疎なパラメータを用いて大規模言語モデルを効率よく Fine-Tune する手法の提案

原田慎太朗　山崎智弘　吉田尚水
株式会社東芝 研究開発センター
{shintaro2.harada,tomohiro2.yamasaki,takami.yoshida}@toshiba.co.jp

概要

本稿では、疎なパラメータを用いて大規模言語モデルの部分的更新を行なうことで効率よく Fine-Tune する手法を提案する。大規模なコーパスで学習された事前学習済みの言語モデルを特定のタスクやドメインに対して Fine-Tune するには、事前学習済みモデルのパラメータを固定し、少量の追加パラメータのみを学習する手法 (Parameter-Efficient Fine-Tuning, PEFT) が主流である。しかし、通常の PEFT は学習時の計算リソースを抑える一方でパラメータをすべて更新するため、知識保持および編集が重要なタスクやドメインの性能が劣化したりモデル解釈性が失われたりする問題がある。そこで本稿では、疎なパラメータを用いることで計算リソースを抑えつつ、新たにモデル解釈性も備えた新しい PEFT を提案する。事前学習済みモデルとして RoBERTa と Llama3-8B に適用することで、言語理解と算術推論ベンチマークで性能が向上すること、およびモデル解釈が可能であることを示す。

1　はじめに

深層学習を用いた言語モデルは大規模化が進んでいる。そのため、特定のタスクやドメインに対してモデルのパラメータをすべて更新する場合、大規模な計算リソースが必要である。例えば、更新するパラメータ数を ϕ としたとき、モデルの状態を 16bit、Optimizer の状態を 32bit で保持しようとすると、16ϕ byte の計算リソースが必要である[1)]。8B サイズの Llama3[1] の場合、$16 \times 8B = 128$ Gbyte 必要である。

この問題に対しては、更新するパラメータ数 ϕ を減らして効率よく Fine-Tune することができる PEFT(Parameter-Efficient Fine-Tuning) を用いることが主流である。PEFT は大きく Soft-Prompt 型 [2, 3, 4, 5] と Adapter 型 [6, 7, 8, 9, 10, 11, 12, 13] に分けられる。本稿で注目する Adapter 型では、ベースモデルのパラメータを固定し、少数の追加パラメータのみを更新することでパラメータ数 ϕ を減らしている。その中には、行列分解 [6, 7, 11] や特殊な操作 [12, 13] によりパラメータ数 ϕ を減らすものがある。これらは追加したパラメータを学習後に元のパラメータに統合できるという特徴があり、追加パラメータの計算による推論速度の低下を避けることができる。

一方これらの PEFT は、対象とするモジュールのパラメータをすべて更新してしまう。先行研究 [14] でも言及されているとおり、特定のタスクおよびドメインに寄与するパラメータは限定的であり、パラメータをすべて更新してしまうと、保持したい事前知識が失われたり、特定の事前知識の編集が困難になったりする。すなわち、(Q1) 事前学習による優れた性能と (Q2) モデル解釈性が失われると考えられる。これらの課題に対して、先行研究 [14] では重要

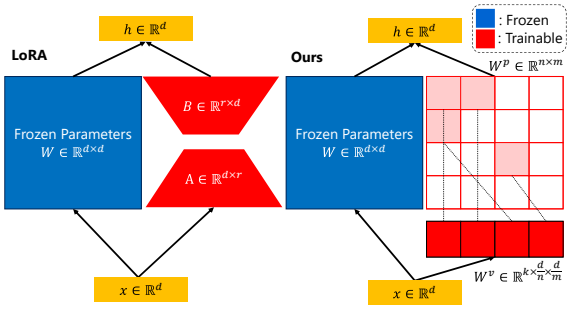

図 1　提案手法の概要図。左に示す先行研究ではすべてのパラメータを上書きしてしまうが、右に示す提案手法では特定のパラメータ（ピンクの部分）のみを更新できる。右の図は、パラメータを行で $n = 4$ 個、列で $m = 4$ 個の計 16 個のブロックに切り分け、$W^p \in \mathbb{R}^{n \times m}$ の中で重要な $k = 4$ 個のブロック位置を特定し、そこにブロックパラメータ $W^v \in \mathbb{R}^{k \times d/n \times d/m}$ を挿入して、疎なパラメータを構築した状態を表している。

1)　https://huggingface.co/blog/Isayoften/optimization-rush

度に基づいて部分的にパラメータを更新する PEFT を提案しているが、強い制約を導入するために複雑な操作を必要としている。また更新されたパラメータの解釈に関しては無視されている。

本稿では、図 1 に示すような疎のパラメータを用いることで、より単純かつモデル解釈可能な PEFT を提案する。実験の結果、(A1) モデルサイズに関わらず部分的なパラメータの更新でも性能が向上すること、(A2) 学習ドメインに対して重要なパラメータを示せることを確認した。

2 PEFT

本節では、大規模モデルを少ない計算リソースで Fine-Tune するための PEFT について説明する。

前節で述べたように、更新するモジュールのパラメータ数を ϕ とする場合、すべてのパラメータを更新するためには計算リソースが 16ϕ byte 必要となる。そこで、Adapter 型の PEFT では

$$h = W'x = (W^o + \Delta W)x \tag{1}$$

のように元のモジュールのパラメータ W^o を固定し、少量の追加パラメータ ΔW のみを更新することで、パラメータ数 ϕ を減らしている。ここで、x は入力ベクトル、h は x に対する潜在ベクトル、$W^o \in \mathbb{R}^{d\times d}$ は対象モジュールのパラメータ、$\Delta W \in \mathbb{R}^{d\times d}$ は追加した学習可能パラメータである。ΔW の作り方にはいくつか種類があり、行列分解に基づいて $\Delta W = BA$ とするもの [6] や、特殊な操作に基づいて $\Delta W = \mathrm{Op}(A)$ とするもの [12] などがある。いずれも推論時は、$W' = W^o + \Delta W$ とパラメータをあらかじめ統合しておくことで、追加の行列計算による推論速度の低下を避けることができる。

しかし ΔW は密なパラメータであるため、元のパラメータ W^o が持つ事前知識を上書きしてしまう問題がある。例えば、ベースの言語モデルを指示調整した後にドメイン学習を行うと、事前および指示調整で獲得した知識を上書きし続けてしまい、知識忘却につながる。また、図 1 のように、ΔW が密なパラメータということは、どのパラメータがどのタスクおよびドメインに対する性能に寄与するのか解釈が難しいという問題がある。

3 疎なパラメータを用いた PEFT

前述したとおり、密なパラメータを追加して更新する PEFT では、保持したい事前知識が失われたり特定の箇所だけの編集が困難な場合がある。また、どのパラメータがどのドメインやタスクに対する性能向上に寄与するのか解釈が難しい。そこで本節では、図 1 に示すとおり、

$$\Delta W = \mathrm{Sparse}(\mathbf{p}, \mathbf{v}W^v) \tag{2}$$

のように疎なパラメータを用いることで、必要な計算リソースを抑えつつ特定のパラメータのみを選択して更新できる新しい PEFT を提案する。後述するように、更新されるパラメータを特定できるのでモデル解釈性も得ることができる。ここで、$\mathrm{Sparse}(\cdot)$ は疎な行列 ΔW を構築する関数であり、引数として、位置を表す $\mathbf{p}$ と値を表す $\mathbf{v}W^v$ がある。疎な行列を構築するために必要な $\mathbf{p}$、$\mathbf{v}$、W は、次の順で導出する。

まず、重要なパラメータの大まかな位置を特定するために、元のパラメータ $W^o \in \mathbf{R}^{d\times d}$ を行と列でそれぞれ n 個と m 個のブロックに分ける。次に、$n \times m$ 個のブロックに対する重要度合いを表す更新可能なパラメータ $W^p \in \mathbf{R}^{n\times m}$ を用意し、上位 k 個のブロックの位置とスコアを

$$\mathbf{p}, \mathbf{v} = \mathrm{TopK}(W^p) \tag{3}$$

のように取得する。ここで、$\mathrm{TopK}(\cdot)$ は入力 W^p において値が上位 k 番目までの位置 $\mathbf{p}$ と値 $\mathbf{v}$ を返す関数である。

得られた k 個の位置とスコアに対し、k 個の更新可能なブロックパラメータ $W^v \in \mathbb{R}^{k\times d/n\times d/m}$ を用意して、

$$\mathbf{v}W^v = [v_1 W^v_1, v_2 W^v_2, ..., v_k W^v_k] \tag{4}$$

のように k 個のスコアでそれぞれのブロックパラメータ W^v_i を重みづける。ここで、v_i は正の実数値、$W^v_i \in \mathbb{R}^{d/n\times d/m}$ は選択された k 個の位置に対する学習可能なブロックパラメータ、$\mathbf{v}$ は各ブロックに対するスコアである。また、ハイパーパラメータ α を用いてスケールする [6]。

このように疎なパラメータを用いると、保持すべきパラメータは W^p と W^v だけであり、パラメータ数は $\phi = n\times m + k\times d/n\times d/m)$ と表現できる。特に、$nm = d$ とすると $\phi = (1+k)d$ となり、元のパラメータ数は d の 2 乗であったものが、先行研究の PEFT と同じく d の 1 乗に抑えることができる。さらに、元のパラメータをブロックに分けて、一部のブロックのみ更新できることから、事前知識の上書きを防いで性能の向上することができる。また、選択した

表 1　言語理解ベンチマーク (GLEU) における定量評価
†：先行研究 [6] から引用

	学習可能パラメータ数	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg
Full†	125.0M	**87.6**	**94.8**	**90.2**	**63.6**	**92.8**	**91.9**	78.7	**91.2**	**86.4**
LoRA	0.9M	87.4	94.4	88.7	60.8	**92.8**	90.8	**79.1**	90.8	85.7
Ours	0.9M	83.2	94.6	90.0	55.2	92.7	86.5	**79.1**	89.9	83.9

表 2　算術推論ベンチマークにおける定量評価

	学習可能パラメータ数	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Avg
LoRA	56.6M	**68.1**	**50.7**	83.3	35.8	87.2	71.2	66.1
Ours	56.3M	67.5	50.1	**84.5**	**36.6**	**87.9**	**72.8**	**66.6**

ブロックのスコアを観察することでモデル解釈性が高まることが期待される。先行研究 [6, 15] でも述べられている通り、学習の初めは ΔW が影響を及ぼさないようにすることが重要であるため、W^p は正規分布、W^v はゼロで初期化する。

4　実験

本節では、まず、モデルサイズに関わらず、疎なパラメータ更新でも性能を向上できることを検証する。その後に、学習データに応じて選択されるパラメータにパターンについて観察する。

4.1　設定

疎なパラメータ更新でも性能を向上できることを検証するために、幅広く利用されている密なパラメータ更新の LoRA (Low-Rank Adaptation)[6] と比較する。ベースのモデルとしては、小さいものとして RoBERTa-125M[2] [16]、大きいものとして Llama3-8B[3] [1] を用いた。ベースラインを先行研究 [6] に設定して、正しく比較できるよう学習パラメータ ϕ は同じになるように設定した。RoBERTa-125M は言語理解ベンチマークの GLEU[17] でそれぞれ学習および評価する。学習設定は先行研究 [6] に従い、独自のハイパーパラメータは $n = 24$、$m = 32$、$k = 15$、$\alpha = 1.0$ に設定した。Llama3-8B は 8bit で量子化して Math10K で学習し、算術推論のベンチマークの MultiArith、GSM8K、AddSub、AQuA、SingleEq、SVAMP で評価する。Math10K には GSM8K と AQuA の学習データのみ含まれる。学習設定は先行研究 [18] に従い、独自のハイパーパラメータは、$n = 64$、$m = 64$、$k = 50$、$\alpha = 2.0$ に設定した。また、ブロックパラメータの重要度を示す $W^p \in \mathbf{R}^{n \times m}$ に対しては、学習データのドメインやタスクに応じてパターンが現れるか人手で確認する。すべての実験で NVIDIA H100 を用い、Llama3-8B には FlashAttention2 [19] を用いた。

2）https://huggingface.co/FacebookAI/roberta-base
3）https://huggingface.co/meta-llama/Meta-Llama-3-8B

4.2　定量評価

はじめに表 1 に言語理解ベンチマークでの定量評価の結果を示す。表 1 からわかるように、密なパラメータを更新を行う先行研究と比較して、疎なパラメータを更新でも SST-2、MRPC、QNLI、RTE のデータセットでより同等以上の性能を示した。しかし、密なパラメータを更新と比較すると MNLI、CoLA、QQP での性能向上は低く、平均が 2 ポイントほど下回った。

続いて表 2 に算術推論ベンチマークでの定量評価の結果を示す。表 2 からわかるように、密なパラメータを更新を行う先行研究と比較して、疎なパラメータを更新でも MultiArith と GSM8K 以外のデータセットで同等以上の性能を示した。平均は 0.5 ポイントほど上回った。

4.3　モデル解釈について

提案手法では、学習した後に獲得された $W^p \in \mathbf{R}^{n \times m}$ に基づいて、学習データに対して選択されたパラメータの位置とスコアを観察できる。図 2 と図 3 にそれぞれ MRPC と CoLA で学習した RoBERTa-125M[16] の self-attention 層の query パラメータに対するブロックスコアを可視化したものを示す。明るいほどスコアが高く、暗いほどスコアが低いことを示している。

図 2 からわかるように、MRPC では最初の 0 層目に明るいブロックは少なく、学習される $k = 15$ 個のブロックで足りている。最後の 11 層目にも明るいブロックは少なく、学習される $k = 15$ 個のブロック

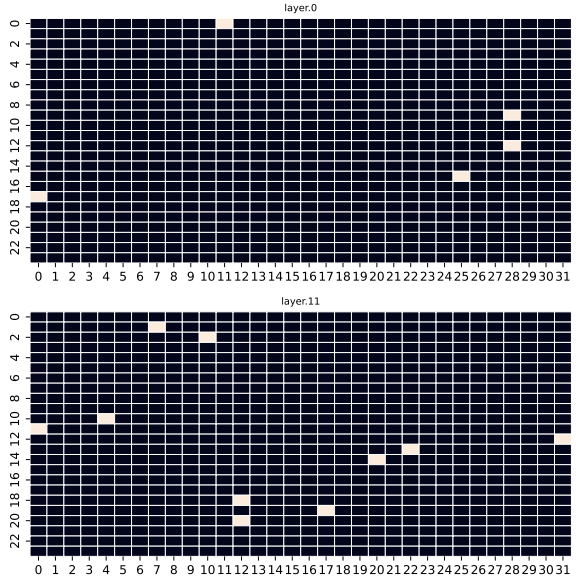

図 2 MRPC で学習した後に獲得された $W^p \in \mathbb{R}^{n\times m}$ のブロックスコアを可視化したもの。スコアが高い明るいブロックは 0 層目にも 11 層目にもほとんど出現しない。

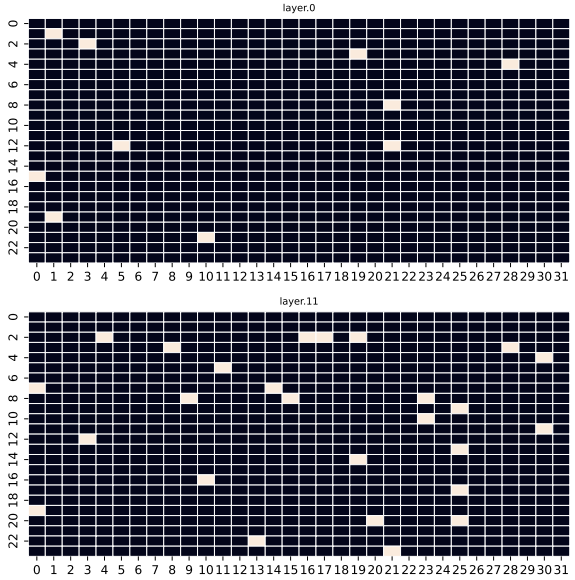

図 3 CoLA で学習された後に獲得された $W^p \in \mathbb{R}^{n\times m}$ のブロックスコアを可視化したもの。スコアが高い明るいブロックは 0 層目にはほとんど出現しないが、11 層目には $k = 15$ 個を超えて出現する。

で足りていることが分かる。同じく図 3 からわかるように、CoLA でも最初の 0 層目に明るいブロックは少なく、学習される $k = 15$ 個のブロックで足りている。しかし最後の 11 層目に明るいブロックが多くなり、学習される $k = 15$ 個のブロックでは足りていないことが分かる。この結果は、LoRA において、入力に近い層にはランクを小さく、出力に近い層はランクを大きくした方が性能が良くなることを示した先行研究 [7] と似ている。これが、CoLA における性能向上の頭打ちの原因だと考えられる。また、W^p の初期値にも強く影響を受けることが考えられるが、この分析については今後の課題とする。

以上のことは、RoBERTa-125M においては学習ドメインやタスクに応じて重要なパラメータが存在していることを示唆しており、学習した後に獲得された $W^p \in \mathbf{R}^{n\times m}$ がモデル解釈に利用できることを示している。また CoLA 以外でも最初の 0 層目に明るいブロックは少なく、最後の 11 層目に明るいブロックが多くなる傾向があることから、先行研究 [7] 同様、入力に近い層は k を小さくし、出力に近い層は k を大きくするとさらなる性能向上が見込まれることが分かる。加えて疎であるため学習ドメインやタスクごとに更新した ΔW は重複しづらく、統合の際に知識の上書きがされにくいことも示唆している。これらは、先行研究 [6] には無いユニークな特徴である。Llama3-8B でのブロックスコアの分析については今後の課題とする。

5 おわりに

本稿では、先行研究 [6] における密なパラメータの更新では元のモデルが持つ事前知識の保持および編集が複雑になり、性能向上とモデル解釈性が失われることを指摘した。そこで、疎なパラメータを用いた新たな PEFT を提案した。先行研究と同様に計算リソースを抑えつつ、新たにモデルパラメータの解釈性を持たせられる特徴がある。結果として、部分的なパラメータ更新でも性能が向上することを示すとともに、どのパラメータが性能に寄与するのか解釈できることも示した。

今後の展望としては、先行研究における密なパラメータの更新では知識忘却につながるが、提案手法ではそれが抑えられることを検証することが考えられる。例えば、指示調整した後にドメイン調整すると、事前学習および指示調整で獲得した知識を上書きしてしまい性能が劣化する可能性があるが、提案手法では解消できる可能性がある。また、複数のパラメータ ΔW で複数のドメインやタスクを Fine-Tune することを想定すると、互いのパラメータ ΔW が干渉しないようにうまく制御できるようにすることも考えられる。

参考文献

[1] Llama-Team. The llama 3 herd of models.

[2] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, **Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics.

[3] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too, 2023.

[4] Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogério Schmidt Feris, Huan Sun, and Yoon Kim. Multitask prompt tuning enables parameter-efficient transfer learning. **ArXiv**, Vol. abs/2303.02861, , 2023.

[5] Tsachi Blau, Moshe Kimhi, Yonatan Belinkov, Alexander Bronstein, and Chaim Baskin. Context-aware prompt tuning: Advancing in-context learning with adversarial methods. **arXiv preprint arXiv:2410.17222**, 2024.

[6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In **International Conference on Learning Representations**, 2022.

[7] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In **The Eleventh International Conference on Learning Representations**, 2023.

[8] Eric L. Buehler and Markus J. Buehler. X-lora: Mixture of low-rank adapter experts, a flexible framework for large language models with applications in protein mechanics and molecular design, 2024.

[9] Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning, 2023.

[10] Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In **ICLR**, 2024.

[11] Wenxuan Tan, Nicholas Roberts, Tzu-Heng Huang, Jitian Zhao, John Cooper, Samuel Guo, Chengyu Duan, and Frederic Sala. More fine-tuning with 10x fewer parameters, 2024.

[12] Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. In **Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024**, 2024.

[13] Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. Mora: High-rank updating for parameter-efficient fine-tuning. **ArXiv**, Vol. abs/2405.12130, , 2024.

[14] Haoyu Wang, Tianci Liu, Ruirui Li, Monica Xiao Cheng, Tuo Zhao, and Jing Gao. RoseLoRA: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, **Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing**, pp. 996–1008, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[15] Baohao Liao, Shaomu Tan, and Christof Monz. Make your pre-trained model reversible: From parameter to memory efficient fine-tuning, 2023.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. **ArXiv**, Vol. abs/1907.11692, , 2019.

[17] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

[18] Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. **arXiv preprint arXiv:2304.01933**, 2023.

[19] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In **International Conference on Learning Representations (ICLR)**, 2024.