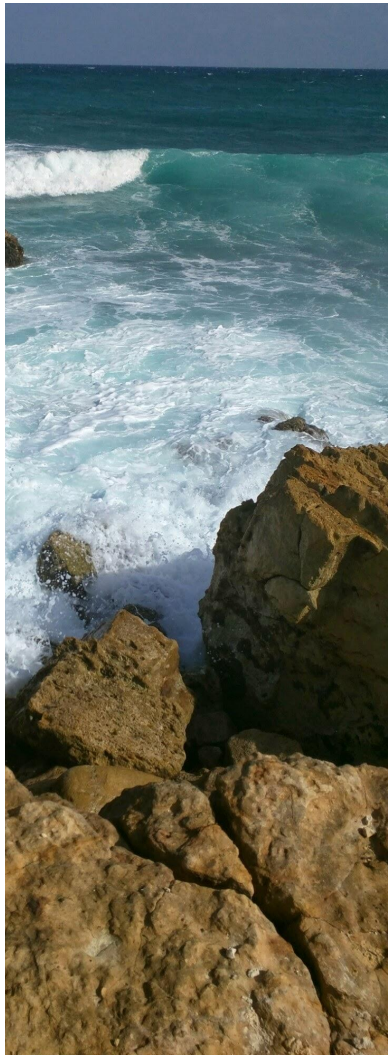
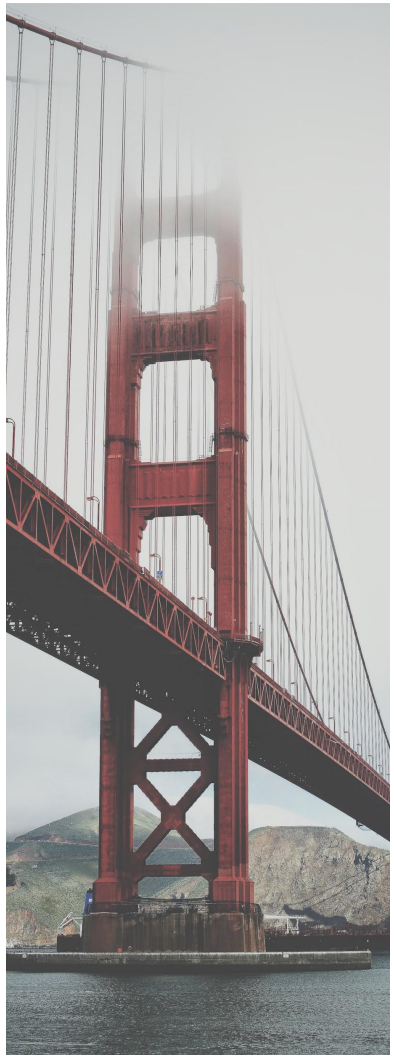


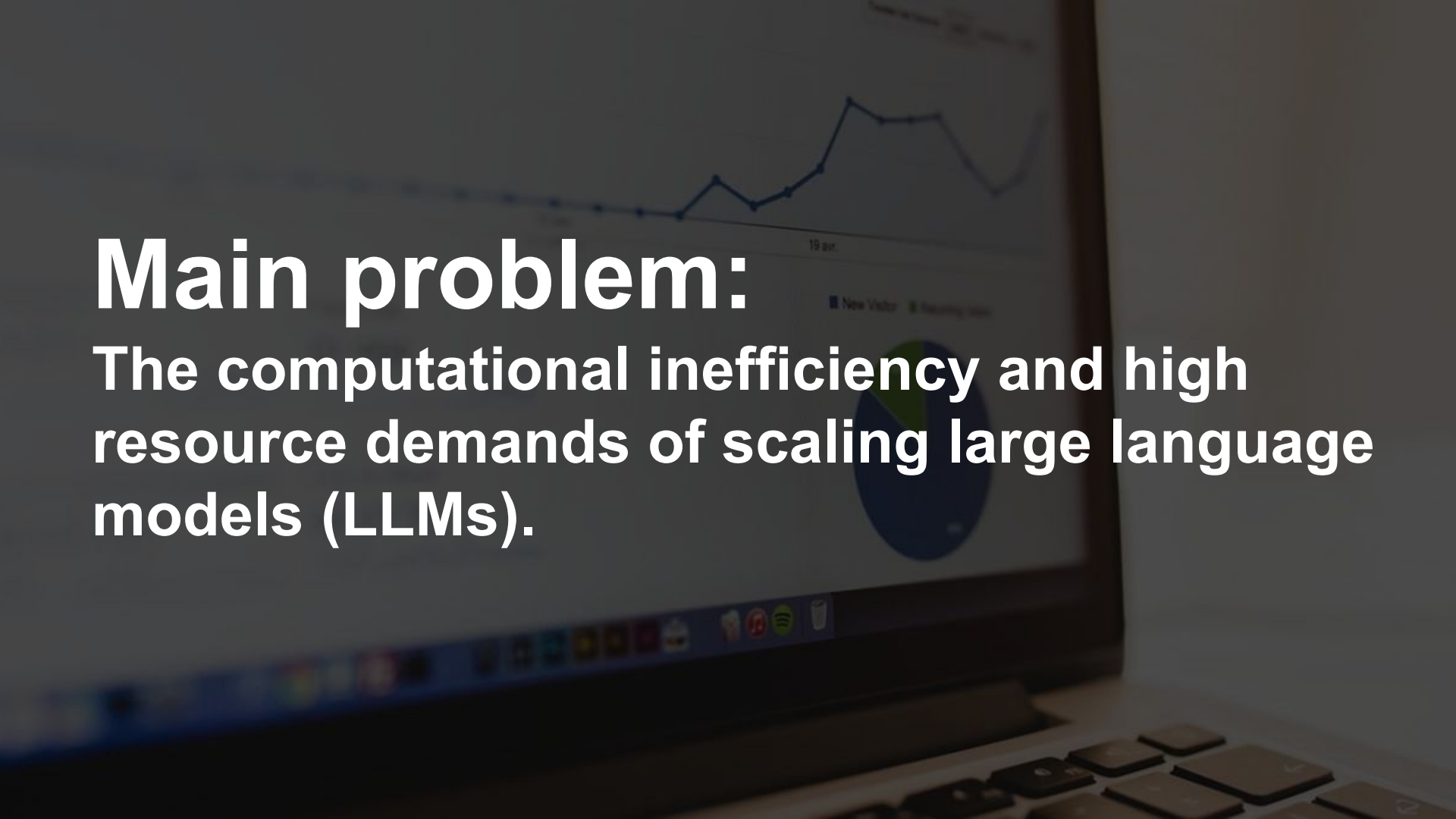


Efficient Large Scale Language Modeling with Mixtures of Experts

Author:Meta AI

Presented by:

Fadi Sultan and Msuega Jnr. Iorpenda



Main problem:

The computational inefficiency and high resource demands of scaling large language models (LLMs).

Large Language Models (LLMs)

- are advanced neural networks designed to process and generate human-like text by learning patterns, context, and relationships within large-scale text datasets.

- based on transformer architectures like GPT and BERT.

SCALING

The process of increasing their **size, capacity, and performance** to achieve better results in natural language processing (NLP) tasks. This involves:

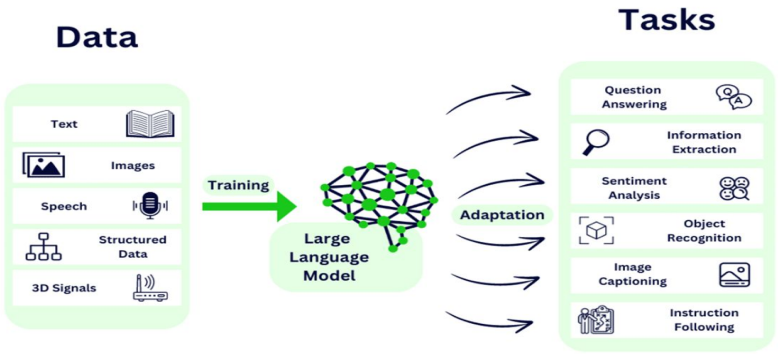
1. **Increasing Model Parameters**:
- Adding more layers, neurons, or trainable parameters to capture complex relationships and patterns.
- Example: Scaling from GPT-2 (1.5 billion parameters) to GPT-3 (175 billion parameters).

2. **Expanding Training Data**.
3. **Leveraging Compute Resources**: Utilizing powerful hardware (e.g., GPUs).

Large language models

- Tasks:

1. Language translation.
2. Question answering.
3. Summarization.
4. and more.

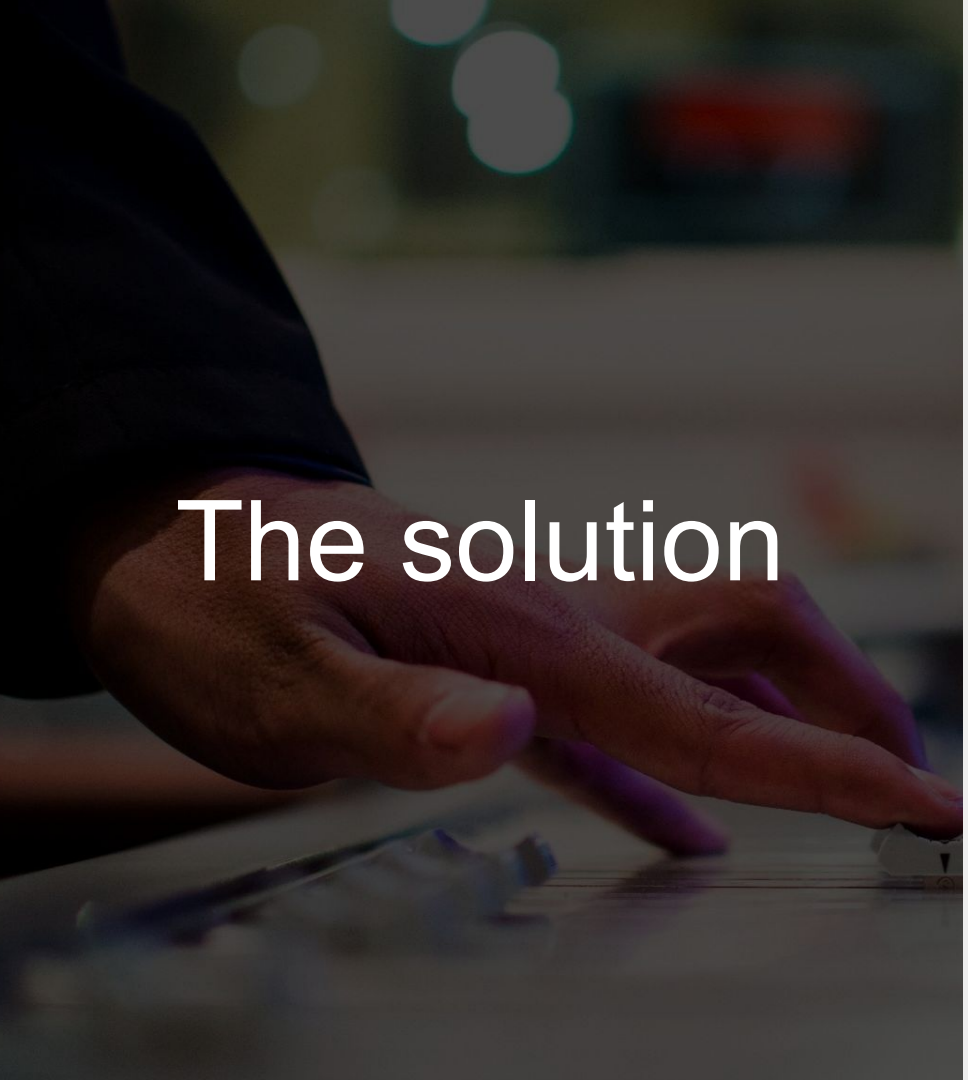


Paris is city in….

Large Language Model

1. France 30%
2. We 20%
3. the 15%
4. Are 10%
5. Were 10%

…..

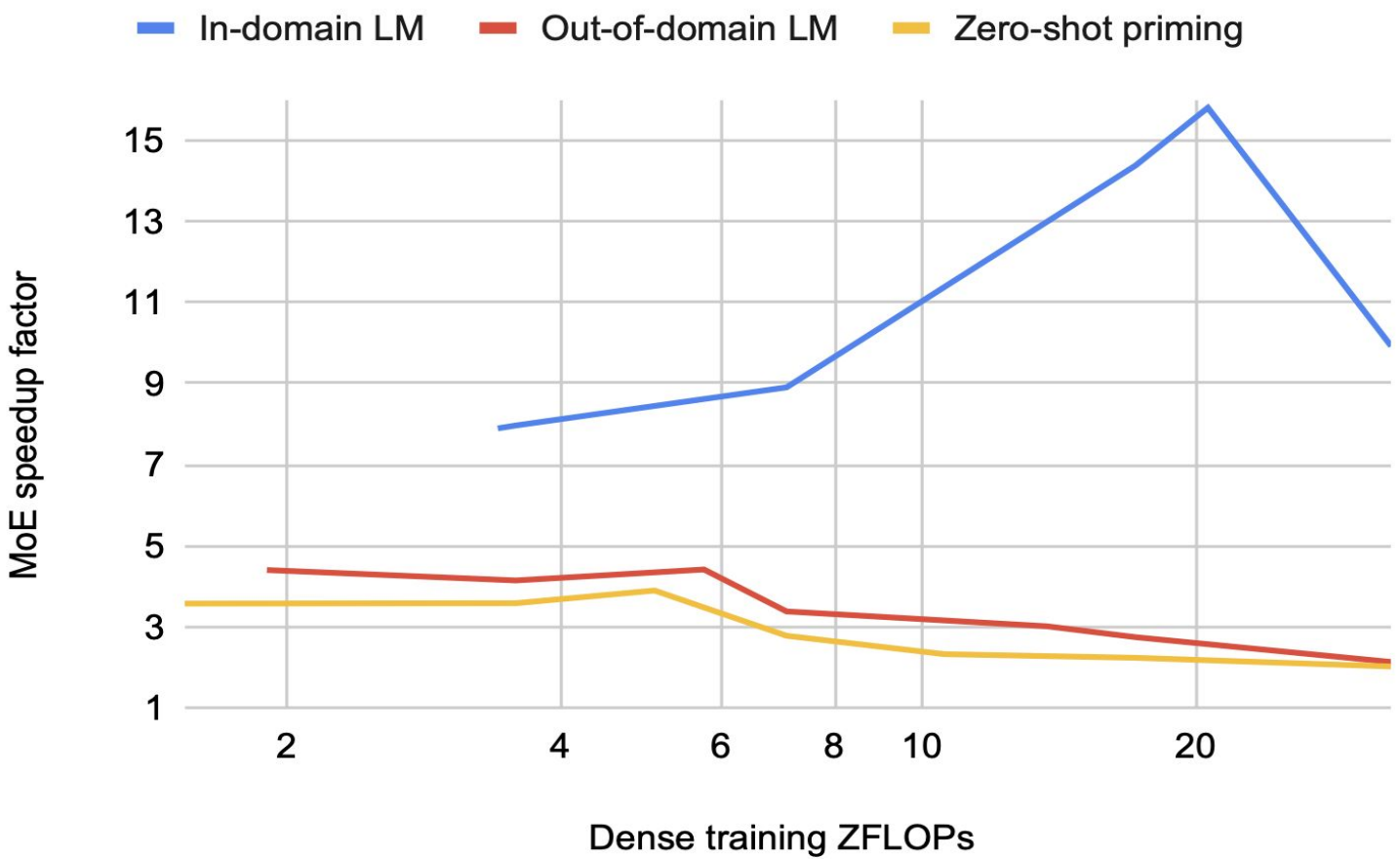


The solution

Using Sparse model (**Mixture of Experts (MoE)**).

Instead of activating all parameters for every input, MoE selectively uses only a subset of its components, called **experts**, to process each input.

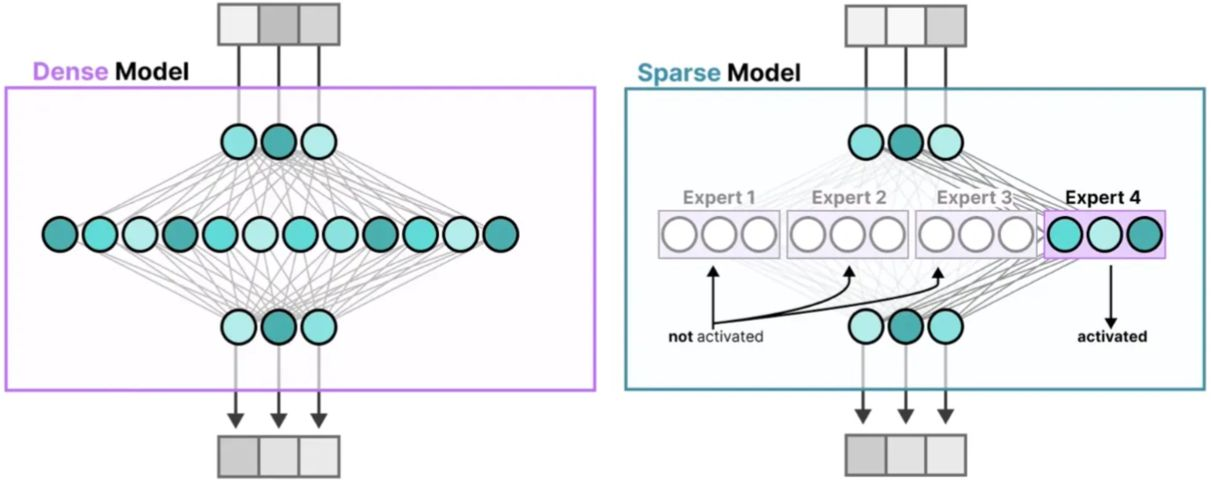
Aspect	Sparse Models	Dense Models
Parameter Activation	Activates a subset of parameters (e.g., specific experts) selectively based on input, reducing redundant computations.	Activates all parameters for every input, leveraging the full model capacity uniformly.
Computational Cost	Lower, as only a portion of the model is utilized per computation, leading to resource savings.	High, as the entire model is engaged for all inputs, increasing resource usage.
Scalability	Can scale to trillions of parameters by activating only the necessary subset, avoiding proportional compute increase.	Scaling significantly increases computational and memory costs due to uniform parameter activation.
Generalization	May exhibit complementary generalization behavior; sparse routing may require optimization for zero-/few-shot tasks.	Strong generalization due to uniform parameter activation, excelling in zero-/few-shot learning tasks.
Example	Mixture of Experts (MoE) architecture, where specific experts are activated for each input.	GPT, BERT, or other transformer-based dense models, which process all inputs through the entire network.



zetta floating-point operations (ZFLOPs):is a measure of computational cost .
Zero shot: Tasks where the model is required to perform without any prior specific training or examples for that task.

Background - Models (1)

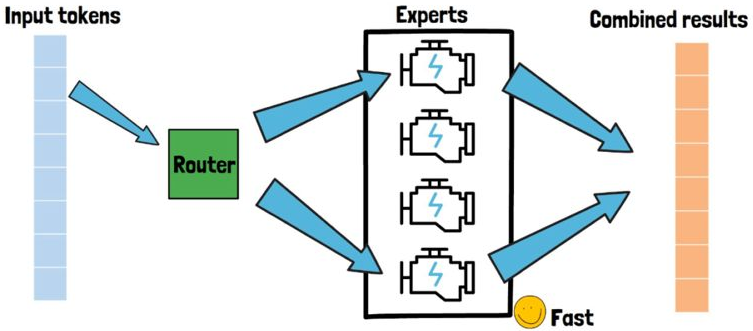
Dense vs. Sparse Models

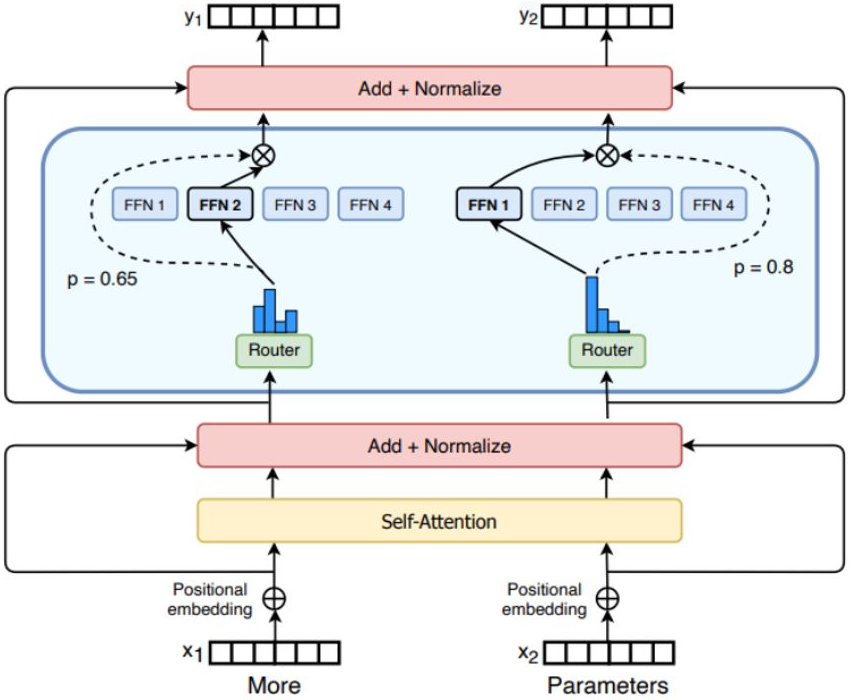


Layer Activation	All layers active for every input.	Only selected experts activated per input.
Parameter Usage	All parameters used always.	Only a fraction of parameters used.
Routing	No routing; sequential processing.	Dynamic routing selects relevant experts.
Scalability	Expensive to scale; compute grows linearly.	Scales efficiently with added experts.
Compute Efficiency	Inefficient for large-scale tasks.	Highly efficient, reduces redundant compute.
Examples	GPT-3, BERT	Switch Transformer, GLaM

Background - Models (2)

Sparsely Gated Mixture-of-Experts (MoE) – How it works





Models

Autoregressive (decoder-only) transformer models.

GPT-3 (dense)					Ours (dense)					Ours (MoE)				
size	*cost*	l	h	e	*size*	*cost*	l	h	e	*size*	*cost*	l	h	e
125M	0.36	12	768	–	125M	0.36	12	768	–	15B	0.43	12	768	512
355M	1.06	24	1024	–	355M	1.06	24	1024	–	52B	1.30	24	1024	512
760M	2.13	24	1536	–		—					—			
1.3B	3.57	24	2048	–	1.3B	3.57	24	2048	–	207B	4.53	24	2048	512
2.7B	7.08	32	2560	–	2.7B	7.08	32	2560	–		—			
6.7B	17.12	32	4096	–	6.7B	17.12	32	4096	–	1.1T	22.27	32	4096	512
13B	32.67	40	5120	–	13B	32.67	40	5120	–		—			
175B	430.17	96	12288	–		—					—			

Pretraining data

The models pretrainon a union of six English- language datasets.

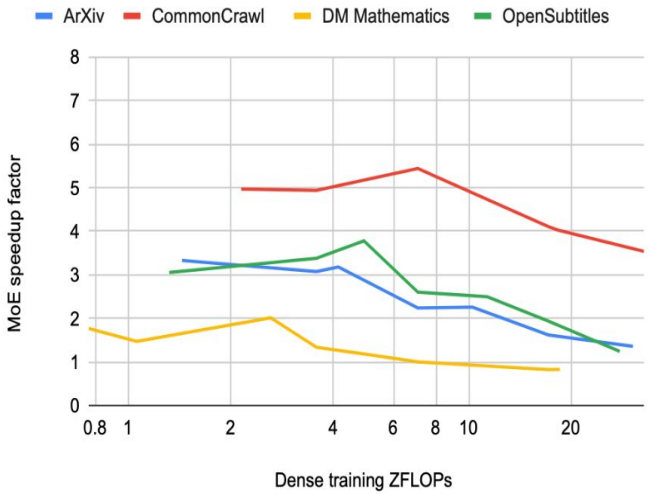
With totalling 112B tokens corresponding to 453GB and they are:

1. BookCorpus
2. English Wikipedia
3. OpenWebText
4. CC-Stories
5. English CC100
6. CC-News

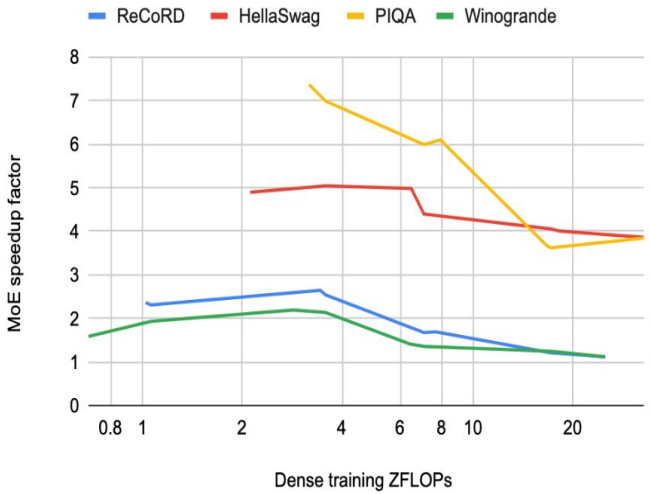
The data had been trained using the same Byte-Pair Encoding (BPE) as GPT-2 and RoBERTa with a vocabulary of 50K subword units.

*RoBERTa improves BERT's training, excelling in NLP tasks like summarization.

Evaluation



(a) Language modeling (the Pile)



(b) Zero-shot priming

Figure 3: **Estimate of how much more efficient MoEs are relative to dense models in representative datasets.**

Results and Analysis (Definitions) (1)

- Language Modeling Perplexity

Metric measures how well a LM predicts a sample of text

$$PPL = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i \mid w_1, \ldots, w_{i-1})\right)$$

$$PPL \propto 1/P_{\text{avg}}$$

- Downstream Task Performance
  - Zero-Shot Learning → Making predictions **without explicit training** by leveraging prior knowledge
  - Few-Shot Learning → Making predictions with training from **minimal labeled data**
  - Supervised Fine-Tuning → Model is trained with **large amount of labeled data**

- Data Domains
  - In-domain modeling
  - Out-of-domain modeling

Same/highly similar domain

Different domains

Results and Analysis (2)

I. Language Modeling Perplexity

Perplexity for dense vs. MoE models across subsets of "The Pile."

In-Domain Gains(validation):

- MoE: 8-16x compute savings for in-domain datasets.
- MoE models achieve lower perplexity on "The Pile" dataset subsets

Out-of-Domain Gains(the pile):

- 2-4x compute savings but less pronounced

Results and Analysis (2)

II. Downstream Task Performance

1. Zero-Shot Learning

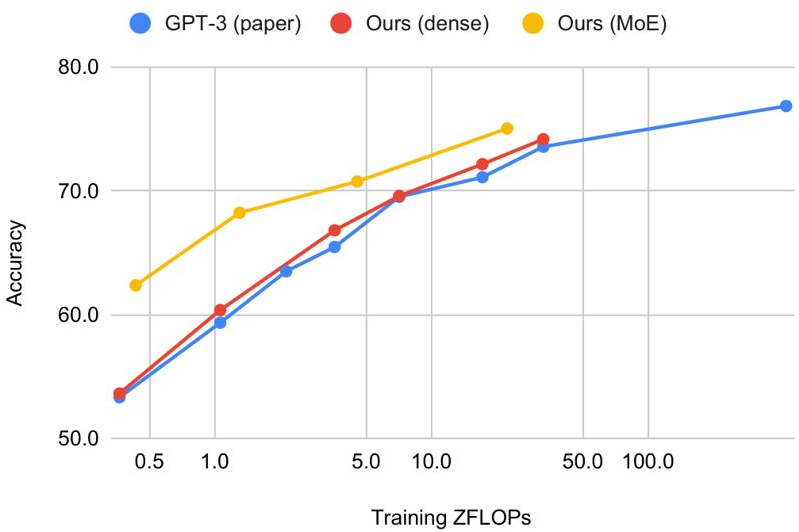


Fig. 4: Zero-shot priming accuracy averaged across 6 tasks as a function of compute cost.

2. Few-Shot Learning

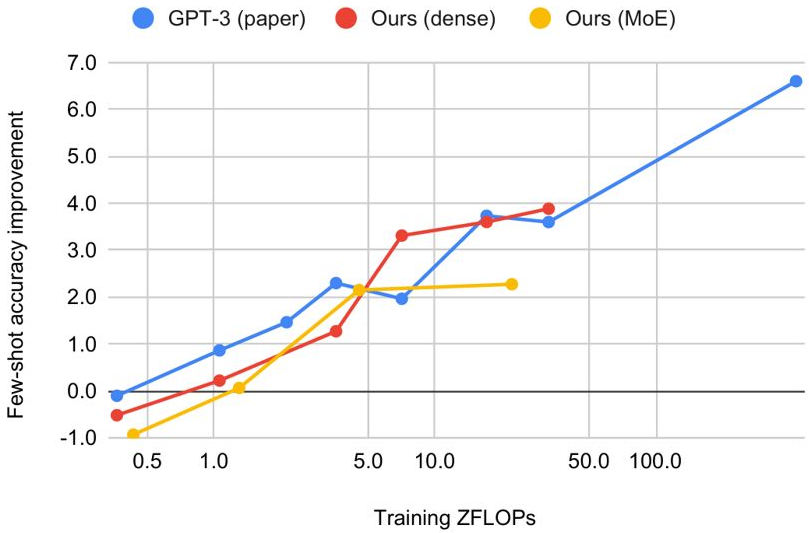


Fig. 5: Absolute accuracy improvement going from zero-shot to few-shot

Results and Analysis (3)

Downstream Task Performance (2)

3. Supervised Fine-Tuning

- ❑ Fine-Tuning Challenges for MoEs

Both dense and MoE models benefit from fine-tuning, (but MoE results are inconsistent)

- Fine-tuning works well on tasks like **SC**, **BQ**, **MN**
- MoEs underperform on tasks like **HS**, **PI**, **WG**

- ❑ **Insight:** Fine-tuning methods for dense models may not be optimal for sparse MoEs.

Table 4: Fully supervised fine-tuning accuracy compared with zero-shot accuracy.

		Ours (Dense)				Ours (MoE)		
		125M	355M	1.3B	2.7B	15B	52B	207B
SC	zero-shot	66.0	71.0	76.8	78.2	73.6	75.9	78.1
	fine-tune	87.8	89.5	93.8	97.0	80.3	84.9	80.9
OB	zero-shot	35.4	42.0	49.4	49.6	42.0	51.0	50.8
	fine-tune	50.6	59.0	67.4	70.8	51.2	51.4	51.0
BQ	zero-shot	56.1	58.6	58.7	60.3	60.9	56.0	54.2
	fine-tune	73.2	75.2	79.6	84.6	71.6	75.3	77.5
MN	zero-shot	46.2	52.1	55.3	56.0	49.3	52.1	52.6
	fine-tune	80.9	84.3	84.1	88.9	77.7	81.2	78.7
SST-2	zero-shot	50.9	50.9	51.6	51.9	51.6	50.9	50.9
	fine-tune	92.9	92.9	94.8	93.4	89.3	90.1	90.3
HS	zero-shot	33.7	46.2	58.4	65.9	53.2	64.9	70.5
	fine-tune	50.7	64.8	74.1	90.0	37.3	45.4	42.2
PI	zero-shot	65.3	70.6	74.6	76.6	74.3	76.8	78.2
	fine-tune	68.2	71.7	71.2	80.3	66.3	66.1	68.3
WG	zero-shot	52.1	54.2	58.1	61.4	53.4	57.4	60.9
	fine-tune	65.7	63.3	67.4	69.5	50.2	56.0	50.4

SC:StoryCloze, **OB**:OpenBookQA, **BQ**:BoolQ, **MN**:MNLI, **HS**:HellaSwag, **PI**:PIQA, **WG**:WinoGrande

Summary

Aspect	Dense Models	Sparse Mixture-of-Experts (MoE) Models
1. Performance	High performance with scaling	Equivalent or superior performance
2. Compute Efficiency	Computationally expensive	8-16x less compute (in-domain)
3. Out-of-Domain Efficiency	High compute requirements	2-4x compute savings
4. Key Innovations	Larger model sizes	MoE layers, routing gates
5. Future Work	Scaling challenges remain	Optimize fine-tuning and generalization


The technology:
GPS + RFID