LORA: THÍ CH ỨNG CẤP THẤP CỦA MẠNG LAN LỚN MÔ HÌNH ĐO LƯỜNG

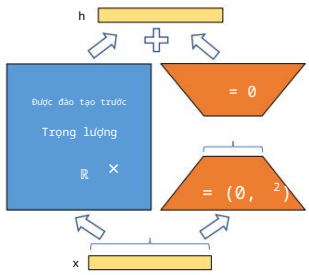
Edward Hu Yelong Shen Phillip Wallis Zeyuan Allen-Zhu
Yuanzhi Li Shean Wang Lu Wang Weizhu Chen
Tập đoàn Microsoft
{edwardhu, vâng, phwallis, zeyuana,
yuanzhil, lung, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(Phiên bản 2)

TRỪU TƯỢNG

Một mô hình quan trọng của xử lý ngôn ngữ tự nhiên bao gồm đào tạo trước trên quy mô lớn về dữ liệu miền chung và sự thích ứng với các nhiệm vụ hoặc miền cụ thể. BẰNG
chúng tôi đào tạo trước các mô hình lớn hơn, tinh chỉnh đầy đủ, đào tạo lại tất cả các tham số mô hình,
trở nên ít khả thi hơn. Lấy GPT-3 175B làm ví dụ – việc triển khai các phiên bản độc lập của
các mô hình được tinh chỉnh, mỗi phiên bản có tham số 175B, là điều bị cấm
đắt. Chúng tôi đề xuất Thích ứng Thứ hạng Thấp , hay LoRA, giúp cố định các trọng số của mô
hình được huấn luyện trước và đưa các ma trận phân rã thứ hạng có thể huấn luyện vào mỗi mô hình.
của kiến trúc Transformer, giúp giảm đáng kể số lượng tham số có thể huấn luyện cho các tác vụ
tiếp theo. So với GPT-3 175B được tinh chỉnh bằng Adam,
LoRA có thể giảm số lượng tham số có thể huấn luyện được 10.000 lần và
Yêu cầu bộ nhớ GPU gấp 3 lần. LoRA hoạt động ngang bằng hoặc tốt hơn việc tinh chỉnh chất
lượng mô hình trên RoBERTa, DeBERTa, GPT-2 và GPT-3, mặc dù có ít tham số có thể đào tạo hơn,
thông lượng đào tạo cao hơn và không giống như bộ điều hợp,
không có độ trễ suy luận bổ sung. Chúng tôi cũng cung cấp một cuộc điều tra thực nghiệm về
thiếu thứ hạng trong việc thích ứng mô hình ngôn ngữ, điều này làm sáng tỏ hiệu quả của
LoRA. Chúng tôi phát hành gói hỗ trợ tích hợp LoRA với PyTorch
lập mô hình và cung cấp các triển khai cũng như điểm kiểm tra mô hình của chúng tôi cho RoBERTa,
DeBERTa và GPT-2 tại https://github.com/microsoft/LoRA.

1 GIỚI THIỆU

Nhiều ứng dụng trong xử lý ngôn ngữ tự nhiên dựa vào việc điều chỉnh một mô hình ngôn ngữ được đào tạo trước, quy mô lớn cho nhiều ứng dụng xuôi dòng. Sự thích ứng như vậy thường được thực hiện thông qua việc tinh chỉnh,
cập nhật tất cả các tham số của mô hình được đào tạo trước. Nhược điểm chính của việc tinh chỉnh là mô hình mới chứa càng nhiều
các thông số như trong mô hình ban đầu. Khi các mô hình lớn hơn được đào tạo cứ sau vài tháng, điều này lại thay đổi từ một sự “bất tiện” đơn thuần đối với GPT-2 (Radford và cộng sự, b) hoặc RoBERTa lớn (Liu và cộng sự, 2019) thành thách thức triển khai quan trọng đối với GPT-3 (Brown và cộng sự, 2020) với 175 tỷ thông số có thể huấn luyện.1



Hình 1: Phép tái tham số hóa của chúng tôi. Chúng tôi chỉ đào tạo A và B.

Nhiều người đã tìm cách giảm thiểu điều này bằng cách chỉ điều chỉnh một số thông số hoặc học các mô-đun bên ngoài cho các nhiệm vụ mới. Bằng cách này, chúng ta chỉ cần để lưu trữ và tải một số lượng nhỏ các tham số dành riêng cho nhiệm vụ ngoài mô hình được đào tạo trước cho từng nhiệm vụ, giúp tăng cường đáng kể hiệu quả hoạt động khi triển khai. Tuy nhiên, các kỹ thuật hiện có

Đóng góp bình đẳng.

0So với V1, bản dự thảo này bao gồm các đường cơ sở tốt hơn, các thử nghiệm về GLUE và nhiều thông tin khác về độ trễ của bộ chuyển đổi.

1Trong khi GPT-3 175B đạt được hiệu suất không hề nhỏ chỉ bằng cách học vài lần, việc tinh chỉnh sẽ nâng cao hiệu suất của nó đáng kể như được nêu trong Phụ lục A.

thường gây ra độ trễ suy luận (Houlsby và cộng sự, 2019; Rebuffi và cộng sự, 2017) bằng cách mở rộng độ sâu mô hình hoặc giảm độ dài chuỗi có thể sử dụng của mô hình (Li & Liang, 2021; Lester và cộng sự, 2021; Ham bardzumyan và cộng sự, 2020; Liu và cộng sự, 2021) (Phần 3). Quan trọng hơn, các phương pháp này thường không phù hợp với các đường cơ sở tinh chỉnh, gây ra sự đánh đổi giữa hiệu quả và chất lượng mô hình.

Chúng tôi lấy cảm hứng từ Li et al. (2018a); Aghajanyan và cộng sự. (2020) cho thấy rằng các mô hình được tham số hóa quá mức đã học trên thực tế nằm ở một chiều nội tại thấp. Chúng tôi đưa ra giả thuyết rằng sự thay đổi về trọng số trong quá trình điều chỉnh mô hình cũng có "thứ hạng nội tại" thấp, dẫn đến phương pháp Thích ứng thứ hạng thấp (LoRA) được đề xuất của chúng tôi. LoRA cho phép chúng tôi huấn luyện gián tiếp một số lớp dày đặc trong mạng thần kinh bằng cách tối ưu hóa ma trận phân rã thứ hạng của sự thay đổi của các lớp dày đặc trong quá trình thích ứng, trong khi vẫn giữ cố định các trọng số được huấn luyện trước, như trong Hình 1. Sử dụng GPT-3 175B làm Ví dụ: chúng tôi cho thấy rằng thứ hạng rất thấp (tức là r trong Hình 1 có thể là một hoặc hai) là đủ ngay cả khi thứ hạng đầy đủ (tức là d) cao tới 12.288, làm cho LoRA trở nên hiệu quả cả về lưu trữ và tính toán.

LoRA sở hữu một số lợi thế chính.

- Một mô hình được đào tạo trước có thể được chia sẻ và sử dụng để xây dựng nhiều mô-đun LoRA nhỏ cho các nhiệm vụ khác nhau. Chúng ta có thể cố định mô hình dùng chung và chuyển đổi nhiệm vụ một cách hiệu quả bằng cách thay thế ma trận A và B trong Hình 1, giảm đáng kể yêu cầu lưu trữ và chuyển đổi nhiệm vụ qua đầu.

- LoRA giúp việc đào tạo hiệu quả hơn và giảm rào cản phần cứng để gia nhập tới 3 lần khi sử dụng trình tối ưu hóa thích ứng vì chúng tôi không cần tính toán độ dốc hoặc duy trì trạng thái trình tối ưu hóa cho hầu hết các tham số. Thay vào đó, chúng tôi chỉ tối ưu hóa các ma trận cấp thấp nhỏ hơn nhiều được chèn vào.

- Thiết kế tuyến tính đơn giản của chúng tôi cho phép chúng tôi hợp nhất các ma trận có thể huấn luyện với các trọng số cố định khi triển khai, không tạo ra độ trễ suy luận so với mô hình được tinh chỉnh hoàn toàn theo kết cấu.

- LoRA trực giao với nhiều phương pháp trước đó và có thể kết hợp với nhiều phương pháp trong số đó, chẳng hạn như như điều chỉnh tiền tố. Chúng tôi cung cấp một ví dụ trong Phụ lục E.

Thuật ngữ và quy ước Chúng tôi thường xuyên tham khảo kiến trúc Máy biến áp và sử dụng các thuật ngữ thông thường cho các kích thước của nó. Chúng tôi gọi kích thước kích thước đầu vào và đầu ra của mô hình lớp Transformer. Chúng tôi sử dụng Wq, Wk, Wv và Wo để chỉ các ma trận chiếu truy vấn/khóa/giá trị/đầu ra trong mô-đun tự chú ý. W hoặc W0 đề cập đến ma trận trọng số được huấn luyện trước và W cập nhật độ dốc tích lũy của nó trong quá trình thích ứng. Chúng tôi sử dụng r để biểu thị thứ hạng của mô-đun LoRA. Chúng tôi tuân theo các quy ước được đặt ra bởi (Vaswani và cộng sự, 2017; Brown và cộng sự, 2020) và sử dụng Adam (Loshchilov & Hutter, 2019; Kingma & Ba, 2017) để tối ưu hóa mô hình và sử dụng thứ nguyên tiếp liệu Transformer MLP df fn = 4 × dmô hình.

2 TUYÊN BỐ VẤN ĐỀ

Mặc dù đề xuất của chúng tôi không liên quan đến mục tiêu đào tạo, nhưng chúng tôi tập trung vào mô hình hóa ngôn ngữ như trường hợp sử dụng tạo động lực. Dưới đây là mô tả ngắn gọn về vấn đề mô hình hóa ngôn ngữ và đặc biệt là việc tối đa hóa xác suất có điều kiện đưa ra lời nhắc về nhiệm vụ cụ thể.

Giả sử chúng ta được cung cấp một mô hình ngôn ngữ tự hồi quy được huấn luyện trước PΦ(y|x) được tham số hóa bởi Φ. Ví dụ: PΦ(y|x) có thể là một trình học đa nhiệm chung chung như GPT (Radford và cộng sự, b; Brown và cộng sự, 2020) dựa trên kiến trúc Transformer (Vaswani và cộng sự, 2017). Hãy cân nhắc việc điều chỉnh mô hình được đào tạo trước này cho phù hợp với các tác vụ tạo văn bản có điều kiện xuôi dòng, chẳng hạn như tóm tắt, hiểu khả năng đọc của máy (MRC) và ngôn ngữ tự nhiên sang SQL (NL2SQL). Mỗi tác vụ xuôi dòng được biểu thị bằng tập dữ liệu huấn luyện gồm các cặp mục tiêu theo ngữ cảnh: Z = {(xi , yi)}i=1,..,N , trong đó cả xi và yi đều là các chuỗi mã thông báo. Ví dụ: trong NL2SQL, xi là một truy vấn ngôn ngữ tự nhiên và yi lệnh SQL tương ứng của nó; để tóm tắt, xi là nội dung của một bài viết và yi là phần tóm tắt của nó.

Trong quá trình tinh chỉnh hoàn toàn, mô hình được khởi tạo thành các trọng số được đào tạo trước Φ0 và được cập nhật thành Φ0 + ΔΦ bằng cách lặp đi lặp lại theo độ dốc để tối đa hóa mục tiêu mô hình hóa ngôn ngữ có điều kiện:

$$\max_{\Phi} \sum_{(x,y) \in Z} \sum_{t=1}^{|y|} \log \left(P_{\Phi}(y_t|x, y_{<t})\right) \quad (1)$$

Một trong những hạn chế chính của việc tinh chỉnh hoàn toàn là đối với mỗi tác vụ tiếp theo, chúng ta học một tập hợp tham số khác nhau ΔΦ có thứ nguyên |ΔΦ| bằng |Φ0|. Do đó, nếu mô hình được đào tạo trước có kích thước lớn (chẳng hạn như GPT-3 với |Φ0| ≈ 175 tỷ), việc lưu trữ và triển khai nhiều phiên bản độc lập của các mô hình được tinh chỉnh có thể là một thách thức, nếu có thể thực hiện được.

Trong bài viết này, chúng tôi áp dụng cách tiếp cận hiệu quả hơn về tham số, trong đó mức tăng tham số dành riêng cho nhiệm vụ ΔΦ = ΔΦ(Θ) được mã hóa thêm bằng một tập hợp tham số có kích thước nhỏ hơn nhiều Θ với |Θ| ≪ |Φ0|. Do đó, nhiệm vụ tìm ΔΦ trở nên tối ưu hóa trên Θ:

$$\max_{\Theta} \sum_{(x,y) \in Z} \sum_{t=1}^{|y|} \log p_{\Phi_0 + \Delta\Phi(\Theta)}(y_t|x, y_{<t}) \quad (2)$$

Trong các phần tiếp theo, chúng tôi đề xuất sử dụng biểu diễn cấp thấp để mã hóa ΔΦ vừa hiệu quả về tính toán vừa tiết kiệm bộ nhớ. Khi mô hình được huấn luyện trước là GPT-3 175B, số lượng tham số có thể huấn luyện |Θ| có thể nhỏ bằng 0,01% của |Φ0|.

3 GIẢI PHÁP HIỆN CÓ CHƯA ĐỦ TỐT ?

Vấn đề chúng tôi đặt ra để giải quyết không hề mới. Kể từ khi bắt đầu học chuyển giao, hàng chục công trình đã tìm cách làm cho việc điều chỉnh mô hình trở nên hiệu quả hơn về mặt tham số và tính toán. Xem Phần 6 để biết khảo sát về một số tác phẩm nổi tiếng. Lấy mô hình ngôn ngữ làm ví dụ, có hai chiến lược nổi bật khi nói đến khả năng thích ứng hiệu quả: thêm các lớp bộ điều hợp (Houlsby và cộng sự, 2019; Rebuffi và cộng sự, 2017; Pfeiffer và cộng sự, 2021; Ruckl ¨ e et al. , 2020) hoặc tối ưu hóa một số hình thức kích hoạt lớp đầu vào (Li & Liang, 2021; Lester và cộng sự, 2021; Hambardzumyan và cộng sự, 2020; Liu và cộng sự, 2021). Tuy nhiên, cả hai chiến lược đều có những hạn chế, đặc biệt là trong kịch bản sản xuất quy mô lớn và nhạy cảm với độ trễ.

Lớp bộ điều hợp giới thiệu độ trễ suy luận Có nhiều biến thể của bộ điều hợp. Chúng tôi tập trung vào thiết kế ban đầu của Houlsby et al. (2019) có hai lớp bộ điều hợp trên mỗi khối Transformer và một lớp mới hơn của Lin et al. (2020) chỉ có một khối cho mỗi khối nhưng có thêm LayerNorm (Ba và cộng sự, 2016). Mặc dù người ta có thể giảm độ trễ tổng thể bằng cách cắt bớt các lớp hoặc khai thác cài đặt đa tác vụ (Ruckl ¨ e et al., 2020; Pfeiffer et al., 2021), không có cách trực tiếp nào để bỏ qua tính toán bổ sung trong các lớp bộ điều hợp. Điều này có vẻ như không phải là vấn đề vì các lớp bộ điều hợp được thiết kế để có ít tham số (đôi khi <1% so với mô hình ban đầu) do có kích thước nút cổ chai nhỏ, điều này hạn chế FLOP mà chúng có thể thêm vào. Tuy nhiên, các mạng nơ-ron lớn dựa vào tính song song của phần cứng để giữ độ trễ ở mức thấp và các lớp bộ điều hợp phải được xử lý tuần tự. Điều này tạo ra sự khác biệt trong cài đặt suy luận trực tuyến trong đó kích thước lô thường nhỏ bằng một. Trong một kịch bản chung không có mô hình song song, chẳng hạn như chạy suy luận trên môi trường GPT-2 (Radford và cộng sự, b) trên một GPU, chúng tôi nhận thấy độ trễ tăng lên rõ rệt khi sử dụng bộ điều hợp, ngay cả với kích thước nút cổ chai rất nhỏ (Bảng 1).

Vấn đề này trở nên tồi tệ hơn khi chúng ta cần phân chia mô hình như đã làm trong Shoeybi et al. (2020); Lep ikhin và cộng sự. (2020), vì độ sâu bổ sung đòi hỏi các hoạt động GPU đồng bộ hơn như AllReduce và Broadcast, trừ khi chúng tôi lưu trữ các tham số bộ điều hợp dư thừa nhiều lần.

Tối ưu hóa trực tiếp lời nhắc là khó Hướng khác, như được minh họa bằng cách điều chỉnh tiền tố (Li & Liang, 2021), đối mặt với một thách thức khác. Chúng tôi nhận thấy rằng việc điều chỉnh tiền tố rất khó tối ưu hóa và hiệu suất của nó thay đổi không đơn điệu trong các tham số có thể huấn luyện được, xác nhận những quan sát tương tự trong bài báo gốc. Về cơ bản hơn, việc dành một phần độ dài chuỗi để điều chỉnh nhất thiết phải làm giảm độ dài chuỗi có sẵn để xử lý tác vụ xuôi dòng, điều mà chúng tôi nghi ngờ khiến việc điều chỉnh lời nhắc kém hiệu quả hơn so với các phương pháp khác. Chúng tôi trì hoãn việc nghiên cứu về thực hiện nhiệm vụ sang Phần 5.

Kích thước lô	32	16	1
Độ dài chuỗi |0|	512	256	128
	0,5 triệu	11 triệu	11 triệu
Tinh chỉnh/LoRA	1449,4±0,8	338,0±0,6	19,8±2,7
Bộ chuyển đổiL	1482,0±1,0 (+2,2%)	354,8±0,5 (+5,0%)	23,9±2,1 (+20,7%)
Bộ chuyển đổi H	1492,2±1,0 (+3,0%)	366,3±0,5 (+8,4%)	25,8±2,2 (+30,3%)

Bảng 1: Độ trễ suy ra của một lần chuyển tiếp trong môi trường GPT-2 được đo bằng mili giây, được tính trung bình trên 100 lần thử. Chúng tôi sử dụng NVIDIA Quadro RTX8000. "|0|" biểu thị số lượng có thể đào tạo các tham số trong các lớp bộ điều hợp. AdaptorL và AdaptorH là hai biến thể của việc điều chỉnh bộ chuyển đổi mà chúng tôi mô tả ở phần 5.1. Độ trễ suy luận được đưa ra bởi các lớp bộ điều hợp có thể đáng kể trong kịch bản trực tuyến, có độ dài chuỗi ngắn. Xem nghiên cứu đầy đủ ở Phụ lục B.

4 PHƯƠNG PHÁP CỦA CHÚNG TÔI

Chúng tôi mô tả thiết kế đơn giản của LoRA và những lợi ích thiết thực của nó. Các nguyên tắc nêu ở đây được áp dụng đến bất kỳ lớp dày đặc nào trong các mô hình học sâu, mặc dù chúng tôi chỉ tập trung vào một số trọng số nhất định trong Transformer mô hình ngôn ngữ trong các thử nghiệm của chúng tôi làm trường hợp sử dụng thúc đẩy.

4.1 MA TRẬN CẬP NHẬT THAM SỐ HẠNG THẤP

Mạng lưới thần kinh chứa nhiều lớp dày đặc thực hiện phép nhân ma trận. Cân nặng ma trận trong các lớp này thường có thứ hạng đầy đủ. Khi thích ứng với một nhiệm vụ cụ thể, Aghajanyan et al. (2020) cho thấy các mô hình ngôn ngữ được đào tạo trước có "kích thước nội tại" thấp và vẫn có thể học hiệu quả mặc dù có phép chiếu ngẫu nhiên tới một không gian con nhỏ hơn. Lấy cảm hứng từ điều này, chúng tôi đưa ra giả thuyết về kích thước của các bản cập nhật về trọng số cũng có "thứ hạng nội tại" thấp trong quá trình thích ứng. Đối với người được đào tạo trước ma trận trọng số $W_0 \in R^{d\times k}$, chúng tôi hạn chế cập nhật của nó bằng cách thể hiện cái sau bằng de r×k thứ hạng thấp thành phần $W_0 + \Delta W = W_0 + BA$, trong đó $B \in R^{d\times r}$, $A \in R^{r\times k}$, và thứ hạng r min(d, k). Trong quá trình huấn luyện, W_0 bị đóng băng và không nhận được cập nhật độ dốc, trong khi A và B chứa các dữ liệu có thể huấn luyện được thông số. Lưu ý cả W_0 và $\Delta W = BA$ đều được nhân với cùng một đầu vào và kết quả tương ứng của chúng vectơ đầu ra được tính tổng theo tọa độ. Với $h = W_0x$, kết quả chuyển tiếp được sửa đổi của chúng tôi mang lại:

$$h = W_0x + \Delta W x = W_0x + BAx \tag{3}$$

Chúng tôi minh họa việc tái tham số hóa của mình trong Hình 1. Chúng tôi sử dụng khởi tạo Gaussian ngẫu nhiên cho A và bằng 0 đối với B, vì vậy $\Delta W = BA$ bằng 0 khi bắt đầu huấn luyện. Sau đó chúng tôi chia tỷ lệ $\Delta W x$ theo đâu $\frac{\alpha}{r}$, là một hằng số trong r. Khi tối ưu hóa với Adam, việc điều chỉnh α gần giống như điều chỉnh việc học rate nếu chúng ta mở rộng quy mô khởi tạo một cách thích hợp. Kết quả là chúng ta chỉ cần đặt α thành r đầu tiên mà chúng ta thử và đừng điều chỉnh nó. Việc chia tỷ lệ này giúp giảm nhu cầu điều chỉnh lại các siêu tham số khi chúng ta thay đổi r (Yang & Hu, 2021).

Tổng quát về Tinh chỉnh đầy đủ. Một hình thức tinh chỉnh tổng quát hơn cho phép huấn luyện một tập hợp con của các tham số được huấn luyện trước. LoRA tiến thêm một bước nữa và không yêu cầu cập nhật độ dốc tích lũy cho ma trận trọng số để có thứ hạng đầy đủ trong quá trình thích ứng. Điều này có nghĩa là khi áp dụng LoRA cho tất cả các ma trận trọng số và huấn luyện tất cả các thành kiến[2], chúng tôi đại khái khôi phục lại biểu cảm tinh chỉnh hoàn toàn bằng cách đặt thứ hạng LoRA r thành thứ hạng của ma trận trọng số được huấn luyện trước. TRONG nói cách khác, khi chúng ta tăng số lượng tham số có thể huấn luyện được, LoRA[3] huấn luyện gần như hội tụ để huấn luyện mô hình ban đầu, trong khi các phương pháp dựa trên bộ chuyển đổi hội tụ về MLP và dựa trên tiền tố các phương thức cho một mô hình không thể có chuỗi đầu vào dài.

Không có độ trễ suy luận bổ sung. Khi được triển khai trong sản xuất, chúng tôi có thể tính toán một cách rõ ràng và lưu trữ $W = W_0 + BA$ và thực hiện suy luận như bình thường. Lưu ý rằng cả W_0 và BA đều thuộc $R^{d\times k}$. Khi cần chuyển sang tác vụ xuôi dòng khác, chúng ta có thể khôi phục W_0 bằng cách trừ BA và sau đó thêm một BA khác, một hoạt động nhanh chóng với rất ít chi phí bộ nhớ. Quan trọng là, điều này

[2] Chúng đại diện cho số lượng tham số không đáng kể so với trọng số.

[3] Một điều tất yếu khi thích nghi với những công việc khó khăn.

đảm bảo rằng chúng tôi không đưa ra bất kỳ độ trễ bổ sung nào trong quá trình suy luận so với mô hình được tinh chỉnh theo cấu trúc.

4.2 ÁP DỤNG LORA CHO MÁY BIẾN ÁP

Về nguyên tắc, chúng ta có thể áp dụng LoRA cho bất kỳ tập hợp con ma trận trọng số nào trong mạng nơron để giảm số lượng tham số có thể huấn luyện được. Trong kiến trúc Transformer, có bốn ma trận trọng số trong mô-đun tự chú ý (Wq, Wk, Wv, Wo) và hai ma trận trong mô-đun MLP. Chúng tôi coi Wq (hoặc Wk, Wv) là một ma trận đơn chiều dmodel ×dmodel, mặc dù thứ nguyên đầu ra thường được cắt thành các đầu chú ý. Chúng tôi giới hạn nghiên cứu của mình ở mức chỉ điều chỉnh trọng số chú ý cho các tác vụ xuôi dòng và cố định các mô-đun MLP (để chúng không được đào tạo về các tác vụ xuôi dòng) để đơn giản hóa và hiệu quả tham số. Chúng tôi nghiên cứu thêm về tác động của việc điều chỉnh các loại ma trận trọng số chú ý khác nhau trong máy biến áp ở phần 7.1. Chúng tôi để lại cuộc điều tra thực nghiệm về việc điều chỉnh các lớp MLP, các lớp LayerNorm và các thành kiến cho công việc trong tương lai.

Lợi ích và hạn chế thực tế. Lợi ích đáng kể nhất đến từ việc giảm mức sử dụng bộ nhớ và lưu trữ. Đối với một Transformer lớn được đào tạo với Adam, chúng tôi giảm mức sử dụng VRAM đó tới 2/3 nếu r dmodel vì chúng tôi không cần lưu trữ trạng thái tối ưu hóa cho các tham số cố định. Trên GPT-3 175B, chúng tôi giảm mức tiêu thụ VRAM trong quá trình đào tạo từ 1,2TB xuống 350GB. Với r = 4 và chỉ các ma trận chiếu truy vấn và giá trị được điều chỉnh, kích thước điểm kiểm tra giảm khoảng 10.000× (từ 350GB xuống 35MB)4 . Điều này cho phép chúng tôi đào tạo với số lượng GPU ít hơn đáng kể và tránh tắc nghẽn I/O. Một lợi ích khác là chúng tôi có thể chuyển đổi giữa các nhiệm vụ trong khi triển khai với chi phí thấp hơn nhiều bằng cách chỉ hoán đổi trọng số LoRA thay vì tất cả các tham số. Điều này cho phép tạo ra nhiều mô hình tùy chỉnh có thể được hoán đổi nhanh chóng trên các máy lưu trữ trọng số được huấn luyện trước trong VRAM. Chúng tôi cũng quan sát thấy tốc độ tăng 25% trong quá trình đào tạo trên GPT-3 175B so với tinh chỉnh hoàn toàn5 vì chúng tôi không cần tính toán độ dốc cho phần lớn các tham số.

LoRA cũng có những hạn chế của nó. Ví dụ: không dễ để phân nhóm đầu vào cho các tác vụ khác nhau với A và B khác nhau trong một lần chuyển tiếp, nếu người ta chọn hấp thụ A và B vào W để loại bỏ độ trễ suy luận bổ sung. Mặc dù có thể không hợp nhất các trọng số và tự động chọn các mô-đun LoRA để sử dụng cho các mẫu trong một đợt đối với các tình huống trong đó độ trễ không quan trọng.

5 THÍ NGHIỆM THỰC NGHIỆM

Chúng tôi đánh giá hiệu suất tác vụ hạ nguồn của LoRA trên RoBERTa (Liu và cộng sự, 2019), De BERTa (He và cộng sự, 2021) và GPT-2 (Radford và cộng sự, b), trước khi mở rộng lên GPT-3 175B (Brown và cộng sự, 2020). Các thử nghiệm của chúng tôi bao gồm nhiều nhiệm vụ khác nhau, từ hiểu ngôn ngữ tự nhiên (NLU) đến tạo (NLG). Cụ thể, chúng tôi đánh giá dựa trên điểm chuẩn GLUE (Wang và cộng sự, 2019) cho RoBERTa và DeBERTa. Chúng tôi làm theo thiết lập của Li & Liang (2021) trên GPT-2 để so sánh trực tiếp và thêm WikiSQL (Zhong và cộng sự, 2017) (truy vấn NL sang SQL) và SAMSum (Gliwa và cộng sự, 2019) (tóm tắt cuộc hội thoại) cho các thử nghiệm quy mô lớn trên GPT-3. Xem Phụ lục C để biết thêm chi tiết về bộ dữ liệu chúng tôi sử dụng. Chúng tôi sử dụng NVIDIA Tesla V100 cho tất cả các thử nghiệm.

5.1 CƠ SỞ

Để so sánh rộng rãi với các đường cơ sở khác, chúng tôi sao chép các thiết lập được sử dụng bởi công việc trước đó và sử dụng lại các số liệu được báo cáo của chúng bất cứ khi nào có thể. Tuy nhiên, điều này có nghĩa là một số đường cơ sở có thể chỉ xuất hiện trong một số thử nghiệm nhất định.

Tinh chỉnh (FT) là một cách tiếp cận phổ biến để thích ứng. Trong quá trình tinh chỉnh, mô hình được khởi tạo theo các trọng số và độ lệch được huấn luyện trước, đồng thời tất cả các tham số mô hình đều trải qua cập nhật độ dốc. Một biến thể đơn giản là chỉ cập nhật một số lớp trong khi đóng băng các lớp khác. Chúng tôi đưa vào một đường cơ sở như vậy được báo cáo trong nghiên cứu trước đây (Li & Liang, 2021) về GPT-2, chỉ điều chỉnh hai lớp cuối cùng (FTTop2).

4Chúng tôi vẫn cần model 350GB trong quá trình triển khai; tuy nhiên, việc lưu trữ 100 mô hình đã điều chỉnh chỉ yêu cầu 350GB + 35MB * 100 ≈ 354GB so với 100 * 350GB ≈ 35TB.

5Đối với GPT-3 175B, thông lượng đào tạo để tinh chỉnh hoàn toàn là 32,5 mã thông báo/giây trên mỗi GPU V100; với cùng một số lượng phân đoạn trọng lượng cho tính song song của mô hình, thông lượng là 43,1 mã thông báo/giây trên mỗi GPU V100 cho LoRA.

Mô hình & Phương pháp #	Có thể đào tạo Thông số	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoBbase (FT)*	125,0 triệu	87,6	94,8	90,2	63,6	92,8	91,9	78,7	91,2	86,4
RoBbase (BitFit)*	0,1 triệu	84,7	93,7	92,7	62,0	91,8	84,0	81,5	90,8	85,2
RoBbase (AdptD)*	0,3M	87,1±.0	94,2±.1	88,5±1,1	60,8±.4	93,1±.1	90,2±.0	71,5±2,7	89,7±.3	84,4
RoBbase (AdptD)*	0,9M	87,3±.1	94,7±.3	88.4± .1	62.6± .9	93.0±.2	90.6±.0	75.9±2.2	90.3±.1	85.4
Cơ sở RobB (LoRA)	0,3M	87,5±.3	95,1±.2	89,7± .7	63,4±1,2	93,3±.3	90,8±.1	86,6±.7	91,5±.2	87,2
RoBlarge (FT)*	355,0M	90,2	96,4	68,0	94,7	86,6	92,2	88,9	92,2	
RoBlarge (LoRA)	0,8M	90,6±.2	96,2±.5	90,9±1,2	68,2±1,9	94,9±.3	91,6±.1	87,4±2,5	92,6±.2	89,0
RoBlarge (AdptP)†	3,0M	90,2±.3	96.1±.3	90.2±.7	68.3±1.0	94.8±.2	91.9±.1	83.8±2.9	92.1±.7	88.4
RoBlarge (AdptP)†	0,8M	90,5±.3	96,6±.2	89,7 ±1,2	67,8±2,5	94,8±.3	91,7±.2	80,1±2,9	91,9±.4	87,9
RoBlarge (AdptH)†	6,0M	89,9±.5	96,2±.3	88,7 ±2,9	66,5±4,4	94,7±.2	92,1±.1	83,4±1,1	91,0±1,7	87,8
RoBlarge (AdptH)†	0,8M	90,3±.3	96,3±.5	87,7 ±1,7	66,3±2,0	94,7±.2	91,5±.1	72,9±2,9	91,5±.5	86,4
RoBlarge (LoRA)†	0,8M	90,6±.2	96,2±.5	90,2±1,0	68,2±1,9	94,8±.3	91,6±.2	85,2±1,1	92,3±.5	88,6
DeBXXL (FT)*	1500,0 triệu	91,8	97,2	93,9	92,9	96,1	92,0	96,0	92,7	
DeBXXL (LoRA)	4.7M	91.9±.2	96.9±.2	92.6±.6	72.4±1.1	96.0±.1	92.9±.1	94.9±.4	93.0±.2	91.3

Bảng 2: RoBERTabase, RoBERTalarge và DeBERTaXXL với các phương pháp thích ứng khác nhau trên Điểm chuẩn GLUE. Chúng tôi báo cáo độ chính xác tổng thể (khớp và không khớp) cho MNLI, Matthew's tương quan cho CoLA, tương quan Pearson cho STS-B và độ chính xác cho các nhiệm vụ khác. Cao hơn thì tốt hơn cho tất cả các số liệu. * chỉ ra những con số được công bố trong các tác phẩm trước đó. † cho biết các lần chạy được định cấu hình trong thiết lập tương tự như Houlsby et al. (2019) để so sánh công bằng.

Chỉ thiên vị hoặc BitFit là đường cơ sở trong đó chúng tôi chỉ huấn luyện các vectơ thiên vị trong khi đóng băng mọi thứ khác. Hiện tại, đường cơ sở này cũng đã được BitFit nghiên cứu (Zaken và cộng sự, 2021).

Điều chỉnh nhúng tiền tố (PreEmbed) chèn các mã thông báo đặc biệt vào trong số các mã thông báo đầu vào. Những mã thông báo đặc biệt này có các từ nhúng có thể huấn luyện được và thường không có trong từ vựng của mô hình. Ở đâu việc đặt các mã thông báo như vậy có thể có tác động đến hiệu suất. Chúng tôi tập trung vào "tiền tố", thêm vào trước các mã thông báo như vậy cho lời nhắc và "infixing", thêm vào lời nhắc; cả hai đều được thảo luận trong Li & Lương (2021). Chúng tôi sử dụng lp (resp. li) biểu thị số lượng mã thông báo tiền tố (resp. infix). Số lượng các tham số có thể huấn luyện được là |Θ| = dmodel × (lp + li).

Điều chỉnh lớp tiền tố (PreLayer) là phần mở rộng của điều chỉnh nhúng tiền tố. Thay vì chỉ học từ nhúng (hoặc tương đương, kích hoạt sau lớp nhúng) cho một số đặc biệt mã thông báo, chúng tôi tìm hiểu các kích hoạt sau mỗi lớp Transformer. Các kích hoạt được tính toán từ các lớp trước đó được thay thế đơn giản bằng các lớp có thể huấn luyện được. Số lượng kết quả của các tham số có thể huấn luyện được là |Θ| = L × dmodel × (lp + li), trong đó L là số lớp Transformer.

Điều chỉnh bộ điều hợp theo đề xuất trong Houlsby et al. (2019) chèn các lớp bộ điều hợp giữa mô-đun tự chú ý (và mô-đun MLP) và kết nối dư tiếp theo. Có hai các lớp được kết nối đầy đủ với các độ lệch trong một lớp bộ điều hợp có tính phi tuyến ở giữa. Chúng tôi gọi đây là thiết kế ban đầu AdaptorH . Gần đây, Lin và cộng sự. (2020) đã đề xuất một thiết kế hiệu quả hơn với lớp bộ điều hợp chỉ được áp dụng sau mô-đun MLP và sau LayerNorm. Chúng tôi gọi nó là AdaptorL . Cái này rất giống với một thiết kế khác được đề xuất trong Pfeiffer et al. (2021), mà chúng tôi gọi là AdaptorP . Chúng tôi cũng bao gồm một lệnh gọi cơ sở khác AdaptorDrop (Ruckl ¨ e et al., 2020) loại bỏ một số lớp bộ điều hợp để có hiệu quả cao hơn (AdapterD). Chúng tôi trích dẫn số liệu từ các công trình trước bất cứ khi nào có thể để tối đa hóa số lượng đường cơ sở mà chúng tôi so sánh; chúng nằm thành hàng có dấu hoa thị (*) ở cột đầu tiên. Trong mọi trường hợp, chúng ta có |Θ| = L^Adpt ×(2×dmodel ×r+r+dmodel)+ 2×L^LN ×dmodel trong đó L^Adpt là số lớp bộ điều hợp và L^LN số lượng LayerNorms có thể huấn luyện được (ví dụ: trong AdaptorL).

LoRA bổ sung các cặp ma trận phân rã thứ hạng có thể huấn luyện song song với các ma trận trọng số hiện có. Như đã đề cập ở Phần 4.2, chúng tôi chỉ áp dụng LoRA cho Wq và Wv trong hầu hết các thử nghiệm để đơn giản. Số lượng tham số có thể huấn luyện được xác định bởi thứ hạng r và hình dạng của các trọng số ban đầu: |Θ| = 2 × L^LoRA × dmodel × r, trong đó L^LoRA là số ma trận trọng số mà chúng ta áp dụng LoRA.

Mô hình & Phương pháp	# Có thể đào tạo Thông số	Thử thách E2E NLG BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354,92M	68,2	0,37M	8,62	46,2	71,0	2,47
GPT-2 M (Bộ chuyển đổiL)*	66,3			8,41	45,0	69,8	2,40
GPT-2 M (Bộ chuyển đổiL)*	11,09M	68,9	11,09M	8,71	46,1	71,3	2,47
GPT-2 M (Bộ chuyển đổiH)	67,3±.6	8,50±.07	46,0±.2	25,19M	68,1	0,35M	69,7	70,7±,2	2,44 ± 0,01
GPT-2 M (FTTop2)*	0,35M	70,4±.1		8,59	46,0	70,8	2,41
GPT-2 M (Lớp trước)*	8,85±.02	46,8±.2		8,81	46,1	71,4	2,49
GPT-2 M (LoRA)						71,8±,1	2,53±.02
GPT-2 L (FT)*	774.03M	68.5		8,78	46,0	69,9	2,45
GPT-2 L (Bộ chuyển đổiL)	0,88M	69,1±.1	8,68±.03	46,3±.0	23,00M	68,9±.3	71,4±, 2	2,49±.0
GPT-2 L (Bộ chuyển đổiL)	8,70±.04	46,1±.1	0,77M	70,3	0,77M	70,4±.1	71,3±,2	2,45 ± 0,02
GPT-2 L (Lớp trước)*	8,89±.02	46,8±.2		8,85	46,2	71,7	2,47
GPT-2 L (LoRA)						72,0±.2	2,47±.02

Bảng 3: GPT-2 vừa (M) và lớn (L) với các phương pháp thích ứng khác nhau trên E2E NLG
Thử thách. Đối với tất cả các số liệu, càng cao càng tốt. LoRA vượt trội hơn một số đường cơ sở có thể so sánh được
hoặc ít tham số có thể huấn luyện được. Khoảng tin cậy được hiển thị cho các thử nghiệm chúng tôi đã chạy. * biểu thị
số đã được công bố trong các tác phẩm trước đó.

5.2 ROBERTA CƠ SỞ/LỚN

RoBERTa (Liu và cộng sự, 2019) đã tối ưu hóa công thức đào tạo trước được đề xuất ban đầu trong BERT (Devlin
et al., 2019a) và tăng cường hiệu suất nhiệm vụ sau này mà không cần giới thiệu nhiều thứ có thể đào tạo hơn
thông số. Trong khi RoBERTa đã bị các mô hình lớn hơn nhiều vượt qua trên bảng xếp hạng NLP
chẳng hạn như tiêu chuẩn GLUE (Wang và cộng sự, 2019) trong những năm gần đây, nó vẫn là một tiêu chuẩn cạnh tranh và
mô hình được đào tạo trước phổ biến về quy mô của nó đối với những người thực hành. Chúng tôi sử dụng cơ sở RoBERTa được đào tạo trước
(125M) và RoBERTa lớn (355M) từ thư viện HuggingFace Transformers (Wolf và cộng sự, 2020)
và đánh giá hiệu suất của các phương pháp thích ứng hiệu quả khác nhau đối với các nhiệm vụ từ GLUE
điểm chuẩn. Chúng tôi cũng nhân rộng Houlsby et al. (2019) và Pfeiffer et al. (2021) theo
cài đặt. Để đảm bảo so sánh công bằng, chúng tôi thực hiện hai thay đổi quan trọng về cách đánh giá LoRA khi
so sánh với các bộ điều hợp. Đầu tiên, chúng tôi sử dụng cùng một kích thước lô cho tất cả các tác vụ và sử dụng độ dài chuỗi
là 128 để phù hợp với đường cơ sở của bộ chuyển đổi. Thứ hai, chúng tôi khởi tạo mô hình thành mô hình được đào tạo trước cho
MRPC, RTE và STS-B, không phải là mô hình đã được điều chỉnh cho phù hợp với MNLI như đường cơ sở tinh chỉnh. Chạy
theo thiết lập hạn chế hơn này từ Houlsby et al. (2019) được gắn nhãn †. Kết quả là
được trình bày trong Bảng 2 (Ba phần trên cùng). Xem Phần D.1 để biết chi tiết về các siêu tham số được sử dụng.

5.3 DEBERTA XXL

DeBERTa (He và cộng sự, 2021) là một biến thể gần đây hơn của BERT được đào tạo trên một nền tảng lớn hơn nhiều
mở rộng quy mô và hoạt động rất cạnh tranh trên các tiêu chuẩn như GLUE (Wang và cộng sự, 2019) và Su perGLUE (Wang
và cộng sự, 2020). Chúng tôi đánh giá liệu LoRA vẫn có thể phù hợp với hiệu suất của một
tinh chỉnh DeBERTa XXL (1.5B) trên GLUE. Kết quả được trình bày trong Bảng 2 (Phần dưới cùng).
Xem Phần D.2 để biết chi tiết về các siêu tham số được sử dụng.

5.4 GPT-2 TRUNG BÌNH/LỚN

Đã chứng minh rằng LoRA có thể là một giải pháp thay thế cạnh tranh cho việc tinh chỉnh hoàn toàn trên NLU, chúng tôi hy vọng
câu trả lời nếu LoRA vẫn chiếm ưu thế trên các mô hình NLG, chẳng hạn như GPT-2 vừa và lớn (Radford và cộng sự,
b). Chúng tôi giữ thiết lập của mình gần nhất có thể với Li & Liang (2021) để so sánh trực tiếp. Quá hạn
do hạn chế về không gian, chúng tôi chỉ trình bày kết quả của mình về Thử thách E2E NLG (Bảng 3) trong phần này.
Xem Phần F.1 để biết kết quả trên WebNLG (Gardent và cộng sự, 2017) và DART (Nan và cộng sự, 2020). Chúng tôi
bao gồm danh sách các siêu tham số được sử dụng trong Phần D.3.

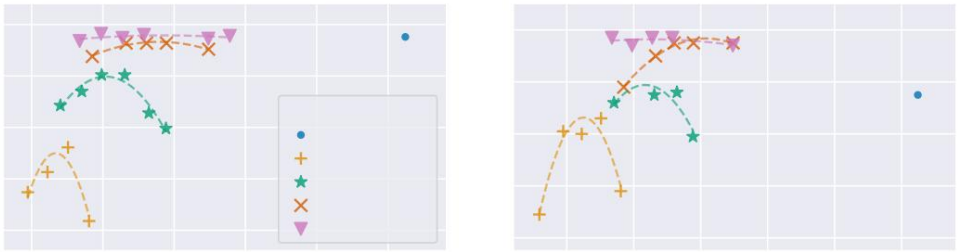
Mô hình & Phương pháp	# Thông số có thể huấn luyện được	WikiSQL Acc. (%)	MNLI-m Tích lũy. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175.255,8 triệu	73,8	89,5	52.0/28.0/44.5
GPT-3 (BitFit)	14,2 triệu	71,3	91,0	51,3/27,4/43,5
GPT-3 (Nhúng sẵn)	3,2	63,1	88,6	48,3/24,2/40,5
GPT-3 (Lớp trước)	triệu 20,2	70,1	89,5	50,8/27,3/43,5
GPT-3 (Bộ chuyển đổiH)	triệu 7,1 triệu	71,9	89,8	53,0/28,9/44,8
GPT-3 (Bộ chuyển đổiH)	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

Bảng 4: Hiệu suất của các phương pháp thích ứng khác nhau trên GPT-3 175B. Chúng tôi báo cáo dạng logic độ chính xác xác thực trên WikiSQL, độ chính xác xác thực trên MultiNLI-matched và Rouge-1/2/L trên SAMSum. LoRA hoạt động tốt hơn các phương pháp trước đây, bao gồm cả việc tinh chỉnh đầy đủ. Kết quả trên WikiSQL có dao động khoảng ±0,5%, MNLI-m khoảng ±0,1% và SAMSum khoảng ±0,2/±0,2/±0,1 cho ba chỉ số.

5.5 TỶ LỆ LÊN GPT-3 175B

Là bài kiểm tra căng thẳng cuối cùng cho LoRA, chúng tôi mở rộng quy mô lên GPT-3 với 175 tỷ thông số. Do độ cao chi phí đào tạo, chúng tôi chỉ báo cáo độ lệch chuẩn điển hình cho một nhiệm vụ nhất định trên các hạt giống ngẫu nhiên, như phản đối việc cung cấp một cho mỗi mục. Xem Phần D.4 để biết chi tiết về các siêu tham số được sử dụng.

Như được hiển thị trong Bảng 4, LoRA khớp hoặc vượt quá đường cơ sở tinh chỉnh trên cả ba bộ dữ liệu. Ghi chú rằng không phải tất cả các phương pháp đều được hưởng lợi đơn điệu từ việc có nhiều tham số có thể huấn luyện hơn, như trong Hình 2. Chúng tôi quan sát thấy hiệu suất giảm đáng kể khi chúng tôi sử dụng hơn 256 mã thông báo đặc biệt cho điều chỉnh nhúng tiền tố hoặc hơn 32 mã thông báo đặc biệt để điều chỉnh lớp tiền tố. Điều này chứng thực những quan sát tương tự trong Li & Liang (2021). Trong khi một cuộc điều tra kỹ lưỡng về hiện tượng này nằm ngoài phạm vi của công việc này, chúng tôi nghi ngờ rằng việc có nhiều mã thông báo đặc biệt hơn sẽ khiến việc phân phối đầu vào dịch chuyển xa hơn so với phân phối dữ liệu trước đào tạo. Một cách riêng biệt, chúng tôi điều tra hiệu suất của các phương pháp thích ứng khác nhau trong chế độ dữ liệu thấp trong Phần F.3.



Hình 2: Độ chính xác xác thực GPT-3 175B so với số lượng tham số có thể huấn luyện của một số điều chỉnh các phương thức trên WikiSQL và MNLI phù hợp. LoRA thể hiện khả năng mở rộng và hiệu suất tác vụ tốt hơn. Xem Phần F.2 để biết thêm chi tiết về các điểm dữ liệu được vẽ.

6 CÔNG TRÌNH LIÊN QUAN

Mô hình ngôn ngữ biến đổi. Máy biến áp (Vaswani và cộng sự, 2017) là máy biến áp theo trình tự kiến trúc tận dụng nhiều sự chú ý của bản thân. Radford và cộng sự. (a) áp dụng nó vào mô hình ngôn ngữ tự hồi quy bằng cách sử dụng một bộ bộ giải mã Transformer. Kể từ đó, ngôn ngữ dựa trên Transformer các mô hình đã thống trị NLP, đạt được tính tiên tiến trong nhiều nhiệm vụ. Một mô hình mới xuất hiện với BERT (Devlin et al., 2019b) và GPT-2 (Radford et al., b) – cả hai đều là lan Transformer lớn

các mô hình đo lường được đào tạo trên một lượng lớn văn bản – trong đó việc tinh chỉnh dữ liệu dành riêng cho nhiệm vụ sau khi đào tạo trước về dữ liệu miền chung mang lại hiệu suất tăng đáng kể so với đào tạo trực tiếp trên dữ liệu dành riêng cho nhiệm vụ. Việc đào tạo các Transformers lớn hơn thường mang lại hiệu suất tốt hơn và vẫn là một hướng nghiên cứu tích cực. GPT-3 (Brown và cộng sự, 2020) là mô hình ngôn ngữ Transformer đơn lớn nhất được đào tạo cho đến nay với 175B tham số.

Kỹ thuật nhanh chóng và tinh chỉnh. Mặc dù GPT-3 175B có thể điều chỉnh hành vi của nó chỉ bằng một vài ví dụ huấn luyện bổ sung, nhưng kết quả phụ thuộc rất nhiều vào lời nhắc đầu vào (Brown và cộng sự, 2020). Điều này đòi hỏi một nghệ thuật thực nghiệm trong việc soạn và định dạng lời nhắc để tối đa hóa hiệu suất của mô hình đối với một nhiệm vụ mong muốn, được gọi là kỹ thuật nhanh chóng hoặc hack nhanh chóng.
Tinh chỉnh đào tạo lại một mô hình được đào tạo trước trên các lĩnh vực chung cho một nhiệm vụ cụ thể Devlin et al. (2019b); Radford và cộng sự. (Một). Các biến thể của nó bao gồm việc học chỉ là một tập hợp con của các tham số Devlin et al. (2019b); Collobert & Weston (2008), tuy nhiên những người thực hành thường đào tạo lại tất cả họ để tối đa hóa hiệu quả hoạt động sau này. Tuy nhiên, mức độ lớn của GPT-3 175B khiến việc tinh chỉnh theo cách thông thường trở nên khó khăn do điểm kiểm tra lớn mà nó tạo ra và rào cản phần cứng cao để gia nhập vì nó có cùng dung lượng bộ nhớ như trước khi đào tạo.

Thích ứng hiệu quả tham số. Nhiều người đã đề xuất chèn các lớp bộ điều hợp giữa các lớp hiện có trong mạng thần kinh (Houlsby và cộng sự, 2019; Rebuffi và cộng sự, 2017; Lin và cộng sự, 2020). Phương pháp của chúng tôi sử dụng cấu trúc thắt cổ chai tương tự để áp đặt ràng buộc cấp thấp đối với các bản cập nhật trọng số. Điểm khác biệt chính về chức năng là các trọng số đã học của chúng tôi có thể được hợp nhất với các trọng số chính trong quá trình suy luận, do đó không gây ra bất kỳ độ trễ nào, điều này không xảy ra đối với các lớp bộ điều hợp (Phần 3).
Một phần mở rộng tạm thời của bộ chuyển đổi là COMPACTER (Mahabadi và cộng sự, 2021), về cơ bản tham số hóa các lớp bộ chuyển đổi bằng cách sử dụng các sản phẩm Kronecker với một số sơ đồ chia sẻ trọng lượng được xác định trước. Tương tự, việc kết hợp LoRA với các phương pháp dựa trên sản phẩm tensor khác có thể có khả năng cải thiện hiệu quả tham số của nó, điều mà chúng tôi để lại cho công việc trong tương lai. Gần đây hơn, nhiều đề xuất tối ưu hóa phần nhúng từ đầu vào thay vì tinh chỉnh, giống như sự khái quát hóa liên tục và khác biệt của kỹ thuật kịp thời (Li & Liang, 2021; Lester và cộng sự, 2021; Hambardzumyan và cộng sự, 2020; Liu và cộng sự, 2020; cộng sự, 2021). Chúng tôi đưa ra những so sánh với Li & Liang (2021) trong phần thử nghiệm của mình.
Tuy nhiên, dòng công việc này chỉ có thể mở rộng quy mô bằng cách sử dụng nhiều mã thông báo đặc biệt hơn trong lời nhắc, chiếm độ dài chuỗi có sẵn cho mã thông báo nhiệm vụ khi học cách nhúng vị trí.

Cấu trúc cấp thấp trong Deep Learning. Cấu trúc thứ hạng thấp rất phổ biến trong machine learning. Rất nhiều bài toán học máy có cấu trúc cấp thấp nội tại nhất định (Li và cộng sự, 2016; Cai và cộng sự, 2010; Li và cộng sự, 2018b; Grasedyck và cộng sự, 2013). Hơn nữa, người ta biết rằng đối với nhiều nhiệm vụ học sâu, đặc biệt là những nhiệm vụ có mạng lưới thần kinh được tham số hóa quá mức, mạng lưới thần kinh đã học sẽ có các thuộc tính xếp hạng thấp sau khi đào tạo (Oymak và cộng sự, 2019). Một số công trình trước đây thậm chí còn áp đặt rõ ràng ràng buộc cấp thấp khi đào tạo mạng lưới thần kinh ban đầu (Sainath và cộng sự, 2013; Povey và cộng sự, 2018; Zhang và cộng sự, 2014; Jaderberg và cộng sự, 2014; Zhao và cộng sự, 2014). , 2016; Kho dak và cộng sự, 2021; Denil và cộng sự, 2014); tuy nhiên, theo hiểu biết tốt nhất của chúng tôi, không có tác phẩm nào trong số này xem xét cập nhật cấp thấp lên mô hình cố định để thích ứng với các tác vụ tiếp theo. Trong tài liệu lý thuyết, người ta biết rằng mạng lưới thần kinh hoạt động tốt hơn các phương pháp học tập cổ điển khác, bao gồm cả các hạt nhân tiếp tuyến thần kinh (có độ rộng hữu hạn) tương ứng (Allen-Zhu và cộng sự, 2019; Li & Liang, 2018) khi lớp khái niệm cơ bản có cấu trúc cấp thấp nhất định (Ghorbani và cộng sự, 2020; Allen-Zhu & Li, 2019; Allen-Zhu & Li, 2020a). Một kết quả lý thuyết khác của Allen-Zhu & Li (2020b) cho thấy rằng khả năng thích ứng ở cấp độ thấp có thể hữu ích cho việc huấn luyện đối thủ. Tóm lại, chúng tôi tin rằng bản cập nhật thích ứng cấp thấp được đề xuất của chúng tôi được thúc đẩy tốt bởi tài liệu.

7 HIỂU CÁC CẬP NHẬT CẤP THẤP

Với lợi thế thực nghiệm của LoRA, chúng tôi hy vọng sẽ giải thích rõ hơn các đặc tính của khả năng thích ứng cấp thấp học được từ các nhiệm vụ xuôi dòng. Lưu ý rằng cấu trúc xếp hạng thấp không chỉ hạ thấp rào cản phần cứng để gia nhập, cho phép chúng tôi chạy song song nhiều thử nghiệm mà còn mang lại khả năng diễn giải tốt hơn về cách các trọng số cập nhật tương quan với các trọng số được đào tạo trước. Chúng tôi tập trung nghiên cứu vào GPT-3 175B, nơi chúng tôi đã đạt được mức giảm lớn nhất các thông số có thể huấn luyện (lên tới 10.000×) mà không ảnh hưởng xấu đến hiệu suất tác vụ.

Chúng tôi thực hiện một chuỗi các nghiên cứu thực nghiệm để trả lời các câu hỏi sau: 1) Với một ràng buộc ngân sách tham số, chúng ta nên điều chỉnh tập hợp con ma trận trọng số nào trong Máy biến áp được huấn luyện trước

để tối đa hóa hiệu suất hạ nguồn? 2) Ma trận thích ứng "tối ưu" ΔW có thực sự thiếu thứ hạng không? Nếu vậy, thứ hạng tốt để sử dụng trong thực tế là gì? 3) Mối liên hệ giữa ΔW và W? ΔW có tương quan cao với W không? ΔW lớn như thế nào so với W?

Chúng tôi tin rằng câu trả lời của chúng tôi cho câu hỏi (2) và (3) sẽ làm sáng tỏ các nguyên tắc cơ bản của việc sử dụng các mô hình ngôn ngữ được đào tạo trước cho các tác vụ tiếp theo, đây là một chủ đề quan trọng trong NLP.

7.1 NÊN ÁP DỤNG LORA CHO MÔN TRỌNG LƯỢNG NÀO TRONG MÁY BIẾN ÁP ?

Với ngân sách tham số hạn chế, chúng ta nên điều chỉnh loại trọng số nào với LoRA để có được hiệu suất tốt nhất trong các nhiệm vụ tiếp theo? Như đã đề cập ở phần 4.2, chúng ta chỉ xét trọng số ma trận trong mô-đun tự chú ý. Chúng tôi đặt ngân sách tham số là 18M (khoảng 35 MB nếu được lưu trữ trong FP16) trên GPT-3 175B, tương ứng với $r = 8$ nếu chúng tôi điều chỉnh một loại trọng số chú ý hoặc $r = 4$ nếu chúng ta điều chỉnh hai loại, cho tất cả 96 lớp. Kết quả được trình bày ở Bảng 5.

	Số tham số có thể huấn luyện = 18M			
Loại trọng lượng	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_k, W_v	W_o
Xếp hạng r	8	8	4	4	2
WikiSQL (±0,5%)	70,4	70,0	73,0	73,2	71,4	73,7	73,7
MultiNLI (±0,1%)	91,0	90,8	91,0	91,3	91,3	91,3	91,7

Bảng 5: Độ chính xác xác thực trên WikiSQL và MultiNLI sau khi áp dụng LoRA cho các loại dữ liệu khác nhau trọng số chú ý trong GPT-3, với cùng số lượng tham số có thể huấn luyện. Điều chỉnh cả W_q và W_v cho hiệu suất tổng thể tốt nhất. Chúng tôi tìm thấy độ lệch chuẩn giữa các hạt ngẫu nhiên là nhất quán cho một tập dữ liệu nhất định mà chúng tôi báo cáo trong cột đầu tiên.

Lưu ý rằng việc đặt tất cả tham số vào W_q hoặc W_k sẽ dẫn đến hiệu suất thấp hơn đáng kể, đồng thời điều chỉnh cả W_q và W_v mang lại kết quả tốt nhất. Điều này cho thấy rằng thậm chí xếp hạng bốn nắm bắt đủ thông tin trong ΔW sao cho phù hợp với nhiều ma trận trọng số hơn điều chỉnh một loại trọng số duy nhất với thứ hạng lớn hơn.

7.2 XẾP HẠNG TỐI ƯU cho LORA LÀ GÌ ?

Chúng tôi hướng sự chú ý đến ảnh hưởng của thứ hạng r đến hiệu suất của mô hình. Chúng tôi điều chỉnh {W_q, W_v}, {W_q, W_k, W_v, W_c} và chỉ W_q để so sánh.

	Loại trọng lượng	r = 1	r = 2	r = 4	r = 8	r = 64
WikiSQL(±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
ĐaNLI (± 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

Bảng 6: Độ chính xác xác thực trên WikiSQL và MultiNLI với thứ hạng r khác nhau. Trước sự ngạc nhiên của chúng tôi, một xếp hạng nhỏ như một đủ để điều chỉnh cả W_q và W_v trên các bộ dữ liệu này trong khi chỉ huấn luyện W_q cần một r lớn hơn. Chúng tôi tiến hành thử nghiệm tương tự trên GPT-2 ở Phần H.2.

Bảng 6 cho thấy điều đáng ngạc nhiên là LoRA đã hoạt động có tính cạnh tranh với r rất nhỏ (nhiều hơn vì vậy đối với {W_q, W_v} thay vì chỉ W_q). Điều này cho thấy ma trận cập nhật ΔW có thể có giá trị rất nhỏ "thứ hạng nội tại".[6] Để hỗ trợ thêm cho phát hiện này, chúng tôi kiểm tra sự chồng chéo của các không gian con đã học bởi sự lựa chọn khác nhau của r và bởi các hạt giống ngẫu nhiên khác nhau. Chúng tôi lập luận rằng việc tăng r không bao gồm a không gian con có ý nghĩa hơn, điều này cho thấy rằng ma trận thích ứng cấp thấp là đủ.

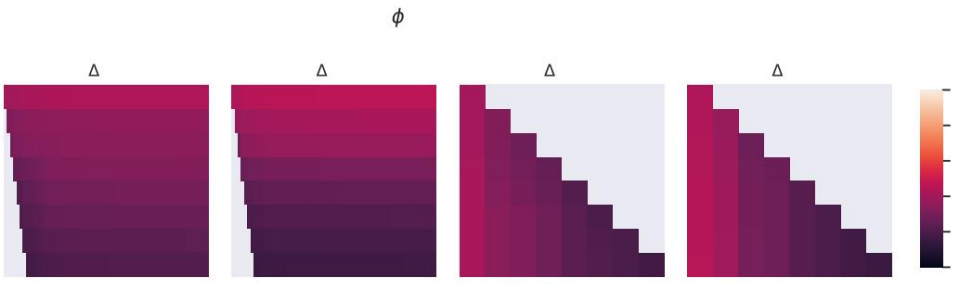
[6] Tuy nhiên, chúng tôi không mong đợi một tỷ lệ r nhỏ sẽ phù hợp với mọi tác vụ hoặc tập dữ liệu. Hãy xem xét suy nghĩ sau đây thử nghiệm: nếu tác vụ xuôi dòng bằng ngôn ngữ khác với ngôn ngữ được sử dụng để đào tạo trước, đào tạo lại toàn bộ mô hình (tương tự LoRA với r = dmodel) chắc chắn có thể hoạt động tốt hơn LoRA với r nhỏ.

Sự tương tự không gian con giữa các r khác nhau. Cho Ar=8 và Ar=64 là các ma trận thích ứng đã học với hạng r = 8 và 64 sử dụng cùng một mô hình được huấn luyện trước, chúng ta thực hiện phân tách giá trị số ít và thu được ma trận đơn nhất số ít UAr=8 và UAr = 64 . [7] Chúng tôi hy vọng vào một sự thay đổi: có bao nhiêu không gian con được bao bọc bởi các vectơ số ít đỉnh i trong UAr=8 (với 1 ≤ i ≤ 8) được chứa trong không gian con được bao bọc bởi các vectơ số ít đỉnh j của UAr=64 (với 1 ≤ j 64)? Chúng tôi đo đại lượng này bằng độ tương tự của không gian con được chuẩn hóa dựa trên khoảng cách Grassmann (Xem Phụ lục G để thảo luận chính thức hơn)

$$\varphi(\text{Ar=8, Ar=64, i, j}) = \frac{\|U_{\text{Ar=8}}^{i} U_{\text{Ar=64}}^{j}\|_F^2}{\text{phút}(i, j)} \quad [0, 1] \tag{4}$$

i trong đó $U_{\text{Ar=8}}$ biểu thị các cột của UAr=8 tương ứng với các vectơ số ít trên cùng. φ(·) có phạm vi [0, 1],

trong đó 1 thể hiện sự chồng lấp hoàn toàn của các không gian con và 0 thể hiện sự phân tách hoàn toàn. Xem Hình 3 để biết φ thay đổi như thế nào khi chúng ta thay đổi i và j. Chúng tôi chỉ xem xét lớp thứ 48 (trong số 96) do hạn chế về không gian, nhưng kết luận cũng đúng với các lớp khác, như được trình bày trong Phần H.1.



Hình 3: Độ tương tự không gian con giữa các vectơ cột của Ar=8 và Ar=64 cho cả Wq và Wv.
Hình thứ ba và thứ tư phóng to hình tam giác phía dưới bên trái trong hai hình đầu tiên. Các hướng trên cùng trong r = 8 được bao gồm trong r = 64 và ngược lại.

Chúng tôi đưa ra một nhận xét quan trọng từ Hình 3.

> Các hướng tương ứng với vectơ số ít trên cùng trùng lặp đáng kể giữa Ar=8 và Ar=64, trong khi các hướng khác thì không. Cụ thể, Wv (tương ứng Wq) của Ar=8 và Wv (tương ứng Wq) của Ar=64 chia sẻ một không gian con của chiều 1 với độ tương tự được chuẩn hóa > 0,5, đưa ra lời giải thích tại sao r = 1 hoạt động khá tốt trong các nhiệm vụ tiếp theo của chúng tôi dành cho GPT-3.

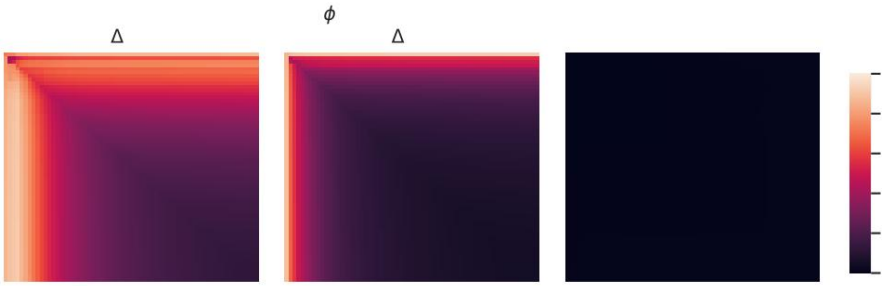
Vì cả Ar=8 và Ar=64 đều được học bằng cách sử dụng cùng một mô hình được đào tạo trước, Hình 3 chỉ ra rằng các hướng vectơ đơn trên cùng của Ar=8 và Ar=64 là hữu ích nhất, trong khi các hướng khác có khả năng chứa hầu hết các nhiễu ngẫu nhiên tích lũy trong quá trình đào tạo. Do đó, ma trận thích ứng thực sự có thể có thứ hạng rất thấp.

Sự tương tự không gian con giữa các hạt ngẫu nhiên khác nhau. Chúng tôi xác nhận thêm điều này bằng cách vẽ biểu đồ tương tự không gian con đã chuẩn hóa giữa hai lần chạy được chọn ngẫu nhiên với r = 64, như trong Hình 4. Wq dường như có "thứ hạng nội tại" cao hơn Wv, vì các hướng giá trị số ít phổ biến hơn được học bởi cả hai đều chạy với giá trị Wq, phù hợp với quan sát thực nghiệm của chúng tôi trong Bảng 6.
Để so sánh, chúng tôi cũng vẽ đồ thị hai ma trận Gaussian ngẫu nhiên, chúng không chia sẻ bất kỳ hướng giá trị số ít chung nào với nhau.

7.3 MA TRẬN THÍ CH ỨNG W SO SÁNH VỚI W NHƯ THẾ NÀO?

Chúng tôi điều tra sâu hơn về mối quan hệ giữa W và W. Cụ thể, W có tương quan cao với W không? (Hoặc về mặt toán học, có phải W hầu hết được chứa trong các hướng kỳ dị trên cùng của W?) Ngoài ra, 7Lưu ý rằng một phân tích tương tự có thể

được thực hiện với B và các ma trận đơn nhất số ít bên trái – chúng tôi sử dụng A cho các thí nghiệm của chúng tôi.



Hình 4: Bên trái và Giữa: Độ tương tự không gian con được chuẩn hóa giữa các vectơ cột của Ar=64 từ hai hạt giống ngẫu nhiên, cho cả Wq và Wv trong lớp thứ 48. Bên phải: bản đồ nhiệt giống nhau giữa các vectơ cột của hai ma trận Gauss ngẫu nhiên. Xem Phần H.1 để biết các lớp khác.

W "lớn" như thế nào so với các hướng tương ứng của nó trong W? Điều này có thể làm sáng tỏ cơ chế cơ bản để điều chỉnh các mô hình ngôn ngữ được đào tạo trước.

Để trả lời những câu hỏi này, chúng ta chiếu W lên không gian con r chiều của W bằng cách tính UWV với U/V là ma trận vectơ đơn trái/phải của W. Sau đó, chúng tôi so sánh chuẩn Frobenius giữa UWV và WF . Để so sánh, chúng tôi cũng tính toán UWV F bằng cách thay thế U, V bằng các vectơ số ít r trên cùng của W hoặc một ma trận ngẫu nhiên.

	r = 4	r = 64	Wq	Wq ngẫu nhiên	Wq	Wq ngẫu nhiên
||U WqV ||F =	0,32	21,67	||Wq||F =	61,95	|| Wq||F =	6,91
	0,02		1,90	37,71	0,33	
			|| Wq||F =	3,57		

Bảng 7: Định mức Frobenius của U WqV trong đó U và V là các hướng vectơ đơn r trên trái/phải của (1) Wq, (2) Wq hoặc (3) ma trận ngẫu nhiên. Ma trận trọng số được lấy từ lớp thứ 48 của GPT-3.

Chúng tôi rút ra một số kết luận từ Bảng 7. Thứ nhất, W có mối tương quan mạnh hơn với W so với ma trận ngẫu nhiên, cho thấy rằng W khuếch đại một số đặc điểm đã có trong W. Thứ hai, thay vì lặp lại các hướng kỳ dị trên cùng của W, W chỉ khuếch đại các hướng không được nhấn mạnh trong W. Thứ ba, hệ số khuếch đại khá lớn: 21,5 ≈ 6,91/0,32 cho r = 4.
Xem Phần H.4 để biết tại sao r = 64 có hệ số khuếch đại nhỏ hơn. Chúng tôi cũng cung cấp hình ảnh trực quan trong Phần H.3 để biết mối tương quan thay đổi như thế nào khi chúng tôi đưa vào nhiều hướng đơn nhất trên cùng từ Wq. Điều này cho thấy ma trận thích ứng cấp thấp có khả năng khuếch đại các tính năng quan trọng cho các nhiệm vụ tiếp theo cụ thể đã được học nhưng không được nhấn mạnh trong mô hình đào tạo trước chung.

8 KẾT LUẬN VÀ CÔNG VIỆC TƯƠNG LAI

Việc tinh chỉnh các mô hình ngôn ngữ khổng lồ cực kỳ tốn kém về mặt phần cứng cần thiết và chi phí lưu trữ/chuyển đổi để lưu trữ các phiên bản độc lập cho các tác vụ khác nhau. Chúng tôi đề xuất LoRA, một chiến lược thích ứng hiệu quả không gây ra độ trễ suy luận cũng như không làm giảm độ dài chuỗi đầu vào trong khi vẫn duy trì chất lượng mô hình cao. Điều quan trọng là nó cho phép chuyển đổi tác vụ nhanh chóng khi được triển khai dưới dạng dịch vụ bằng cách chia sẻ phần lớn các tham số mô hình. Mặc dù chúng tôi tập trung vào các mô hình ngôn ngữ Transformer, nhưng các nguyên tắc đề xuất thường có thể áp dụng cho bất kỳ mạng thần kinh nào có các lớp dày đặc.

Có nhiều hướng đi cho công việc sau này. 1) LoRA có thể được kết hợp với các phương pháp điều chỉnh hiệu quả khác, có khả năng mang lại sự cải tiến trực giao. 2) Cơ chế đằng sau việc tinh chỉnh hoặc LoRA vẫn chưa rõ ràng - làm thế nào các tính năng được học trong quá trình đào tạo trước được chuyển đổi để thực hiện tốt các nhiệm vụ tiếp theo? Chúng tôi tin rằng LoRA khiến việc trả lời câu hỏi này trở nên dễ dàng hơn là hoàn toàn ổn

điều chỉnh. 3) Chúng tôi chủ yếu dựa vào phương pháp phỏng đoán để chọn ma trận trọng số để áp dụng LoRA. Có cách nào nguyên tắc hơn để làm điều đó? 4) Cuối cùng, sự thiếu hụt thứ hạng của W cho thấy rằng W cũng có thể bị thiếu thứ hạng, điều này cũng có thể là nguồn cảm hứng cho các tác phẩm trong tương lai.

NGƯỜI GIỚI THIỆU

Armen Aghajanyan, Luke Zettlemoyer và Sonal Gupta. Chiều kích nội tại giải thích tính hiệu quả của việc tinh chỉnh mô hình ngôn ngữ. arXiv:2012.13255 [cs], tháng 12 năm 2020. URL http://arxiv.org/abs/2012.13255.

Zeyuan Allen-Zhu và Yuanzhi Li. ResNet có thể học được điều gì hiệu quả, vượt xa hạt nhân? Trong NeurIPS, 2019. Phiên bản đầy đủ có tại http://arxiv.org/abs/1905.10337.

Zeyuan Allen-Zhu và Yuanzhi Li. Chỉnh sửa tính năng ngược: Học sâu thực hiện sâu như thế nào học hỏi. bản in trước arXiv arXiv:2001.04413, 2020a.

Zeyuan Allen-Zhu và Yuanzhi Li. Thanh lọc tính năng: Quá trình đào tạo đối thủ hoạt động mạnh mẽ như thế nào học kĩ càng. bản in trước arXiv arXiv:2005.10190, 2020b.

Zeyuan Allen-Zhu, Yuanzhi Li và Zhao Song. Một lý thuyết hội tụ cho việc học sâu thông qua quá trình tham số hóa. Trong ICML, 2019. Phiên bản đầy đủ có tại http://arxiv.org/abs/1811. 03962.

Jimmy Lei Ba, Jamie Ryan Kiros và Geoffrey E. Hinton. Chuẩn hóa lớp, 2016.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhari wal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever và Dario Amodei. Mô hình ngôn ngữ là những người học ít cơ hội. arXiv:2005.14165 [cs], tháng 7 năm 2020. URL http://arxiv.org/abs/2005.14165.

Jian-Feng Cai, Emmanuel J Candes và Zuowei Shen. Một thuật toán ngưỡng giá trị số ít cho hoàn thành ma trận. Tạp chí SIAM về tối ưu hóa, 20(4):1956–1982, 2010.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio và Lucia Specia. Nhiệm vụ Semeval-2017 1: Đánh giá sự tương đồng về ngữ nghĩa của văn bản đa ngôn ngữ và xuyên ngôn ngữ. Kỷ yếu Hội thảo quốc tế về đánh giá ngữ nghĩa lần thứ 11 (SemEval-2017), 2017. doi: 10.18653/v1/s17-2001. URL http://dx.doi.org/10.18653/v1/S17-2001.

Ronan Collobert và Jason Weston. Một kiến trúc thống nhất để xử lý ngôn ngữ tự nhiên: mạng lưới thần kinh sâu với khả năng học tập đa nhiệm. Trong Kỷ yếu của hội nghị quốc tế lần thứ 25 về Học máy, ICML '08, trang 160–167, New York, NY, Hoa Kỳ, tháng 7 năm 2008. Hiệp hội Máy tính. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390177. URL https://doi.org/10.1145/1390156.1390177.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato và Nando de Freitas. Dự đoán các thông số trong học sâu, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee và Kristina Toutanova. Bert: Đào tạo trước máy biến áp hai chiều sâu để hiểu ngôn ngữ, 2019a.

Jacob Devlin, Ming-Wei Chang, Kenton Lee và Kristina Toutanova. BERT: Đào tạo trước Máy biến áp hai chiều sâu để hiểu ngôn ngữ. arXiv:1810.04805 [cs], tháng 5 năm 2019b. URL http://arxiv.org/abs/1810.04805. arXiv: 1810.04805.

William B. Dolan và Chris Brockett. Tự động xây dựng một kho ngữ liệu các câu diễn giải. Trong Kỷ yếu Hội thảo quốc tế lần thứ ba về diễn giải (IWP2005), 2005. URL https:// aclanthology.org/I05-5002.

Claire Gardent, Anastasia Shimorina, Shashi Narayan và Laura Perez-Beltrachini. Thử thách webnlg: Tạo văn bản từ dữ liệu rdf. Trong Kỷ yếu của Hội nghị quốc tế lần thứ 10 về tạo ngôn ngữ tự nhiên, trang 124–133, 2017.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz và Andrea Montanari. Khi nào làm thần kinh mạng tốt hơn các phương pháp kernel? bản in trước arXiv arXiv:2006.13409, 2020.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek và Aleksander Wawer. Samsum corpus: Tập dữ liệu hội thoại có chú thích của con người để tóm tắt trừu tượng. CoRR, abs/1911.12237, 2019. URL http://arxiv.org/abs/1911.12237.

Lars Grasedyck, Daniel Kressner và Christine Tobler. Một cuộc khảo sát tài liệu về tensor cấp thấp các kỹ thuật xấp xỉ. GAMM-Mitteilungen, 36(1):53–78, 2013.

Jihun Ham và Daniel D. Lee. Phân tích phân biệt Grassmann: một quan điểm thống nhất về học tập dựa trên không gian con. Trong ICML, trang 376–383, 2008. URL https://doi.org/10.1145/1390156. 1390204.

Karen Hambardzumyan, Hrant Khachatrian và Jonathan May. WARP: Lập trình lại đối thủ cấp độ từ. arXiv:2101.00121 [cs], tháng 12 năm 2020. URL http://arxiv.org/abs/ 2101.00121. arXiv: 2101.00121.

Bành Thành Hà, Lưu Hiểu Đông, Cao Kiện Phong, và Weizhu Chen. Deberta: Bert tăng cường giải mã với sự chú ý không bị xáo trộn, năm 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan và Sylvain Gelly. Học chuyển giao tham số hiệu quả cho NLP. arXiv:1902.00751 [cs, stat], tháng 6 năm 2019. URL http://arxiv.org/abs/1902. 00751.

Max Jaderberg, Andrea Vedaldi và Andrew Zisserman. Tăng tốc mạng lưới thần kinh tích chập với các bản mở rộng cấp thấp. bản in trước arXiv arXiv:1405.3866, 2014.

Mikhail Khodak, Neil Tenenholtz, Lester Mackey và Nicolo Fusi. Khởi tạo và chính quy hóa của các lớp thần kinh được nhân tố hóa, năm 2021.

Diederik P. Kingma và Jimmy Ba. Adam: Phương pháp tối ưu hóa ngẫu nhiên, 2017.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer và Zhifeng Chen. Gshard: Mở rộng quy mô các mô hình khổng lồ với tính toán có điều kiện và phân đoạn tự động, 2020.

Brian Lester, Rami Al-Rfou và Noah Constant. Sức mạnh của quy mô để điều chỉnh nhanh chóng theo thông số hiệu quả. arXiv:2104.08691 [cs], tháng 4 năm 2021. URL http://arxiv.org/abs/2104.08691. arXiv: 2104.08691.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu và Jason Yosinski. Đo lường chiều kích nội tại của cảnh quan khách quan. arXiv:1804.08838 [cs, stat], tháng 4 năm 2018a. URL http://arxiv.org/abs/1804.08838. arXiv: 1804.08838.

Xiang Lisa Li và Percy Liang. Điều chỉnh tiền tố: Tối ưu hóa lời nhắc liên tục để tạo. arXiv:2101.00190 [cs], tháng 1 năm 2021. URL http://arxiv.org/abs/2101.00190.

Yuanzhi Li và Yingyu Liang. Tìm hiểu các mạng thần kinh được tham số hóa quá mức thông qua việc giảm độ dốc ngẫu nhiên trên dữ liệu có cấu trúc. Trong những tiến bộ trong hệ thống xử lý thông tin thần kinh, 2018.

Yuanzhi Li, Yingyu Liang và Andrej Risteski. Đảm bảo khôi phục xấp xỉ ap xếp hạng thấp có trọng số thông qua việc giảm thiểu xen kẽ. Trong Hội nghị quốc tế về học máy, tr. 2358–2367. PMLR, 2016.

Yuanzhi Li, Tengyu Ma và Hongyang Zhang. Chính quy hóa thuật toán trong mạng lưới thần kinh và cảm biến ma trận được tham số hóa quá mức với các kích hoạt bậc hai. Trong Hội thảo về học tập The ory, trang 2–47. PMLR, 2018b.

Triệu Giang Lâm, Andrea Madotto và Pascale Fung. Khám phá mô hình ngôn ngữ tạo sinh linh hoạt thông qua học chuyển giao tham số hiệu quả. Trong Những phát hiện của Hiệp hội Ngôn ngữ học tính toán: EMNLP 2020, trang 441–459, Trực tuyến, tháng 11 năm 2020. Hiệp hội Ngôn ngữ học tính toán. doi: 10.18653/v1/2020.findings-emnlp.41. URL https://aclanthology. org/2020.findings-emnlp.41.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang và Jie Tang. GPT
Cũng hiểu. arXiv:2103.10385 [cs], tháng 3 năm 2021. URL http://arxiv.org/abs/
2103.10385. arXiv: 2103.10385.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike
Lewis, Luke Zettlemoyer và Veselin Stoyanov. Roberta: Quá trình đào tạo trước bert được tối ưu hóa mạnh mẽ
cách tiếp cận, 2019

Ilya Loshchilov và Frank Hutter. Chính quy hóa phân rã trọng lượng tách rời. bản in trước arXiv
arXiv:1711.05101, 2017.

Ilya Loshchilov và Frank Hutter. Chính sách giảm cân tách rời, 2019.

Rabeeh Karimi Mahabadi, James Henderson và Sebastian Ruder. Compacter: Hiệu quả cấp thấp
các lớp bộ điều hợp siêu phức tạp, 2021.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh,
Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, và những người khác. Dart: Cấu trúc miền mở
bản ghi dữ liệu để tạo văn bản. bản in trước arXiv arXiv:2007.02871, 2020.

Jekaterina Novikova, Ondˇrej Dusek và Verena Rieser. Tập dữ liệu e2e: Những thách thức mới cho mục đích cuối cùng
thế hệ đầu cuối. bản in trước arXiv arXiv:1706.09254, 2017.

Samet Oymak, Zalan Fabian, Mingchen Li và Mahdi Soltanolkotabi. Đảm bảo tổng quát hóa cho mạng lưới thần kinh
thông qua việc khai thác cấu trúc cấp thấp của jacobian. bản in trước arXiv
arXiv:1906.05392, 2019.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Ruckl¨ e, Kyunghyun Cho và Iryna Gurevych. Adaptor- fusion: Thành
phần nhiệm vụ không phá hủy cho việc học chuyển giao, 2021.

Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohammadi và San jeev Khudanpur. Hệ số
ma trận cấp thấp bán trực giao cho mạng lưới thần kinh sâu. TRONG
Interspeech, trang 3743–3747, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans và Ilya Sutskever. Cải thiện khả năng ngôn ngữ thông qua đào
tạo trước sáng tạo. trang 12, a.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei và Ilya Sutskever. Ngôn ngữ
Các mô hình là Người học đa nhiệm không được giám sát. trang 24, b.

Pranav Rajpurkar, Robin Jia và Percy Liang. Biết những gì bạn không biết: Những câu hỏi không thể trả lời
cho đội hình. CoRR, abs/1806.03822, 2018. URL http://arxiv.org/abs/1806.03822.

Sylvestre-Alvise Rebuffi, Hakan Bilen và Andrea Vedaldi. Học nhiều lĩnh vực trực quan với
bộ điều hợp còn lại. arXiv:1705.08045 [cs, stat], tháng 11 năm 2017. URL http://arxiv.org/
abs/1705.08045. arXiv: 1705.08045.

Andreas Ruckl¨ e, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, và
Iryna Gurevych. Adaptordrop: Về hiệu quả của bộ chuyển đổi trong máy biến áp, 2020.

Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy và Bhuvana Ramabhadran. Hệ số hóa ma trận thứ
hạng thấp để đào tạo mạng lưới thần kinh sâu với mục tiêu đầu ra có chiều cao.
Năm 2013, hội nghị quốc tế IEEE về âm học, xử lý giọng nói và tín hiệu, trang 6655–
6659. IEEE, 2013.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper và Bryan
Catanzaro. Megatron-lm: Đào tạo mô hình ngôn ngữ nhiều tỷ tham số sử dụng mô hình par allelism, 2020.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng,
và Christopher Potts. Các mô hình sâu đệ quy về cấu trúc ngữ nghĩa trên một cảm xúc
bờ cây. Trong Kỷ yếu Hội nghị năm 2013 về các phương pháp thực nghiệm trong ngôn ngữ tự nhiên
Đang xử lý, trang 1631–1642, Seattle, Washington, Hoa Kỳ, tháng 10 năm 2013. Hiệp
hội Ngôn ngữ học Tính toán. URL https://aclanthology.org/D13-1170.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser và Illia Polosukhin. Sự chú ý là tất cả những gì bạn cần. Trong Kỷ yếu của Hội nghị quốc tế lần thứ 31 về Hệ thống xử lý thông tin thần kinh, trang 6000–6010, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy và Samuel R. Bowman. Keo: Nền tảng phân tích và điểm chuẩn đa tác vụ để hiểu ngôn ngữ tự nhiên, 2019.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy và Samuel R. Bowman. Superglue: Một tiêu chuẩn chính xác hơn cho các hệ thống hiểu ngôn ngữ có mục đích chung, năm 2020.

Alex Warstadt, Amanpreet Singh và Samuel R Bowman. Phán quyết chấp nhận mạng lưới thần kinh. bản in trước arXiv arXiv:1805.12471, 2018.

Adina Williams, Nikita Nangia và Samuel Bowman. Một kho ngữ liệu thách thức rộng rãi cho việc hiểu câu thông qua suy luận. Trong Kỷ yếu Hội nghị năm 2018 của Chi hội Bắc Mỹ của Hiệp hội Ngôn ngữ học Tính toán: Công nghệ Ngôn ngữ Con người, Tập 1 (Bài báo dài), trang 1112–1122, New Orleans, Louisiana, tháng 6 năm 2018. Hiệp hội Ngôn ngữ học Tính toán. doi: 10.18653/v1/N18-1101. URL https://www.aclweb.org/anthology/N18-1101.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gug ger, Mariama Drame, Quentin Lhoest và Alexander M. Rush. Transformers: Xử lý ngôn ngữ tự nhiên tiên tiến. Trong Kỷ yếu của Hội nghị năm 2020 về các phương pháp thực nghiệm trong xử lý ngôn ngữ tự nhiên: Trình diễn hệ thống, trang 38–45, Trực tuyến, tháng 10 năm 2020. Với tư cách là hiệp hội Ngôn ngữ học tính toán. URL https://www.aclweb.org/anthology/ 2020.emnlp-demos.6.

Greg Yang và Edward J. Hu. Học tính năng trong Mạng thần kinh có chiều rộng vô hạn. arXiv:2011.14522 [cond-mat], tháng 5 năm 2021. URL http://arxiv.org/abs/2011.14522. arXiv: 2011.14522.

Elad Ben Zaken, Shauli Ravfogel và Yoav Goldberg. Bitfit: Tinh chỉnh tham số đơn giản, hiệu quả cho các mô hình ngôn ngữ mặt nạ dựa trên máy biến áp, năm 2021.

Yu Zhang, Ekapol Chuangsuwanich và James Glass. Trích xuất các tính năng tắc nghẽn mạng thần kinh sâu bằng cách sử dụng hệ số ma trận cấp thấp. Năm 2014, hội nghị quốc tế IEEE về âm học, xử lý giọng nói và tín hiệu (ICASSP), trang 185–189. IEEE, 2014.

Yong Zhao, Jinyu Li và Yifan Gong. Thích ứng cấp thấp cộng với đường chéo cho mạng lưới thần kinh sâu. Năm 2016 Hội nghị quốc tế của IEEE về Âm học, Xử lý giọng nói và tín hiệu (ICASSP), trang 5005–5009. IEEE, 2016.

Victor Zhong, Caiming Xiong và Richard Socher. Seq2sql: Tạo các truy vấn có cấu trúc từ ngôn ngữ tự nhiên bằng cách sử dụng phương pháp học tăng cường. CoRR, abs/1709.00103, 2017. URL http://arxiv.org/abs/1709.00103 .

MÔ HÌNH NGÔN NGỮ LỚN VẪN CẦN CẬP NHẬT THÔNG SỐ

Học ít lần hoặc kỹ thuật nhanh chóng sẽ rất thuận lợi khi chúng ta chỉ có một số mẫu đào tạo. Tuy nhiên, trong thực tế, chúng ta thường có đủ khả năng để quản lý vài nghìn ví dụ đào tạo trở lên cho các ứng dụng nhạy cảm với hiệu suất. Như được hiển thị trong Bảng 8, việc tinh chỉnh cải thiện đáng kể hiệu suất của mô hình so với phương pháp học vài lần trên các tập dữ liệu lớn và nhỏ. Chúng tôi lấy kết quả vài lần chụp GPT-3 trên RTE từ bài báo GPT-3 (Brown và cộng sự, 2020). Để khớp với MNLI, chúng tôi sử dụng tổng cộng hai bản trình diễn cho mỗi lớp và sáu ví dụ trong ngữ cảnh.

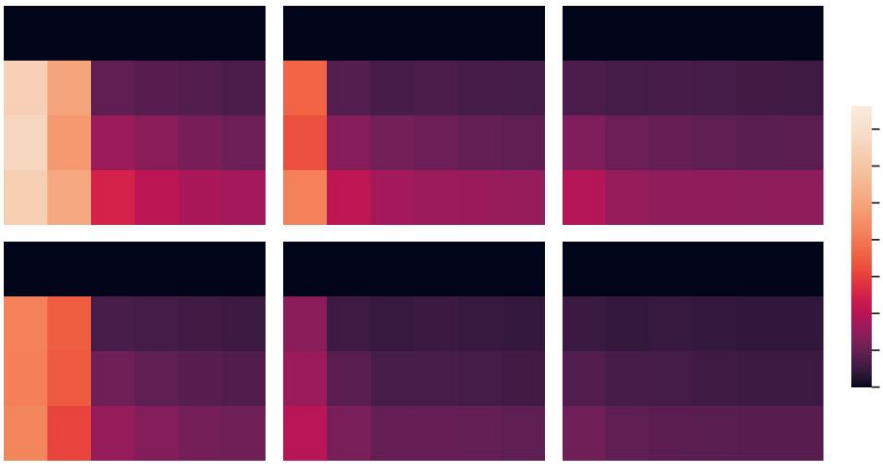
Phương pháp	MNLI-m (Giá trị Acc./%)	RTE (Giá trị Acc./%)
GPT-3 Í t phát bắn	40,6	69,0
Tinh chỉnh GPT-3	89,5	85,4

Bảng 8: Tinh chỉnh hoạt động tốt hơn đáng kể so với phương pháp học vài lần trên GPT-3 (Brown và cộng sự, 2020).

B ĐỘ TRỄ SUY NGHĨ ĐƯỢC GIỚI THIỆU BỞI CÁC LỚP ADAPTER

Các lớp bộ điều hợp là các mô-đun bên ngoài được thêm vào mô hình được đào tạo trước theo cách tuần tự, trong khi đề xuất của chúng tôi, LoRA, có thể được xem là các mô-đun bên ngoài được thêm vào theo cách song song. Do đó, các lớp bộ điều hợp phải được tính toán ngoài mô hình cơ sở, điều này chắc chắn sẽ gây ra độ trễ bổ sung. Trong khi như đã chỉ ra trong Ruckl ¨ e et al. (2020), độ trễ do các lớp bộ chuyển đổi gây ra có thể được giảm thiểu khi kích thước lô mô hình và/hoặc độ dài chuỗi đủ lớn để tận dụng tối đa khả năng song song của phần cứng. Chúng tôi xác nhận quan sát của họ bằng một nghiên cứu về độ trễ tương tự trên môi trường GPT-2 và chỉ ra rằng có những tình huống, đặc biệt là suy luận trực tuyến trong đó kích thước lô nhỏ, trong đó độ trễ tăng thêm có thể đáng kể.

Chúng tôi đo độ trễ của một lần chuyển tiếp duy nhất trên NVIDIA Quadro RTX8000 bằng cách lấy trung bình hơn 100 lần thử. Chúng tôi thay đổi kích thước lô đầu vào, độ dài chuỗi và kích thước nút cổ chai của bộ điều hợp r. Chúng tôi thử nghiệm hai thiết kế bộ chuyển đổi: thiết kế ban đầu của Houlsby et al. (2019), mà chúng tôi gọi là AdaptorH và một biến thể gần đây, hiệu quả hơn của Lin et al. (2020), mà chúng tôi gọi là AdaptorL . Xem Phần 5.1 để biết thêm chi tiết về thiết kế. Chúng tôi biểu thị tỷ lệ phần trăm chậm lại so với đường cơ sở không có bộ chuyển đổi trong Hình 5.



Hình 5: Tỷ lệ phần trăm độ trễ suy luận bị chậm lại so với đường cơ sở không có bộ chuyển đổi (r = 0).
Hàng trên cùng hiển thị kết quả cho AdaptorH và hàng dưới cùng AdaptorL . Kích thước lô và độ dài chuỗi lớn hơn giúp giảm thiểu độ trễ, nhưng tốc độ chậm có thể lên tới hơn 30% trong trường hợp trực tuyến, có độ dài chuỗi ngắn. Chúng tôi điều chỉnh bản đồ màu để hiển thị tốt hơn.

C CHI TIẾT DỮ LIỆU

GLUE Benchmark là một tập hợp đa dạng các nhiệm vụ hiểu ngôn ngữ tự nhiên. Nó bao gồm MNLI (suy luận, Williams và cộng sự (2018)), SST-2 (phân tích tình cảm, Socher và cộng sự (2013)), MRPC (phát hiện diễn giải, Dolan & Brockett (2005)), CoLA (khả năng chấp nhận ngôn ngữ, Warstadt et al.
(2018)), QNLI (suy luận, Rajpurkar và cộng sự (2018)), QQP8 (trả lời câu hỏi), RTE (suy luận),

¨https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

và STS-B (sự tương đồng về văn bản, Cer và cộng sự (2017)). Phạm vi bao phủ rộng khiến điểm chuẩn GLUE trở thành thước đo tiêu chuẩn để đánh giá các mô hình NLU như RoBERTa và DeBERTa. Các bộ dữ liệu riêng lẻ được phát hành theo các giấy phép cho phép khác nhau.

WikiSQL được giới thiệu trong Zhong et al. (2017) và chứa 56, 355/8, 421 ví dụ đào tạo/xác nhận. Nhiệm vụ là tạo các truy vấn SQL từ các câu hỏi ngôn ngữ tự nhiên và sơ đồ bảng.
Chúng tôi mã hóa ngữ cảnh dưới dạng x = {lược đồ bảng, truy vấn} và mục tiêu dưới dạng y = {SQL}. Bộ dữ liệu được phát hành theo Giấy phép 3 khoản BSD.

SAMSum được giới thiệu trong Gliwa et al. (2019) và chứa 14.732/819 ví dụ đào tạo/kiểm tra. Nó bao gồm các cuộc trò chuyện được dàn dựng giữa hai người và các bản tóm tắt trừu tượng tương ứng được viết bởi các nhà ngôn ngữ học. Chúng tôi mã hóa ngữ cảnh dưới dạng các câu nói được nối "\n", theo sau là "\n\n" và nhắm mục tiêu dưới dạng y = {tóm tắt}. Bộ dữ liệu được phát hành theo giấy phép phi thương mại: Creative Commons BY-NC-ND 4.0.

Thử thách E2E NLG được giới thiệu lần đầu tiên tại Novikova et al. (2017) dưới dạng tập dữ liệu để đào tạo các hệ thống tạo ngôn ngữ tự nhiên dựa trên dữ liệu từ đầu đến cuối và thường được sử dụng để đánh giá dữ liệu thành văn bản. Bộ dữ liệu E2E bao gồm khoảng 42.000 khóa đào tạo, 4.600 xác nhận và 4.600 ví dụ kiểm tra từ miền nhà hàng. Mỗi bảng nguồn được sử dụng làm đầu vào có thể có nhiều tham chiếu. Mỗi đầu vào mẫu (x, y) bao gồm một chuỗi các cặp giá trị vị trí, cùng với văn bản tham chiếu ngôn ngữ tự nhiên tương ứng. Bộ dữ liệu được phát hành theo Creative Commons BY-NC-SA 4.0.

DART là tập dữ liệu chuyển dữ liệu thành văn bản trong miền mở được mô tả trong Nan et al. (2020). Đầu vào DART được cấu trúc dưới dạng chuỗi bộ ba ENTITY – RELATION – ENTITY. Với tổng số 82K ví dụ, DART là tác vụ chuyển dữ liệu thành văn bản lớn hơn và phức tạp hơn đáng kể so với E2E. Bộ dữ liệu được phát hành theo giấy phép MIT.

WebNLG là một tập dữ liệu thường được sử dụng khác để đánh giá dữ liệu thành văn bản (Gardent và cộng sự, 2017). Với 22K ví dụ trong tổng số WebNLG bao gồm 14 danh mục riêng biệt, 9 trong số đó được nhìn thấy trong quá trình đào tạo. Vì 5 trong tổng số 14 danh mục không được nhìn thấy trong quá trình đào tạo nhưng được thể hiện trong tập kiểm tra nên việc đánh giá thường được chia thành các danh mục "đã nhìn thấy" (S), danh mục "không nhìn thấy" (U) và "tất cả" (A). Mỗi ví dụ đầu vào được biểu thị bằng một chuỗi bộ ba CHỦ ĐỀ – THUỘC TÍ NH – ĐỐI TƯỢNG. Bộ dữ liệu được phát hành theo Creative Commons BY-NC-SA 4.0.

D SIÊU THÔNG SỐ ĐƯỢC SỬ DỤNG TRONG THÍ NGHIỆM

D.1 ROBERTA

Chúng tôi đào tạo bằng cách sử dụng AdamW với lịch trình giảm tốc độ học tập tuyến tính. Chúng tôi quét tốc độ học tập, số giai đoạn đào tạo và quy mô lô cho LoRA. Theo dõi Liu và cộng sự. (2019), chúng tôi khởi tạo các mô-đun LoRA theo điểm kiểm tra MNLI tốt nhất của mình khi thích ứng với MRPC, RTE và STS-B, thay vì khởi tạo thông thường; mô hình được đào tạo trước vẫn bị đóng băng cho tất cả các nhiệm vụ. Chúng tôi báo cáo trung vị trên 5 hạt giống ngẫu nhiên; kết quả cho mỗi lần chạy được lấy từ kỷ nguyên tốt nhất. Để so sánh công bằng với thiết lập trong Houlsby et al. (2019) và Pfeiffer et al. (2021), chúng tôi giới hạn độ dài chuỗi mô hình ở mức 128 và sử dụng kích thước lô cố định cho tất cả các tác vụ. Điều quan trọng là chúng tôi bắt đầu với mô hình lớn RoBERTa được đào tạo trước khi thích ứng với MRPC, RTE và STS-B, thay vì mô hình đã được điều chỉnh cho MNLI.
Các lần chạy với thiết lập hạn chế này được đánh dấu bằng †. Xem các siêu tham số được sử dụng trong các lần chạy của chúng tôi trong Bảng 9.

D.2 DEBERTA

Chúng tôi lại huấn luyện bằng cách sử dụng AdamW với lịch trình giảm tốc độ học tập tuyến tính. Theo dõi ông và cộng sự. (2021), chúng tôi điều chỉnh tỷ lệ học tập, xác suất bỏ học, các bước khởi động và quy mô lô. Chúng tôi sử dụng cùng độ dài trình tự mô hình mà (He và cộng sự, 2021) đã sử dụng để giữ cho sự so sánh của chúng tôi được công bằng. Theo dõi ông và cộng sự. (2021), chúng tôi khởi tạo các mô-đun LoRA theo điểm kiểm tra MNLI tốt nhất khi thích ứng với MRPC, RTE và STS-B, thay vì khởi tạo thông thường; mô hình được đào tạo trước vẫn bị đóng băng cho tất cả các nhiệm vụ. Chúng tôi báo cáo trung vị trên 5 hạt giống ngẫu nhiên; kết quả cho mỗi lần chạy được lấy từ kỷ nguyên tốt nhất. Xem các siêu tham số được sử dụng trong các lần chạy của chúng tôi trong Bảng 10.

Phương pháp	Tập dữ liệu	MNLI	SST-2	MRPC	Cola	QNLI	QQP	RTE	STS-B
	Trình tối ưu hóa	AdamW							
	Tỷ lệ khởi động	0,06							
	Lịch trình LR	tuyến tính							
Căn cứ RoBERTa LoRA	Kích thước lô	16	16	16	32	32	32	32	16
	# Kỷ nguyên	5	60	30	80	25	80	80	40
	Tốc độ học tập	5E-04	4E-04	4E-04	4E-04	5E-04	5E-04	4E-04	4E-04
	Cấu hình LoRA.	rq = rv = 8							
	LoRA α	8							
	Max Seq. Len.	512							
RoBERTa lớn LoRA	Kích thước lô	4	4	4	4	4	4	8	8
	# Kỷ nguyên	10	20	10	20	20	10	20	30
	Tỷ lệ học tập	3E-04	4E-04	2E-04	4E-04	3E-04	2E-04	3E-04	2E-04
	Cấu hình LoRA.	rq = rv = 8							
	LoRA α	16							
	Max Seq. Len.	128	128	512	128	512	512	512	512
RoBERTa lớn LoRA†	Kích thước lô	4							
	# Kỷ nguyên	10	20	10	20	20	10	20	10
	Tỷ lệ học tập	3E-04	4E-04	3E-04	4E-04	3E-04	2E-04	3E-04	3E-04
	Cấu hình LoRA.	rq = rv = 8							
	LoRA α	16							
	Max Seq. Len.	128							
RoBERTa lớn AdptP (3M)†	Kích thước lô	32							
	# Kỷ nguyên	10	20	20	3E-05	20	10	20	20
	Tỷ lệ học tập	3E-04	3E-05	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04
	Nút thắt r	64							
	Max Seq. Len.	128							
RoBERTa lớn AdptP (0,8M)†	Kích thước lô	32							
	# Kỷ nguyên	5	20	20	3E-04	20	10	20	20
	Tỷ lệ học tập	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04
	Nút thắt r	16							
	Max Seq. Len.	128							
RoBERTa lớn AdptH (6M)†	Kích thước lô	32							
	# Kỷ nguyên	10	20	5	10	3E-04	10	5	20
	Tỷ lệ học tập	3E-04	3E-05	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04
	Nút thắt r	64							
	Max Seq. Len.	128							
RoBERTa lớn AdptH (0,8M)†	Kích thước lô	32							
	# Kỷ nguyên	10	20	5	10	3E-04	10	5	20
	Tốc độ học tập	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04	3E-04
	Nút thắt r	8							
	Max Seq. Len.	128							

Bảng 9: Các siêu tham số chúng tôi sử dụng cho RoBERTa trên điểm chuẩn GLUE.

D.3 GPT-2

Chúng tôi đào tạo tất cả các mô hình GPT-2 bằng cách sử dụng AdamW (Loshchilov & Hutter, 2017) với phương pháp học tuyến tính lịch trình tỷ lệ cho 5 kỷ nguyên. Chúng tôi sử dụng kích thước lô, tốc độ học và kích thước chùm tia tìm kiếm được mô tả trong Lý & Lương (2021). Theo đó, chúng tôi cũng điều chỉnh các siêu tham số trên cho LoRA. Chúng tôi báo cáo có nghĩa là trên 3 hạt ngẫu nhiên; kết quả cho mỗi lần chạy được lấy từ kỷ nguyên tốt nhất. Các siêu tham số được sử dụng cho LoRA trong GPT-2 được liệt kê trong Bảng 11. Đối với những dữ liệu được sử dụng cho các đường cơ sở khác, hãy xem Li & Liang (2021).

D.4 GPT-3

Đối với tất cả các thử nghiệm GPT-3, chúng tôi huấn luyện bằng AdamW (Loshchilov & Hutter, 2017) trong 2 kỷ nguyên với cỡ lô gồm 128 mẫu và hệ số giảm trọng lượng là 0,1. Chúng tôi sử dụng độ dài chuỗi 384 cho

Phương pháp	Tập dữ liệu	MNLI	SST-2	MRPC	Cola	QNLI	QQP	RTE	STS-B
	Trình tối ưu hóa	AdamW							
	Tỷ lệ khởi động	0,1							
	Lịch trình LR	tuyến tính							
DeBERTa XXL LoRA	Kích thước lô 8 8 # Kỷ nguyên 5 11 Tỷ lệ học32 tập 1E-04 6E-05 2E-04 1E-04								4
	1E-04 1E-04 2E-04 2E-04	8 16	30	4 10	6 8			4 11	10
	Giảm trọng lượng 0 0,01 CLS Bỏ học 0,15 0,2 cấu hình LoRA.	0,01 0	0,01 0	0 0,01 0,1 rq = rv = 8				0,01 0,2	0,1 0,2
	LoRA α								
	Max Seq. Len. 256	128	128	64	512	320	320		128

Bảng 10: Các siêu tham số cho DeBERTa XXL trong các nhiệm vụ có trong điểm chuẩn GLUE.

Tập dữ liệu	E2E	WebNLG	DART
	Đào tạo		
Trình tối ưu hóa	AdamW		
Giảm cân	0,01	0,01	0,0
vấn đề bỏ học	0,1	0,1	0,0
Kích thước lô	8		
# kỷ nguyên	5		
Các bước khởi động	500		
Lịch trình tỷ lệ học tập	tuyến tính		
Nhãn mịn	0,1	0,1	0,0
Tỷ lệ học	0,0002		
Thích ứng LoRA α	rq = rv = 4 32		
	Sự suy luận		
Kích thước chùm tia	10		
Độ dài Hình phạt	0,9	0,8	0,8
không lặp lại kích thước ngram	4		

Bảng 11: Các siêu tham số cho GPT-2 LoRA trên E2E, WebNLG và DART.

WikiSQL (Zhong và cộng sự, 2017), 768 cho MNLI (Williams và cộng sự, 2018) và 2048 cho SAMSum (Gliwa và cộng sự, 2019). Chúng tôi điều chỉnh tốc độ học tập cho tất cả các kết hợp phương pháp-tập dữ liệu. Xem Phần D.4 để biết thêm chi tiết về các siêu tham số được sử dụng. Để điều chỉnh việc nhúng tiền tố, chúng ta tìm lp và li tối ưu lần lượt là 256 và 8, tổng cộng 3,2 triệu tham số có thể huấn luyện. Chúng tôi sử dụng lp = 8 và li = 8 cho điều chỉnh lớp tiền tố với 20,2M tham số có thể huấn luyện để đạt được hiệu suất tổng thể tốt nhất. Chúng tôi trình bày hai ngân sách tham số cho LoRA: 4,7M (rq = rv = 1 hoặc rv = 2) và 37,7M (rq = rv = 8 hoặc rq = rk = rv = ro = 2). Chúng tôi báo cáo hiệu suất xác thực tốt nhất từ mỗi lần chạy. Đào tạo siêu tham số được sử dụng trong thử nghiệm GPT-3 của chúng tôi được liệt kê trong Bảng 12.

E KẾT HỢP LORA VỚI ĐIỀU CHỈNH TIỀN TỐ

LoRA có thể được kết hợp một cách tự nhiên với các phương pháp tiếp cận dựa trên tiền tố hiện có. Trong phần này, chúng tôi đánh giá hai sự kết hợp của LoRA và các biến thể điều chỉnh tiền tố trên WikiSQL và MNLI.

LoRA+PrefixEmbed (LoRA+PE) kết hợp LoRA với điều chỉnh nhúng tiền tố, nơi chúng tôi chèn lp + li mã thông báo đặc biệt có phần nhúng được coi là tham số có thể huấn luyện. Để biết thêm về điều chỉnh nhúng tiền tố, hãy xem Phần 5.1.

LoRA+PrefixLayer (LoRA+PL) kết hợp LoRA với điều chỉnh lớp tiền tố. Chúng ta cũng chèn lp + li mã thông báo đặc biệt; tuy nhiên, thay vì để các biểu diễn ẩn của các mã thông báo này phát triển tự nhiên

Siêu tham số	Tinh chỉnh	Bộ điều hợp	BitFit	PreLayer	PreLayerH	LoRA
Trình tối ưu hóa	AdamW					
Kích thước lô	128					
# kỷ nguyên	2					
Mã thông báo khởi động	250.000					
Lịch trình LR	tuyến tính					
Tỷ lệ học	5,00E-06	5,00E-04	1,00E-04	1,6E-03	1,00E-04	2,00E-04

Bảng 12: Các siêu tham số huấn luyện được sử dụng cho các phương pháp điều chỉnh GPT-3 khác nhau. Chúng tôi sử dụng cùng một siêu tham số cho tất cả các tập dữ liệu sau khi điều chỉnh tốc độ học.

phục hồi, chúng tôi thay thế chúng sau mỗi khối Transformer bằng một vectơ bất khả tri đầu vào. Như vậy, cả các phần nhúng và kích hoạt khối Transformer tiếp theo được coi là các tham số có thể huấn luyện được. Vì thêm về điều chỉnh lớp tiền tố, xem Phần 5.1.

Trong Bảng 15, chúng tôi trình bày kết quả đánh giá của LoRA+PE và LoRA+PL trên WikiSQL và MultiNLI. Trước hết, LoRA+PE hoạt động tốt hơn đáng kể cả LoRA và điều chỉnh nhúng tiền tố trên WikiSQL, chỉ ra rằng LoRA có phần trực giao với việc điều chỉnh việc nhúng tiền tố. TRÊN MultiNLI, sự kết hợp của LoRA+PE không hoạt động tốt hơn LoRA, có thể vì LoRA bản thân nó đã đạt được hiệu suất tương đương với mức cơ bản của con người. Thứ hai, chúng tôi nhận thấy LoRA+PL hoạt động kém hơn LoRA một chút ngay cả với nhiều thông số có thể huấn luyện hơn. Chúng tôi ghi nhận điều này vì thực tế là việc điều chỉnh lớp tiền tố rất nhạy cảm với việc lựa chọn tốc độ học và do đó làm cho việc tối ưu hóa trọng số LoRA trở nên khó khăn hơn trong LoRA+PL.

F THÍ NGHIỆM THỰC NGHIỆM BỔ SUNG

F.1 THÍ NGHIỆM BỔ SUNG VỀ GPT-2

Chúng tôi cũng lặp lại thử nghiệm của mình trên DART (Nan và cộng sự, 2020) và WebNLG (Gardent và cộng sự, 2017) theo thiết lập của Li & Liang (2021). Kết quả được thể hiện ở Bảng 13. Tương tự với kết quả của chúng tôi trong Thử thách E2E NLG, được báo cáo trong Phần 5, LoRA hoạt động tốt hơn hoặc ít nhất là ngang bằng với các phương pháp tiếp cận dựa trên tiền tố với cùng số lượng tham số có thể huấn luyện được.

Phương pháp	# Có thể đào tạo Thông số	DART BLEU	MET	TER
GPT-2 vừa				
Tinh chỉnh 354M	0,37M	46,2	0,39	0,46
Bộ chuyển đổiL		42,4	0,36	0,48
Bộ chuyển đổiL	11 triệu	45,2	0,38	0,46
FTTop2	24M	41,0	0,34	0,56
lớp trước	0,35M	46,4	0,38	0,46
LoRA	0,35M	47,1±,2	0,39	0,46
GPT-2 Lớn				
Tinh chỉnh 774M		47,0	0,39	0,46
Bộ chuyển đổiL	0,88M	45,7±.1	0,38	0,46
Bộ chuyển đổiL	23M	47,1	0,39	0,45
lớp trước	0,77M	±.1	0,38	0,45
LoRA	0,77M	46,7 47,5±.1	0,39	0,45

Bảng 13: GPT-2 với các phương pháp thích ứng khác nhau trên DART. Phương sai của MET và TER là nhỏ hơn 0,01 cho tất cả các phương pháp thích ứng.

Phương pháp	WebNLG								
	BLEU			MET			TER		
	bạn	S	AUSAUSA						
			GPT-2 vừa						
Tinh chỉnh (354M)	27,7	64,2	46,5	,30	, 45	,38	, 76	,33	,53
AdaptorL (0,37M)	45,1	54,5	50,2	,36	,39	,38	,46	,40	,43
AdaptorL (11M)	48,3	60,4	, 38	,43	9,41	,45	,35	,39	
FTTop2 (24M)	18,9	53,6	0,23	0,38	0,31	0,99	0,49	0,72	
Tiền tố (0,35M)	45,6	62,9	, 38	,54	, 41	,49	,35	,40	
LoRA (0,35M)	46,7±,4	62,1±, 2	55,3±,2	, 38	,44	,41	,46	,33	,39
			GPT-2 Lớn						
Tinh chỉnh (774M)	43.1	65,3	55,5	,38	, 46	,42	,53	,33	,42
AdaptorL (0.88M)	49.8±.0	61.1±.0	56.0±.0	.38	.43	.41	.44	.35	.39
Bộ chuyển đổiL (23M)	49,2±.1	64,7±.2	57,7±.1	.39	.46	.43	.46	.33	.39
Tiền tố (0,77M)	63,4	.39	47,7	.45	.42	.48	.34	.40	56,3
LoRA (0,77M)	48,4±.3	64,0±.3	57,0±.1	.39	.45	.42	.45	.32	.38

Bảng 14: GPT-2 với các phương thức thích ứng khác nhau trên WebNLG. Phương sai của MET và TER nhỏ hơn 0,01 cho tất cả các thử nghiệm chúng tôi đã thực hiện. “U” biểu thị danh mục chưa thấy, “S” biểu thị đã thấy các danh mục và “A” biểu thị tất cả các danh mục trong bộ thử nghiệm của WebNLG.

F.2 THÍ NGHIỆM BỔ SUNG TRÊN GPT-3

Chúng tôi trình bày các lần chạy bổ sung trên GPT-3 với các phương pháp thích ứng khác nhau trong Bảng 15. Trọng tâm là xác định sự cân bằng giữa hiệu suất và số lượng tham số có thể huấn luyện được.

F.3 CHẾ ĐỘ DỮ LIỆU THẤP

Để đánh giá hiệu suất của các phương pháp thích ứng khác nhau trong chế độ dữ liệu thấp. chúng tôi ngẫu nhiên các ví dụ huấn luyện mẫu 100, 1k và 10k từ tập huấn luyện đầy đủ của MNLI để tạo thành dữ liệu thấp nhiệm vụ MNLI-n. Trong Bảng 16, chúng tôi trình bày hiệu quả của các phương pháp thích ứng khác nhau trên MNLI n. Thật ngạc nhiên, PrefixEmbed và PrefixLayer hoạt động rất kém trên tập dữ liệu MNLI-100, với PrefixEmbed chỉ hoạt động tốt hơn một chút so với cơ hội ngẫu nhiên (37,6% so với 33,3%). Lớp tiền tố hoạt động tốt hơn PrefixEmbed nhưng vẫn kém hơn đáng kể so với Fine-Tune hoặc LoRA trên MNLI 100. Khoảng cách giữa các phương pháp tiếp cận dựa trên tiền tố và LoRA/Tinh chỉnh trở nên nhỏ hơn khi chúng tôi tăng số lượng ví dụ đào tạo, điều này có thể gợi ý rằng tiền tố- cách tiếp cận dựa trên không thích hợp cho các tác vụ dữ liệu thấp trong GPT-3. LoRA đạt được hiệu suất tốt hơn so với việc tinh chỉnh trên cả hai MNLI-100 và MNLI-Full, và các kết quả tương đương trên MNLI-1k và MNLI-10K khi xem xét (± 0,3) phương sai do hạt ngẫu nhiên.

Các siêu tham số huấn luyện của các phương pháp thích ứng khác nhau trên MNLI-n được báo cáo trong Bảng 17. Chúng tôi sử dụng tốc độ học tập nhỏ hơn cho PrefixLayer trên bộ MNLI-100, giống như tổn thất huấn luyện. không giảm khi tốc độ học lớn hơn.

G ĐO LƯỜNG SỰ TƯƠNG TỰ GIỮA CÁC KHÔNG GIAN CON

Trong bài báo này chúng tôi sử dụng thước đo $\varphi(A, B, i, j) = \psi(U^{A}_{MỘT,}$ Uj B) = $\frac{\text{bạn}^{\sim}_{MỘT}\ UB\ \overset{2}{F}}{\text{phút}\{i,j\}}$ để đo không gian con độ tương tự giữa hai ma trận trực chuẩn cột $U^{\sim}_{MỘT}$ $R^{d\times i}$ và bạn$^{j}_{B}$ $R^{d\times j}$, thu được bởi lấy các cột của ma trận số ít bên trái của A và B. Chúng tôi chỉ ra rằng sự giống nhau này chỉ đơn giản là đảo ngược của Phép chiếu tiêu chuẩn đo khoảng cách giữa các không gian con Ham & Lee (2008).

Phương pháp	Siêu tham số	# Tham số có thể huấn luyện	WikiSQL	MNLI-m
Tinh chỉnh	-	175B	73,8	89,5
Tiền tốNhúng	lp = 32, li = 8	0,4 triệu	55,9	84,9
	lp = 64, li = 8	0,9 triệu	58,7	88,1
	lp = 128, li = 8	1,7 triệu	60,6	88,0
	lp = 256, li = 8	3,2 triệu	63,1	88,6
	lp = 512, li = 8	6,4 triệu	55,9	85,8
Lớp tiền tố	lp = 2, li = 2	5,1 triệu	68,5	89,2
	lp = 8, li = 0	10,1 triệu	69,8	88,2
	lp = 8, lý = 8	20,2 triệu	70,1	89,5
	lp = 32, lý = 4	44,1 triệu	66,4	89,6
	lp = 64, lý = 0	76,1 triệu	64,9	87,9
Bộ chuyển đổi H	r = 1	7,1 triệu	71,9	89,8
	r = 4	21,2 triệu	73,2	91,0
	r = 8	40,1 triệu	73,2	91,5
	r = 16	77,9 triệu	73,2	91,5
	r = 64	304,4 triệu	72,6	91,5
LoRA	rv = 2	4,7 triệu	73,4	91,7
	rq = rv = 1 rq	4,7 triệu	73,4	91,3
	= rv = 2 rq =	9,4 triệu	73,3	91,4
	rk = rv = ro = 1 rq = rv =	9,4 triệu	74,1	91,2
	4 rq = rk = rv	18,8 triệu	73,7	91,3
	= ro = 2 rq = rv = 8 rq =	18,8 triệu	73,7	91,7
	rk = rv = ro	37,7 triệu	73,8	91,6
	= 4 rq = rv = 64 rq = rk =	37,7 triệu	74,0	91,7
	rv = ro = 64 rq	301,9 triệu	73,6	91,4
	= rv = 8, lp = 8, li = 4 rq	603,8 triệu	73,9	91,4
LoRA+PE	= rv = 32, lp = 8, li = 4 rq =	37,8 triệu	75,0	91,4
	rv = 64, lp = 8, li = 4 rq = rv	151,1 triệu	75,9	91,1
	= 8, lp = 8, li = 4	302,1 triệu	76,2	91,3
LoRA+PL		52,8 triệu	72,9	90,2

Bảng 15: Phân tích siêu tham số của các phương pháp điều chỉnh khác nhau trên WikiSQL và MNLI. Cả hai điều chỉnh nhúng tiền tố (PrefixEmbed) và điều chỉnh lớp tiền tố (PrefixLayer) hoạt động kém hơn khi chúng tôi tăng số lượng tham số có thể huấn luyện, trong khi hiệu suất của LoRA ổn định. Hiệu suất là được đo bằng độ chính xác xác nhận.

Phương pháp	MNLI(m)-100	MNLI(m)-1k	MNLI(m)-10k	MNLI(m)-392K
GPT-3 (Tinh chỉnh)	60,2	85,8	88,9	89,5
GPT-3 (Tiền tố được nhúng)	37,6	75,2	79,5	88,6
GPT-3 (Lớp tiền tố)	48,3	82,5	85,9	89,6
GPT-3 (LoRA)	63,8	85,6	89,2	91,7

Bảng 16: Độ chính xác xác thực của các phương pháp khác nhau trên các tập hợp con của MNLI sử dụng GPT-3 175B. MNLI n mô tả một tập con có n ví dụ huấn luyện. Chúng tôi đánh giá với bộ xác nhận đầy đủ. LoRA thực hiện cho thấy hiệu quả lấy mẫu thuận lợi so với các phương pháp khác, bao gồm cả tinh chỉnh.

Để cụ thể, hãy để các giá trị số ít của $U^{\top}_{\text{MỘT}}$ bạn$^{j}_{B}$ là $\sigma_1, \sigma_2, \cdots, \sigma_p$ trong đó $p = \min\{i, j\}$. Chúng tôi biết rằng Chỉ số Phép chiếu Ham & Lee (2008) được định nghĩa là:

$$d(U^{\top}_{\text{MỘT}}, U_j B) = p \frac{\sum_{i=1} \sigma^2_i}{} \quad [0, \sqrt{p}]$$

Siêu tham số	Thích ứng	MNLI-100 MNLI-1k MNLI-10K MNLI-392K
Trình tối ưu hóa	-	AdamW
Mã thông báo khởi động	-	250.000
Lịch trình LR	-	tuyến tính
Kích thước lô	-	20 20 100 128
# kỷ nguyên	-	40 40 4 2
Tỷ lệ học	Tinh chỉnh	5,00E-6
	Tiền tốNhúng	2,00E-04 2,00E-04 4,00E-04 5,00E-05 5,00E-04
	Lớp tiền tố	5,00E-05 5,00E-05 2,00E-4 1,00E-04
	LoRA	
Thích ứng-	Tiền tốNhúng lp	16 32 64 256
Cụ thể	Tiền tốNhúng li	d_{model}
	Tiền tốTune	lp = lý = 8
	LoRA	rq = rv = 8

Bảng 17: Các siêu tham số được sử dụng cho các phương pháp điều chỉnh GPT-3 khác nhau trên MNLI(m)-n.

trong đó sự giống nhau của chúng tôi được định nghĩa là:

$$\varphi(A, B, i, j) = \psi(U_{\text{MỘT},} \text{Uj B}) = \frac{\sum_{\text{tôi=1}}^{P} \sigma^2}{P} = \frac{1}{P} \quad 1 \quad d(U_{\text{MỘT},} \text{Uj B})^2$$

Sự giống nhau này thỏa mãn rằng nếu $U_{\text{MỘT}}^{\text{tôi}}$ và bạn $_{B}^{j}$ có chung một cột thì φ(A, B, i, j) = 1. Nếu chúng hoàn toàn trực giao, khi đó φ(A, B, i, j) = 0. Ngược lại, φ(A, B, i, j) (0, 1).

H THÍ NGHIỆM BỔ SUNG VỀ MA TRẬN CẤP THẤP

Chúng tôi trình bày các kết quả bổ sung từ cuộc điều tra của chúng tôi về ma trận cập nhật thứ hạng thấp.

H.1 TƯƠNG QUAN GIỮA CÁC MÔ- ĐUN LORA

Xem Hình 6 và Hình 7 để biết các kết quả được trình bày trong Hình 3 và Hình 4 khái quát hóa như thế nào đối với các các lớp.

H.2 TÁC DỤNG CỦA r TRÊN GPT-2

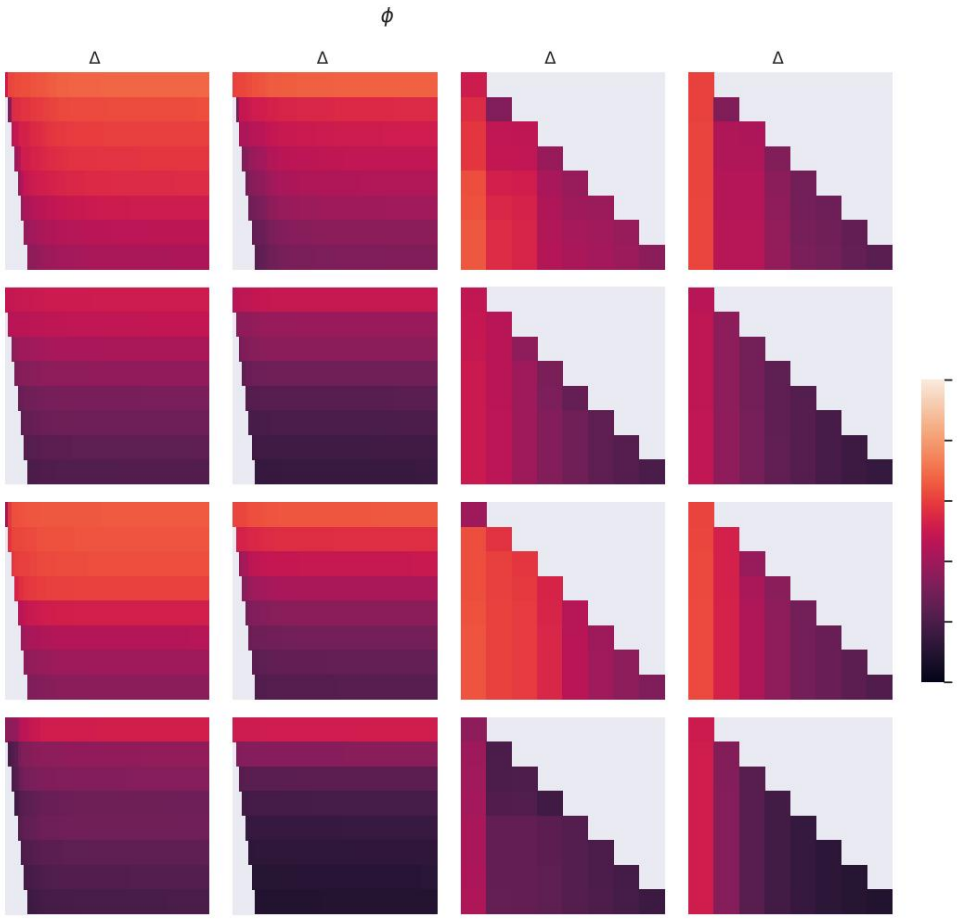
Chúng tôi lặp lại thử nghiệm của mình về tác động của r (Phần 7.2) trong GPT-2. Sử dụng Thử thách E2E NLG tập dữ liệu làm ví dụ, chúng tôi báo cáo số liệu mất xác thực và kiểm tra đạt được bằng các lựa chọn khác nhau của r sau khi luyện tập 26.000 bước. Chúng tôi trình bày kết quả của mình trong Bảng 18. Thứ hạng tối ưu cho GPT-2 Phương tiện nằm trong khoảng từ 4 đến 16 tùy thuộc vào số liệu được sử dụng, tương tự như số liệu của GPT-3 175B. Lưu ý rằng mối quan hệ giữa kích thước mô hình và thứ hạng tối ưu cho sự thích ứng vẫn còn là một mối quan hệ mở. câu hỏi.

H.3 TƯƠNG QUAN GIỮA W VÀ W

Xem Hình 8 để biết độ tương tự của không gian con được chuẩn hóa giữa W và W với r thay đổi.

Một lần nữa lưu ý rằng W không chứa các hướng kỳ dị trên cùng của W, vì sự giống nhau giữa 4 hướng hàng đầu trong W và 10% hướng hàng đầu trong W hầu như không vượt quá 0,2. Điều này đưa ra bằng chứng rằng W chứa các hướng “dành riêng cho nhiệm vụ” mà mặt khác không được nhấn mạnh trong W.

Một câu hỏi thú vị tiếp theo cần trả lời là chúng ta cần “mạnh” đến mức nào để khuếch đại những nhiệm vụ cụ thể đó. hướng dẫn, để việc điều chỉnh mô hình hoạt động tốt?



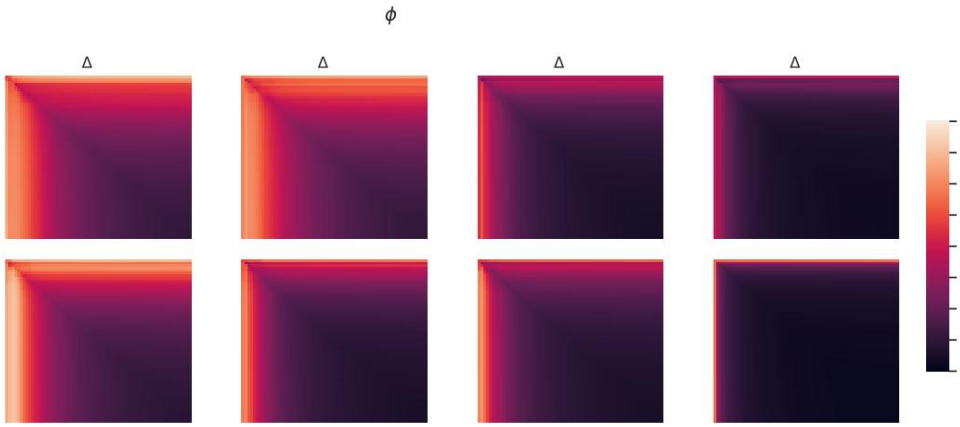
Hình 6: Độ tương tự không gian con được chuẩn hóa giữa các vectơ cột Ar=8 và Ar=64 cho cả Wq và Wv từ các lớp 1, 32, 64 và 96 trong Máy biến áp 96 lớp.

H.4 HỆ SỐ KHUẾCH ĐẠI

Người ta có thể coi hệ số khuếch đại đặc trưng một cách tự nhiên vì tỷ $\frac{\|WF\|}{\|UW\ VF\|}$, nơi U và V lệ này là các ma trận số ít bên trái và bên phải của phân tách SVD của W. (Nhớ lại UU WVV cung cấp "hình chiếu" của W lên không gian con kéo dài bởi W.)

Theo trực giác, khi W chủ yếu chứa các chỉ dẫn dành riêng cho nhiệm vụ, đại lượng này đo lường mức độ khuếch đại của chúng bởi W. Như được hiển thị trong Phần 7.3, với r = 4, hệ số khuếch đại này lớn bằng 20. Nói cách khác, có (nói chung) bốn hướng đặc trưng trong mỗi lớp (trong toàn bộ không gian đặc trưng từ mô hình được huấn luyện trước W), cần được khuếch đại với hệ số rất lớn là 20, để đạt được độ chính xác được báo cáo của chúng tôi cho nhiệm vụ cụ thể ở hạ nguồn. Và người ta có thể mong đợi một tập hợp các hướng đặc trưng rất khác nhau sẽ được khuếch đại cho từng tác vụ tiếp theo khác nhau.

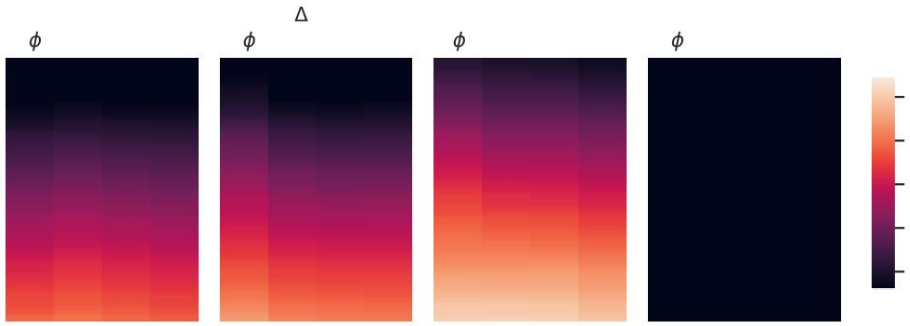
Tuy nhiên, người ta có thể nhận thấy rằng với r = 64, hệ số khuếch đại này chỉ vào khoảng 2, nghĩa là hầu hết các hướng đã học trong W với r = 64 đều không được khuếch đại nhiều. Điều này không có gì đáng ngạc nhiên, và trên thực tế (một lần nữa) đưa ra bằng chứng rằng thứ hạng nội tại cần thiết để thể hiện "các hướng cụ thể cho nhiệm vụ" (do đó để điều chỉnh mô hình) là thấp. Ngược lại, những hướng đó trong phiên bản hạng 4 của W (tương ứng với r = 4) được khuếch đại với hệ số lớn hơn nhiều là 20.



Hình 7: Độ tương tự không gian con được chuẩn hóa giữa các vectơ cột của Ar=64 từ hai vectơ ngẫu nhiên chạy hạt giống, cho cả Wq và Wv từ lớp 1, 32, 64 và 96 trong máy biến áp Trans 96 lớp.

Xếp hạng r	val loss	BLEU	NIST	METEOR	ROUGE L	CIDEr
1	1,23	68,72	8,7215	0,4565	69,17	0,7052	2.4329
2	1,21	8,7413	0,4590	70,38	8,8439	0,7052	2.4639
4	1,18	8 1,17	0,4689	69,57	8,7457	0,4636	0,7186	2.5349
16	1,16	32	69,61	8,7483	0,4629	69,33	0,7196	2.5196
1,16	64	1,16	8,7736	0,4642	6	9,24	8,7174	0,7177	2.4985
128	1,16	256	0,4651	68,73	8,6718	0,4628	0,7105	2.5255
1,16	512	1,16	68,92	8,6982	0,4629	68,78	0,7180	2.5070
1024	1,17	8,6857	0,4637	69,37	8,7495	0,7127	2.5030
		0,4659				0,7128	2.5012
						0,7128	2.5025
						0,7149	2.5090

Bảng 18: Mất xác thực và số liệu của bộ kiểm tra về Thử thách E2E NLG mà LoRA đạt được với xếp hạng r khác nhau bằng GPT-2 Medium. Không giống như trên GPT-3 nơi r = 1 đủ cho nhiều tác vụ, ở đây đỉnh cao hiệu suất ở r = 16 đối với mất xác thực và r = 4 đối với BLEU, gợi ý GPT-2 Phương tiện có thứ hạng nội tại tương tự về khả năng thích ứng so với GPT-3 175B. Lưu ý rằng một số của chúng tôi siêu tham số được điều chỉnh trên r = 4, khớp với số lượng tham số của đường cơ sở khác và do đó có thể không tối ưu cho các lựa chọn khác của r.



Hình 8: Độ tương tự không gian con được chuẩn hóa giữa các hướng kỳ dị của Wq và các hướng của Wq với r thay đổi và đường cơ sở ngẫu nhiên. Wq khuếch đại các hướng quan trọng nhưng không được nhấn mạnh trong W. W với r lớn hơn có xu hướng thu được nhiều hướng đã được nhấn mạnh trong W.