

Mixture of Experts LLMs

NLP: Fall 2025

Anoop Sarkar

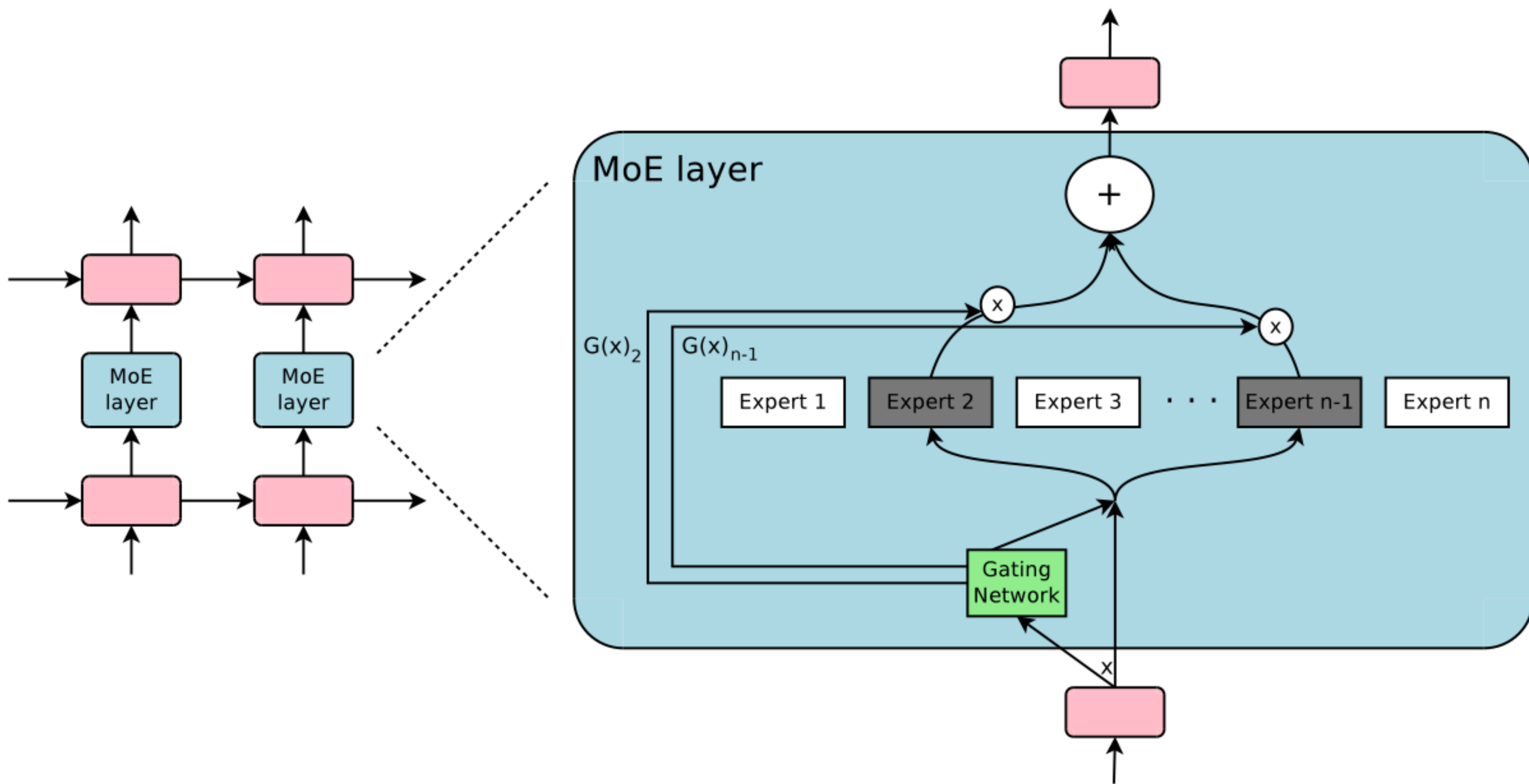
OUTRAGEOUSLY LARGE NEURAL NETWORKS: THE SPARSELY-GATED MIXTURE-OF-EXPERTS LAYER

Noam Shazeer¹, Azalia Mirhoseini^{*†1}, Krzysztof Maziarsz^{*2}, Andy Davis¹, Quoc Le¹, Geoffrey Hinton¹ and Jeff Dean¹

¹Google Brain, {noam,azalia,andydavis,qvl,geoffhinton,jeff}@google.com

²Jagiellonian University, Cracow, krzysztof.maziarz@student.uj.edu.pl

<https://arxiv.org/pdf/1701.06538>



Softmax gating

$$G_{\sigma}(x) = \textit{Softmax}(x \cdot W_g)$$

Noisy top-k gating

$$G(x) = \textit{Softmax}(\textit{KeepTopK}(H(x), k))$$

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

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

Gating is trained using simple backpropagation

(unlike Bengio et al 2015 who use RL to learn a boolean gating function)

Balancing Expert Utilization

Problem: large weights for the same few experts

- Define the **importance** of an expert relative to a batch of training examples to be the batchwise sum of the gate values for that expert.

- Let X be the number of training examples

- Let CV be the Coefficient of Variation $CV = \frac{\sigma}{\mu}$ Experiments show that gate values naturally diversify

$$Importance(X) = \sum_{x \in X} G(x)$$

$$L_{importance}(X) = w_{importance} \cdot CV(Importance(X))^2$$

Balancing Expert Utilization

Problem: experts receive different number of training examples

- Define the **load** of an expert to be the number of examples assigned to it.
- We want equal number of training examples across all experts
- Define $P(x, i)$ as the probability that $G(x)_i$ is nonzero.

$$P(x, i) = \Pr\left((x \cdot W_g)_i + \text{StandardNormal}() \cdot \text{Softplus}((x \cdot W_{\text{noise}})_i) > \text{kth_excluding}(H(x), k, i)\right)$$

$$\text{Load}(X)_i = \sum_{x \in X} P(x, i)$$

$$L_{\text{load}}(X) = w_{\text{load}} \cdot \text{CV}(\text{Load}(X))^2$$

GLaM: Efficient Scaling of Language Models with Mixture-of-Experts

**Nan Du^{*1} Yanping Huang^{*1} Andrew M. Dai^{*1} Simon Tong¹ Dmitry Lepikhin¹ Yuanzhong Xu¹
Maxim Krikun¹ Yanqi Zhou¹ Adams Wei Yu¹ Orhan Firat¹ Barret Zoph¹ Liam Fedus¹ Maarten Bosma¹
Zongwei Zhou¹ Tao Wang¹ Yu Emma Wang¹ Kellie Webster¹ Marie Pellat¹ Kevin Robinson¹
Kathleen Meier-Hellstern¹ Toju Duke¹ Lucas Dixon¹ Kun Zhang¹ Quoc V Le¹ Yonghui Wu¹
Zhifeng Chen¹ Claire Cui¹**

<https://arxiv.org/abs/2112.06905>

Mixture of Experts (MoE) for LLMs

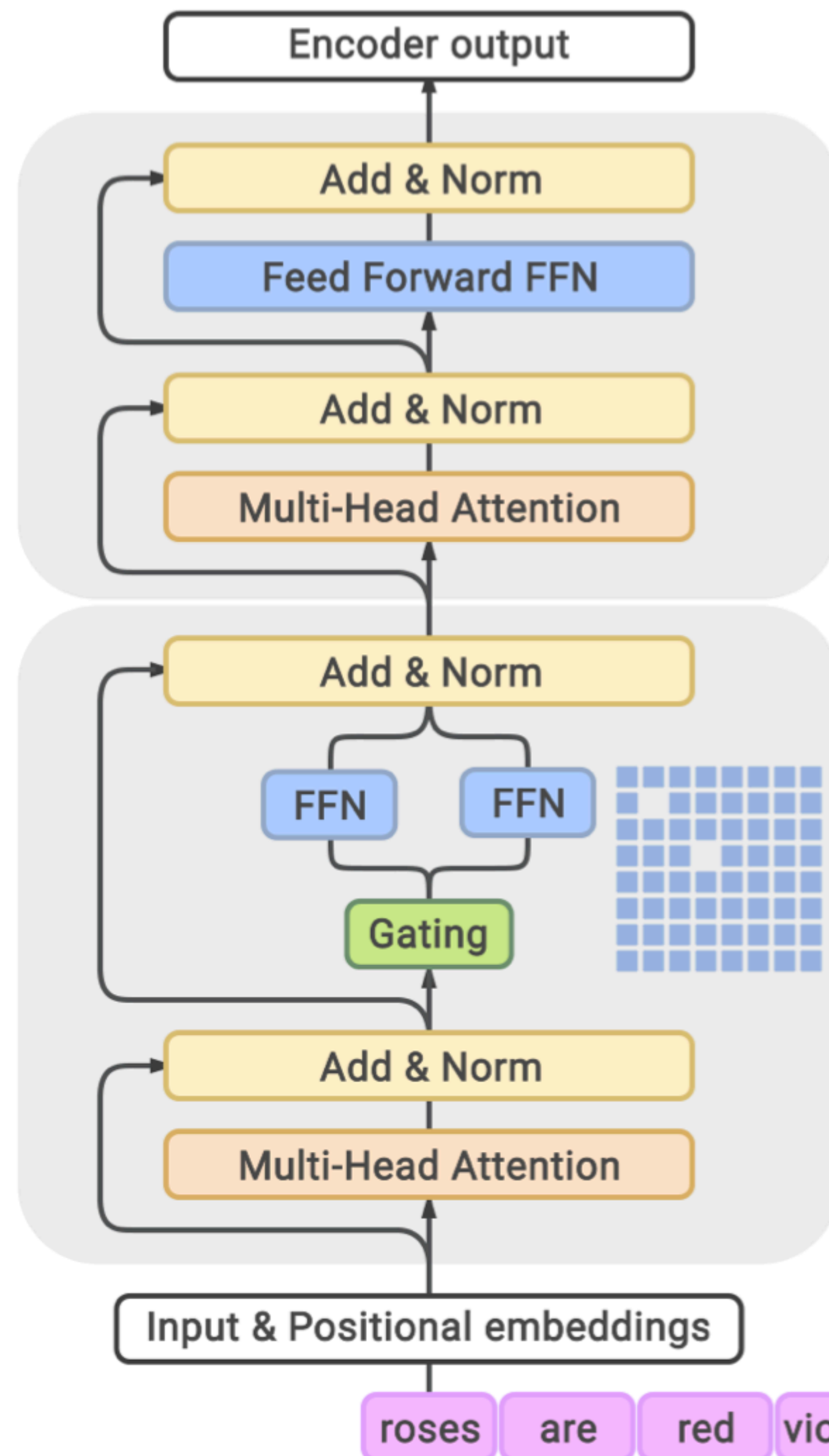
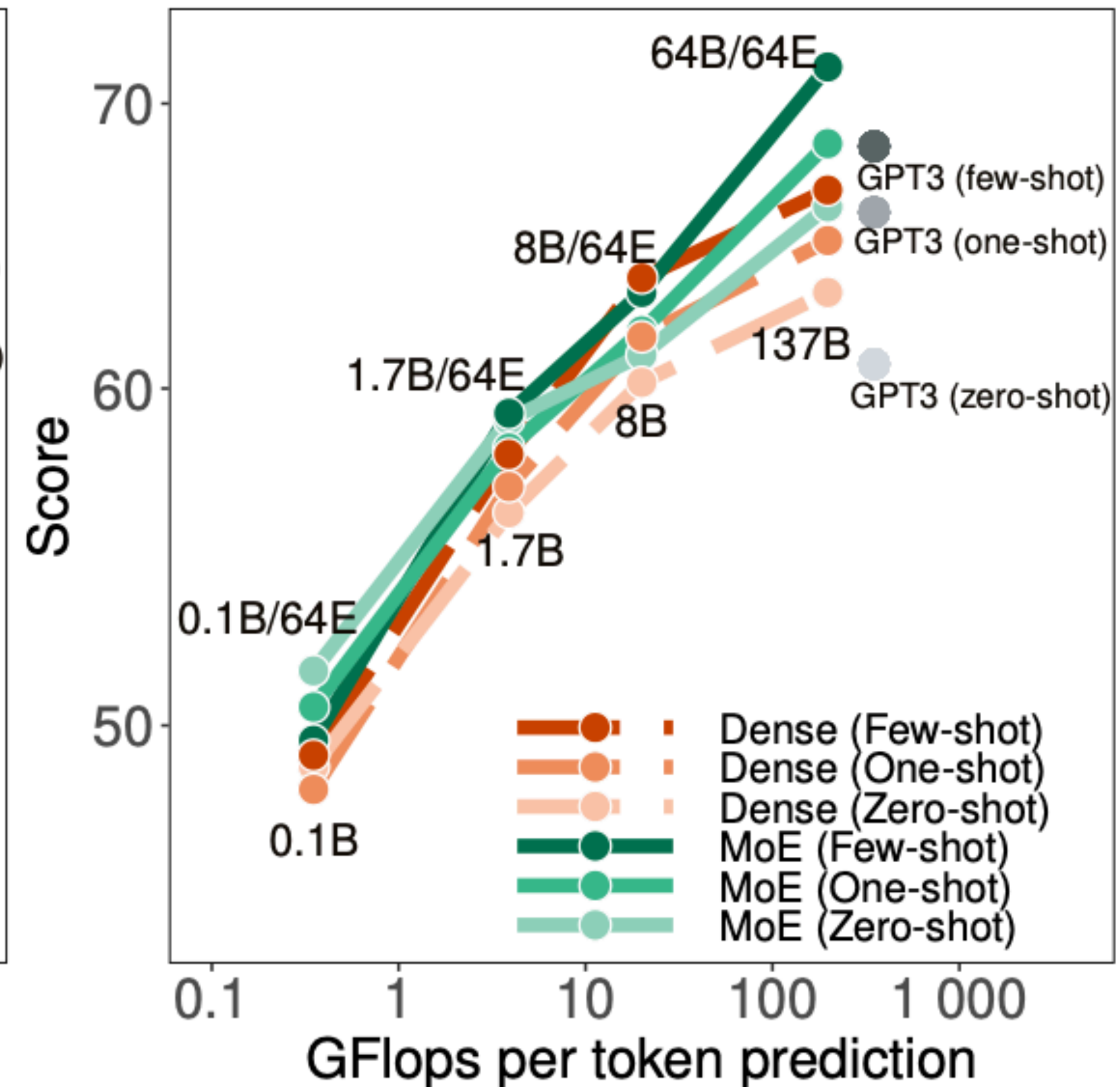
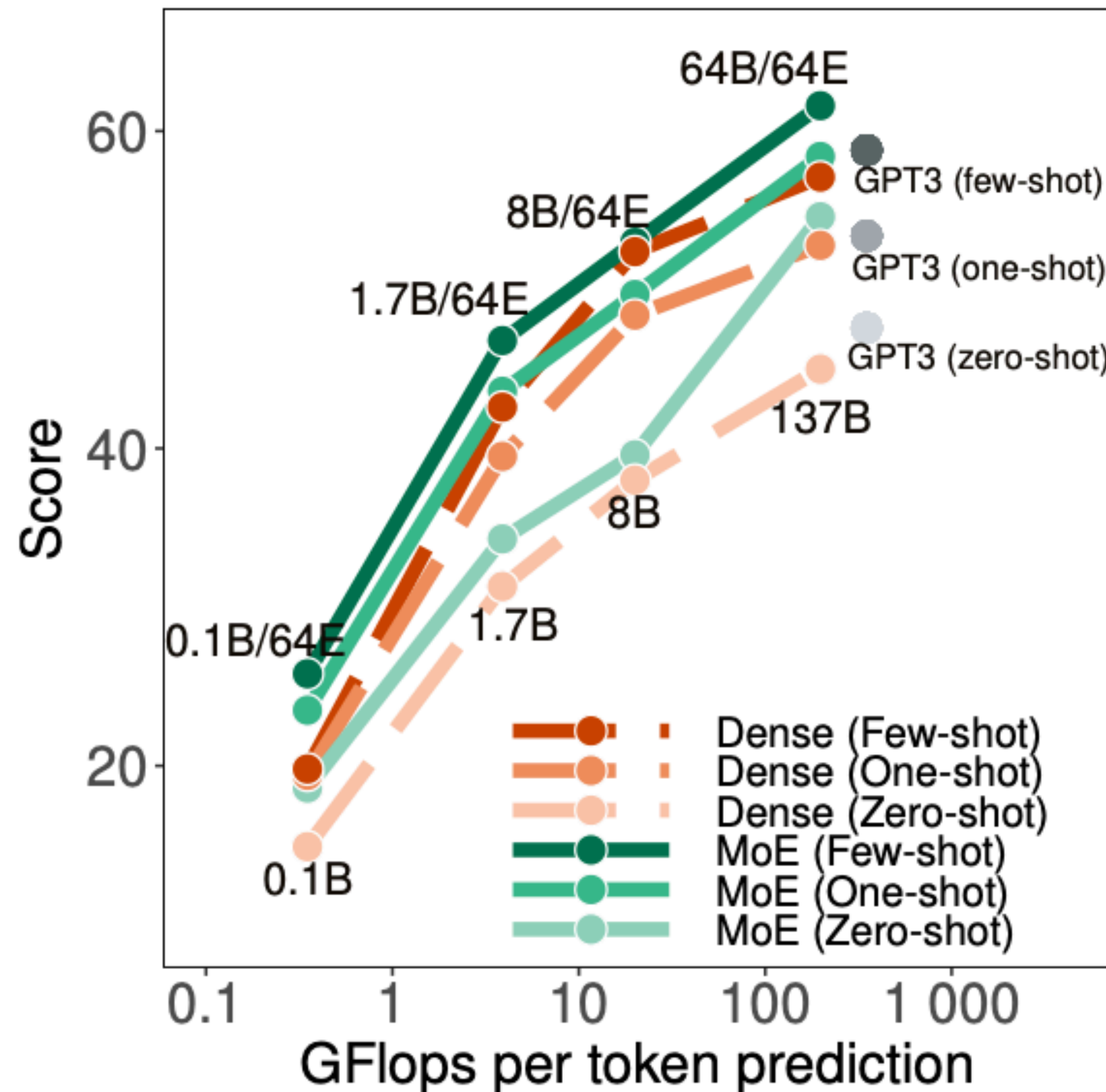


Figure 2. GLaM model architecture. Each MoE layer (the bottom block) is interleaved with a Transformer layer (the upper block). For each input token, *e.g.*, 'roses', the *Gating* module dynamically selects two most relevant experts out of 64, which is represented by the blue grid in the MoE layer. The weighted average of the outputs from these two experts will then be passed to the upper Transformer layer. For the next token in the input sequence, two different experts will be selected.

Mixture of Experts (MoE) for LLMs

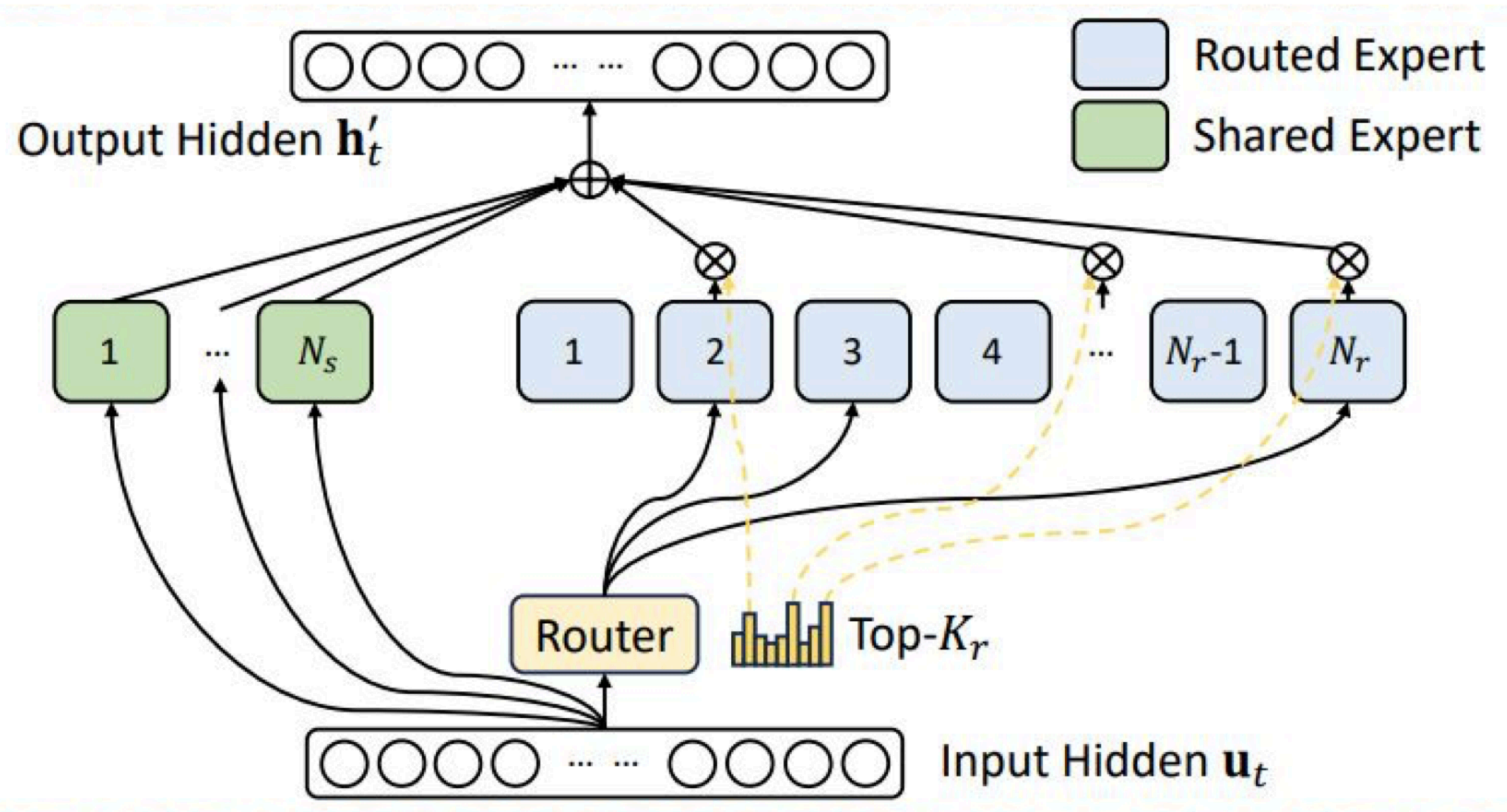
Better effective FLOPs per token prediction in causal LMs



Mixture of Experts

- GPT-4 architecture has not been confirmed but rumored to be MoE
- Mistral open sourced MoE models but did not release training details
- Deepseek V3 released many details about training large MoE models

Deepseek MoE



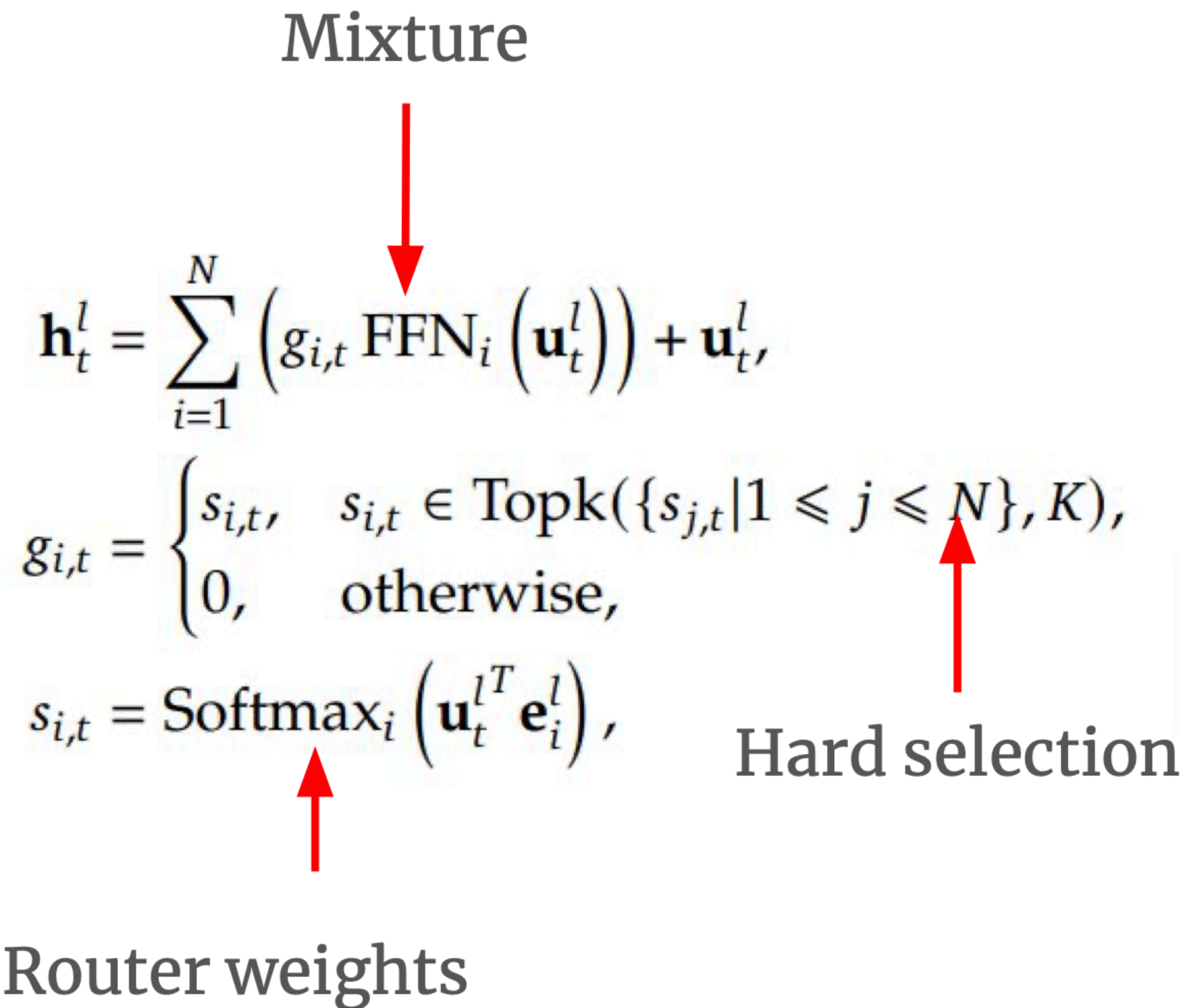
Deepseek MoE

Mixture

$$\mathbf{h}_t^l = \sum_{i=1}^N \left(g_{i,t} \text{FFN}_i \left(\mathbf{u}_t^l \right) \right) + \mathbf{u}_t^l,$$
$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \text{Topk}(\{s_{j,t} | 1 \leq j \leq N\}, K), \\ 0, & \text{otherwise,} \end{cases}$$
$$s_{i,t} = \text{Softmax}_i \left(\mathbf{u}_t^{lT} \mathbf{e}_i^l \right),$$

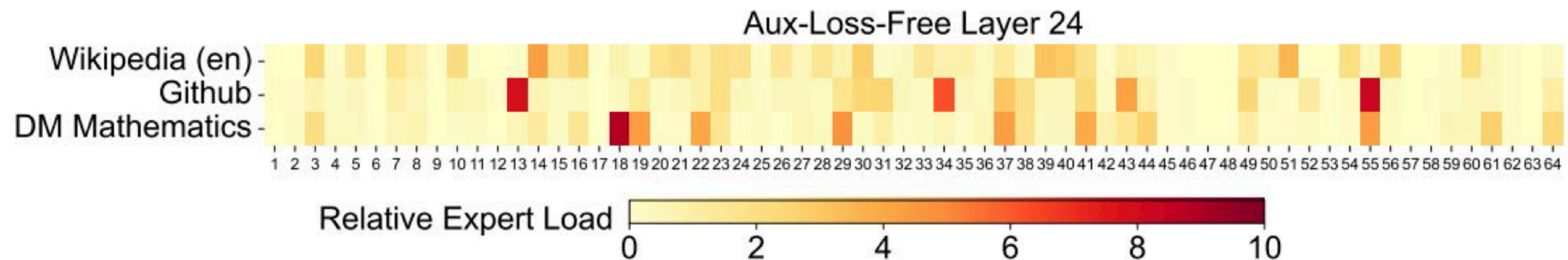
Router weights

Hard selection



Deepseek MoE

- #Total parameters = 671B
- #Activated parameters = 37B
- Different layers exhibit expert specialization



Extra Slides

PaLM: Scaling Language Modeling with Pathways

Aakanksha Chowdhery* Sharan Narang* Jacob Devlin*
Maarten Bosma Gaurav Mishra Adam Roberts Paul Barham
Hyung Won Chung Charles Sutton Sebastian Gehrmann Parker Schuh Kensen Shi
Sasha Tsvyashchenko Joshua Maynez Abhishek Rao† Parker Barnes Yi Tay
Noam Shazeer‡ Vinodkumar Prabhakaran Emily Reif Nan Du Ben Hutchinson
Reiner Pope James Bradbury Jacob Austin Michael Isard Guy Gur-Ari
Pengcheng Yin Toju Duke Anselm Levskaya Sanjay Ghemawat Sunipa Dev
Henryk Michalewski Xavier Garcia Vedant Misra Kevin Robinson Liam Fedus
Denny Zhou Daphne Ippolito David Luan‡ Hyeontaek Lim Barret Zoph
Alexander Spiridonov Ryan Sepassi David Dohan Shivani Agrawal Mark Omernick
Andrew M. Dai Thanumalayan Sankaranarayanan Pillai Marie Pellat Aitor Lewkowycz
Erica Moreira Rewon Child Oleksandr Polozov† Katherine Lee Zongwei Zhou
Xuezhi Wang Brennan Saeta Mark Diaz Orhan Firat Michele Catasta† Jason Wei
Kathy Meier-Hellstern Douglas Eck Jeff Dean Slav Petrov Noah Fiedel

PaLM: model architecture

- **SwiGLU Activation** – We use SwiGLU activations ($\text{Swish}(xW) \cdot xV$) for the MLP intermediate activations because they have been shown to significantly increase quality compared to standard ReLU, GeLU, or Swish activations ([Shazeer, 2020](#)). Note that this does require three matrix multiplications in the MLP rather than two, but [Shazeer \(2020\)](#) demonstrated an improvement in quality in compute-equivalent experiments (i.e., where the standard ReLU variant had proportionally larger dimensions).
- **Parallel Layers** – We use a “parallel” formulation in each Transformer block ([Wang & Komatsuzaki, 2021](#)), rather than the standard “serialized” formulation. Specifically, the standard formulation can be written as:

$$y = x + \text{MLP}(\text{LayerNorm}(x + \text{Attention}(\text{LayerNorm}(x))))$$

Whereas the parallel formulation can be written as:

$$y = x + \text{MLP}(\text{LayerNorm}(x)) + \text{Attention}(\text{LayerNorm}(x))$$

The parallel formulation results in roughly 15% faster training speed at large scales, since the MLP and Attention input matrix multiplications can be fused. Ablation experiments showed a small quality degradation at 8B scale but no quality degradation at 62B scale, so we extrapolated that the effect of parallel layers should be quality neutral at the 540B scale.

PaLM: model architecture

- **Multi-Query Attention** – The standard Transformer formulation uses k attention heads, where the input vector for each timestep is linearly projected into “query”, “key”, and “value” tensors of shape $[k, h]$, where h is the attention head size. Here, the key/value projections are shared for each head, i.e. “key” and “value” are projected to $[1, h]$, but “query” is still projected to shape $[k, h]$. We have found that this has a neutral effect on model quality and training speed ([Shazeer, 2019](#)), but results in a significant cost savings at autoregressive decoding time. This is because standard multi-headed attention has low efficiency on accelerator hardware during auto-regressive decoding, because the key/value tensors are not shared between examples, and only a single token is decoded at a time.
- **RoPE Embeddings** – We use RoPE embeddings ([Su et al., 2021](#)) rather than absolute or relative position embeddings, since RoPE embeddings have been shown to have better performance on long sequence lengths.
- **Shared Input-Output Embeddings** – We share the input and output embedding matrices, which is done frequently (but not universally) in past work.

PaLM: model architecture

- **No Biases** – No biases were used in any of the dense kernels or layer norms. We found this to result in increased training stability for large models.
- **Vocabulary** – We use a SentencePiece ([Kudo & Richardson, 2018a](#)) vocabulary with 256k tokens, which was chosen to support the large number of languages in the training corpus without excess tokenization. The vocabulary was generated from the training data, which we found improves training efficiency. The vocabulary is completely lossless and reversible, which means that whitespace is completely preserved in the vocabulary (especially important for code) and out-of-vocabulary Unicode characters are split into UTF-8 bytes, with a vocabulary token for each byte. Numbers are always split into individual digit tokens (e.g., “123.5 → 1 2 3 . 5”).

PaLM: model hyperparameters

Model	Layers	# of Heads	d_{model}	# of Parameters (in billions)	Batch Size
PaLM 8B	32	16	4096	8.63	256 \rightarrow 512
PaLM 62B	64	32	8192	62.50	512 \rightarrow 1024
PaLM 540B	118	48	18432	540.35	512 \rightarrow 1024 \rightarrow 2048

Table 1: Model architecture details. We list the number of layers, d_{model} , the number of attention heads and attention head size. The feed-forward size d_{ff} is always $4 \times d_{\text{model}}$ and attention head size is always 256.

PaLM: training data

Total dataset size = 780 billion tokens	
Data source	Proportion of data
Social media conversations (multilingual)	50%
Filtered webpages (multilingual)	27%
Books (English)	13%
GitHub (code)	5%
Wikipedia (multilingual)	4%
News (English)	1%

Table 2: Proportion of data from each source in the training dataset. The multilingual corpus contains text from over 100 languages, with the distribution given in Appendix Table [29](#).

PaLM: Pathways data parallelism

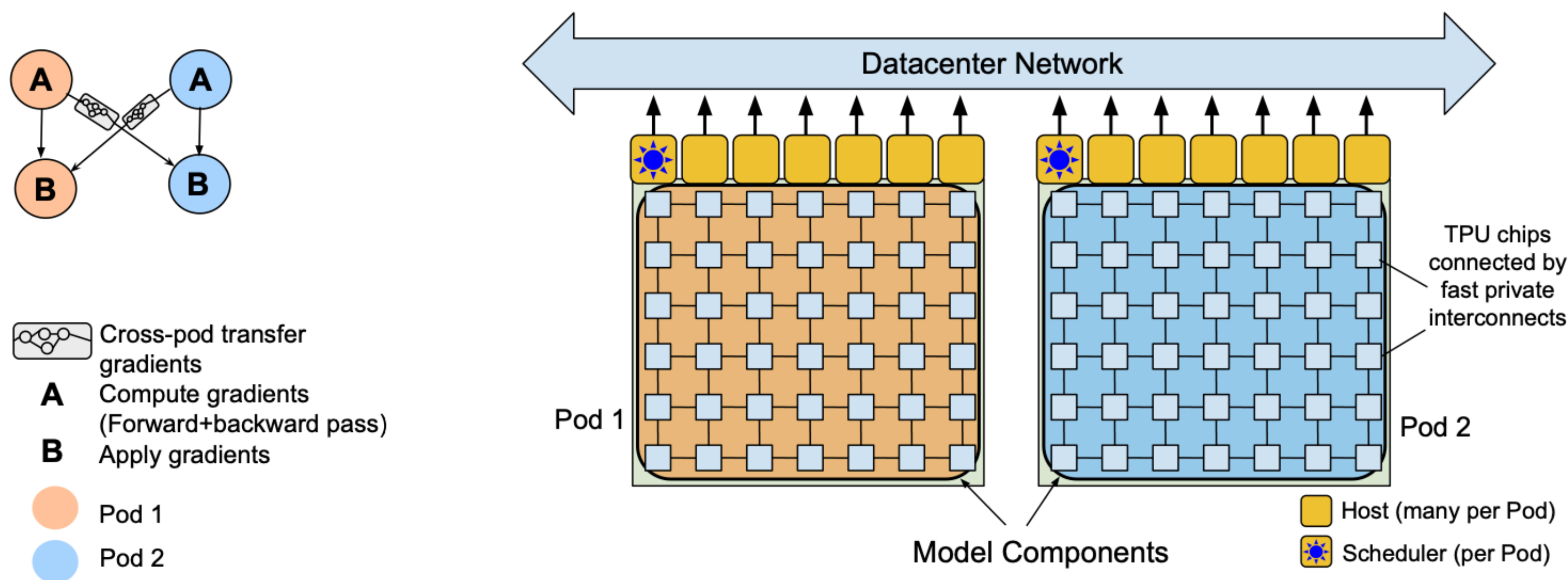


Figure 2: The Pathways system (Barham et al., 2022) scales training across two TPU v4 pods using two-way data parallelism at the pod level.