LLM 事前学習の効率化と性質改善：埋め込み層および出力層のパラメータ固定による再利用

李 宰成 [1]　矢野 一樹 [1]　高橋 良允 [1]　柴田 圭悟 [1]　池田 航 [1]　鈴木 潤 [1,2,3]
[1] 東北大学　[2] 理化学研究所　[3] 国立情報学研究所
lee.jaesung@dc.tohoku.ac.jp

概要

LLM の効率的な段階的事前学習法を提案する．提案法では，事前に適切な埋め込み層と出力層のパラメータを獲得し，それを初期値として再利用した上で固定して事前学習をする．この手順を取ることで，標準的な事前学習より計算量およびメモリ要求量を削減可能であることを示す．また，事前に獲得した埋め込み層と出力層の単語ベクトルとしての性能が標準的な事前学習で得られる表現よりも優れており，それが LLM の最終的な性能向上に寄与する可能性があることを示す．

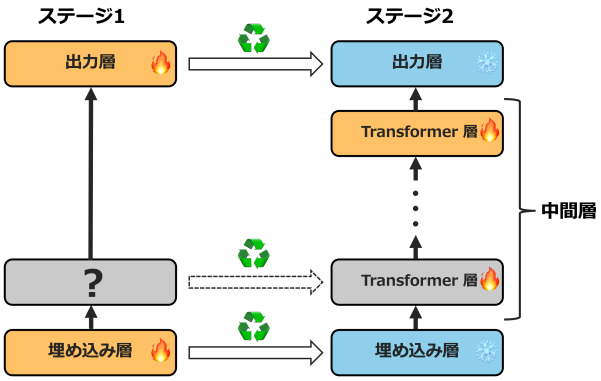

図 1　「埋め込み層と出力層」の再利用と固定を伴う二段階学習．ステージ 1 で構築した埋め込み層と出力層をステージ 2 で再利用かつ固定し学習する．

1　はじめに

大規模言語モデル（LLM）の性能は訓練データ量とパラメータ数に大きく依存する [1]．その結果，LLM の事前学習には大量の計算資源を必要とし，容易に LLM 構築に関連する研究に取り組むことができない状況にある．この状況を打破するため，より効率的な事前学習の方法が求められる．

効率的な事前学習を求め，計算量とメモリ使用量を削減する方法がそれぞれ提案されている．計算量を削減するために，段階的な学習を用い，事前学習の前段階でモデルの適切な初期値を取得し，それを再利用する方法が提案されている [2, 3, 4]．学習時のメモリ使用量を削減する方法として，一部のパラメータを固定する代わりに，少数のパラメータを代理として訓練する方法が提案されている [5, 6, 7]．

本研究では，埋め込み層および出力層のパラメータ固定を伴う再利用による二段階学習を提案する．これは，概念的には前述した事前学習の前段階としてモデルの適切な初期値を取得する方法の亜種と言える．提案法の特色は，LLM で用いられる Transofrmer の中間層は同一の Transformer 層の繰り返しで構成されているのに対し，埋め込み層と出力層は中間層の構造と比較すると特別な構造と見做すことができる．このモデル構造の観測から，Transformer のモデル構造において，特別な構造となっている埋め込み層と出力層を事前に適切に学習して用意できれば，同一の構造の繰り返しである中間層はより容易に学習できるようになり，結果的に学習効率が上がるのではないかという仮説に基づいて考案された方法となる．

提案法を標準的な事前学習法と比較した結果，計算量とメモリ使用量を削減可能であると示された．さらに，構築した埋め込み層と出力層の埋め込み表現は単語ベクトルとして性能が標準的な事前学習で得られる表現よりも優れているという結果が得られた．この分析結果より，埋め込み層および出力層の単語ベクトルとしての性能が，LLM の性能に寄与する可能性があるという新たな知見が得られた．

2　関連研究

適切な初期値を利用した事前学習　事前学習の計算量削減のため，適切な初期値から事前学習を開始する研究が注目されつつある．パッチ学習 [2] と呼ばれる方法では，トークンをまとめた「パッチ」と

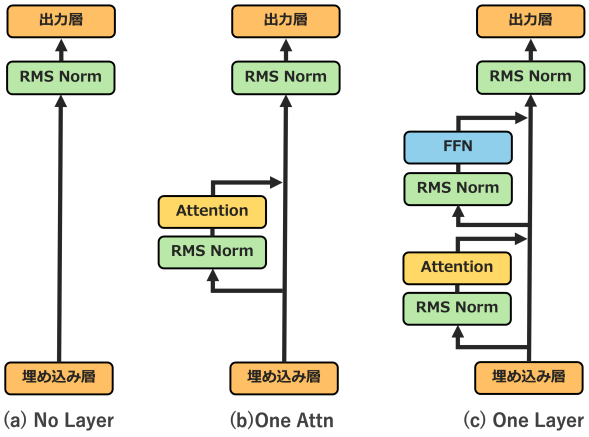

図 2　埋め込み層と出力層の構築方法.（a）層が存在しないモデル（b）一層の Attention 層モデル（c）一層の Transformer 層モデル

いう単位を入力とし，パッチ列の自己回帰モデルによってモデルを学習する．その後，獲得したモデルパラメータを，標準的な事前学習の初期値として再利用することで，全体の計算量を半分程度に削減しつつ，同等の性能を達成した．また別のアプローチとして，モデル成長 [3, 4] と呼ばれる段階的学習では，あらかじめ小さなモデルを学習し，その重みをより大きなモデルの初期値として再利用することで，大きなモデルをスクラッチから学習するよりも学習の計算量を大幅に削減できることを示した．

3　再利用と固定による二段階学習

提案する二段階学習の手順は以下の通りである．

手順 1　何らかのモデルを学習し，適切な埋め込み層と出力層を構築

手順 2　手順 1 で構築した埋め込み層と出力層をより大きなモデルの初期値として代入

手順 3　再利用した埋め込み層と出力層を固定した状態で中間層のパラメータのみを学習

図 1 におけるステージ 1 は手順 1 に，ステージ 2 は手順 3 にそれぞれ対応する．また手順 2 はステージ 1 と 2 の間の矢印部分に相当する．

3.1　Pre-LN Transformer

LLM の多くは，学習安定化のために Pre-LN Transformer [8] を採用している．隠れ状態の次元数を d，Transformer 層への入力（埋め込み層の出力）を $x_{\text{input}} \in \mathbb{R}^d$，出力層への入力を Logit とすると，Pre-LN Transformer の Logit は以下の式で得られる：

$$\text{Logit} = \text{RMS}\Big(\text{Total\_Activation}(x_{\text{input}}) + x_{\text{input}}\Big)W_{\text{out}}, \quad (1)$$

ここで，Total_Activation は，中間層全域にわたる Attention 層と FFN 層の出力の合計であり，それぞれの l 層の出力を $a_{\text{A}}^{(l)}$，$a_{\text{F}}^{(l)}$ とすると，Total_Activation は以下のとおりである：

$$\text{Total\_Activation}(x_{\text{input}}) = \sum_{l=1}^{L}\Big(a_{\text{A}}^{(l)} + a_{\text{F}}^{(l)}\Big) \quad (2)$$

$$a_{\text{F}}^{(l)} = \text{FFN}\big(\text{RMS}(a_{\text{A}}^{(l)} + x^{(l-1)})\big) \quad (3)$$

$$a_{\text{A}}^{(l)} = \text{Attention}\big(\text{RMS}(x^{(l-1)})\big) \quad (4)$$

ここで，$x^{(l-1)}$ は l 層目の Transformer 層への入力であり，$x^{(0)} = x_{\text{input}}$ である．また，$W_{\text{out}} \in \mathbb{R}^{d\times V}$ は出力層と呼ばれる（次元数 × 語彙数）の重み行列，RMS は層正規化として用いられる RMSNorm [9] である．以上のように，Pre-LN Transformer では埋め込み層の出力 x_{input} に対して Attention 層と FFN 層の出力が逐次加算される形式となっている．

3.2　埋め込み層と出力層の構築方法

ステージ 1 において，適切な埋め込み層と出力層の初期値を獲得するために，Pre-LN Transfomer を基にした 3 つのモデルを検討した（図 2）．

（a）No Layer　中間層が存在しないため，x_{input} に加算する Total_Activation が 0 となる．式 (1) の Logit の計算は以下の式に変更になる：

$$\text{Logit} = \text{RMS}(x_{\text{input}})W_{\text{out}}, \quad (5)$$

式からわかるように，x_{input} に変化が加えられずに，RMS 層を通り出力層まで到達する．つまり，No Layer はバイグラム言語モデルとみなせる．

（b）One Attn　一層の Attention 層のみで構成する．Logit の計算は以下のようになる：

$$\text{Logit} = \text{RMS}(a_{A}^{(1)} + x_{\text{input}})W_{\text{out}}, \quad (6)$$

Attention 層により，通常の LLM と同様に前方文脈を混ぜ合わせる機能を持つ．よって，No Layer より多くの文脈情報を考慮し次トークン予測を行える．

（c）One Layer　一層の Transformer 層で構成する．Logit の計算が次のようになる：

$$\text{Logit} = \text{RMS}(a_{A}^{(1)} + a_{F}^{(1)} + x_{\text{input}})W_{\text{out}}, \quad (7)$$

Attention 層と FFN 層の出力により x_{input} が変換される．One Attn に FFN 層が追加されることで，モデルの表現能力が高まる．

ステージ 2 において，上記の 3 通りの方法で構築した埋め込み層と出力層を，より深い層を持つモデルの静的埋め込み表現として再利用し，固定するこ

表 1　ステージ 2 で構築したモデルの下流タスクの評価結果. PPL：Perplexity の平均を表し値が低いほど望ましい，Acc.：Accuracy の平均を表し値が高いほど望ましい.

Size	Method	LR	PPL ↓	Acc. ↑(%)
570M	ベースライン	8e-4	20.70	46.27
	Base Reuse	1e-3	20.26	46.69
	No Layer	2e-3	22.61	45.31
	One Attn (None)	4e-3	21.14	46.57
	One Attn (Attn)	3e-3	20.24	47.19
	One Layer (None)	2e-3	20.35	47.20
	One Layer (Attn)	3e-3	**20.05**	**47.56**
	One Layer (Attn&FFN)	4e-3	20.28	46.89
1B	ベースライン	6e-4	17.65	49.40
	Base Reuse	9e-4	17.77	49.33
	One Attn (Attn)	9e-4	17.26	50.32
	One Layer (None)	9e-4	16.90	50.27
	One Layer (Attn)	9e-4	**16.74**	**50.53**

表 2　ベースラインに対する計算量の削減率. ステップ数の左側の値は訓練を打ち止めた際のステップ数．右側はベースラインのステップ数．削減率はステージ 1 も含めた合計計算コストの削減率.

Size	Method	ステップ数	削減率（%）
570M	One Layer (Attn)	16.0k / 20.0k	11.52
1B	One Layer (Attn)	13.0k / 20.0k	28.33

とで，性能を維持しながら計算量とメモリ使用量の削減を実現する.

4　実験

提案する埋め込み層と出力層の再利用と固定に基づく二段階事前学習が，性能を維持しつつ，計算量と最大メモリ要求量を削減可能か実験により検証する.

4.1　データセットおよびモデル設定

事前学習の訓練・開発データとして FineWeb-Edu [10] を選択した．ステージ 1 およびステージ 2 の両方で，同量かつ同一の学習データを用いた.

モデル設定は，LLaMA [11] を踏襲した．ステージ 1 とステージ 2 では中間層の構成方法と学習率のみ異なり，それ以外の訓練設定は全て同一である.[1] なお，ベースラインモデルとして，ステージ 2 のモデルサイズと同一のモデルを標準の事前学習（ランダムな初期値からのフルパラメータ学習）により学習し，比較対象とした.

1）　モデル設定や学習設定の詳細は付録 A に記す.

4.2　下流タスクにおける評価結果

表 1 に二段階学習によりステージ 2 を終えたモデルの下流タスクの評価結果を示した．下流タスク評価ベンチマークデータセットとして，FineWeb-Edu の開発データ 10M トークンと，WikiText [12] の開発データ 0.3M トークンのデータを使用し，パープレキシティ（Perplexity）を算出した．さらに，LAMBADA [13]，Winogrande [14]，PIQA [15]，HellaSwag[16]，ARC [17]，OpenBookQA [18] を用いて，正解率 Acc. を算出した.[2]

表 1 の Method 欄にある「( )」内は，ステージ 1 で得られた中間層の重みを，ステージ 2 の第 1 層目の Transformer 層で初期値として再利用するサブレイヤを示している．つまり，「(Attn)」と表記される場合，ステージ 1 で学習した Attention 層をステージ 2 で再利用することを意味する．また，「(None)」は，ステージ 1 で学習した中間層は再利用しないことを意味する．さらに，各手法について学習率を調整したうえで、下流タスクの平均スコア（Avg.）が最も高い学習率設定のモデルのみを結果として記載する.[3]

なお、Base Reuse は，ステージ 2 において学習済みベースラインの埋め込み層と出力層のみを再利用し，それらを固定する方法で，比較対象として用いている.

モデルサイズが 570M の場合の評価結果では，One Layer (Attn) が最も高いスコアを示し、次いで One Layer (None)，One Attn (Attn) の順となり，いずれも Base Reuse を上回った．一方、No Layer は唯一ベースラインを下回る結果となった．モデルサイズを 1b に拡大した条件下では，モデルサイズ 570M において Avg. スコアが高かった上位 3 つの手法を検証したところ，570M と同様の傾向が確認された.

さらに，提案法では，学習率をベースラインよりも高く設定することで，性能が向上する傾向が見られる.

4.3　計算量とメモリ使用量の削減

4.2 節で示した結果は，計算量の観点を考慮せずに，ステージ 2 を規定のデータ量学習した際のタスク性能を示している．そこで次に，提案法の各段階

2）　ゼロショットでモデルを評価した．また，Brown et al.[19] らに従い Accuracy（acc-norm スコア）を報告する.
3）　学習率の探索をしたのは，各設定によって最適な学習率が変化するためである

表 3　モデルサイズ 570M の単語ベクトル評価結果. Word Similarity タスク（WordSim）をスピアマンの順位相関係数により評価した.

Type	Method	WordSim			
		Slex	WSR	WSS	Avg.
埋め込み層	ベースライン	.306	.186	.426	.306
	No Layer	.417	.306	.565	.429
	One Attn	.466	**.580**	.751	.599
	One Layer	**.500**	.575	**.765**	**.613**
出力層	ベースライン	.527	**.518**	.689	.578
	No Layer	.330	.275	.566	.390
	One Attn	.489	.460	.686	.545
	One Layer	**.539**	.504	**.717**	**.586**

における総計算量とベースラインの計算量を比較する．比較の際に，Avg. スコアのステップ数[4]による推移を確認し，ベースラインの最終ステップ時点での性能を上回ったステップ数で学習を終了した場合の計算量を求めた.[5]ベースラインに対する計算量の削減結果を表 2 に示す．性能を維持しながら，モデルサイズ 570M では最大で 11.52%，モデルサイズ 1B では最大で 28.33%の計算量を削減できた．

次に，ステージ 2 におけるメモリ使用量の削減率について述べる．ステージ 2 では埋め込み層と出力層のパラメータを固定するため，ベースラインに対してメモリ使用量が削減される．算出の結果，ベースラインに対するメモリ削減率は 16.1%となった．

5　分析および考察

4.2 節で示した実験結果が得られた理由，また，埋め込み層と出力層のパラメータを固定が望ましい理由について分析および考察を行う．

5.1　単語ベクトルとしての評価

モデルサイズ 570M のベースラインとステージ 1 で構築した埋め込み層および出力層に対して，単語ベクトルとしての性能を評価し，それらの性質を調査する．単語ベクトル評価ベンチマークデータセットとして，Word Similarity タスクの SLex [20], WSR,WSS [21] を使用した.[6]表 3 に評価結果を示す．

埋め込み層の評価結果において，One Layer が最も高い性能を示し，次いで One Attn，No Layer，そして標準の学習手法であるベースラインの性能が著しく低いことが確認された．一方，出力層の評価結果では，再び One Layer が最も高い性能を示し，次にベースライン，続いて One Attn，そして No Layer の性能が著しく低い結果となった．

5.2　埋め込み層・出力層を固定する利点

事前学習の安定性　表 1 の評価結果に示されている通り，提案法はベースラインよりも学習率を大きく設定することができる．これは埋め込み層と出力層を固定することにより，中間層から見た性質の違うパラメータがなくなることで，学習率を増やしても発散しづらくなったためと考えられる．

単語ベクトル性能の維持　ステージ 1 により単語ベクトルとして高い性能を獲得した埋め込み層や出力層をステージ 2 の初期値として利用する場合でもパラメータ固定をしないと，ステージ 2 の学習中に徐々に単語ベクトルとしての性能が劣化してしまう．すると折角良い性質を持った埋め込み層や出力層を用意しても意味がなくなってしまう．このような理由により，埋め込み層と出力層はステージ 2 の学習中はパラメータ固定し，良い性質維持することが望ましいと考えられる．

6　おわりに

LLM の効率的な事前学習法の一つとして，埋め込み層および出力層のパラメータ固定を伴う再利用による二段階学習を提案した．提案法では，ステージ 1 として，最終的に構築したい LLM の埋め込み層，出力層，1 層目の中間層を用いて標準的な事前学習を行う．次に，最終的に構築したいモデルの初期値にステージ 1 で得たモデルパラメータを代入する．その後，埋め込み層および出力層を固定し，標準的な事前学習により事前学習済みモデルを得る．

実験により，標準的な事前学習より計算量およびメモリ要求量を削減可能であると示された．また，学習が安定することから従来よりも学習率を上げて学習できるといった副次効果があることもわかった．さらに，構築した埋め込み層と出力層の埋め込み表現は単語ベクトルとしての性能が標準的な事前学習で得られる表現よりも優れており，それが LLM の最終的な性能向上に寄与することが示唆された．

この分析結果より，埋め込み層および出力層の単語ベクトルとしての性能が，LLM の性能に寄与する可能性があるという新たな知見が得られた．

4）　ステージ 1 とステージ 2 では 20k ステップで学習が終了する設定である．
5）　計算量とメモリ使用量の算出方法は付録 B に記す．
6）　評価方法の詳細は付録 C に記す．

謝辞

本研究は，JST ムーンショット型研究開発事業 JPMJMS2011-35 (fundamental research)，および，文部科学省の補助事業「生成 AI モデルの透明性・信頼性の確保に向けた研究開発拠点形成」の支援を受けたものです．また，本研究は九州大学情報基盤研究開発センター研究用計算機システムの一般利用を利用しました．さらに，本研究成果（の一部）は，データ活用社会創成プラットフォーム mdx [22] を利用して得られた物です．

参考文献

[1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. **arXiv preprint arXiv:2001.08361**, 2020.

[2] Chenze Shao, Fandong Meng, and Jie Zhou. Patch-level training for large language models. **arXiv preprint arXiv:2407.12665**, 2024.

[3] Wenyu Du, Tongxu Luo, Zihan Qiu, Zeyu Huang, Yikang Shen, Reynold Cheng, Yike Guo, and Jie Fu. Stacking your transformers: A closer look at model growth for efficient llm pre-training. **arXiv preprint arXiv:2405.15319**, 2024.

[4] Sheng Shen, Pete Walsh, Kurt Keutzer, Jesse Dodge, Matthew Peters, and Iz Beltagy. Staged training for transformer language models. In **International Conference on Machine Learning(ICML)**, pp. 19893–19908, 2022.

[5] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. **In International Conference on Learning Representations(ICLR)**, 2022.

[6] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In **International conference on machine learning(ICML)**, pp. 2790–2799, 2019.

[7] 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(ICLR)**, 2022.

[8] Rui Xiong, Yi Yang, Di He, Kai Zheng, Shuxin Zheng, Ling Lan, Zhijie Liu, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In **Proceedings of the 37th International Conference on Machine Learning(ICML)**, pp. 10524–10533, 2020.

[9] Biao Zhang and Rico Sennrich. Root mean square layer normalization. **Advances in Neural Information Processing Systems(NeurLPS)**, Vol. 32, , 2019.

[10] Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024.

[11] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. **arXiv preprint arXiv:2302.13971**, 2023.

[12] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. **In International Conference on Learning Representations(ICLR)**, 2017.

[13] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset, Aug 2016.

[14] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In **The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020**, pp. 8732–8740. AAAI Press, 2020.

[15] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In **Thirty-Fourth AAAI Conference on Artificial Intelligence(AAI)**, 2020.

[16] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics(ACL)**, 2019.

[17] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. **ArXiv**, Vol. abs/1803.05457, , 2018.

[18] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In **EMNLP**, 2018.

[19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. **Advances in neural information processing systems(NeurIPS)**, Vol. 33, pp. 1877–1901, 2020.

[20] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. **Computational Linguistics(CL)**, Vol. 41, No. 4, pp. 665–695, 2015.

[21] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In **Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics**, pp. 19–27, 2009.

[22] Toyotaro Suzumura, Akiyoshi Sugiki, Hiroyuki Takizawa, Akira Imakura, Hiroshi Nakamura, Kenjiro Taura, Tomohiro Kudoh, Toshihiro Hanawa, Yuji Sekiya, Hiroki Kobayashi, Yohei Kuga, Ryo Nakamura, Renhe Jiang, Junya Kawase, Masatoshi Hanai, Hiroshi Miyazaki, Tsutomu Ishizaki, Daisuke Shimotoku, Daisuke Miyamoto, Kento Aida, Atsuko Takefusa, Takashi Kurimoto, Koji Sasayama, Naoya Kitagawa, Ikki Fujiwara, Yusuke Tanimura, Takayuki Aoki, Toshio Endo, Satoshi Ohshima, Keiichiro Fukazawa, Susumu Date, and Toshihiro Uchibayashi. mdx: A cloud platform for supporting data science and cross-disciplinary research collaborations. In **2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)**, pp. 1–7, 2022.

[23] Noam Shazeer. Glu variants improve transformer. **arXiv preprint arXiv:2002.05202**, 2020.

[24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. **OpenAI blog**, Vol. 1, No. 8, p. 9, 2019.

[25] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Ethan Rutherford, Diego De Las Casas, Anna Guy, Jacob Menick, Katie Millican, et al. Training compute-optimal large language models. In **Advances in Neural Information Processing Systems (NeurIPS)**, 2022.

[26] Diederik P Kingma. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

[27] Teven Le Scao, Thomas Wang, Daniel Hesslow, Lucile Saulnier, Stas Bekman, M Saiful Bari, Stella Biderman, Hady Elsahar, Niklas Muennighoff, Jason Phang, et al. What language model to train if you have one million gpu hours? **In Findings of the Association for Computational Linguistics(EMNLP)**, p. 765–782, 2022.

[28] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. **arXiv preprint arXiv:1909.08053**, 2019.

[29] Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. Gpt-neox-20b: An open-source autoregressive language model. **Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models**, pp. 95–136, 2022.

[30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. **Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers(EACL)**, pp. 157–163, 2017.

表 4　モデルのパラメータ設定.

パラメータ数	570M	1B
次元数	1280	1600
層数	24	28
ヘッドの数	20	20
文脈長	1024	1024
トークン数	11.4B	20B
バッチサイズ	560	1008
総ステップ数	20000	20000
Adam β_1	0.9	0.9
Adam β_2	0.95	0.95

A　詳細なモデル設定

表 4 にモデル設定とハイパーパラメータを示す．モデルは LLaMA [11] のアーキテクチャ設計を踏襲し，層正規化に RMSNorm [9]，活性化関数に SwiGLU [23] を用い，語彙として GPT-2 [24] のものを使用する．なお，Grouped-Query Attention は適用しない．

検証するモデルのパラメータ数はステージ 2 において 570M および 1B とし，学習トークン数は Chinchilla 則 [25] に基づき，570M では 11.4B トークン，1B では 20B トークンに設定する．オプティマイザーには AdamW [26] を用い，重み減衰（weight decay）は 0.1 とし，勾配クリッピング（gradient clipping）は 1.0 を使用した．学習率のスケジューリングにはコサイン学習率スケジューラを利用し，学習率は 1,000 ステップで最大値に達し，最終的にはその 1/10 に減衰するように設定した．さらに，初期化方法は [27, 28, 29] を参考にし Attention 層や FFN 層の出力を小さくする初期化を採用した．

ステージ 1 では学習率を変動させ，検証損失が最も低い学習率設定のモデルをステージ 2 に再利用するようにし，ステージ 2 においても学習率によるハイパーパラメータ探索を行なった．具体的には，ステージ 1 における学習率の探索範囲はモデルサイズによらず，9e-4 から 4e-3 とする．ステージ 2 ではモデルサイズ 570M においては，ベースラインでは 7e-4 から 4e-3，提案法では 8e-4 から 4e-3 とし，モデルサイズ 1B においてはベースラインでは 4e-4 から 9e-4，ステージ 2 では 6e-4 から 9e-4 にする．

ステージ 1 で最も検証損失が低かった設定は，モデルサイズ 570M では No Layer は 8e-4，One Attn は 1e-3，One Layer は 2e-3 となり，モデルサイズ 1B では One Attn は 2e-3，One Layer は 2e-3 となった．

B　計算量とメモリ使用量の算出

下流タスクの Avg. スコアのステップ数による推移を確認し，ベースラインの最終ステップ時点での性能を上回ったステップ数で学習を終了した場合の計算量を求める．提案法の二段階学習では，ベースラインと比べ，ステージ 1 で学習する中間層を余分に学習しており[7]，ステージ 2 では埋め込み層と出力層を固定するため逆伝播の計算は発生しないが，順伝播の計算は行われる．計算量 C の FLOPS を算出する式としては，パラメータ数を N，学習トークン数を T とすると $C \approx 6 * N * T$ となる．ここで，係数 6 は 1 ステップあたりに必要な浮動小数点演算の回数を示しており，順伝播で 2 回分，逆伝播などその他の計算で 4 回分の浮動小数点演算があることに対応する．したがって，学習可能なパラメータ数を $N_{\text{trainable}}$，固定されているパラメータ数を $N_{\text{untrainable}}$ とすると，FLOPS は $C \approx (6 * N_{\text{trainable}} + 2 * N_{\text{untrainable}}) * T$ と計算できる．本稿では N を中間層と出力層のパラメータ数の合計とし，埋め込み層の計算は入力トークンの参照のみであるためコストを無視した．

次に，メモリ使用量の算出方法を述べる．メモリ使用量の計算はモデルのパラメータ数，勾配情報，およびオプティマイザの状態のサイズによって推定できると仮定する[8]．具体的には，パラメータ数 P_{model} および勾配情報は 16 ビット浮動小数点数で表現し，オプティマイザの状態は 32 ビット浮動小数点数で表現すると，バイト単位におけるモデルのメモリ使用量は $16 * P_{\text{model}}$ と計算できる．ここで係数 16 の内訳は，パラメータ数と勾配情報はそれぞれ 2，オプティマイザの状態は 12 に相当する．パラメータ数は中間層と埋め込み層，出力層の合計とする．さらに，ステージ 2 ではこれらが固定されるため，係数 16 のうち 14 が消失し 2 のみ有効となる．よって，埋め込み層と出力層のパラメータ数を $P_{\text{emb\&head}}$，中間層のパラメータ数を P_{layer} とするとステージ 2 におけるメモリ使用量は $16 * P_{\text{layer}} + 2 * P_{\text{emb\&head}}$ となる．

7）出力層の直前に位置する RMS 層も余分に学習するが，全体に対して小さな計算量であるため考慮に入れない．

8）さらに，一般的な Transformer モデルおよび Adam を用いた混合精度トレーニングを使用すると仮定する．

表 5　**類似度マップにおける Top-100 一致度.** 語彙に含まれる各単語ベクトル間のコサイン類似度マップを作成し，各単語に対して類似度が Top-100 に入る単語集合の一致度で層同士の類似性を評価する．

(Method 1,	Method 2)	埋め込み層	出力層
(ベースライン,	No Layer)	.102	.120
(ベースライン,	One Attn)	.148	.362
(ベースライン,	One Layer)	.156	.395
(No Layer,	One Attn)	.328	.171
(No Layer,	One Layer)	.316	.143
(One Attn,	One Layer)	.586	.433

C　単語ベクトルの評価方法

5.1 節で述べた単語ベクトルの評価では，使用するデータセットから，語彙外の単語が一語でも含まれるデータ（問題）を破棄する．ただし，先頭にスペースを表す特殊記号「Ġ」を付与することで語彙内となる単語は破棄せず，付与後の埋め込み表現を使用した．さらに，先頭に「Ġ」が付与された後の状態でも語彙に含まれる単語は，付与後の埋め込み表現を優先的に用いた．

D　考察

One Layer(None) と比較し One Attn(None) がうまく機能しなかった点，さらに 5.1 節における埋め込み層および出力層の単語ベクトルとしての評価結果に対する考察を述べる．

単語ベクトルのタスクにおいて，Skip-Gram word2vec では埋め込み層の方が出力層より優れており，ニューラル言語モデルでは出力層のほうが埋め込み層を上回ることが経験的に示されている [30]．5.1 節の結果も同様であり，これは No Layer が後方の単語だけを予測する窓サイズ 1 の Skip-Gram モデルである一方，One Attn，One Layer，ベースラインはいずれもニューラル言語モデルであるためと考えられる．

ステージ 1 に限定すると，One Layer が最も優れており，No Layer が最も劣る結果となったのは，No Layer がバイグラム言語モデルとなり，手前の単語以外を考慮できないことが原因だと考えられる．つまり，x_{input} に文脈に応じた変換作用 Total_Activation が加算されない事が原因だと推測する．このことから，One Attn は入力 x_{input} に対する変換能力が限定的であるため，No Layer と近い性能に留まったと説明できる．つまり，Total_Activation が小さい場合，すなわち x_{input} が支配的な場合には式 (5) の計算に近づき，バイグラムを行うための出力層に近い挙動になると考えられる．モデル間の埋め込み層と出力層の埋め込み表現の比較結果（表 5）において，No Layer に対して最も類似しているのは One Attn であることはこの仮説を支持する．また，ベースラインの埋め込み層の評価結果が One Layer より著しく低い理由としては，多層の Transformer 層を持ち x_{input} を多段階で変換できることから、埋め込み層の単語ベクトルとしての性能が向上しにくいことが要因だと考えられる．

まとめると，モデルの中間層に存在する Transformer 層（あるいはサブレイヤ）が増えるほど，埋め込み層の単語ベクトルとしての性能は低下し，出力層の性能は向上する．逆に減少すると，埋め込み層の性能は向上し，出力層は低下する可能性がある．