Aligning Sizes of Intermediate Layers by LoRA Adapter for Knowledge Distillation

Takeshi Suzuki
Institute of Science Tokyo
suzuki.t.dp@m.titech.ac.jp

Hiroaki Yamada
Institute of Science Tokyo
yamada@comp.isct.ac.jp

Takenobu Tokunaga
Institute of Science Tokyo
take@c.titech.ac.jp

Abstract

Intermediate Layer Distillation (ILD) is a variant of Knowledge Distillation (KD), a method for compressing neural networks. ILD requires mapping to align the intermediate layer sizes of the teacher and student models to compute the loss function in training, while this mapping is not used during inference. This inconsistency may reduce the effectiveness of learning in intermediate layers. In this study, we propose LoRAILD, which uses LoRA adapters to eliminate the inconsistency. However, our experimental results show that LoRAILD does not outperform existing methods. Furthermore, contrary to previous studies, we observe that conventional ILD does not outperform vanilla KD. Our analysis of the distilled models' intermediate layers suggests that ILD does not improve language models' performance.

1 Introduction

The LLM's performance is rapidly improving on various natural language processing (NLP) tasks at the cost of the huge parameter size, resulting in enormous computational costs. Therefore, reducing the parameter size while retaining the model's performance is an important research topic.

Knowledge distillation (KD) (Buciluǎ et al., 2006; Hinton et al., 2015) is one of the model compression methods. KD employs two models: a teacher model and a student model. The teacher model is already trained for a specific task. The teacher's output serves as soft labels that guide the student model that mimics the teacher's behavior.

While KD usually uses the output of the teacher, intermediate layer distillation (ILD) (Romero et al., 2015) uses the information of intermediate layers as well; ILD has been claimed to be superior to the vanilla KD in previous studies (Sun et al., 2019; Passban et al., 2021; Haidar et al., 2022).

In this study, we introduce LoRAILD, which is designed to improve the conventional ILD by employing the LoRA (Hu et al., 2021) adapter in aligning sizes of intermediate layers between the teacher and the student. We evaluate the performance of LoRAILD through empirical comparisons with conventional KD baselines. In general, we find that our LoRAILD does not outperform the conventional ILD baselines, and even the conventional ILD is not necessarily superior to the vanilla KD, at least in our experimental settings.

2 Background

2.1 Knowledge Distillation

In KD (Buciluǎ et al., 2006; Hinton et al., 2015), a student model is trained using a loss function based on the difference between its output and the teacher's output, as well as a loss function calculating errors against gold labels. KD's combined loss function L is defined as (1).

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

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

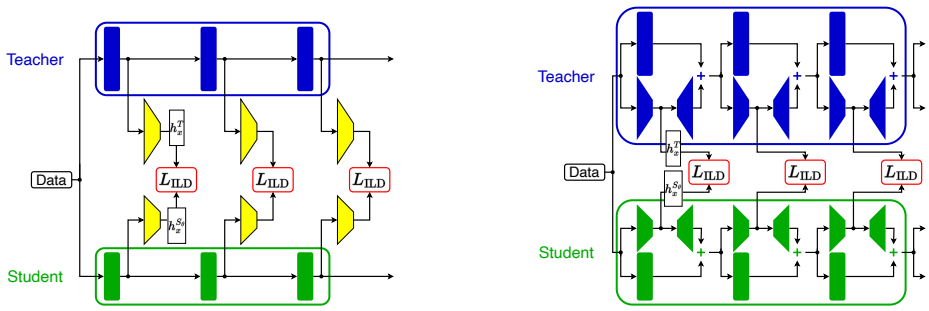
where L_{CE} is the cross-entropy loss, $\mathrm{KL}(\cdot, \cdot)$ is the KL divergence. $\mathrm{Teacher}(X)$ and $\mathrm{Student}(X)$ are the probability distributions obtained as outputs when X is input to the teacher and the student, respectively. In training the student, the parameters of the teacher are fixed.

2.2 Intermedite Layer Distillation

Intermediate layer distillation (ILD) (Romero et al., 2015) is a variant of KD. ILD uses not only the teacher and student outputs but also the information of their intermediate layers. ILD requires the alignment of the number of layers and layer sizes between the student and the teacher.

When the number of layers between two models differs, adjustment is necessary, which is the focal topic of past ILD research. PKD (Sun et al., 2019) heuristically selects the same number of layers from the teacher as those in the student. In

The Sixth Workshop on Insights from Negative Results in NLP, pages 100–105



(a) Previous ILD (b) LoRAILD

Figure 1: Overview of LoRAILD (blue: teacher, green: student, yellow: linear mapping)

PKD, the selected layers are consistent through the student training. ALP-KD (Passban et al., 2021) constructs groups of teacher layers, and each group corresponds to a layer in the student. The outputs from layers in a group are aggregated by calculating the weighted average. RAIL-KD (Haidar et al., 2022) dynamically selects layers of the teacher at random in each epoch.

When the intermediate layer size is inconsistent between the teacher and the student, the size must be aligned to compute the loss function. Previous studies (Romero et al., 2015; Haidar et al., 2022) employed some mapping methods for the alignment. FitNets (Romero et al., 2015) employed convolutional regressors, and RAIL-KD (Haidar et al., 2022) used linear layers for mapping. During student training, these mappings are trained alongside all the other parameters of the student. However, the mappings are used only in the student training phase, so the model's structure is inconsistent between training and inference.

This inconsistency might degrade the effectiveness of ILD. If only the mappings are tuned very well with a task while the intermediate layers of the student might not develop good features, removing the mappings in inference will degrade the student's performance.

3 Method

3.1 LoRAILD

To tackle the problem of structure inconsistency, we propose LoRAILD, which employs LoRA adapters replacing the conventional mappings. As LoRAILD utilizes the LoRA adapters both in the training and the inference phase by design, the discrepancy of model structure no longer exists.

LoRA (Hu et al., 2021) is one of the methods to reduce the computational cost during training by assigning two low-rank matrices of size $\mathbb{R}^{i\times r}$ and $\mathbb{R}^{r\times o}$ to a certain module in the model and only training them. i and o are the number of input and output sizes of the module to be assigned, respectively. Since low-rank matrices have the same architecture as linear mapping and the r value, which is the output size of the input-side matrix, can be set manually, we can convert the intermediate layer size of the teacher and the student into the same by setting the r properly. Thus, they can be used as a substitute for conventional linear mappings. Moreover, these matrices are used in the inference phase. Therefore, we can maintain linear mappings by using low-rank matrices. Figure 1b shows the outline of LoRAILD. In the conventional ILD, the loss is calculated using the output of a linear layer (yellow part of Figure 1a). In contrast, in LoRAILD, the loss function is calculated using the output of the input-side matrix of the LoRA adapter (trapezoidal part of Figure 1b).

The loss function is given by equation (4).

$$L_{\mathrm{ILD}} = \frac{1}{N}\sum_{x\in batch}\left\|\frac{h_x^T}{\|h_x^T\|_2}-\frac{h_x^{S_\theta}}{\|h_x^{S_\theta}\|_2}\right\|_2^2 \quad (3)$$

$$L = \lambda_1 L_{\mathrm{CE}} + \lambda_2 L_{\mathrm{KD}} + \lambda_3 L_{\mathrm{ILD}} \quad (4)$$

N is the batch size, h_x^* is the output of the input-side matrix of the LoRA adapter for x. The outputs of each layer are concatenated to compute the loss function. The T and S_θ denote the teacher and the student, respectively. The L_{CE} and L_{KD} are the same as in equation (1). This is the same as the loss function used in previous ILD methods (Sun et al., 2019; Haidar et al., 2022).

3.2 Alignment of layers

While LoRA adaptors address the size alignment issue, we still need to fill the gap between the num-

ber of layers in the teacher and the student. In this study, we employ the following three methods.

Fixed Always select the same layer during student training.

Average Allocate some layers of the teacher to each layer of the student, and average the output of the allocated layers.

Random Randomly select a layer of the teacher for each layer of the student. The sequence of the selected layers is preserved. The random selection is performed at each mini-batch (Random step) or each epoch (Random epoch).

Fixed, Average, and Random are almost the same as those used in the previous studies: PKD (Sun et al., 2019), ALP-KD (Passban et al., 2021), and RAIL-KD (Haidar et al., 2022). Unlike ALP-KD, h is simply averaged, and a weighted average is not used in the Average method.

Appendix A shows the alignment patterns between the student and teacher layers for Fixed and Average.

3.3 Curriculum Learning

LoRAILD did not perform well in our preliminary trials, where we observed that L_{ILD} decreased first and L_{CE} did not decrease well. In order to make sure L_{CE} to decrease, we introduced curriculum learning in which only L_{CE} and L_{KD} are trained first, and L_{ILD} is added to L later.

4 Experiment

4.1 Experimantal Settings

In our experiments, we use RoBERTa-large (Liu et al., 2019) for the teacher and DistilRoBERTa-base (Sanh et al., 2019) for the student. LoRA adapters are added to both models. The teacher trains only LoRA adapters, and the student trains both LoRA and the original model.

The dataset used in this experiment consists of six tasks from the GLUE (Wang et al., 2018) benchmark: CoLA, MRPC, QNLI, RTE, SST-2, and STS-B. Since the GLUE benchmark does not publish the gold labels for the test set, we use the original validation set as a test set, 10 percent of the original training set for validation, and the remaining 90 percent for training. The evaluation metrics are the Matthews correlation coefficient for CoLA, the F1 score for MRPC, the Pearson correlation coefficient for STS-B, and accuracy for the others.

Baselines are the finetuned student without KD (w/o KD), the model with normal KD (Vanilla KD), RAIL-KD ($\mathrm{RAIL\text{-}KD_c}$, $\mathrm{RAIL\text{-}KD_l}$), and the model $\mathrm{RAIL\text{-}KD_c}$ with curriculum learning (Curriculum); curriculum learning was not used in the original RAIL-KD paper (Haidar et al., 2022). In $\mathrm{RAIL\text{-}KD_l}$, L_{ILD} is computed per layer, while $\mathrm{RAIL\text{-}KD_c}$ uses concatenated intermediate outputs. The hyperparameters are listed in Appendix C.

All reported metrics are the average of five runs, and we conduct one-tailed permutation tests at a significance level of 2.5%.

4.2 Result

	CoLA	MRPC	QNLI	RTE	SST-2	STS-B
Teacher	0.594	0.884	0.947	0.798	0.959	0.912
w/o KD	0.567	0.876	0.906	0.676	0.919	0.881
Vanilla KD	0.566	0.874	0.912	0.622	0.925	0.883
RAIL-KD						
$\mathrm{RAIL\text{-}KD_c}$	0.568	**0.892**	0.916	0.658	**0.929**	0.882
$\mathrm{RAIL\text{-}KD_l}$	0.585	0.882	0.907	0.522	0.907	0.886
Curriculum	0.568	0.889	0.912	**0.677**	0.928	0.881
LoRAILD						
Fixed	0.565	0.880	0.912	0.651	0.922	0.880
Average	**0.596**	0.874	0.913	0.656	0.921	**0.886**
Random step	0.592	0.846	**0.916**	0.637	0.920	0.881
Random epoch	0.573	0.886	0.911	0.659	0.915	0.882

Table 1: Results on test set

Table 1 shows the result. Bold figures indicate the best performance for each task. They are significantly higher than underlined ones. None of the LoRAILD-based models showed clear improvement from the baselines. Fixed and Random epoch did not outperform previous methods in any task. Average showed the best scores in CoLA and STS-B; however, we could not confirm their statistical significance. Although Random step achieved the best score in QNLI among all the models, its improvement is subtle and not statistically distinguishable from $\mathrm{RAIL\text{-}KD_c}$. Given the little improvement in LoRAILD, the LoRA adapters in LoRAILD could not perform as we expected, and they might disturb intermediate layers' learning.

Moreover, none of the RAIL-KD models outperforms w/o KD and Vanilla KD. These results contradict the previous study's outcome; they reported that RAIL-KD (Haidar et al., 2022) outperformed w/o KD and Vanilla KD at all tasks in their

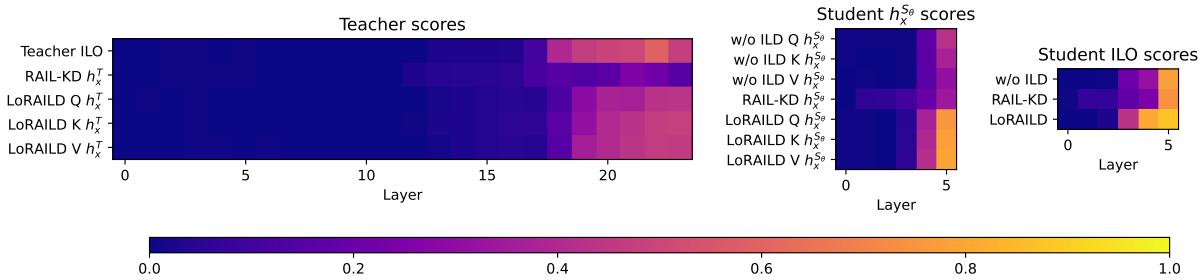


Figure 2: Clustering scores (ILO means intermediate layer output)
left: Teacher, middle: Student's $h_x^{S_\theta}$, right: Students' intermediate layer output

experiment. The lower performance of RAIL-KD than Vanilla KD aligns with our hypothesis that removing the linear mapping during inference may degrade the effectiveness of ILD.

5 Analysis

5.1 Analysis method

To examine how the intermediate layers of the models embed features in RAIL-KD and LoRAILD, we conduct a cluster analysis of features used in loss calculation and intermediate layer outputs. For both the teacher and the student of LoRAILD, RAIL-KD, and w/o ILD ($\lambda_3 = 0$), we obtain h_x^* and direct outputs from intermediate layers, where the models are fed with our training set. Note that h_x^T in this section is the value before layer alignment. We cluster the obtained vectors by k-means (MacQueen, 1967) clustering ($k = 2$[1]). We use the same tasks as our experiment except for STS-B, which is regression.

We evaluate the clusters by calculating the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985; Steinley, 2004) against gold clusters that are constructed according to the gold labels.

The higher ARI suggests the student acquires better representation for solving the task through the training. A higher ARI for the teacher means its intermediate layers provide more useful information to train the student.

5.2 Analysis result

Figure 2 illustrates the ARI scores. The left and middle matrices indicate the scores calculated for h_x^T and $h_x^{S_\theta}$ from the teacher and the student, respectively. The right matrix indicates the scores for the outputs from the student's intermediate layer. Each cell corresponds to an aggregated score across the tasks by averaging. In BERT, the latter layers process the semantics of sentences (Tenney et al., 2019; Jawahar et al., 2019). As the tasks used in this analysis, except for the CoLA, concern the semantics of the sentences, the latter layers play important roles in these tasks.

In the left figure, h_x^T in RAIL-KD indicates a lower score in the latter layers (≥ 18) compared to the original intermediate layer output of the teacher (Teacher ILO), suggesting information useful for the tasks is not provided to the student. On the other hand, LoRAILD scored higher in the latter layers, indicating that the teacher conveys more useful information to the student. We initially hypothesized that removing the linear mapping could degrade the performance because it also removes information learned in the linear mapping. However, we found that the linear mapping in RAIL-KD does not actually learn information about the task; rather, it degrades the quality of the teacher signal from Teacher ILO.

In the middle and right figures, RAIL-KD does not show improvement from w/o ILD in the latter layers (≥ 4), while LoRAILD successfully does.

This result indicates that our LoRAILD provides a better method to align the intermediate layers between the student and the teacher.

6 Conclusion

Our experimental result (Table 1) showed that ILD did not improve performance from Vanilla KD regardless of using the intermediate layers (LoRAILD) or not (RAIL-KD). Although our analysis revealed that our LoRAILD provided better alignment for intermediate layers, this improvement did not improve the downstream tasks. Thus, we conclude that the current ILD approach has little impact on the performance of the language model.

[1]All the tasks employed in our analysis are binary classification.

7 Limitation

Different experimental settings from the previous studies Although our experimental results report negative results not aligned with those reported in RAIL-KD (Haidar et al., 2022), our finding does not directly reject the previous results. Instead, our results suggest that ILD might not be generalizable to different settings as initially expected.

The RAIL-KD and other baselines used in our experiment employ different base models, datasets, and hyperparameters from their original ones, so our experiment does not completely replicate their original settings.

We acknowledge that our teacher models are different from the previous studies. This is because the teacher models used in the previous study are not publicly available. Also, the architecture of the teacher model we used differs from that of the previous studies because of the addition of the LoRA adapter.

As for the data set, as described in section 4.1, our dataset split is different. 90% of the GLUE train set was used as our train set, the remaining 10% of the GLUE train set as a validation set, and the original validation set as a test set.

Due to the different conditions described above, we had to perform a hyperparameter search to find the optimal values for our settings.

Scope of our experiments Our study addressed a scenario in which the intermediate layer size of the teacher and student models is different, and we always have to transform the outputs and align their sizes. Thus, the intermediate layer output from the teacher model was not directly used as the teacher signal for the student model in any of our experiments. We acknowledge that we have no conclusion for a case where the intermediate layer size is the same between the teacher and student and ILD can use the direct outputs from the teacher's intermediate layers without any alignment. That case is outside of the scope of this paper.

References

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, page 535–541, New York, NY, USA. Association for Computing Machinery.

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

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arxiv preprint, arXiv:2106.09685.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. Journal of Classification, 2(1):193–218.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. arXiv, abs/1907.11692.

J. B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281–297. University of California Press.

Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. 2021. Alp-kd: Attention-based layer projection for knowledge distillation. Proceedings of the AAAI Conference on Artificial Intelligence, 35(15):13657–13665.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. FitNets: Hints for Thin Deep Nets. arXiv preprints, arXiv:1412.6550.

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

Douglas Steinley. 2004. Properties of the Hubert-Arabie adjusted rand index. Psychol Methods, 9(3):386–396.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. 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), pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. 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, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

A Alignment

Fixed	
Student	Teacher
0	3
1	7
2	11
3	15
4	19
5	23

Average	
Student	Teacher
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

Table 2: Alignment

Table 2 shows the alignment of the student and teacher models for the alignment methods Fixed and Average. L_{ILD} was computed using the vector of concatenated $h_x^{S_\theta}$ from layer 0 to layer 5 of the student model, and the vector of concatenated h_x^T from layers 3, 7, ..., and 23 of the teacher model.

In Average, the concatenation of averages of h_x^T from layer 0, 1, 2, and 3, ... , average of h_x^T from layer 20, 21, 22, and 23 was used to calculate L_{ILD}.

B Results on Validation Set

Table 3 shows results on validation set as well as test set.

	CoLA	MRPC	QNLI	RTE	SST-2	STS-B
Teacher	0.673	0.894	0.935	0.779	0.960	0.920
w/o KD	0.615	0.872	0.894	0.692	0.947	0.896
Vanilla KD	0.635	0.888	0.903	0.716	0.954	0.916
RAIL-KD						
RAIL-KD$_c$	0.625	0.902	0.905	0.716	0.954	0.917
RAIL-KD$_l$	0.633	0.899	0.899	0.597	0.952	0.916
Curriculum	**0.650**	**0.903**	**0.907**	**0.748**	0.955	**0.920**
LoRAILD						
Fixed	0.628	0.883	0.905	0.688	0.955	0.912
Average	0.637	0.871	0.904	0.717	0.954	0.919
Random step	0.633	0.848	0.904	0.680	**0.956**	0.913
Random epoch	0.627	0.888	0.904	0.726	0.955	0.912

Table 3: Result for valitation set

C Hyperparameters

The hyperparameters used in the experiments are shown in Table 4. In the middle part of the table, epochs where we initiated curriculum learning are illustrated. In the curriculum learning, $(\lambda_1, \lambda_2, \lambda_3)$ is varied from the initial to the final state; only the RTE task experimented with two different final states.

Learning rate	{1, 2, 5}e-{4, 5}
r (output size of mapping)	32
Epoch (Teacher model)	4
Epoch (Student model)	20

Curriculum Learning	
Tasks	Epoch to start
CoLA	5-10
MRPC	5-10
QNLI	5-10
RTE	5-10
SST-2	2-4
STS-B	2-4

States	$(\lambda_1, \lambda_2, \lambda_3)$
Initial state	$(0.5, 0.5, 0)$
Final state	$(0.333, 0.333, 0.333)$
Final state(RTE only)	$(0.5, 0, 0.5)$

Table 4: Hyperparameters