RAIL-KD: RAndom Intermediate Layer Mapping for Knowledge Distillation

Md Akmal Haidar[1*] **Nithin Anchuri**[1,2*] **Mehdi Rezagholizadeh**[1]
Abbas Ghaddar[1] **Philippe Langlais**[2] **Pascal Poupart**[3]
[1] Huawei Noah's Ark Lab
[2] RALI/DIRO, Université de Montréal, Canada
[3] David R. Cheriton School of Computer Science, University of Waterloo
{mehdi.rezagholizadeh,abbas.ghaddar}@huawei.com
felipe@iro.umontreal.ca, ppoupart@uwaterloo.ca

Abstract

Intermediate layer knowledge distillation (KD) can improve the standard KD technique (which only targets the output of teacher and student models) especially over large pre-trained language models. However, intermediate layer distillation suffers from excessive computational burdens and engineering efforts required for setting up a proper layer mapping. To address these problems, we propose a RAndom Intermediate Layer Knowledge Distillation (RAIL-KD) approach in which, intermediate layers from the teacher model are selected randomly to be distilled into the intermediate layers of the student model. This randomized selection enforces that all teacher layers are taken into account in the training process, while reducing the computational cost of intermediate layer distillation. Also, we show that it acts as a regularizer for improving the generalizability of the student model. We perform extensive experiments on GLUE tasks as well as on out-of-domain test sets. We show that our proposed RAIL-KD approach outperforms other state-of-the-art intermediate layer KD methods considerably in both performance and training-time.

1 Introduction

Pre-trained Language Models (PLMs), such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2020) and XLNet (Yang et al., 2019) have shown remarkable abilities to match and even surpass human performances on many Natural Languages Understanding (NLU) tasks (Rajpurkar et al., 2018; Wang et al., 2018, 2019). However, the deployment of these models in real world applications (e.g. edge devices) come with challenges, mainly due to large model size and inference time.

In this regard, several model compression techniques such as quantization (Shen et al., 2019; Zafrir et al., 2019; Kumar et al., 2022; Prato et al., 2020), pruning (Guo et al., 2019; Gordon et al., 2020; Michel et al., 2019),matrix factorization (Tahaei et al., 2021; Lioutas et al., 2020), optimizing the Transformer architecture (Fan et al., 2019; Wu et al., 2020b; Lu et al., 2020), and knowledge distillation (Sanh et al., 2019a; Jiao et al., 2019; Sun et al., 2020b; Wang et al., 2020a; Rashid et al., 2021; Passban et al., 2021; Jafari et al., 2021; Rezagholizadeh et al., 2021) have been developed to reduce the model size and latency, while maintaining comparable performance to the original model.

KD, which is the main focus of this work, is a neural model compression approach that involves training a small *student* model with the guidance of a large pre-trained *teacher* model. In the original KD technique (Buciluǎ et al., 2006; Hinton et al., 2014; Turc et al., 2019), the teacher output predictions are used as soft labels for supervising the training of the student. There has been several attempts in the literature to reduce the teacher-student performance gap by leveraging data augmentation (Fu et al., 2020; Li et al., 2021; Jiao et al., 2019; Kamalloo et al., 2021, 2022), adversarial training (Zaharia et al., 2021; Rashid et al., 2020, 2021), and intermediate layer distillation (ILD) (Wang et al., 2020b,a; Ji et al., 2021; Passban et al., 2021).

When it comes to BERT compression, ILD leads to clear gains in performances (Sanh et al., 2019a; Jiao et al., 2019; Wang et al., 2020a) due to its ability to enhance the knowledge transfer beyond logits matching. This is done by mapping intermediate layer representations of both models to a common space[1], and then matching them via regression (Sun et al., 2019) or cosine similarity (Sanh et al., 2019a) losses. One major problem with ILD is the absence of an appropriate strategy to select layers to be matched on both sides, reacting to the *skip*

[*] Work done while at Huawei.

[1] In some cases, the representations are directly matched if the teacher and student have the same hidden size.

Findings of the Association for Computational Linguistics: NAACL 2022, pages 1389 - 1400

Model	Layer Mapping	Complexity	Limitation
PKD (Sun et al., 2019)	Extra Hyperparameter	$O(m)$	Extensive Search
CKD (Wu et al., 2020a)	Extra Hyperparameter	$O(m)$	Extensive Search
ALP-KD (Passban et al., 2021)	Attention	$O(m \times n)$	Slow Training time
CoDIR (Sun et al., 2020a)	Contrastive Learning	$O(K \times m)$	Slow Training time
RAIL-KD^{l} (our)	Random Selection	$O(m)$	-
RAIL-KD^{c} (our)	Random Selection	$O(m)$	-

Table 1: Main characteristics and limitation of different approaches that tackle the *skip and search problem*. Concat indicates if the approach support concatenated layers distillation. n and m refer to the teacher and student layer number respectively, while K refers to number of negative samples of CoDIR.

and search problem (Passban et al., 2021). Some solutions in the literature mostly rely on layer combination (Wu et al., 2020a), attention-based layer projection (Passban et al., 2021) and contrastive learning (Sun et al., 2020a). While these solutions are all effective to some extent, to the best of our knowledge, there is no work in the literature doing a comprehensive evaluation of these techniques in terms of both efficiency and performance.

A case in point is that the aforementioned solutions to the layer skip and search problem do not scale to very deep networks. We propose RAIL-KD (RAndom Intermediate Layer KD), a simple yet effective method for intermediate layer mapping which randomly selects k out of n intermediate layers of the teacher at each epoch to be distilled to the corresponding student layers. Since the layer selection is done randomly, all the intermediate layers of the teacher will have a chance to be selected for distillation. Our method adds no computational cost to the training, still outperforming all aforementioned methods on the GLUE benchmark (Wang et al., 2018). Moreover, we observe larger gains when distilling from large teacher models, as well as when student models are evaluated on out-of-domain datasets. Last, we report the results on 5 random seeds in order to verify the contribution of the random selection process, thus making the comparison fair with previous methods. The main contributions of our paper are as follows:

- We introduce RAIL-KD, a more efficient and scalable intermediate layer distillation approach.
- To the best of our knowledge, we are the first to perform a comprehensive study of the ILD techniques in terms of both efficiency and performance.
- We consider the distillation of models such as BERT and RoBERTa, and compare different up-to-date distillation techniques on out-of-domain test sets.

2 Related Work

In recent years, a wide range of methods have tried to expand knowledge transfer of transformer-based (Vaswani et al., 2017) NLU models beyond logits matching. DistillBERT (Sanh et al., 2019a) added a cosine similarity loss between teacher and student embeddings layers. TinyBERT (Jiao et al., 2019), MobileBERT (Sun et al., 2020b), and MiniLM (Wang et al., 2020b) matched the intermediate layers representations and self-attention distributions of the teacher and the student.

In PKD, Sun et al. (2019) used deterministic mapping strategies to distill a 12-layer BERT teacher to a 6-layer BERT student. PKD-LAST and PKD-SKIP refer to matching layers $\{1-5\}$ of the student with layers $\{7-11\}$ and $\{2,4,6,8,10\}$ of the teacher respectively. However, these works ignored the impact of layer selection, as they used a fixed layer-wise mapping.[2]

Researchers have found that tuning the layer mapping scheme can significantly improve the performance of ILD techniques (Sun et al., 2019). Nevertheless, finding the optimal mapping can be challenging, which is referred to as the *layer skip and search problems* by Passban et al. (2021). To address the layer skip problem, CKD (Wu et al., 2020a) is built on top of PKD by partitioning all the intermediate layers of the teacher to the number of student layers. Then, the combined representation of the layers of each partition is distilled into a number of subset corresponding to the number of student layers. However, finding the optimal

[2]e.g. matching the first (or last) k layers of the student with their corresponding teacher layers.

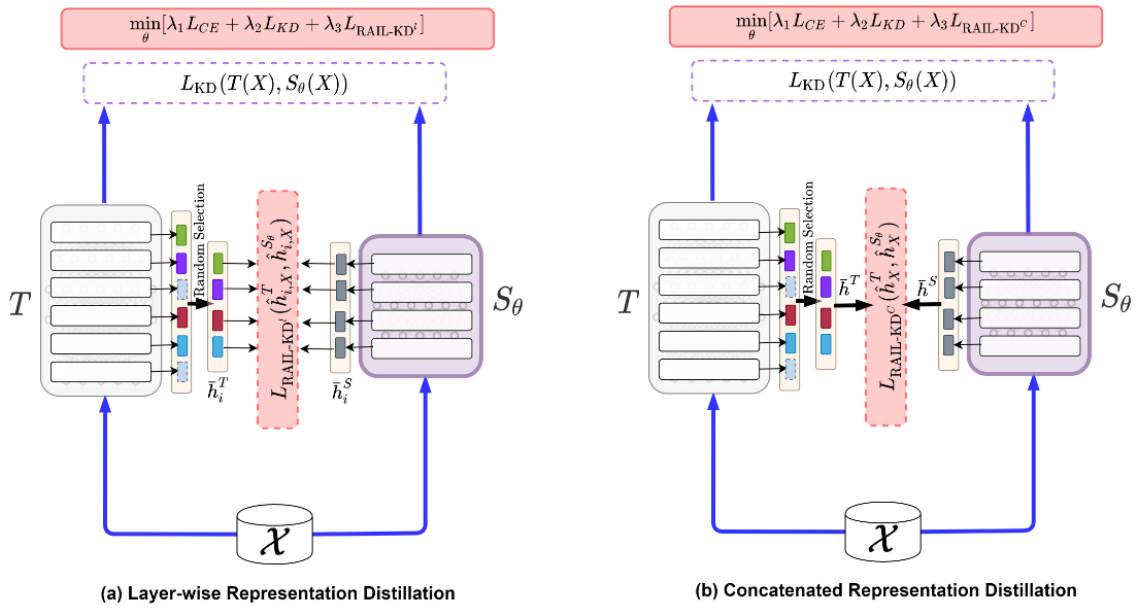

Figure 1: Proposed RAIL-KD technique for efficient intermediate layer distillation. (a) This version shows a layer-wise projection which is indicated as RAIL-KD^l in the paper. (b) This variant named RAIL-KD^c, concatenates the intermediate representations of each network before distillation.

partitioning scheme requires running exhaustive experiments. Given teacher and student BERT models with n and m layers respectively (where $n >> m$), it is not trivial to choose the teacher layers that can be incorporated in the distillation process and how we should map them to the student layers (*search*).

ALP-KD (Passban et al., 2021) overcomes this issue by computing attention weights between each student layer and all the intermediate layers of the teacher. The learned attention weights for each student layer are used to obtain a weighted representation of all teacher layers. Although ALP-KD has shown promising results on 12-layer BERT-based compression, attending to all layers of the teacher adds considerable computational overhead to the training phase. This can become computationally prohibitive when scaling to very large models such as RoBERTa-large (Liu et al., 2020) or GPT-2 (Radford et al., 2019). Alternatively, CoDIR (Sun et al., 2020a) exploited contrastive learning (Tian et al., 2019) to perform intermediate layers matching between the teacher and the student models with no deterministic mapping. Similar to ALP-KD, this approach also requires excessive training time due to the contrastive loss calculation and the use of negative samples from a memory bank.

Table 1 summarizes the main characteristics of the existing state-of-the-art intermediate layer distillation techniques (PKD, CKD, and CoDIR) used for pre-trained language models compared with our proposed RAIL-KD. First, PKD and CKD treat the mapping as an extra hyperparameter that requires extensive experiments in order to find the optimal mapping. Second, ALP-KD (Passban et al., 2021) and CoDIR (Sun et al., 2020a) use the attention mechanism and contrastive learning respectively to address the issue, but at the expense of extra computational cost.

Our proposed RAIL-KD method does not add any computational cost to the distillation process, while empirically outperforming previous methods. For instance, RAIL-KD is roughly two-times faster than CoDIR in a 24 to 6 layer compression. In addition, it does not require extensive experiments to find the optimal mapping scheme. In this work, we position ourselves to works that tackle the *skip and search problem*.[3] In other words, we don't compare with works like TinyBERT (Jiao et al., 2019) or MiniLM (Wang et al., 2020b), which use extra losses like self-attention distribution matching. However, we expect that these methods, as well as state-of-the-art (Rashid et al., 2021; He et al., 2021) ones can take full advantage of RAIL-KD, since they use a deterministic layer mapping scheme.

[3]This only concerns works that perform intermediate layer distillation

3 RAIL-KD

The RAIL-KD method is sketched in Figure 1. RAIL-KD transfers intermediate knowledge of a pre-trained teacher T with n intermediate layers to a student model S_θ with m intermediate layers. In contrast to traditional intermediate layer distillation techniques which keep the selected layers of the teacher for distillation fixed during training, in RAIL-KD, at each epoch, a few intermediate layers from the teacher model are selected randomly for distillation. Here for simplicity, we set the number of selected intermediate layers of the teacher model equal to that of the student model.

Let (X, y) denote a training sample $X = (x_0, \cdots, x_{L-1})$ which is a sequence of L tokens and y its corresponding label. In Figure 1, our Random Selection operator is applied to the intermediate layers of the teacher to randomly select m out of n layers. The intermediate layer representations of the m selected layers of the teacher and the student model corresponding to the X input can be described as $\mathcal{H}_X^T = \{H_{1,X}^T, \cdots, H_{m,X}^T\}$ and $\mathcal{H}_X^{S_\theta} = \{H_{1,X}^{S_\theta}, \cdots, H_{m,X}^{S_\theta}\}$ respectively, where $H_{i,X}^T = \cup_{k=0}^{L-1}\{h_{i,x_k}^T\} \in R^{L\times d_1}$ and $H_{i,X}^{S_\theta} = \cup_{k=0}^{L-1}\{h_{i,x_k}^{S_\theta}\} \in R^{L\times d_2}$. Here, d_1 and d_2 indicate the hidden dimension of the layers of the teacher and the student models respectively. To obtain $H_{i,X}^T$ and $H_{j,X}^S$, we need to find the individual representation of each token x_k at each layer i, which is indicated as h_{i,x_k}^T and h_{i,x_k}^S for the teacher and student networks respectively.

At this stage, we need to obtain an aggregated representation of the sequence X at each layer. In this regard, one can either use the <CLS> token representation or use the mean-pooling of the sequence representations of the layer. Since in Sun et al. (2020a), the mean-pooling representation shows better results, we adopt it to compute the sentence representation of each layer. Mean-pooling is a row-wise average over $H_{i,X}^T$, $H_{i,X}^{S_\theta}$ to get $\bar{h}_{i,X}^T \in R^{d_1}$ $\bar{h}_{i,X}^{S_\theta} \in R^{d_2}$ (Sun et al., 2020a):

$$\bar{h}_{i,X}^T = \frac{1}{L}\sum_{k=0}^{L-1} h_{i,x_k}^T \;\; ; \;\; \bar{h}_{i,X}^{S_\theta} = \frac{1}{L}\sum_{k=0}^{L-1} h_{i,x_k}^{S_\theta} \quad (1)$$

After obtaining aggregated layer representations for both the student and teacher networks, our RAIL-KD proposal is to randomly select m layer representations from the teacher through training to perform the intermediate layer distillation (ILD). RAIL-KD does ILD in two different forms: using layer-wise distillation (see Fig. 1(a)) or by concatenating layer representations (see Fig. 1(b)) which are described in the following two sub-sections.

3.1 Layer-wise RAIL-KD

In this setting, the representations $\bar{h}_{i,X}^T \in R^{d_1}$ and $\bar{h}_{i,X}^{S_\theta} \in R^{d_2}$ are projected into a same-size lower-dimensional space $\hat{h}_{i,X}^T, \hat{h}_{i,X}^{S_\theta} \in R^u$ using $(d_1 \times u)$ and $(d_2 \times u)$ linear mappings respectively. Assume that the set $\mathcal{A} = \{a_\kappa | a_\kappa \sim \{1, 2, ..., n\}, 1 \leq \kappa \leq m\}$ contains indices of selected m layers from the teacher, then to calculate the layer-wise loss we have:

$$L_{\text{RAIL-KD}^l} = \sum_{X\in\mathcal{X}}\sum_{i\in\mathcal{A}} \alpha_i \left(|| \frac{\hat{h}_{i,X}^T}{||\hat{h}_{i,X}^T||_2} - \frac{\hat{h}_{i,X}^{S_\theta}}{||\hat{h}_{i,X}^{S_\theta}||_2} ||_2^2 \right) \quad (2)$$

where $\mathcal{X}$ denotes the training set, and α_i is a hyper-parameter to assign a custom weights to the layer-wise distillation loss. It is worth mentioning that in our experiments we set $\alpha_i = 1$.

3.2 Concatenated RAIL-KD

In this setting, intermediate layer representations are concatenated and then distilled: $\bar{h}_X^T = [\bar{h}_{i,X}^T]_{i\in\mathcal{A}}$, $\bar{h}_X^{S_\theta} = [\bar{h}_{j,X}^{S_\theta}]_{j=1}^m$ which are further mapped into the same lower-dimensional space $\hat{h}_X^T, \hat{h}_X^{S_\theta} \in R^u$ using $(md_1 \times u)$ and $(md_2 \times u)$ linear mappings to calculate the concatenated distillation loss.

$$L_{\text{RAIL-KD}^c} = \sum_{X\in\mathcal{X}} || \frac{\hat{h}_X^T}{||\hat{h}_X^T||_2} - \frac{\hat{h}_X^{S_\theta}}{||\hat{h}_X^{S_\theta}||_2} ||_2^2 \quad (3)$$

Any type of loss such as contrastive (Sun et al., 2020a), or mean-square-error (MSE) (Passban et al., 2021; Sun et al., 2019) can be applied for our RAIL-KD approach.

3.3 Training Loss

The intermediate representation distillation loss $L_{\text{RAIL-KD}}$ is combined with the original KD loss L_{KD}, which is used to distill the knowledge from the output logits of the teacher model T to the output logits of the student model S_θ, and the original cross-entropy loss L_{CE}. The total loss function for training the student model is:

$$\mathcal{L} = \lambda_1 L_{\text{CE}} + \lambda_2 L_{\text{KD}} + \lambda_3 L_{\text{RAIL-KD}^{l/c}} \quad (4)$$

where λ_1, λ_2, and λ_3 are hyper-parameters of our model to minimize the total loss, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

4 Experimental Protocol

4.1 Datasets and Evaluation

We evaluate RAIL-KD on 8 tasks from the GLUE benchmark (Wang et al., 2018): 2 single-sentence (CoLA and SST-2) and 5 sentence-pair (MRPC, RTE, QQP, QNLI, and MNLI) classification tasks, and 1 regression task (STS-B). Following prior works (Sun et al., 2019; Passban et al., 2021; Jiao et al., 2019; Sun et al., 2020a), we use the same metrics as the GLUE benchmark for evaluation. Moreover, to further show the generalization capability of our RAIL-KD method on out-of-domain (OOD) across tasks, we use Scitail (Khot et al., 2018), PAWS (Paraphrase Adversaries from Word Scrambling) (Zhang et al., 2019), and IMDb (Internet Movie Database) (Maas et al., 2011) test sets to evaluate the models fine-tuned on MNLI, QQP, and SST-2 tasks respectively.

4.2 Implementation Details

We run extensive experiments on 3 different teachers in order to ensure a fair comparison with a wide range of prior works, and also to show the effectiveness of RAIL-KD. We experiment with the 12-layer BERT-base-uncased (Devlin et al., 2019) model as teacher (BERT$_{12}$) and the 6-layer DistilBERT (Sanh et al., 2019a) as student (DistillBERT$_6$) to compare with PKD (Sun et al., 2019) and ALP-KD (Passban et al., 2021). Also, we consider 24-layer RoBERTa-large (Liu et al., 2020) and 6-layer DistilRoberta (Sanh et al., 2019b) as the backbone for teacher (RoBERTa$_{24}$) and student (DistilRoberta$_6$) respectively to compare results when $n >> m$. Furthermore, we perform evaluation using the 12 layers RoBERTa-base (RoBERTa$_{12}$) model as a teacher to be able to directly compare our figures with the ones of CoDIR.

We re-implement PKD (Sun et al., 2019) and ALP-KD (Passban et al., 2021) approaches using the default settings proposed in the respective papers. We used early stopping based on performance on the development set, while making sure that the figures are in line with the ones reported in the papers. More precisely, the best layer setting for PKD teacher BERT$_{12}$ is $\{2, 4, 6, 8, 10\}$ to distill into DistilBERT$_6$. For DistilRoBERTa$_6$, we choose the intermediate layers 4, 8, 12, 16, 20 from the teacher RoBERTa$_{24}$ model for distillation that we found to work the best on the development set.

Using ALP-KD, we compute attention weights for the intermediate layers of the teacher (i.e., 1 to 11 for BERT$_{12}$ and 1 to 23 for RoBERTa$_{24}$ models) to calculate the weighted intermediate representations of the teacher for each intermediate layer of the student model (i.e., 1 to 5 layers of the student models). Since, the hidden dimensions of the RoBERTa$_{24}$ and DistilRoBERTa$_6$ are different, we linearly transform them into same lower-dimensional space. We train the PKD and ALP-KD models following (Sun et al., 2019; Passban et al., 2021).

For RAIL-KD^l, at each epoch we randomly select 5 layers from the intermediate layers of the teacher (i.e., from layers 1 to 11 for BERT$_{12}$ model and 1 to 23 for RoBERTa$_{24}$ model). Then, we sort the layer indexes and perform layer-wise distillation (Figure 1(a)) for RAIL-KD^l. For RAIL-KD^c, we concatenated the representations of the sorted randomly selected intermediate layers and then perform concatenated representation distillation (Figure 1(b)).

We use a linear transformation to map the intermediate representations (layer-wise or concatenated representations) into 128-dimensional space ($u = 128$) and normalize them before computing the loss $L_{\text{RAIL-KD}^{l/c}}$ for both BERT$_{12}$ and RoBERTa$_{24}$ distillations. We fixed $\alpha_i = 1$, $\lambda_1, \lambda_2, \lambda_3 = 1/3$ for our proposed approaches [4]. We search learning rate from {1e-5, 2e-5, 5e-5, 4e-6}, batch size from {8, 16, 32}, and fixed the epoch number to 40 for all the experiments. we run all experiments 5 times and report average score, in order to validate the credibility of our results. We ran all the experiments on a single NVIDIA V100 GPU using mixed-precision training (Micikevicius et al., 2018) and PyTorch (Paszke et al., 2019) framework.

5 Results

Table 2 shows the performances of models trained on GLUE tasks, and evaluated on their respective DEV and TEST sets for 12-layer to 6-layer distillation. BERT^{12} and DistilBERT^6 are used as backbone for the teacher and student models respectively. The baselines are fine-tuned without KD (w/o KD) and with Vanilla KD. Moreover, we di-

[4] We didn't find a significant improvement when changing these values.

Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Avg.
					DEV				
Teacher	61.3	93.0	90.6	88.4	91.0	84.7	91.5	68.2	83.7
w/o KD	53.3	90.1	90.0	86.5	90.4	82.3	89.1	61.7	80.4
Vanilla KD	55.8	90.3	90.3	86.6	90.5	82.7	89.6	68.5	81.9
PKD	56.1	91.3	90.7	87.4	91.2	83.3	90.2	69.3	82.5
ALP-KD	56.8	90.8	90.6	87.5	91.0	83.4	90.2	70.4	82.7
RAIL-KD^l	**58.8**	**92.8**	**91.0**	87.8	91.2	83.5	**90.3**	70.4	83.2
RAIL-KD^c	57.2	91.9	90.8	**87.9**	**91.4**	**83.5**	90.1	**72.2**	**83.2**
					TEST				
Teacher	52.0	92.9	87.8	82.3	88.9	84.3	90.7	66.0	81.0
w/o KD	50.7	91.7	87.2	80.4	88.3	81.4	88.4	57.6	78.6
Vanilla KD	50.9	91.0	87.7	81.0	88.5	82.2	88.7	60.6	79.2
PKD	50.6	92.0	87.2	81.7	89.1	82.7	89.0	60.6	79.5
ALP-KD	50.2	90.8	87.6	81.9	89.0	82.7	88.9	**61.8**	79.5
RAIL-KD^l	**51.3**	92.3	87.9	**82.1**	**89.2**	82.6	89.0	60.8	79.7
RAIL-KD^c	50.6	**92.5**	**88.2**	81.4	88.9	**82.8**	**89.3**	61.3	**79.8**

Table 2: DEV and TEST performances on GLUE benchmark when BERT_{12} and DistillBERT_6 are used as backbone for the teacher and students variants respectively. Bold mark describes the best results.

Model	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	Avg.
					DEV				
Teacher	68.1	96.4	91.9	92.3	91.5	90.1	94.6	86.3	88.9
PKD	62.3	91.6	90.9	88.9	91.6	84.4	91.1	71.1	84.0
ALP-KD	62.7	91.7	91.1	89.1	91.4	84.3	90.8	71.1	84.0
RAIL-KD^l	**65.4**	**93.8**	90.1	89.4	91.9	84.8	92.0	72.9	85.1
RAIL-KD^c	65.3	93.7	**91.4**	**89.4**	**92.0**	**84.8**	**92.0**	**72.9**	**85.2**
					TEST				
Teacher	68.1	96.4	91.9	92.3	91.5	90.2	94.6	86.3	85.3
PKD	50.2	89.4	88.9	84.5	92.3	84.0	90.2	62.7	80.3
ALP-KD	53.6	89.6	89.2	84.6	92.8	83.6	90.4	**64.4**	81.0
RAIL-KD^l	53.4	89.5	88.9	84.8	**93.6**	**84.5**	91.1	63.5	81.2
RAIL-KD^c	**53.6**	**89.6**	**89.6**	**84.8**	93.4	83.9	**91.6**	63.8	**81.3**

Table 3: DEV and TEST performances on GLUE benchmark when RoBERTa_{24} and DistillRoberta_6 are used as backbone for the teacher and student variants respectively. Bold mark describes the best results.

Model	CoLA	SST-2	MRPC	QQP	MNLI	QNLI	RTE	Avg.
RoBERTa_{12}	62.0	95.3	90.1	89.4	87.2	93.2	72.7	84.6
CoDIR	53.6	93.6	**89.4**	89.1	83.2	90.4	**65.6**	81.0
RAIL-KD^c	**54.2**	**93.6**	88.4	**89.5**	**83.9**	**91.7**	64.5	**81.2**

Table 4: GLUE test results of RAIL-KD^c when using Roberta_{12} and DistilRoBERTa_6 as backbone for teacher and students. Results of CoDIR are directly copied from their paper (Sun et al., 2020a).

rectly compare RAIL-KD^{lc} results with PKD and ALP-KD as more competing techniques.

First, we observe that the performance gap between ILD methods and vanilla-KD is tight (0.8% and 0.3% on DEV and TEST sets respectively). Moreover, as we expect, ALP-KD performs better on DEV and similar on TEST compared to PKD with 0.2% improvement on the DEV results. Second, results show that RAIL-KD outperforms the best ILD methods by a margin of 0.5% and 0.3% on average on DEV and TEST sets respectively. We notice that, except on RTE TEST, our RAIL-KD variants obtained the highest per-task performances. Third, we observe that RAIL-KD variants perform very similarly, which indicates that our method is effective on concatenated as well as layer-wise distillation.

Similar trends are seen on the 24- to 6-layer model compression experiments, which are reported in Table 3. In this experiment, we used Roberta$_{24}$ and DistillRoberta$_6$ as teacher and students models respectively. Overall, RAIL-KD outperforms the best baseline by 1.2% and 0.3% on DEV and TEST sets respectively. Interestingly, the gap on DEV compared with PKD and ALP-KD is larger than the one reported on BERT^{12} experiments, and PKD TEST socres are much lower from that of ALP and RAIL-KD. This might be because PKD skips a large number of intermediate layers on RoBERTa$_{24}$, and the computational cost of ALP-KD attention weights over a large number of teacher layers might produce smaller weights on Roberta$_{24}$ compared to BERT$_{12}$.

Furthermore, we demonstrate the effectiveness of RAIL-KD by directly comparing it with CoDIR (Sun et al., 2020a), the current state-of-the-art ILD method. It uses the contrastive objective and a memory bank to extract a large number of negative samples for contrastive loss calculations. Table 4 shows GLUE test results of both approaches when distilling RoBERTa$_{12}$ to DistillRoberta$_6$. CoDIR results are adopted from their paper, and we followed their experimental protocol by not reporting scores on STS-B. On average, RAIL-KD performs on par with CoDIR (+0.2%) and outperforms it on 5 out of 8 datasets, while being almost two-times faster as shown in the next section.

5.1 Training Speed-up

Table 5 shows the training time speed up against the teacher of different techniques on 8 GLUE tasks. We measured the speed up by calculating the $student\_train\_time/teacher\_train\_time$ using RoBERTa^{2}4 and DistilRoBERTa^6 as backbone for teacher and student respectively. We used this configuration because CoDIR pretrained student models are not available and we can only run CoDIR code out-of-the-box.

Model	Teacher	PKD	ALP	CoDIR	RAIL^l	RAIL^c
Speed-up	1.00×	1.89×	1.75×	1×	1.89×	**1.96**×

Table 5: Training time speedup against the teacher for different techniques using the same setting of Table 4.

Our results indicate that random layer mapping not only delivers consistently better results than the deterministic mapping technique such as PKD, but it has less computational overhead during training (two-times faster than CoDIR), while avoiding extensive search experiments to find an optimal mapping. Furthermore, using attention for layer selection (ALP-KD) or contrastive learning (CoDIR) leads to slightly worse performance result than random selection.

5.2 Impact of Random Layer Selection

To evaluate the impact of random layer selection on the performance of RAIL-KD, we report the standard deviation of the DistilBERT$_6$ student models (Table 2 models) on the 8 GLUE tasks. As Table 6 shows, the variances of RAIL-KD is in the same range for each task, for instance, RAIL-KD variance is at the same scale compared with PKD and ALP-KD on CoLA and MRPC, and even lower on RTE. This indicates that the gains of RAIL-KD are significant, and are not due to chance in our random selection of layers to distill.

5.3 Out-of-Distribution Test

We further validate the generalization ability of student models by measuring their robustness to in-domain and out-of-domain evaluation. We do so by evaluating models fine-tuned on MLI, QQP and SST-2 and then evaluated on SciTail, PAWS, and IMDB respectively. These datasets contains counterexamples to biases found in the training data (McCoy et al., 2019; Schuster et al., 2019; Clark et al., 2019). Performances of BERT$_{12}$/Roberta$_{24}$ teacher and

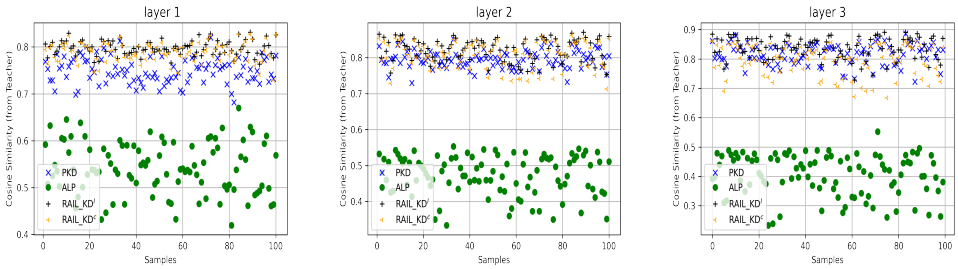

Figure 2: Cosine similarity between the intermediate layer representations of the BERT_{12} teacher and DistillBERT_6 student models computed on the SST-2 dataset.

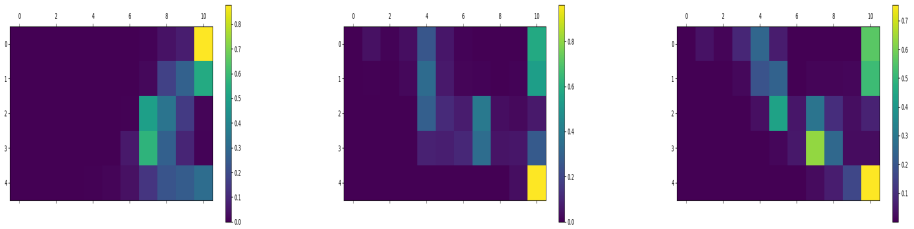

Figure 3: Distribution of attention weights learned by DistilBERT_6 ALP-KD on CoLA (left), RTE (middle), and MRPC (right). x-axis and y-axis are the teacher and student layer index respectively.

	CoLA	**SST-2**	**MRPC**	**STS-B**
PKD	±0.14	±0.54	±0.24	±0.64
ALP-KD	±0.95	±0.33	±0.70	±0.93
RAIL-KD^l	±0.49	±0.29	±0.25	±0.50
RAIL-KD^c	±0.51	±0.34	±0.40	±0.81
	QQP	**MNLI**	**QNLI**	**RTE**
PKD	±0.53	±0.33	±0.49	±1.50
ALP-KD	±0.27	±0.41	±0.30	±1.30
RAIL-KD^l	±0.64	±0.27	±0.65	±0.40
RAIL-KD^c	±0.14	±0.32	±0.25	±0.23

Table 6: Standards deviation (5 runs) of DistilBERT_6 ILD models on 8 GLUE tasks.

DistilBERT_6/DistilRoBERTa_6 student variants are reported in Table 7. Also, we compute the unweighted average score of the three tasks.

First, we notice high variability in models rank and some inconsistencies in performances across tasks when compared with in-domain results. This was also reported in prior works on out-of-domain training and evaluation (Clark et al., 2019; Mahabadi et al., 2020; Utama et al., 2020; Sanh et al., 2020). Still, RAIL-KD clearly outperforms all baselines across tasks. Surprisingly, we observe that PKD and ALP-KD perform poorly (on all three tasks) compared to the Vanilla KD baseline.

Interestingly, we observe that RAIL-KD^l performs consistently better (1.1% on average) than

Model	**SciTail**	**PAWS**	**IMDB**	**Avg.**
Teacher	70.3/82.7	43.3/43.3	84.6/88.9	66.0/71.6
w/o KD	68.7/74.9	36.5/34.7	81.3/85.8	62.2/65.1
Vanilla KD	68.6/76.1	42.2/36.6	82.0/86.1	64.3/66.3
PKD	68.0/74.8	39.9/36.5	80.9/85.4	62.9/65.6
ALP-KD	66.9/74.7	40.7/35.7	78.7/82.8	62.1/64.4
RAIL-KD^l	68.6/**76.6**	39.0/**36.9**	83.2/**87.3**	63.6/**67.0**
RAIL-KD^c	**68.7**/75.6	**43.7**/36.2	**85.0**/85.9	**65.8**/65.9

Table 7: Out-of-domain performances of models trained on MNLI, QQP, SST-2 and evaluated on SciTail, PAWS, and IMDB respectively. BERT_{12}/Roberta_{24} and DistilBERT_6/DistilRoBERTa_6 are used as backbone for the teacher and students respectively. For each setting, we report the unweighted average score on the 3 tasks.

RAIL-KD^c on Roberta_{24} compression, while RAIL-KD^c performs better (1.1% on average) on BERT_{12}. These results suggest that layer-wise distillation approach is more effective than concatenated distillation when we have a large capacity gap (layer number) between the teacher and the student, and vice versa.

6 Analysis

We run extensive analysis to better understand why RAIL-KD performs better than the other baselines. We visualize the layer-wise cosine similarity between the intermediate representations of the

teacher and the student networks. Figure 2 shows the cosine similarity score between three intermediate layer representations of BERT^{12} teacher (i.e. layers 2, 4 and 6) and the first three layer representations of the student for PKD, ALP-KD, RAIL-KD$^{l/c}$ students on 100 samples randomly selected from the SST-2 dataset. Due to space constraints, we only plot the scores for the first three layers of the student model. Similar trends are seen from the other layers.

We found that RAIL-KD allows the student to mimic teacher layers similar to PKD and much better than ALP-KD, despite that the mapping scheme varies at each epoch. Moreover, we observe that ALP-KD results have less similarity scores in the upper intermediate layers. PKD gives lower similarity scores in the lower layers while improving in the upper layers. In contrast, our approach gives more stable similarity scores for all layers while getting closer to the teacher representation in the upper layers.

We further investigate the attention weights learned by ALP-KD, and find out that they mostly focus on few layers (sparse attention). Figure 3 illustrates the distribution of weights, averaged on all training samples of DistilBERT$_6$ ALP-KD studnet on CoLA (left), RTE (middle), and MRPC (right) [5]. The figure clearly shows (light colors) that most of ALP weights are concentrated on top layers of the teacher. For instance, layers 1,2,5 of the three students mostly attend to the last layer of BERT$_{12}$. This is an indicator that ALP-KD overfits to the information driven from last layers. In contrast, the randomness in layer selection of RAIL-KD ensures a *uniform focus* on teacher layers. This may explain the poor performance of ALP-KD on out-of-domain evaluation compared with RAIL-KD.

From Figure 3, we see clearly that ALP-KD mostly prefers the upper layers of the teacher. On the other hand, the deterministic nature of PDK allows it to match better particular layers of the teacher (e.g. bottom ones as shown in Figure 2), but PKD never sees the layers that are skipped by the mapping. Consequently, it is expected that even though PDK can mimic bottom layers well, it is worse overall because it completely ignores some layers of the teacher. Random layer selection allow RAIL-KD to mimic all teacher layers while delivering high performances.

[5]Similar trends found on other datasets.

7 Conclusion and Future Work

We introduced a novel, simple, and efficient intermediate layer KD approach that outperforms the conventional approaches with performance improvement and efficient training time. RAIL-KD selects random intermediate layers from the teacher which equals to the number of intermediate layers of the student model. The selected intermediate layers are then sorted to distill their representations into the student model. RAIL-KD yields better regularization, which helps performance. Furthermore, our approach shows better performance for larger model distillation with faster training time, which opens up an avenue to investigate our approach for super-large models.

Acknowledgments

We thank Mindspore[6] for the partial support of this work, which is a new deep learning computing framework. We thank the anonymous reviewers for their insightful comments.

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*, pages 535–541.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. 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 4060–4073.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *https://arxiv.org/abs/1810.04805*.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Jie Fu, Xue Geng, Zhijian Duan, Bohan Zhuang, Xingdi Yuan, Adam Trischler, Jie Lin, Chris Pal, and Hao Dong. 2020. Role-wise data augmentation for knowledge distillation. *arXiv preprint arXiv:2004.08861*.

[6]https://www.mindspore.cn/

Mitchell A Gordon, Kevin Duh, , and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*.

Fu-Ming Guo, Sijia Liu, Finlay S Mungall, Xue Lin, and Yanzhi Wang. 2019. Reweighted proximal pruning for large-scale language representation. *arXiv preprint arXiv:1909.12486*.

Xuanli He, Islam Nassar, Jamie Kiros, Gholamreza Haffari, and Mohammad Norouzi. 2021. Generate, annotate, and learn: Generative models advance self-training and knowledge distillation. *arXiv preprint arXiv:2106.06168*.

Geoffrey Hinton, Oriol Vinyals, and Jeff. Dean. 2014. Distilling the knowledge in a neural network. *NIPS Workshop, https://arxiv.org/abs/1503.02531*.

Aref Jafari, Mehdi Rezagholizadeh, Pranav Sharma, and Ali Ghodsi. 2021. Annealing knowledge distillation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2493–2504, Online. Association for Computational Linguistics.

Mingi Ji, Byeongho Heo, and Sungrae Park. 2021. Show, attend and distill: Knowledge distillation via attention-based feature matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Ehsan Kamalloo, Mehdi Rezagholizadeh, and Ali Ghodsi. 2022. When chosen wisely, more data is what you need: A universal sample-efficient strategy for data augmentation. *arXiv preprint arXiv:2203.09391*.

Ehsan Kamalloo, Mehdi Rezagholizadeh, Peyman Passban, and Ali Ghodsi. 2021. Not far away, not so close: Sample efficient nearest neighbour data augmentation via minimax. *arXiv preprint arXiv:2105.13608*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Krtin Kumar, Peyman Passban, Mehdi Rezagholizadeh, Yiusing Lau, and Qun Liu. 2022. From fully trained to fully random embeddings: Improving neural machine translation with compact word embedding tables.

Tianda Li, Ahmad Rashid, Aref Jafari, Pranav Sharma, Ali Ghodsi, and Mehdi Rezagholizadeh. 2021. How to select one among all? an extensive empirical study towards the robustness of knowledge distillation in natural language understanding.

Vasileios Lioutas, Ahmad Rashid, Krtin Kumar, Md Akmal Haidar, and Mehdi Rezagholizadeh. 2020. Improving word embedding factorization for compression using distilled nonlinear neural decomposition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2774–2784.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Roberta: A robustly optimized fbertg pretraining approach. *arXiv preprint arXiv:1907.11692*.

Hou Lu, Huang Zhiqi, Shang Lifeng, Jiang Xin, Chen Xiao, and Liu Qun. 2020. Dynabert: Dynamic bert with adaptive width and depth. *arXiv preprint arXiv:2004.04037*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*.

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. 2020. End-to-end bias mitigation by modelling biases in corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8706–8716. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? in neurips. In *NeurIPS*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *In International Conference on Learning Representations*.

Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. 2021. ALP-KD: attention-based layer projection for knowledge distillation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13657–13665. AAAI Press.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances*

in neural information processing systems, 32:8026–8037.

Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. 2020. Fully quantized transformer for machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1–14.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Ahmad Rashid, Vasileios Lioutas, Abbas Ghaddar, and Mehdi Rezagholizadeh. 2020. Towards zero-shot knowledge distillation for natural language processing. *arXiv preprint arXiv:2012.15495*.

Ahmad Rashid, Vasileios Lioutas, and Mehdi Rezagholizadeh. 2021. Mate-kd: Masked adversarial text, a companion to knowledge distillation. *arXiv preprint arXiv:2105.05912*.

Mehdi Rezagholizadeh, Aref Jafari, Puneeth Salad, Pranav Sharma, Ali Saheb Pasand, and Ali Ghodsi. 2021. Pro-kd: Progressive distillation by following the footsteps of the teacher. *arXiv preprint arXiv:2110.08532*.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019b. Distilroberta, a distilled version of roberta: smaller, faster, cheaper and lighter. https://huggingface.co/distilroberta-base.

Victor Sanh, Thomas Wolf, Yonatan Belinkov, and Alexander M Rush. 2020. Learning from others' mistakes: Avoiding dataset biases without modeling them. *arXiv preprint arXiv:2012.01300*.

Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus, and Regina Barzilay. 2019. Towards debiasing fact verification models. 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 3410–3416.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2019. Q-bert: Hessian based ultra low precision quantization of bert. *arXiv preprint arXiv:1909.05840*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. *https://arxiv.org/abs/1908.09355*.

Siqi Sun, Zhe Gan, Yu Cheng, Yuwei Fang, Shuohang Wang, and Jingjing Liu. 2020a. Contrastive distillation on intermediate representations for language model compression. In *EMNLP*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020b. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Marzieh S Tahaei, Ella Charlaix, Vahid Partovi Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. 2021. Kroneckerbert: Learning kronecker decomposition for pre-trained language models via knowledge distillation. *arXiv preprint arXiv:2109.06243*.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Towards debiasing nlu models from unknown biases. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7597–7610.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2020a. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. *arXiv preprint arXiv:2012.15828*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020b. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*.

Yimeng Wu, Peyman Passban, Mehdi Rezagholizadeh, and Qun Liu. 2020a. Why skip if you can combine: A simple knowledge distillation technique for intermediate layers. *https://arxiv.org/abs/2010.03034*.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2020b. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeuRIPS*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*.

George-Eduard Zaharia, Andrei-Marius Avram, Dumitru-Clementin Cercel, and Traian Rebedea. 2021. Dialect identification through adversarial learning and knowledge distillation on romanian BERT. In *Proceedings of the Eighth Workshop on NLP for Similar Languages, Varieties and Dialects, VarDial@EACL 2021, Kiyv, Ukraine, April 20, 2021*, pages 113–119. Association for Computational Linguistics.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. *arXiv preprint arXiv:1904.01130*.