LoRA を活用した言語モデルの中間層蒸留

鈴木偉士 [1]　山田寛章 [1]　徳永健伸 [1]
[1] 東京科学大学　情報理工学院
{suzuki.t.dp@m, yamada@c, take@c}.titech.ac.jp

概要

近年，言語モデルの巨大化により，計算コストが大きく増加したため，性能を保ちつつモデルのパラメータ数を削減する手法が求められている．その一つに知識蒸留があり，中間層蒸留はその一種である．モデルの中間層出力も損失関数の計算に用いる中間層蒸留は有効とされてきたが，線形写像を推論時に用いないため，学習の効果が保証されない問題があった．本研究では，中間層蒸留の線形写像を LoRA のアダプターで代替し，推論時に除かれない線形写像を実現した LoRAILD を提案し，実験を行った．その結果，中間層蒸留の効果に対する否定的な結果を得た．

1　はじめに

事前学習言語モデルは様々な言語処理のタスクにおいて，高い性能を実現した．しかし，近年では，パラメータサイズ 1750 億の GPT-3[1] や，4050 億の Llama3.1[2]，パラメータ数は不明だが GPT-4[3] などに見られるように，言語モデルのパラメータ数が巨大化し，計算コストが大きく増加したため，モデルの性能を保ったままパラメータ数を削減する手法が求められている．

知識蒸留 [4] は，モデル圧縮の手法の一つである．パラメータ数の大きい教師モデルの出力と，パラメータ数の小さい生徒モデルの出力を用いて損失関数を計算し，出力同士の差が小さくなるように学習をすることで，生徒モデルが教師モデルと同様の出力が出来るようにする．通常，知識蒸留はモデルの最終出力を用いるが，中間層同士の出力でも損失関数を計算する中間層蒸留 (ILD) が存在し，より性能を向上させることが出来ると先行研究で主張されている [5, 6, 7]．

中間層蒸留では，教師モデルと生徒モデルの中間層出力の次元数が異なる場合，そのままでは損失関数を計算できない．先行研究では，この問題を，線形写像を用いて次元数を揃えることで解決した [7]．しかし，この線形写像は学習時のみで用いられ，推論時には無視される．この方法では線形写像の存在を前提とした学習が行われるため，推論の為に有効な学習となっている保証がない．

本研究では，LoRA のアダプターをこの線形写像の代わりとして用いることで，推論時にも線形写像が維持される LoRAILD を提案し，実験を行った．しかし，LoRAILD は既存手法を上回らず，さらに，先行研究の手法も性能を再現できなかった．

また，中間層出力のクラスター分析を行い，その結果から，言語モデルにおいて中間層蒸留は有効でないと結論付けた．

本研究のソースコードは GitHub[1)] にて公開している．

2　関連研究

2.1　知識蒸留 (Vanilla KD)

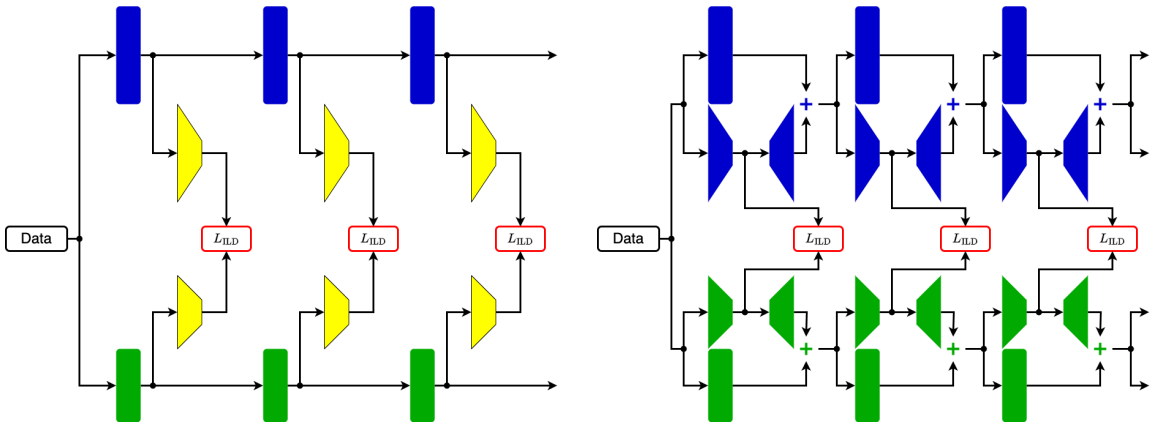
知識蒸留 [4] とは，学習元のモデル（教師モデル）と圧縮先のモデル（生徒モデル）を用意し，生徒モデルの出力を教師モデルの出力に近づけるように学習することで，性能を保ちつつ小さなモデルを得られる手法である．生徒モデルの学習時，正解ラベルとの誤差による損失に加えて，自身の出力と教師モデルの出力との差による損失を用いて学習する．損失関数は以下のとおりである．

$$L = \lambda L_{\mathrm{CE}} + (1-\lambda) L_{\mathrm{KD}} \tag{1}$$

$$L_{\mathrm{KD}} = \mathrm{KL}(\mathrm{Teacher}(X), \mathrm{Student}(X)) \tag{2}$$

L_{CE} は交差エントロピー誤差，$\mathrm{KL}(\cdot,\cdot)$ は KL ダイバージェンス，$\mathrm{Teacher}(X)$，$\mathrm{Student}(X)$ はそれぞれ，教師モデル，生徒モデルにデータ X を入力した際に出力として得られる確率分布とする．生徒モデルの学習では，教師モデルのパラメータは固定され，生徒モデルのパラメータのみが更新される．

1) https://github.com/TKC002/LoRAILD



(a) 従来　　(b) LoRAILD

図 1: 手法の概要 (青: 教師モデル，緑: 生徒モデル，黄: 線形写像)

2.2 中間層蒸留 (ILD)

中間層蒸留とは，通常の知識蒸留に加えて，教師モデルと生徒モデルの中間層同士でも損失を計算し，学習に用いる手法である．教師モデルと生徒モデルの層数が異なる場合，そのままでは中間層の損失関数を計算できない．そのため，教師モデルの層を生徒モデルに対応させる層のアライメントの手法が必要となる．先行研究では，このアライメントの手法が模索されてきた．

Sun ら [5] は，教師モデルから生徒モデルの層数分の層を選び，損失関数に用いる PKD を提案した．PKD では，学習の間，同じ教師モデルの層を使用し続ける．Passban ら [6] は教師モデルの層の重み付き平均を損失関数に用いる ALP-KD を提案した．また，Haidar ら [7] は PKD と同様に教師モデルの一部の層を選ぶが，選び方をエポックごとにランダムに変更する RAIL-KD を提案した．

教師モデルと生徒モデルの中間層出力の次元数が異なる場合，中間層出力を線形写像などを用いて次元数を揃える必要がある．この操作は，学習時でのみ用いられる．生徒モデルは線形写像などが存在する前提で学習を行うため，適切に学習されているかの保証がない．

2.3 LoRA

LoRA[8] は，学習時の計算コストを削減する手法の一つである．モデルの層など，特定のモジュールに対し，サイズが $\mathbb{R}^{i\times r}, \mathbb{R}^{r\times o}$ である 2 つの低ランク行列を割り当てる．i と o はそれぞれモジュールの入力次元数と出力次元数であり，r はランクと呼ばれるパラメータである．低ランク行列は元のモジュールよりもパラメータ数が小さいので，学習対象を低ランク行列のみとすることで，訓練するパラメータ数を削減できる．

3 提案手法

3.1 LoRAILD

本研究では，LoRAILD を提案する．LoRAILD は，LoRA の低ランク行列の前半部分を従来の中間層蒸留の線形写像の代替として用いる手法である．図 1a と図 1b に，それぞれ従来の中間層蒸留と LoRAILD の概要図を掲載する．従来の中間層蒸留では，次元数を揃えるための線形写像 (図 1a の黄色の部分) の出力を用いて損失を計算していたが，LoRAILD では，LoRA アダプター (図 1b の台形の部分) の前半部分の出力を用いて損失関数を計算する．線形写像が LoRA アダプターであるため，従来の中間層蒸留と異なり，推論時にも用いられる．

各ミニバッチに対する損失関数は以下のとおりである．

$$L_{\text{ILD}} = \sum_{x\in batch} \left\| \frac{h_x^T}{\|h_x^T\|_2} - \frac{h_x^{S_\theta}}{\|h_x^{S_\theta}\|_2} \right\|_2^2 \tag{3}$$

$$L = \lambda_1 L_{\text{CE}} + \lambda_2 L_{\text{KD}} + \lambda_3 L_{\text{ILD}} \tag{4}$$

h_x^* は，データ x に対する LoRA アダプターの前半の出力である．各層の出力を連結して損失関数の計算に用いる．T と S_θ はそれぞれ教師モデル，生徒モデルを表す．L_{CE}，L_{KD} は式 1 のものをそのまま用いる．この損失関数は，従来の中間層蒸留と同じものである．

表 1: 実験結果 (r = 32), 左: 検証データ, 右: 評価データ

		検証データ						評価データ					
		CoLA	MRPC	QNLI	RTE	SST-2	STS-B	CoLA	MRPC	QNLI	RTE	SST-2	STS-B
w/o KD		0.615	0.872	0.894	0.692	0.947	0.896	0.567	0.876	0.906	0.676	0.919	0.881
Vanilla KD		0.635	0.888	0.903	0.716	0.954	0.916	0.566	0.874	0.912	0.622	0.925	0.883
RAIL-KD	RAIL-KD$_c$	0.625	0.902	0.905	0.716	0.954	0.917	0.568	**0.892**	0.916	0.658	**0.929**	0.882
	RAIL-KD$_l$	0.633	0.899	0.899	0.597	0.952	0.916	0.585	0.882	0.907	0.522	0.907	0.886
	Curriculum	**0.650**	**0.903**	**0.907**	**0.748**	0.955	**0.920**	0.568	0.889	0.912	**0.677**	0.928	0.881
LoRAILD	Fixed	0.628	0.883	0.905	0.688	0.955	0.912	0.565	0.880	0.912	0.651	0.922	0.880
	Average	0.637	0.871	0.904	0.717	0.954	0.919	**0.596**	0.874	0.913	0.656	0.921	**0.886**
	Random step	0.633	0.848	0.904	0.680	**0.956**	0.913	0.592	0.846	**0.916**	0.637	0.920	0.881
	Random epoch	0.627	0.888	0.904	0.726	0.955	0.912	0.573	0.886	0.911	0.659	0.915	0.882

3.2 層のアライメント

教師モデルと生徒モデルの層数が異なる場合，アライメントが必要になる．本研究では以下の 3 通りの実験を行った．

Fixed 学習中，常に同じ層を選択する．

Average 生徒モデルの各層に，教師モデルの層を配分し，配分した層の出力を平均する．

Random ランダムに教師モデルの層を選択する．選んだ層の順序は維持する．ミニバッチごとに選ぶ Random step とエポックごとに選ぶ Random epoch を実験した．

Fixed，Average，Random はそれぞれ，先行研究の PKD[5]，ALP-KD[6]，RAIL-KD[7] で用いられたものとほぼ同じものである．Fixed と Average に関しては，付録 A に対応させた層を記載した．

3.3 カリキュラム学習

カリキュラム学習を行わず，LoRAILD の実験を行った結果，L_{ILD} が先に減少し，L_{CE} の減少が遅くなり，最終的な性能が向上しなかった．そのため，まず，L_{CE} と L_{KD} のみの学習を行い，後から L_{ILD} を加えるカリキュラム学習を導入した．

4 実験

4.1 実験設定

実験では，教師モデルと生徒モデルにそれぞれ RoBERTa-large，DistilRoBERTa-base[9] を用いた．教師モデルと生徒モデル共に LoRA アダプターを追加し，教師モデルでは，LoRA のみの学習を行い，生徒モデルでは，LoRA と元のモデルの両方を学習した．

データセットには，GLUE[10] の中から，CoLA，MRPC，QNLI，RTE，SST-2，STS-B の 6 タスクを利用した．GLUE ベンチマークでは評価用セットの正解ラベルが公開されていないため，検証用セットとして公開されているデータを本実験での評価に用いた．また，GLUE から公開されている訓練用セットの 10%を本実験での検証用として，残りの 90%を本実験での訓練用として用いている．

比較手法は，生徒モデルのみを用いた学習 (w/o KD)，通常の知識蒸留 (Vanilla KD)，RAIL-KD(RAIL-KD$_c$, RAIL-KD$_l$) と，元論文に無いカリキュラム学習を行った RAIL-KD$_c$ とした (Curriculum)．実験は 5 回行い，有意水準 2.5%の並べ替え検定を片側検定で行うことで有意差を判断した．

また，ハイパーパラメータは，付録 B に掲載した．

4.2 実験結果

r = 32 の評価データでの結果を，表 1 に示す．太字は各タスクでの最高性能のものを表す．太字のものは，下線が引かれているものより有意に高い．

提案手法は，検証データ，評価データともに，一部のタスクでは，従来の手法を上回ったが，全体的な性能は改善しなかった．また，RAIL-KD に関して，特に評価データにおいては，w/o KD や Vanilla KD と比べて全体的に改善したものはなく，中間層蒸留が有効であるという先行研究の主張を再現できなかった．

5 分析

5.1 手法

LoRAILD が有効でない原因が，中間層の学習が適切に行われていないことにあると考え，その仮説の検証のために中間層出力を用いてクラスター分析を行った．分析の対象は，LoRA アダプターの出力

表 2: クラスター分析の結果

中間層出力

Task	RAIL-KD$_{\mathrm{I}}$ 0	1	2	3	4	5	Module	LoRAILD 0	1	2	3	4	5
CoLA	-0.001	-0.001	0.001	0.001	0.002	0.311		-0.003	-0.003	-0.001	-0.001	0.339	0.400
MRPC	0.018	0.025	0.024	0.179	0.313	0.347		0.020	0.023	0.056	0.250	0.266	0.235
QNLI	0.008	0.008	0.008	0.008	0.010	0.619		0.008	0.006	0.007	0.193	0.610	0.664
RTE	-0.003	0.000	0.000	0.001	0.001	0.000		-0.003	-0.003	-0.003	-0.003	0.067	0.084
SST-2	-0.001	-0.001	-0.001	0.000	0.400	0.677		0.000	0.000	0.000	0.672	0.698	0.707
$h_x^{S_\theta}$(損失関数の計算に用いるベクトル)													
CoLA	-0.003	0.004	0.007	0.007	0.008	0.187	Query	0.005	0.000	-0.001	-0.001	0.011	0.294
							Key	0.004	-0.001	0.000	-0.001	-0.003	0.341
							Value	0.004	-0.001	0.000	0.002	0.003	0.333
MRPC	0.006	0.012	0.011	0.073	0.313	0.410	Query	0.005	0.019	0.023	0.083	0.151	0.230
							Key	0.008	0.019	0.023	0.106	0.096	0.256
							Value	0.004	0.019	0.016	0.074	0.205	0.281
QNLI	0.006	0.007	0.008	0.007	0.007	0.060	Query	0.017	0.014	0.004	0.010	0.361	0.619
							Key	0.021	0.014	0.004	0.010	0.412	0.616
							Value	0.020	0.013	0.004	0.015	0.376	0.614
RTE	-0.003	-0.001	0.000	0.002	0.000	0.003	Query	-0.003	-0.003	-0.003	-0.003	-0.003	0.049
							Key	-0.003	-0.003	-0.003	-0.003	0.010	0.070
							Value	-0.003	-0.003	-0.003	-0.003	-0.002	0.062
SST-2	0.000	0.000	-0.001	0.000	0.000	0.033	Query	-0.001	0.002	-0.001	0.001	0.595	0.694
							Key	-0.001	0.001	-0.001	0.007	0.630	0.696
							Value	-0.001	0.002	-0.001	0.005	0.630	0.697

を加算した生徒モデルの中間層出力と，実際に損失関数の計算に用いる $h_x^{S_\theta}$ の 2 つとした．分析の手法は Ajusted Rand Score[11, 12] とし，正解ラベルは元のデータの正解ラベルとした．クラスタリングスコアが高いほど，中間層の時点でラベルを弁別する情報を学習できており，中間層蒸留が有効に働いていると解釈できる．

比較手法は RAIL-KD$_{\mathrm{I}}$ とし，生徒モデルの中間層出力と損失関数の計算に用いるベクトルを分析対象とした．

5.2 分析結果と考察

評価データに対するクラスタリングスコアを表 2 に示す．4 章で述べた 5 回の実験で得られたチェックポイントを用い，出力されたスコアを平均した．左が RAIL-KD$_{\mathrm{I}}$，右が LoRAILD である．0 から 5 の数字は生徒モデルの層の番号で，0 が最も入力に近い層，5 が最終出力に近い層である．また，濃い青になるほどスコアが高いことを表す．

中間層出力では，LoRAILD が MRPC を除き，最終層の数値で RAIL-KD$_{\mathrm{I}}$ を上回った．また，CoLA, QNLI, SST-2 については，RAIL-KD$_{\mathrm{I}}$ と比較して出力側から見て，より深い層までスコアが高い．$h_x^{S_\theta}$ でも，MRPC を除いて LoRAILD の後半の層でのスコアが RAIL-KD$_{\mathrm{I}}$ よりも，高くなっている．

以上のことから，LoRAILD は，RAIL-KD$_{\mathrm{I}}$ よりも，中間層の損失関数で多くの情報を学習できていると考えられる．

LoRAILD は中間層で学習できているにもかかわらず，表 1 に示した実際の性能では，LoRAILD は高性能を発揮できなかった．そのため，先行研究の主張と異なり，言語モデルにおいては中間層蒸留の有効性が低いと考えられる．

6 結論

本研究では，中間層蒸留の線形写像が推論時に用いられない問題を解決するため，LoRA アダプターを線形写像として用いることで，推論時にも線形写像が利用される LoRAILD を提案したが，有効性を証明できなかった．また，中間層出力に対するクラスター分析も行い，中間層蒸留の効果に対する否定的な結果を得た．

参考文献

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-Shot Learners. arxiv preprint, p. arXiv:2005.14165, May 2020.

[2] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 Herd of Models. arxiv preprint, p. arXiv:2407.21783, July 2024.

[3] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. GPT-4 Technical Report. arXiv preprint, p. arXiv:2303.08774, March 2023.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. arxiv preprint, p. arXiv:1503.02531, March 2015.

[5] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 4323–4332, Hong Kong, China, November 2019. Association for Computational Linguistics.

[6] Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. Alp-kd: Attention-based layer projection for knowledge distillation. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 15, pp. 13657–13665, May 2021.

[7] Md Akmal Haidar, Nithin Anchuri, Mehdi Rezagholizadeh, Abbas Ghaddar, Philippe Langlais, and Pascal Poupart. RAIL-KD: RAndom intermediate layer mapping for knowledge distillation. In Findings of the Association for Computational Linguistics: NAACL 2022, pp. 1389–1400, Seattle, United States, July 2022. Association for Computational Linguistics.

[8] 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. arxiv preprint, p. arXiv:2106.09685, June 2021.

[9] 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.

[10] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[11] Lawrence Hubert and Phipps Arabie. Comparing partitions. Journal of Classification, Vol. 2, No. 1, pp. 193–218, Dec 1985.

[12] Douglas Steinley. Properties of the Hubert-Arabie adjusted rand index. Psychol Methods, Vol. 9, No. 3, pp. 386–396, September 2004.

A　層のアライメント

表 3 にアライメント手法 Fixed，Average における生徒モデルと教師モデルのアライメントを示す．Fixed では，生徒モデルの 0 から 5 層目までの $h_x^{S_\theta}$ を連結したベクトルと，教師モデルの 3，7，...，23 層目の h_x^T を連結したベクトルを用いて L_{ILD} を計算した．Average では { 教師モデルの 0，1，2，3 層の h_x^T の平均 }，...，{ 教師モデルの 20，21，22，23 層の h_x^T の平均 } を連結したものを L_{ILD} の計算に用いた．

表 3: アライメント

Fixed の場合

生徒モデル	教師モデル
0	3
1	7
2	11
3	15
4	19
5	23

Average の場合

生徒モデル	教師モデル
0	0, 1, 2, 3
1	4, 5, 6, 7
2	8, 9, 10, 11
3	12, 13, 14, 15
4	16, 17, 18, 19
5	20, 21, 22, 23

B　ハイパーパラメータ

実験に用いたハイパーパラメータを表 4 に示す．カリキュラム学習では，初期状態から最終状態まで $(\lambda_1, \lambda_2, \lambda_3)$ を変動させる．RTE タスクのみ，最終状態を 2 種類として実験した．

表 4: ハイパーパラメータ

学習率	{1, 2, 5}e-{4, 5} から選択
r	{8, 32, 64, 128} から選択
エポック数	20

カリキュラム学習

カリキュラム学習開始エポック	
CoLA	5-10 から選択
MRPC	5-10 から選択
QNLI	5-10 から選択
RTE	5-10 から選択
SST-2	2-4 から選択
STS-B	2-4 から選択
$(\lambda_1, \lambda_2, \lambda_3)$	
初期状態	$(0.5, 0.5, 0)$
最終状態	$(0.333, 0.333, 0.333)$
最終状態 (RTE のみ追加)	$(0.5, 0, 0.5)$

C　比較手法

- w/o KD: KD なし
- Vanilla KD: 通常の KD
- RAIL-KD$_{\mathrm{c}}$: RAIL-KD で，中間層出力を連結してから一度に損失関数を計算する
- RAIL-KD$_{\mathrm{c}}$: RAIL-KD で，層ごとに損失関数を計算する
- Curriculum: カリキュラム学習を行った RAIL-KD$_{\mathrm{c}}$