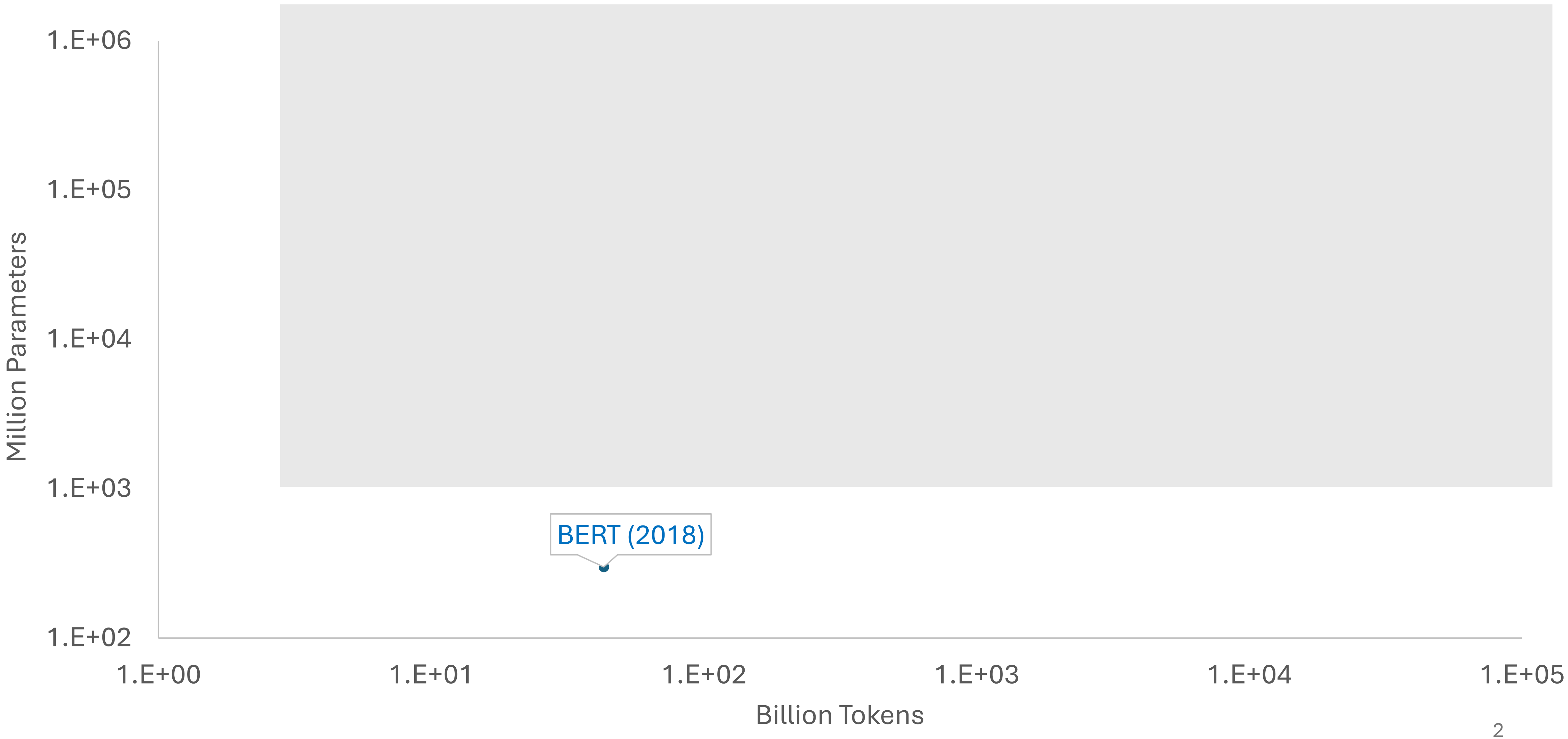
Mixture-of-Experts in the Era of LLMs

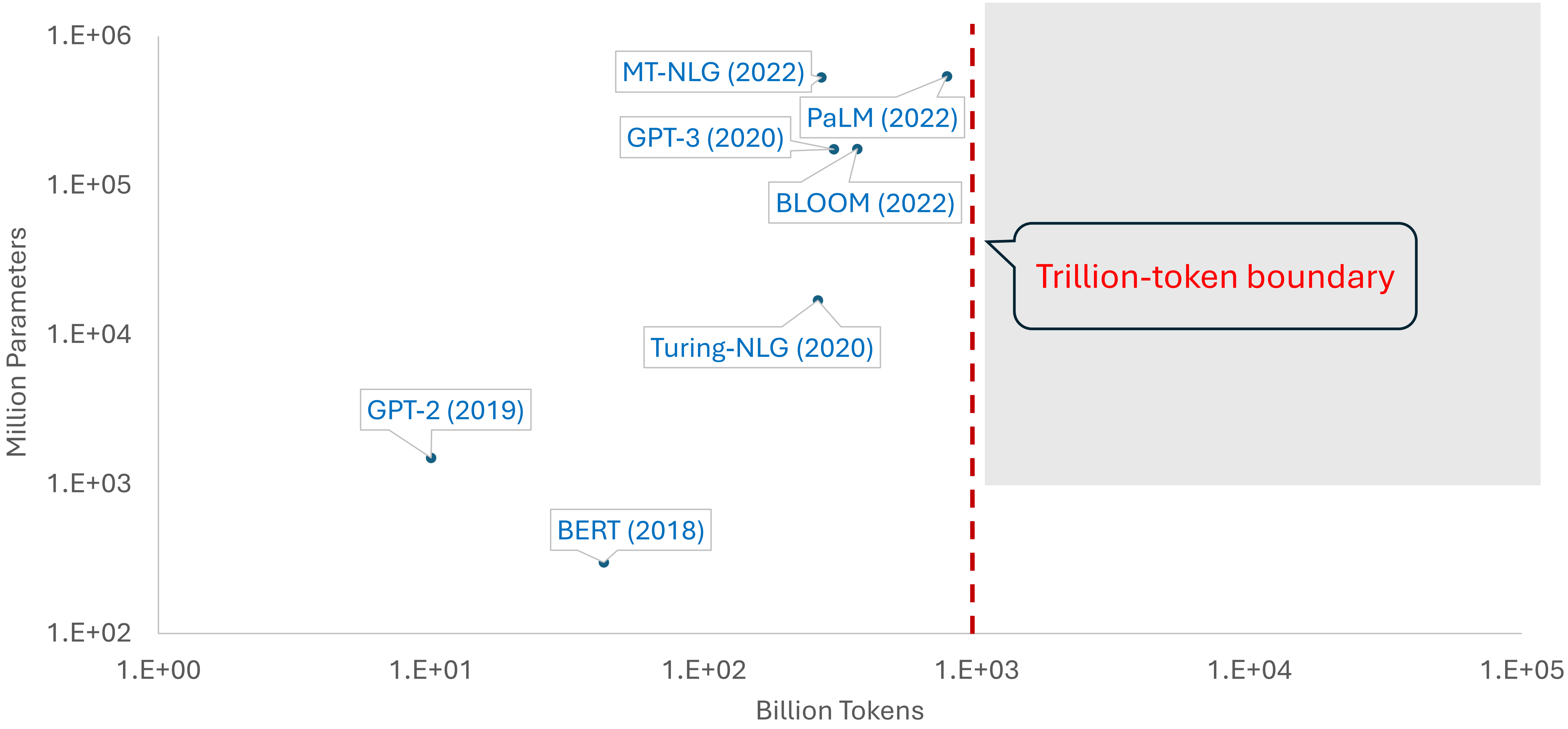
Minjia Zhang
University of Illinois at Urbana-Champaign
minjiaz@illinois.edu

ILLINOIS
Computer Science
GRAINGER COLLEGE OF ENGINEERING

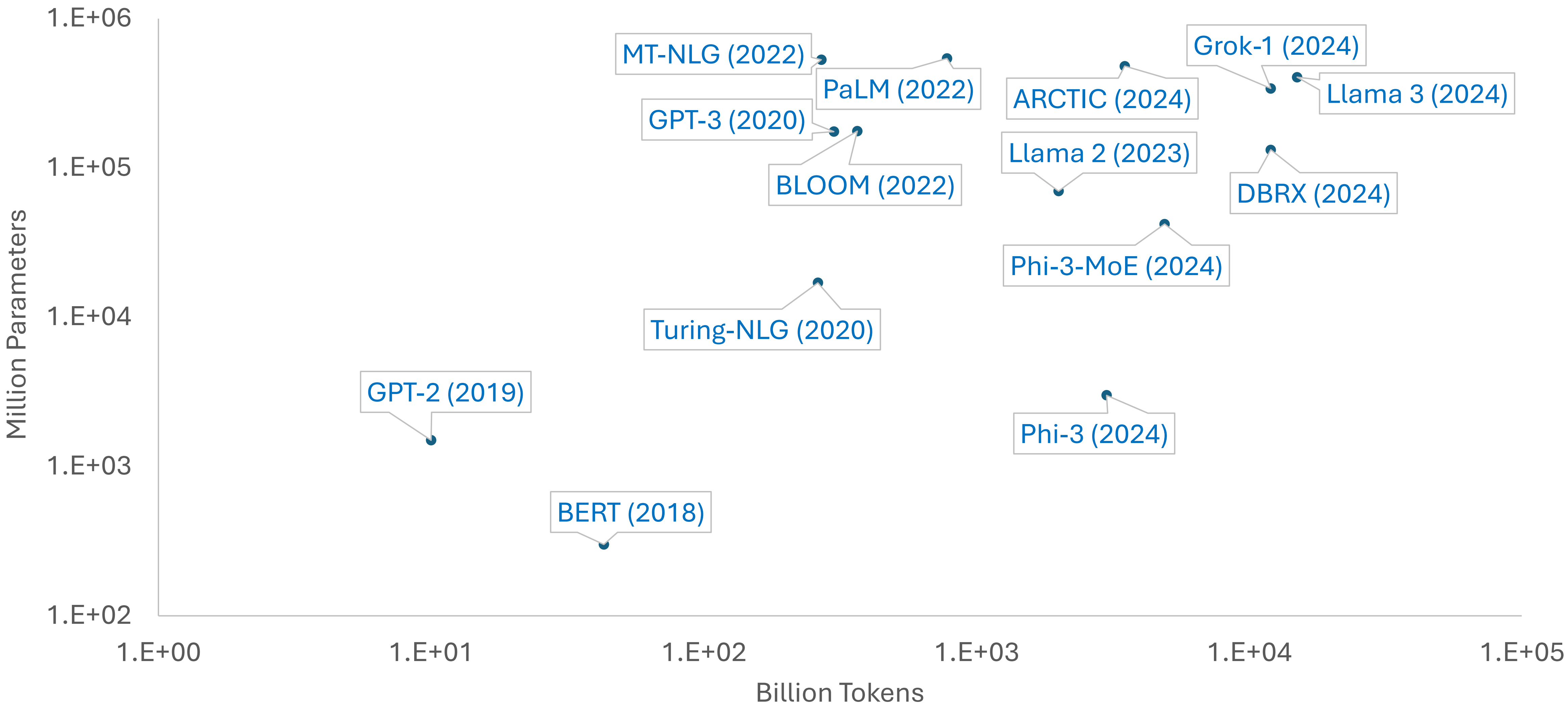
Scaling drives SOTA Deep Learning



Scaling drives SOTA Deep Learning



Scaling drives SOTA Deep Learning



AI Scale is Limited By Compute

- Compute is the primary challenge of training massive models
- Ambitious model at scale and time to train

Model	Model Size	Hardware	Days to Train
BLOOM	176B	384 A100 GPUs	115 days
OPT	175B	992 A100 GPU	56 days
MT-NLG	530B	2200 A100 GPU	60 days
PaLM	540B	6144 TPU v4	57 days

Next jump in scale:

- Next-generation hardware
- Significant investment in GPUs

Next AI Scale?

- Can we achieve next generation model quality on current generation of hardware?

- From a computation perspective sparse Mixture-of-Experts provides a promising path
  - Scale at sub-linear cost

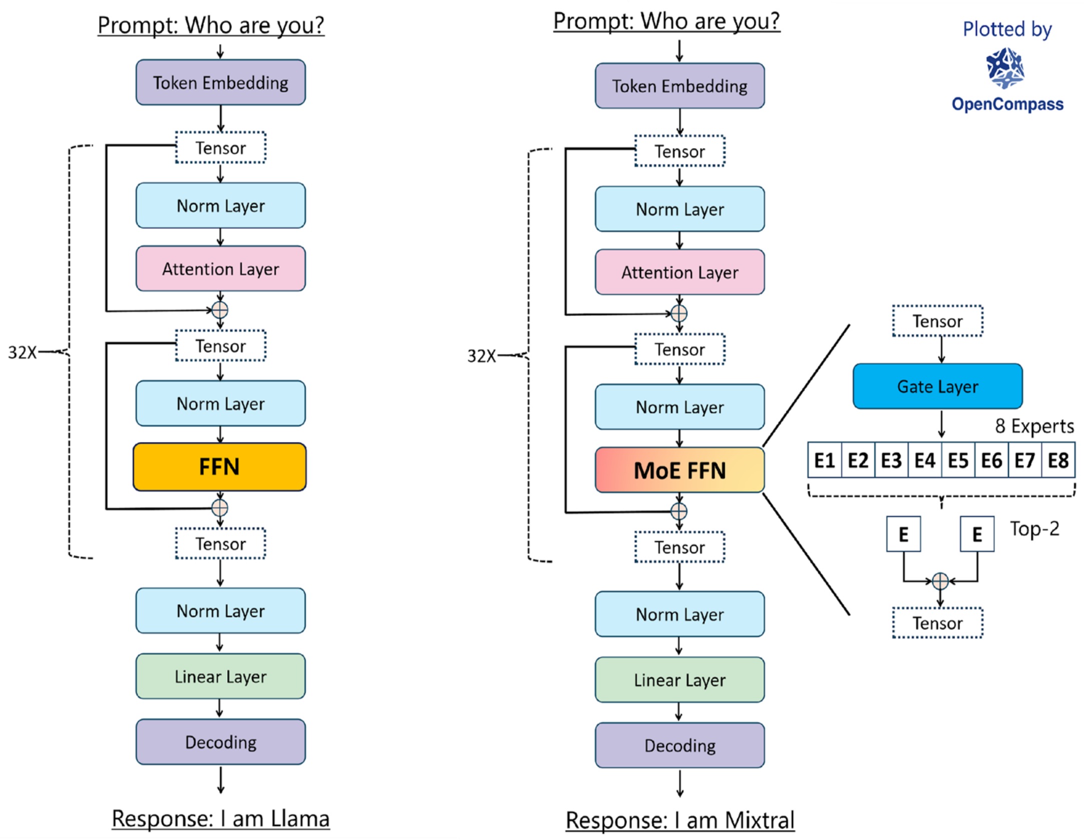
Recap: MoE Models are Sparse and Need Less Compute

Dense Models:

- All parameters are used in forward and backward paths
- Increasing model capacity needs more computation
- **Larger model size → Higher compute requirements (FLOPs)**

Sparse MoE models

- Sparse utilization of subset of parameters based on input
- Same computation is needed regardless of the model size
- **Larger model size → Similar/Same Compute requirements**

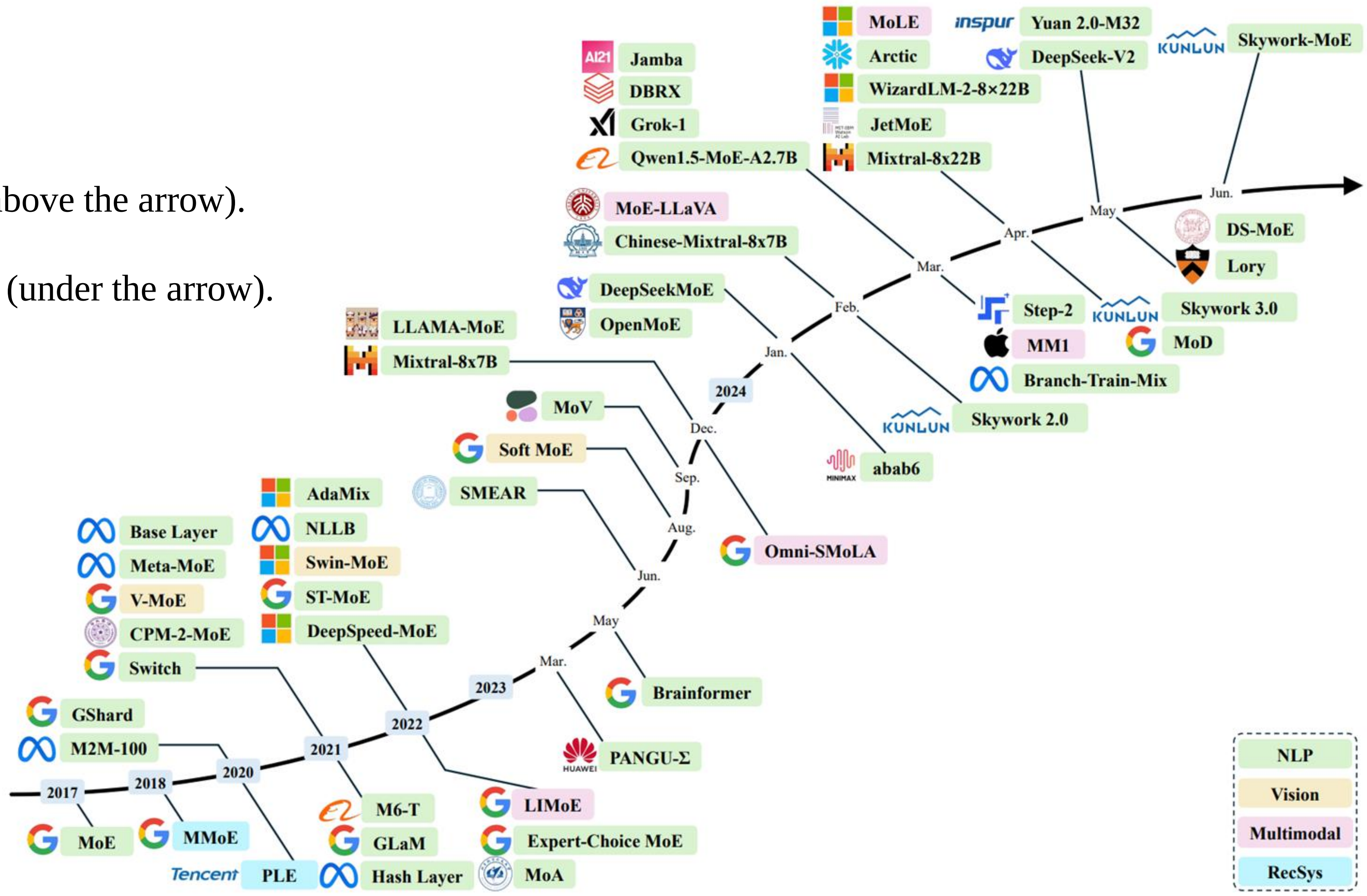


Mixture of Experts (MoE): Overview

- MoE models have been around for a while..
- Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer
  - Harder to scale, instability during training, and inefficient training
- GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding
  - 600B models beating 96-layer dense models, 10x training speedup, generic sharding framework (Tensorflow XLA)
  - Less stability with larger models, full precision training
- Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity
  - More efficient training
    - Top-1 gating instead of top-2/top-k, Better initialization conditions, Mixed precision training: FP32 gating (instead of FP16), Stable training with larger models
  - SOTA results on language understanding task

MoE: Road Map

- Open-source (above the arrow).
- Private models (under the arrow).



MoE Design

What should we care when designing a MoE?

Network types	FFN, Attention
Fine-grained experts	32 experts/128 experts/…
Shared experts	Isolated experts
Activation Function	ReLU/GEGLU/SwiGLU
MoE frequency	Every two layer/Each layer/…
Training auxiliary loss	Auxiliary loss/Z-loss/…

Fine-Grained and Shared Experts

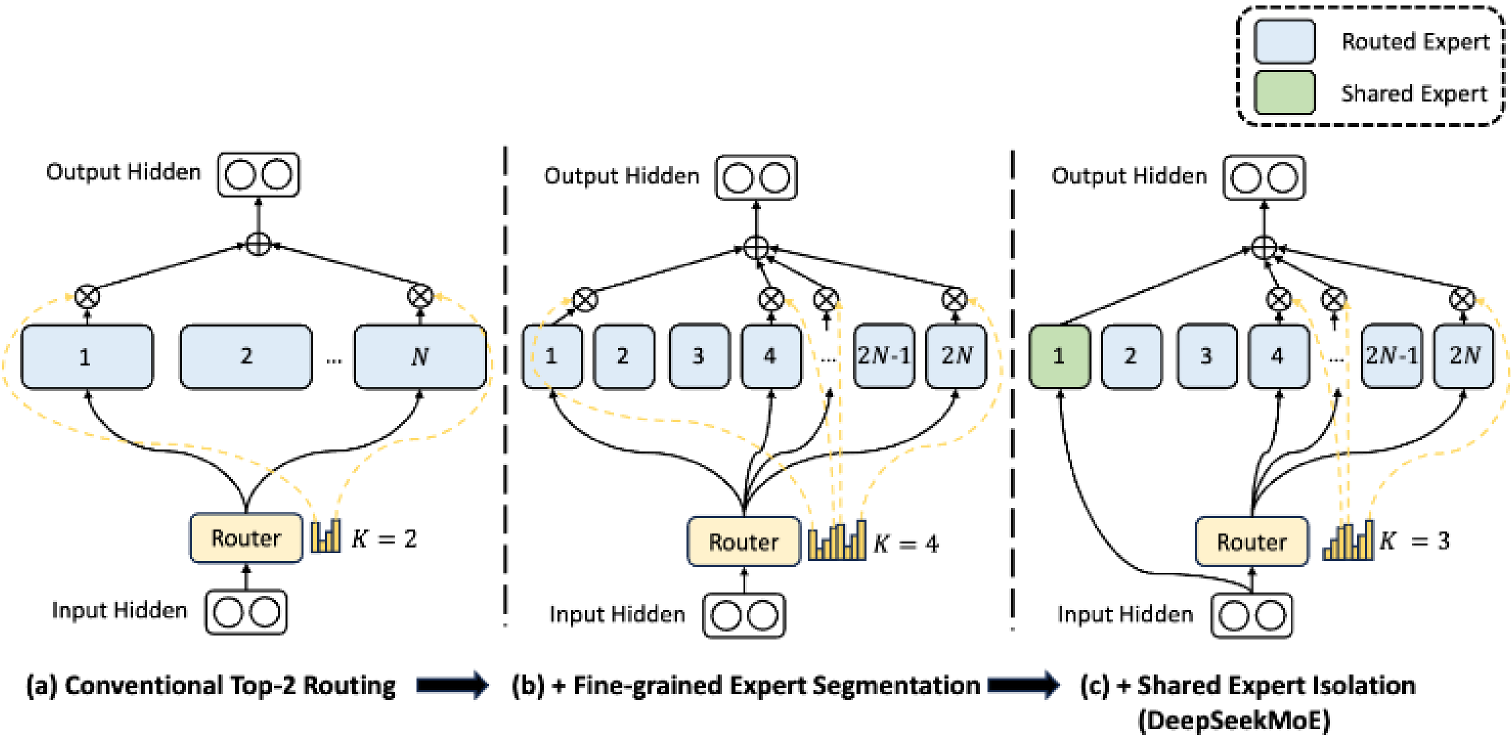
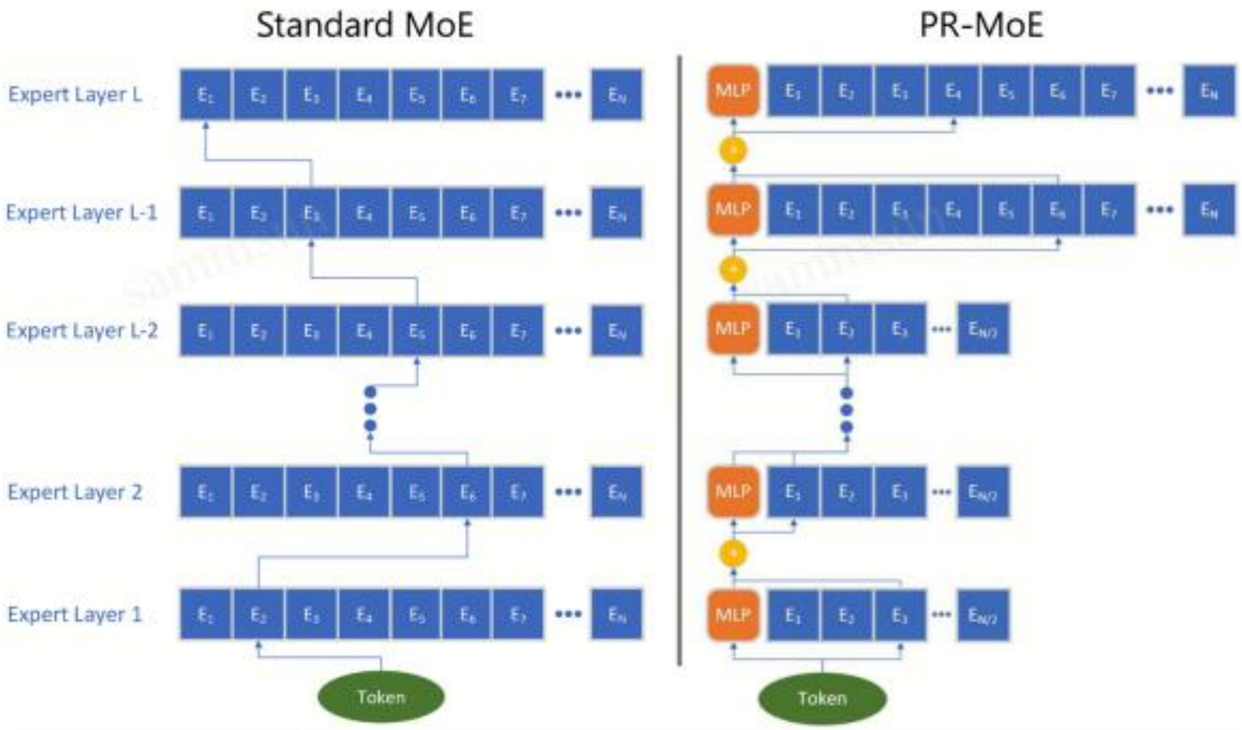


Figure 2 | Illustration of DeepSeekMoE. Subfigure (a) showcases an MoE layer with the conventional top-2 routing strategy. Subfigure (b) illustrates the fine-grained expert segmentation strategy. Subsequently, subfigure (c) demonstrates the integration of the shared expert isolation strategy, constituting the complete DeepSeekMoE architecture. It is noteworthy that across these three architectures, the number of expert parameters and computational costs remain constant.

Pyramid Design of Experts



- Utilizes more experts in the last few layers as compared to previous layers
- Positive results compared with the baseline MoE

Model (num. params)	LAMBADA	PIQA	BoolQ	RACE-h	TriviaQA	WebQs
350M+MoE-128 (13B)	62.70	**74.59**	**60.46**	35.60	**16.58**	5.17
350M+PR-MoE-32/64 (**4B**)	**63.65**	73.99	59.88	**35.69**	16.30	**4.73**
1.3B+MoE-128 (52B)	69.84	76.71	64.92	**38.09**	**31.29**	7.19
1.3B+PR-MoE-64/128 (**31B**)	**70.60**	**77.75**	**67.16**	**38.09**	28.86	**7.73**

DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale

MoE Experts Design

Reference	Models	Expert Count (Activ./Total)	d_{model}	d_{ffn}	d_{expert}	#L	#H	d_{head}	Placement Frequency	Activation Function	Share Expert Count
GShard [86] (2020)	600B	2/2048	1024	8192	d_{ffn}	36	16	128	1/2	ReLU	0
	200B	2/2048	1024	8192	d_{ffn}	12	16	128	1/2	ReLU	0
	150B	2/512	1024	8192	d_{ffn}	36	16	128	1/2	ReLU	0
	37B	2/128	1024	8192	d_{ffn}	36	16	128	1/2	ReLU	0
Switch [49] (2021)	7B	1/128	768	2048	d_{ffn}	12	12	64	1/2	GEGLU	0
	26B	1/128	1024	2816	d_{ffn}	24	16	64	1/2	GEGLU	0
	395B	1/64	4096	10240	d_{ffn}	24	64	64	1/2	GEGLU	0
	1571B	1/2048	2080	6144	d_{ffn}	15	32	64	1	ReLU	0
GLaM [44] (2021)	0.1B/1.9B	2/64	768	3072	d_{ffn}	12	12	64	1/2	GEGLU	0
	1.7B/27B	2/64	2048	8192	d_{ffn}	24	16	128	1/2	GEGLU	0
	8B/143B	2/64	4096	16384	d_{ffn}	32	32	128	1/2	GEGLU	0
	64B/1.2T	2/64	8192	32768	d_{ffn}	64	128	128	1/2	GEGLU	0
DeepSpeed-MoE [121] (2022)	350M/13B	2/128	1024	$4d_{model}$	d_{ffn}	24	16	64	1/2	GeLU	0
	1.3B/52B	2/128	2048	$4d_{model}$	d_{ffn}	24	16	128	1/2	GeLU	0
	PR-350M/4B	2/32-2/64	1024	$4d_{model}$	d_{ffn}	24	16	64	1/2, 10L-32E, 2L-64E	GeLU	1
	PR-1.3B/31B	2/64-2/128	2048	$4d_{model}$	d_{ffn}	24	16	128	1/2, 10L-64E, 2L-128E	GeLU	1
ST-MoE [197] (2022)	0.8B/4.1B	2/32	1024	2816	d_{ffn}	27	16	64	1/4, add extra FFN	GEGLU	0
	32B/269B	2/64	5120	20480	d_{ffn}	27	64	128	1/4, add extra FFN	GEGLU	0
Mixtral [74] (2023)	13B/47B	2/8	4096	14336	d_{ffn}	32	32	128	1	SwiGLU	0
	39B/141B	2/8	6144	16384	d_{ffn}	56	48	128	1	SwiGLU	0
LLAMA-MoE [149] (2023)	3.0B/6.7B	2/16	4096	11008	688	32	32	128	1	SwiGLU	0
	3.5B/6.7B	4/16	4096	11008	688	32	32	128	1	SwiGLU	0
	3.5B/6.7B	2/8	4096	11008	1376	32	32	128	1	SwiGLU	0
DeepSeekMoE [30] (2024)	0.24B/1.89B	8/64	1280	-	$\frac{1}{4}d_{ffn}$	9	10	128	1	SwiGLU	1
	2.8B/16.4B	8/66	2048	10944	1408	28	16	128	1, except 1st layer	SwiGLU	2
	22B/145B	16/132	4096	-	$\frac{1}{8}d_{ffn}$	62	32	128	1, except 1st layer	SwiGLU	4
OpenMoE [172] (2024)	339M/650M	2/16	768	3072	d_{ffn}	12	12	64	1/4	SwiGLU	1
	2.6B/8.7B	2/32	2048	8192	d_{ffn}	24	24	128	1/6	SwiGLU	1
	6.8B/34B	2/32	3072	12288	d_{ffn}	32	24	128	1/4	SwiGLU	1
Qwen1.5-MoE [151] (2024)	2.7B/14.3B	8/64	2048	5632	1408	24	16	128	1	SwiGLU	4
DBRX [34] (2024)	36B/132B	4/16	6144	10752	d_{ffn}	40	48	128	1	SwiGLU	0
Jamba [94] (2024)	12B/52B	2/16	4096	14336	d_{ffn}	32	32	128	1/2, 1:7 Attention:Mamba	SwiGLU	0
Skywork-MoE [154] (2024)	22B/146B	2/16	4608	12288	d_{ffn}	52	36	128	1	SwiGLU	0
Yuan 2.0-M32 [166] (2024)	3.7B/40B	2/32	2048	8192	d_{ffn}	24	16	256	1	SwiGLU	0

Key points:

- Most recent models place MoE each layer.
- Some of recent models apply shared experts.

A Survey on Mixture of Experts

Auxiliary Loss

Training with different auxiliary loss:

Reference	Auxiliary Loss	Coefficient
Shazeer et al.[135], V-MoE[128]	$L_{importance} + L_{load}$	$w_{importance} = 0.1$, $w_{load} = 0.1$
GShard[86], **Switch-T**[49], GLaM[44], Mixtral-8x7B[74], DBRX[34], Jamba[94], DeepSeekMoE[30], DeepSeek-V2[36], Skywork-MoE[154]	L_{aux}	$w_{aux} = 0.01$
ST-MoE[197], OpenMoE[172], MoA[182], JetMoE [139]	$L_{aux} + L_z$	$w_{aux} = 0.01$, $w_z = 0.001$
Mod-Squad[21], Moduleformer[140], DS-MoE[117]	L_{MI}	$w_{MI} = 0.001$

Importance loss: encourages all experts to have equal importance

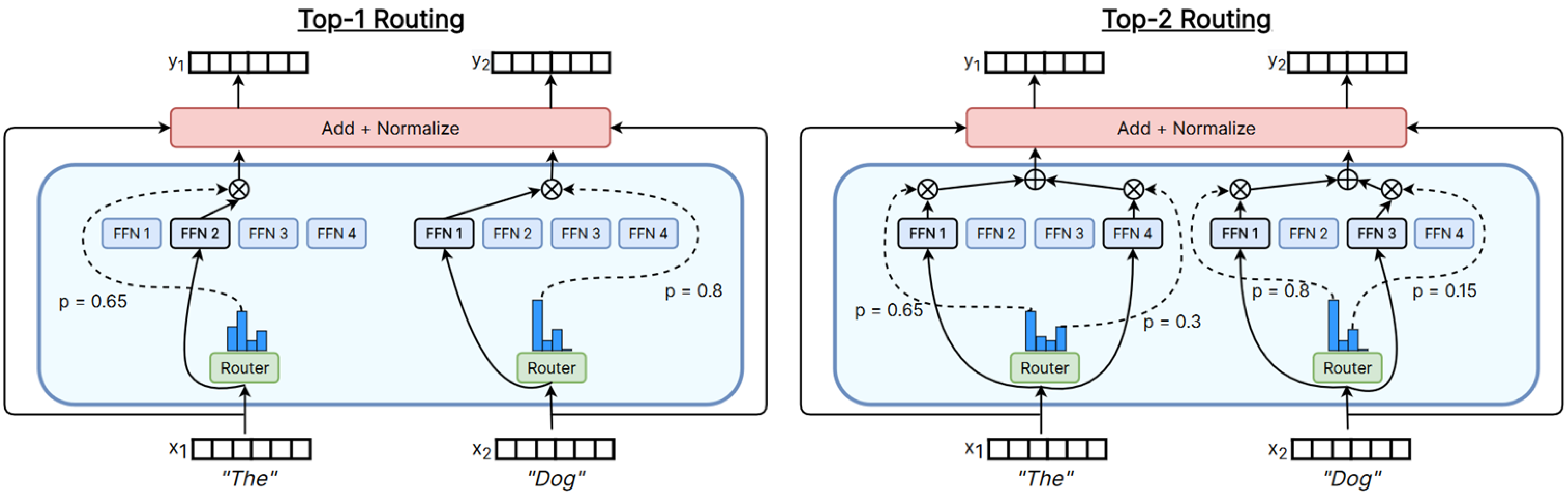
Load loss: ensure balanced loads

Auxiliary loss: mitigating load imbalance losses

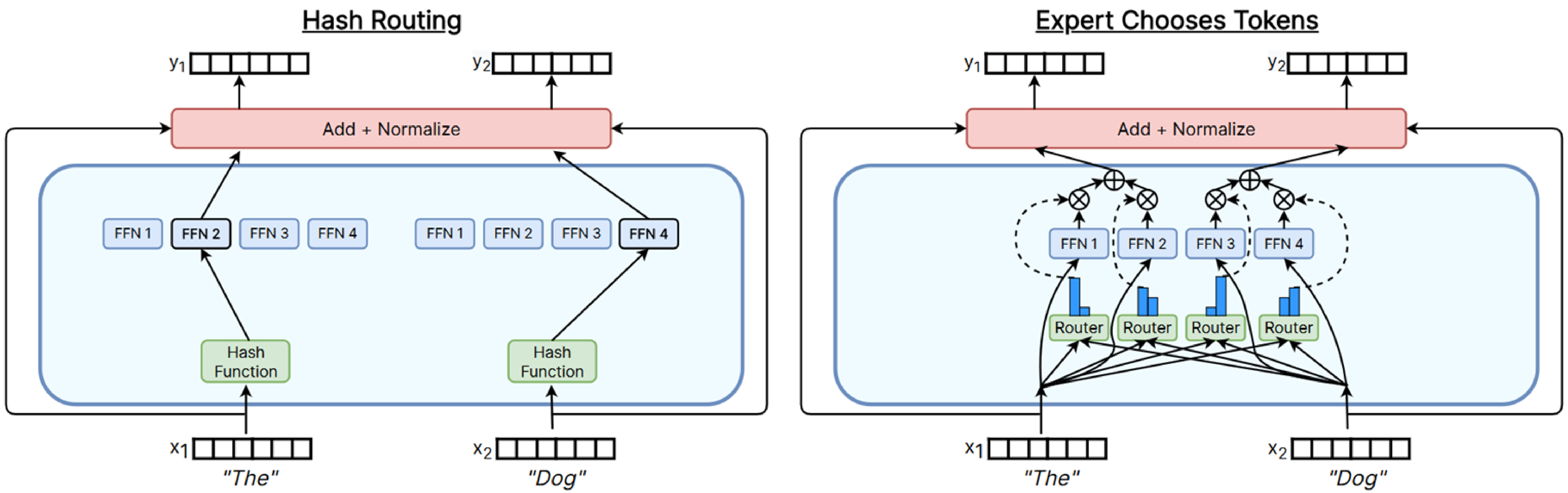
Z-loss: improving training stability by penalizing large logits

MI-loss: mutual information (MI) between experts and tasks to build task-expert alignment

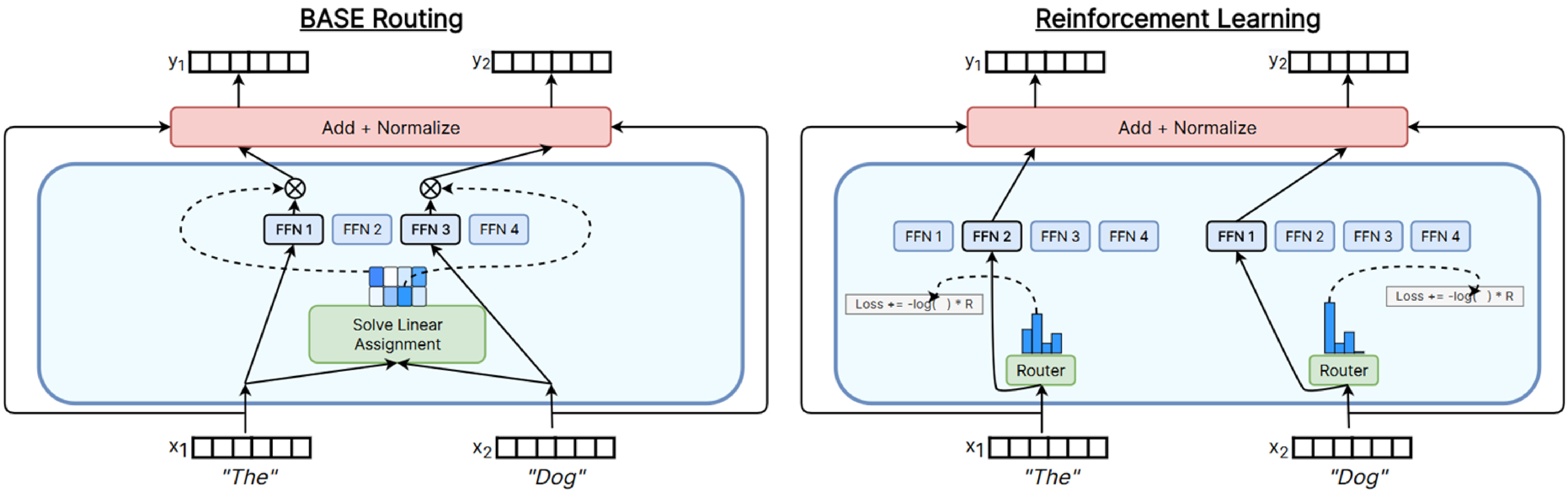
Routing Algorithms



Routing Algorithms



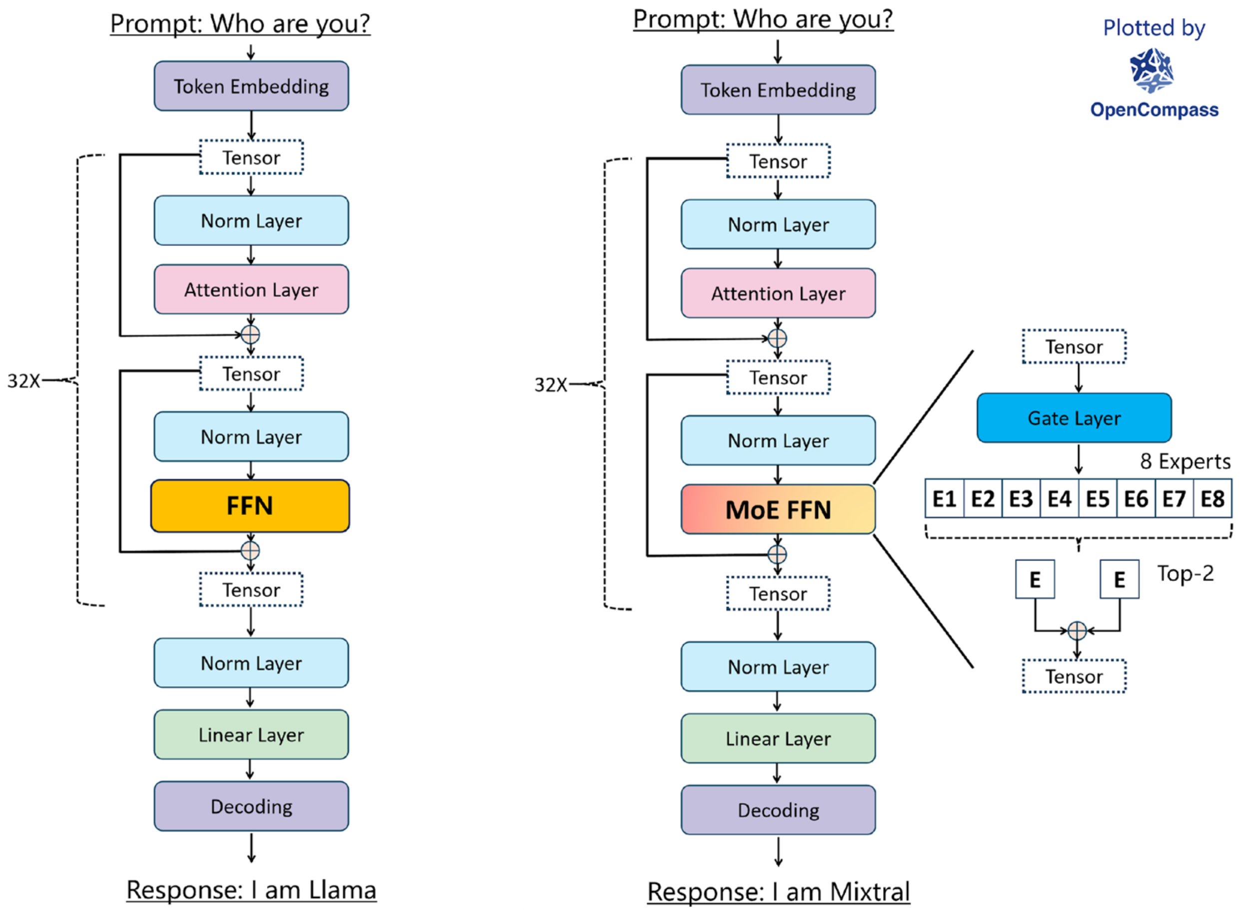
Routing Algorithms



Training MoE – Mixtral-MoE

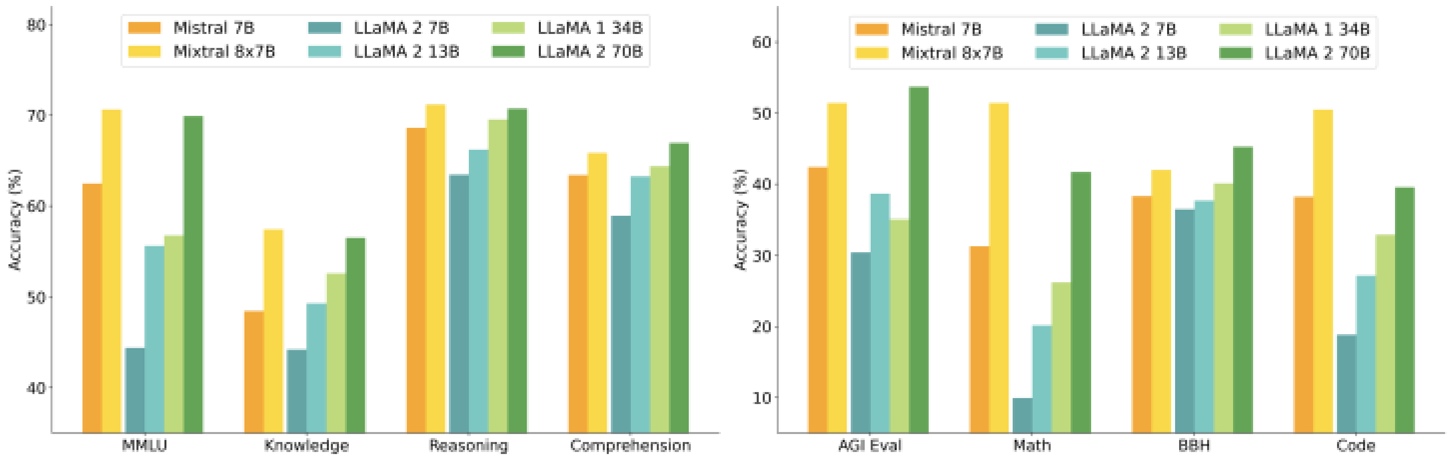
Example 1: Mixtral 8x22B (7B) (April, 2024)

Total 141B parameters, 39B activate parameters, (8 experts and 2 experts are selected)



Training MoE – Mixtral-MoE

Example 1：Mixtral 8x7B (22B)



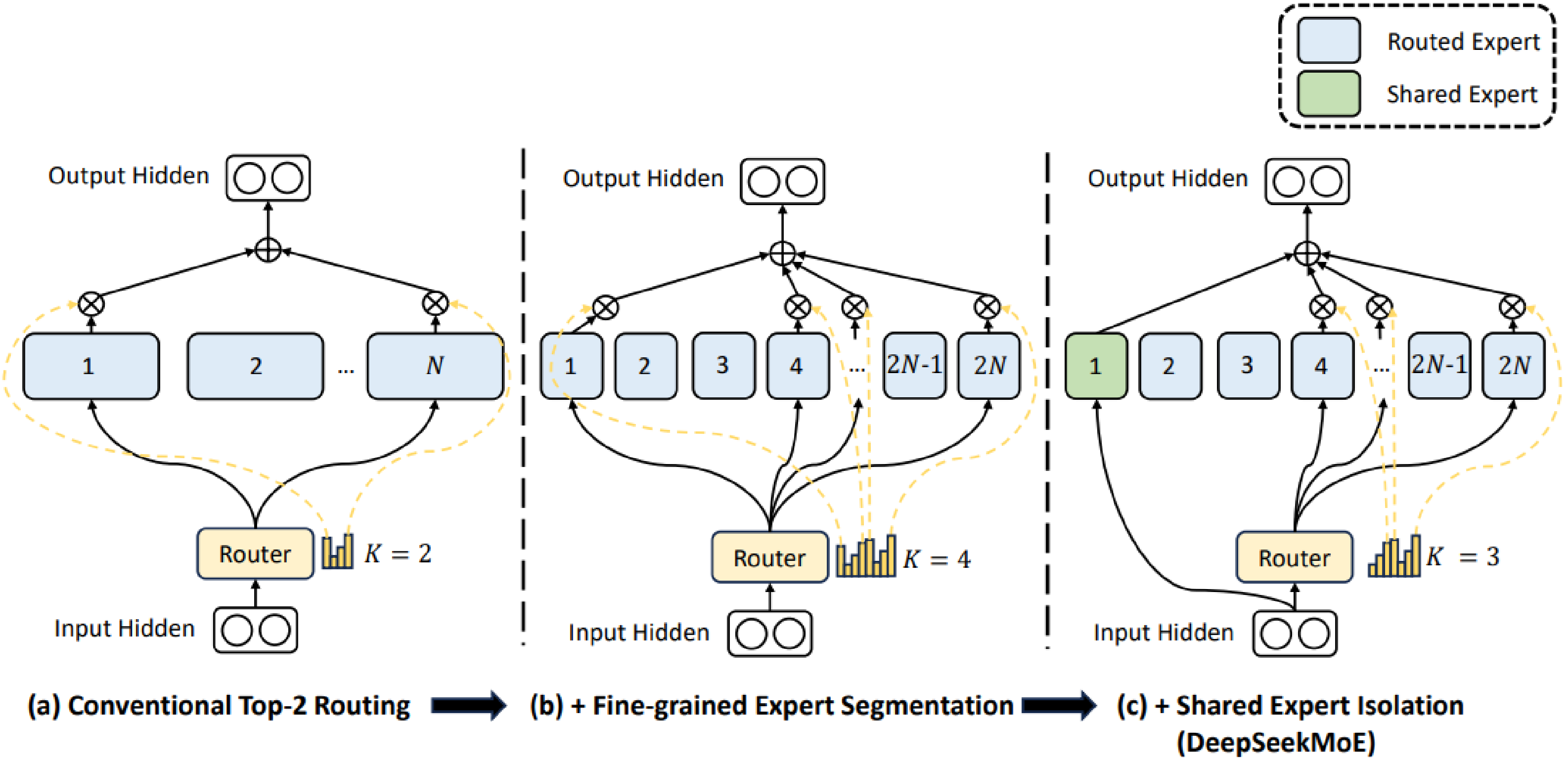
Model	Active Params	MMLU	HellaS	WinoG	PIQA	Arc-e	Arc-c	NQ	TriQA	HumanE	MBPP	Math	GSM8K
LLaMA 2 7B	7B	44.4%	77.1%	69.5%	77.9%	68.7%	43.2%	17.5%	56.6%	11.6%	26.1%	3.9%	16.0%
LLaMA 2 13B	13B	55.6%	80.7%	72.9%	80.8%	75.2%	48.8%	16.7%	64.0%	18.9%	35.4%	6.0%	34.3%
LLaMA 1 33B	33B	56.8%	83.7%	76.2%	82.2%	79.6%	54.4%	24.1%	68.5%	25.0%	40.9%	8.4%	44.1%
LLaMA 2 70B	70B	69.9%	**85.4%**	**80.4%**	82.6%	79.9%	56.5%	25.4%	**73.0%**	29.3%	49.8%	13.8%	69.6%
Mistral 7B	7B	62.5%	81.0%	74.2%	82.2%	80.5%	54.9%	23.2%	62.5%	26.2%	50.2%	12.7%	50.0%
Mixtral 8x7B	13B	**70.6%**	84.4%	77.2%	**83.6%**	**83.1%**	**59.7%**	**30.6%**	71.5%	**40.2%**	**60.7%**	**28.4%**	**74.4%**

Table 2: Comparison of Mixtral with Llama. Mixtral outperforms or matches Llama 2 70B performance on almost all popular benchmarks while using 5x fewer active parameters during inference.

Training MoE - DeepSeek

Example 2：Deepseek-MoE

Deepseek-MoE 16B, total 16.4B parameters, 2.8B activate parameters.
Each MoE layer consists of 2 shared experts and 64 routed experts (select 6 experts).

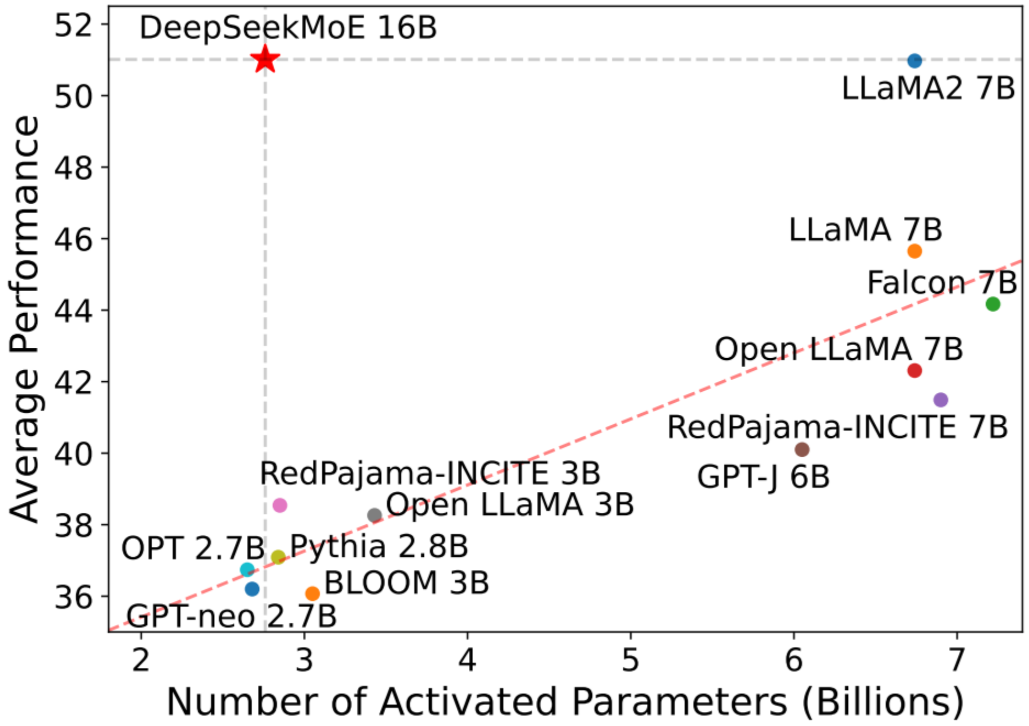


Key points:

- Fine-grained experts
- Shared experts

Training MoE - DeepSeek

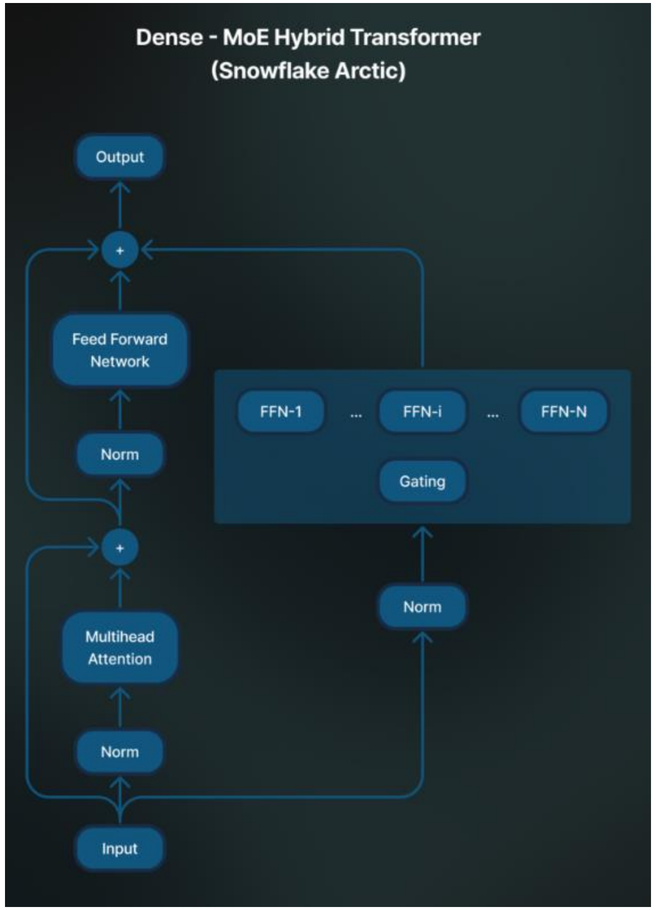
Example 2： Deepseek-MoE



Metric	# Shot	DeepSeek 7B (Dense)	DeepSeekMoE 16B
# Total Params	N/A	6.9B	16.4B
# Activated Params	N/A	6.9B	2.8B
FLOPs per 4K Tokens	N/A	183.5T	74.4T
# Training Tokens	N/A	2T	2T
Pile (BPB)	N/A	0.75	**0.74**
HellaSwag (Acc.)	0-shot	75.4	**77.1**
PIQA (Acc.)	0-shot	79.2	**80.2**
ARC-easy (Acc.)	0-shot	**67.9**	**68.1**
ARC-challenge (Acc.)	0-shot	48.1	**49.8**
RACE-middle (Acc.)	5-shot	**63.2**	61.9
RACE-high (Acc.)	5-shot	**46.5**	**46.4**
DROP (EM)	1-shot	**34.9**	32.9
GSM8K (EM)	8-shot	17.4	**18.8**
MATH (EM)	4-shot	3.3	**4.3**
HumanEval (Pass@1)	0-shot	26.2	**26.8**
MBPP (Pass@1)	3-shot	**39.0**	**39.2**
TriviaQA (EM)	5-shot	59.7	**64.8**
NaturalQuestions (EM)	5-shot	22.2	**25.5**
MMLU (Acc.)	5-shot	**48.2**	45.0
WinoGrande (Acc.)	0-shot	**70.5**	**70.2**
CLUEWSC (EM)	5-shot	**73.1**	72.1
CEval (Acc.)	5-shot	**45.0**	40.6
CMMLU (Acc.)	5-shot	**47.2**	42.5
CHID (Acc.)	0-shot	**89.3**	**89.4**

Training MoE - Arctic

Example 3： Arctic (Dense and Sparse)



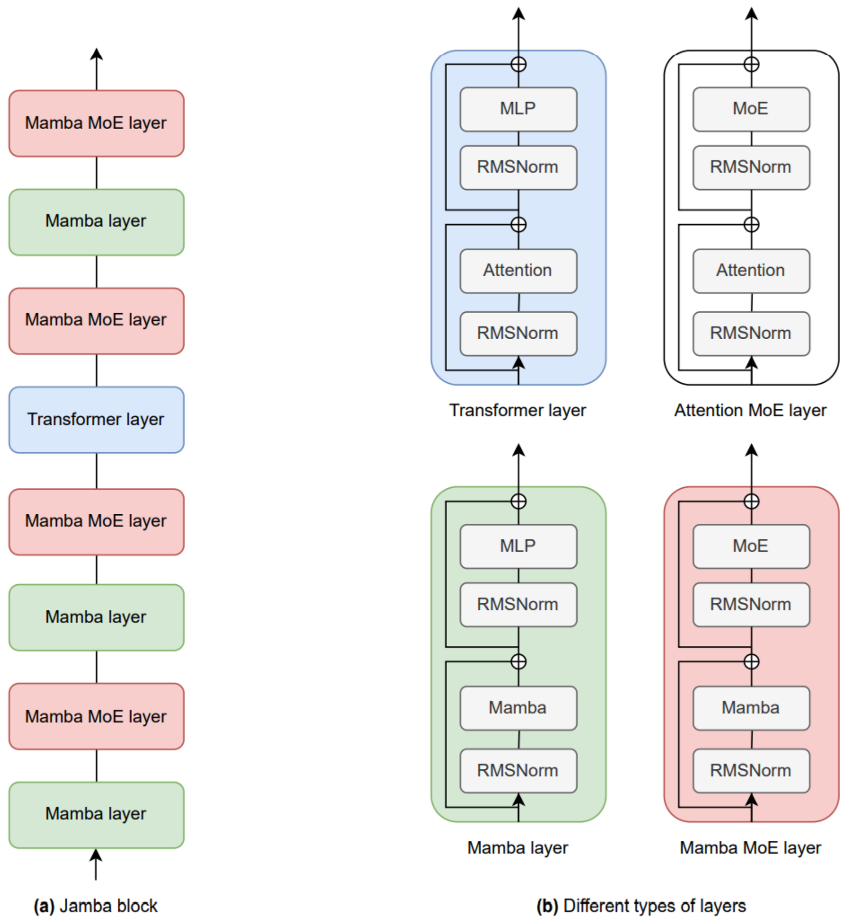
Arctic uses a unique Dense-MoE Hybrid transformer architecture.

- It combines a 10B dense transformer model with a residual 128×3.66B MoE MLP.
- 480B total and 17B active parameters chosen using a top-2 gating.

	Snowflake Arctic	DBRX	Llama 3 8B	Llama 2 70B	Llama 3 70B	Mixtral 8x7B	Mixtral 8x22B
Active Parameters	17B	36B	8B	70B	70B	13B	44B
ENTERPRISE							
SQL Generation (*Spider*)	79.0	76.3	69.9	62.8	80.2	71.3	79.2
Coding (*HumanEval+, MBPP+*)	64.3	61.0	59.2	33.7	71.9	48.1	69.9
Instruction Following (*IFEval*)	57.4	54.8	42.7	-	43.6	52.2	61.5
ACADEMIC							
Math (*GSM8K*)	74.2	73.5	75.4	52.6	91.4	63.2	84.2
Common Sense (*Avg of 11 metrics*)	73.1	74.8	68.5	72.1	72.6	74.1	75.6
World Knowledge (*MMLU*)	67.3	73.3	65.7	68.6	79.8	70.4	77.5

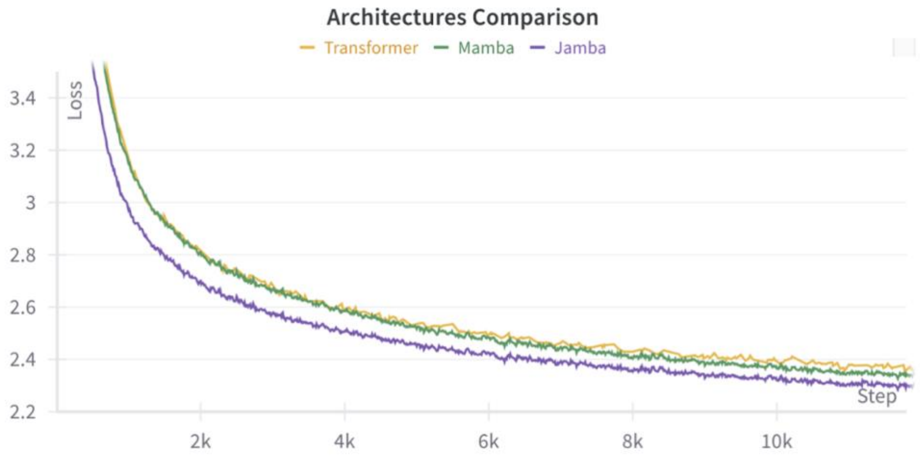
Training MoE - Jamba

Example 4： Jamba (Hybrid architecture)



Jamba is a hybrid decoder architecture that mixes Transformer layers with Mamba layers, in addition to a mixture-of-experts (MoE) module.

	Available params	Active params
LLAMA-2	6.7B	6.7B
Mistral	7.2B	7.2B
Mixtral	46.7B	12.9B
Jamba	52B	12B



Jamba: A Hybrid Transformer-Mamba Language Model

Unified Scaling Law

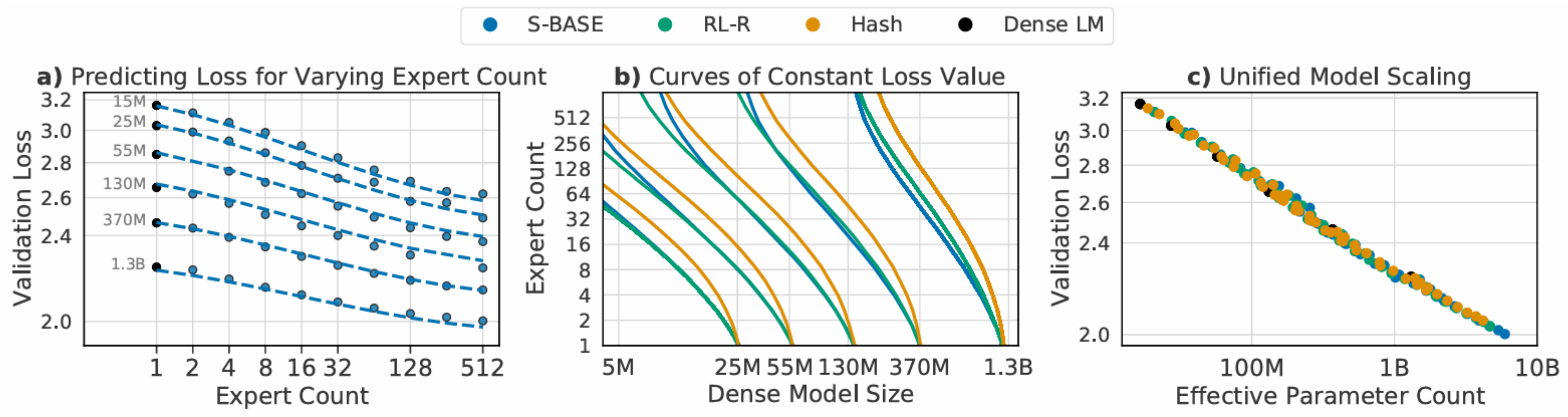
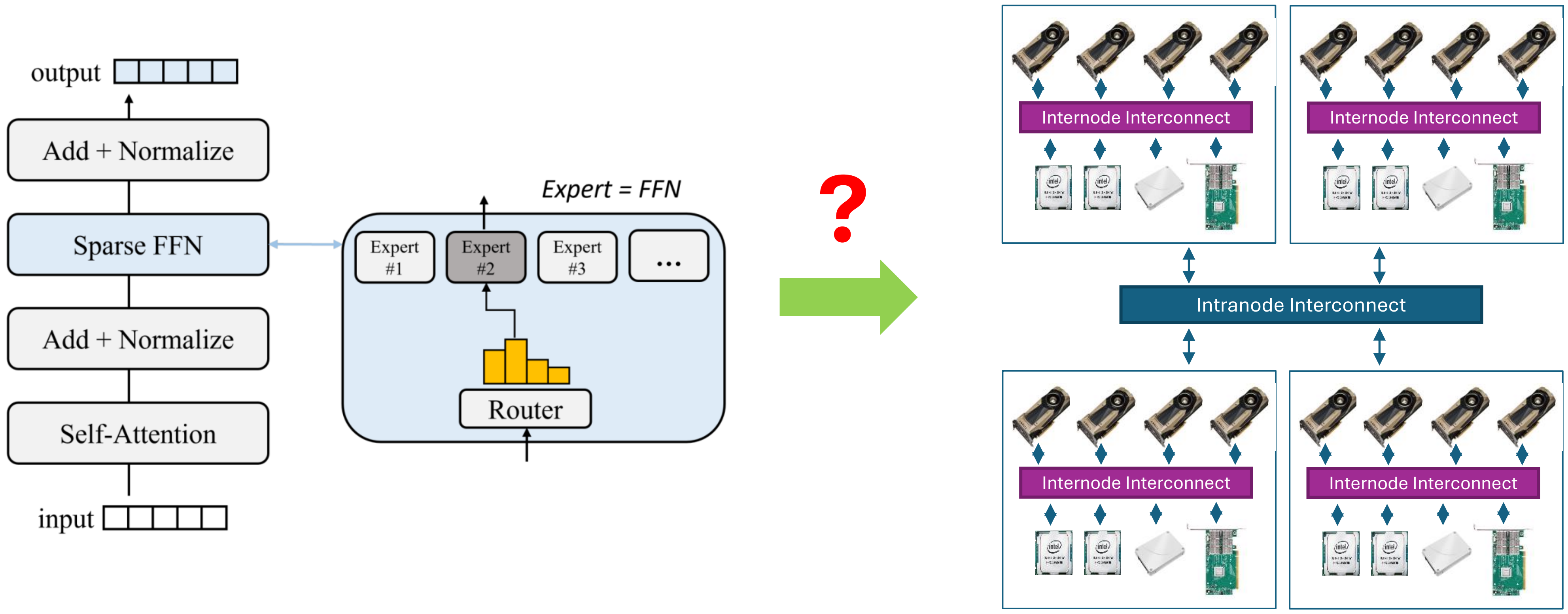


Figure 1: **(a)** The performance achieved by Routing Networks when varying the number of experts for a fixed dense model size is described by a bilinear function (Eq. 1), **(b)** whose level curves indicate how to trade model size with expert count to maintain a fixed performance, **(c)** and which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

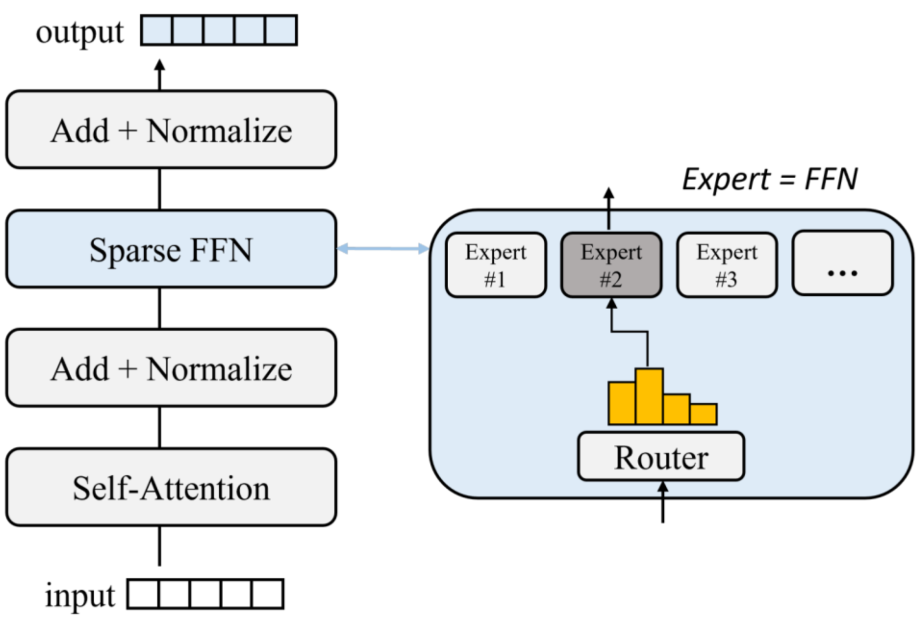
Unified Scaling Laws for Routed Language Models

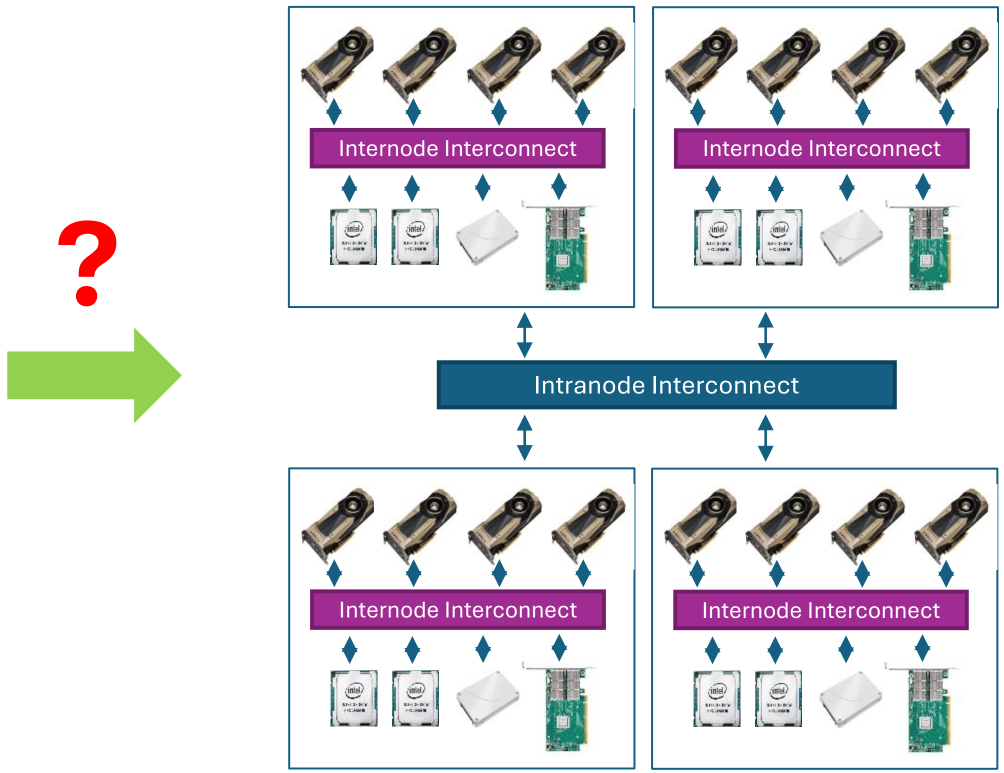
MoE Scaling Challenges on Modern Hardware with Massive Parallelism



MoE Scaling Challenges on Modern Hardware with Massive Parallelism

- How to break the memory wall to enable massive MoEs?
- How to efficiently route tokens to different experts across GPUs?
- How to minimize communication overhead while achieving high per-GPU compute throughput?

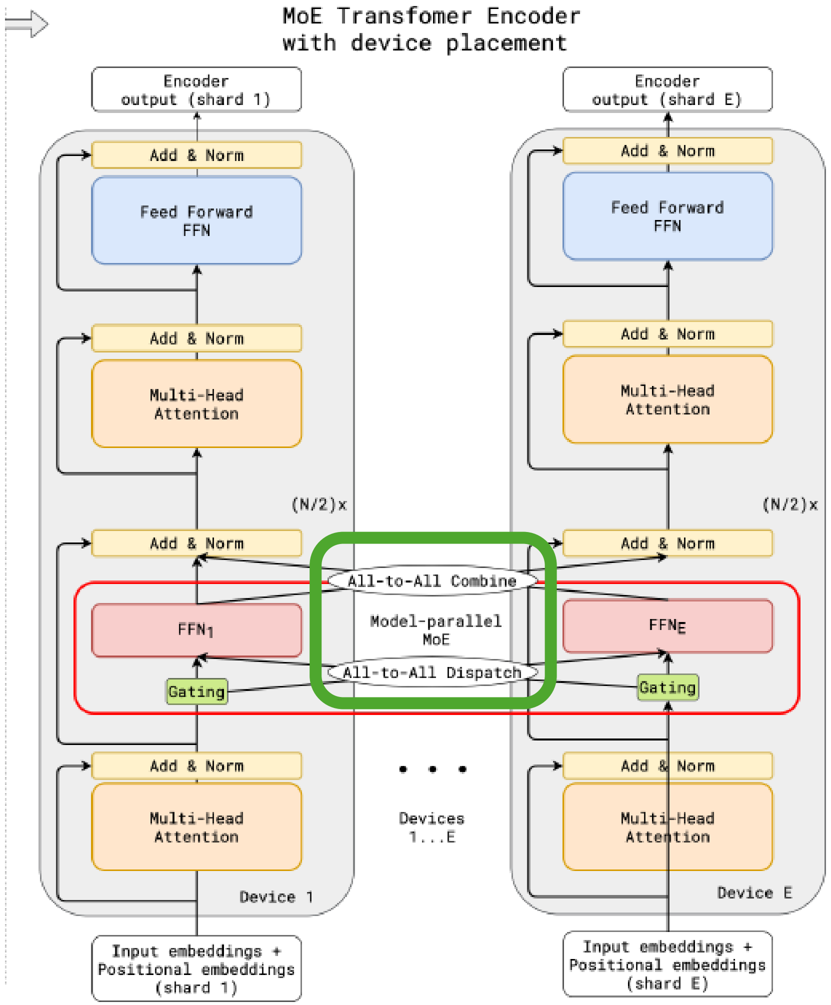




Expert Parallelism

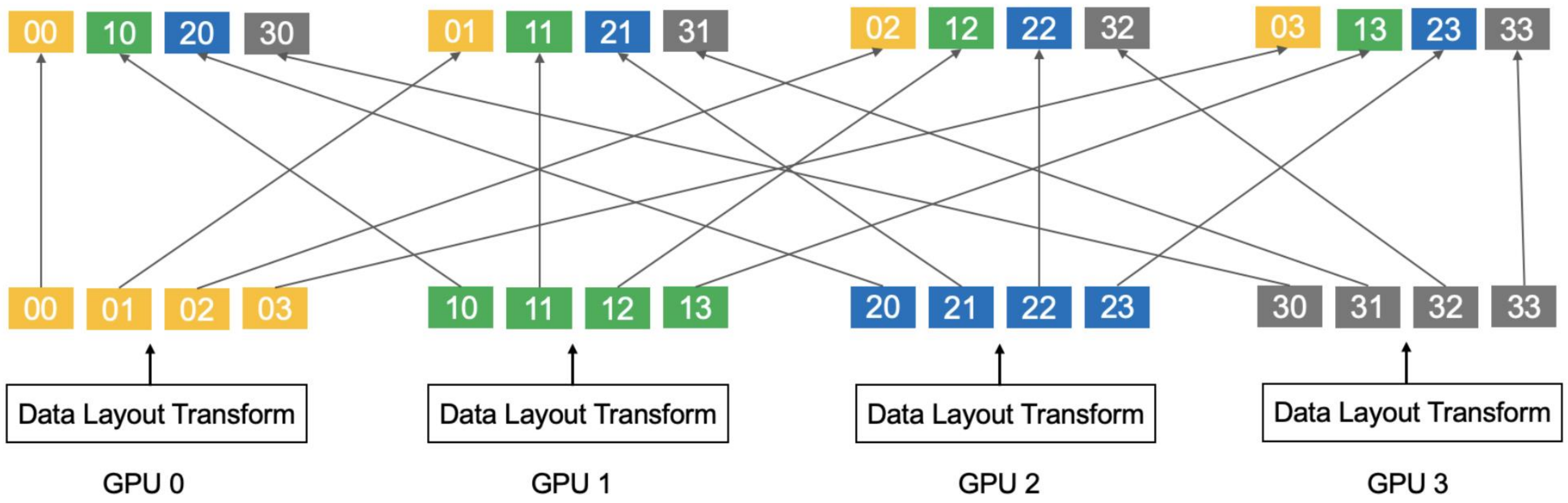
- Expert parameters – partitioned (sharded)
  - Like model parallelism (MP)
  - Each expert process a subset of tokens

- **Two All-to-All(s) in Forward and Backward**



Expert Parallelism

1. Gating function: decide target experts for each token
2. **Dispatch phase**:
   a. 1st layout transformation: tokens to the same target experts are grouped in a continuous memory buffer
   b. 1st All2All: dispatch tokens to their corresponding experts
3. Expert compute: each expert process its tokens
4. **Combine phase**:
   a. 2nd All2All: combine processed tokens back to their GPUs
   b. 2nd layout transformation: restore tokens to their original positions



How to Design Highly-Scalable Training Systems for Trillion-Parameter MoEs?

- DeepSpeed-MoE [1]

  - 4D parallelism for scaling both the base model and expert layers

- DeepSpeed-TED [2]

  - Further push the limit of MoE scalability by eliminating unnecessary communication in hybrid parallelism

- Tutle [3]

  - System and algorithm co-design achieving excellent scalability at 2048 A100 GPUs

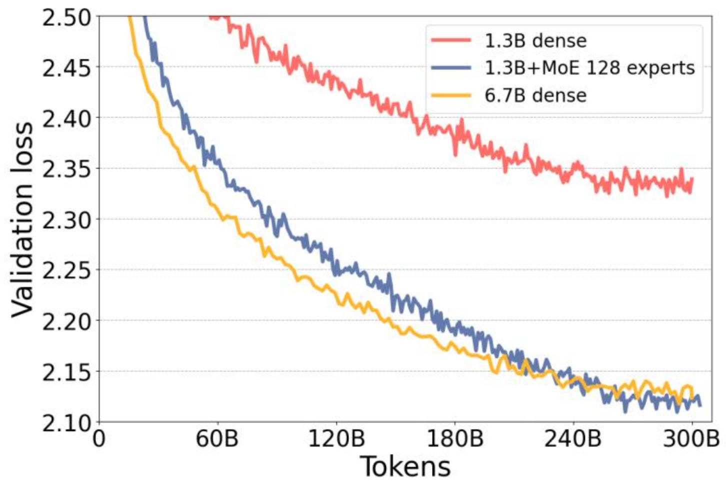
DeepSpeed-MoE: Multidimensional Parallelism

Short Name	Flexible Parallelism Combinations	Benefit
E	Expert	Scales the model size by increasing the number of experts
E+D	Expert + Data	Accelerates training throughput by scaling to multiple data parallel groups
E+Z	Expert + ZeRO	Partitions the nonexpert parameters to support larger base models
E+D+M	Expert + Data + Model	Supports massive hidden sizes and even larger base models than E+Z
E+D+Z	Expert + Data + ZeRO	
E+Z-Off+M	Expert + ZeRO-Offload + Model	Leverages both GPU and CPU memory for large MoE models on limited GPU resources

Optimal parallelism strategy depends on model and hardware specifics

DeepSpeed-MoE: Cheaper GPT Model Training with MoE

- 1.3B+MoE with 128 experts, compared to 1.3B and 6.7B dense (GPT-3 like)
- **8x** more parameters to same accuracy using MoE
- **5x** lower training cost to same accuracy using MoE



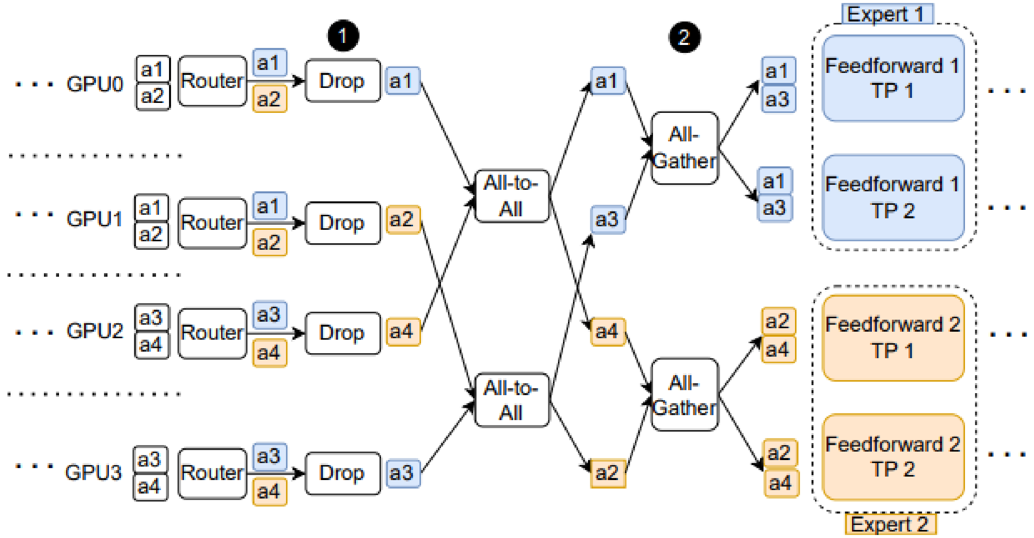
Case	Model size	LAMBADA: completion prediction	PIQA: commonsense reasoning	BoolQ: reading comprehension	RACE-h: reading comprehension	TriviaQA: question answering	WebQs: question answering
Dense GPT:							
(1) 350M	350M	52.03	69.31	53.64	31.77	3.21	1.57
(2) 1.3B	1.3B	63.65	73.39	63.39	35.60	10.05	3.25
(3) 6.7B	6.7B	**71.94**	**76.71**	**67.03**	37.42	23.47	5.12
Standard MoE GPT:							
(4) 350M+MoE-128	13B	62.70	74.59	60.46	35.60	16.58	5.17
(5) 1.3B+MoE-128	52B	69.84	**76.71**	64.92	**38.09**	**31.29**	**7.19**

	Training samples per sec	Throughput gain/ Cost Reduction
6.7B dense	70	1x
1.3B+MoE-128	372	**5x**

DeepSpeed-TED

- Further push the limit of MoE scalability by eliminating unnecessary communication

Duplicate token dropping (DTD): Eliminating unnecessary tokens, e.g., in all2all and all-gather from EP + TP.

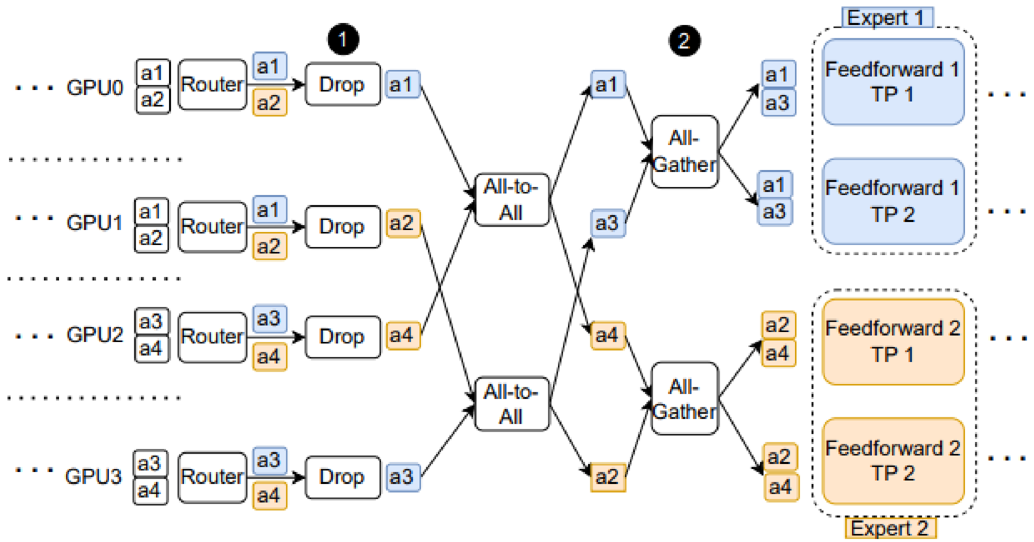


A Hybrid Tensor-Expert-Data Parallelism Approach to Optimize Mixture-of-Experts Training

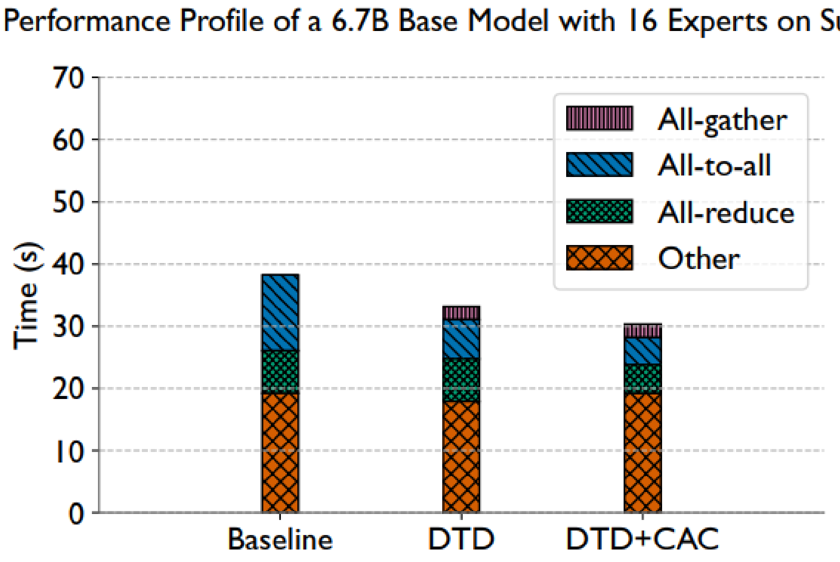
DeepSpeed-TED

- Further push the limit of MoE scalability by eliminating unnecessary communication

Duplicate token dropping (DTD): Eliminating unnecessary tokens, e.g., in all2all and all-gather from EP + TP.



Communication-aware Activation Checkpointing (CAC): selective activation checkpointing by avoiding all2all during recomputation



128 GPUs

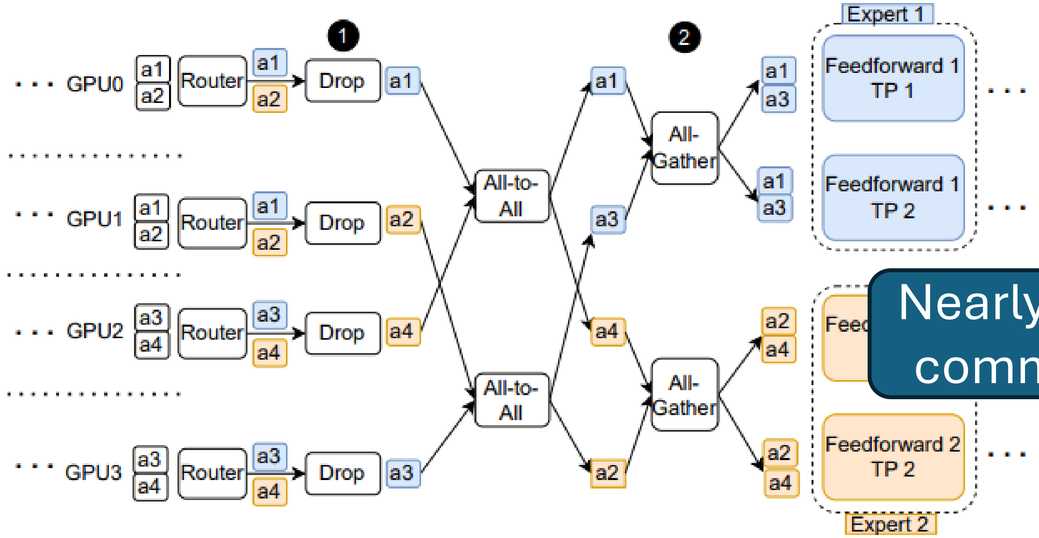
A Hybrid Tensor-Expert-Data Parallelism Approach to Optimize Mixture-of-Experts Training

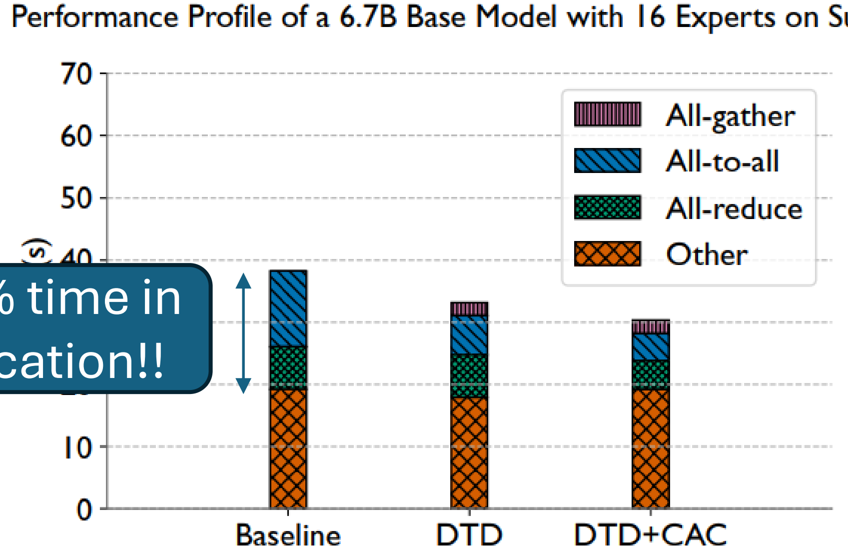
DeepSpeed-TED

- Further push the limit of MoE scalability by eliminating unnecessary communication

Duplicate token dropping (DTD): Eliminating unnecessary tokens, e.g., in all2all and all-gather from EP + TP.

Communication-aware Activation Checkpointing (CAC): selective activation checkpointing by avoiding all2all during recomputation





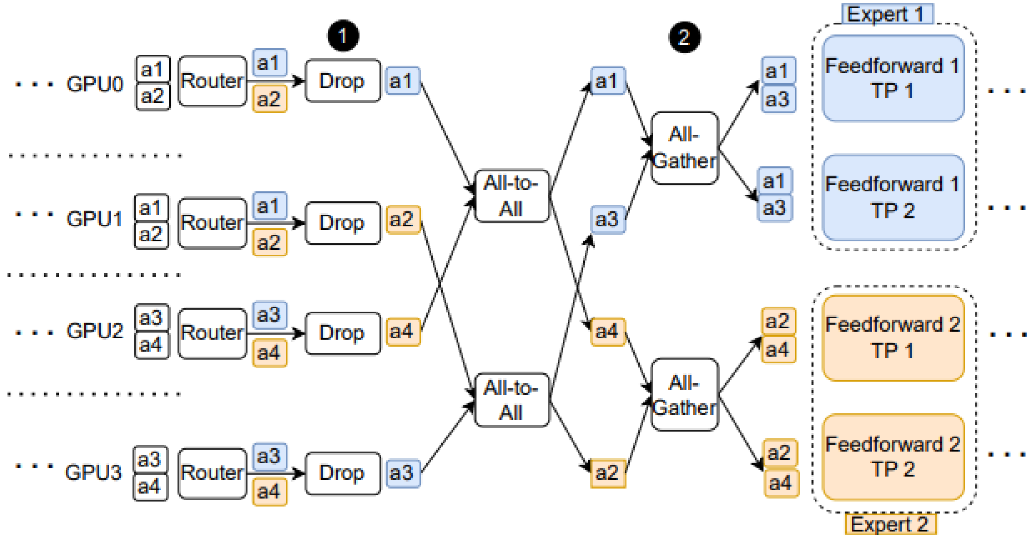
128 GPUs

A Hybrid Tensor-Expert-Data Parallelism Approach to Optimize Mixture-of-Experts Training

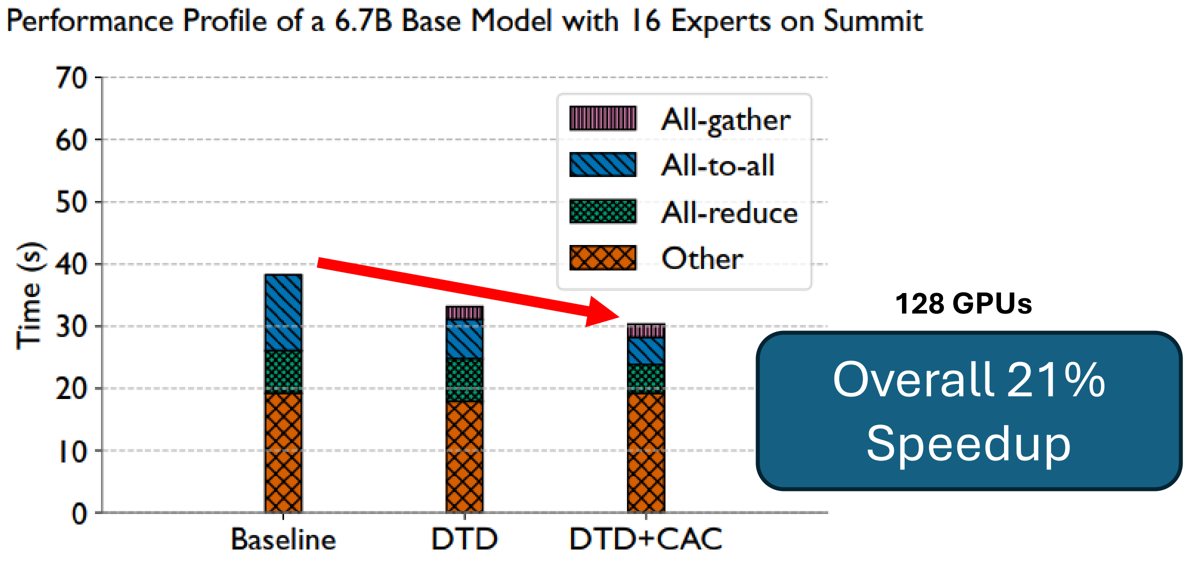
DeepSpeed-TED

- Further push the limit of MoE scalability by eliminating unnecessary communication

Duplicate token dropping (DTD): Eliminating unnecessary tokens, e.g., in all2all and all-gather from EP + TP.



Communication-aware Activation Checkpointing (CAC): selective activation checkpointing by avoiding all2all during recomputation



[A Hybrid Tensor-Expert-Data Parallelism Approach to Optimize Mixture-of-Experts Training]

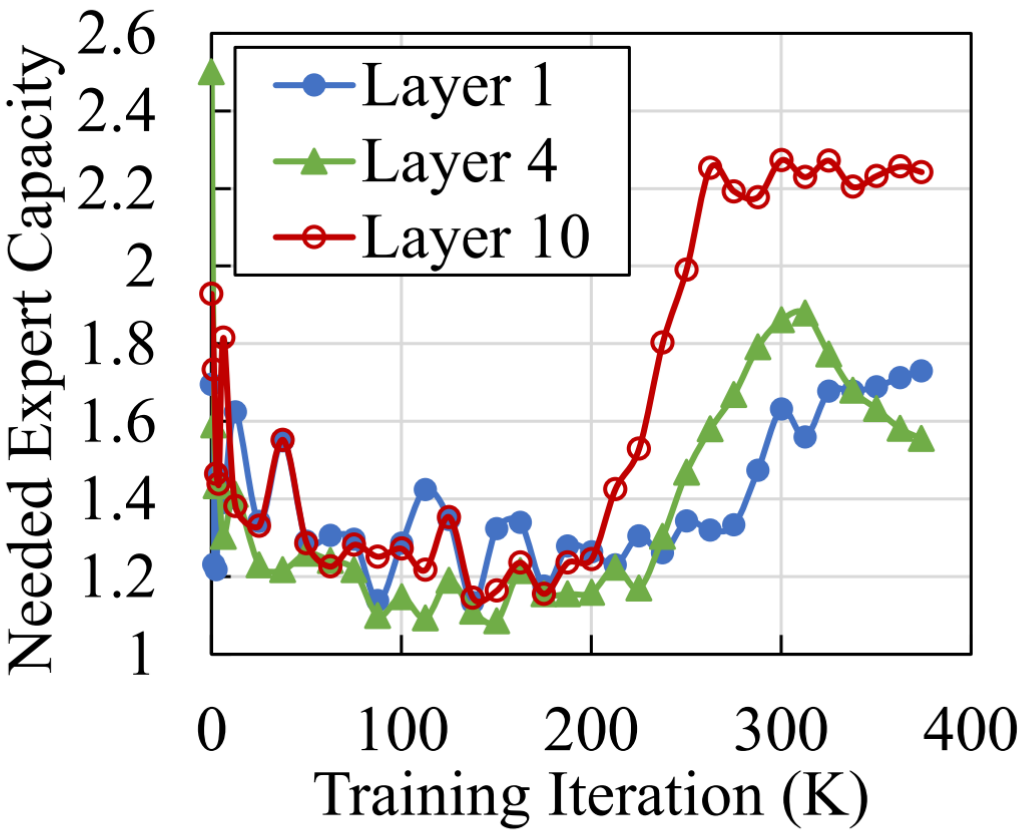
Tutle: Adaptive MoE at Scale

- Key idea: system-algorithm co-design
- Dynamically adapt parallelism
- 2D hierarchical all2all
- Adaptive pipeline

Tutel: Adaptive mixture-of-experts at scale

Tutle: Adaptive MoE at Scale

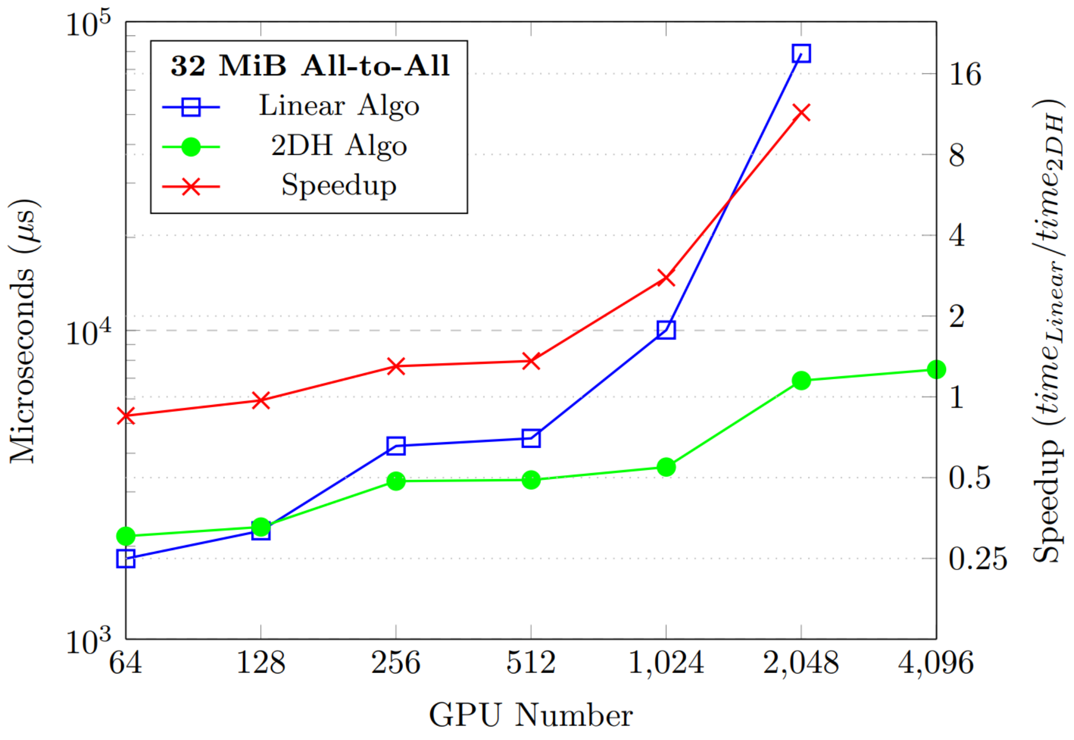
- Observation: Workload per expert changes during training
- Solution: Dynamically adapt parallelism DP + EP vs. TP + EP



Tutel: Adaptive mixture-of-experts at scale

Tutle: Adaptive MoE at Scale

- Observation: All2all is expensive across nodes and with many small messages
- Solution 1: Take into account of network hierarchy with 2D hierarchical all2all: Intra-node all2all + Inter-node all2all
- Solution 2: Leverage highly-optimized communication collectives from MSCCL



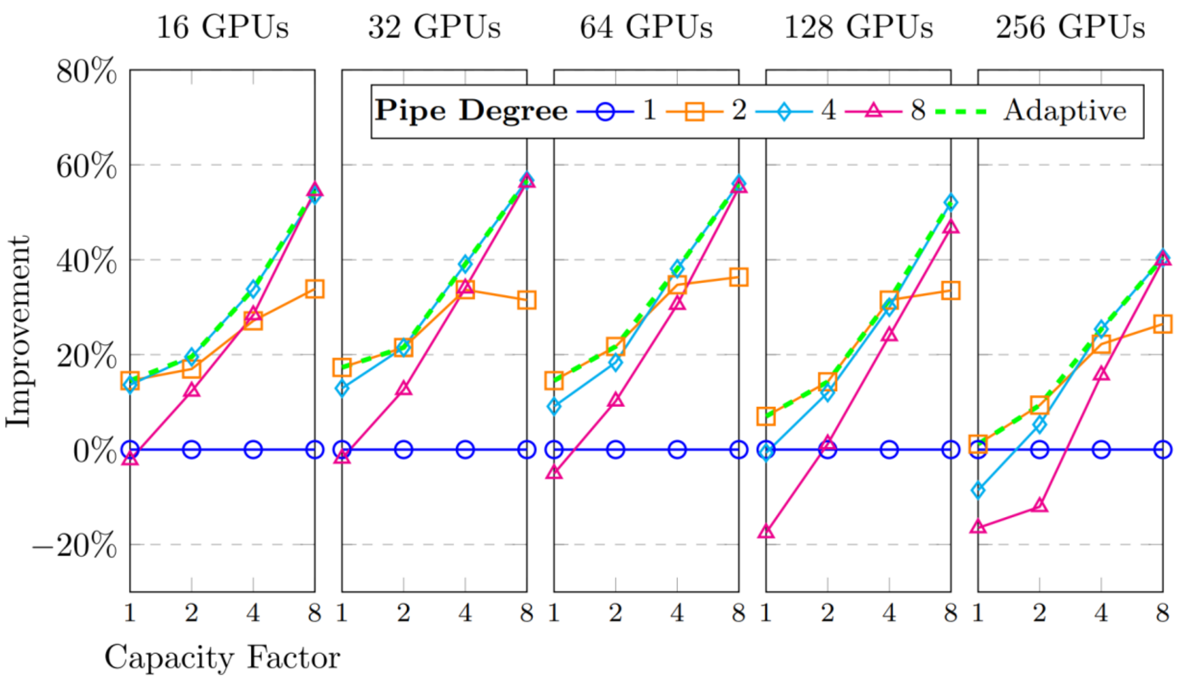
Up to 10x all2all speedup

Tutel: Adaptive mixture-of-experts at scale

Tutle: Adaptive MoE at Scale

- Observation: Token partitioning + concurrent CUDA kernels => pipeline parallelism that overlap all2all with FFN layer compute

- Solution: Adaptive pipeline degree based on workloads

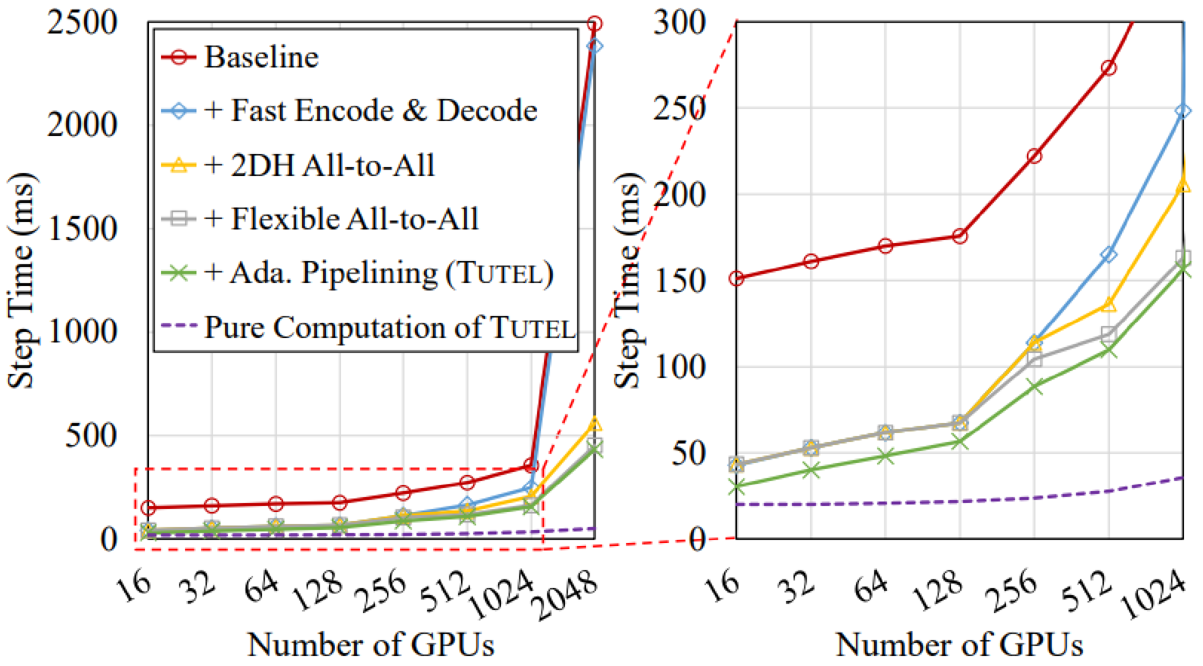


Up to 57% improvement in comparison to pipeline degree 1

Tutel: Adaptive mixture-of-experts at scale

Tutle: Adaptive MoE at Scale

- Dynamically adaptive parallelism
- Dynamic pipelining
- 2D hierarchical all2all



5.7× end-to-end speed at 2048 A100 GPUs!

Tutel: Adaptive mixture-of-experts at scale

Moving Forward

- Novel MoE architecture and training objective design
- Expect more optimizations against the training efficiency of MoE models, e.g., parameter-efficient MoE, multi-modal MoE
- Subsequent extensions of MoE based foundation models to diverse use cases
- System optimizations that leverage heterogeneous hardware resource to lower the cost of training and fine-tuning MoE
- Efficient MoE inference systems to achieve low latency and high-throughput

Thank you!
Q&A

Minjia Zhang
minjiza@illinois.edu