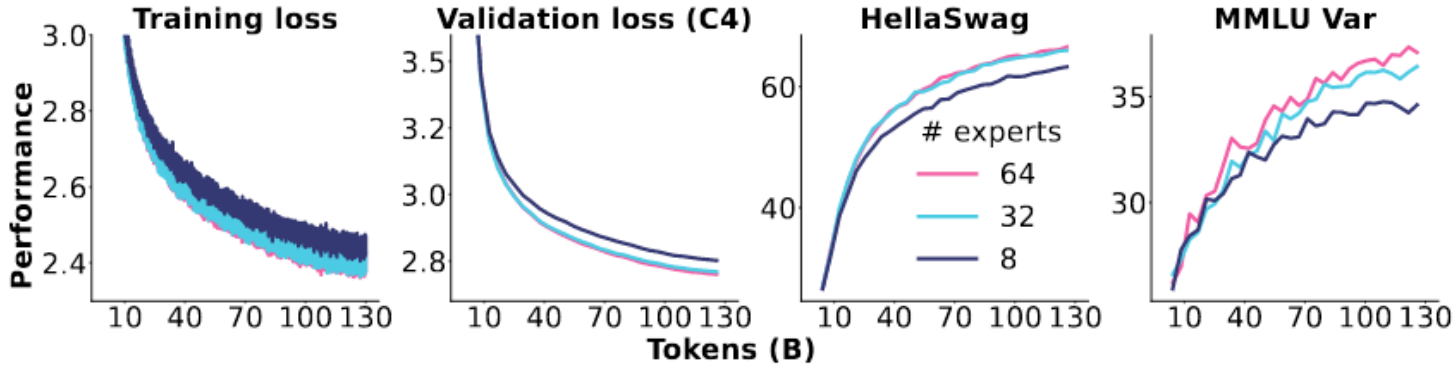


Figure 5: **Expert granularity.** We vary the number of experts in tandem with the FFN dimension to ensure that active and total parameters and thus compute cost remain the same. For example, for 64 experts, the FFN dimension is 1,024 and 8 experts are activated, while for 32 experts it is 2,048 with 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

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

Slide from Aran Nayebi & Matt Gormley

Scaling Laws for Routed (MoE) LMs

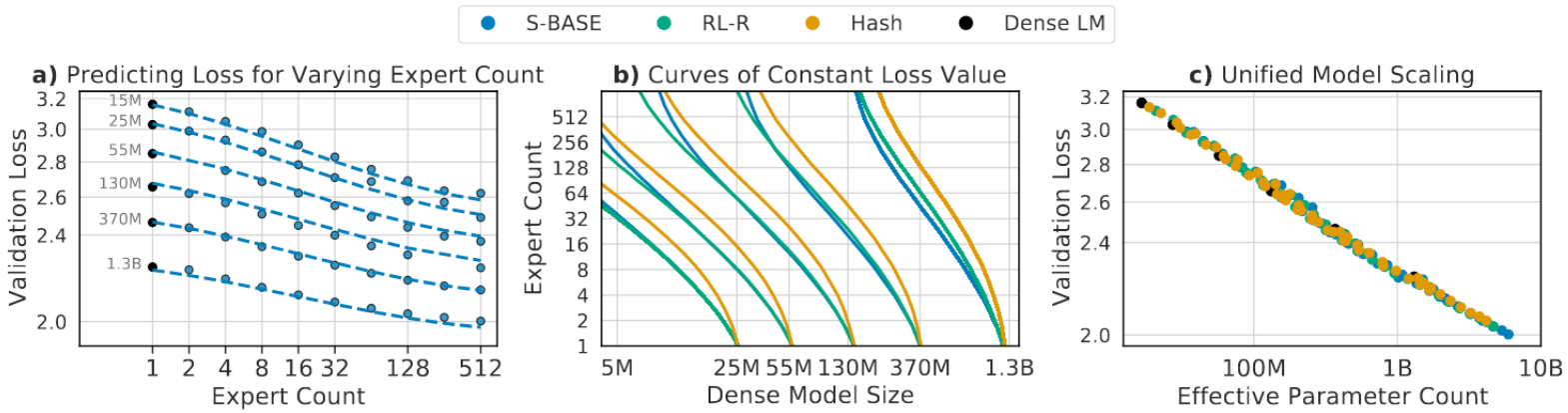


Figure 1. **(a)** The performance achieved by Routing Networks when varying the number of experts for a fixed dense model size is described by a bilinear function (Eq. 1), **(b)** whose level curves indicate how to trade model size with expert count to maintain a fixed performance, **(c)** and which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

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

Slide from Aran Nayebi & Matt Gormley

Summary of Mixture of Experts

MoE is a simple and effective way to improve speed with large models

- Replace each FF layer's MLP with many smaller MLP experts, and learn a router to assign sparse weights to the different experts
- Only need to compute and update the experts with nonzero weights
- Fewer active parameters with larger trained parameters gives a better tradeoff of speed versus accuracy

Outline

KV Caching

Autoregressive LLMs predict one token at a time, but the prefix is mostly shared across predictions.

- Causal attention makes the computation for token t independent of all tokens t' after t, so there is no need to compute it more than once.
- A token's computation is only dependent on the key and value vectors ("KV") of past tokens, so we only need to store these ("KV-cache")
- When the KV-cache becomes large, further optimisations are useful
- *Maybe denoising attention is relevant?*

Taxonomy of KV Cache Management

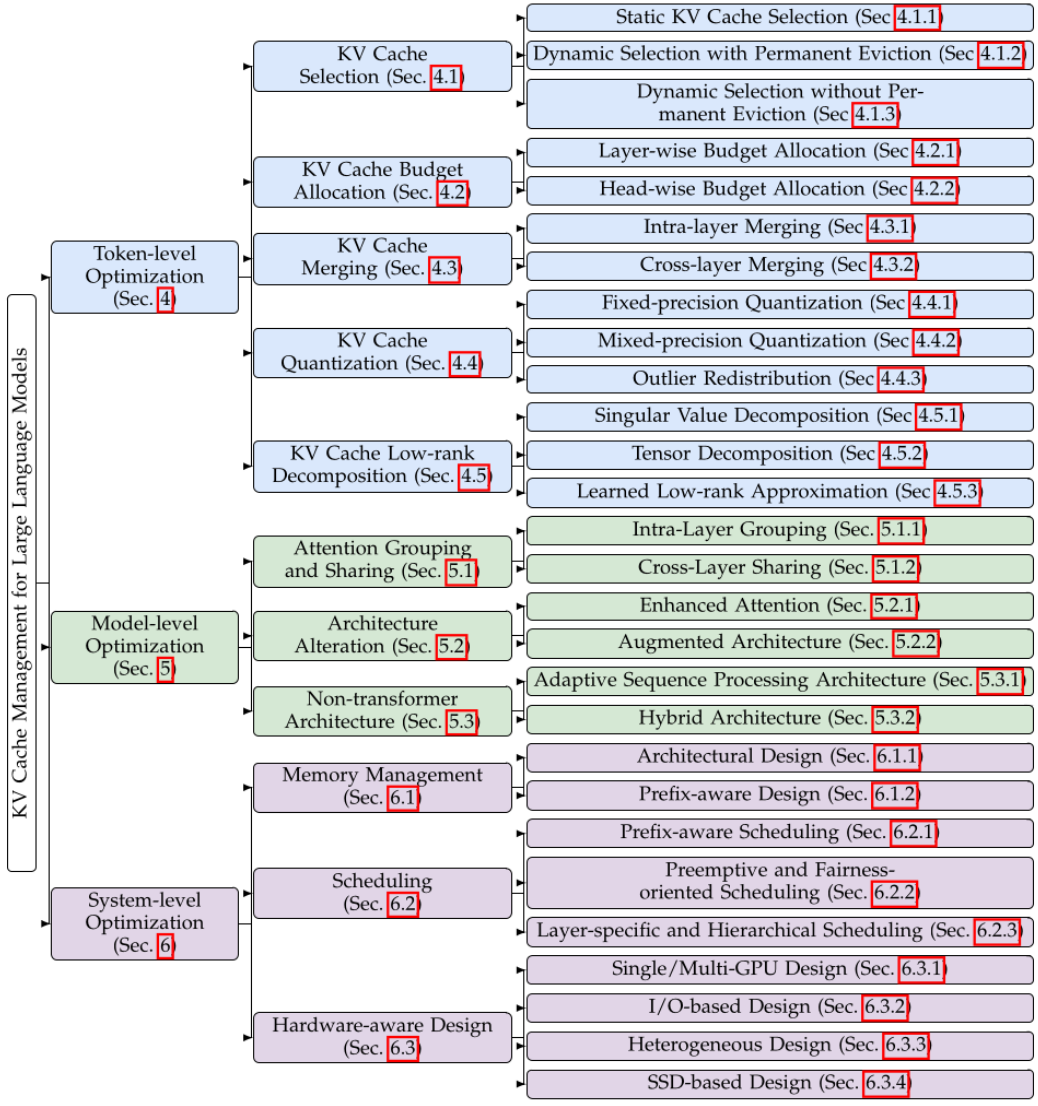


Fig. 2. Taxonomy of KV Cache Management for Large Language Models.
"A Survey on Large Language Model Acceleration based on KV Cache Management".
Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, Lei Chen
Accepted to *TMLR*, 2025.

Quantization

Inference in LLMs does not require high precision weights.

1. Pretrain with floating-point weights
2. Approximate weights with a discete set of integer values (many alternative mappings)
3. Optionally, finetune resulting model using quantise weights for inference and real-valued weights for backprop

Distillation

Large numbers of parameters are needed for learning, but not for inference.

1. Pretrain with large number of parameters
2. Train a smaller model to approximate the hidden representations of the large model
3. Use the small model at inference time