PEA-KD: PARAMETER-EFFICIENT AND ACCURATE KNOWLEDGE DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

How can we efficiently compress a model while maintaining its performance? Knowledge Distillation (KD) is one of the widely known methods for model compression. In essence, KD trains a smaller student model based on a larger teacher model and tries to retain the teacher model's level of performance as much as possible. However, existing KD methods suffer from the following limitations. First, since the student model is smaller in absolute size, it inherently lacks model complexity. Second, the absence of an initial guide for the student model makes it difficult for the student to imitate the teacher model to its fullest. Conventional KD methods yield low performance due to these limitations.

In this paper, we propose Pea-KD (Parameter-efficient and accurate Knowledge Distillation), a novel approach to KD. Pea-KD consists of two main parts: Shuffled Parameter Sharing (SPS) and Pretraining with Teacher's Predictions (PTP). Using this combination, we are capable of alleviating the KD's limitations. SPS is a new parameter sharing method that allows a greater model complexity for the student model. PTP is a KD-specialized initialization method, which can act as a good initial guide for the student. When combined, this method yields a significant increase in student model's performance. Experiments conducted on different datasets and tasks show that the proposed approach improves the student model's performance by 4.4% on average in four GLUE tasks, outperforming existing KD baselines by significant margins.

1 INTRODUCTION

How can we improve the accuracy of knowledge distillation (KD) with smaller number of parameters? KD uses a well-trained large teacher model to train a smaller student model. Conventional KD method (Hinton et al. (2006)) trains the student model using the teacher model's predictions as targets. That is, the student model uses not only the true labels (hard distribution) but also the teacher model's predictions (soft distribution) as targets. Since better KD accuracy is directly linked to better model compression, improving KD accuracy is valuable and crucial.

Naturally, there have been many studies and attempts to improve the accuracy of KD. Sun et al. (2019) introduced Patient KD which utilizes not only the teacher model's final output but also the intermediate outputs generated from the teacher's layers. Jiao et al. (2019) applied additional KD in the pretraining step of the student model. However, existing KD methods share the limitation of students having lower model complexity compared to their teacher models, since they are small in size. In addition, there are no proper initial guides for the student model, which is important especially when the student models are small. These limitations lead to insufficient accuracy of student models.

In this paper, we propose Pea-KD (Parameter-efficient and accurate Knowledge Distillation), a novel KD method designed especially for Transformer-based models (Vaswani et al. (2017)), which significantly improves the student model's accuracy. Pea-KD is composed of two modules, Shuffled Parameter Sharing (SPS) and Pretraining with Teacher's Predictions (PTP). When combined, these two methods alleviate the aforementioned KD's limitations and yield higher accuracy. Pea-KD is based on the following two main ideas.

1. We apply SPS in order to increase the effective model complexity of the student model while not increasing the number of parameters. SPS has two steps: 1) stacking the layers that share parameters, and 2) shuffling the parameters between shared pairs of layers. Doing so increases the model's effective complexity which enables the student to better replicate the teacher model and improves accuracy (details in Section 3.2).
2. We design an effective pretraining task called PTP for a student in KD. Through PTP, the student model obtains additional information about the teacher and the task itself, which helps the student acquire the teacher's knowledge more efficiently during the KD process (details in Section 3.3).

Throughout the paper we use PeaBERT (Parameter-efficient and accurate BERT), which is Pea-KD applied on BERT, as an example to investigate our proposed approach. We summarize our main contributions as follows:

- **Novel framework for KD.** We propose SPS and PTP, a novel parameter sharing method and a novel KD-specialized initialization method. These methods serve as a new framework for KD to significantly improve accuracy.
- **Performance.** When tested on four of the widely used GLUE tasks, PeaBERT improves student's accuracy up to 14.8%, and 4.4% on average compared to the original BERT model. PeaBERT also outperforms existing state-of-the-art KD baselines by 3.5% on average.
- **Generality.** Our proposed method Pea-KD can be applied to any transformer-based models and any classification tasks with minimal modification. Our method can thus be generally applied to many ongoing KD studies in Natural Language Processing.

The rest of the paper is organized as follows. Section 2 covers related works. Section 3 describes our proposed method in detail. Section 4 presents experimental results. Then we conclude in Section 5.

2 Related Work

Pretrained Language Models. The framework of first pre-training language models and then finetuning for downstream tasks has now become the industry standard for Natural Language Processing (NLP) models. Pretrained language models, such as BERT (Devlin et al. (2018)), XLNet (Yang et al. (2019)), RoBERTa (Liu et al. (2019)) and ELMo (Peters et al. (2018)) prove how powerful pretrained language models can be. Specifically, BERT is a language model consisting of multiple Transformer layers. By pretraining using Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), BERT has achieved the state-of-the-art performance on a wide range of NLP tasks, such as the GLUE benchmark (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016). However, these modern pretrained models are very large in size and contain millions of parameters, making them nearly impossible to apply on edge devices with limited amount of resources. In our work, we address this challenging problem by applying a novel KD method on the BERT model. Our approach can be easily applied to other transformer-based models as well.

Model Compression. As deep learning algorithms started getting adopted, implemented, and researched in diverse fields, high computation costs and memory shortage have started to become challenging factors. Especially in NLP, pretrained language models typically require a large set of parameters. This results in extensive cost of computation and memory. As such, Model Compression, which is to compress a model while preserving the performance as much as possible, has now become an important task for deep learning. There have already been many attempts to tackle this problem, including quantization (Gong et al. (2014)) and weight pruning (Han et al. (2015)). One promising approach is KD (Hinton et al. (2015)) which we focus on in this paper.

Knowledge Distillation (KD). As briefly covered in Section 1, KD transfers knowledge from a well-trained and large teacher model to a smaller student model. KD uses the teacher models predictions on top of the true labels to train the student model. It is proven through many experiments that the student model learns to imitate the soft distribution of the teacher models predictions, and ultimately performs better than learning solely from the original data. There have already been many attempts to compress BERT using KD. Patient Knowledge Distillation (Sun et al. (2019))

extracts knowledge not only from the final prediction of the teacher, but also from the intermediate layers. TinyBERT (Jiao et al. (2019)) uses a two-stage learning framework which applies knowledge distillation in both pretraining and task-specific finetuning. DistilBERT (Sanh et al. (2019)) uses half of the layers of BERT-base model and applies KD during pretraining and finetuning of BERT. Zhao et al. (2019) trains the student model with smaller vocabulary set and lower hidden state dimensions. Unfortunately, these existing methods do not give sufficient accuracy due to the student model's insufficient complexity and absence of a clear guideline for initialization. In this paper, we propose a new KD approach using parameter sharing and KD-specific initialization to alleviate the above issues. Our method improves the student model's performance significantly and shows the state-of-the-art performance among distillation methods on BERT.

Parameter Sharing. The idea of parameter sharing across different layers is a widely used idea for model compression. There have been several attempts to use parameter sharing in transformer architecture and BERT model. However, existing parameter sharing methods exhibit a large tradeoff between model performance and model size. They reduce the model's size significantly, but as a result, also suffer from a great loss in performance. In this paper, we propose a novel parameter sharing method which uses a shuffling mechanism to reduce this tradeoff, resulting in an improved performance using the same number of parameters.

3 Proposed Methods

We propose PeaBERT, a novel KD method applied on BERT that shows a higher KD accuracy with smaller number of parameters compared to existing methods. PeaBERT consists of two main modules: SPS and PTP, which together boost the student model's accuracy. In the following, we provide an overview of the main challenges in KD and our methods to address them in Section 3.1. We then discuss the precise procedures of SPS and PTP in Sections 3.2 and 3.3. Lastly, we explain our final method, PeaBERT and the training details in Section 3.4.

3.1 Overview

BERT-base model contains over 109 million parameters. Its extensive size makes model deployment often infeasible and computationally expensive in many cases, such as on mobile devices. As a result, industry practitioners commonly use a smaller version of BERT and apply KD. However, the existing KD methods entail the following challenges:

1. **Insufficient model complexity of the student model.** Since the student model contains fewer number of parameters than the teacher model, it's model complexity is also lower. The smaller and simpler the student model gets, the gap between the student and the teacher grows, making it increasingly difficult for the student to replicate the teacher model's accuracy. The limited complexity hinders the student model's performance. How can we enlarge the student model's complexity while maintaining the same number of parameters?
2. **Absence of proper initial guide for the student model.** Most of the existing KD methods do not consider the student model's initialization to be crucial. In most cases, a truncated version of pretrained BERT-base model is used. There is no widely accepted and vetted guide to selecting the student's initial state of the KD process. In reality, this hinders the student from efficiently reproducing the teacher's results. How can we effectively initialize the student model to achieve a better KD accuracy?

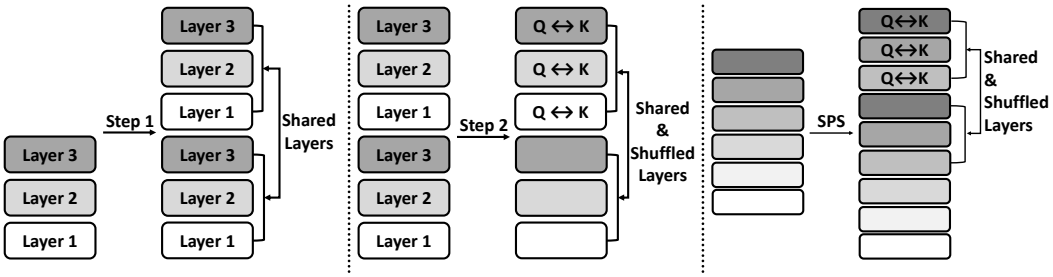
We propose the following main ideas to address the challenges:

1. **Shuffled Parameter Sharing (SPS): amplifying 'effective' model complexity of the student.** To address the complexity limitation, we introduce SPS. SPS increases the student's effective model complexity while using the same number of parameters (see details in Section 3.2). As a result, the SPS-applied student model achieves a better accuracy without running into the usual computational challenges.
2. **Pretraining with Teacher's Predictions (PTP): a novel pretraining task utilizing teacher's predictions for student initialization.** To address the limitation of the initial guide, we propose PTP, a novel pretraining method for the student by utilizing teacher

model's predictions. Through PTP, the student model pre-learns information about the teacher and the task itself. It helps the student better acquire the teacher's knowledge during the KD process (see details in Section 3.3).

The following subsections describe the precise procedures of SPS, PTP, and PeaBERT in detail.

3.2 SHUFFLED PARAMETER SHARING (SPS)



(a) SPS step 1 (b) SPS step 1 + step 2 (c) SPS for six-layer student

Figure 1: Graphical representation of SPS: (a) the first step of SPS, (b) the second step of SPS, and (c) modified SPS for a 6-layer student.

SPS improves student's effective model complexity while not increasing the number of parameters, addressing the complexity limitations of a typical KD. SPS is composed of the following two steps.

Step1. Paired Parameter Sharing. We start with doubling the number of layers in the student model. We then share the parameters between the bottom half and the upper half of the model, as graphically represented in Figure 1a. By doing so the model now has twice the number of layers and thus a higher effective model complexity while maintaining the same number of actual parameters used.

Step2. Shuffling. We shuffle the Query and Key parameters between the shared pairs. That is, for the shared upper half of layers we use the original Query parameters as Key parameters, and the original Key parameters as Query parameters. Typical parameter sharing methods use the exact same parameters in the exact same position several times which limits the model's representation power. Instead, our idea is to shuffle the Query and Key parameters between the shared pairs, which enriches the model capacity of the student to have a better representation power. We call this architecture SPS, which is depicted in Figure 1b. For the 6-layer student case we slightly modify the architecture as in Figure 1c (we apply SPS on the top 3 layers only).

In sum, the SPS model has the same number of parameters as the original student model but has much greater effective model complexity. In Section 4, we validate through experiments that step1 (Paired Parameter Sharing) and step2 (Shuffling) indeed increase the effective model complexity, directly contributing to accuracy improvement.

3.3 PRETRAINING WITH TEACHER'S PREDICTIONS (PTP)

There can be several candidates for KD-specialized initialization. We propose a pretraining approach called PTP, and experimentally show that it improves KD accuracy significantly. The intuition here is that by PTP, the student model acquires additional knowledge about both the teacher as well as the downstream task. With this additional information, the student obtains the teacher's knowledge more efficiently during the actual KD process.

Most of the previous studies on KD do not elaborate on the initialization of the student model. There are some studies that use a pretrained student model as an initial state, but those pretraining tasks are irrelevant to either the teacher model or the downstream task. To the best of our knowledge, our study is the first case that pretrains the student model with a task relevant to the teacher model and its downstream task. PTP consists of the following two steps.

Step 1. Creating artificial data based on the teacher's predictions (PTP labels).

We first input the training data in the teacher model and collect the teacher model's predictions. We then define "confidence" as the following. We apply softmax function to the teacher model's

predictions, and the maximum value of the predictions is defined as the confidence. Next, with a specific threshold "t" (a hyperparameter between 0.5 and 1.0), we assign a new label to the training data according to the rules listed in Table 1. We call these new artificial labels PTP labels.

Table 1: Assigning new PTP labels to the training data.

PTP label	Teacher's prediction correct	confidence $>$ t
confidently correct	True	True
unconfidently correct	True	False
confidently wrong	False	True
unconfidently wrong	False	False

Step 2. Pretrain the student model to predict the PTP labels. Using the artificial PTP labels (data x, PTP label) we created, we now pretrain the student model to predict the PTP label when x is provided as an input. In other words, the student model is trained to predict the PTP labels given the downstream training dataset. We train the student model until convergence.

Once these two steps are complete, we use this PTP-pretrained student model as the initial state for finetuning on the downstream task.

3.4 PeaBERT: SPS and PTP combined

3.4.1 Overall Framework of PeaBERT

PeaBERT applies SPS and PTP together on BERT for its maximum impact on the performance. Given a student model, PeaBERT first transforms it into an SPS model and applies PTP. Once PTP is completed, we use this model as the initial state of the student model during the KD process. The overall framework of PeaBERT is depicted in Figure 2.

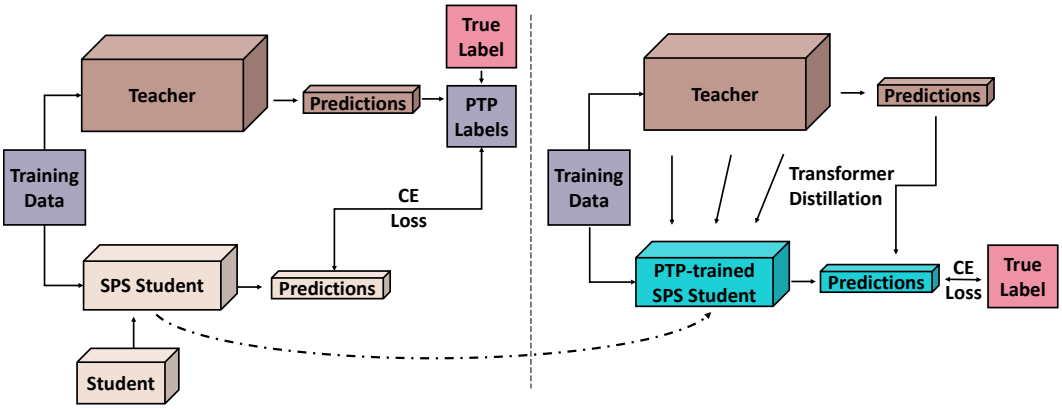


Figure 2: The overall framework of PeaBERT. The left half represents applying SPS and PTP to a student model. The right half represents the learning baseline.

3.4.2 Learning Details of PeaBERT

For the starting point of the KD process, a well finetuned teacher model should be used. We use the 12 layer BERT-base model as the teacher. The learned parameters are denoted as:

$$\hat{\theta}^t = \arg\min_{\theta^t} \sum_{i \in N} \mathcal{L}_{CE}(y_i, \sigma(z_t(x_i; \theta^t))) \tag{1}$$

where the θ^t denotes parameters of the teacher, σ denotes the softmax function, x_i denotes the training data, z_t denotes the teacher model's output predictions, y_i denotes the true labels, and $\mathcal{L}_{CE}$ denotes cross-entropy loss. We then pretrain the student model with PTP labels using the following loss:

$$\hat{\theta}^s_{\mathcal{PTP}} = \arg\min_{\theta^s} \sum_{i \in N} \mathcal{L}_{CE}(y_i^{\mathcal{PTP}}, \sigma(z_s(x_i; \theta^s))) \tag{2}$$

where $y_i^{\mathcal{PTP}}$ denotes the PTP labels and the subscript s denotes the student model. When PTP is completed, we use the $\hat{\theta}^s_{\mathcal{PTP}}$ as the initial state of the KD process. Our KD process uses three loss terms in total: a cross-entropy loss between student model's final predictions and ground-truth labels, a Kullback-Leiber divergence loss between the final predictions of student and teacher models, and a mean-squared error loss between the intermediate layers of student and teacher models.

The loss function is as follows:

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{CE}(y_i, \sigma(z_s(x_i;\theta^s)) + \alpha\mathcal{L}_{\mathcal{KL}}(\sigma(z_s(x_i;\theta^s), \sigma_T(z_t(x_i;\hat{\theta}^t)))) + \beta \sum_{(k_1,k_2)\in\mathcal{K}} ||z_s^{k1} - z_t^{k2}||_2^2 \tag{3}$$

where $\mathcal{L}_{\mathcal{KL}}$ denotes the Kullback-Leibler divergence loss, $\mathcal{K}$ denotes the index pairs of layers we use for intermediate layer-wise distillation, and z^k denotes the output logits of the k-th layer. α and β are hyperparameters.

Note that during the KD process, we use a softmax-temperature T which controls the softness of teacher model's output predictions (Hinton et al. (2015)):

$$\sigma_T(z_i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}} \tag{4}$$

4 EXPERIMENTS

We discuss experimental results to assess the effectiveness of our proposed method. Our goal is to answer the following questions.

- **Q1. Overall performance.** How does PeaBERT perform compared to the currently existing KD methods? (Section 4.2)
- **Q2. Effectiveness of SPS.** To what extent does SPS improve the effective complexity of the student model, without increasing the number of parameters? (Section 4.3)
- **Q3. Effectiveness of PTP.** Is the new PTP-training a good initialization method? Compared to the conventionally-used truncated version of the BERT-base model, what is the impact that our PTP initialization method has on the model performance? (Section 4.4)

4.1 EXPERIMENTAL SETTINGS

Datasets. We use four of the most widely used datasets in the General Language Understanding Evaluation (GLUE) benchmark (Wang et al. (2018)): SST-2[1], QNLI[2], RTE[3], and MRPC[4]. For sentiment classification, we use the Stanford Sentiment Treebank (SST-2) (Socher et al. (2013)). For natural language inference, we use QNLI(Rajpurkar et al. (2016)) and RTE. For paraphrase similarity matching, we use Microsoft Research Paraphrase Corpus (MRPC) (Dolan & Brockett (2005)). Specifically, SST-2 is a movie review dataset with binary annotations where the binary label indicates positive and negative reviews. QNLI is a task for predicting whether a pair of a question and an answer is an entailment or not. RTE is based on a series of textual entailment challenges and MRPC contains pairs of sentences and corresponding labels, where the labels indicate the semantic equivalence relationship between the sentences in each pair.

Competitors. We use Patient Knowledge Distillation (PatientKD, Sun et al. (2019)) as our baseline learning method to compare and quantify the effectiveness of our proposed PeaBERT. Patient Knowledge Distillation is one of the most widely used baselines, and is a variant of original Knowledge Distillation method (Hinton et al. (2006)). We conduct experiments on BERT model (Devlin et al. (2018)) and compare the results of PeaBERT to the original BERT. In addition, we compare the results with other state-of-the-art BERT-distillation models, including DistilBERT(Sanh et al. (2019)) and TinyBERT(Jiao et al. (2019)).

Training Details. We use the 12-layer original BERT model (Devlin et al. (2018)) as a teacher model and further finetune the teacher for each task independently. The student models are created

[1]https://nlp.stanford.edu/sentiment/index.html

[2]https://rajpurkar.github.io/SQuAD-explorer/

[3]https://aclweb.org/aclwiki/RecognizingTextualEntailment

[4]https://www.microsoft.com/en-us/download/details.aspx?id=52398

Table 2: Overall results of PeaBERT compared to the state-of-the-art KD baseline, PatientKD. The results are evaluated on the test set of GLUE official benchmark. The subscript numbers denote the number of independent layers of the student.

Method	RTE (Acc)	MRPC (F1)	SST-2 (Acc)	QNLI (Acc)	Avg
$BERT_1$-PatientKD	52.8	80.6	83.6	64.0	70.3
$PeaBERT_1$	53.0	81.0	86.9	78.8	75.0
$BERT_2$-PatientKD	53.5	80.4	87.0	80.1	75.2
$PeaBERT_2$	64.1	82.7	88.2	86.0	80.3
$BERT_3$-PatientKD	58.4	81.9	88.4	85.0	78.4
$PeaBERT_3$	64.5	85.0	90.4	87.0	81.7

Table 3: PeaBERT in comparison to other state-of-the-art competitors in dev set. The cited results of the competitors are from the official papers of each method. For accurate comparison, model dimensions are fixed to six layers across all models compared.

Method	# of parameters	RTE (Acc)	MRPC (F1)	SST-2 (Acc)	QNLI (Acc)	Avg
DistilBERT	42.6M	59.9	87.5	91.3	89.2	82.0
TinyBERT	42.6M	70.4	90.6	93.0	91.1	86.3
PeaBERT	42.6M	73.6	92.9	93.5	90.3	87.6

using the same architecture as the original BERT, but the number of layers is reduced to either 1, 2, 3, or 6 depending on experiments. That is, we initialize the student model using the first n-layers of parameters from the pretrained original BERT obtained form Google's official BERT repo[5]. We use the standard baseline PatientKD, and the following hyperparameter settings: training batch size from $\{32, 64\}$, learning rate from $\{1, 2, 3, 5\}\cdot 10^{-5}$, number of epochs from $\{4, 6, 10\}$, α between $\{0.1$ and $0.7\}$, and β between $\{0$ and $500\}$.

4.2 Overall Performance

We summarize the performance of our proposed PeaBERT against the standard baseline PatientKD in Table 2. We also compare the results of PeaBERT against the competitors DistilBERT and TinyBERT in Table 3. We observe the following from the results.

First, we see from Table 2 that PeaBERT consistently yields higher performance in downstream tasks across all three model sizes. Notably, PeaBERT shows improved accuracy by an average of 4.7% for the 1-layer student, 5.1% for the 2-layer student, and 3.3% for the 3-layer student (note that we use F1 score for MRPC when calculating the average accuracy throughout the experiments). It shows maximum improvement of 14.8% in $PeaBERT_1$-QNLI. These results strongly validate the effectiveness of PeaBERT across varying downstream tasks and student model sizes.

Second, using the same number of parameters, PeaBERT outperforms the state-of-the-art KD baselines DistilBERT and TinyBERT, by 5.6% and 1.3% on average. We use a 6-layer student model for this comparison. An inspiring advantage of PeaBERT is that it achieves remarkable accuracy improvement just by using the downstream dataset without touching the original pretraining tasks. Unlike its competitors DistilBERT and TinyBERT, PeaBERT does not touch the original pretraining tasks, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). This reduces training time significantly. For example, DistilBERT took approximately 90 hours with eight 16GB V100 GPUs while PeaBERT took a minimum of one minute ($PeaBERT_1$ with RTE) to a maximum of one hour ($PeaBERT_3$ with QNLI) using just two NVIDIA T4 GPUs.

Finally, another advantage of PeaBERT is that it can be directly applied to other transformer-based models with minimal modifications. The SPS method can be directly applied to any transformer-based models, and the PTP method can be applied to any classification task.

4.3 Effectiveness of SPS

We perform an ablation study to verify our first main claim that SPS enlarges the student model's complexity and improves accuracy without increasing the number of parameters. We compare three

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

Table 4: An ablation study to validate each step of SPS. A three-layer student model is used. The results are derived using GLUE dev set.

Method	# of parameters	RTE (Acc)	MRPC (F1)	SST-2 (Acc)	QNLI (Acc)	Avg
$BERT_3$	21.3M	61.4	84.3	89.4	84.8	80.0
SPS-1	21.3M	63.5	85.8	89.6	85.5	81.1
SPS-2	21.3M	68.6	86.8	90.2	86.5	83.0

models $BERT_3$, SPS-1, and SPS-2. $BERT_3$ is the original BERT model with 3 layers, which applies none of the SPS steps. SPS-1 applies only the first step (paired parameter sharing) in the SPS process to $BERT_3$. SPS-2 applies both the first step and the second step (shuffling) to $BERT_3$.

The results are summarized in Table 4. Compared to the original $BERT_3$, SPS-1 shows an improved accuracy in all the downstream datasets with an average of 1.1%. Comparing SPS-1 with SPS-2, we note that SPS-2 consistently shows even better performance with an average of 1.9%. We conclude that both steps of the SPS process increase the effective model complexity of student model, without increasing the number of parameters.

4.4 Effectiveness of PTP

Table 5: An ablation study to verify the effectiveness of PTP. The results are derived using GLUE dev set.

Model	# of parameters	RTE (Acc)	MRPC (F1)	SST-2 (Acc)	QNLI (Acc)	Avg
$PeaBERT_3$-p	21.3M	68.6	86.8	90.2	86.5	82.9
$PeaBERT_3$	21.3M	70.8	88.0	91.2	87.1	84.3

We perform an ablation study to validate the effectiveness of using PTP as an initial guide for the student model. Similarly to how we validate the effectiveness of SPS in Section 4.3, we use $BERT_3$ as our representative model. We compare our $PeaBERT_3$ to its variant $PeaBERT_3$-p. $PeaBERT_3$-p is the PeaBERTmodel without PTP. This essentially applies only SPS to $BERT_3$. $PeaBERT_3$ goes one step further and applies both SPS and PTP.

The results are reported in Table 5. Note that applying PTP increases performance across all four of the datasets with an average of 1.4%, proving the effectiveness of using PTP to increase model performance. This validates our second main claim that initializing a student model with KD-specialized method (PTP) prior to applying KD improves accuracy. As existing KD methods do not place much emphasis on the initialization process, this finding highlights a potentially major, undiscovered path to improving model accuracy. Further and deeper researches related to KD-specialized initialization could be promising.

5 Conclusion

In this paper, we propose Pea-KD, a new KD method for transformer-based distillation, and show its efficacy. Our goal is to address and reduce the limitations of the currently available KD methods: insufficient model complexity and absence of proper initial guide for the student. We first introduce SPS, a new parameter sharing approach that uses a shuffling mechanism, which enhances the complexity of the student model while using the same number of parameters. We then introduce PTP, a KD-specific initialization method for the student model. Our proposed PeaBERT comes from applying these two methods SPS and PTP on BERT. Through extensive experiments conducted using multiple datasets and varying model sizes, we show that our method improves KD accuracy by an average of 4.4% on the GLUE test set. We also show that PeaBERT works well across different datasets, and outperforms the original BERT as well as other state-of-the-art baselines on BERT distillation by an average of 3.5%.

As a future work, we plan to delve deeper into the concept of KD-specialized initialization of the student model. Also, since PTP and SPS are independent processes on their own, we plan to combine PTP and SPS with other model compression techniques, such as weight pruning and quantization.

REFERENCES

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pp. 1135–1143, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

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

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

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5753–5763, 2019.

Sanqiang Zhao, Raghav Gupta, Yang Song, and Denny Zhou. Extreme language model compression with optimal subwords and shared projections. *arXiv preprint arXiv:1909.11687*, 2019.