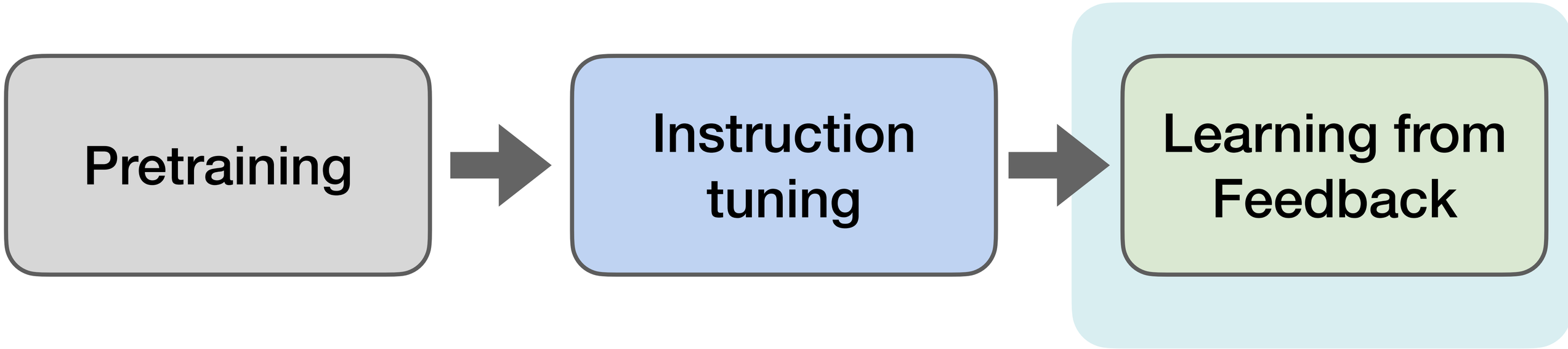
Large Language Model Basics and MoE Architecture

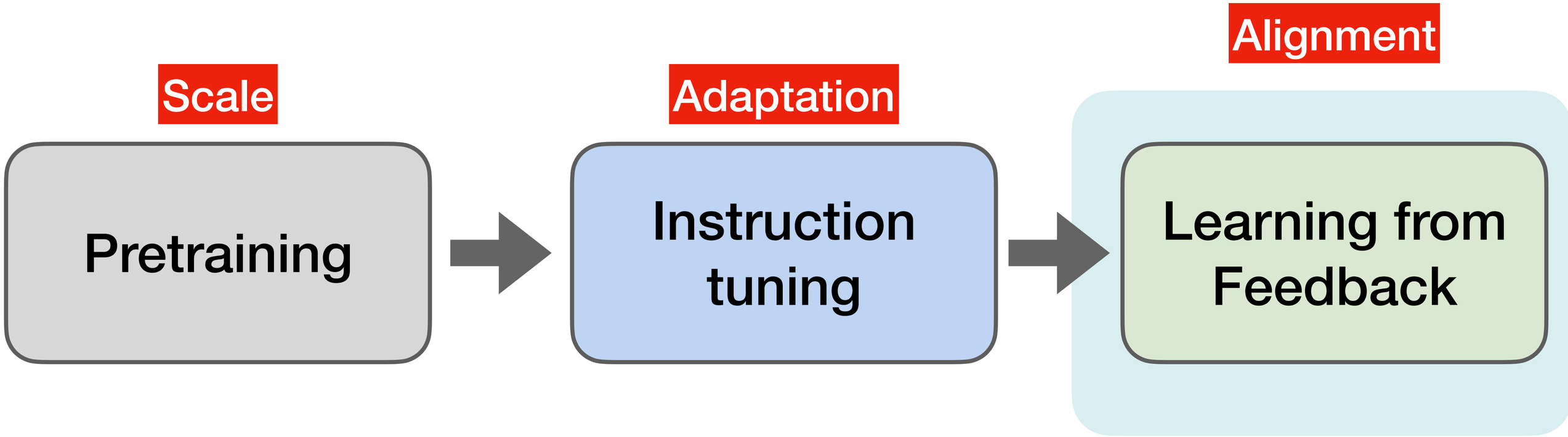
Yen-Ting Lin 林彥廷

Sep 30, 2024 @ ADL Recitation, NTU

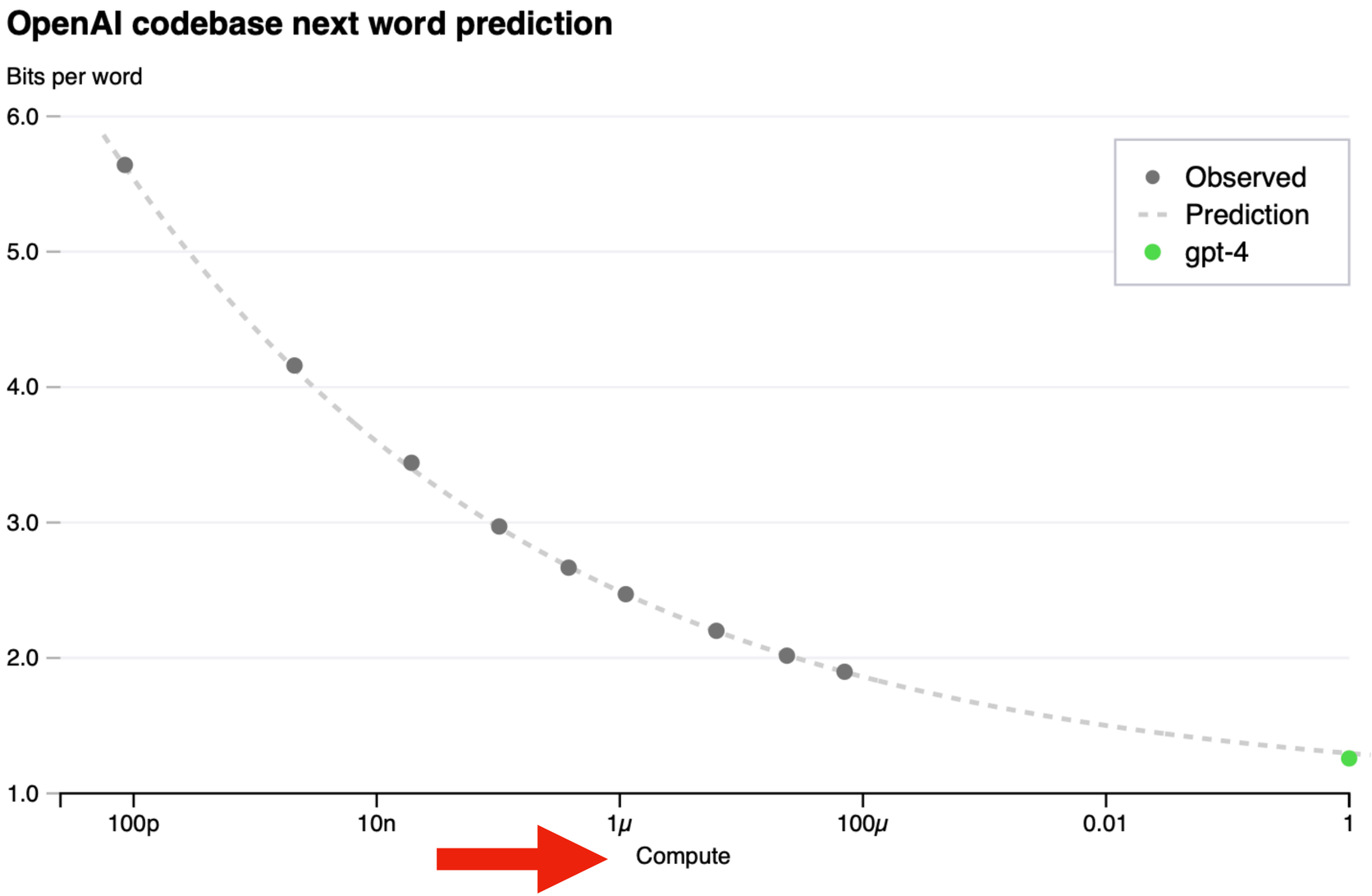
LLM Development



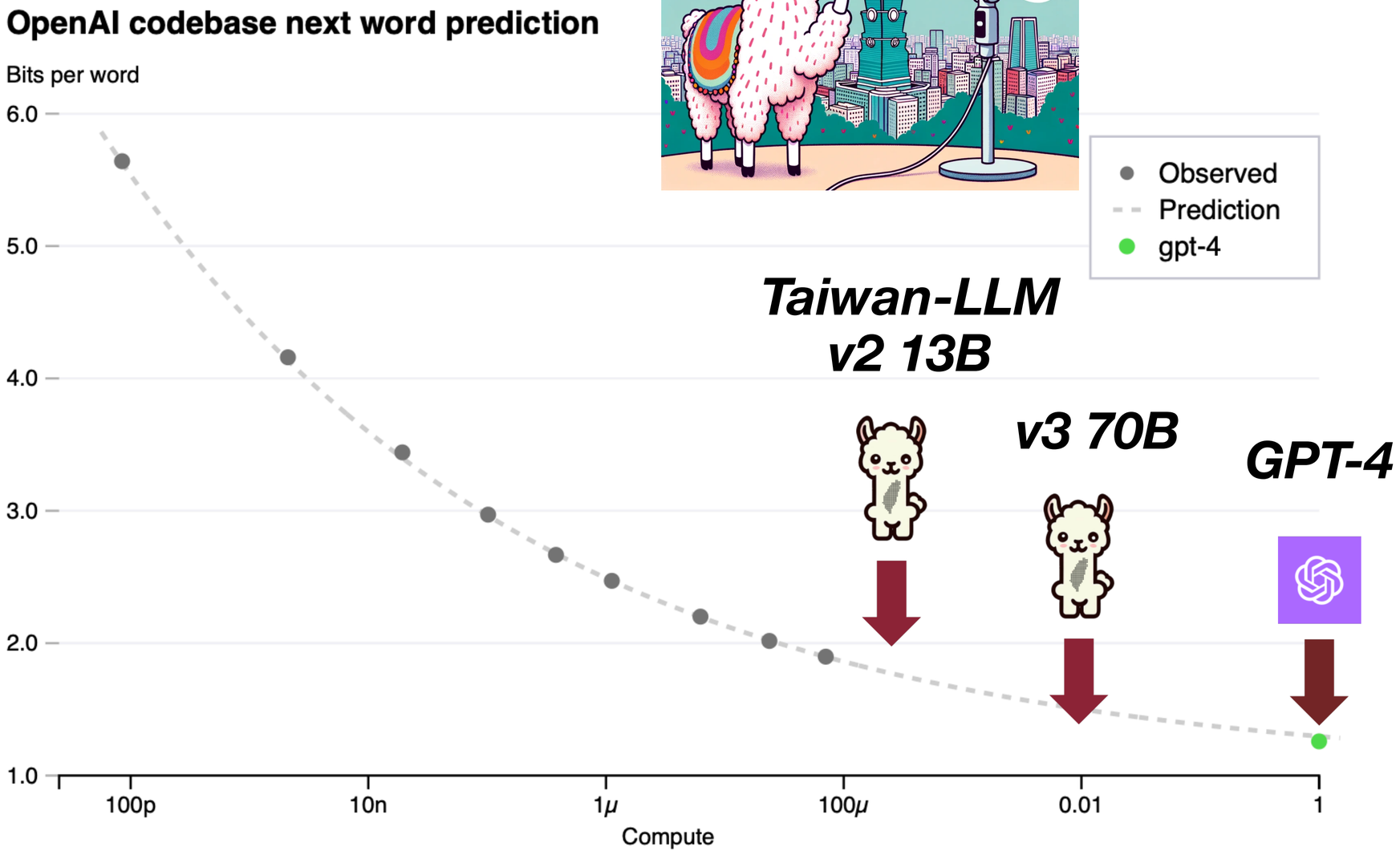
LLM Development



Scaling Law


OpenAI codebase next word prediction
Bits per word
6.0
5.0
4.0
3.0
2.0
1.0
Observed
Prediction
gpt-4
100p
10n
1μ
100μ
0.01
1
Compute

Scaling Law


OpenAI codebase next word prediction
Bits per word
6.0
5.0
4.0
3.0
2.0
1.0
100p
10n
1μ
100μ
0.01
1
Compute
Observed
Prediction
gpt-4
Taiwan-LLM
v2 13B
v3 70B
GPT-4

Emergent Abilities

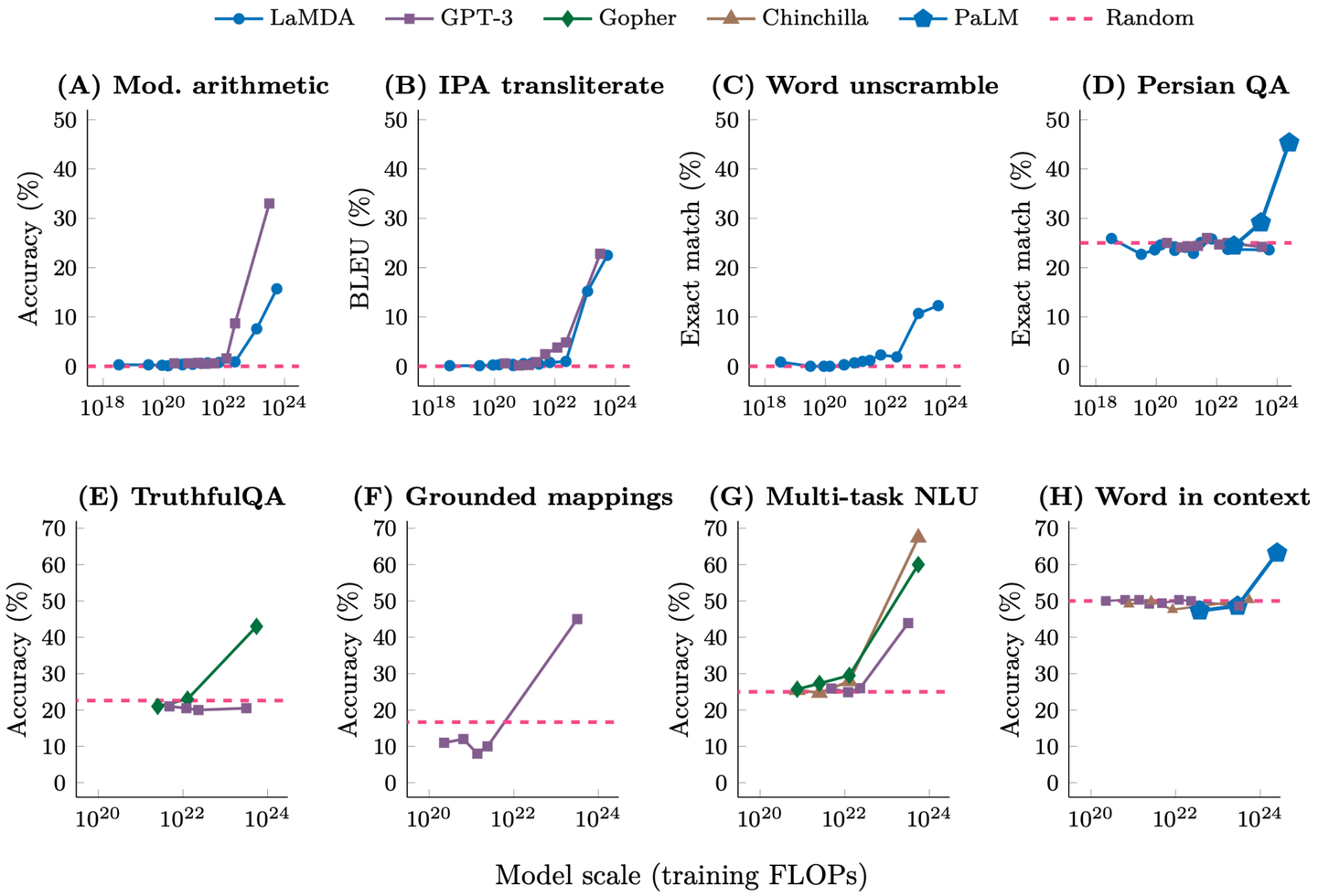
An ability is emergent if it is not present in smaller models

but is present in larger models.

Emergent Abilities of Large Language Models

Jason Wei [1] *jasonwei@google.com*

Emergent Abilities



Emergent Abilities of Large Language Models

Jason Wei [1]

jasonwei@google.com

How to Scale?

Model scale (training FLOPs)

Training FLOPs ~= 6 x Model Size x Number of Tokens

How to Scale?

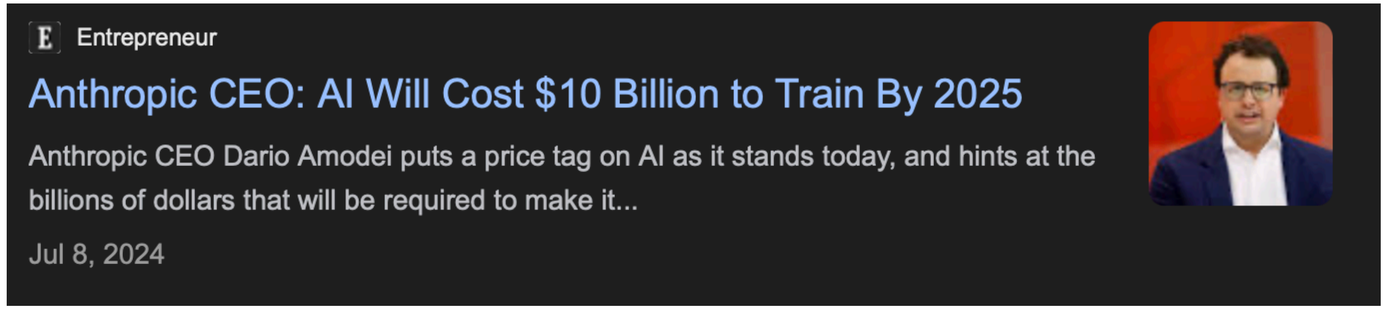
Model scale (training FLOPs)

Training FLOPs ~= 6 x Model Size x Number of Tokens

Taiwan-LLM Training FLOPs = 6 x 70e9 x 500e9 = 2e23

GPT-4 FLOPs = 2e25

Cost of Frontier Models



GPT-4 FLOPs = 2e25	~100M USD	2023 Q1
GPT-5 FLOPs = 2e26?	~1B USD	2024 Q4?
GPT-6 FLOPs = 2e27?	~10B USD	2025?

Chinchilla's Law

DeepMind

Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

Research Question

- Given a fixed FLOPs budget, C, how should one trade-off model size and number of training tokens
- Final pre-training loss as a function of L(N, D)
  - N: # parameters
  - D: # training tokens

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \mathrm{FLOPs}(N,D)=C}{\operatorname{argmin}} L(N, D).$$

Fix FLOPs

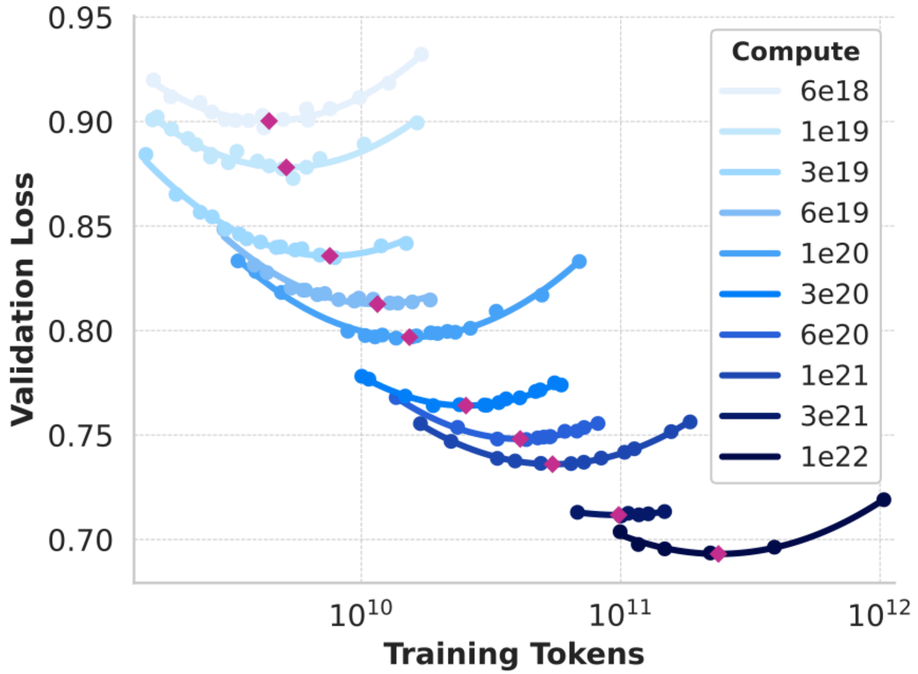


Figure 2 Scaling law IsoFLOPs curves between 6×10^{18} and 10^{22} FLOPs. The loss is the negative log-likelihood on a held-out validation set. We approximate measurements at each compute scale using a second degree polynomial.

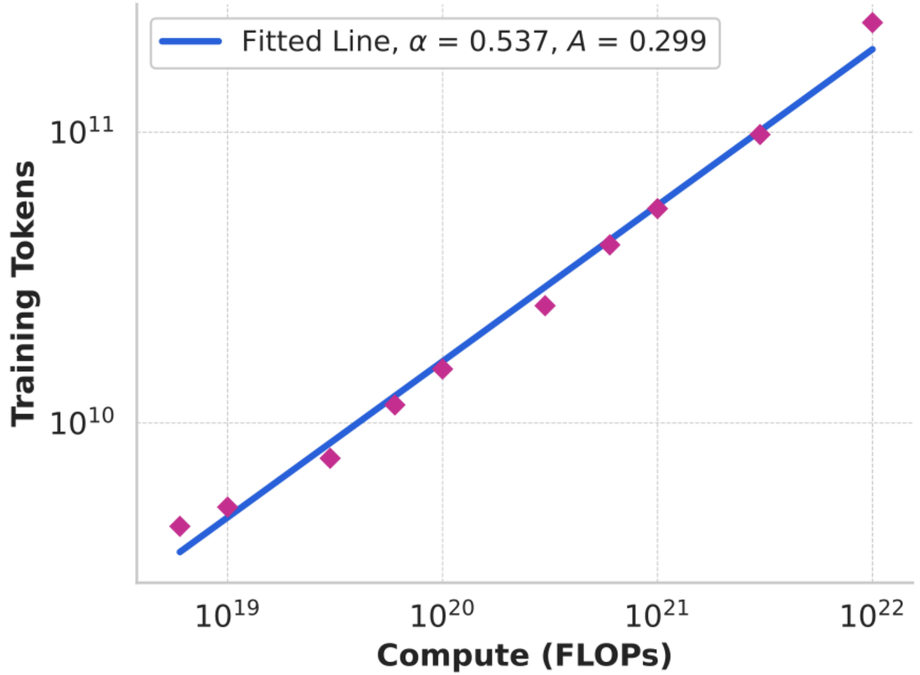


Figure 3 Number of training tokens in identified compute-optimal models as a function of pre-training compute budget. We include the fitted scaling-law prediction as well. The compute-optimal models correspond to the parabola minimums in Figure 2.

Figure source: The Llama 3 Herd of Models

Pertaining loss as downstream performance indicator?

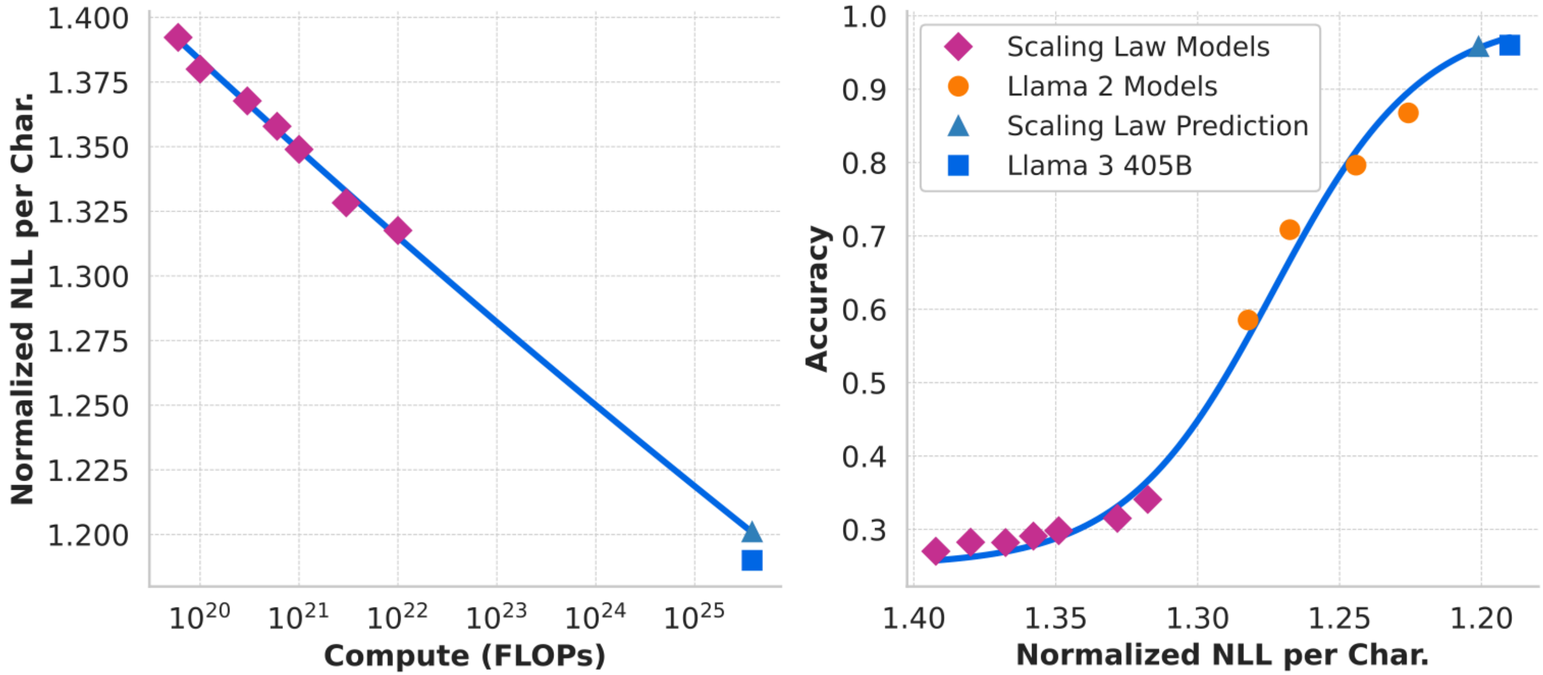


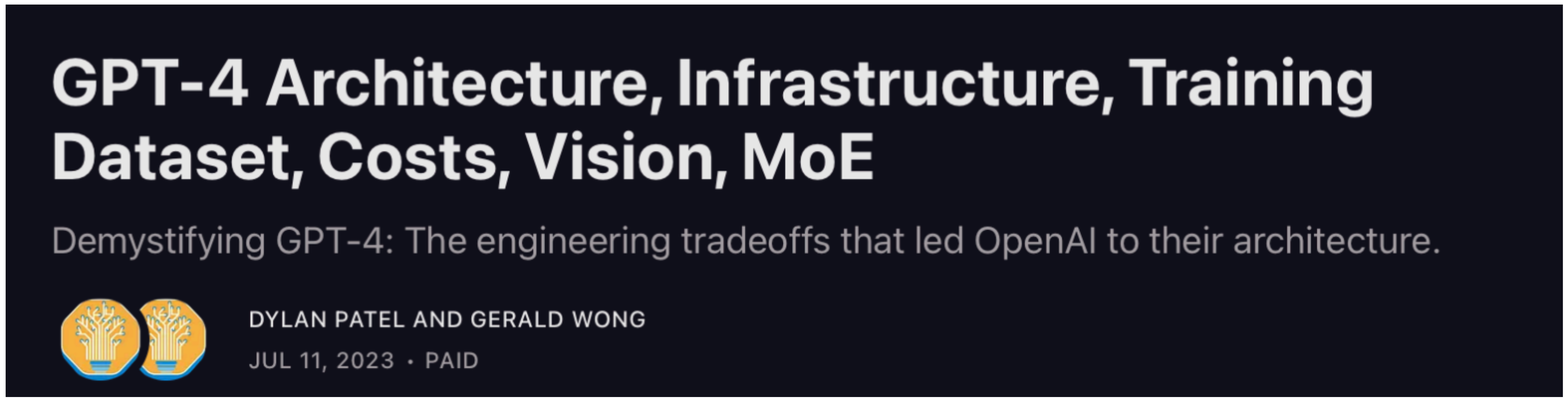
Figure 4 Scaling law forecast for ARC Challenge. *Left:* Normalized negative log-likelihood of the correct answer on the ARC Challenge benchmark as a function of pre-training FLOPs. *Right:* ARC Challenge benchmark accuracy as a function of the normalized negative log-likelihood of the correct answer. This analysis enables us to predict model performance on the ARC Challenge benchmark before pre-training commences. See text for details.

Figure source: The Llama 3 Herd of Models

Transformer Architecture

- Vanilla Transformers vs LLaMa (請見去年實習課影片)
  - RMS Normalization
  - Rotary Positional Encoding
  - KV-Cache
  - Group Query Attention
  - SwiGLU
- **Mixture of Expert**

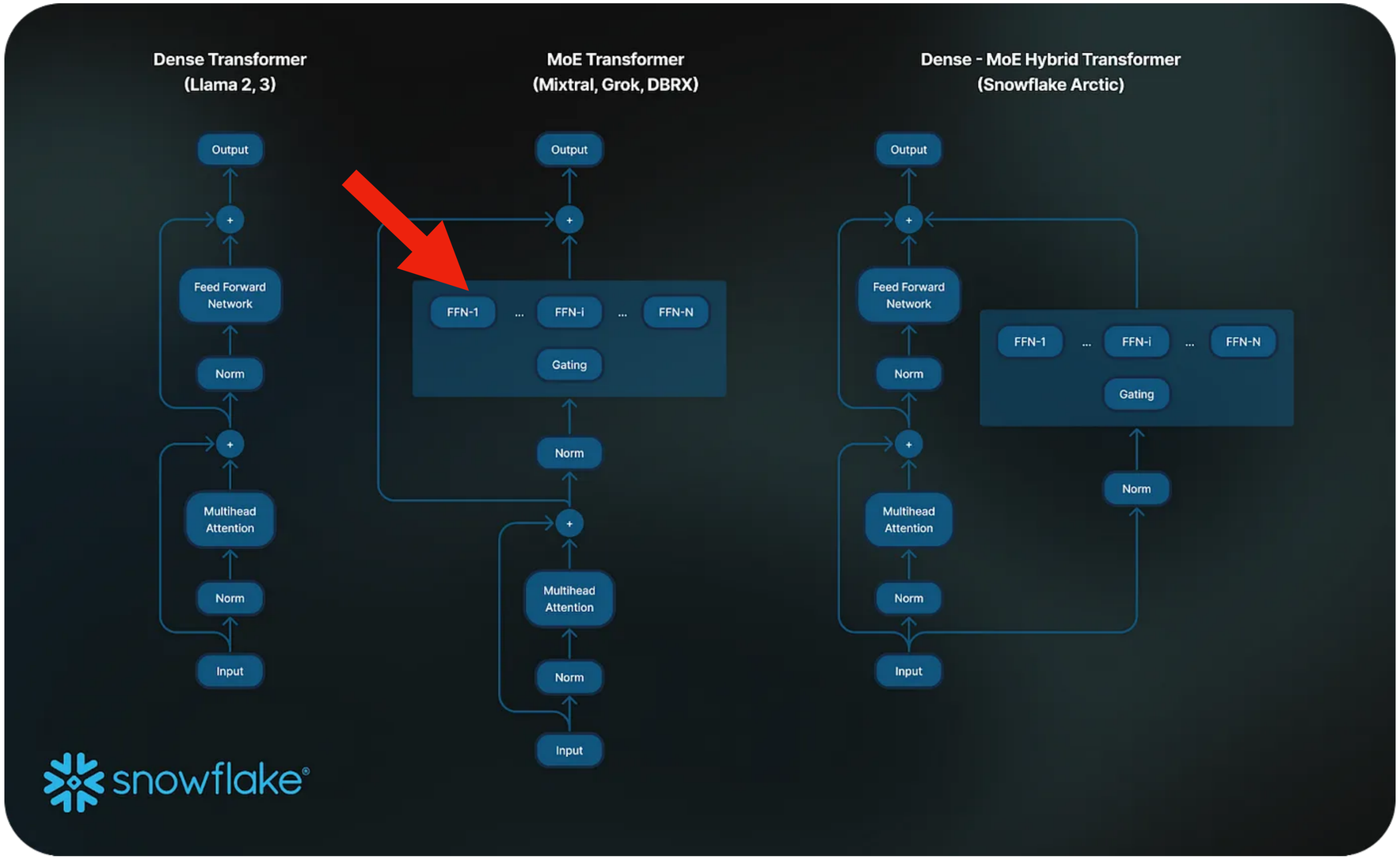
Rumor:
GPT4 / Gemini Architecrue



Open-weight MoE

	Org	#Experts	Active Params / Total Params
Mixtral 8x7B	Mistral	8	13B / 56B
Grok-1	X.AI	8	86B / 314B
DBRX	Databricks	16	36B / 132B
Arctic	Snowflake	128	17B / 480B
DeepSeekMoE	Deepseek	162 (2 shared)	21B / 236B

Dense vs MoE


Dense Transformer
(Llama 2, 3)
MoE Transformer
(Mixtral, Grok, DBRX)
Dense - MoE Hybrid Transformer
(Snowflake Arctic)
Output
Feed Forward Network
Norm
Multihead Attention
Norm
Input
FFN-1
FFN-i
FFN-N
Gating
snowflake

共享 Expert

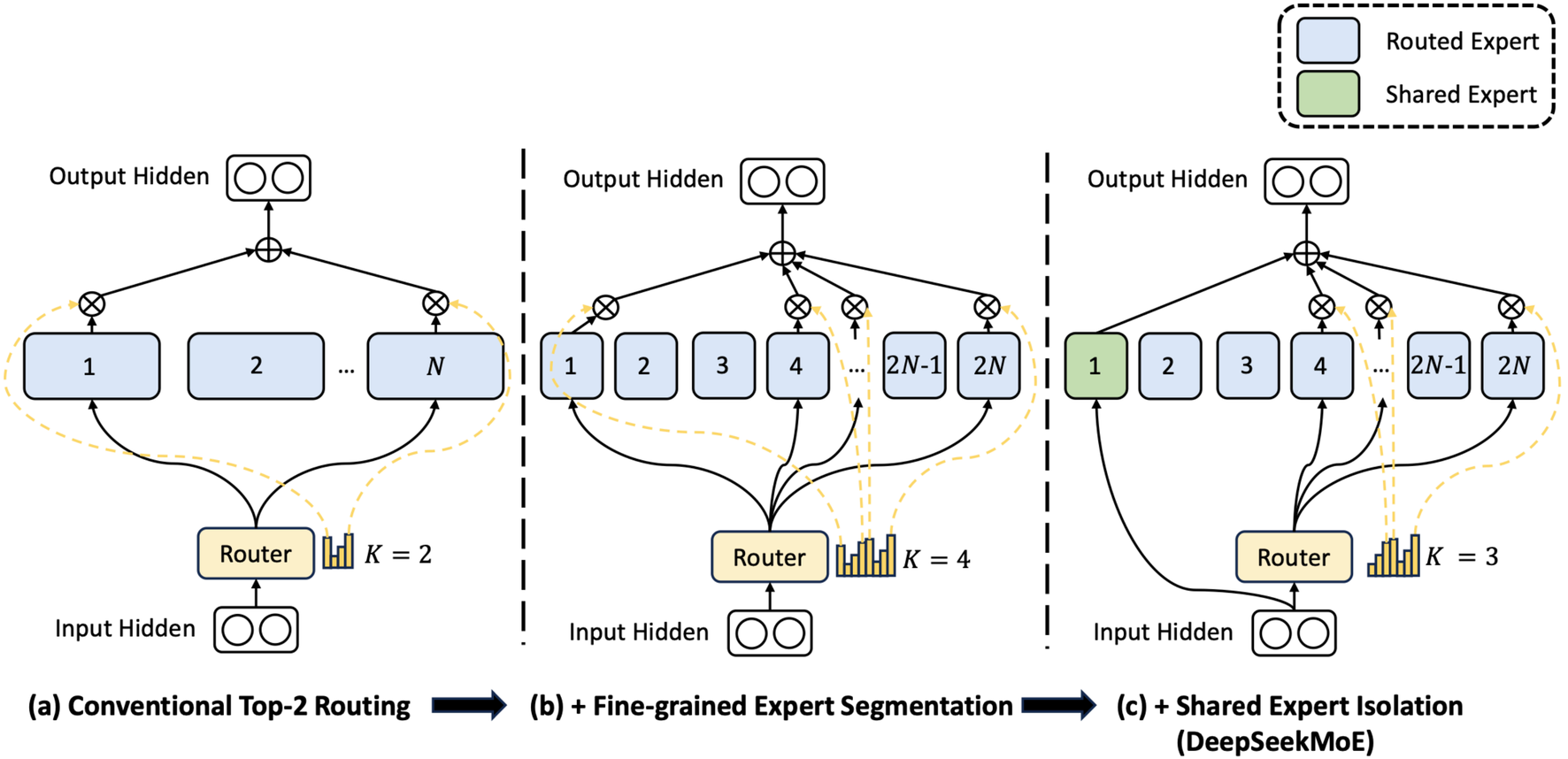


Figure source: DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models

Attention 也可以 MoE?

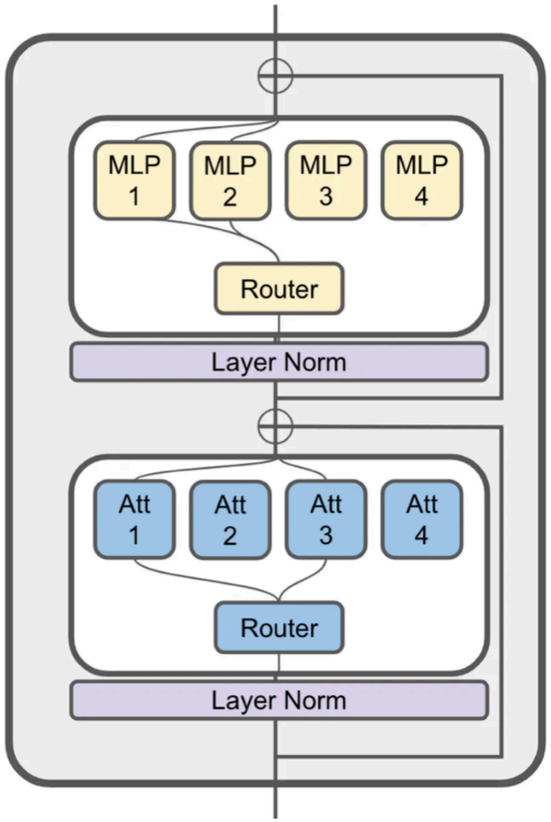


Figure 1: JetMoE architecture

Figure source:JetMoE: Reaching Llama2 Performance with 0.1M Dollars

Attention 也可以 MoE?

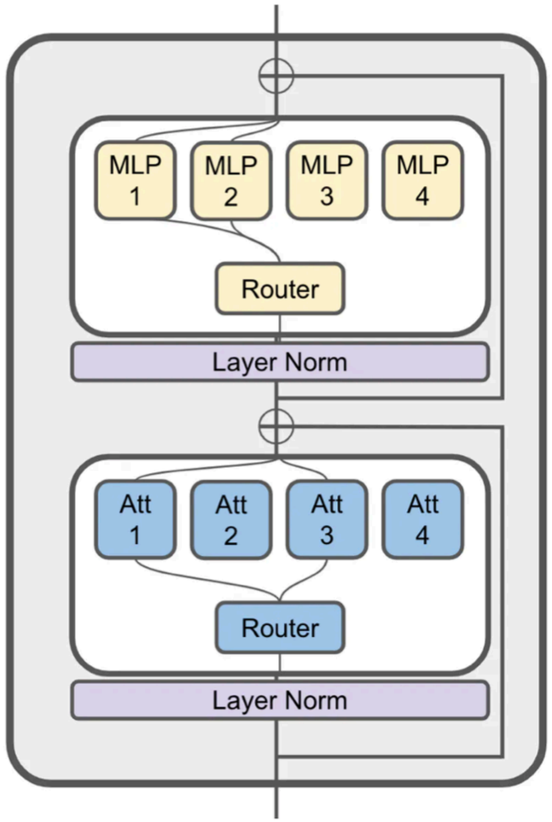


Figure 1: JetMoE architecture

Llama 3.1 405B 參數分佈

Tensors	Shape	Precision
model		
model.embed_tokens.weight	[128 256, 16 384]	BF16
model.layers.0		
model.layers.0.input_layernorm.weight	[16 384]	BF16
model.layers.0.mlp.down_proj.weight	[16 384, 53 248]	BF16
model.layers.0.mlp.gate_proj.weight	[53 248, 16 384]	BF16
model.layers.0.mlp.up_proj.weight	[53 248, 16 384]	BF16
model.layers.0.post_attention_layernorm.weight	[16 384]	BF16
model.layers.0.self_attn.k_proj.weight	[1 024, 16 384]	BF16
model.layers.0.self_attn.o_proj.weight	[16 384, 16 384]	BF16
model.layers.0.self_attn.q_proj.weight	[16 384, 16 384]	BF16
model.layers.0.self_attn.v_proj.weight	[1 024, 16 384]	BF16

Attn 做 MoE 好處不大

Figure source:JetMoE: Reaching Llama2 Performance with 0.1M Dollars

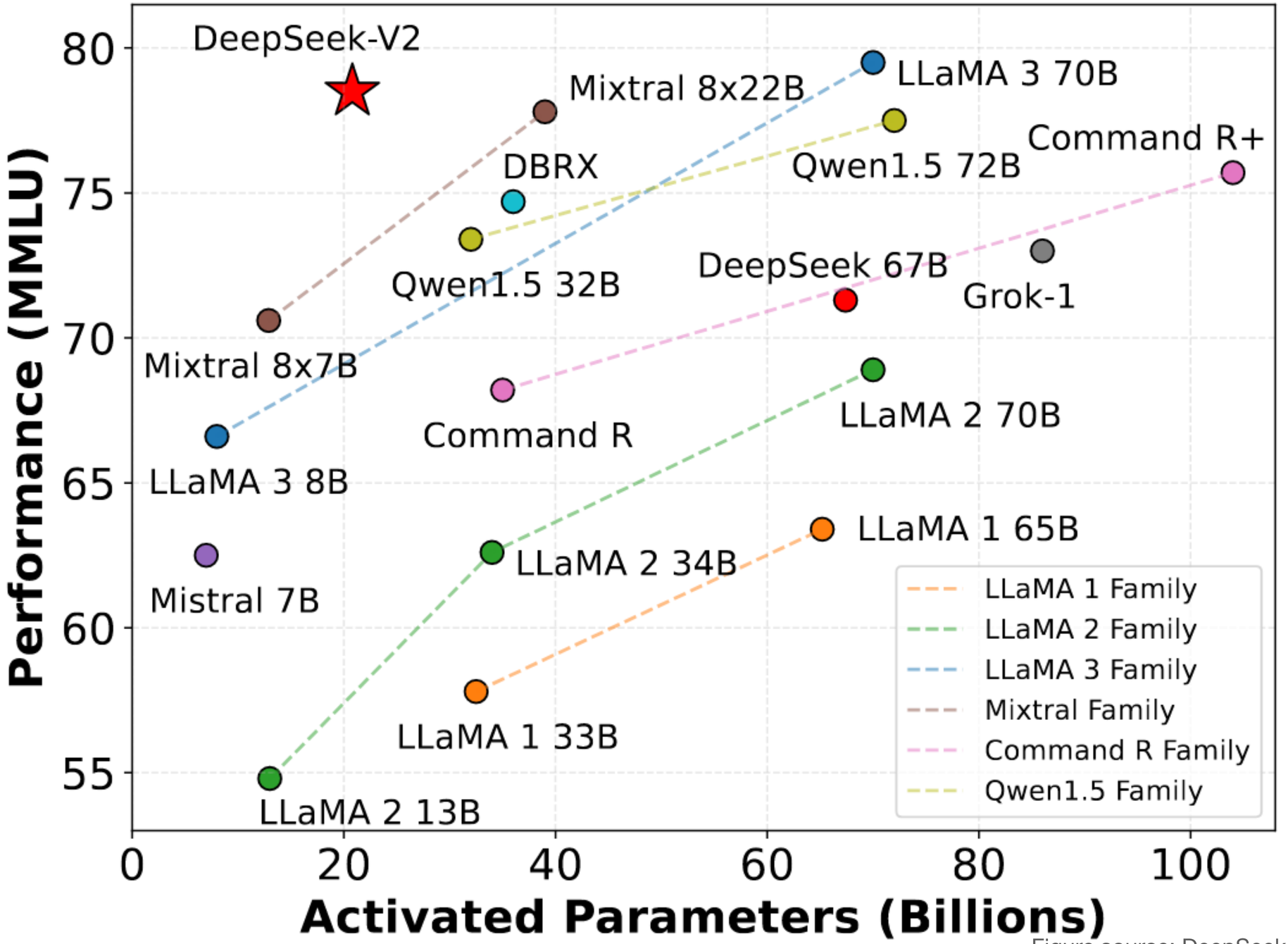


Figure source: DeepSeek

Token 先决

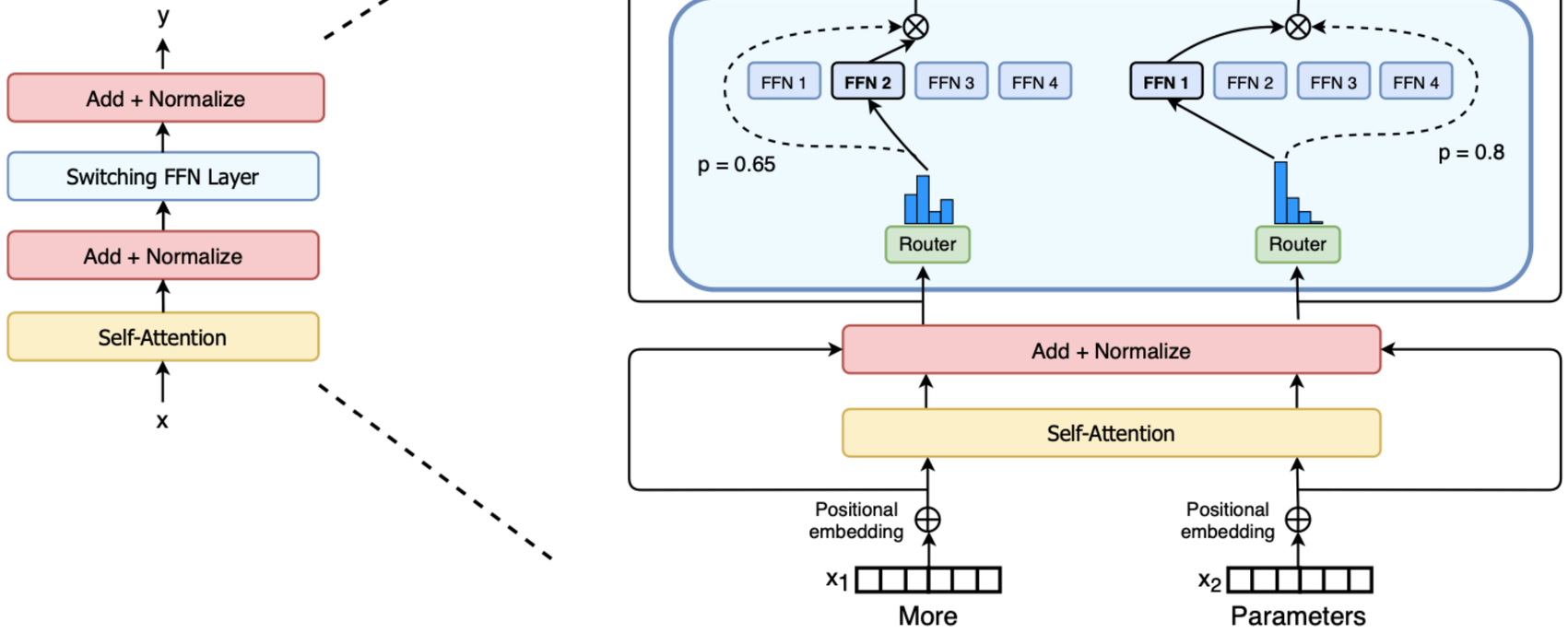
Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
LIAMFEDUS@GOOGLE.COM
Barret Zoph*
BARRETZOPH@GOOGLE.COM
Noam Shazeer
NOAM@GOOGLE.COM
Google, Mountain View, CA 94043, USA

SWITCH TRANSFORMERS

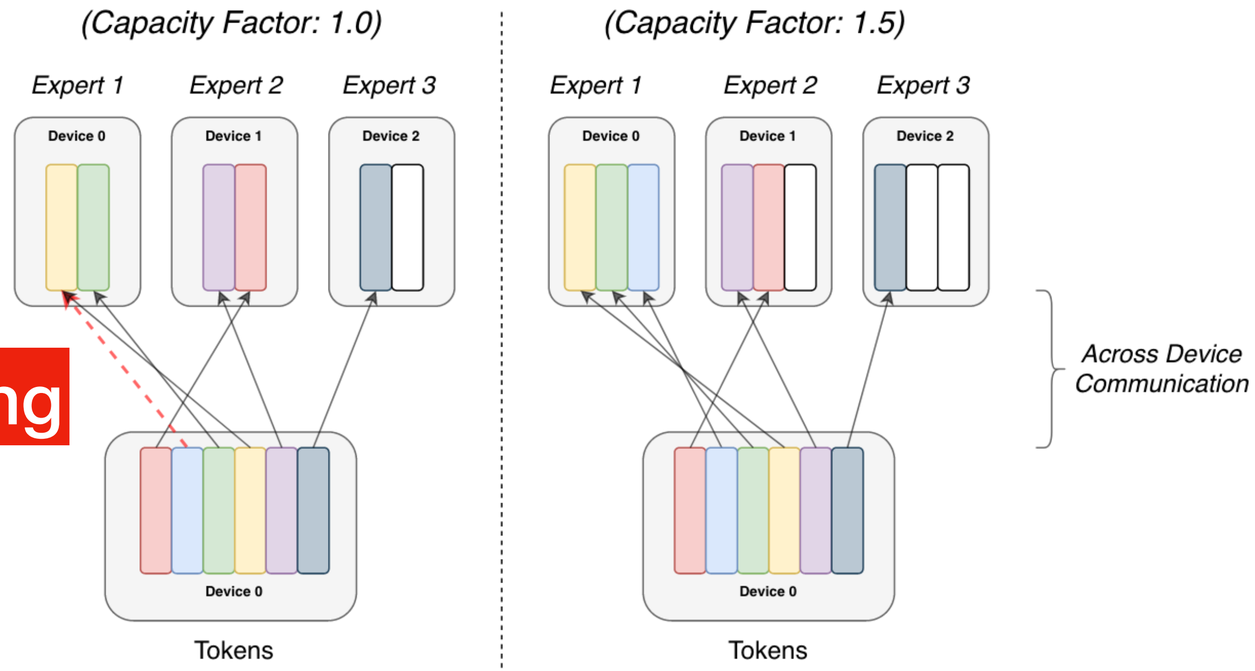
$$\sum_{e=1}^{E} \mathcal{G}_{s,e} \cdot \mathrm{FFN}_e(x_s)$$

$$G(x) := \mathrm{Softmax}(\mathrm{TopK}(x \cdot W_g)),$$



Token 塞車

Token dropping



$$expert\_capacity = \frac{num\_tokens}{num\_experts} \times capacity\_factor$$

Figure source: Stanford CS25: V1 I Mixture of Experts (MoE) paradigm and the Switch Transformer

負載平衡

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i \tag{4}$$

where f_i is the fraction of tokens dispatched to expert i,

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax}\, p(x) = i\} \tag{5}$$

and P_i is the fraction of the router probability allocated for expert i, [2]

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x). \tag{6}$$

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

負載平衡

	Expert 1	Expert 2
Token 1	0.5	0.5
Token 2	0.5	0.5
Token 3	0.5	0.5

f_1 = 2/3

P_1 = (0.5+0.5+0.5) / 3

f_2 = 1/3

P_2 = (0.5+0.5+0.5) / 3

Loss = 2/3 x 0.5 +
1/3 x 0.5
= 0.5

	Expert 1	Expert 2
Token 1	0.51	0.49
Token 2	0.51	0.49
Token 3	0	1

f_1 = 2/3

P_1 = (0.51+0.51) / 3

f_2 = 1/3

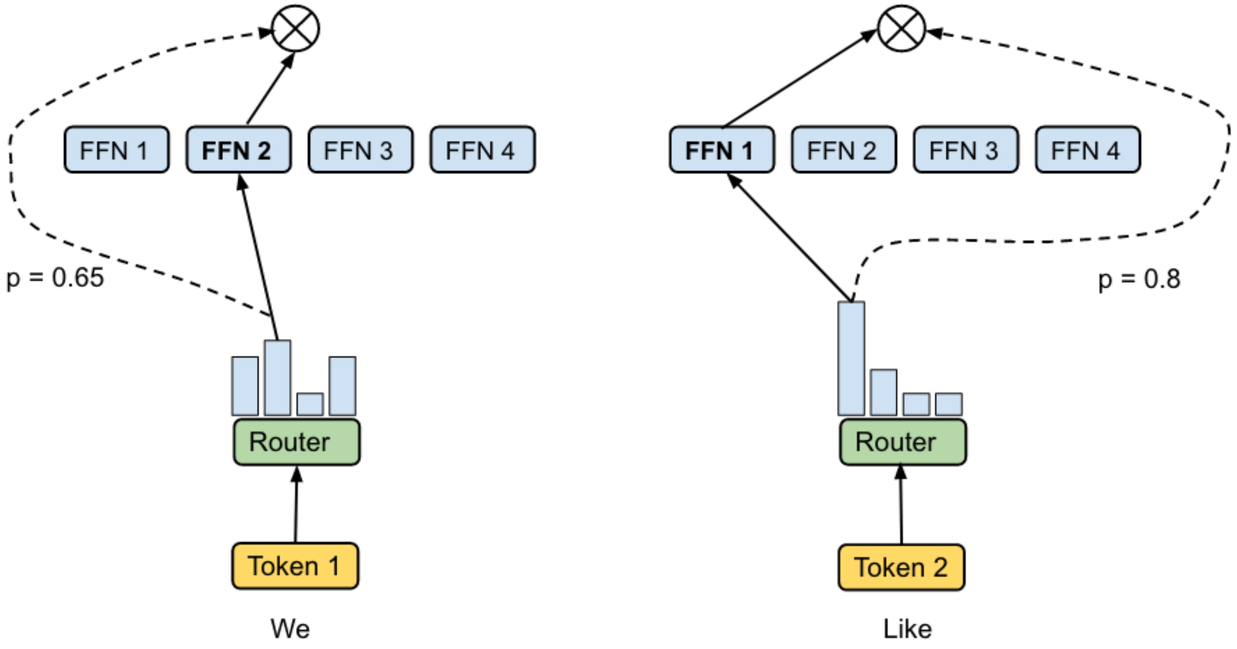
P_2 = (0.49+0.49+1)
/ 3

Loss = 0.447

Token 先決的問題

- 負載不平衡 / Balancing Loss 不穩定　　（Expert-choice）
- 每一個 token 用相同的 FLOPs　　（Mixture of Depth）

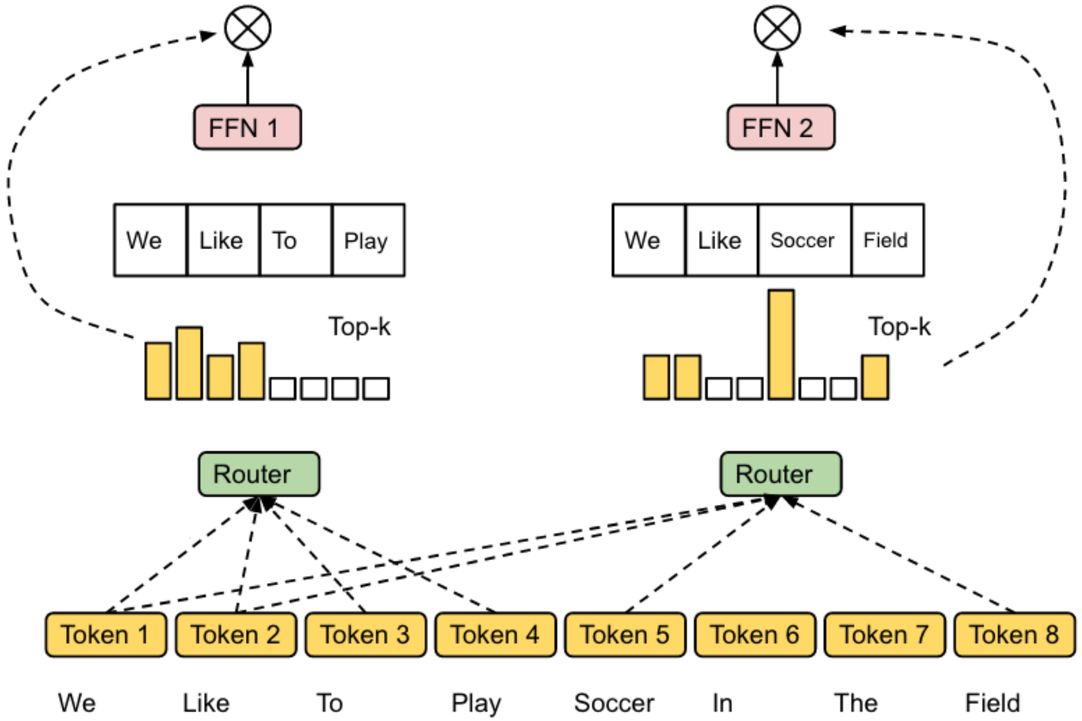
Token 先决



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

Expert 先决



效果顯著...嗎？

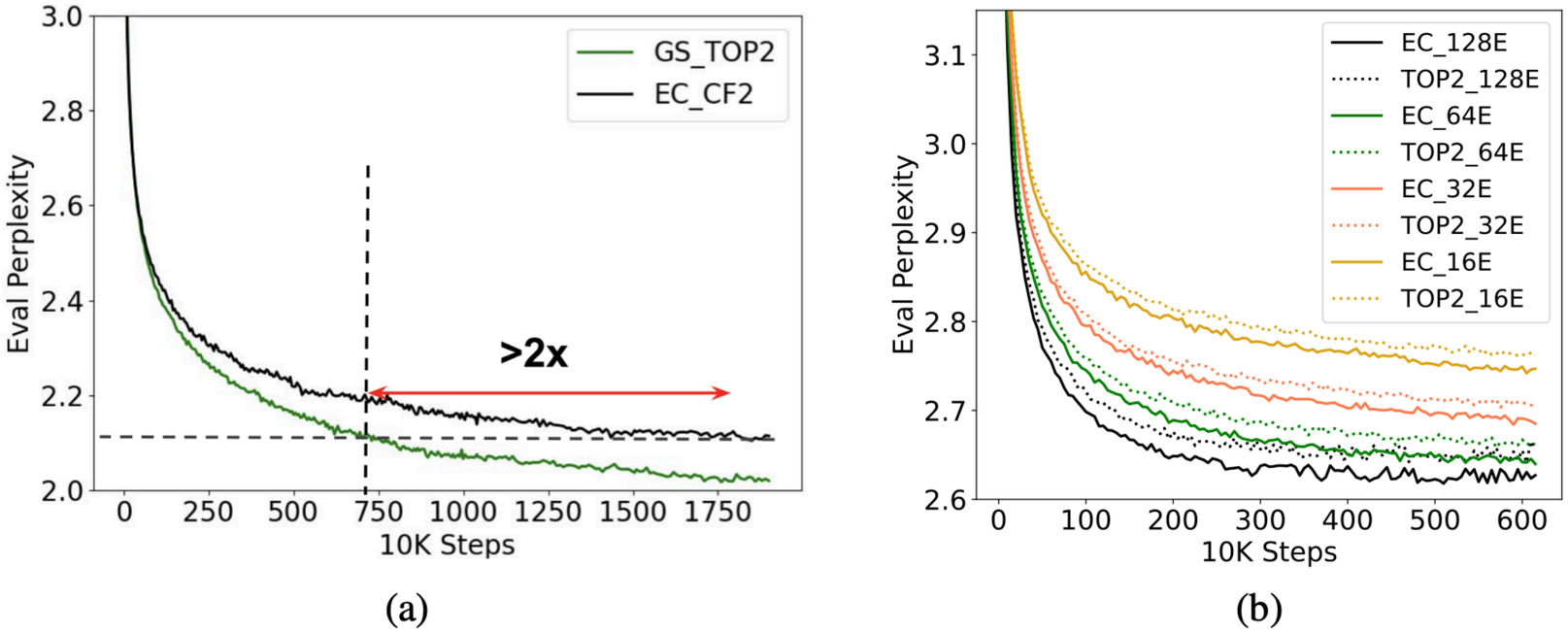


Figure 2: (a) Training convergence is more than 2x faster using our method compared to GShard top-2 gating. (b) Training perplexity scales strongly with the number of experts while keeping the expert size fixed. EC consistently outperforms GShard top-2 gating.

Figure source: Mixture-of-Experts with Expert Choice Routing

效果顯著...嗎？

6 Limitations

The expert choice method might not immediately apply to auto-regressive text generation as our current implementation takes in the past and future tokens to perform the top-k selection. One possible solution is to collect a large batch of input sequences, dispatch tokens of the same sequence into separate groups, and perform expert choice routing for each group. Another scenario where the expert choice method does not immediately apply is when the batch size becomes very small during serving or inference. A global top-k can be selected instead and we can cap the number of times each expert or token gets selected. We leave these possible improvements for future work.

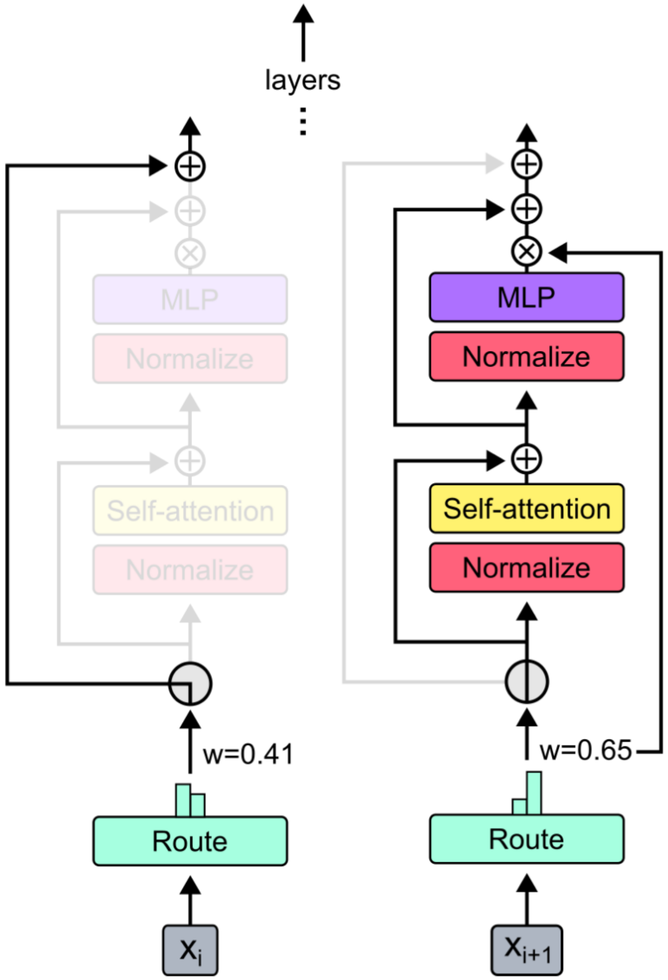
需要多解決一個問題: 訓練與推論時 router 的輸入不同

Figure source: Mixture-of-Experts with Expert Choice Routing

Token 先決的問題

- 負載不平衡 / Balancing Loss 不穩定　　（Expert-choice）
- 每一個 token 用相同的 FLOPs　　（Mixture of Depth）

跳層



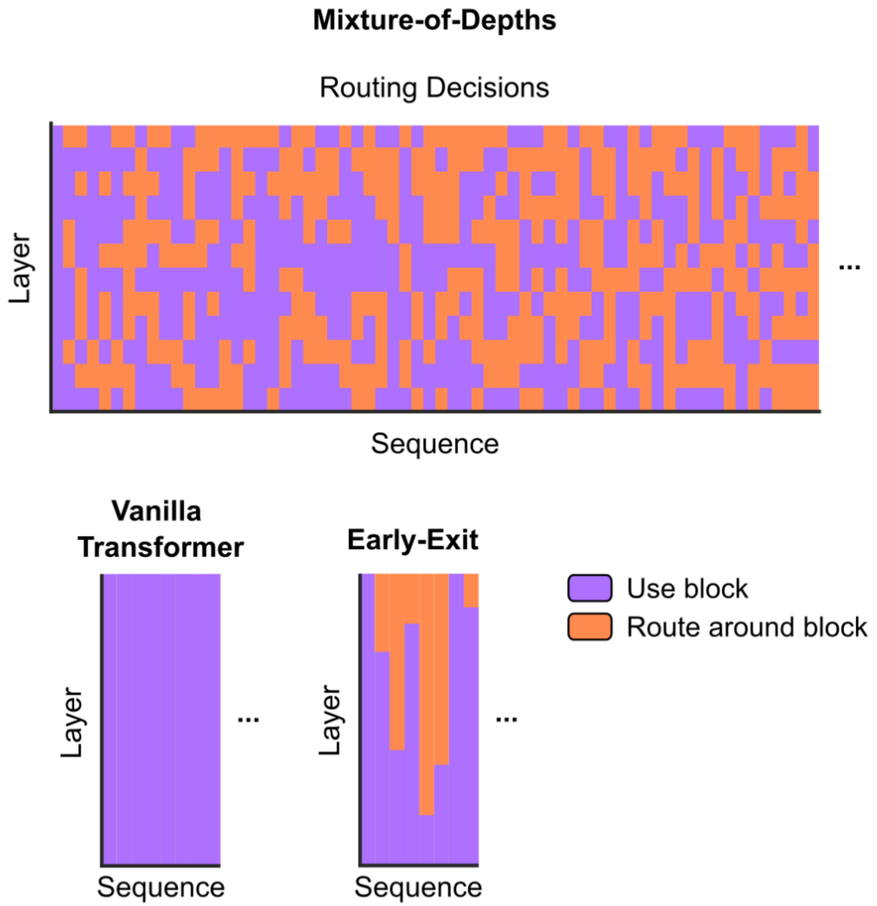


Figure source: Mixture-of-Depths: Dynamically allocating compute in transformer-based language models