敵対的学習を用いた知識蒸留への中間層蒸留と対照学習の導入

鈴木 偉士　山田 寛章　徳永 健伸
東京工業大学 情報理工学院
{suzuki.t.dp@m, yamada@c, take@c}.titech.ac.jp

概要

知識蒸留 (KD) とは，大規模なニューラルネットワークを圧縮する手法の一つである．言語モデル向け KD の中で最高性能の手法は，敵対的学習に中間層出力と対照学習を導入した CILDA と呼ばれる手法である．CILDA の学習は最大化ステップと最小化ステップに分かれているが，中間層出力と対照学習は最大化ステップでのみ活用されている．本研究では，最小化ステップに中間層蒸留と対照学習を導入し，性能を向上させることを目指した．しかし，既存手法に対して有意な差は確認できなかったため，原因分析のために CILDA 単体の再現実験を行ったところ，先行研究の主張とは異なり，GLUE における複数のタスクで CILDA がそれ以前の手法の性能を上回らないという結果を得た．

1　はじめに

BERT[1] や RoBERTa[2] のような事前学習済み言語モデルは，様々な言語処理タスクにおいて，高い性能を発揮しているが，近年では，GPT-3[3] や，PaLM[4] に見られるように，パラメータ数が大幅に増加し，通常のデバイスでは扱えなくなっている．

知識蒸留 (KD)[5] は，ニューラルネットワークを圧縮する手法の一つである．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の KL ダイバージェンスを小さくする学習をすることで，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．先行研究により，知識蒸留は，教師モデルと生徒モデルの最終的な出力間だけでなく，中間層の出力間の誤差の利用 [6, 7, 8]，データ拡張と敵対的学習の導入 [9]，対照学習の導入 [10] により，性能が向上することが報告されている．Haidar ら [11] は知識蒸留に中間層の利用，敵対的学習，対照学習を導入した CILDA と呼ばれる手法を提案し，GLUE タスクにおいて SOTA を達成した．

しかし，Haidar らの研究 [11] では，データ拡張と敵対的学習に用いられる Generator の学習にのみ中間層の利用と対照学習の導入を行っており，生徒モデルの学習については通常の KD から改良されていない．本研究では CILDA の生徒モデルの学習時に中間層の利用と対照学習を導入し，さらに性能を向上させることを目指す．

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 [5] とは，モデル圧縮の手法の 1 つである．この手法では，パラメータ数が小さい生徒モデルがタスクの正解ラベルとの誤差のほかに，パラメータ数が大きく性能の高い教師モデルの出力との誤差を用いて学習する．損失関数は以下のとおりである．

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

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

ただし，L_{CE} は交差エントロピー誤差，$KL(,)$ は KL ダイバージェンス，$Teacher(X)$，$Student(X)$ はそれぞれ，教師モデル，生徒モデルにデータ X を入力した際の出力する確率分布とする．学習対象は生徒モデルのみであり，教師モデルのパラメータは固定される．

2.2　RAIL-KD

RAIL-KD[8] は通常の知識蒸留に加えて中間層の出力の情報も用いる中間層蒸留の一種である．RAIL-KD では，各エポックごとに教師モデルから生徒モデルの中間層の数だけ前から順にランダムに中間層を選び，生徒モデルの中間層との誤差を計算

1) https://github.com/TKC002/CILDA-plus-minILD

する．誤差の計算の方法には以下の 2 通りがある．

$$L_{RAIL-KD^l} = \sum_{x \in X} \sum_{\kappa=0}^{m} \left\| \frac{\hat{h}^T_{a_\kappa, x}}{\|\hat{h}^T_x\|_2} - \frac{\hat{h}^{S_\theta}_{\kappa, x}}{\|\hat{h}^{S_\theta}_x\|_2} \right\|_2^2 \tag{3}$$

$$L_{RAIL-KD^c} = \sum_{x \in X} \left\| \frac{\hat{h}^T_x}{\|\hat{h}^T_x\|_2} - \frac{\hat{h}^{S_\theta}_x}{\|\hat{h}^{S_\theta}_x\|_2} \right\|_2^2 \tag{4}$$

$\hat{h}^*_*$, X, κ, a はそれぞれモデルの中間層の出力，訓練データ，生徒モデルの中間層の数，選択した中間層のインデックスの配列である．T は教師モデル，S_θ は生徒モデルを表す．

$L_{RAIL-KD^l}$ では，1 層ずつの中間層同士の誤差の和をとり，$L_{RAIL-KD^c}$ では，選択した層を連結してから誤差をとる．教師モデルと生徒モデルの中間層の出力の次元数が異なる場合は損失関数の計算前に h に対して線形変換を行い，次元数を揃える．

最終的な損失関数は以下のようになる．

$$L = \lambda_1 L_{CE} + \lambda_2 L_{KD} + \lambda_3 L_{RAIL-KD} \tag{5}$$

2.3 MATE-KD

MATE-KD[9] は知識蒸留に敵対的学習を導入した手法である．この手法では学習は最大化ステップと最小化ステップの 2 つのステップに分かれている．

最大化ステップでは，まず訓練データに一定の確率でマスクをかける．その後 Generator と呼ばれる事前学習済みマスク型言語モデルを用いてマスクされた部分を埋めることでデータ拡張を行う．拡張されたデータを用いて以下の損失関数を最大化するように Generator を学習する．

$$L_{ADV} = KL(Teacher(X'), Student(X')) \tag{6}$$

ただし，X' は拡張されたデータである．最大化ステップでは生徒モデルのパラメータは固定される．

最小化ステップでは以下の損失関数を最小化するように生徒モデルを学習する．最小化ステップの間は Generator のパラメータは固定される．

$$L = \lambda_1 L_{CE} + \lambda_2 L_{KD} + \lambda_3 L_{ADV} \tag{7}$$

学習時は最大化ステップを n_G ステップ行ってから最小化ステップを n_S ステップ行うことを繰り返す．

2.4 CILDA

CILDA[11] は知識蒸留に中間層出力の利用，敵対的学習, 対照学習を導入した手法である．MATE-KD と同様に最大化ステップと最小化ステップに分かれている．

最大化ステップでは以下の損失関数を最大化する．

$$L_G = \alpha_1 L_{ADV} + \alpha_2 L_{CRD} \tag{8}$$

$$L_{CRD} = -\log \frac{\exp(< \bar{h}^T_k, \bar{h}^{S_\theta}_k > /\tau_2)}{\sum_{j=0}^{K} \exp(< \bar{h}^T_j, \bar{h}^{S_\theta}_k > /\tau_2)} \tag{9}$$

ただし，$\bar{h}^*_*$ はモデルの全ての中間層の出力を連結し線形変換を施したもの，k はミニバッチの中のデータのインデックス，K はバッチサイズ，$<, >$ はコサイン類似度とする．

最小化ステップでは以下の損失関数に拡張されたデータと元データを入力して最小化する．

$$L = \lambda_1 L_{CE} + \lambda_2 L_{KD} \tag{10}$$

3 提案手法

提案手法を本論文では CILDA+minILD と呼ぶ．CILDA+minILD では，CILDA の最小化ステップに中間層出力の利用と対照学習を導入する．そのために損失関数に L_{CRD} を追加する．損失関数は以下のようになる．

$$L = \lambda_1 L_{CE} + \lambda_2 L_{KD} + \lambda_3 \frac{1}{\log K} L_{CRD} \tag{11}$$

K はバッチサイズである．L_{CRD} の値はバッチサイズの対数に依存し，バッチサイズが大きいほど値が大きくなる．そのため L_{CRD} の係数に $\frac{1}{\log K}$ をかける．最大化ステップにおいても同様に L_{CRD} の係数に $\frac{1}{\log K}$ をかける．

4 実験

4.1 実験設定

4.1.1 データセット

蒸留対象のタスクは GLUE[12] の中から，学習に要する時間の短い CoLA，MRPC，RTE，STS-B の 4 タスクを利用する．評価指標は先行研究 [11] に倣い，それぞれマシューズ相関係数，f1 スコア，正解率，ピアソンの相関係数とする．

4.1.2 モデル

教師モデルには，RoBERTa-large[2] を元に，各タスク毎に教師モデルをファインチューニングし，dev set において各評価指標で最高性能を示したモデルを用いた．ファインチューニングは，バッチサイズ 128，3 エポックとして，異なる 4 つの学

習率 1e-05, 2e-05, 5e-05, 1e-04 について各 5 回行った．生徒モデルには DistilRoBERTa[13] を使用した．MATE-KD，CILDA，CILDA+minILD で用いる Generator には RoBERTa-large[2] を用いた．

4.1.3 比較手法

提案手法と比較する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，MATE-KD，CILDA を用いた．RAIL-KD の $L_{RAIL-KD}$ には $L_{RAIL-KD^c}$ を用いた．

4.1.4 ハイパーパラメータ

学習率を 1e-05, 2e-05, 5e-05, 1e-04 の 4 通りからハイパーパラメータ探索で選択し，バッチサイズは w/o KD，Vanilla KD，RAIL-KD，MATE-KD では 128，CILDA，CILDA+minILD では 64 とした．損失関数の係数は，Vanilla KD では λ は 1/2，RAIL-KD と MATE-KD の $\lambda_1, \lambda_2, \lambda_3$ はすべて 1/3 とした．CILDA の λ_1, λ_2 は元データを入力する際にはともに 1/3 とし，拡張されたデータに対してはそれぞれ 2/9, 1/9 とした．α_1, α_2 はともに 1/2，τ_2 は 2 とした．CILDA+minILD では元データに対しては $\lambda_1 = 2/9, \lambda_2 = 2/9, \lambda_3 = 2/9$，拡張されたデータに対しては $\lambda_1 = 1/6, \lambda_2 = 1/12, \lambda_3 = 1/12$ とした．RAIL-KD，CILDA，CILDA+minILD での中間層の線形変換後の次元数は 128 とした．ハイパーパラメータ探索では 5 エポックの実験を 1 回行い，5 エポックの時点で dev set でもっとも性能の高いハイパーパラメータを選択した．MATE-KD, CILDA, CILDA+minILD では $n_G = 10, n_S = 100$ とした．

4.1.5 動作環境

本研究では PyTorch フレームワークを使用し，モデルは Huggingface Transformers のものを使用した．また，GPU には 4 枚の Nvidia 社製 RTX A6000 を使用した．節 4.1.4 に記載したバッチサイズは 4 枚の GPU の合計であり，GPU1 枚あたりのバッチサイズは上記の 4 分の 1 である．

4.1.6 本実験

本実験ではハイパーパラメータ探索で選んだ学習率を用いて 20 エポックの実験を 5 回行う．1 エポックごとにチェックポイントを保存し，各実験で最も評価指標の高いモデルを使用する．その後，有意水準を 0.05 として並べ替え検定を片側検定で行う．

4.2 結果

表 1 に各実験の GLUE の dev set における平均を示す．括弧内の数字は標準偏差である．タスクごとに最良のものを太字としており，下線が引かれているものよりも統計的に有意に評価指標が高い．教師モデルの行は，平均的に最も性能が高い学習率における結果である．

表 1 GLUE dev set

手法 ＼ タスク	CoLA	MRPC	RTE	STS-B
教師モデル	0.620 (.022)	0.916 (.004)	0.710 (.107)	0.911 (.002)
w/o KD	0.619 (.010)	0.897 (.006)	0.674 (.016)	0.885 (.004)
Vanilla KD	0.619 (.008)	0.906 (.005)	0.703 (.007)	0.886 (.005)
RAIL-KD	0.627 (.013)	0.911 (.005)	0.645 (.017)	0.887 (.003)
MATE-KD	**0.646** (.003)	0.909 (.003)	**0.722** (.012)	**0.894** (.002)
CILDA	0.610 (.006)	0.913 (.004)	0.703 (.014)	0.868 (.002)
CILDA+miniLD	0.612 (.011)	**0.914** (.004)	0.700 (.014)	0.866 (.002)

提案手法である CILDA+minILD は使用したタスクでは MRPC 以外で MATE-KD を超える結果にならなかった．また，先行研究では MATE-KD を超える結果を示した CILDA も MRPC 以外で MATE-KD を超えることはなかった．

5 CILDA の有効性の再検証

本研究では，CILDA のスコアが MATE-KD を超えず先行研究とは異なる結果となった．そのことが CILDA+minILD が有効でなかった理由であるという仮説をたて，CILDA の有効性を検証した．

CILDA のスコアが低かった原因としてハイパーパラメータが適切でなかったからである可能性がある．先行研究では，バッチサイズは 8, 16, 32 から，学習率は 1e-05，2e-05，4e-06 から選択しており本研究で使用したハイパーパラメータとは異なる．

5.1 再検証の実験設定

この仮説を検証するため，CILDA を実験を先行研究のハイパーパラメータに合わせて実験を行った．n_G は CoLA，MRPC，RTE では 20，STS-B では 10 とした．n_S はすべてのタスクにおいて 100 とした．ハイパーパラメータ探索は 5 エポックの

表 2 RoBERTa の dev set における結果

手法 ＼ タスク	CoLA	MRPC	RTE	STS-B
教師モデル	0.620 (.022)	0.916 (.004)	0.710 (.107)	0.911 (.002)
w/o KD	0.619 (.010)	0.897 (.006)	0.674 (.016)	0.885 (.004)
Vanilla KD	0.619 (.008)	0.906 (.005)	0.703 (.007)	0.886 (.005)
RAIL-KD	0.627 (.013)	0.911 (.005)	0.645 (.017)	0.887 (.003)
MATE-KD	**0.646** (.003)	0.909 (.003)	**0.722** (.012)	**0.894** (.002)
CILDA	0.632 (.006)	**0.917** (.004)	0.688 (.006)	0.867 (.003)

表 3 BERT の dev set における結果

手法 ＼ タスク	CoLA	MRPC	RTE	STS-B
教師モデル	0.634 (.019)	0.892 (.021)	0.677 (.010)	0.886 (.011)
w/o KD	0.503 (.005)	0.881 (.004)	0.602 (.018)	0.860 (.002)
Vanilla KD	0.519 (.013)	0.883 (.006)	0.575 (.014)	0.863 (.002)
RAIL-KD	0.532 (.015)	0.888 (.006)	0.611 (.018)	0.861 (.003)
MATE-KD	**0.551** (.007)	0.883 (.003)	**0.634** (.017)	**0.872** (.002)
CILDA	0.511 (.018)	**0.890** (.007)	0.624 (.023)	0.841 (.008)

実験を 5 回行い，5 エポック時点での評価指標の平均が最も高いバッチサイズ，学習率の組を探索する．その後本実験を行う．本実験の手順は節 4.1.6 に示したものと同じである．実験では GPU に RTX A6000 を 1 枚使用した．追加検証のため RoBERTa，BERT[1]，BART[14] の 3 通りのモデルを用いて実験を行った．BERT の実験では教師モデルと Generator を BERT-large（cased），生徒モデルを DistilBERT[13]，BART の実験では教師モデルと Generator を BART-large，生徒モデルを BART-base とした．BART の実験では教師モデルのファインチューニングを 10 エポックとし，各エポックでチェックポイントを保存し，最良のチェックポイントを用いた．

5.2 再検証実験結果

表 2，3，4 に結果を示す．RoBERTa の実験では，節 4.2 と比較して CILDA のスコアは向上したが，傾向は変わらず MARC のみ MATE-KD を上回る結果となった．BERT の実験でも RoBERTa の実験と同様の傾向となった．BART の実験では RoBERTa，BERT の場合と異なり，タスク毎に最高性能の手法が異なっているため，知識蒸留に中間層蒸留，敵対的学習，対照学習を導入することが BART においては有効であるとは確認できなかった．ただし，Vanilla KD は w/o KD と比較して高いスコアが得られた．そのため知識蒸留自体は BART においても有効である可能性がある．表 4 には記載していないが CoLA，RTE，STS-B では w/o KD と Vanilla KD の間に有意差は確認できず，MRPC のみ Vanila KD の方が有意に高いという結果であった．

表 4 BART の dev set における結果

手法 ＼ タスク	CoLA	MRPC	RTE	STS-B
教師モデル	0.617 (.009)	0.916 (.004)	0.843 (.020)	0.901 (.017)
w/o KD	0.532 (.013)	0.890 (.013)	0.742 (.021)	0.899 (.004)
Vanilla KD	0.543 (.013)	0.914 (.005)	**0.747** (.014)	**0.901** (.006)
RAIL-KD	0.548 (.017)	**0.915** (.003)	0.742 (.010)	0.899 (.004)
MATE-KD	**0.566** (.014)	0.906 (.004)	0.740 (.010)	0.900 (.044)
CILDA	0.556 (.012)	0.914 (.004)	0.742 (.017)	0.895 (.018)

先行研究では CILDA が MATE-KD を超える性能を示したと報告されているが，どのモデルを使った実験でも，そのような結果を再現できなかった．そのため，MATE-KD と CILDA の違いである L_{CRD} の利用は有効でなかったと考えられる．提案手法の CILDA+minILD は L_{CRD} を CILDA の最小化ステップでも用いるという手法であるため，L_{CRD} が有効な損失関数でなかったことが CILDA+minILD の性能が低かった原因と考えられる．

6 終わりに

本研究では，CILDA の最小化ステップに中間層出力の利用と対照学習を適用した CILDA+minILD を提案した．しかし，従来の手法である MATE-KD よりも低い性能となり，その原因が CILDA の性能が低いことにあると結論づけた．今後は，本研究で扱わなかった GLUE の他のタスクでも同様の結果が得られるかの検証，CILDA の性能が MATE-KD よりも低かった原因の究明，本研究で最も高い性能を発揮した MATE-KD を改良することによる知識蒸留の性能の向上などを行っていきたい．

参考文献

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] 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.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, 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, and Dario Amodei. Language models are few-shot learners. **arXiv**, Vol. abs/2005.14165, , 2020.

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. **arXiv**, Vol. abs/2204.02311, , 2022.

[5] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. **arXiv**, Vol. abs/1503.02531, , 2015.

[6] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In **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.

[7] 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.

[8] 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.

[9] Ahmad Rashid, Vasileios Lioutas, and Mehdi Rezagholizadeh. MATE-KD: Masked adversarial TExt, a companion to knowledge distillation. In **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. 1062–1071, Online, August 2021. Association for Computational Linguistics.

[10] Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. Contrastive distillation on intermediate representations for language model compression. In **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 498–508, Online, November 2020. Association for Computational Linguistics.

[11] Md Akmal Haidar, Mehdi Rezagholizadeh, Abbas Ghaddar, Khalil Bibi, Phillippe Langlais, and Pascal Poupart. CILDA: Contrastive data augmentation using intermediate layer knowledge distillation. In **Proceedings of the 29th International Conference on Computational Linguistics**, pp. 4707–4713, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

[12] 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.

[13] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. **arXiv**, Vol. abs/1910.01108, , 2019.

[14] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics.

A　ハイパーパラメータ

ハイパーパラメータ探索の結果，採用したハイパーパラメータを表 5，6，7 に示す．数字が 1 つの場合は学習率，2 つある場合は上が学習率，下がバッチサイズである．CILDA 検証の行は 5 章の実験のものを表す．

表 5　RoBERTa の実験

手法 ＼ タスク	CoLA	MRPC	RTE	STS-B
w/o KD	1e-05	1e-05	5e-05	1e-04
KD	2e-05	2e-05	1e-05	1e-04
RAIL-KD	2e-05	5e-05	1e-04	1e-04
MATE-KD	2e-05	5e-05	1e-05	1e-04
CILDA	2e-05	5e-05	2e-05	1e-04
CILDA+minILD	2e-05	2e-05	2e-05	1e-04
CILDA 検証	4e-06 8	1e-05 8	1e-05 8	2e-05 32

表 6　BERT の実験

手法 ＼ タスク	CoLA	MRPC	RTE	STS-B
w/o KD	2e-05	2e-05	5e-05	1e-04
KD	5e-05	2e-05	1e-04	1e-04
RAIL-KD	5e-05	2e-05	2e-05	5e-05
MATE-KD	1e-04	2e-05	1e-05	1e-04
CILDA 検証	2e-05 8	1e-05 8	1e-05 16	1e-05 8

表 7　BART の実験

手法 ＼ タスク	CoLA	MRPC	RTE	STS-B
w/o KD	2e-05	1e-04	5e-05	1e-04
KD	5e-05	2e-05	2e-05	1e-04
RAIL-KD	5e-05	2e-05	2e-05	1e-04
MATE-KD	5e-05	2e-05	5e-05	5e-05
CILDA 検証	2e-05 16	1e-05 8	2e-05 8	2e-05 16

B　教師モデルの性能

RoBERTa，BERT，BART の実験において，実際に使用した教師モデルの性能を表 8 に示す．

表 8　教師モデルの性能

モデル ＼ タスク	CoLA	MRPC	RTE	STS-B
RoBERTa	0.658	0.922	0.783	0.913
BERT	0.646	0.913	0.700	0.893
BART	0.643	0.929	0.874	0.917