Mixture of Experts (MOE) in LLM

Yao Dou and Sanath Kamath

Mixture of Experts (MOE) in LLM

Yao Dou and Sanath Kamath

Independent mirrored components in a model

This MOE idea is introduced in 1990s.

1991

Adaptive Mixtures of Local Experts

Robert A. Jacobs
Michael I. Jordan
Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139 USA

Steven J. Nowlan
Geoffrey E. Hinton
Department of Computer Science, University of Toronto, Toronto, Canada M5S 1A4

1993

Hierarchical mixtures of experts and the EM algorithm

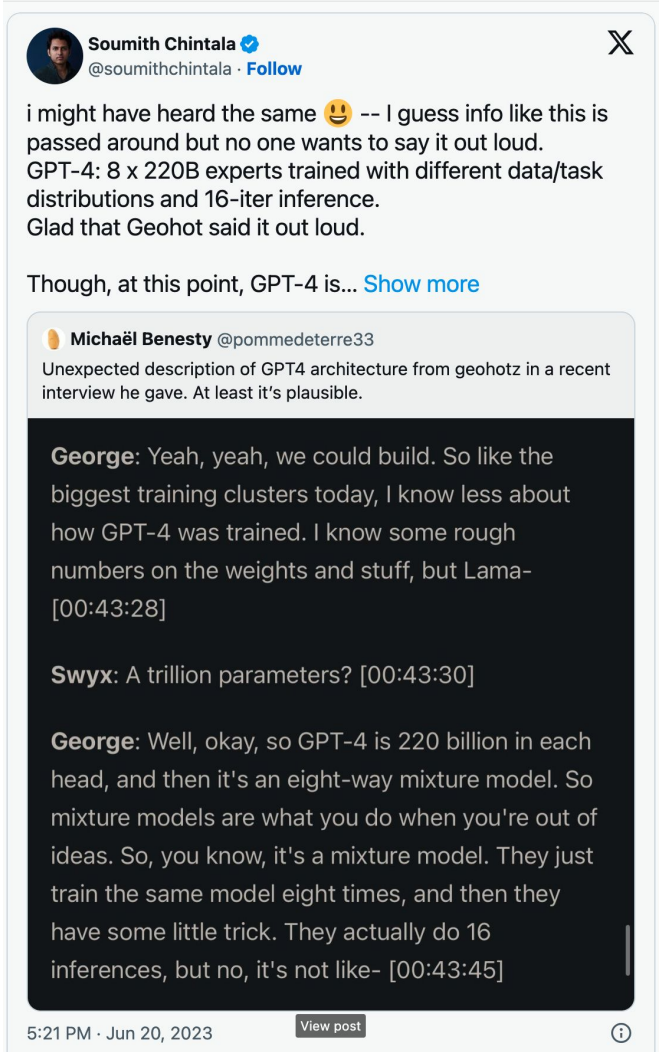
Michael I. Jordan
Department of Brain and Cognitive Sciences
MIT
Cambridge, MA 02139

Robert A. Jacobs
Department of Psychology
University of Rochester
Rochester, NY 14627

Mixtral 8 * 7B

Mixtral of Experts

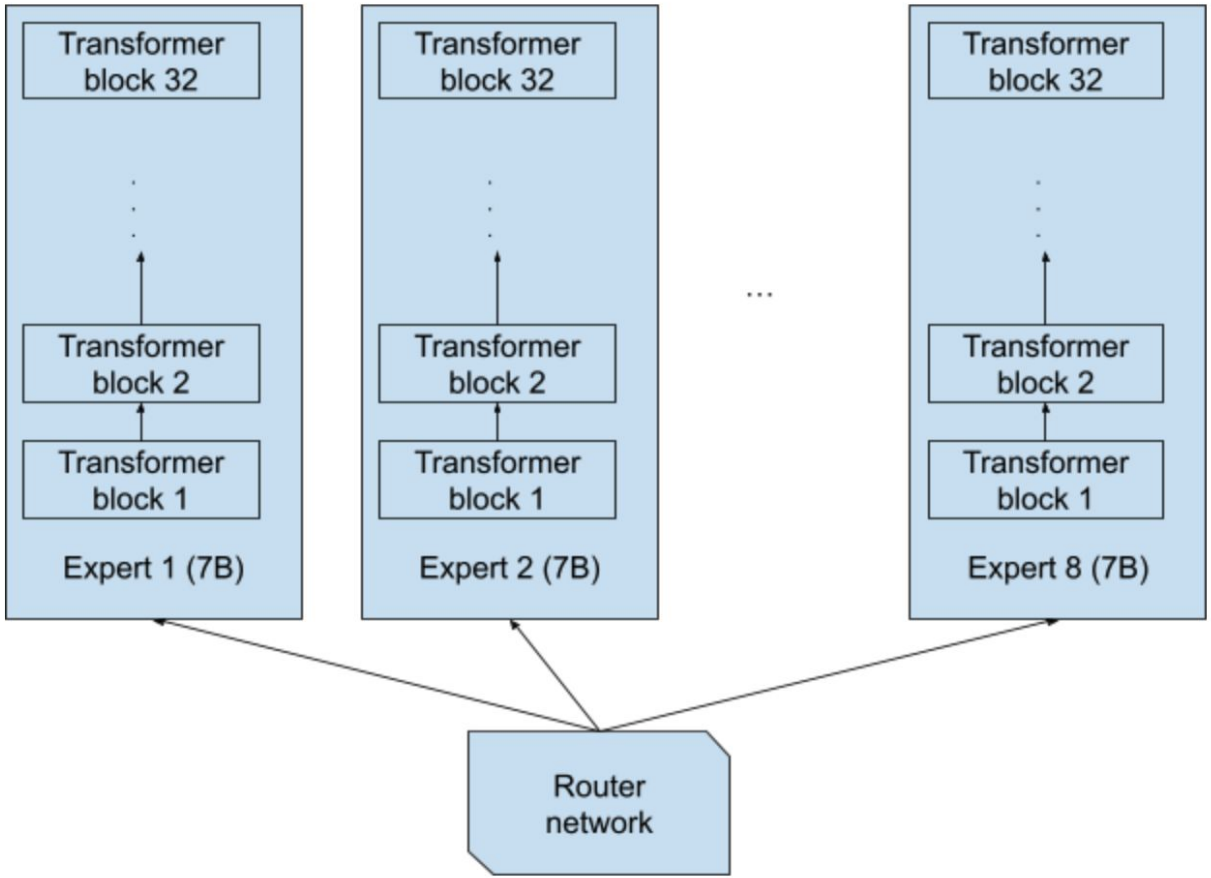
Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch,
Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas,
Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour,
Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux,
Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao,
Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, William El Sayed


Soumith Chintala
@soumithchintala · Follow
i might have heard the same 😃 -- I guess info like this is passed around but no one wants to say it out loud.
GPT-4: 8 x 220B experts trained with different data/task distributions and 16-iter inference.
Glad that Geohot said it out loud.
Though, at this point, GPT-4 is... Show more
Michaël Benesty @pommedeterre33
Unexpected description of GPT4 architecture from geohotz in a recent interview he gave. At least it's plausible.
George: Yeah, yeah, we could build. So like the biggest training clusters today, I know less about how GPT-4 was trained. I know some rough numbers on the weights and stuff, but Lama- [00:43:28]
Swyx: A trillion parameters? [00:43:30]
George: Well, okay, so GPT-4 is 220 billion in each head, and then it's an eight-way mixture model. So mixture models are what you do when you're out of ideas. So, you know, it's a mixture model. They just train the same model eight times, and then they have some little trick. They actually do 16 inferences, but no, it's not like- [00:43:45]
5:21 PM · Jun 20, 2023
View post

GPT-4: 8 * 220B

A plausible design of MOE

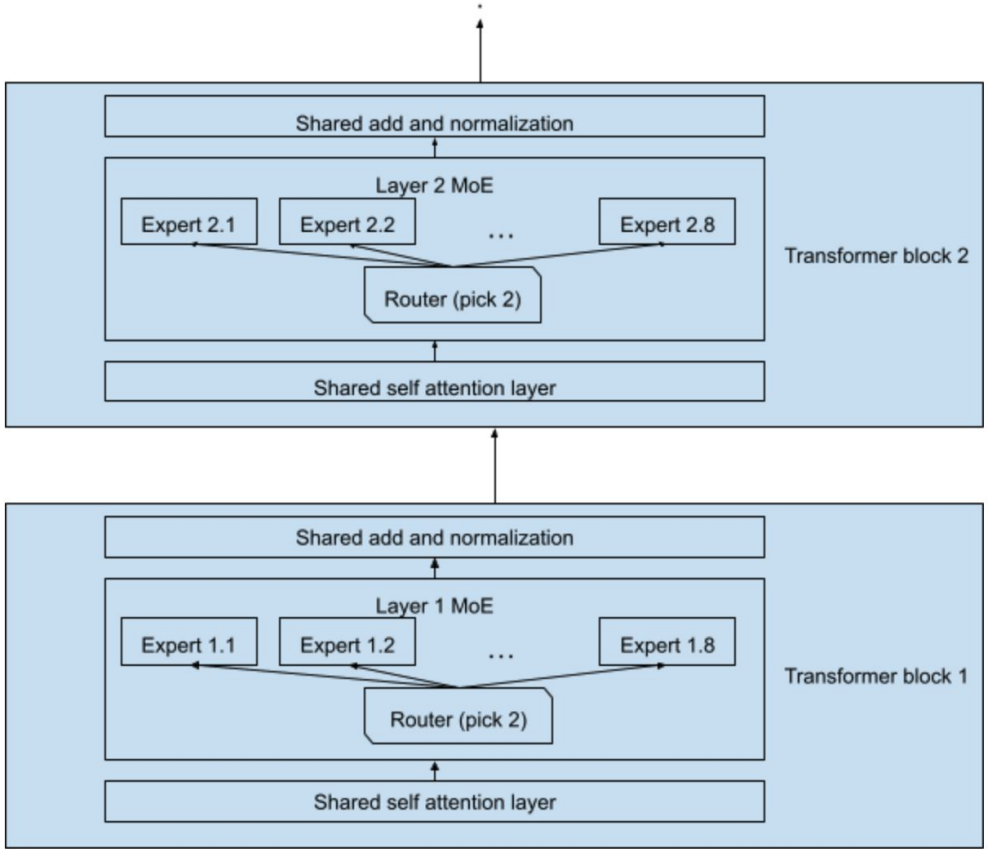
Expert is a single model



https://developer.nvidia.com/blog/applying-mixture-of-experts-in-llm-architectures/

The **actual** design of MOE

Expert is the FFN layer in each Transformer block.

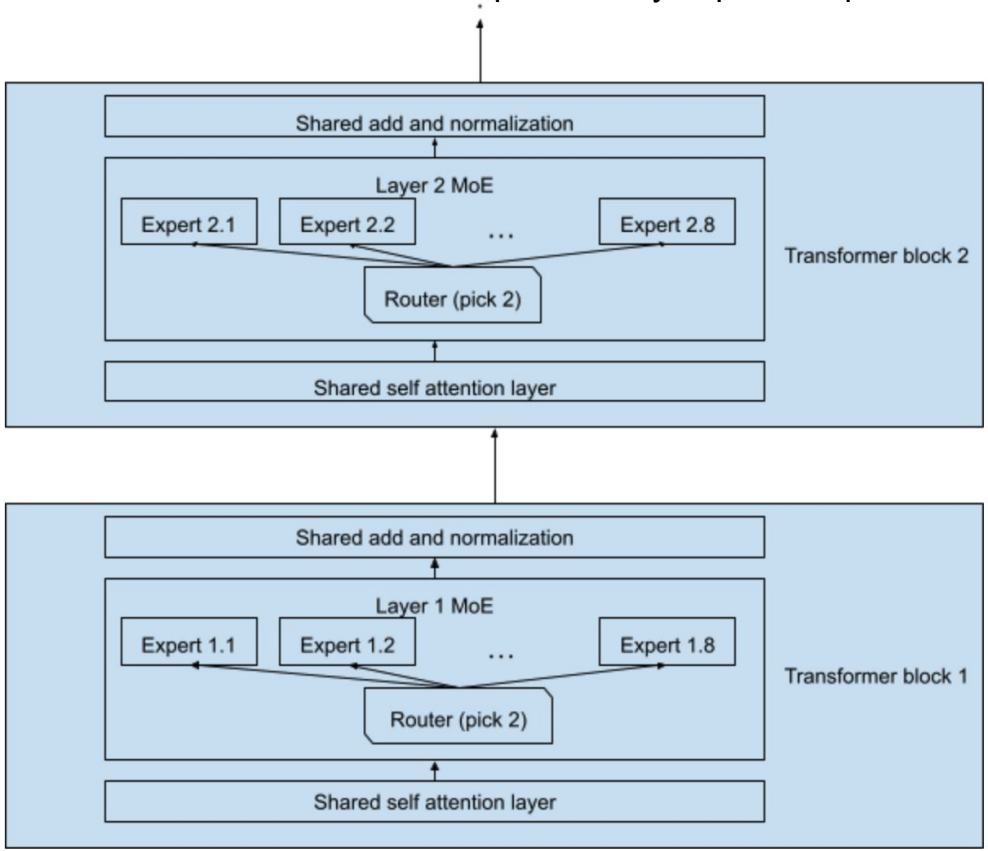


https://developer.nvidia.com/blog/applying-mixture-of-experts-in-llm-architectures/

The **actual** design of MOE

Expert is the FFN layer in each Transformer block.

Reason: FFN is the most computationally expensive part in a Transformer (most parameters)

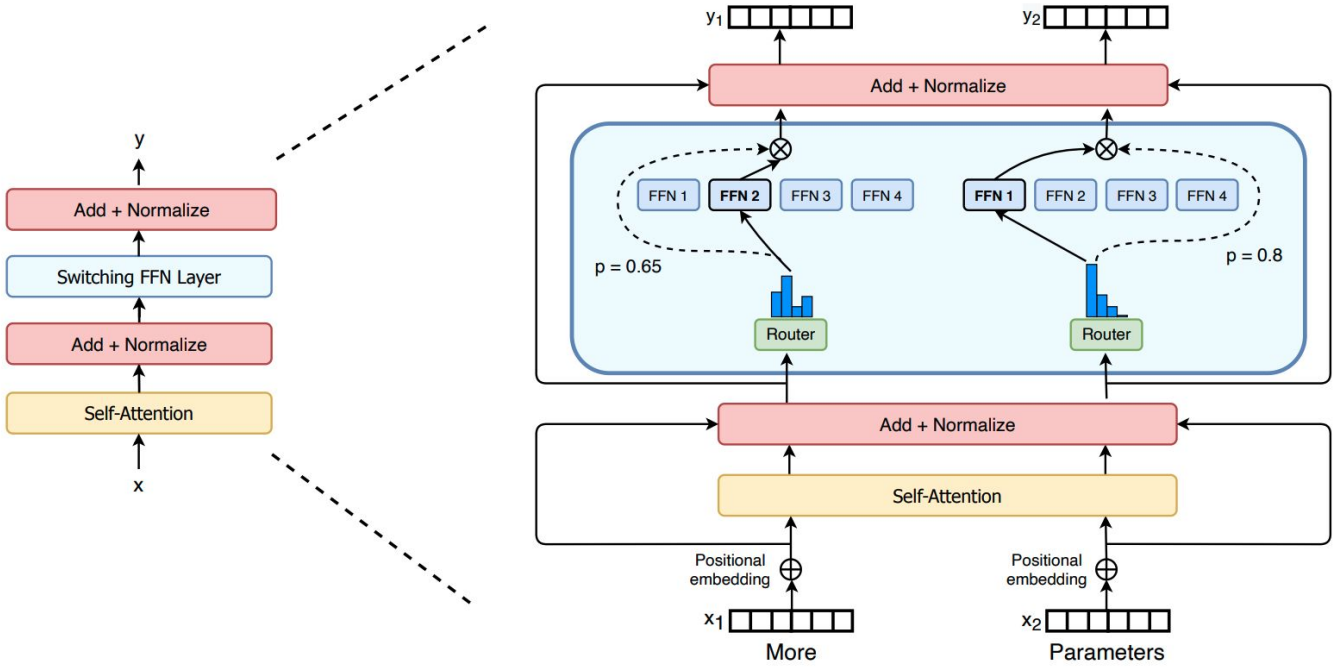


For instance, in the PaLM [5] model with the parameter number of 540B, the 90% of these parameters are within its FFN layers

https://developer.nvidia.com/blog/applying-mixture-of-experts-in-llm-architectures/

Why MOE?

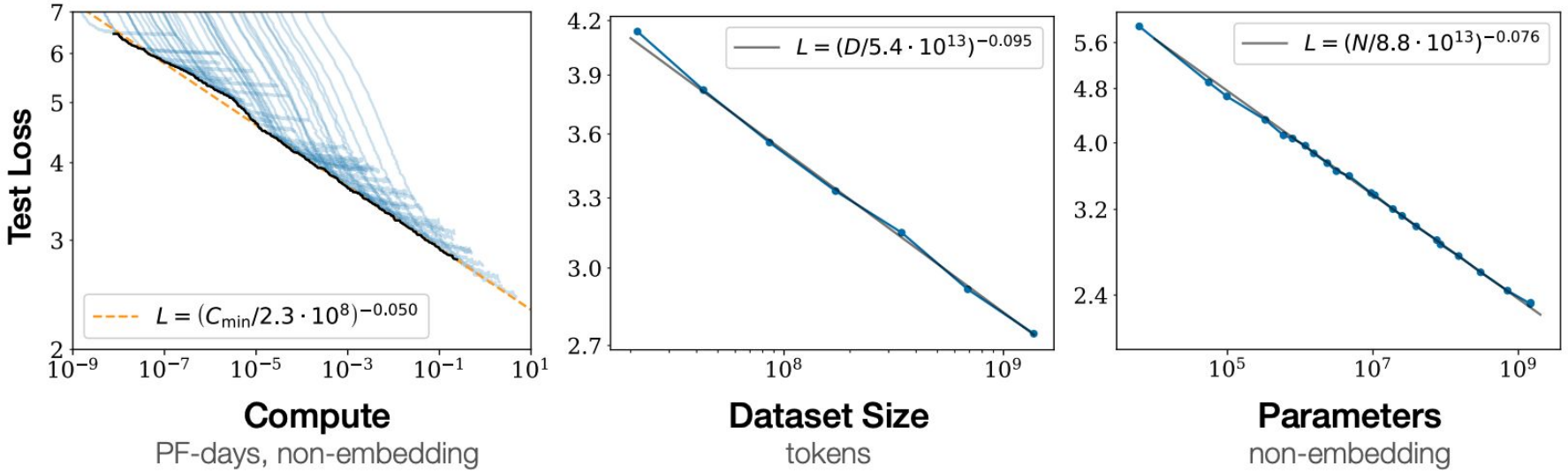
1. Scaling Parameter Counts
2. Specialization



Why MOE?

1. Scaling Parameter Counts

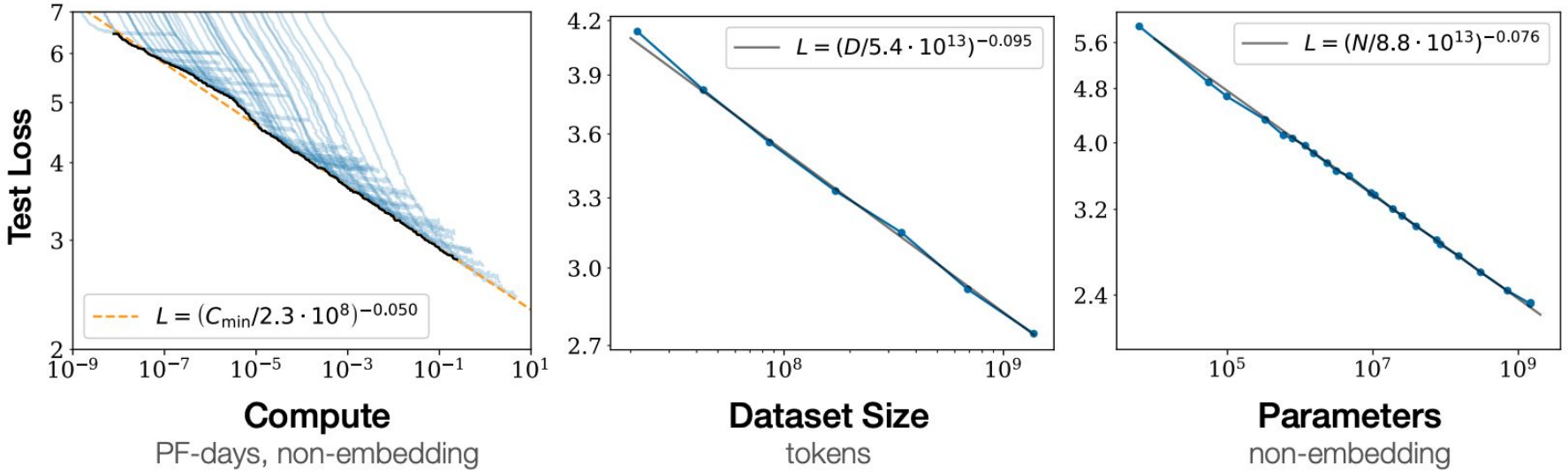
From OpenAI's "Scaling Laws for Neural Language Models" 2020



Why MOE?

1. Scaling Parameter Counts

From OpenAI's "Scaling Laws for Neural Language Models" 2020



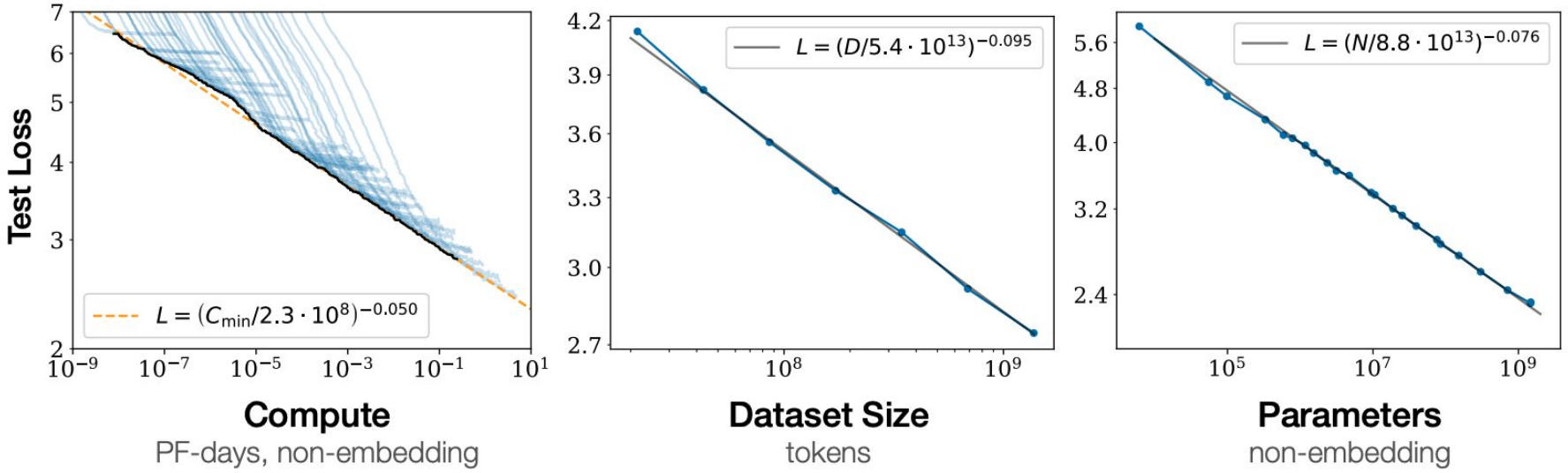
For Dense model, the compute per token scales together with the parameter counts

MOE decouples this by keeping the compute per token fix while scaling the parameters (more experts)

Why MOE?

1. Scaling Parameter Counts

From OpenAI's "Scaling Laws for Neural Language Models" 2020



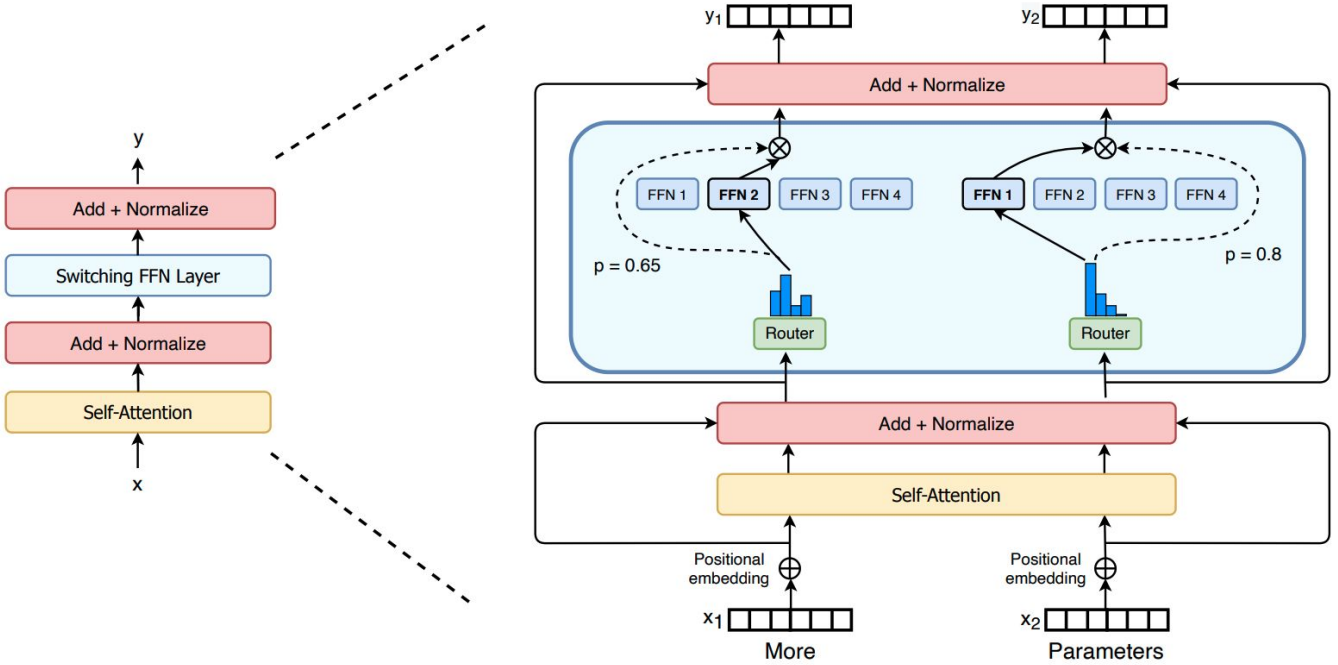
Basically same compute as Dense model, with just more memory.

For Dense model, the compute per token scales together with the parameter counts

MOE decouples this by keeping the compute per token fix while scaling the parameters (more experts)

Why MOE?

2. Specialization



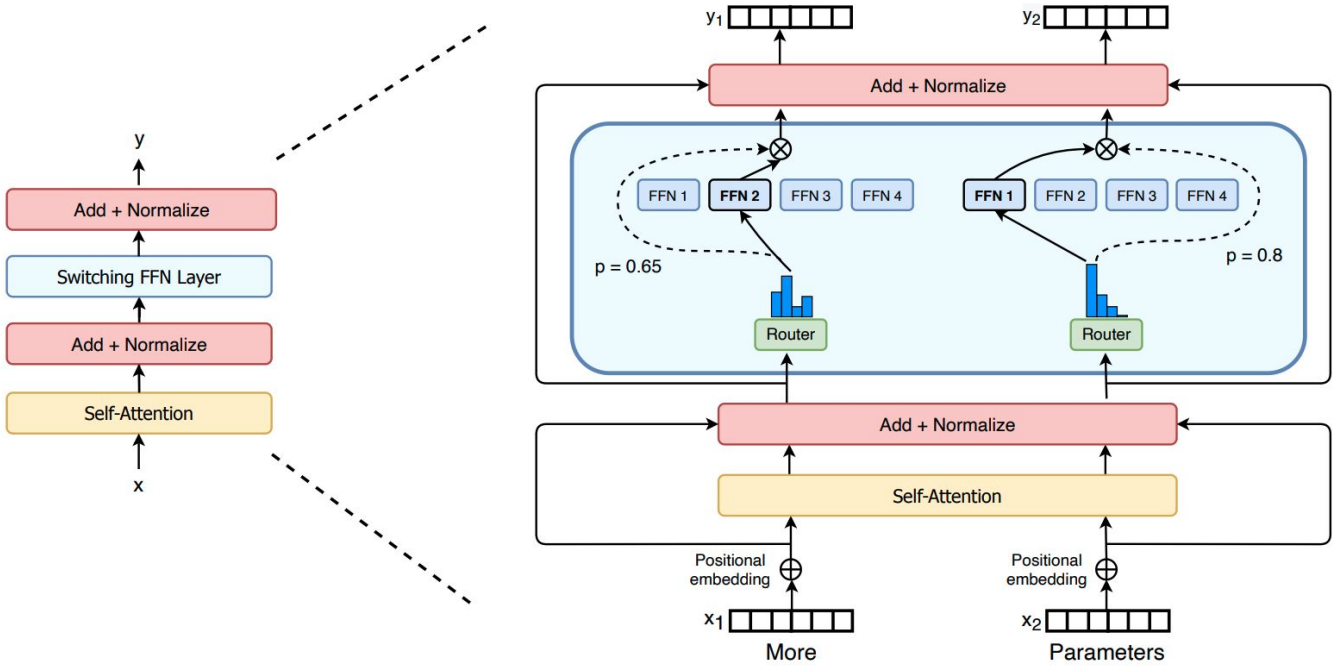
Why MOE?

2. Specialization

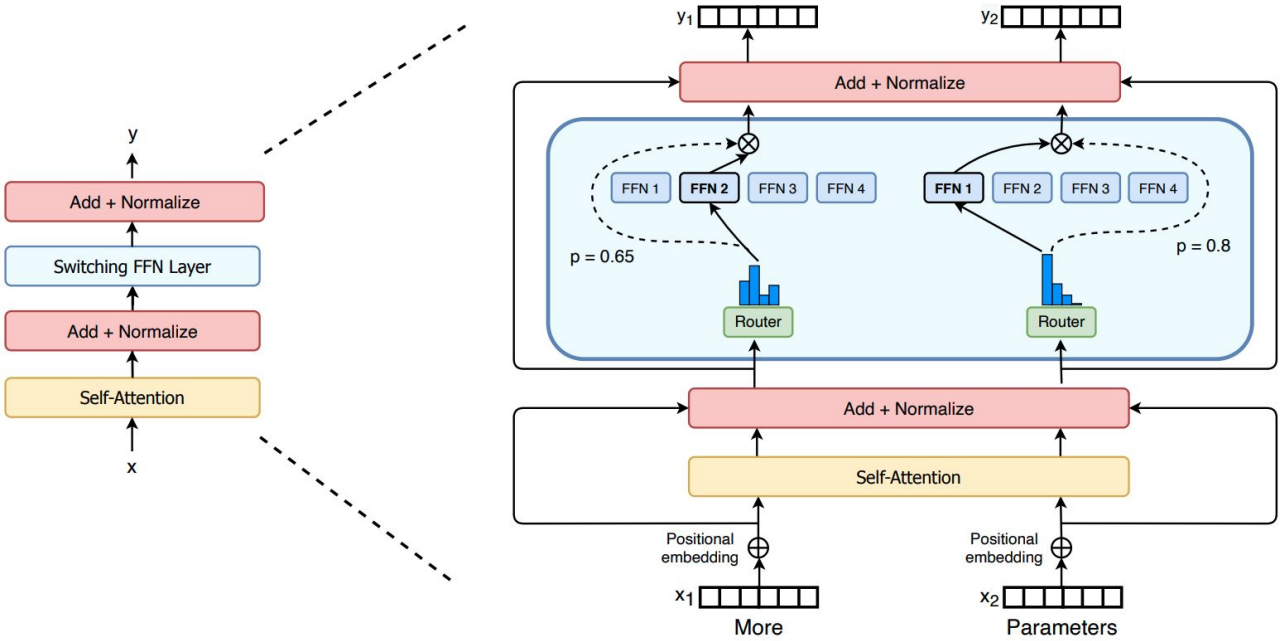
The brain is a mixture of experts too! (Why and how the brain weights contributions from a mixture of experts)

Different input token can go to different experts

We may have experts for number (1, 2, 3), experts for code, experts for instruction tokens



Mixture of Experts



Limitations:

- Complexity
- Communication Costs
- Training instability

Routing

- The router takes an input x and sends it to the top-k experts based on a “gate value” which represents how much that expert’s opinion matters
- Let W_r be the router variable
- Step 1: Calculate logits h(x)
- Step 2: Normalize via softmax over the N experts to compute the gate values p_i
- Step 3: Let T be the top-k experts. Compute the output y as a weighted sum of the gate values and the expert computation

$$h(x) = W_r \cdot x$$

$$p_i(x) = \frac{e^{h(x)_i}}{\sum_j^N e^{h(x)_j}}.$$

$$y = \sum_{i \in \mathcal{T}} p_i(x) E_i(x).$$

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
LIAMFEDUS@GOOGLE.COM

Barret Zoph*
BARRETZOPH@GOOGLE.COM

Noam Shazeer
NOAM@GOOGLE.COM
Google, Mountain View, CA 94043, USA

Switch Transformers: Scaling to Trillion Parameter Models with Simple and ~~Efficient Sparsity~~

William Fedus*
LIAMFEDUS@GOOGLE.COM

Barret Zoph*
BARRETZOPH@GOOGLE.COM

Noam Shazeer
NOAM@GOOGLE.COM
Google, Mountain View, CA 94043, USA

Switch Transformer

- Instead of routing to multiple experts, route to just one!
- Benefits:
  - Less router computation
  - Capacity of each expert can be at least halved
  - Simpler implementation and reduced communication costs
- What are some other benefits to routing to just one expert?

Switch Transformer

- Instead of routing to multiple experts, route to just one!
- Benefits:
  - Less router computation
  - Capacity of each expert can be at least halved
  - Simpler implementation and reduced communication costs
- What are some other benefits to routing to just one expert?

Capacity and Load

- Expert Capacity: The number of tokens each expert computes each batch
  - Low expert capacity -> Dropped tokens
  - High expert capacity -> Wasted computation/memory
  - Q: Does the number of experts have an effect on the number of dropped tokens?
- A good solution for load balancing is to add an auxiliary loss
  - Alpha is a constant hyperparameter and N is the number of experts (Why multiply by N?)
  - f_i is the fraction of tokens dispatched to expert i
  - P_i is the fraction of the router probability allocated to expert i
  - The loss is minimized when all f_i and P_i are equal to 1/N

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$

Capacity and Load

- Expert Capacity: The number of tokens each expert computes each batch
  - Low expert capacity -> Dropped tokens
  - High expert capacity -> Wasted computation/memory
  - Q: Does the number of experts have an effect on the number of dropped tokens?
- A good solution for load balancing is to add an auxiliary loss
  - Alpha is a constant hyperparameter and N is the number of experts (Why multiply by N?)
  - f_i is the fraction of tokens dispatched to expert i
  - P_i is the fraction of the router probability allocated to expert i
  - The loss is minimized when all f_i and P_i are equal to 1/N

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$

Implementation Details

- The Switch-Base and Switch-Large models are trained on the Colossal Clean Crawled Corpus (C4) dataset and is “FLOP matched” to the T5-Base and T5-Large models respectively
- They also train a Switch-XXL (395B) and Switch-C (1571B)
- The MoE Transformer uses top-2 routing and therefore has higher FLOPs
- For multilingual tasks, the authors use the mC4 dataset

Results (Pre-Training)

Model	Capacity Factor	Quality after 100k steps (↑) (Neg. Log Perp.)	Time to Quality Threshold (↓) (hours)	Speed (↑) (examples/sec)
T5-Base	—	-1.731	Not achieved†	1600
T5-Large	—	-1.550	131.1	470
MoE-Base	2.0	-1.547	68.7	840
Switch-Base	2.0	-1.554	72.8	860
MoE-Base	1.25	-1.559	80.7	790
Switch-Base	1.25	-1.553	65.0	910
MoE-Base	1.0	-1.572	80.1	860
Switch-Base	1.0	-1.561	**62.8**	1000
Switch-Base+	1.0	**-1.534**	67.6	780

Table 1: Benchmarking Switch versus MoE. Head-to-head comparison measuring per step and per time benefits of the Switch Transformer over the MoE Transformer and T5 dense baselines. We measure quality by the negative log perplexity and the time to reach an arbitrary chosen quality threshold of Neg. Log Perp.=-1.50. All MoE and Switch Transformer models use 128 experts, with experts at every other feed-forward layer. For Switch-Base+, we increase the model size until it matches the speed of the MoE model by increasing the model hidden-size from 768 to 896 and the number of heads from 14 to 16. All models are trained with the same amount of computation (32 cores) and on the same hardware (TPUv3). Further note that all our models required pre-training beyond 100k steps to achieve our level threshold of -1.50. † T5-Base did not achieve this negative log perplexity in the 100k steps the models were trained.

Results (Pre-Training)

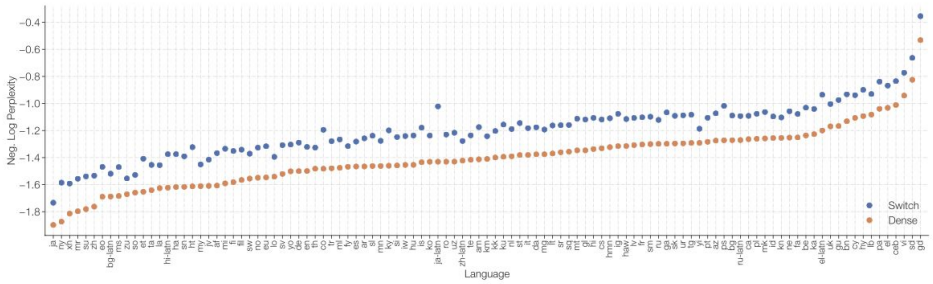


Figure 7: Multilingual pre-training on 101 languages. Improvements of Switch T5 Base model over dense baseline when multi-task training on 101 languages. We observe Switch Transformers to do quite well in the multi-task training setup and yield improvements on all 101 languages.

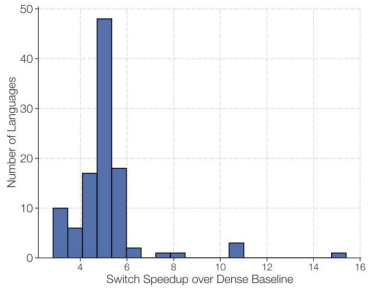


Figure 8: Multilingual pre-training on 101 languages. We histogram for each language, the step speedup of Switch Transformers over the FLOP matched T5 dense baseline to reach the same quality. Over all 101 languages, we achieve a mean step speed-up over mT5-Base of 5x and, for 91% of languages, we record a 4x, or greater, speedup to reach the final perplexity of mT5-Base.

Results (Pre-Training)

Model	Parameters	FLOPs/seq	d_{model}	FFN_{GEGLU}	d_{ff}	d_{kv}	Num. Heads
T5-Base	0.2B	124B	768	✓	2048	64	12
T5-Large	0.7B	425B	1024	✓	2816	64	16
T5-XXL	11B	6.3T	4096	✓	10240	64	64
Switch-Base	7B	124B	768	✓	2048	64	12
Switch-Large	26B	425B	1024	✓	2816	64	16
Switch-XXL	395B	6.3T	4096	✓	10240	64	64
Switch-C	1571B	890B	2080		6144	64	32

Model	Expert Freq.	Num. Layers	Num Experts	Neg. Log Perp. @250k	Neg. Log Perp. @ 500k
T5-Base	–	12	–	-1.599	-1.556
T5-Large	–	24	–	-1.402	-1.350
T5-XXL	–	24	–	-1.147	-1.095
Switch-Base	1/2	12	128	-1.370	-1.306
Switch-Large	1/2	24	128	-1.248	-1.177
Switch-XXL	1/2	24	64	**-1.086**	**-1.008**
Switch-C	1	15	2048	-1.096	-1.043

Table 9: Switch model design and pre-training performance. We compare the hyper-parameters and pre-training performance of the T5 models to our Switch Transformer variants. The last two columns record the pre-training model quality on the C4 data set after 250k and 500k steps, respectively. We observe that the Switch-C Transformer variant is 4x faster to a fixed perplexity (with the same compute budget) than the T5-XXL model, with the gap increasing as training progresses.

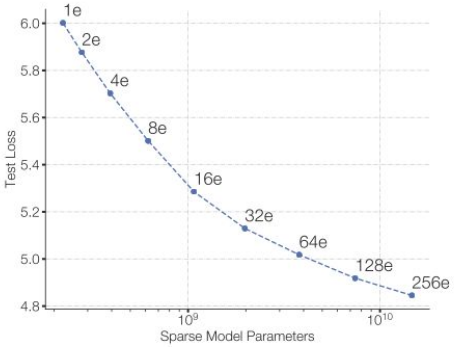
Results (Fine-Tuning)

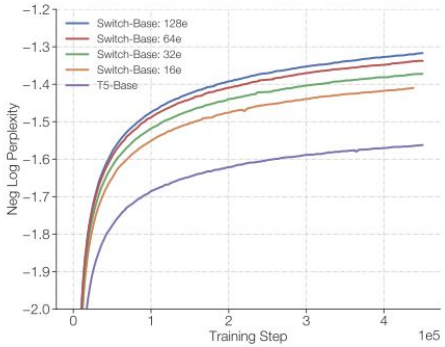
Model	GLUE	SQuAD	SuperGLUE	Winogrande (XL)
T5-Base	84.3	85.5	75.1	66.6
Switch-Base	**86.7**	**87.2**	**79.5**	**73.3**
T5-Large	87.8	88.1	82.7	79.1
Switch-Large	**88.5**	**88.6**	**84.7**	**83.0**

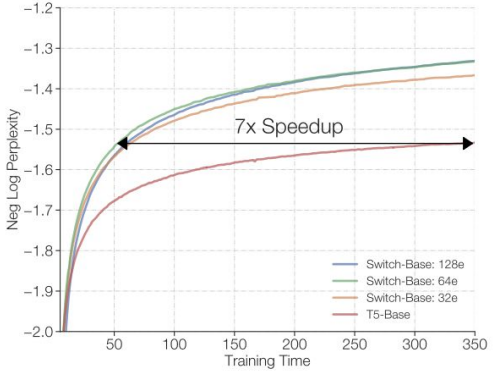
Model	XSum	ANLI (R3)	ARC Easy	ARC Chal.
T5-Base	18.7	51.8	56.7	**35.5**
Switch-Base	**20.3**	**54.0**	**61.3**	32.8
T5-Large	20.9	56.6	**68.8**	**35.5**
Switch-Large	**22.3**	**58.6**	66.0	**35.5**

Model	CB Web QA	CB Natural QA	CB Trivia QA
T5-Base	26.6	25.8	24.5
Switch-Base	**27.4**	**26.8**	**30.7**
T5-Large	27.7	27.6	29.5
Switch-Large	**31.3**	**29.5**	**36.9**

Scaling Properties







Scaling Properties

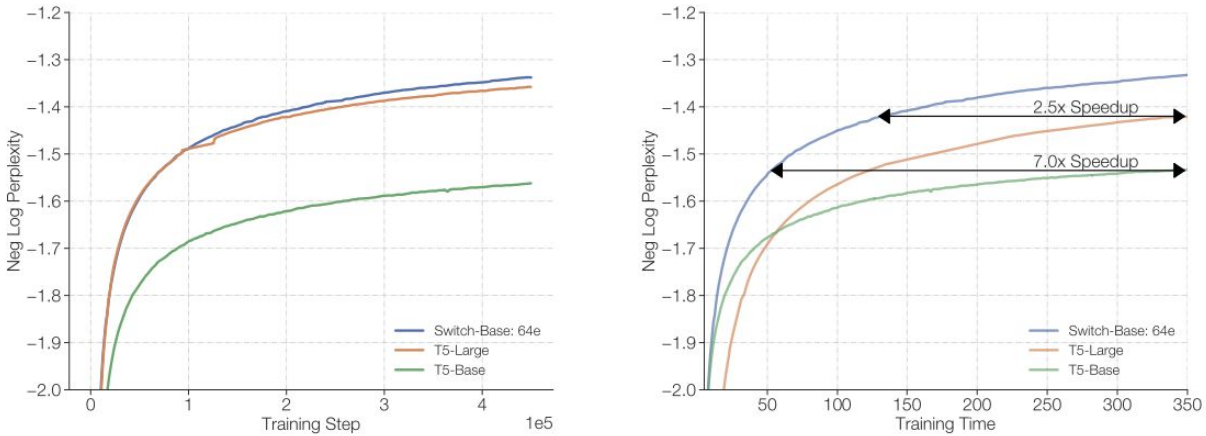


Figure 6: Scaling Transformer models with Switch layers or with standard dense model scaling. Left Plot: Switch-Base is more sample efficient than both the T5-Base, and T5-Large variant, which applies 3.5x more FLOPS per token. Right Plot: As before, on a wall-clock basis, we find that Switch-Base is still faster, and yields a 2.5x speedup over T5-Large.

Scaling Properties

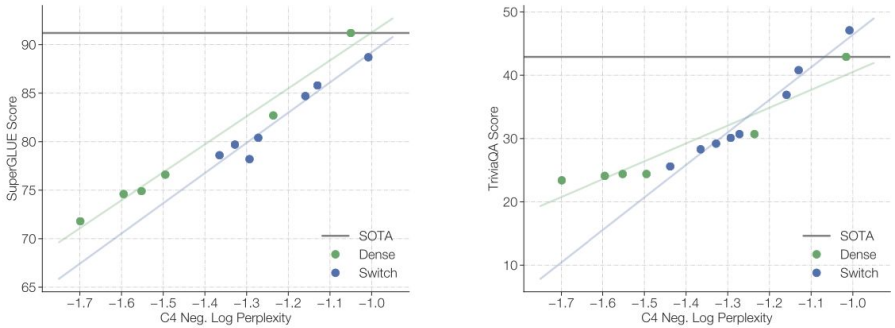


Figure 13: Upstream pre-trained quality to downstream model quality. We correlate the upstream performance with downstream quality on both SuperGLUE and TriviaQA (SOTA recorded without SSM), reasoning and knowledge-heavy benchmarks, respectively (validation sets). We find that, as with the baseline, the Switch model scales with improvements in the upstream pre-training task. For SuperGLUE, we find a loosely linear relation between negative log perplexity and the average SuperGLUE score. However, the dense model often performs better for a fixed perplexity, particularly in the large-scale regime. Conversely, on the knowledge-heavy task, TriviaQA, we find that the Switch Transformer may follow an improved scaling relationship – for a given upstream perplexity, it does better than a dense counterpart. Further statistics (expensive to collect and left to future work) would be necessary to confirm these observations.

Discussion

- The Switch Transformer is more sample efficient than comparable dense models
- While their Switch-XXL model faced training instability, their Switch-C model faced no instability at all according to the authors
- Q: How can we extend the Switch Transformer to multi-modal applications?

More on MOE

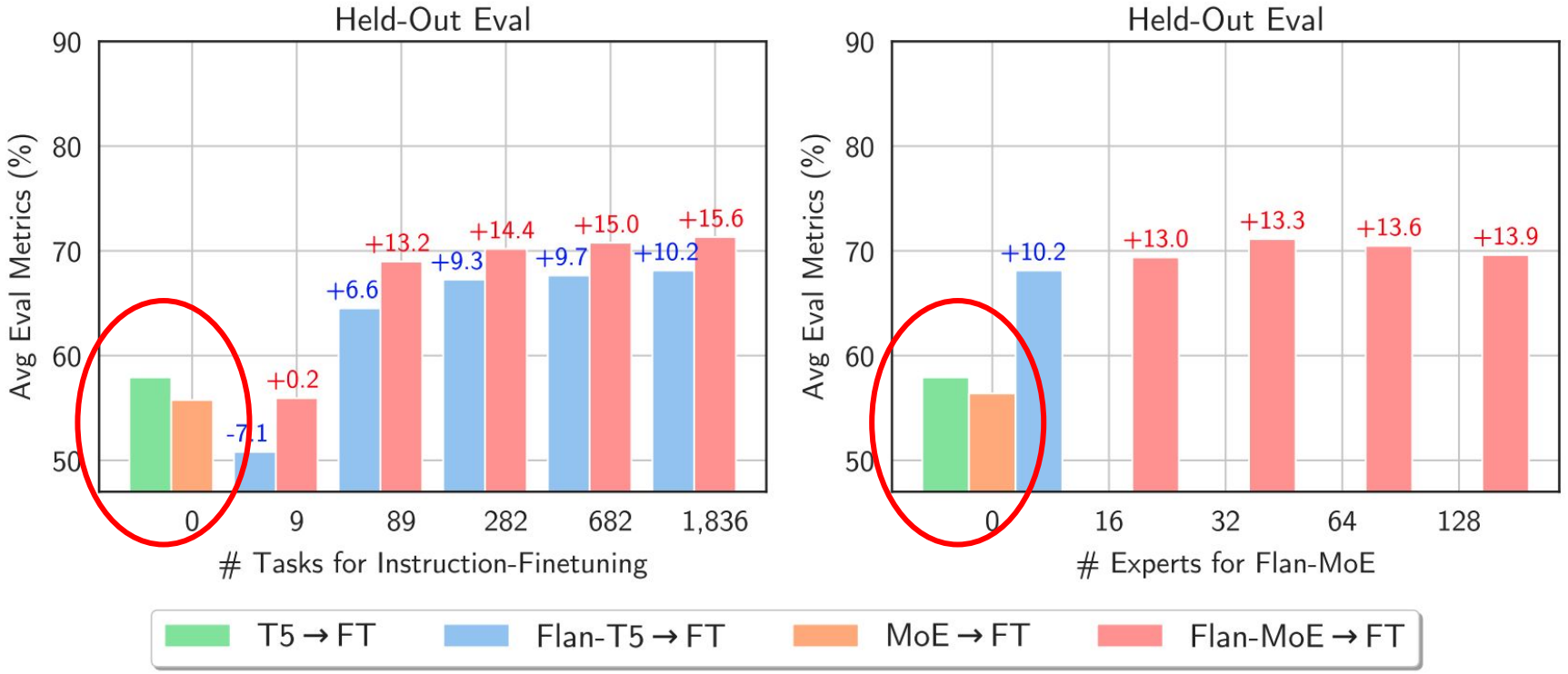
Mixture-of-Experts Meets Instruction Tuning: A Winning Combination for Large Language Models

2023 May

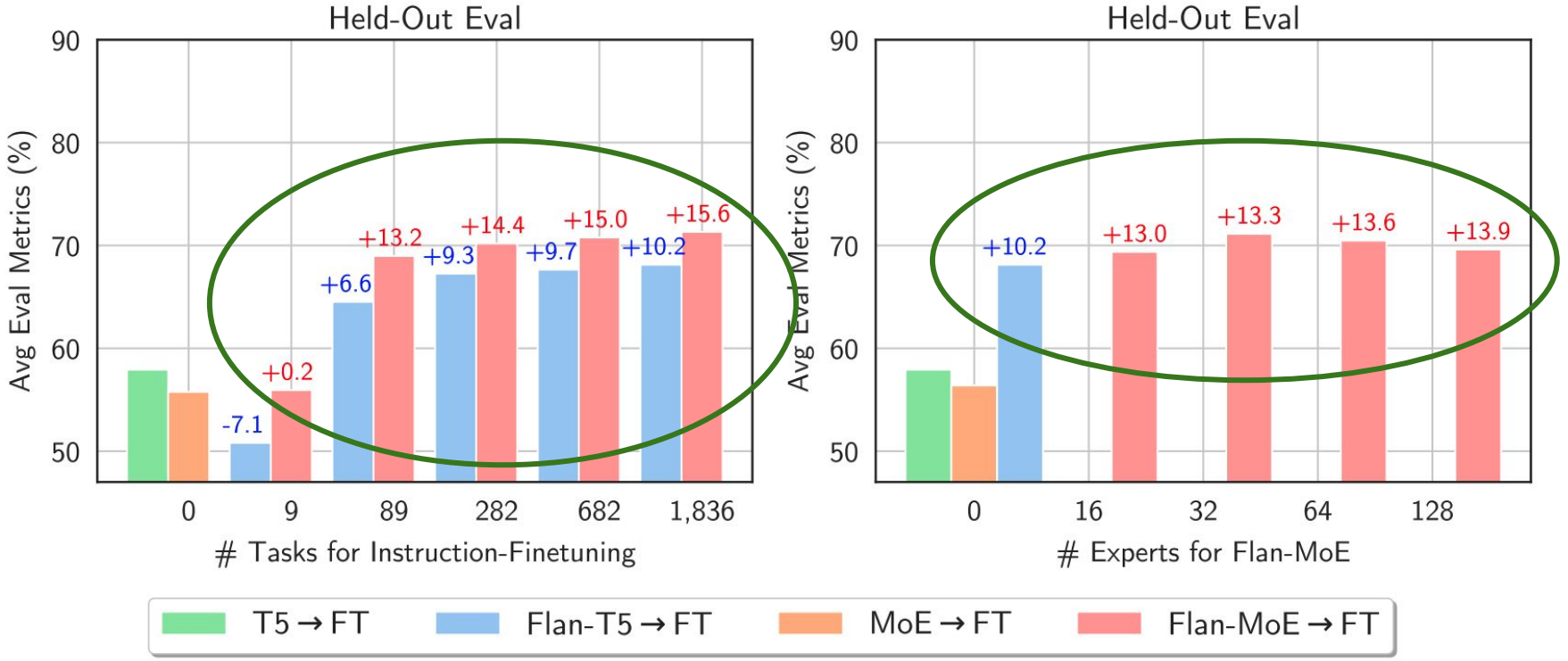
– We demonstrate that **in the absence of instruction tuning**, MoE models **fall short** in performance when compared to dense models on downstream tasks.

– We highlight that **when supplemented with instruction tuning**, MoE models **exceed** the performance of dense models on downstream tasks, as well as on held-out zero-shot and few-shot tasks.

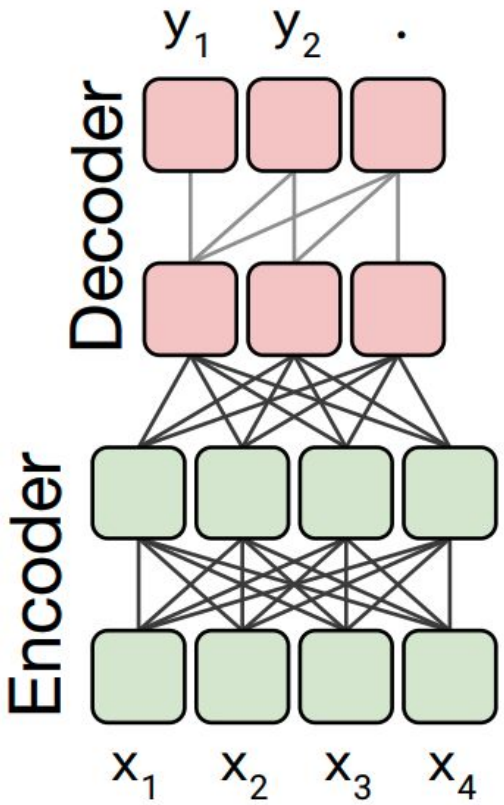
in the absence of instruction tuning, MoE models **fall short of** dense ones.

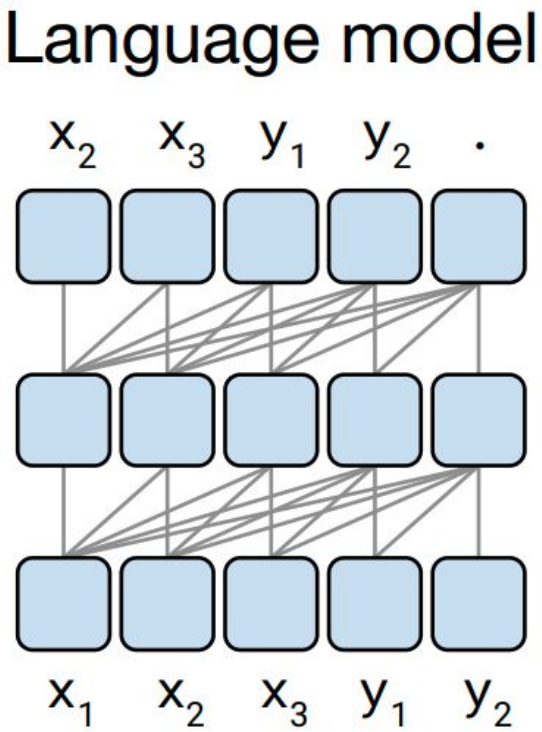


when supplemented with instruction tuning, MoE models **exceed** dense ones.



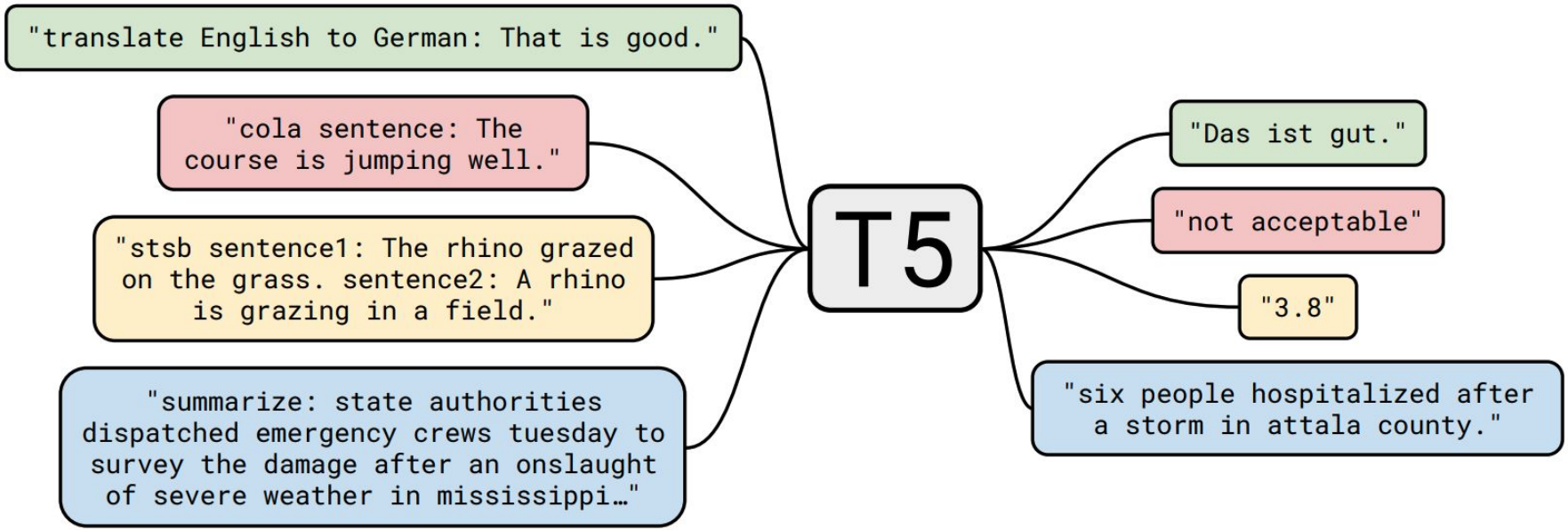
Review on T5 (encoder-decoder) model





Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (2020)

Review on T5 (encoder-decoder) model

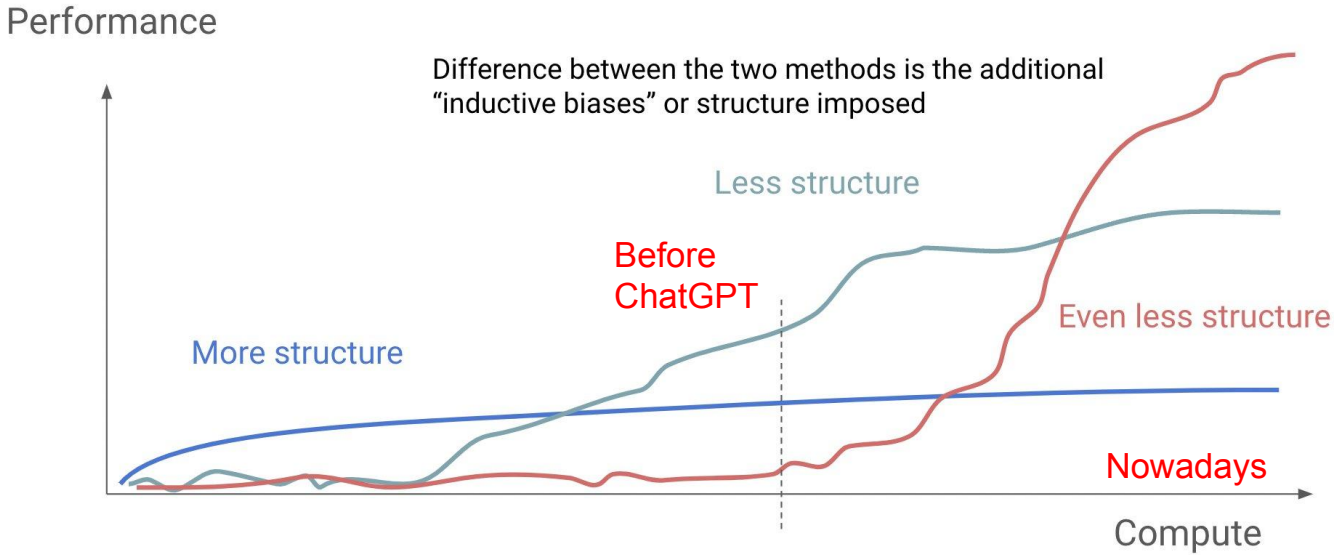


Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (2020)

Transit from T5 to GPT-x

Having separate parameters was natural when Transformer was first introduced with translation as the main evaluation task; input is in one language and output is in another.

Modern language models used in multiturn chat interfaces make this assumption awkward. Output in the current turn becomes the input of the next turn. Why treat them separately?



From: https://x.com/hwchung27/status/1800676312916656592

Vision LM (Encoder - decoder): LLaVA

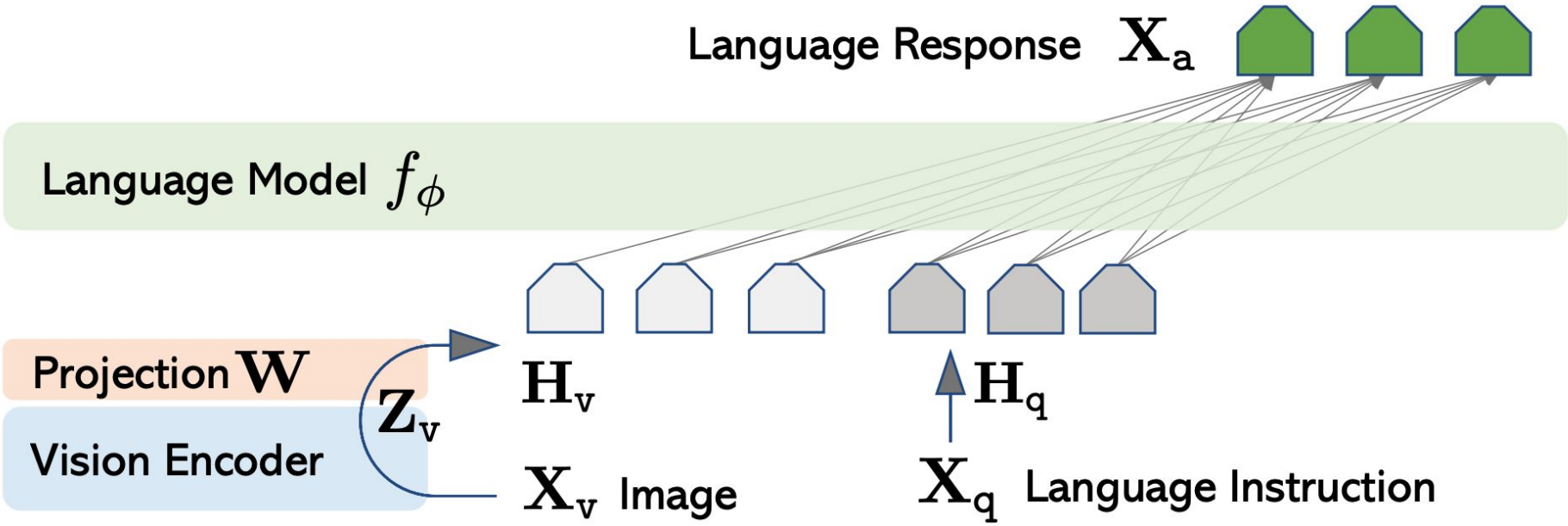
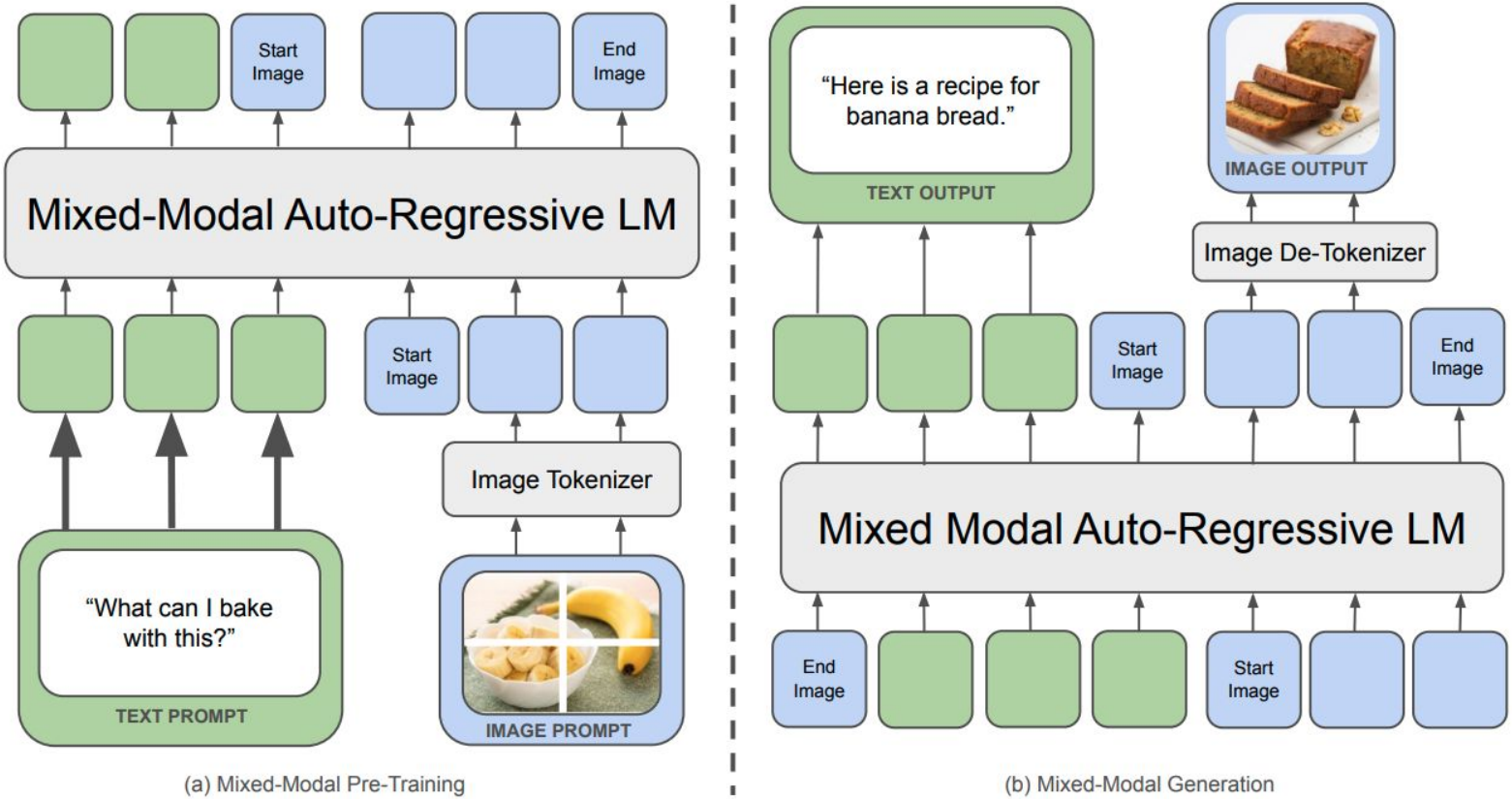


Figure 1: LLaVA network architecture.

Vision LM (decoder): Google Gemini and Meta Chameleon



GLM: the decoder MOE counterpart of the Switch Transformer

Model Name	Model Type	n_{params}	$n_{\text{act-params}}$
BERT	Dense Encoder-only	340M	340M
T5	Dense Encoder-decoder	13B	13B
GPT-3	Dense Decoder-only	175B	175B
Jurassic-1	Dense Decoder-only	178B	178B
Gopher	Dense Decoder-only	280B	280B
Megatron-530B	Dense Decoder-only	530B	530B
GShard-M4	MoE Encoder-decoder	600B	1.5B
Switch-C	MoE Encoder-decoder	1.5T	1.5B
GLaM (64B/64E)	MoE Decoder-only	1.2T	96.6B

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

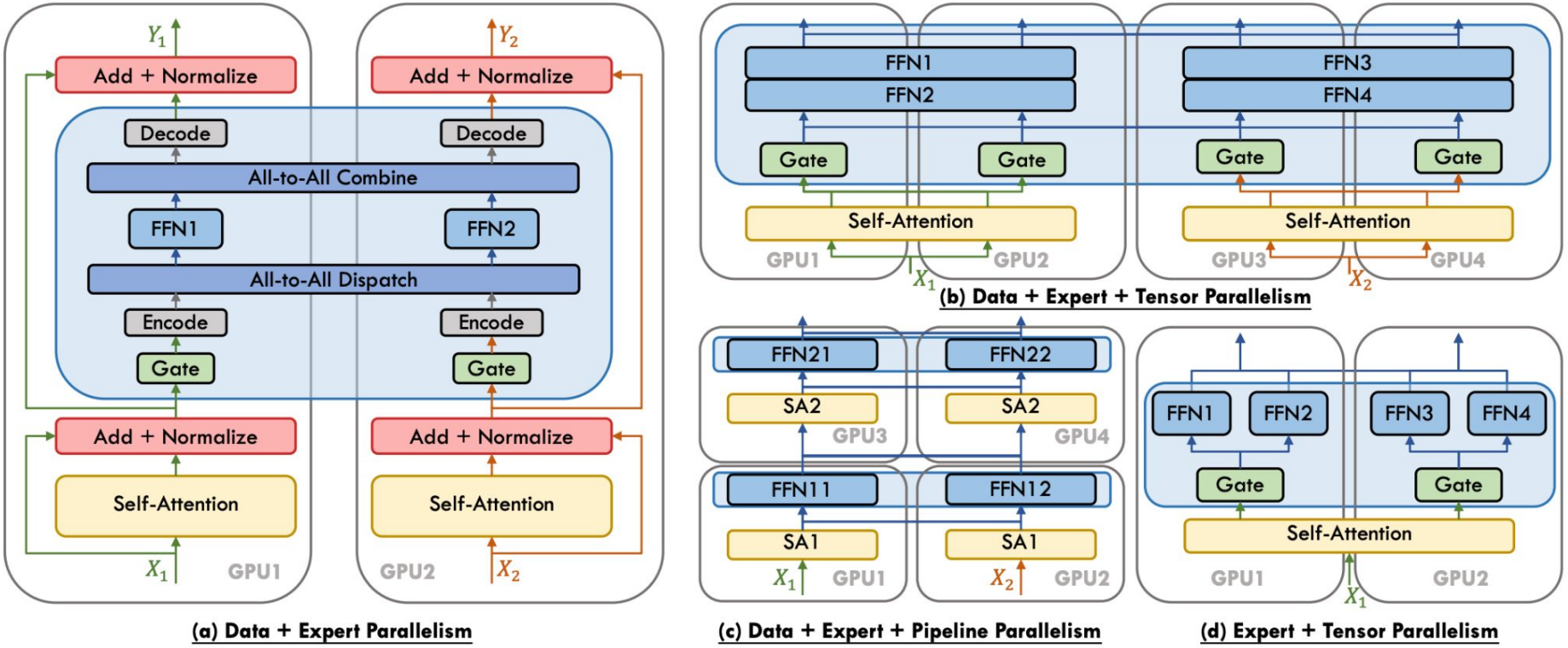
Some parallel strategies for MoE

Data Parallelism: Divide data into different GPUs

Expert Parallelism: Divide experts into different GPUs

Tensor Parallelism: Divide one layer (tensor) into different GPUs

Pipeline Parallelism: Divide layers into different GPUs



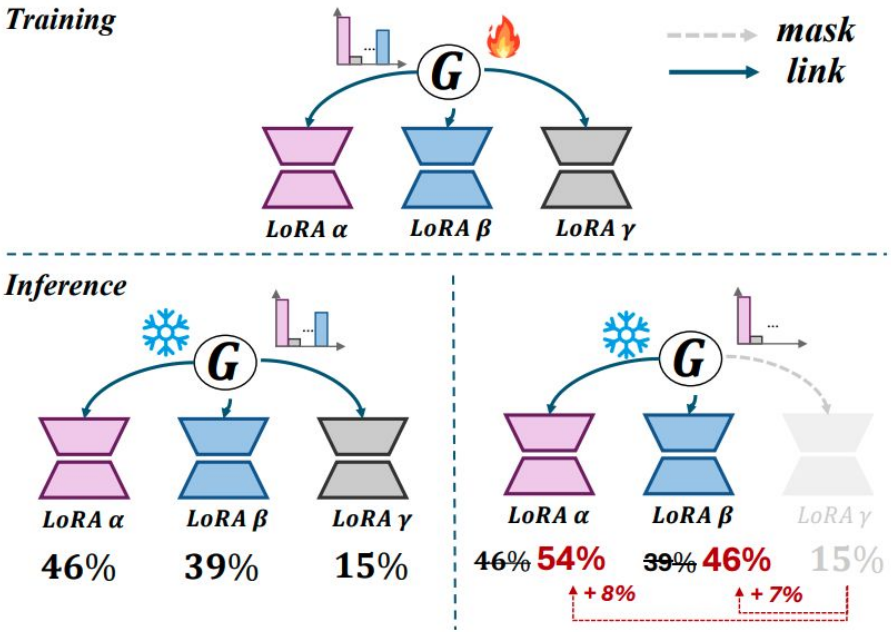
From: A Survey on Mixture of Experts (Cai, et al., 2024)

What about Dense MOE

MIXTURE OF LORA EXPERTS

Xun Wu[1,2]*, **Shaohan Huang**[1,✉], **Furu Wei**[1]
[1]Microsoft Research Asia [2]Tsinghua Univeristy
wuxun21@mails.tsinghua.edu.cn; {shaohanh, fuwei}@microsoft.com

Dense MOE is useful in LoRA scenarios, without sacrifice computational efficiency.



Balance the load on each expert

Recap: auxiliary loss

A good solution for load balancing is to add an auxiliary loss

- Alpha is a constant hyperparameter and N is the number of experts (Why multiply by N?)
- f_i is the fraction of tokens dispatched to expert i
- P_i is the fraction of the router probability allocated to expert i
- The loss is minimized when all f_i and P_i are equal to 1/N

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$

Expert Choice – Another way to balance the load on each expert

Mixture-of-Experts with Expert Choice Routing

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon

Google, Mountain View, CA, USA
{yanqiz, taole, hanxiaol, dunan, huangyp, vzhao, adai, zhifengc, qvl, jlaudon}@google.com

Expert Choice – Another way to balance the load on each expert

expert. Previous methods add an auxiliary loss on load balancing to mitigate the issue. However, this auxiliary loss does not guarantee a balanced load, especially during the important early stages of training. Indeed, **we empirically observe that the over-capacity ratio can reach 20%–40% for some experts in token choice routing**, indicating that a significant portion of the tokens routed to these experts will be dropped.

Expert Choice – Another way to balance the load on each expert

Instead of token choose which expert, each expert choose the top-k tokens, k is the capacity of the expert.

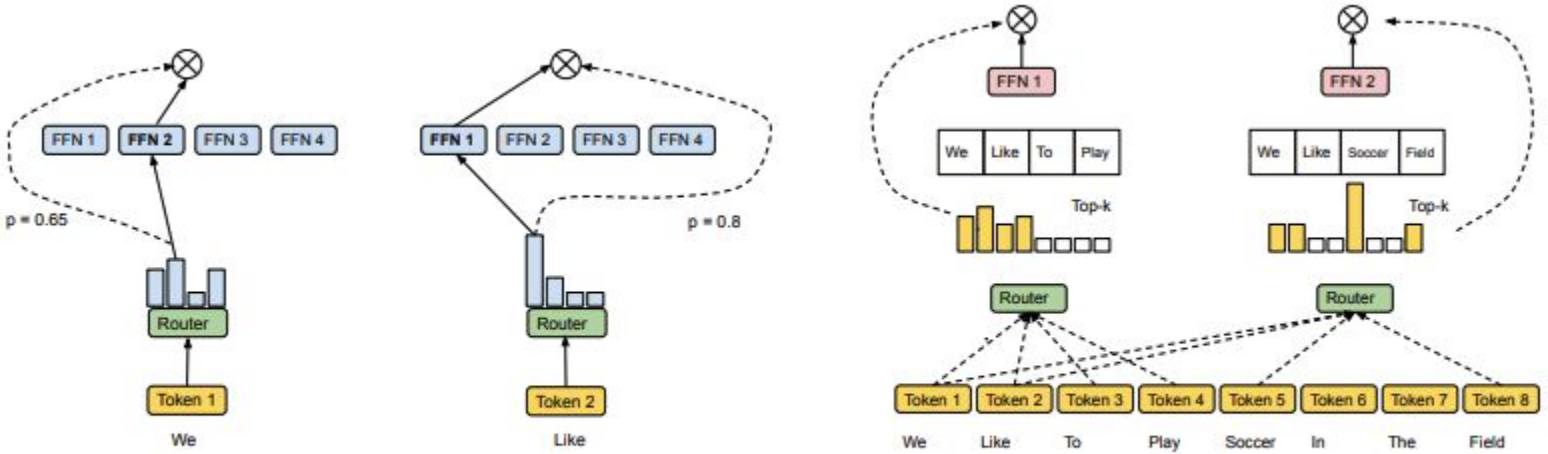
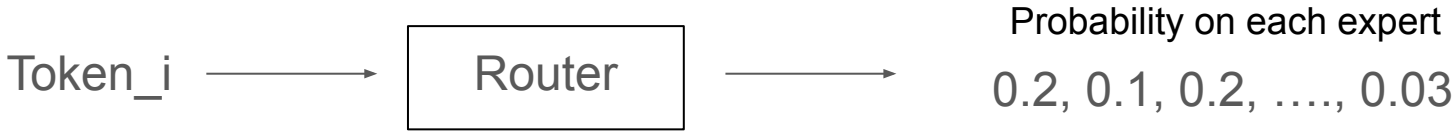


Figure 1: High-level Comparison Between Conventional MoE and expert choice MoE.

Expert Choice – Another way to balance the load on each expert

Instead of token choose which expert, each expert choose the top-k tokens, k is the capacity of the expert.

$$S = \text{Softmax}(X \cdot W_g), \quad S \in \mathbb{R}^{n \times e}$$



Expert Choice – Another way to balance the load on each expert

Instead of token choose which expert, each expert choose the top-k tokens, k is the capacity of the expert.

$$S = \text{Softmax}(X \cdot W_g), \quad S \in \mathbb{R}^{n \times e}$$

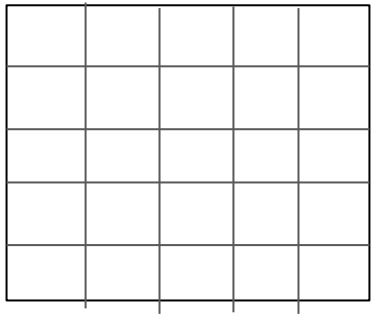
Token_i → Router → Probability on each expert

0.2, 0.1, 0.2, …., 0.03

$$G, I = \text{TopK}(S^\top, k), P = \text{Onehot}(I)$$

e experts

n tokens



So all tokens form a token-to-expert matrix

Expert Choice – Another way to balance the load on each expert

Instead of token choose which expert, each expert choose the top-k tokens, k is the capacity of the expert.

$$S = \text{Softmax}(X \cdot W_g), \quad S \in \mathbb{R}^{n \times e}$$

Token_i → Router → Probability on each expert

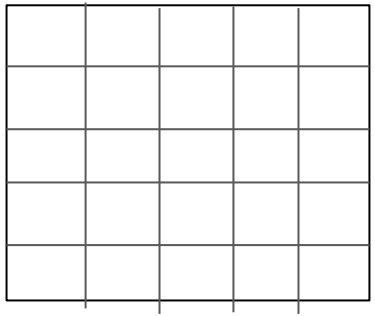
0.2, 0.1, 0.2, …., 0.03

$$G, I = \text{TopK}(S^\top, k), P = \text{Onehot}(I)$$

e experts

So all tokens form a token-to-expert matrix

n tokens



Take top-k on each column.

Expert Choice – Another way to balance the load on each expert

Instead of token choose which expert, each expert choose the top-k tokens, k is the capacity of the expert.

$$S = \text{Softmax}(X \cdot W_g), \quad S \in \mathbb{R}^{n \times e}$$

Token_i → Router → Probability on each expert

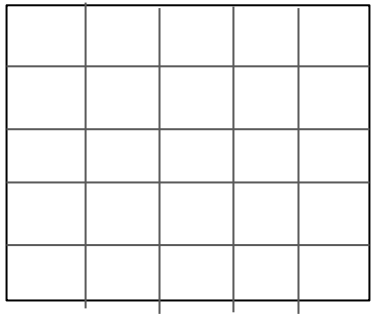
0.2, 0.1, 0.2, …., 0.03

$$G, I = \text{TopK}(S^\top, k), P = \text{Onehot}(I)$$

e experts

So all tokens form a token-to-expert matrix

n tokens



Take top-k on each column.

Guarantee balance loading

Expert Choice – Another way to balance the load on each expert

Instead of token choose which expert, each expert choose the top-k tokens, k is the capacity of the expert.

$$S = \text{Softmax}(X \cdot W_g), \quad S \in \mathbb{R}^{n \times e}$$

Token_i → Router → Probability on each expert

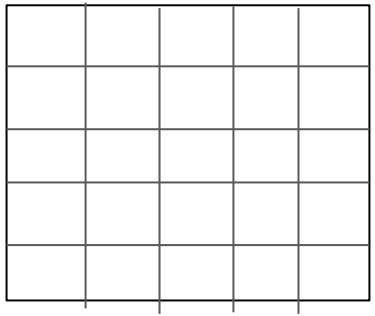
0.2, 0.1, 0.2, …., 0.03

$$G, I = \text{TopK}(S^\top, k), P = \text{Onehot}(I)$$

e experts

So all tokens form a token-to-expert matrix

n tokens



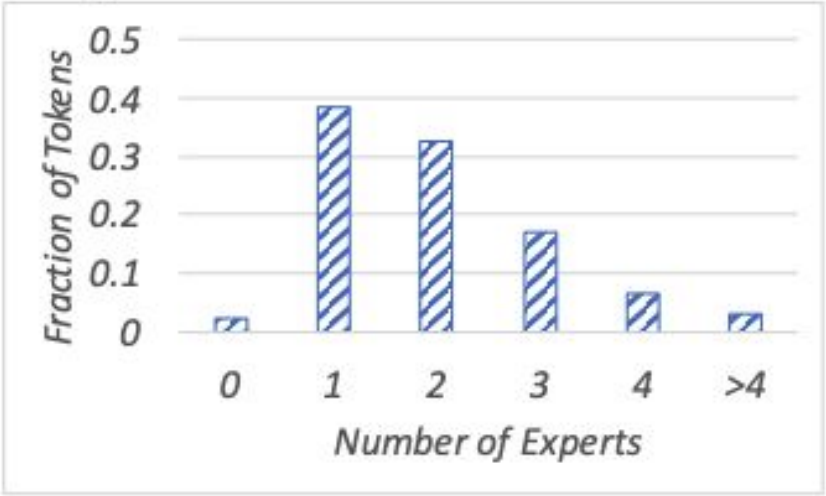
Take top-k on each column.

Guarantee balance loading

Expert Choice – Another way to balance the load on each expert

So each token can go to various number of experts

Figure 3: Distribution of the number of experts routed to per token in a 100M/64E model.



Expert Choice – Another way to balance the load on each expert

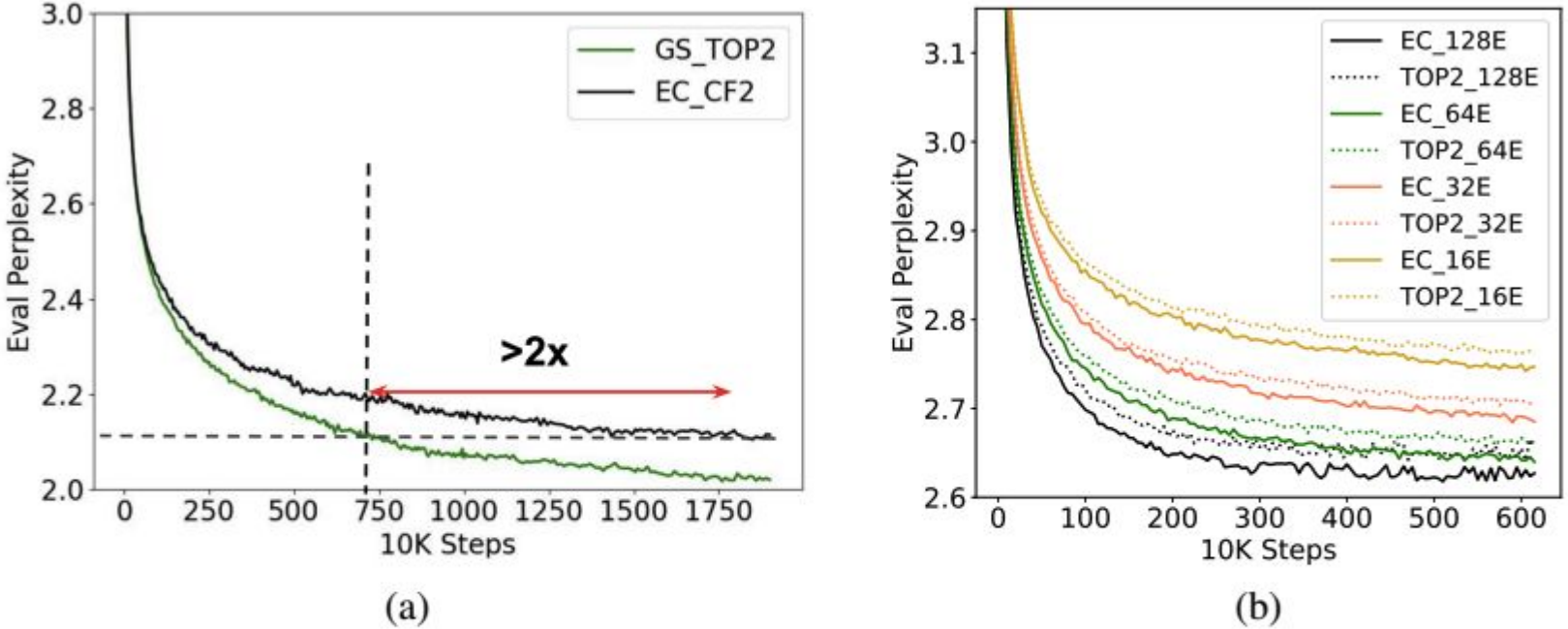


Figure 2: (a) Training convergence is more than 2x faster using our method compared to GShard top-2 gating. (b) Training perplexity scales strongly with the number of experts while keeping the expert size fixed. EC consistently outperforms GShard top-2 gating.

The new technique behind the recent Phi 3.5-MOE

GRIN: *GR*adient-*IN*formed MoE

Liyuan Liu* Young Jin Kim Shuohang Wang Chen Liang
Yelong Shen Hao Cheng Xiaodong Liu Masahiro Tanaka
Xiaoxia Wu Wenxiang Hu Vishrav Chaudhary
Zeqi Lin Chenruidong Zhang Jilong Xue
Hany Awadalla Jianfeng Gao* Weizhu Chen*

Microsoft

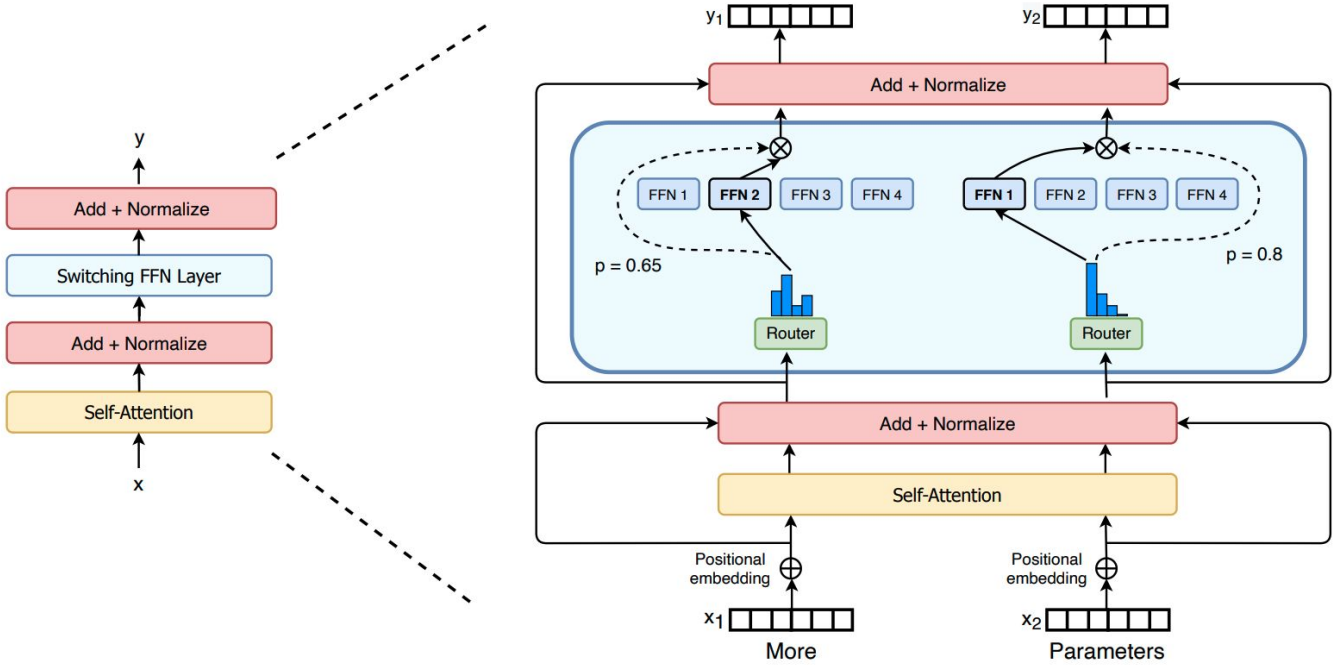
Why MOE?

2. Specialization

The brain is a mixture of experts too! (Why and how the brain weights contributions from a mixture of experts)

Different input token can go to different experts

We may have experts for number (1, 2, 3), experts for code, experts for instruction tokens



Do different tokens go to different experts?

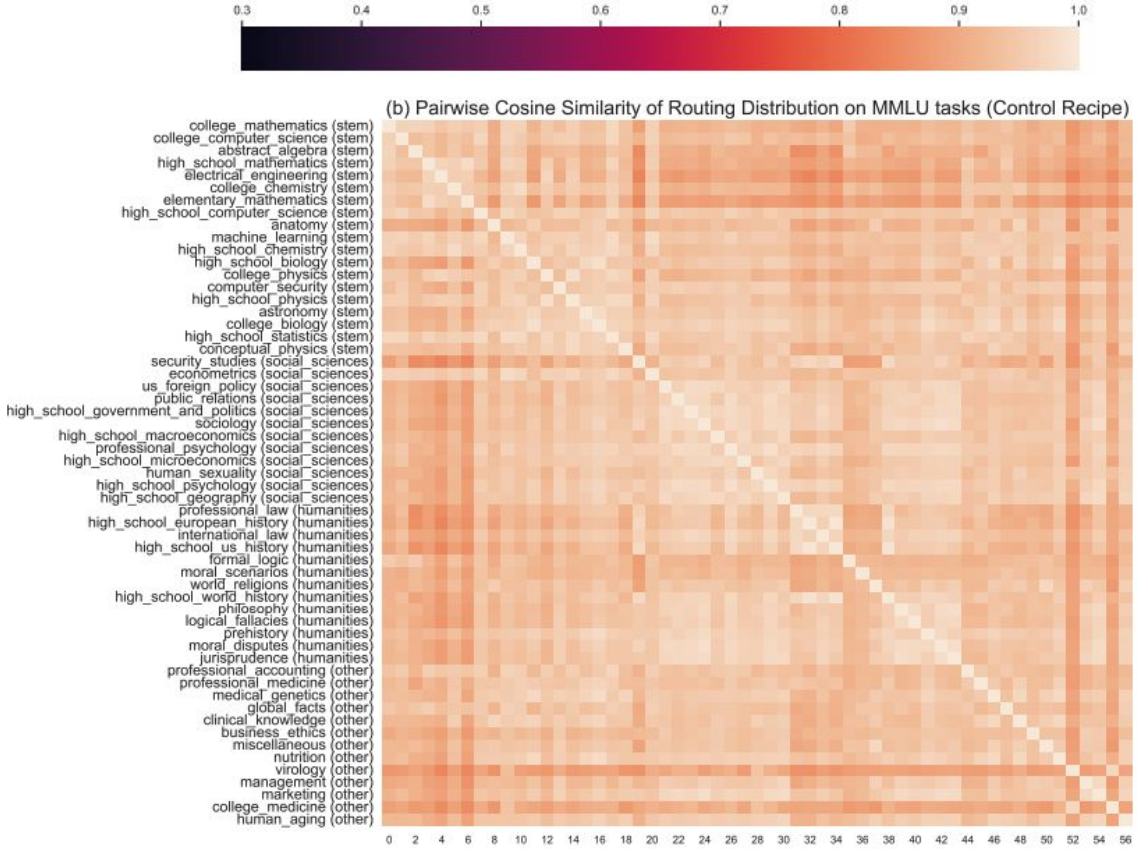


Figure 6 (b): MoE Routing distribution similarity across MMLU 57 tasks for the control recipe. This figure shares the same setting and the same color bar with Figure 6 (a).

No, the routing distribution across 57 tasks are very similar.

The reason is because the routing (Top-k) is not differentiable, gradient only backpropagate to selected experts not the non-selected experts.

$$\sum_{i=0}^{n-1} Gating(\boldsymbol{z})_i \cdot TopK(\boldsymbol{z})_i \cdot Expert(\boldsymbol{x}, \boldsymbol{w}_i), \tag{1}$$

where $\boldsymbol{z} = Router(\boldsymbol{x}, \boldsymbol{r})$, $\boldsymbol{r}$ is the router parameters, $Gating(\cdot)$ is a gating function (usually softmax), and $Expert(\cdot)$ is a FNN. In our study, we define use a linear network as the router, i.e., $Router(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{x} \cdot \boldsymbol{r}^T$ As to $TopK(\boldsymbol{z})$, it is the TopK function, i.e., $TopK(\boldsymbol{z})_i := 1$ if $\boldsymbol{z}_i$ is among the TopK coordinates of $\boldsymbol{z}$ and $TopK(\boldsymbol{z})_i := 0$ otherwise.

The reason is because the routing (Top-k) is not differentiable, gradient only backpropagate to selected experts not the non-selected experts.

GRIN replaces this top-k with a gradient estimator

$$\sum_{i=0}^{n-1} Gating(\boldsymbol{z})_i \cdot TopK(\boldsymbol{z})_i \cdot Expert(\boldsymbol{x}, \boldsymbol{w}_i), \tag{1}$$

where $\boldsymbol{z} = Router(\boldsymbol{x}, \boldsymbol{r})$, $\boldsymbol{r}$ is the router parameters, $Gating(\cdot)$ is a gating function (usually softmax), and $Expert(\cdot)$ is a FNN. In our study, we define use a linear network as the router, i.e., $Router(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{x} \cdot \boldsymbol{r}^T$ As to $TopK(\boldsymbol{z})$, it is the TopK function, i.e., $TopK(\boldsymbol{z})_i := 1$ if $\boldsymbol{z}_i$ is among the TopK coordinates of $\boldsymbol{z}$ and $TopK(\boldsymbol{z})_i := 0$ otherwise.

The reason is because the routing (Top-k) is not differentiable, gradient only backpropagate to selected experts not the non-selected experts.

GRIN replaces this top-k with a gradient estimator

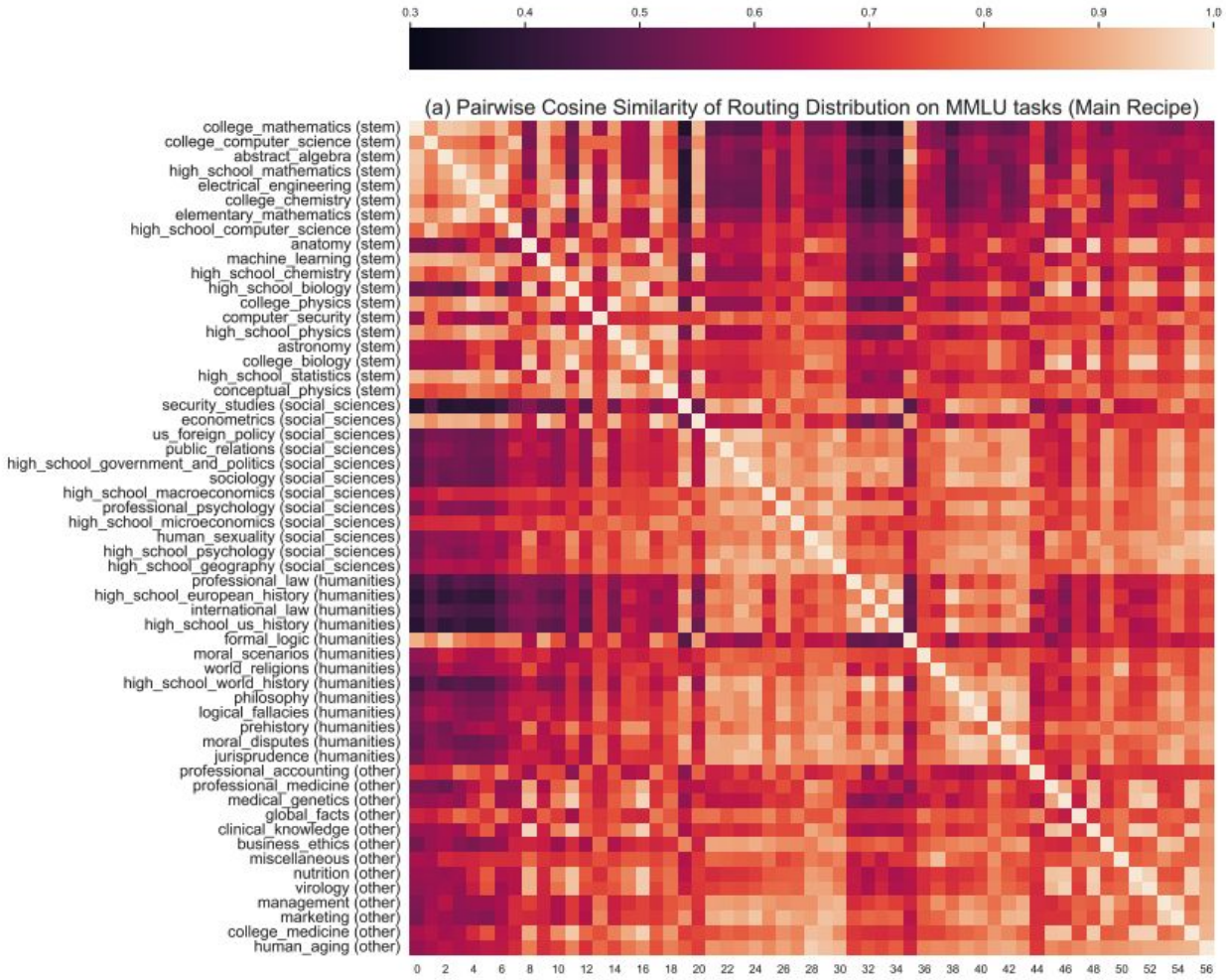
$$\sum_{i=0}^{n-1} Gating(\boldsymbol{z})_i \cdot TopK(\boldsymbol{z})_i \cdot Expert(\boldsymbol{x}, \boldsymbol{w}_i), \tag{1}$$

where $\boldsymbol{z} = Router(\boldsymbol{x}, \boldsymbol{r})$, $\boldsymbol{r}$ is the router parameters, $Gating(\cdot)$ is a gating function (usually softmax), and $Expert(\cdot)$ is a FNN. In our study, we define use a linear network as the router, i.e., $Router(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{x} \cdot \boldsymbol{r}^T$ As to $TopK(\boldsymbol{z})$, it is the TopK function, i.e., $TopK(\boldsymbol{z})_i := 1$ if $\boldsymbol{z}_i$ is among the TopK coordinates of $\boldsymbol{z}$ and $TopK(\boldsymbol{z})_i := 0$ otherwise.

See this paper for details on the gradient estimator

Bridging Discrete and Backpropagation: Straight-Through and Beyond

GRIN MOE has a much more different routing distributions across tasks.

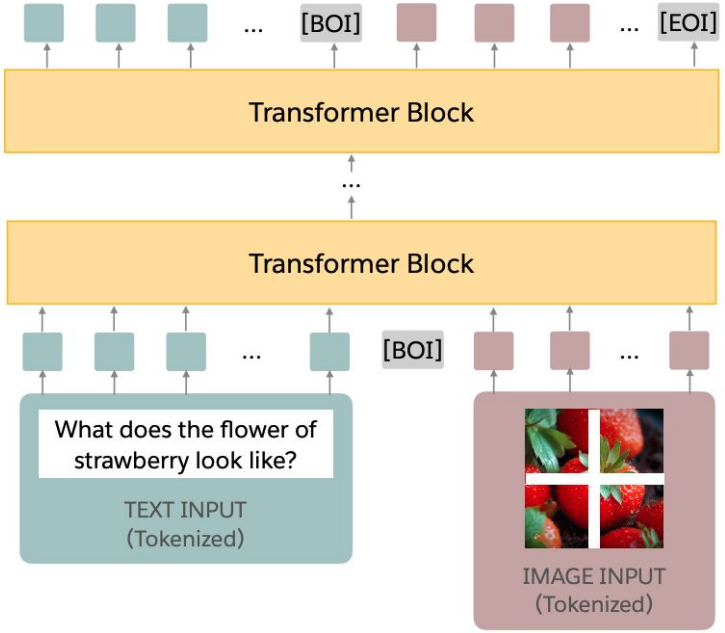


MoMa: Efficient Early-Fusion Pre-training with Mixture of Modality-Aware Experts

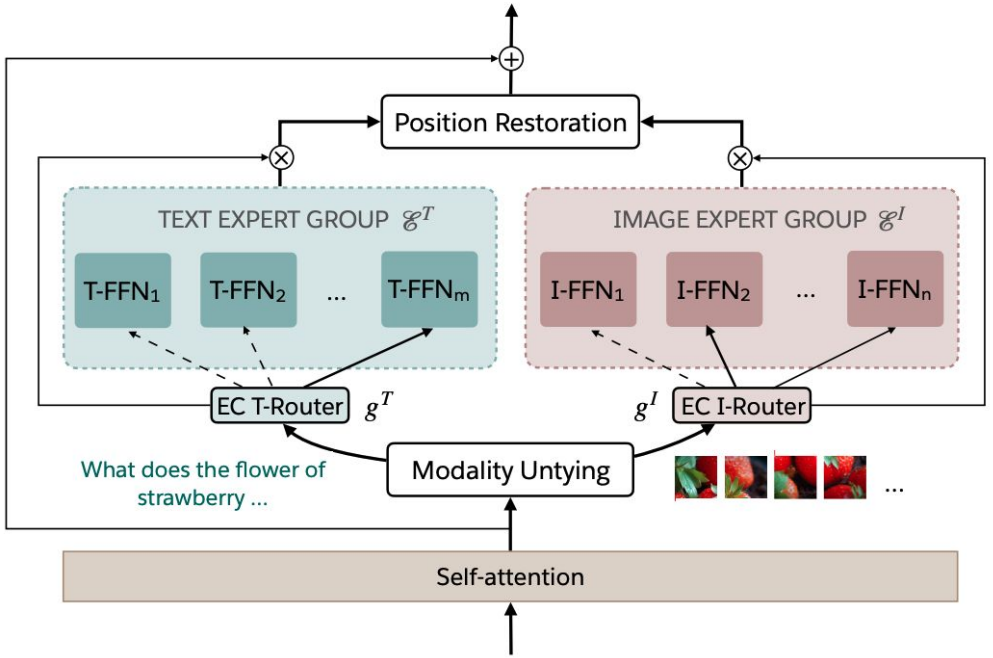
2024 July

MoMa - Mixture of Modality-aware Experts

In **Chameleon**, images are tokenized using a learned image tokenizer that encodes a 512 × 512 image into 1024 discrete tokens from a codebook of size 8192. Text is tokenized using a BPE tokenizer with a vocabulary size of 65,536, which includes the 8192 image codebook tokens. This unified tokenization scheme enables the

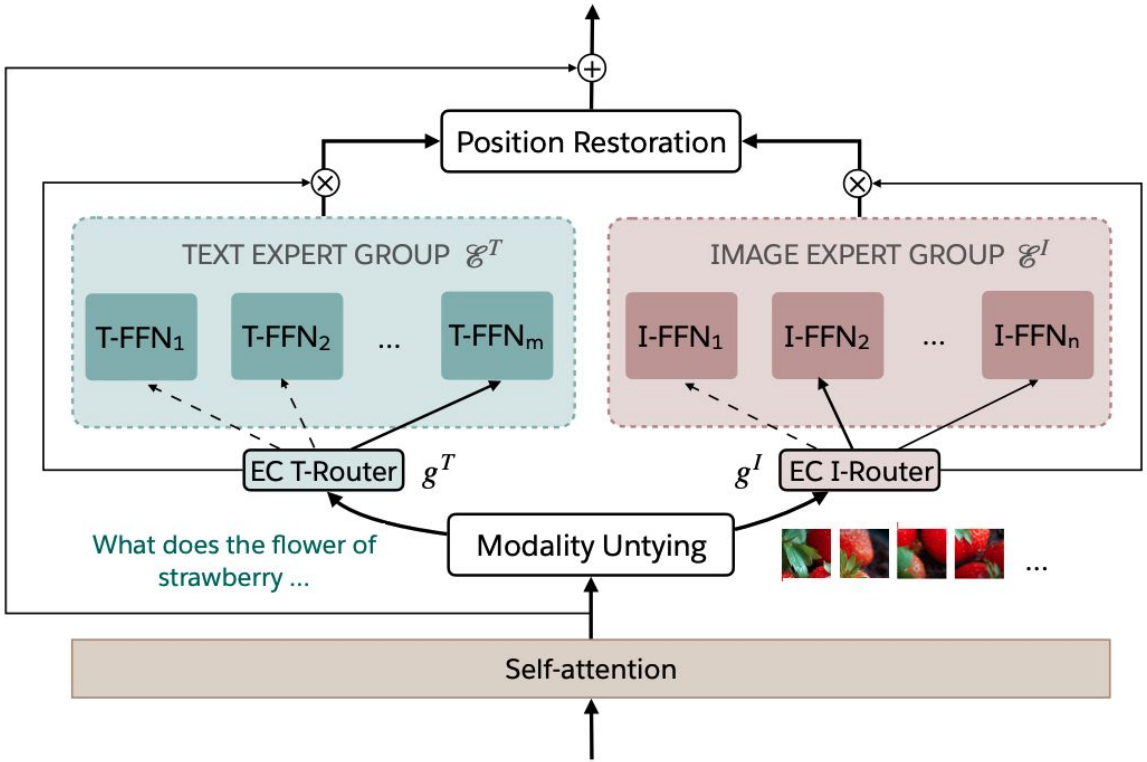


(a) Early-fusion mixed-modal LLM architecture.



(b) Mixture of modality-aware experts (MoMa) transformer block.

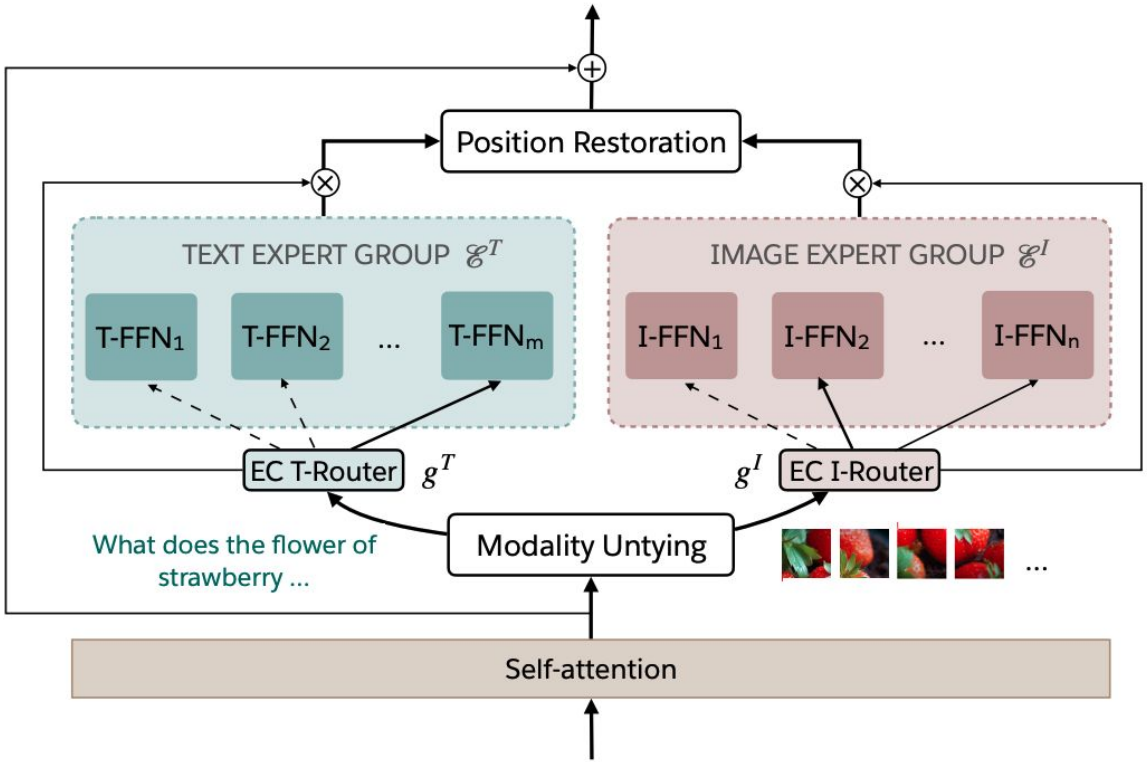
MoMa - Mixture of Modality-aware Experts



(b) Mixture of modality-aware experts (MoMa) transformer block.

They use expert choice (EC)

MoMa - Mixture of Modality-aware Experts



(b) Mixture of modality-aware experts (MoMa) transformer block.

Expert Choice Router in MoMa

1. We use Sigmoid as the non-linearity in the router scoring function, enabling independent calculation of token-to-expert affinity scores for each token.

Instead of softmax across all experts.

$$y = \begin{cases} \sum_{j=1}^{m} g^{\mathcal{T}}(x)_j \cdot \text{FFN}^{\mathcal{T}}_{\text{SwiGLU}_j}(x) & \text{if } x \in \mathcal{T} \\ \sum_{j=1}^{n} g^{\mathcal{I}}(x)_j \cdot \text{FFN}^{\mathcal{I}}_{\text{SwiGLU}_j}(x) & \text{if } x \in \mathcal{I} \end{cases}$$

$$g^{\mathcal{M}}(x)_j = \begin{cases} \text{Sigmoid}(x \cdot W_g^{\mathcal{M}})_j & \text{if } x \text{ is selected by } E_j \\ 0 & \text{otherwise} \end{cases}$$

Expert Choice Router in MoMa

1. We use Sigmoid as the non-linearity in the router scoring function, enabling independent calculation of token-to-expert affinity scores for each token.

Instead of softmax across all experts.

2. We introduce auxiliary routers, inspired by Raposo et al. (2024), which predict the likelihood of an expert selecting a token solely based on its hidden state representation. These routers are trained after the main model training is completed and employed during inference to ensure causality. We discuss the

cannot directly apply the expert-choice routing for MoE and layer-choice routing for MoD during inference time, as the top-k token selection within a batch breaks causality.

Expert Choice Router in MoMa

1. We use Sigmoid as the non-linearity in the router scoring function, enabling independent calculation of token-to-expert affinity scores for each token.

Instead of softmax across all experts.

2. We introduce auxiliary routers, inspired by Raposo et al. (2024), which predict the likelihood of an expert selecting a token solely based on its hidden state representation. These routers are trained after the main model training is completed and employed during inference to ensure causality. We discuss the

cannot directly apply the expert-choice routing for MoE and layer-choice routing for MoD during inference time, as the top-k token selection within a batch breaks causality.

We employ a two-stage training approach, where the main model and auxiliary routers are trained separately. **First, we train the main model to convergence. Then, we train the auxiliary routers using binary cross-entropy loss,** supervised by the ground-truth top-k routing assignments computed over an entire batch.

Upcycling Komatsuzaki et al. (2023) in MoMa

The insight we identified is that MoE routers are responsible for partitioning the representation space for each expert. However, **this representation space is sub-optimal in the early stages of model training,** leading to a sub-optimally trained routing function.

Specifically, we **begin by training an architecture consisting of 1 FFN expert per modality**. After a predetermined number of steps, we upcycle this model by **converting each modality-specific FFN into an expert-choice MoE module,** initializing each expert with the expert trained from the first stage. We reset the learning rate scheduler while preserving the data loader state from the previous stage, thereby ensuring the second stage training is exposed to refreshed data.

Upcycling Komatsuzaki et al. (2023) in MoMa

The insight we identified is that MoE routers are responsible for partitioning the representation space for each expert. However, **this representation space is sub-optimal in the early stages of model training,** leading to a sub-optimally trained routing function.

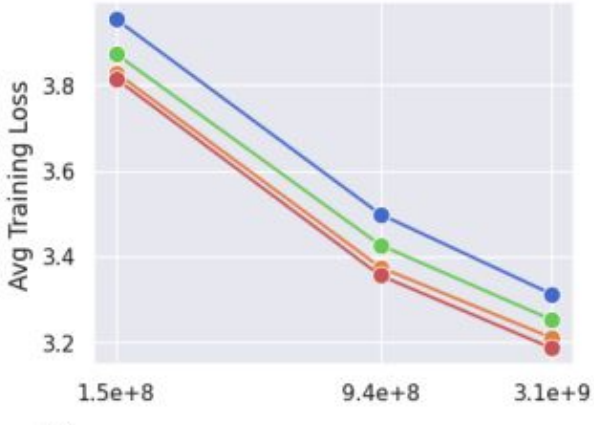
Specifically, we **begin by training an architecture consisting of 1 FFN expert per modality**. After a predetermined number of steps, we upcycle this model by **converting each modality-specific FFN into an expert-choice MoE module,** initializing each expert with the expert trained from the first stage. We reset the learning rate scheduler while preserving the data loader state from the previous stage, thereby ensuring the second stage training is exposed to refreshed data.

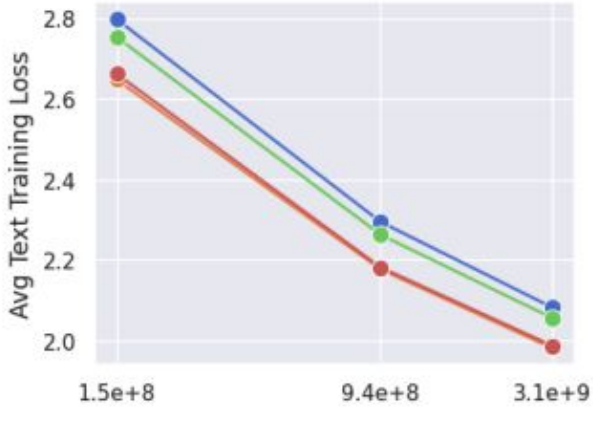
To promote expert specialization, we augment the MoE routing function with Gumbel noise (Liu et al., 2022b; Geng et al., 2020), allowing our router to differentiably sample experts. This is expressed in Equation 5:

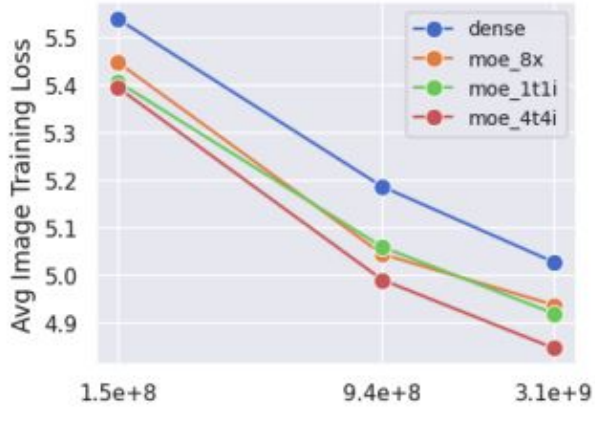
$$\text{Gumbel-Sigmoid}(x) = \text{Sigmoid}(x + G^{'} - G^{''}) \tag{5}$$

where $G^{'}$ and $G^{''}$ are independent Gumbel noise samples.

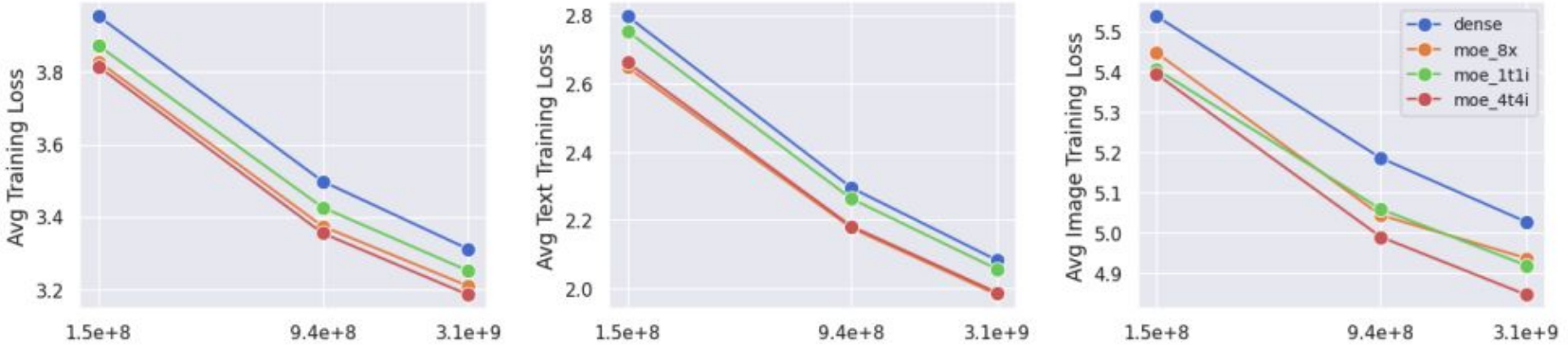
Modality group improves over normal MOE.







But Modality group MOE has the same text loss only improves on image.

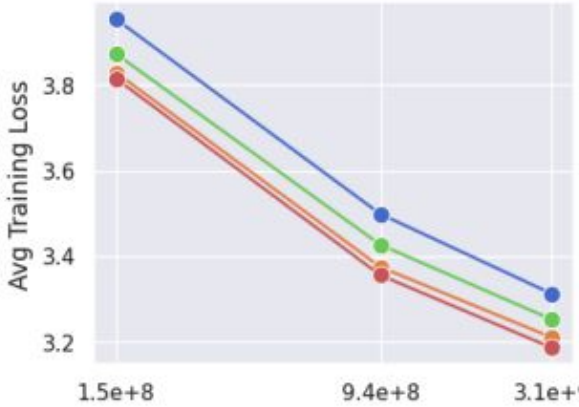


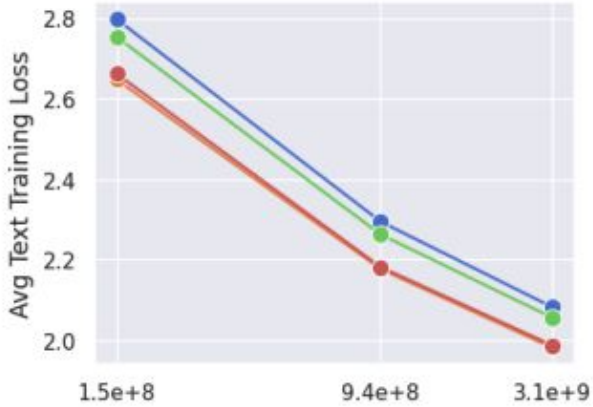
Text-Only: We use a variety of textual datasets, including a combination of the pre-training data used to train LLaMa-2 (Touvron et al., 2023) and CodeLLaMa (Roziere et al., 2023) for a total of **2.9 trillion** text-only tokens.

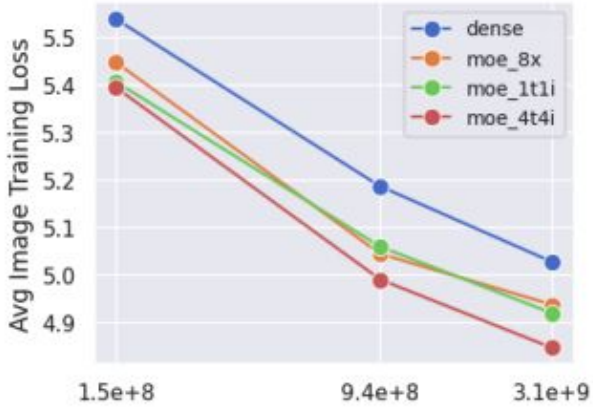
Text-Image: The text-image data for pre-training is a combination of publicly available data sources and licensed data. The images are then resized and center cropped into 512×512 images for tokenization. In total, we include 1.4 billion text-image pairs, which produces **1.5 trillion** text-image tokens.

Text/Image Interleaved: We procure data from publicly available web sources, not including data from Meta's products or services, for a total of **400 billion** tokens of interleaved text and image data similar to Laurençon et al. (2023). We apply the same filtering for images, as was applied in **Text-To-Image**.

But Modality group MOE has the same text loss only improves on image.







Text-Only: We use a variety of textual datasets, including a combination of the pre-training data used to train LLaMa-2 (T 3) for a total of **2.9 trillion** text-only tokens.

Maybe it's because the text data is much more than image?

Text-Image: The ublicly available data sources and licensed data. The images are then resized and center cropped into 512 × 512 images for tokenization. In total, we include 1.4 billion text-image pairs, which produces **1.5 trillion** text-image tokens.

Text/Image Interleaved: We procure data from publicly available web sources, not including data from Meta's products or services, for a total of **400 billion** tokens of interleaved text and image data similar to Laurençon et al. (2023). We apply the same filtering for images, as was applied in **Text-To-Image**.