ALP-KD: Attention-Based Layer Projection for Knowledge Distillation

Peyman Passban[2,*], **Yimeng Wu**[1], **Mehdi Rezagholizadeh**[1], **Qun Liu**[1]
[1]Huawei Noah's Ark Lab, [2]Amazon
passban.peyman@gmail.com
{yimeng.wu,mehdi.rezagholizadeh,qun.liu}@huawei.com

Abstract

Knowledge distillation is considered as a training and compression strategy in which two neural networks, namely a *teacher* and a *student*, are coupled together during training. The teacher network is supposed to be a trustworthy predictor and the student tries to mimic its predictions. Usually, a student with a lighter architecture is selected so we can achieve compression and yet deliver high-quality results. In such a setting, distillation only happens for final predictions whereas the student could also benefit from teacher's supervision for internal components.

Motivated by this, we studied the problem of distillation for intermediate layers. Since there might not be a one-to-one alignment between student and teacher layers, existing techniques skip some teacher layers and only distill from a subset of them. This shortcoming directly impacts quality, so we instead propose a combinatorial technique which relies on attention. Our model fuses teacher-side information and takes each layer's significance into consideration, then performs distillation between combined teacher layers and those of the student. Using our technique, we distilled a 12-layer BERT (Devlin et al. 2019) into 6-, 4-, and 2-layer counterparts and evaluated them on GLUE tasks (Wang et al. 2018). Experimental results show that our combinatorial approach is able to outperform other existing techniques.

Introduction

Knowledge distillation (KD) (Buciluǎ, Caruana, and Niculescu-Mizil 2006; Hinton, Vinyals, and Dean 2015) is a commonly-used technique to reduce the size of large neural networks (Sanh et al. 2019). Apart from this, we also consider it as a complementary and generic add-on to enrich the training process of any neural model (Furlanello et al. 2018).

In KD, a student network ($\mathcal{S}$) is glued to a powerful teacher ($\mathcal{T}$) during training. These two networks can be trained simultaneously or $\mathcal{T}$ can be a pre-trained model. Usually, $\mathcal{T}$ uses more parameters than $\mathcal{S}$ for the same task, therefore it has a higher learning capacity and is expected to provide reliable predictions. On the other side, $\mathcal{S}$ follows its teacher with a simpler architecture. For a given input, both models provide predictions where those of the student are penalized by an ordinary loss function (using *hard* labels) as well as predictions received from $\mathcal{T}$ (also known as *soft* labels).

Training a (student) model for a natural language processing (NLP) task can be formalized as a multi-class classification problem to minimize a cross-entropy (*ce*) loss function, as shown in Equation 1:

$$\mathcal{L}_{ce} = -\sum_{i=1}^{N}\sum_{w\in V}[\mathbb{1}(y_i = w)\times \log p_{\mathcal{S}}(y_i = w|x_i, \theta_{\mathcal{S}})] \quad (1)$$

where $\mathbb{1}(.)$ is the indicator function, V is a vocabulary set (or different classes in a multi-class problem), N is the number of tokens in an input sequence, and y is a prediction of the network $\mathcal{S}$ with a parameter set $\theta_{\mathcal{S}}$ given an input x.

To incorporate teacher's supervision, KD accompanies $\mathcal{L}_{ce}$ with an auxiliary loss term, $\mathcal{L}_{KD}$, as shown in Equation 2:

$$\mathcal{L}_{KD} = -\sum_{i=1}^{N}\sum_{w\in V}[p_{\mathcal{T}}(y_i = w|x_i, \theta_{\mathcal{T}})\times \log p_{\mathcal{S}}(y_i = w|x_i, \theta_{\mathcal{S}})] \quad (2)$$

Since $\mathcal{S}$ is trained to behave identically to $\mathcal{T}$, model compression can be achieved if it uses a simpler architecture than its teacher. However, if these two models are the same size KD would still be beneficial. What $\mathcal{L}_{KD}$ proposes is an ensemble technique by which the student is informed about teacher's predictions. The teacher has better judgements and this helps the student learn how much it deviates from true labels.

This form of KD that is referred to as Regular KD (RKD) throughout this paper, only provides $\mathcal{S}$ with external supervision for final predictions, but this can be extended to other components such as intermediate layers too. The student needs to be aware of the information flow inside teacher's layers and this becomes even more crucial when distilling from deep teachers. Different alternatives have been proposed to this end, which compare networks' internal layers in addition to final predictions (Jiao et al. 2020; Sun et al. 2020, 2019), but they suffer from other types of problems. The main goal in this paper is to study such models and address their shortcomings.

[*]Work done while Peyman Passban was at Huawei.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Problem Definition

To utilize intermediate layers' information (and other components in general), a family of models exists that defines a dedicated loss function to measure how much a student diverges from its teacher in terms of internal representations. In particular, if the goal is to distill from an n-layer teacher into an m-layer student, a subset of m (out of n) teacher layers is selected whose outputs are compared to those of student layers (see Equation 3 for more details). Figure 1 illustrates this concept.

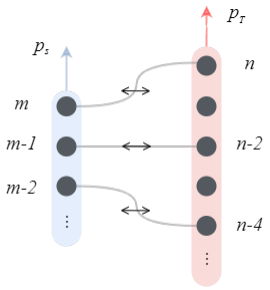

Figure 1: Student and teacher models have m and n layers, respectively. Each node is an intermediate layer and links are cross-model connections. In this example, every other layer of the teacher is skipped in order to match the size of the student. The output of nodes connected to each other are compared via a loss function (shown with $\leftrightarrow$) to ensure that the student model has similar internal representations as its teacher.

As the figure shows, each student layer is connected to a single, dedicated peer on the teacher side, e.g. the n-*th* teacher layer corresponds to the m-*th* student layer. Since outputs of these two layers are compared to each other, we hope that both models generate as similar outputs as possible at points n and m. With this simple technique, teacher's knowledge can be used to supervise student's intermediate layers.

Experimental results show that intermediate layer matching could be quite effective, but in our study we realized that it may suffer from two shortcomings:

- If $n \gg m$, multiple layers in $\mathcal{T}$ have to be ignored for distillation but we know that those layers consist of precious information for which we spend expensive resources to learn. This issue is referred to as the *skip* problem in this paper.
- Moreover, it seems the way teacher layers are kept/skipped is somewhat arbitrary as there is no particular strategy behind it. Before training, we lack enough knowledge to judge which subset of teacher layers contributes more to the distillation process, so there is a good chance of skipping significant layers if we pick them in an arbitrary fashion. Finding the best subset of layers to distill from requires an exhaustive search or an expert in the field to signify connections. We refer to this issue as the *search* problem.

In order to resolve the aforementioned issues we propose an alternative, which is the main contribution of this paper. Our solution does not skip any layer but utilizes *all* information stored inside $\mathcal{T}$. Furthermore, it combines teacher layers through an attention mechanism, so there is no need to deal with the search problem. We believe that the new notion of combination defined in this paper is as important as our novel KD architecture and can be adapted to other tasks too.

The remainder of this paper is organized as follows: First, we briefly review KD techniques used in similar NLP applications, then we introduce our methodology and explain how it addresses existing shortcomings. We accompany our methodology with experimental results to show whether the proposed technique is useful. Finally, we conclude the paper and discuss future directions.

Related Work

KD was originally proposed for tasks other than NLP (Buciluǎ, Caruana, and Niculescu-Mizil 2006; Hinton, Vinyals, and Dean 2015). Kim and Rush (2016) adapted the idea and proposed a sequence-level extension for machine translation. Freitag, Al-Onaizan, and Sankaran (2017) took a step further and expanded it to a multi-task scenario. Recently, with the emergence of large NLP and language understanding (NLU) models such as ELMO (Peters et al. 2018) and BERT (Devlin et al. 2019) KD has gained extra attention. Deep models can be trained in a better fashion and compressed via KD, which is favorable in many ways. Therefore, a large body of work in the field such as Patient KD (PKD) (Sun et al. 2019) has been devoted to compressing/distilling BERT (and similar) models.

PKD is directly related to this work, so we discuss it in more detail. It proposes a mechanism to match teacher and student models' intermediate layers by defining a third loss function, $\mathcal{L}_P$, in addition to $\mathcal{L}_{ce}$ and $\mathcal{L}_{KD}$, as shown in Equation 3:

$$\mathcal{L}_P = -\sum_{i=1}^{N}\sum_{j=1}^{m} || \frac{h_{\mathcal{S}}^{i,j}}{||h_{\mathcal{S}}^{i,j}||_2} - \frac{\mathcal{A}(j)^i}{||\mathcal{A}(j)^i||_2} ||_2^2 \qquad (3)$$

where $h_{\mathcal{S}}^{i,j}$ is the output[1] of the j-th student layer for the i-th input. A subset of teacher layers selected for distillation is denoted with an alignment function $\mathcal{A}$, e.g. $\mathcal{A}(j) = h_{\mathcal{T}}^{l}$ implies that the output of the j-th student layer should be compared to the output of the l-th teacher layer ($h_{\mathcal{S}}^{i,j} \leftrightarrow h_{\mathcal{T}}^{i,l}$).

PKD is not the only model that utilizes internal layers' information. Other models such as TinyBERT (Jiao et al. 2020) and MobileBERT (Sun et al. 2020) also found it crucial for training competitive student models. However, as Equation 3 shows, in these models only m teacher layers (the number of teacher layers returned by $\mathcal{A}$) can contribute to distillation. In the presence of deep teachers and small students, this limitation can introduce a significant amount of information loss. Furthermore, what is denoted by $\mathcal{A}$ directly impacts quality. If $\mathcal{A}$ skips an important layer the student model may fail to provide high-quality results.

[1]By the output, we mean the output of the layer for the *CLS* token. For more details about *CLS* see Devlin et al. (2019).

To tackle this problem, Wu et al. (2020) proposed a combinatorial technique, called CKD. In their model, $\mathcal{A}(j)$ returns a subset of teacher layers instead of a single layer. Those layers are combined together and distillation happens between the combination result and the j-th student layer, as shows in equation 4:

$$\begin{aligned} \hat{\mathcal{C}}^j &= \mathcal{F}_c(h^k_{\mathcal{T}});\ h^k_{\mathcal{T}} \in \mathcal{A}(j) \\ \mathcal{C}^j &= \mathcal{F}_r(\hat{\mathcal{C}}^j) \\ \cup_{j=1}^{m} \mathcal{A}(j) &= \{h^1_{\mathcal{T}}, ..., h^n_{\mathcal{T}}\} \end{aligned} \tag{4}$$

where $\hat{\mathcal{C}}^j$ is the result of a combination produced by the function $\mathcal{F}_c$ given a subset of teacher layers indicated by $\mathcal{A}(j)$. In Wu et al. (2020), $\mathcal{F}_c$ is implemented via a simple concatenation. Depending on the form of combination used in Equation 4, there might be a dimension mismatch between $\hat{\mathcal{C}}^j$ and the student layer $h^j_{\mathcal{S}}$. Accordingly, there is another function, $\mathcal{F}_r$, to reform the combination result into a comparable shape to the student layer. CKD uses a single projection layer to control the dimension mismatch.

With the combination technique (concatenation+projection), CKD could solve the skip problem but the search problem still remains unanswered. Similar to PKD, CKD also requires a search process, but it looks for the best subset of teacher layers instead of the best single layer. These two models are directly related to this research so we consider them as baselines in our experiments.

The application of KD in NLP and NLU is not limited to the aforementioned models. Aguilar et al. (2020) followed the same architecture as PKD but they introduced a new training regime, called progressive training. In their method, lower layers are trained first and training is progressively shifted to upper layers. They claim that the way internal layers are trained during KD can play a significant role. Liu et al. (2019) investigated KD from another perspective. Instead of focusing on the compression aspect, they kept the size of student models equal to their teachers and showed how KD could be treated as a complementary training ingredient.

Tan et al. (2019) squeezed multiple translation engines into one transformer (Vaswani et al. 2017) and showed that knowledge can be distilled from multiple teachers. Wei et al. (2019) introduced a novel training procedure where there is no need for an external teacher. A student model can learn from its own checkpoints. At each validation step, if the current checkpoint is better than the best existing checkpoint, student learns from it otherwise the best stored checkpoint is considered as a teacher.

Methodology

For a given student model $\mathcal{S}$ and a teacher model $\mathcal{T}$ we show all intermediate layers with sets $H_{\mathcal{S}} = \{h^1_{\mathcal{S}}, ..., h^m_{\mathcal{S}}\}$ and $H_{\mathcal{T}} = \{h^1_{\mathcal{T}}, ..., h^n_{\mathcal{T}}\}$, respectively. Based on the pipeline designed by current models for intermediate layer KD, there must be a connection between $H_{\mathcal{S}}$ and $H_{\mathcal{T}}$ during training and each student layer can only correspond to a single peer on the teacher side. As previously mentioned, layer connections are denoted by $\mathcal{A}$.

A common heuristic to devise $\mathcal{A}$ is to divide teacher layers into m buckets with approximately the same sizes and pick only one layer from each (Jiao et al. 2020; Sun et al. 2019). Therefore, for the j-th layer of the student model, $\mathcal{A}(j)$ returns a single teacher layer among those that reside in the j-th bucket. Figure 2a illustrates this setting. Clearly, this is not the best way of connecting layers, because they are picked in a relatively arbitrary manner. More importantly, no matter what heuristic is used there still remain $n-m$ layers in this approach whose information is not used in distillation.

To address this issue, we simply propose a combinatorial alternative whereby all layers inside buckets are taken into consideration. Our technique is formulated in Equation 5:

$$\begin{aligned} \mathcal{C}^j &= \sum_{h^k_{\mathcal{T}} \in \mathcal{A}(j)} \alpha_{jk}\ h^k_{\mathcal{T}} \\ \alpha_{jk} &= \frac{\exp(h^j_{\mathcal{S}} \,.\, h^k_{\mathcal{T}})}{\sum_{h^{k'}_{\mathcal{T}} \in \mathcal{A}(j)} \exp(h^j_{\mathcal{S}} \,.\, h^{k'}_{\mathcal{T}})} \\ \cup_{j=1}^{m} \mathcal{A}(j) &= H_{\mathcal{T}} = \{h^1_{\mathcal{T}}, ..., h^n_{\mathcal{T}}\} \end{aligned} \tag{5}$$

This idea is similar to that of CKD, but we use an attention mechanism (Bahdanau, Cho, and Bengio 2015) instead of a concatenation for layer combination. Experimental results demonstrate that this form of combination is more useful. We refer to this idea as **A**ttention-based **L**ayer **P**rojection for **KD** or ALP-KD in short.

According to the equation, if a student layer associates with a particular bucket, all layers inside that bucket are combined/used for distillation and $\mathcal{C}^j$ is a vector representation of such a combination. Our model benefits from all n teacher layers and skips none as there is a dedicated $\mathcal{C}$ vector for each student layer. Figure 2b visualizes this setting.

Weights (α values) assigned to teacher layers are learnable parameters whose values are optimized during training. They show the contribution of each layer to the distillation process. They also reflect the correlation between student and teacher layers, i.e. if a student layer correlates more with a set of teacher layers weights connecting them should receive higher values. In other words, that specific layer is playing the role of its teacher peers on the student side. To measure the correlation, we use the *dot product* in our experiments but any other function for similarity estimation could be used in this regard.

Equation 5 addresses the *skip* problem with a better combination mechanism and is able to provide state-of-the-art results. However, it still suffers from the *search* problem as it relies on buckets and we are not sure which bucketing strategy works better. For example, in Figure 2b the first bucket consists of the first three layers of the teacher but it does not mean that we cannot append a fourth layer. In fact, a bucket with four layers might perform better. Buckets can also share layers; namely, a teacher layer can belong to multiple buckets and can be used numerous times in distillation. These constraints make it challenging to decide about buckets and their boundaries, but it is possible to resolve this dilemma through a simple modification in our proposed model.

To avoid bucketing, we span the attention mask over all teacher layers rather than over buckets. To implement this

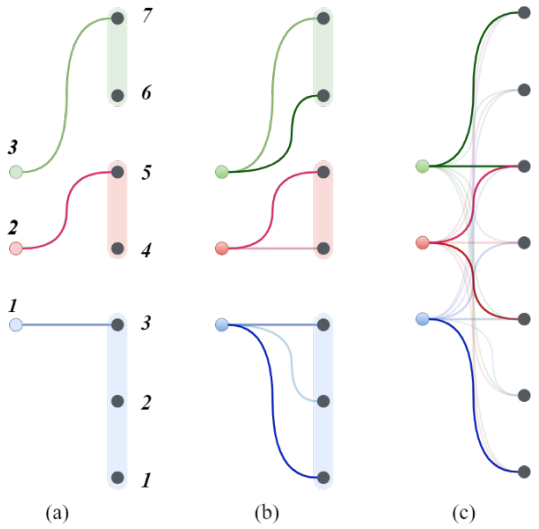

Figure 2: Three pairs of $\mathcal{S}$ and $\mathcal{T}$ networks with different forms of layer connections. In Figure 2a, teacher layers are divided into 3 buckets and only one layer from each bucket is connected to the student side, e.g. $h^5_\mathcal{T}$ is the source of distillation for $h^2_\mathcal{S}$ ($h^5_\mathcal{T} \leftrightarrow h^2_\mathcal{S}$). In Figure 2b, a weighted average of teacher layers from each bucket is considered for distillation, e.g. $\mathcal{A}(2) = \{h^4_\mathcal{T}, h^5_\mathcal{T}\}$ and $\mathcal{C}^2 = \alpha_{24} h^4_\mathcal{T} + \alpha_{25} h^5_\mathcal{T}$ ($\mathcal{C}^2 \leftrightarrow h^2_\mathcal{S}$). In Figure 2c, there is no bucketing and all teacher layers are considered for projection. Links with higher color intensities have higher attention weights.

extension, $\mathcal{A}(j)$ needs to be replaced with $H_\mathcal{T}$ in Equation 5. Therefore, for any student layer such as $h^j_\mathcal{S}$ there would be a unique set of n attention weights and $\mathcal{C}^j$ would be a weighted average of *all* teacher layers, as shown in Equation 6:

$$\begin{aligned} \mathcal{C}^j &= \sum_{h^k_\mathcal{T} \in \mathcal{A}(j)} \alpha_{jk}\, h^k_\mathcal{T} \\ \mathcal{A}(j) &= H_\mathcal{T} \quad \forall j \in \{1, 2, ..., m\} \end{aligned} \tag{6}$$

This new configuration, which is illustrated in Figure 2c, proposes a straightforward way of combining teacher layers and addresses both *skip* and *search* problems at the same time.

To train our student models, we use a loss function which is composed of $\mathcal{L}_{ce}$, $\mathcal{L}_{KD}$, and a dedicated loss defined for ALP-KD, as shown in Equation 7:

$$\begin{aligned} \mathcal{L} &= \beta \mathcal{L}_{ce} + \eta \mathcal{L}_{KD} + \lambda \mathcal{L}_{ALP} \\ \mathcal{L}_{ALP} &= \sum_{i=1}^{N} \sum_{j=1}^{m} MSE(h^{i,j}_\mathcal{S}, \mathcal{C}^{i,j}) \end{aligned} \tag{7}$$

where *MSE()* is the mean-square error and $\mathcal{C}^{i,j}$ shows the value of $\mathcal{C}^j$ when the teacher is fed with the i-th input. β, η, and λ are hyper-parameters of our model to minimize the final loss.

Experimental Study

A common practice in our field to evaluate the quality of a KD technique is to feed $\mathcal{T}$ and $\mathcal{S}$ models with instances of standard datasets and measure how they perform. We followed the same tradition in this paper and selected a set of eight GLUE tasks (Wang et al. 2018) including CoLA, MNLI, MRPC, QNLI, QQP, RTE, SST-2, and STS-B datasets to benchmark our models. Detailed information about datasets is available in the appendix section.

In NLP/NLU settings, $\mathcal{T}$ is usually a pre-trained model whose parameters are only fine-tuned during training. On the other side, $\mathcal{S}$ can be connected to $\mathcal{T}$ to be trained thoroughly or can alternatively be initialized with $\mathcal{T}$'s parameters to be fine-tuned similar to its teacher. This helps the student network generate better results and converge faster. Fine-tuning is more common than training in our context and we thus fine-tune our models rather than training. This concept is comprehensively discussed by Devlin et al. (2019) so we skip its details and refer the reader to their paper. We have the same fine-tuning pipeline in this work.

In our experiments, we chose the original BERT model[2] (also known as $\text{BERT}_{\text{Base}}$) as our teacher. We are faithful to the configuration proposed by Devlin et al. (2019) for it. Therefore, our in-house version also has 12 layers with 12 attention heads and the hidden and feed-forward dimensions are 768 and 3072, respectively. Our students are also BERT models only with fewer layers ($|H_\mathcal{S}| = m\,; m < 12$). We use the teacher BERT to initialize students, but because the number of layers are different ($12 \neq m$) we only consider its first m layers. We borrowed this idea from PKD (Sun et al. 2019) in the interest of fair comparisons.

In order to maximize each student's performance we need to decide about the learning rate, batch size, the number of fine-tuning iterations, and β, η, and λ. To this end, we run a grid search similar to Sun et al. (2019) and Wu et al. (2020). In our setting, the batch size is set to 32 and the learning rate is selected from $\{1e-5, 2e-5, 5e-5\}$. η and λ take values from $\{0, 0.2, 0.5, 0.7\}$ and $\beta = 1 - \eta - \lambda$.

We trained multiple models with different configurations and compared our results to RKD- and PKD-based students. To the best of our knowledge, these are the only alternatives that use BERT as a teacher and their students' architecture relies on ordinary Transformer blocks (Vaswani et al. 2017) with the same size as ours, so any comparison to any other model with different settings would not be fair. Due to CKD's similarity to our approach we also re-implemented it in our experiments. The original CKD model was proposed for machine translation and for the first time we evaluate it in NLU tasks. Table 1 summarizes our experiments.

The teacher model with 12 layers and 109M parameters has the best performance for all datasets.[3] This model can be compressed, so we reduce the number of layers to 4 and train another model ($\mathcal{S}_{\text{NKD}}$). The rest of the configuration (attention

[2] https://github.com/google-research/bert

[3] Similar to other papers, we evaluate our models on validation sets. Testset labels of GLUE datasets are not publicly available and researchers need to participate in leaderboard competitions to evaluate their models on testsets.

Problem	Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Average
N/A	$\mathcal{T}_{\text{BERT}}$	57.31	83.39	86.76	91.25	90.96	68.23	92.67	88.82	82.42
N/A	$\mathcal{S}_{\text{NKD}}$	31.05	76.83	77.70	85.13	88.97	61.73	88.19	87.29	74.61
skip, *search*	$\mathcal{S}_{\text{RKD}}$	29.22	79.31	79.41	86.77	90.25	65.34	90.37	87.45	76.02
skip, *search*	$\mathcal{S}_{\text{PKD}}$	32.13	79.26	80.15	86.64	90.23	65.70	90.14	87.26	76.44
search	$\mathcal{S}_{\text{CKD-NO}}$	31.23	79.42	80.64	86.93	88.70	66.06	90.37	87.62	76.37
search	$\mathcal{S}_{\text{CKD-PO}}$	31.95	79.53	80.39	86.75	89.89	**67.51**	90.25	87.55	76.73
search	$\mathcal{S}_{\text{ALP-NO}}$	**34.21**	79.26	79.66	**87.11**	**90.72**	65.70	90.37	87.52	76.82
search	$\mathcal{S}_{\text{ALP-PO}}$	33.86	**79.74**	79.90	86.95	90.25	66.43	**90.48**	87.52	76.89
none	$\mathcal{S}_{\text{ALP}}$	33.07	79.62	**80.72**	87.02	90.54	67.15	90.37	**87.62**	**77.01**

Table 1: Except the teacher ($\mathcal{T}_{\text{BERT}}$) which is a 12-layer model, all other models have 4 layers. Apart from the number of layers, all students have the same architecture as the teacher. The first column shows what sort of problems each model suffers from. NKD stands for *No KD* which means there is no KD technique involved during training this student model. *NO* and *PO* are different configurations for mapping internal layers. Boldfaced numbers show the best student score for each column over the validation set. Scores in the first column are Matthew's Correlations. SST-B scores are Pearson correlations and the rest are accuracy scores.

heads, hidden dimension etc) remains untouched. There is no connection between the teacher and $\mathcal{S}_{\text{NKD}}$ and it is trained separately with no KD technique. Because of the number of layers, performance drops in this case but we still gain a lot in terms of memory as this new model only has 53M parameters. To bridge the performance gap between the teacher and $\mathcal{S}_{\text{NKD}}$, we involve KD in the training process and train new models, $\mathcal{S}_{\text{RKD}}$ and $\mathcal{S}_{\text{PKD}}$, with RKD and PKD techniques, respectively.

$\mathcal{S}_{\text{RKD}}$ is equivalent to a configuration known as DistilBERT in the literature (Sanh et al. 2019). To have precise results and a better comparison, we trained/fine-tuned all models in the same experimental environment. Accordingly, we do not borrow any result from the literature but reproduce them. This is the reason we use the term equivalent for these two models. Furthermore, DistilBERT has an extra Cosine embedding loss in addition to those of $\mathcal{S}_{\text{RKD}}$. When investigating the impact of intermediate layers in the context of KD, we wanted $\mathcal{L}_P$ to be the only difference between RKD and PKD, so incorporating any other factor could hurt our investigation and we thus avoided the cosine embedding loss in our implementation.

PKD outperforms RKD with an acceptable margin in Table 1 and that is because of the engagement of intermediate layers. For $\mathcal{S}_{\text{PKD}}$, we divided teacher layers into 3 buckets (4 layers in each) and picked the first layer of each bucket to connect to student layers, i.e. $\mathcal{A}(1) = h^1_{\mathcal{T}}$, $\mathcal{A}(2) = h^5_{\mathcal{T}}$, and $\mathcal{A}(3) = h^9_{\mathcal{T}}$. There is no teacher layer assigned to the last layer of the student. This form of mapping maximizes PKD's performance and we figured out this via an empirical study.

Results discussed so far demonstrate that cross-model layer mapping is effective, but it can be improved even more if the skip issue is settled. Therefore, we trained two other students using CKD. The setting for these models is identical to PKD, namely teacher layers are divided into 3 buckets. The first 4 teacher layers reside in the first bucket. The fifth to eighth layers are in the second bucket and the rest are covered by the third bucket. Layers inside the first bucket are concatenated and passed through a projection layer to match the student layers' dimension. The combination result for the first bucket is assigned to the first student layer ($\mathcal{C}^1 \leftrightarrow h^1_{\mathcal{S}}$). The same procedure is repeated with the second and third buckets for $h^2_{\mathcal{S}}$ and $h^3_{\mathcal{S}}$. Similar to PKD, there is no teacher layer connected to the last student layer. This configuration is referred to as **N**o **O**verlap (**NO**), that indicates buckets share no layers with each other.

In addition to **NO** we designed a second configuration, **PO**, which stands for **P**artial **O**verlap. In **PO**, each bucket shares its first layer with the preceding bucket, so the first bucket includes the first to fifth layers, the second bucket includes the fifth to ninth layers, and from the ninth layer onward reside in the third bucket. We explored this additional configuration to see the impact of different bucketing strategies in CKD.

Comparing $\mathcal{S}_{\text{CKD}}$ to $\mathcal{S}_{\text{PKD}}$ shows that the combination (concatenation+projection) idea is useful in some cases, but for others the simple skip idea is still better. Even defining different bucketing strategies did not change it drastically, and this leads us to believe that a better form of combination such as an attention-based model is required.

In $\mathcal{S}_{\text{ALP}}$ extensions, we replace the CKD's concatenation with attention and results improve. ALP-KD is consistently better than all other RKD, PKD, and CKD variations and this justifies the necessity of using attention for combination. $\mathcal{S}_{\text{ALP-NO}}$ and $\mathcal{S}_{\text{ALP-PO}}$ also directly support this claim. In $\mathcal{S}_{\text{ALP}}$, we followed Equation 6 and spanned the attention mask over all teacher layers. This setting provides a model that requires no engineering adjustment to deal with *skip* and *search* problems and yet delivers the best result on average.

Training Deeper/Shallower Models Than 4-Layer Students

So far we compared 4-layer ALP-KD models to others and observed superior results. In this section, we design additional experiments to study our technique's behaviour from the size perspective. The original idea of PKD was proposed to distill from a 12-layer BERT to a 6-layer student (Sun et al. 2019). In such a scenario, only every other layer of the teacher is skipped and it seems the student model should not suffer from the skip problem dramatically. We repeated this experiment to understand if our combination idea is still useful or its impact diminishes when student and teacher models have closer architectures. Table 2 summarizes findings of this

Problem	Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Average
N/A	$\mathcal{T}_{\text{BERT}}$	57.31	83.39	86.76	91.25	90.96	68.23	92.67	88.82	82.42
N/A	$\mathcal{S}_{\text{NKD}}$	40.33	79.91	81.86	87.57	90.21	65.34	90.02	88.49	77.97
skip, *search*	$\mathcal{S}_{\text{RKD}}$	45.51	81.41	83.82	88.21	90.56	67.51	91.51	88.70	79.65
skip, *search*	$\mathcal{S}_{\text{PKD}}$	45.78	**82.18**	85.05	89.31	90.73	68.23	91.51	88.56	80.17
search	$\mathcal{S}_{\text{CKD-NO}}$	**48.49**	81.91	83.82	89.53	90.64	67.51	91.40	88.73	80.25
search	$\mathcal{S}_{\text{CKD-PO}}$	46.99	81.99	83.82	89.44	**90.82**	67.51	91.17	88.62	80.05
search	$\mathcal{S}_{\text{ALP-NO}}$	46.40	81.99	**85.78**	**89.71**	90.64	**68.95**	**91.86**	**88.81**	**80.52**
search	$\mathcal{S}_{\text{ALP-PO}}$	46.02	82.04	84.07	89.16	90.56	68.23	91.74	88.72	80.07
none	$\mathcal{S}_{\text{ALP}}$	46.81	81.86	85.05	89.67	90.73	68.59	**91.86**	88.68	80.41

Table 2: The teacher model $\mathcal{T}_{\text{BERT}}$ has 12 and all other student models have 6 layers.

experiment.

Among 6-layer students, $\mathcal{S}_{\text{ALP-NO}}$ has the best average score which demonstrates that the combinatorial approach is still useful. Moreover, the supremacy of attention-based combination over the simple concatenation holds for this setting too. $\mathcal{S}_{\text{ALP}}$ is the second best and yet our favorite model as it requires no layer alignment before training.

The gap between PKD and ALP-KD is narrowed in 6-layer models compared to 4-layer students, and this might be due to an implicit relation between the size and need for combining intermediate layers. We focused on this hypothesis in another experiment and this time used the same teacher to train 2-layer students. In this scenario, student models are considerably smaller with only 39M parameters. Results of this experiment are reported in Table 3.

For CKD and ALP-KD, we combine all teacher layers and distill into the first layer of the student. Similar to previous experiments, there is no connection between the last layer of 2-layer students and the teacher model and KD happens between $h^1_{\mathcal{S}}$ and $H_{\mathcal{T}}$. For PKD, we need to decide which teacher layers should be involved in distillation, for which we assessed three configurations with the first ($h^1_{\mathcal{S}} \leftrightarrow h^1_{\mathcal{T}}$), sixth ($h^1_{\mathcal{S}} \leftrightarrow h^6_{\mathcal{T}}$), and twelfth ($h^1_{\mathcal{S}} \leftrightarrow h^{12}_{\mathcal{T}}$) layers. $\mathcal{S}_{\text{ALP}}$ outperforms other students in this case too and this time the gap between PKD and ALP-KD is even more visible. This result points out to the fact that when teacher and student models differ significantly, intermediate layer combination becomes crucial.

Qualitative Analysis

We tried to visualize attention weights to understand what happens during training and why ALP-KD leads to better performance. Figure 3 illustrates results related to this experiment. From the SST-2 dataset, we randomly selected 10 examples and stimulated both teacher and student models to emit attention weights between the first layer of the student ($h^1_{\mathcal{S}}$) and all teacher layers ($H_{\mathcal{T}}$). We carried out this experiment with 2-, 4-, and 6-layer $\mathcal{S}_{\text{ALP}}$ models. The x and y axes in the figure show the attention weights and 10 examples, respectively.

As seen in Figure 3a, the first half of the teacher model is more active, which is expected since we distill into the first layer of the student. However, $h^1_{\mathcal{S}}$ receives strong signals from other layers in the second half too, e.g. in *Example-10* there is a strong connection between $h^{11}_{\mathcal{T}}$ and $h^1_{\mathcal{S}}$. This visualization demonstrates that all teacher layers participate in distillation and defining buckets or skipping layers might not be the best approach. A similar situation arises when distilling into the 4-layer model in Figure 3b as the first half is still more active. For the 6-layer model, we see a different pattern where there is a concentration in attention weights around the middle layers of the teacher and $h^1_{\mathcal{S}}$ is mainly fed by layers $h^4_{\mathcal{T}}$ to $h^7_{\mathcal{T}}$.

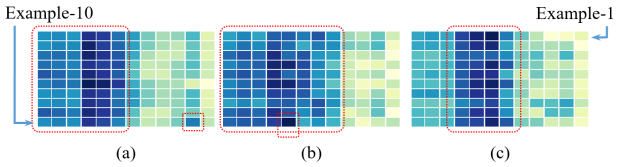

Figure 3: Visualizing attention weights between the first layer of the student model and all teacher layers for 10 samples from SST-2. Weights belong to $\mathcal{S}_{\text{ALP}}$ with 2 (a), 4 (b), and 6 (c) layers.

Considering the distribution of attention weights, any skip- or even concatenation-based approach would fail to reveal the maximum capacity of KD. Such approaches assume that a single teacher layer or a subset of adjacent layers affect the student model, whereas almost all of them participate in the process. Apart from previously reported results, this visualization again justifies the need for an attention-based combination in KD.

Our technique emphasizes on intermediate layers and the necessity of having similar internal representations between student and teacher models, so in addition to attention weights we also visualized the output of intermediate layers. The main idea behind this analysis is to show the information flow inside student models and how ALP-KD helps them mimic their teacher. Figures 4a and 4b illustrate this experiment.

We randomly selected 100 samples from the SST-2 dataset and visualized what hidden representations of $\mathcal{S}_{\text{ALP}}$, $\mathcal{S}_{\text{PKD}}$, and $\mathcal{T}$ models (from Table 1) look like when stimulated with these inputs. Student models have 4 layers but due to space limitations we only show middle layers' outputs, namely $h^2_{\mathcal{S}}$ (Figure 4a) and $h^3_{\mathcal{S}}$ (Figure 4b). $h^1_{\mathcal{S}}$ and $h^4_{\mathcal{S}}$ also expressed

Problem	Model	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Average
N/A	$\mathcal{T}_{\text{BERT}}$	57.31	83.39	86.76	91.25	90.96	68.23	92.67	88.82	82.42
N/A	$\mathcal{S}_{\text{NKD}}$	14.50	72.73	72.06	79.61	86.89	57.04	85.89	40.80	63.69
skip, *search*	$\mathcal{S}_{\text{RKD}}$	24.50	74.90	73.53	81.04	87.40	59.21	87.39	41.87	66.23
skip, *search*	$\mathcal{S}_{\text{PKD-1}}$	23.09	74.65	72.55	81.27	87.68	57.40	88.76	43.37	66.1
skip, *search*	$\mathcal{S}_{\text{PKD-6}}$	22.48	74.57	73.04	80.74	87.70	57.40	88.65	42.92	65.94
skip, *search*	$\mathcal{S}_{\text{PKD-12}}$	22.46	74.33	72.79	81.22	87.88	57.40	88.76	45.39	66.28
search	$\mathcal{S}_{\text{CKD}}$	**24.69**	74.67	73.04	**81.60**	87.10	58.84	88.65	43.71	66.54
none	$\mathcal{S}_{\text{ALP}}$	24.61	**74.78**	**73.53**	81.24	**88.01**	**59.57**	**88.88**	**46.04**	**67.08**

Table 3: The teacher model $\mathcal{T}_{\text{BERT}}$ has 12 and all other student models have 2 layers. $\mathcal{S}_{\text{PKD-}l}$ indicates that $h^l_{\mathcal{T}}$ is used for distillation.

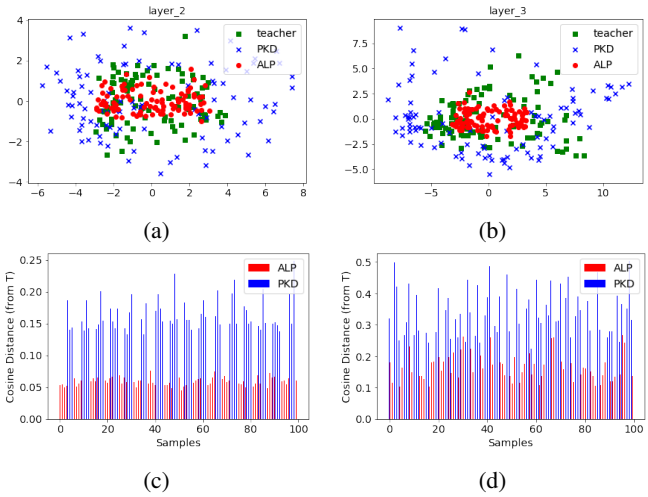

Figure 4: Visualizing intermediate layers' outputs and their distance from the teacher in ALP-KD and PKD students. Teacher-, ALP-KD-, and PKD-related information is visualized with green, red, and blue colors, respectively. Figures 4a and 4c provide information about h^2_{ALP}, h^2_{PKD}, and $h^5_{\mathcal{T}}$, and Figures 4b and 4d report information about h^3_{ALP}, h^3_{PKD}, and $h^9_{\mathcal{T}}$. In the bottom figures, the *x* axis shows samples and the *y* axis is the Cosine distance from the teacher.

very similar attitudes.

The output of each intermediate layer is a 768-dimensional vector, but for visualization purposes we consider the first two principal components extracted via PCA (Wold, Esbensen, and Geladi 1987). During training, $h^5_{\mathcal{T}}$ and $h^9_{\mathcal{T}}$ are connected to $h^2_{\mathcal{S}}$ and $h^3_{\mathcal{S}}$ as the source of distillation in PKD, so we also include those teacher layers' outputs in our visualization. As the figure shows, ALP-KD's representations are closer to teacher's and it demonstrates that our technique helps train better students with closer characteristics to teachers.

We conducted another complementary analysis where we used the output of the same teacher and student layers from the previous experiment and measured their distance for all 100 examples. Results of this experiment are illustrated in Figures 4c and 4d for the second and third student layers, respectively. Internal representations generated by PKD are more distant from those of the teacher compared to ALP-KD's representations, e.g. the distance between $h^{20,2}_{\text{PKD}}$ (the output of the second PKD layer for the 20-*th* example in Figure 4c) and $h^{20,5}_{\mathcal{T}}$ is around 0.20 whereas this number is only 0.05 for ALP-KD. This is an indication that the ALP-KD student follows its teacher better than the PKD student. To measure distance, we used the Cosine similarity in this experiment.

Conclusion and Future Work

In this paper, we discussed the importance of distilling from intermediate layers and proposed an attention-based technique to combine teacher layers without skipping them. Experimental results show that the combination idea is effective. Our findings in this research can be summarized as follows:

- It seems to distill from deep teachers with multiple internal components combination is essential.
- The more teacher and student models differ in terms of the number of layers, the more intermediate layer combination becomes crucial.
- Although a simple concatenation of layers is still better than skipping in many cases, to obtain competitive results an attention-based combination is required.
- ALP-KD can be tuned to combine layers inside buckets and this approach is likely to yield state-of-the-art results, but if there is no enough knowledge to decide about buckets, a simple attention mask over all teacher layers should solve the problem.

As our future direction, we are interested in applying ALP-KD to other tasks to distill from extremely deep teachers into compact students. Moreover, we will work on designing better attention modules. Techniques that are able to handle sparse structures could be more useful in our architecture. Finally, we like to adapt our model to combine other internal components such as attention heads.

Acknowledgments

We would like to thank our anonymous reviewers as well as Chao Xing and David Alfonso Hermelo from Huawei Noah's Ark Lab for their valuable feedback.

Appendix

GLUE Datasets

Datasets used in our experiments are as follows:

- **CoLA**: A corpus of English sentences drawn from books and journal articles with $8,551$ training and $1,043$ validation instances. Each example is a sequence of words with a label indicating whether it is a grammatical sentence (Warstadt, Singh, and Bowman 2019).
- **MNLI**: A multi-genre natural language inference corpus including sentence pairs with textual entailment annotations (Williams, Nangia, and Bowman 2018). The task defined based on this dataset is to predict whether the premise entails the hypothesis, contradicts it, or neither, given a premise sentence and a hypothesis. The dataset has two versions, *matched* (test and training examples are from the same domain) and *mismatched*, that we use the matched version. This dataset has $392,702$ training and $9,815$ validation examples.
- **MRPC**: A corpus of sentence pairs with human annotations. The task is to decide whether sentences are semantically equivalent (Dolan and Brockett 2005). The training and validation sets have $3,668$ and 408 examples, respectively.
- **QNLI**: A dataset built for a binary classification task to assess whether a sentence contains the correct answer to a given query (Rajpurkar et al. 2016). The set has $104,743$ training and $5,463$ validation examples.
- **QQP**: A set of question pairs with $363,849$ training and $40,430$ validation instances collected from the well-known question answering website Quora. The task is to determine if a given pair of questions are semantically equivalent (Iyer, Dandekar, and Csernai 2017).
- **RTE**: A combined set of $2,490$ training and 277 validations examples collected from four sources for a series of textual entailment challenges (Dagan, Glickman, and Magnini 2005; Bar-Haim et al. 2006; Giampiccolo et al. 2007; Bentivogli et al. 2009).
- **SST-2**: A sentiment analysis dataset with sentence-level (positive/negative) labels. The training and validation sets include $67,349$ and 872 sentences, respectively (Socher et al. 2013).
- **STS-B**: A collection of sentence pairs used for semantic similarity estimation. Each pair has a similarity score from 1 to 5. This dataset has $5,749$ training and $1,500$ validation examples (Cer et al. 2017).

Hardware

Each model is fine-tuned on a single *NVIDIA 32GB V100* GPU. The fine-tuning time, based on the dataset size, can vary from a few hours to one day on a single GPU.

References

Aguilar, G.; Ling, Y.; Zhang, Y.; Yao, B.; Fan, X.; and Guo, C. 2020. Knowledge Distillation from Internal Representations. In *AAAI*, 7350–7357.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Bar-Haim, R.; Dagan, I.; Dolan, B.; Ferro, L.; Giampiccolo, D.; Magnini, B.; and Szpektor, I. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*, volume 6, 6–4. Venice.

Bentivogli, L.; Clark, P.; Dagan, I.; and Giampiccolo, D. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. In *TAC*.

Buciluǎ, C.; Caruana, R.; and Niculescu-Mizil, A. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 535–541.

Cer, D.; Diab, M.; Agirre, E.; Lopez-Gazpio, I.; and Specia, L. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 1–14.

Dagan, I.; Glickman, O.; and Magnini, B. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 177–190. Springer.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Dolan, W. B.; and Brockett, C. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Freitag, M.; Al-Onaizan, Y.; and Sankaran, B. 2017. Ensemble distillation for neural machine translation. *arXiv preprint arXiv:1702.01802* .

Furlanello, T.; Lipton, Z.; Tschannen, M.; Itti, L.; and Anandkumar, A. 2018. Born again neural networks. In *International Conference on Machine Learning*, 1607–1616. PMLR.

Giampiccolo, D.; Magnini, B.; Dagan, I.; and Dolan, B. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, 1–9. Association for Computational Linguistics.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Iyer, S.; Dandekar, N.; and Csernai, K. 2017. First Quora Dataset Release: Question Pairs. *https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs* .

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 4163–4174.

Kim, Y.; and Rush, A. M. 2016. Sequence-Level Knowledge Distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1317–1327.

Liu, X.; He, P.; Chen, W.; and Gao, J. 2019. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482* .

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*, 2227–2237. Association for Computational Linguistics.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*, 2383–2392. The Association for Computational Linguistics.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* .

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Sun, S.; Cheng, Y.; Gan, Z.; and Liu, J. 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)*, 4314–4323.

Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; and Zhou, D. 2020. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2158–2170.

Tan, X.; Ren, Y.; He, D.; Qin, T.; and Liu, T.-Y. 2019. Multilingual Neural Machine Translation with Knowledge Distillation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=S1gUsoR9YX.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *BlackboxNLP@EMNLP*, 353–355. Association for Computational Linguistics.

Warstadt, A.; Singh, A.; and Bowman, S. R. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics* 7: 625–641.

Wei, H.-R.; Huang, S.; Wang, R.; Dai, X.; and Chen, J. 2019. Online Distilling from Checkpoints for Neural Machine Translation. 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)*, 1932–1941.

Williams, A.; Nangia, N.; and Bowman, S. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1112–1122.

Wold, S.; Esbensen, K.; and Geladi, P. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2(1-3): 37–52.

Wu, Y.; Passban, P.; Rezagholizadeh, M.; and Liu, Q. 2020. Why Skip If You Can Combine: A Simple Knowledge Distillation Technique for Intermediate Layers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.