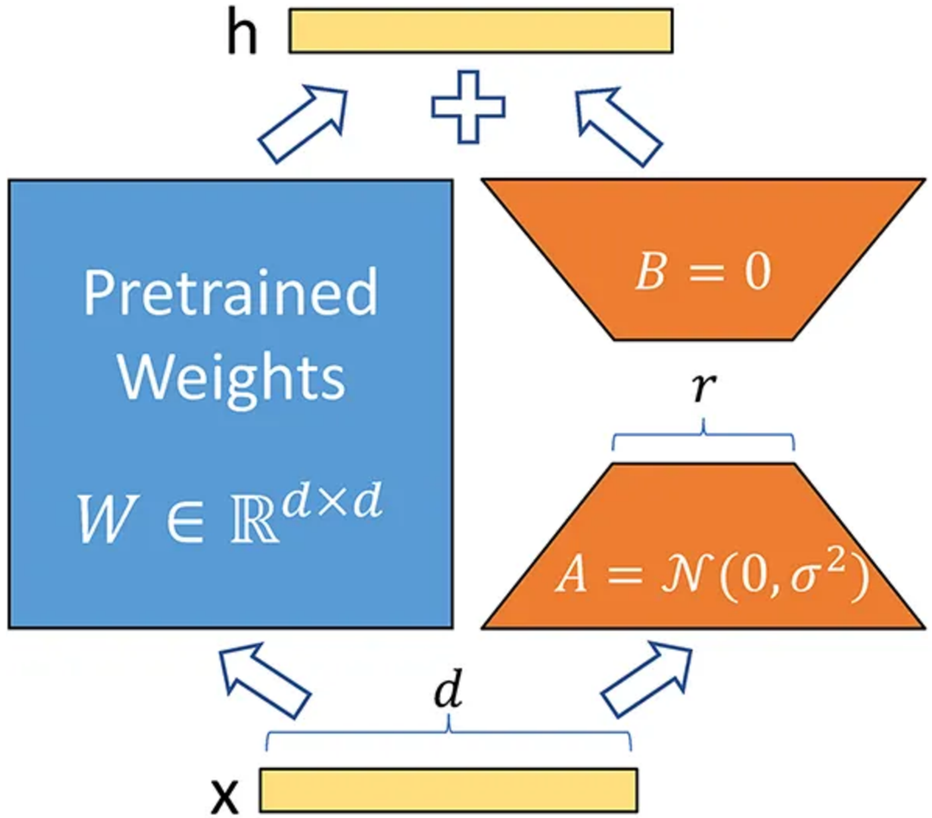
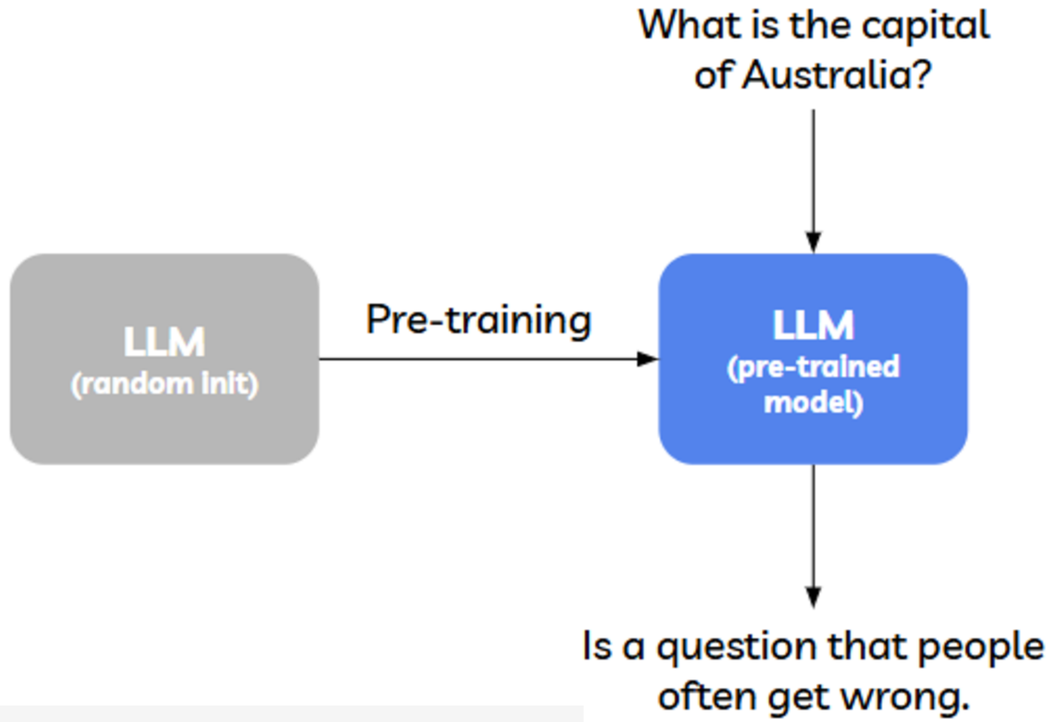
LoRA: Low-Rank Adaptation of Large Language Models



HOÀNG TIẾN ANH

A Minimal Explanation of Large Language Model

Step 1: Pre-training



Prompt: Explain the moon landing to a 6 year old in a few sentences.

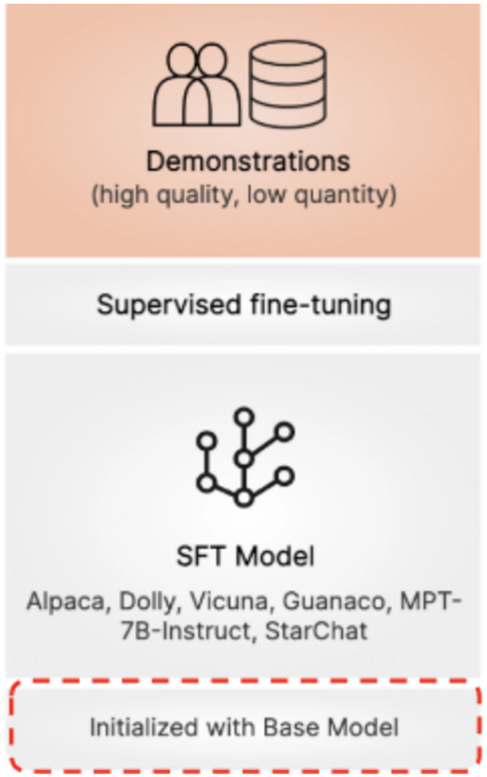
Completion: GPT-3

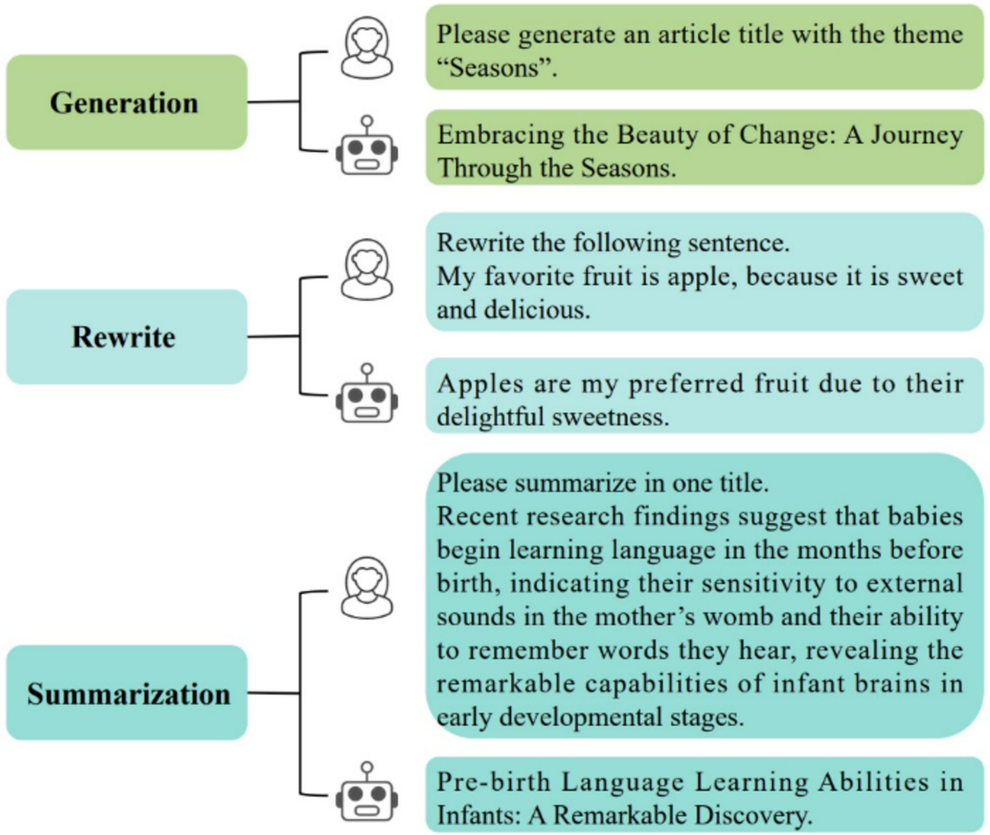
```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.
```

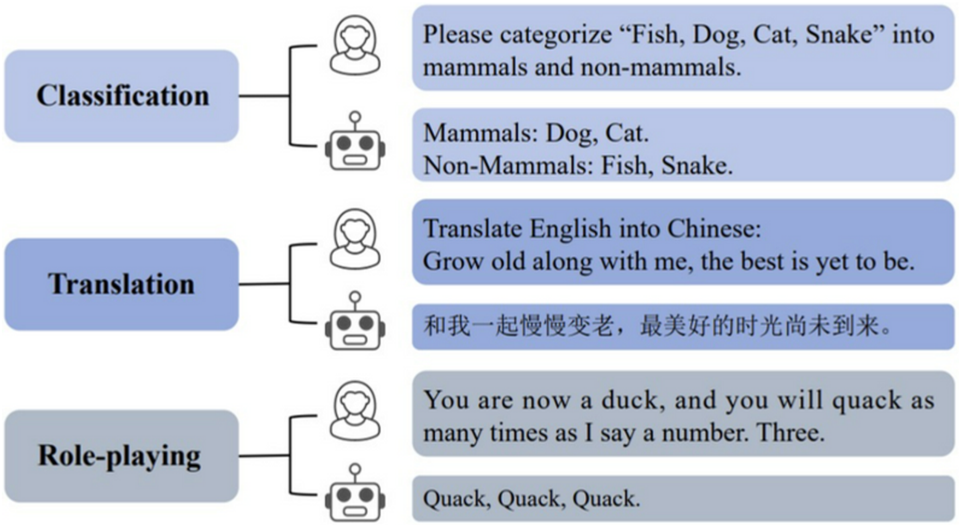
A Minimal Explanation of Large Language Model

Step 2: Fine tuning

tinh chỉnh LLM để cho nó trả lời như mình mong muốn.

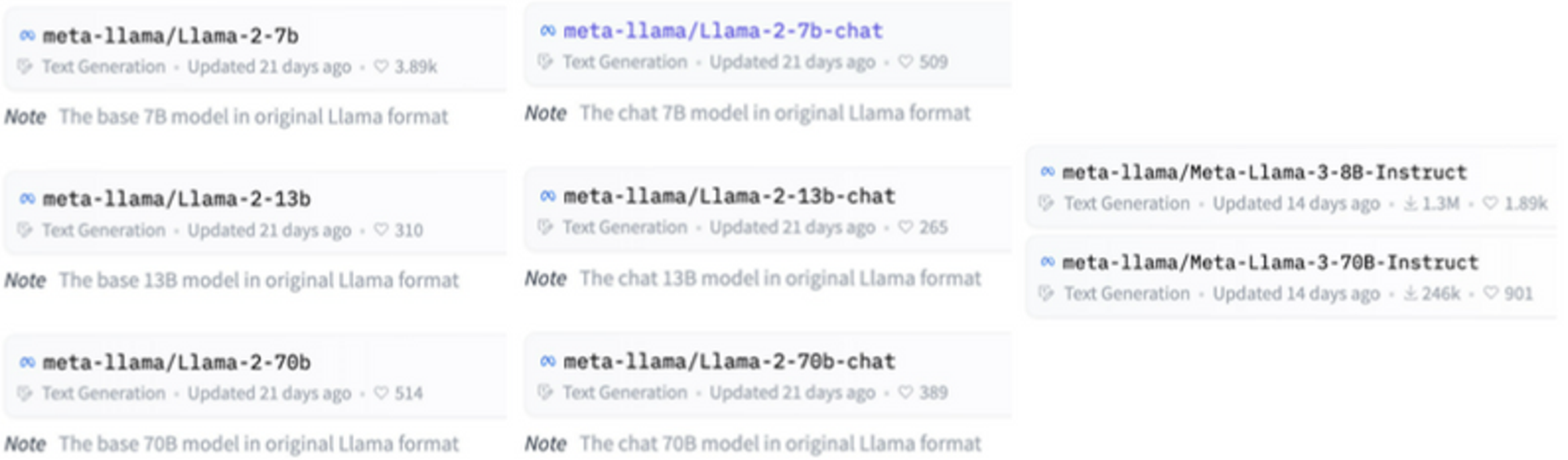




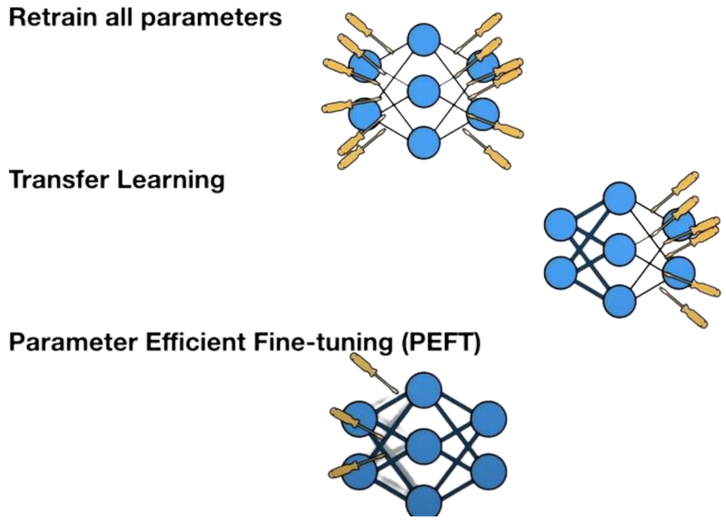


A Minimal Explanation of Large Language Model

Pre-training vs. Supervised Fine-tuning



Introduction



- Nhiều ứng dụng trong NLP dựa vào việc điều chỉnh một mô hình ngôn ngữ lớn đã được pretrained để phù hợp với nhiều nhiệm vụ cụ thể.
- Nhược điểm chính: việc fine tuning mô hình yêu cầu lưu trữ số lượng tham số lớn tương đương với mô hình gốc.
    - ví dụ: BERT-large, GPT-2 (354M), GPT-3 (175B), LLaMa, ...
- Nhiều nhà nghiên cứu đã tìm cách khắc phục vấn đề này bằng cách chỉ điều chỉnh một số tham số hoặc học các mô-đun bên ngoài cho các nhiệm vụ mới,
    - chỉ cần tải một số lượng nhỏ tham số dành riêng cho nhiệm vụ.
- Việc tinh chỉnh tất cả các tham số trong một mô hình lớn là quá tốn kém.

LORA

- Mô hình pre-trained có "low instrisic dimemsion" (chỉ một tập hợp nhỏ các tham số cần được update để có hiệu quả tốt)
- LoRA tận dụng điều này bằng cách tập trung vào việc điều chỉnh các ma trận có hạng thấp thay vì toàn bộ tham số của mô hình, giúp giảm đáng kể chi phí tính toán và lưu trữ mà vẫn đảm bảo hiệu suất cao.

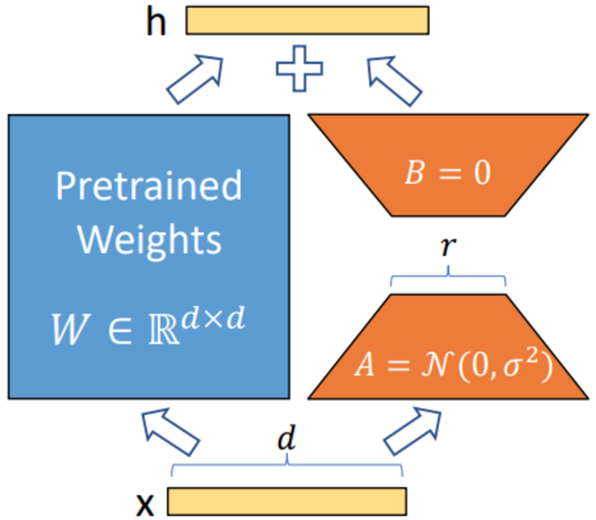
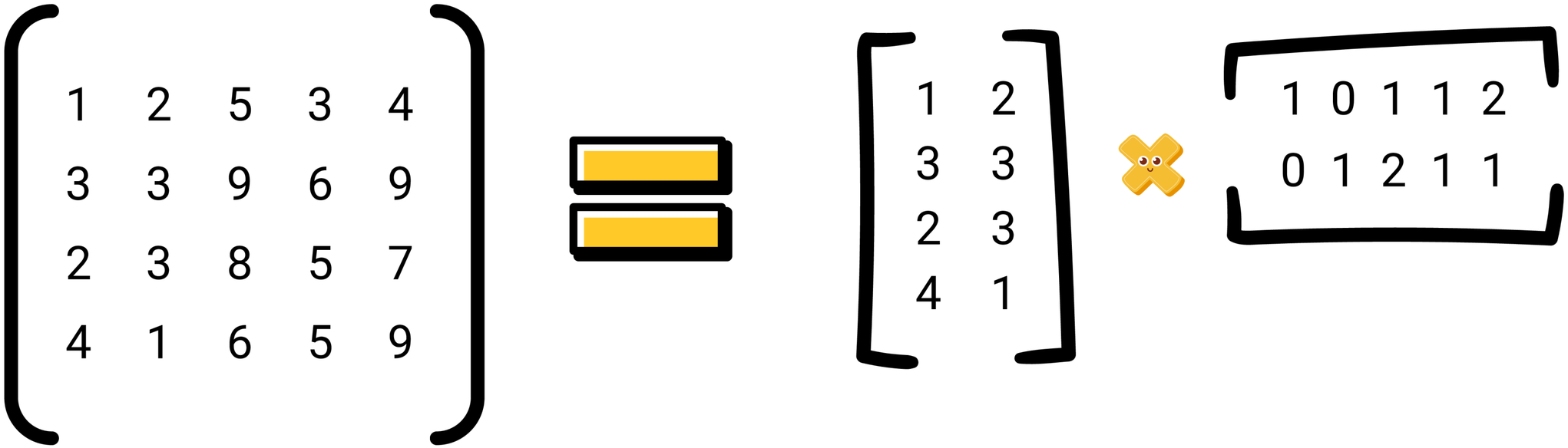


Figure 1: Our reparametrization. We only train A and B.

Low rank matrices



4x5

20

4x2

2x5

rank = 2

Practical Benefits and Limitations

- VRAM usage:
  - Tổng dung lượng VRAM cần thiết = tổng tham số gốc + tham số được tinh chỉnh.
  - Không cần theo dõi trạng thái của optimizer cho các tham số bị đóng băng, giúp tiết kiệm tài nguyên.
  - Ví dụ với GPT-3: 1.2TB xuống 350GB
- Kích thước checkpoint: $\frac{2\gamma r}{d_{\text{model}}}$
  - Với GPT-3: giảm từ 350GB xuống chỉ còn 35MB.
- Hiệu quả:
  - Tăng tốc 25% trong quá trình huấn luyện.
  - Không tăng độ trễ trong quá trình suy luận (inference latency).

Methods other

- Full Fine-tune
- Bias-only: Chỉ điều chỉnh các bias của mô hình, giảm đáng kể số lượng tham số cần cập nhật. Phạm vi điều chỉnh hạn chế, ảnh hưởng đến hiệu suất.
- Prefix-embedding tuning:
  - Thêm các token đặc biệt (prefix) vào đầu các token đầu vào.
- Prefix-layer tuning:
  - Tương tự Prefix-embedding tuning nhưng đi sâu hơn: học các embedding cho prefix sau mỗi lớp của mô hình.
  - Cách tiếp cận này cho phép mô hình học được sự thích nghi chi tiết hơn qua các layer khác nhau.
- Adapter tuning:
  - Thêm các lớp adapter giữa module (self-attention) hoặc module MLP và kết nối dư (residual connection).
  - Adapter được huấn luyện trong khi giữ nguyên các tham số gốc của mô hình, giảm chi phí điều chỉnh và tăng tính mô-đun hóa.

Results

r_q=r_v=4

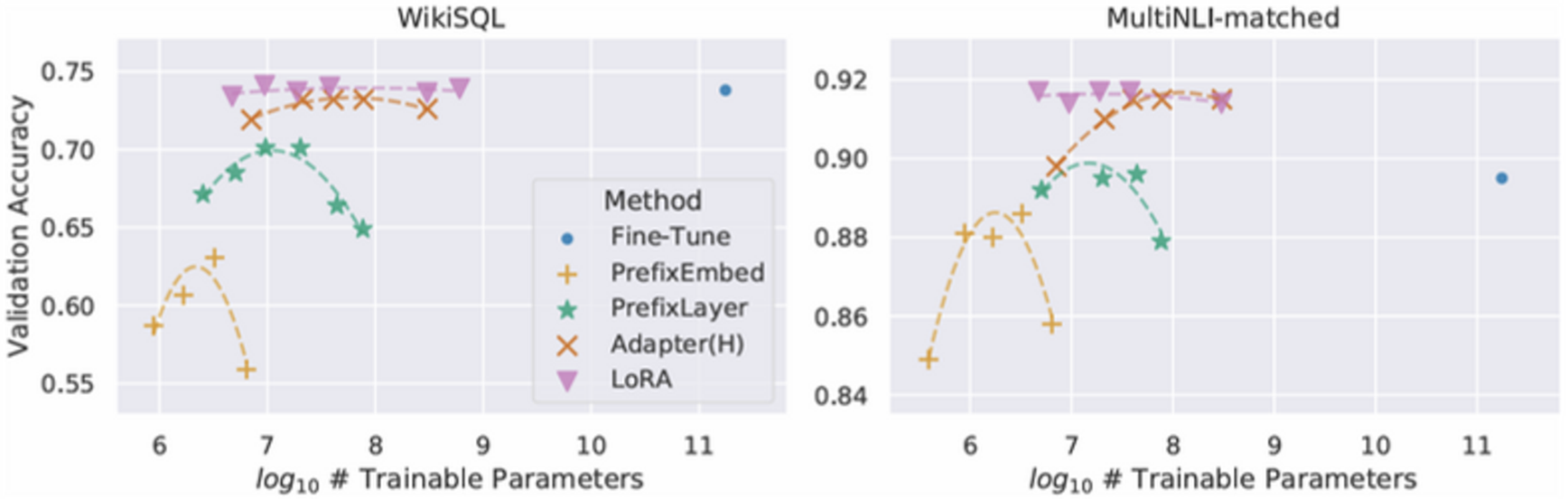
Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($Adapter^H$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($Adapter^H$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

Model	Parameters	Accuracy
PhoBERT-base	135M	96
PhoBERT-base (LoRA)	887k	90

Thực nghiệm :

- Data : UIT-VSFC (phân loại cảm xúc dựa trên phản hồi sinh viên)
- r = 8

Results



- Fine-Tune: Duy trì hiệu suất ổn định khi tăng tham số nhưng không mở rộng tốt.
- PrefixEmbed & PrefixLayer: Hiệu suất suy giảm khi số lượng token đặc biệt vượt quá ngưỡng tối ưu.
- Adapter(H): Đạt hiệu suất tốt nhưng không bằng LoRA.
- LoRA: Thể hiện khả năng mở rộng và hiệu suất vượt trội hơn so với các phương pháp khác.

Chúng ta nên áp dụng LoRA vào ma trận trọng số nào trong Transformer ?

- Chỉ điều chỉnh các trong module self-attention.
- Điều này cho thấy rằng điều chỉnh nhiều ma trận trọng số với rank nhỏ hơn hiệu quả hơn là điều chỉnh một loại trọng số với rank lớn hơn.

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

Rank r nào là tối ưu cho LoRA?

- rank nhỏ chỉ bằng 1 đã đủ để điều chỉnh cả W_q và W_v trên các tập dữ liệu này, trong khi việc chỉ điều chỉnh W_q yêu cầu một rank lớn hơn.
- Chỉ cần rank nhỏ là đạt được hiệu suất tốt => ma trận cập nhật có intrinsic rank rất nhỏ.
- Nếu bộ dữ liệu khác biệt và phức tạp hơn so với dữ liệu huấn luyện ban đầu, nên sử dụng giá trị hạng cao (64–256). Ngược lại, nếu vấn đề đơn giản và không có bộ dữ liệu phức tạp mới, giá trị hạng thấp (4–12) là đủ.

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

Ma trận Adaptation ΔW so với W:

- ΔW có sự tương quan mạnh mẽ hơn với W so với ma trận ngẫu nhiên => ΔW làm nổi bật một số đặc trưng đã có trong W.
- LoRA có thể làm tăng cường các đặc trưng quan trọng cho các nhiệm vụ đầu ra cụ thể.
- Làm nổi bật các hướng chưa được nhấn mạnh: ΔW không lặp lại các hướng đặc trưng hàng đầu của W, mà chỉ làm nổi bật các hướng chưa được nhấn mạnh.
- Yếu tố khuếch đại lớn: Yếu tố khuếch đại là rất lớn (21.5 ~ 6.91/0.32 cho r=4)

	$r = 4$			$r = 64$		
	ΔW_q	W_q	Random	ΔW_q	W_q	Random
$\|\|U^\top W_q V^\top\|\|_F =$	0.32	21.67	0.02	1.90	37.71	0.33
$\|\|W_q\|\|_F = 61.95$	$\|\|\Delta W_q\|\|_F = 6.91$			$\|\|\Delta W_q\|\|_F = 3.57$		

Conclusion

- Việc fine tuning LLM là rất tốn kém về phần cứng và chi phí lưu trữ/chuyển đổi để lưu trữ, LoRA đề xuất một chiến lược adaptation hiệu quả không làm tăng độ trễ inference và cũng không giảm độ dài chuỗi đầu vào trong khi vẫn duy trì chất lượng mô hình cao.
- Nó cho phép chuyển đổi nhiệm vụ nhanh chóng bằng cách chia sẻ phần lớn các tham số của mô hình.
- Mặc dù tác giả tập trung vào các mô hình Transformer, các nguyên lý đề xuất có thể áp dụng rộng rãi cho bất kỳ mạng nơ-ron nào với các dense layer.

FUTURE WORK

- Kết hợp LoRA với các phương pháp adaptation khác để cải thiện hiệu quả.
- Nghiên cứu rõ ràng cơ chế của LoRA và cách các đặc trưng học được trong huấn luyện trước được chuyển đổi cho các nhiệm vụ đầu ra.
- Tìm kiếm các phương pháp lý thuyết hơn để chọn ma trận trọng số cho LoRA, hiện tại chỉ chọn theo kinh nghiệm
- Khám phá việc thiếu hạng (rank-deficient) trong ΔW và W

Thank you