CSC290/420 Machine Learning Systems for Efficient AI
Training IV - Mixture of Experts

Sreepathi Pai

November 5, 2025

URCS

Outline

Mixture of Experts

Training MoEs - Conditional Computation

Inference

Outline

AI Scaling

- Dogma: Models with more parameters perform better
- But training larger models is slower
  - "quadratically slow"
- Can larger models be trained at sublinear costs?

Mixture of Experts

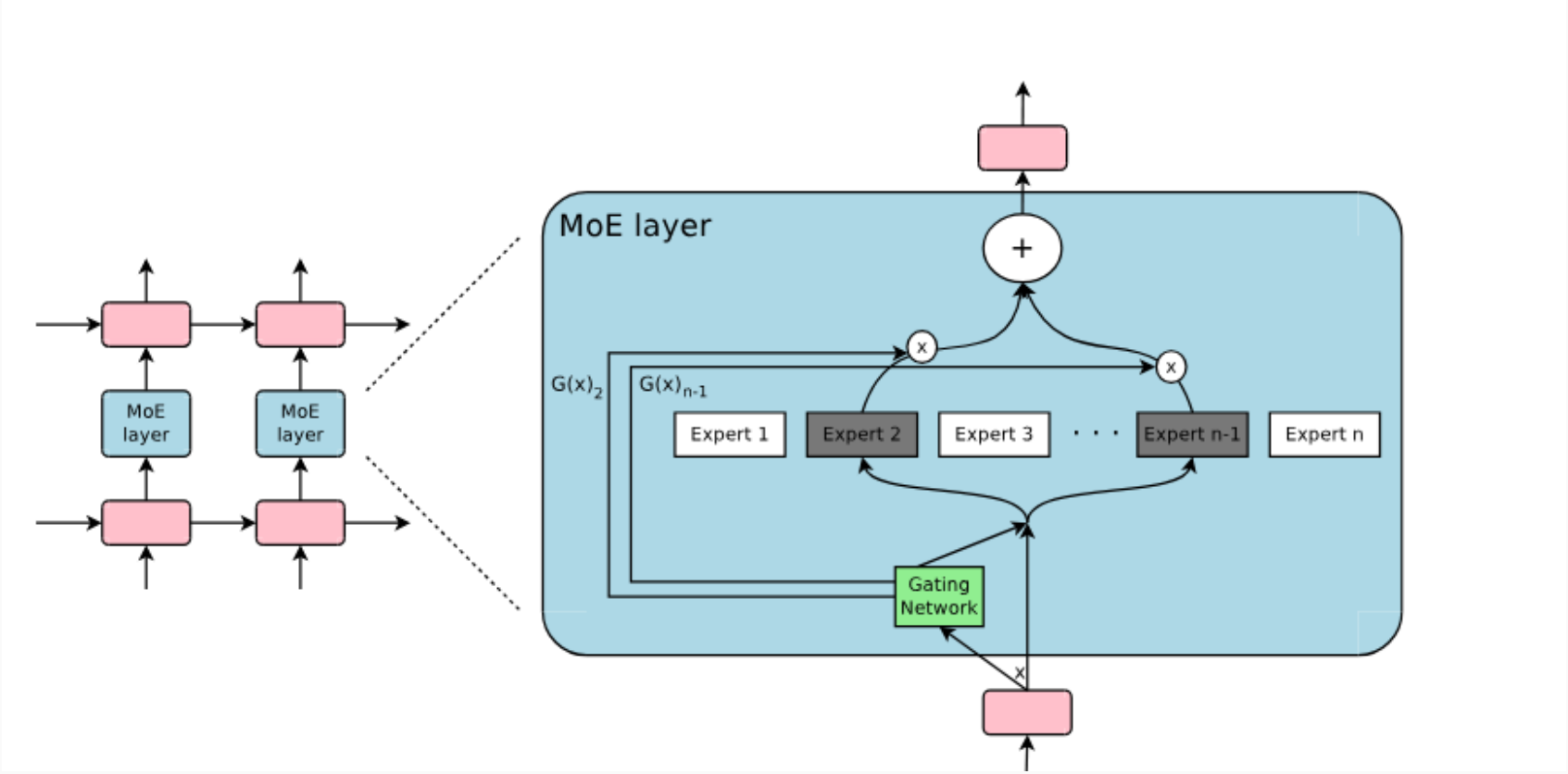
- Mixture of experts (from our very own Prof. Robbie Jacobs)
- $y = \Sigma_{i=1}^{n} G(x) i E_i(x)$
- I.e., the output is the weighted sum of product of the output of a gate network G and an expert E_i
- If $G(x)_i$ is 0, then $E_i(x)$ doesn't need to be computed!

Common Notions

- This is a sparse network and/or sparse computation
- Performance can be compared to a dense network of the same size
  - Model/Task performance
  - System performance (latency, throughput)
- Algorithmic considerations: number of operations, amount of memory

Outline

Sparsely Gated Mixture-of-Experts Layer



Shazeer et al., Outrageously Large Neural Networks: The Sparsely-Gated Mixture-Of-Experts Layer

Lowered Computational Intensity aka Shrinking Batch Problem

- If there b examples in a batch, and k experts are chosen out of n, then each expert will only receive
- Solution:
  - Use data parallelism (DP) at the model level over d devices
  - But only one copy of each expert (i.e. model parallelism)
  - then route examples from different batches to the expert. thus kbd/n examples per expert
- Requires synchronization amongst DP devices

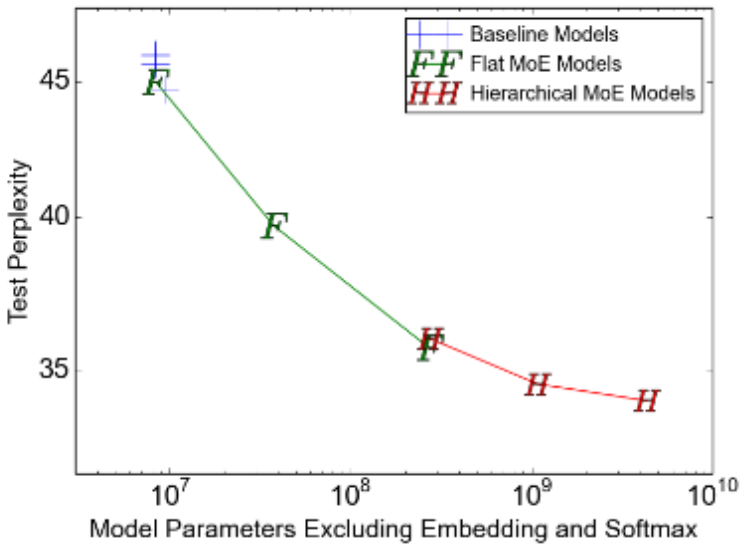
Network Bandwidth

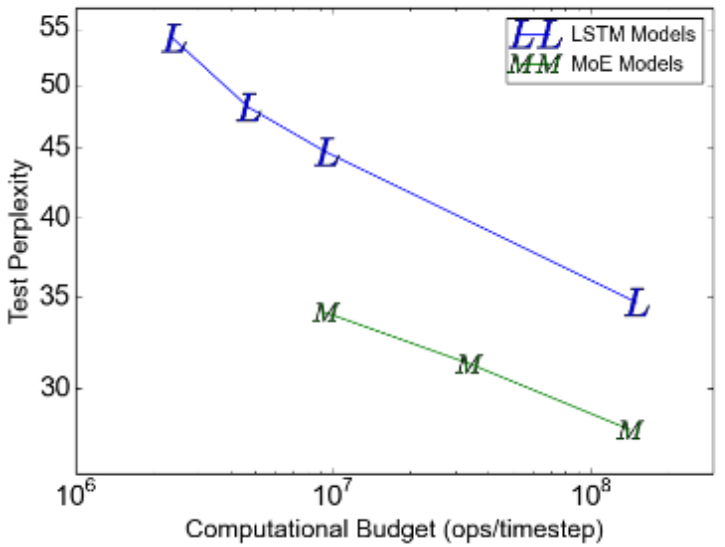
- Inputs and outputs of experts must be sent across network
  - Since only one copy of each expert
- However, communication sizes depend on hidden layers of expert layers
  - Can be changed to reduce amount of data transferred

Load Balancing

- No guarantee that the Gating network is "fair" or sends tokens to each expert equally
- Some experts might enter a virtuous cycle
  - more tokens, more training, better results, ...
- Bad for system performance [underutilized GPUs]
- Train the gating network to be more fair [or have less variance]
  - I.e. no expert is more "important" than another

Benefits

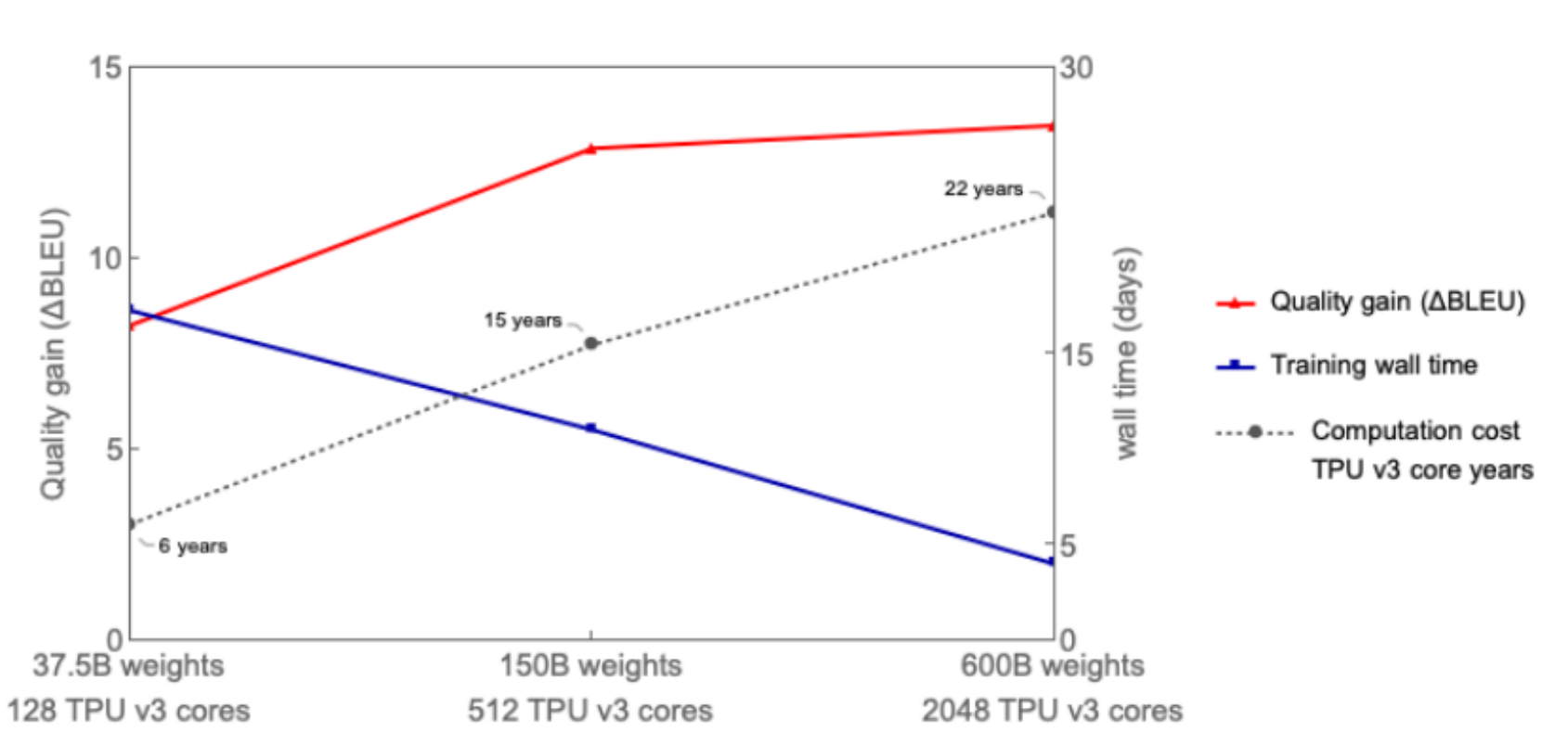

Baseline Models
Flat MoE Models
Hierarchical MoE Models
Test Perplexity
45
40
35
10^7
10^8
10^9
10^{10}
Model Parameters Excluding Embedding and Softmax


LSTM Models
MoE Models
Test Perplexity
55
50
45
40
35
30
10^6
10^7
10^8
Computational Budget (ops/timestep)

Infrastructure

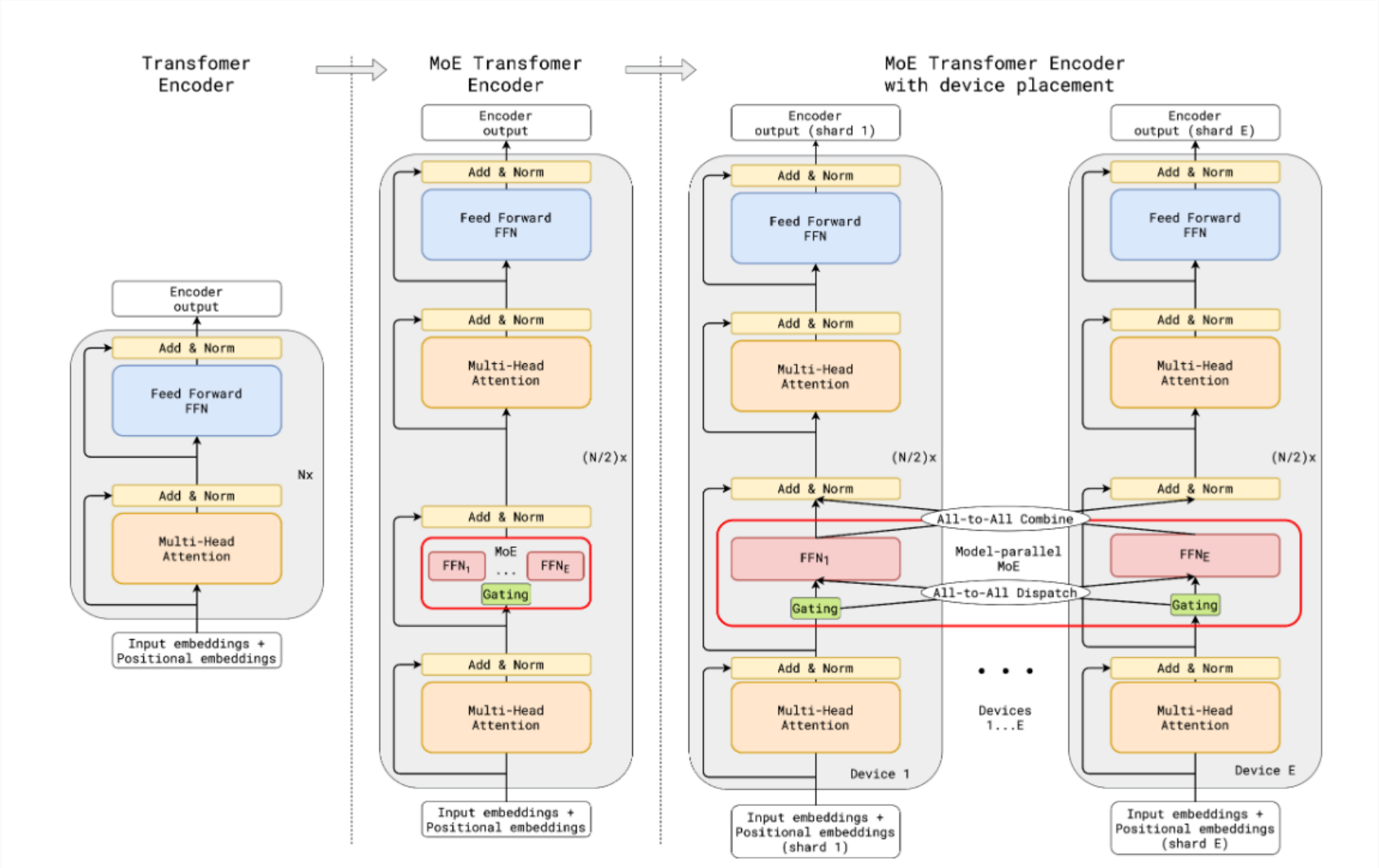
- 16-32 Tesla K40
- 10^{11} parameters (100B-ish)
- Thousands of experts

GShard: MoE applied to Transformers



Lepikhin et al., GShard: Scaling Giant Models With Conditional Computation And Automatic Sharding

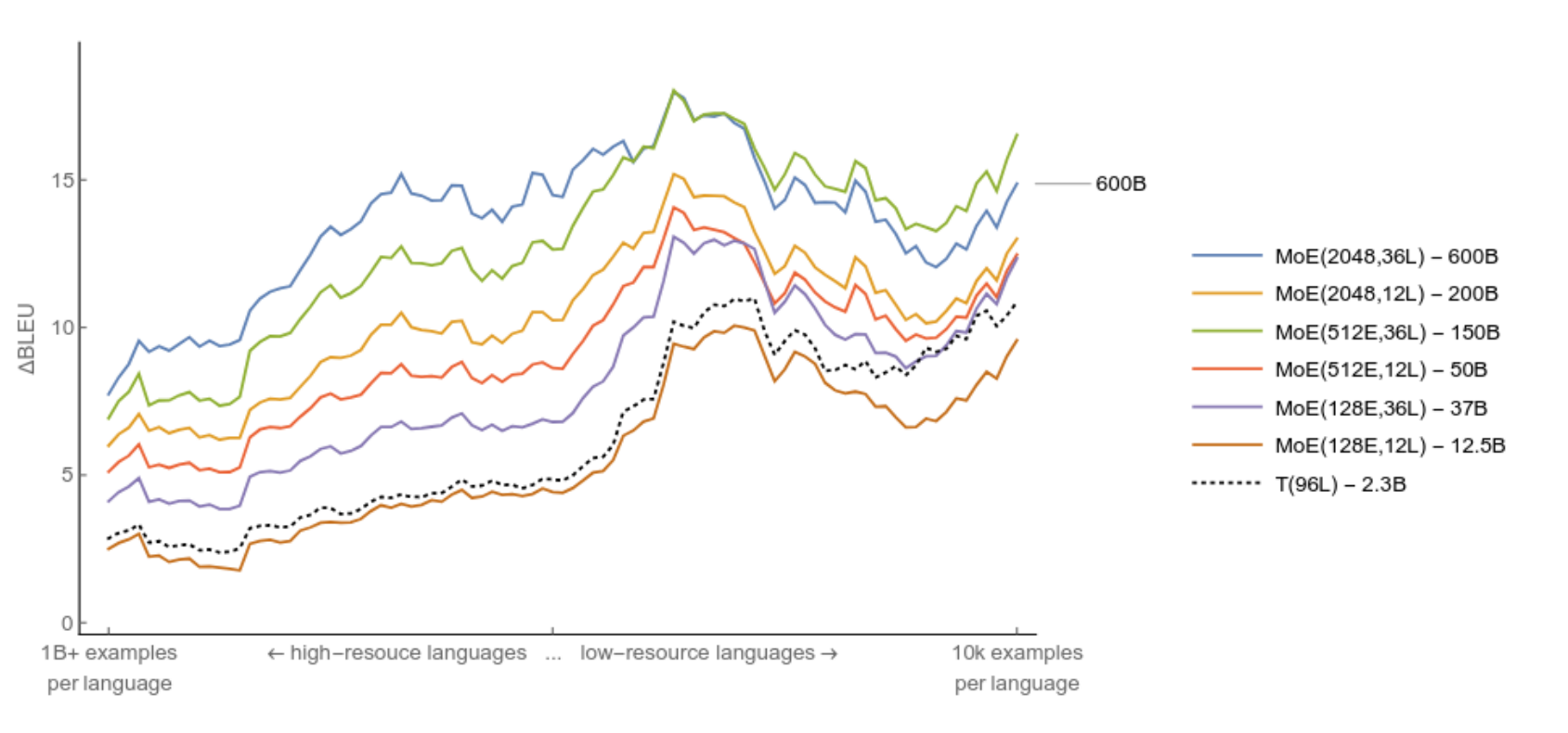
Design


Transfomer Encoder
MoE Transfomer Encoder
MoE Transfomer Encoder with device placement
Encoder output
Add & Norm
Feed Forward FFN
Multi-Head Attention
Nx
(N/2)x
MoE
FFN1
FFNE
Gating
Input embeddings + Positional embeddings
Encoder output (shard 1)
Encoder output (shard E)
All-to-All Combine
All-to-All Dispatch
Model-parallel MoE
Devices 1...E
Device 1
Device E
Input embeddings + Positional embeddings (shard 1)
Input embeddings + Positional embeddings (shard E)

Other Design aspects

- Parallel Load balancing
- Random Routing

Performance



Training Efficiency

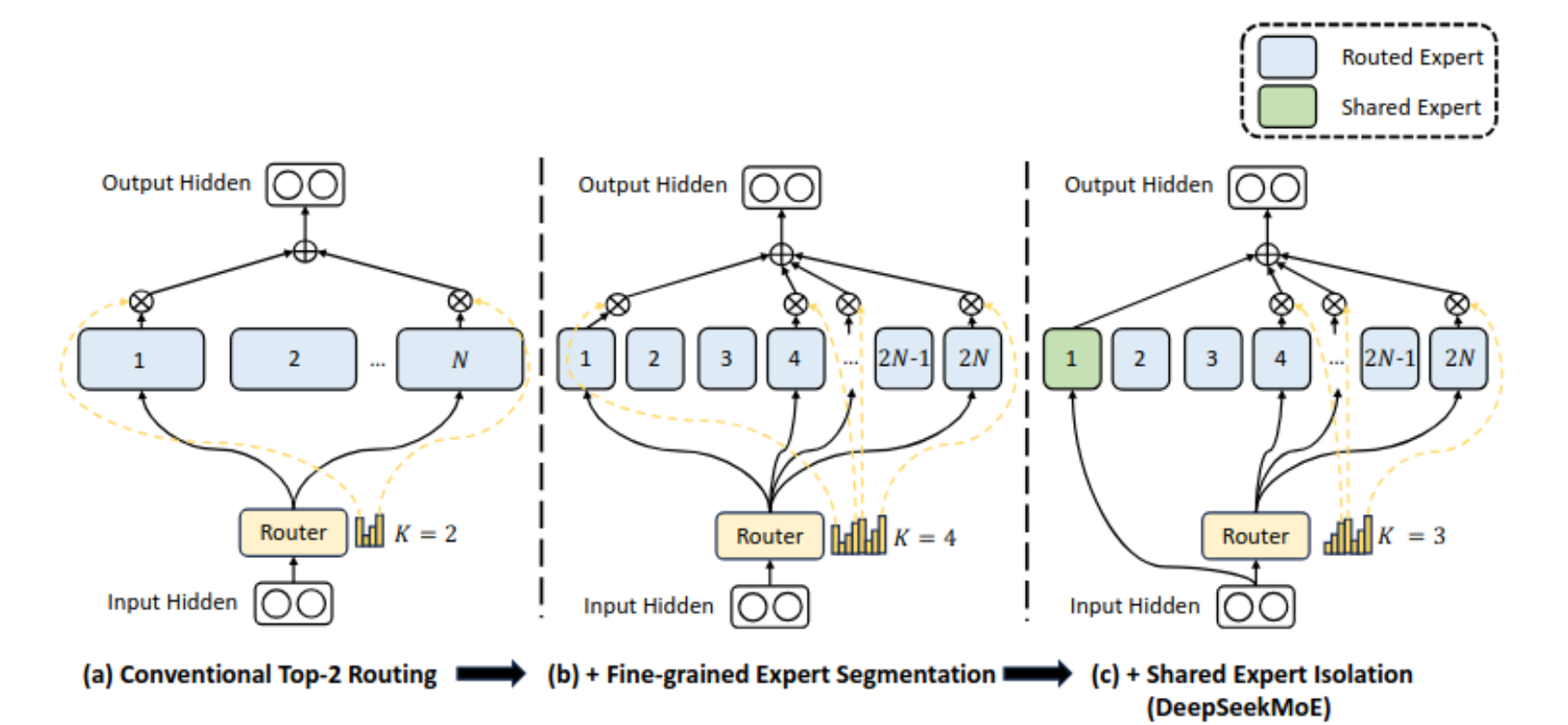
Model	Cores	Steps / sec.	Batch sz (Tokens)	TPU core years	Training days	BLEU avg.	Billion tokens to cross-entropy of 0.7	0.6	0.5
MoE(2048E, 36L)	2048	0.72	4M	22.4	**4.0**	**44.3**	82	175	542
MoE(2048E, 12L)	2048	2.15	4M	7.5	1.4	41.3	176	484	1780
MoE(512E, 36L)	512	1.05	1M	15.5	11.0	43.7	66	170	567
MoE(512E, 12L)	512	3.28	1M	4.9	3.5	40.0	141	486	-
MoE(128E, 36L)	128	0.67	1M	6.1	17.3	39.0	321	1074	-
MoE(128E, 12L)	128	2.16	1M	1.9	5.4	36.7	995	-	-
T(96L)	2048	-	4M	$\sim$235.5	$\sim$42	36.9	-	-	-
Bilingual Baseline	-	-		$\sim$ 29	-	30.8	-	-	-

Table 1: Performance of MoE models with different number of experts and layers.

Some more implementation details

- Implemented as a compiler with user annotations
- Communication reflects device topology
- $O(1)$ memory usage vis-a-vis number of experts
- Somewhat TPU-specific
- See paper appendices for more details

DeepSeekMoE



Dai et al., DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models

Fine Grained Expert Segmentation

- Hypothesis: Limited experts means those experts need to have more general knowledge
- But more experts means more parameters and more memory
- Instead: segment each expert into m smaller experts
  - by changing dimensions of feed-forward networks in each expert
- Then, increase the number of experts activated by m

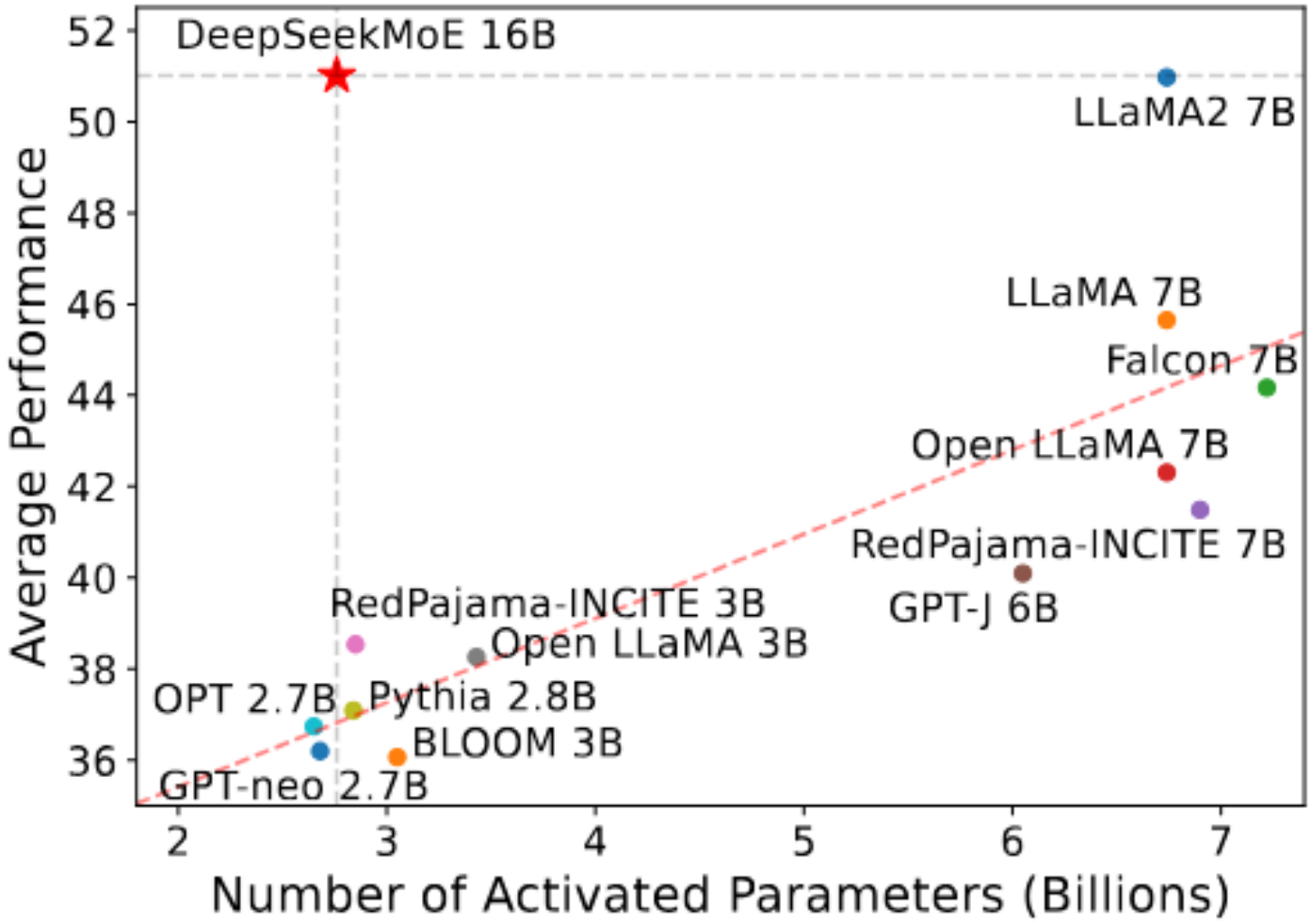
Shared Experts

- Hypothesis: all experts gain shared common knowledge which is inefficient
- Have K_s shared experts that handle every token
- Decrease dynamic experts activated by K_s to keep total number of activated experts same
- Similar to Residual MoE of DeepSpeed-MoE (later)

Other concerns

- Device-Level Balancing
  - Make the gating network assign same computation across devices

DeepSeekMOE performance



Other MoE designs

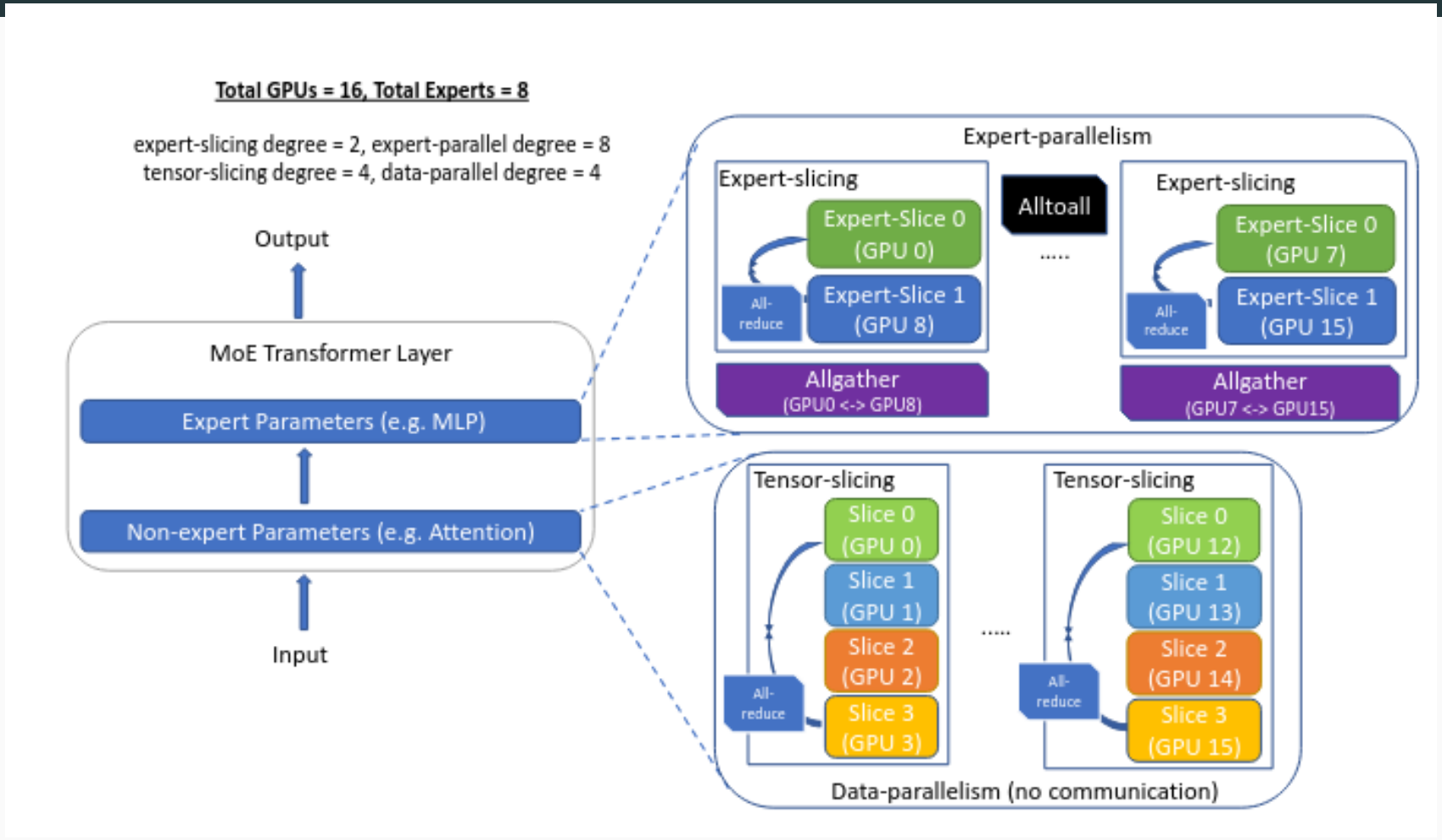
- DeepSpeed-MoE
  - Phenomenon I: Where should MoE layers be present?
  - Phenomenon II: aka Shared Experts/Residual MoE
- Knowledge Distillation (aka Mixture of Students)

Outline

DeepSpeed-MoE Inference

- Inference performance bounded by:
  - need only 1 expert vs need all experts
- Three contributions:
  - carefully parallelism + tokens with same critical path routed to same device
  - communication scheduling with parallelism
  - optimized MoE kernels

Combining Different forms of Parallelism



- Expert Parallelism
  - across devices
- Non-expert Parallelism
  - Tensor sliced within a single node

Communication Primitives

- Implement hierarchical AlltoAll
  - Increases volume of communication
  - But reduces hops
- Uses observation that communication for small batch sizes is latency bound vs bandwidth bound
- Also takes advantage of tensor slicing + expert slicing
  - Merge parts of the AllToAll with AllReduce

Optimized Kernels for MoE

- MoE-specific kernels
  - Gating network
- Involves manipulation of a sparse tensor
  - and multiple kernels for different steps (top-k, softmax, cumulative sum, etc.)
- Kernel Fusion
- Use "Blelloch Scan" (Prefix Scan?) to perform cumulative sum in parallel