Co-training and Co-distillation for Quality Improvement and Compression of Language Models

Anonymous ACL submission

Abstract

The increasing size and computational complexity of pre-trained language models (PLMs) can make them difficult to use in serving downstream tasks under resource-constrained environments. While conventional knowledge distillation (KD) can compress PLMs by transferring knowledge from a larger model to a smaller model, this typically results in a trade-off between inference efficiency and performance. In this work, we propose a novel Co-Training and Co-Distillation (CTCD) framework that improves the quality of PLMs while increasing their inference efficiency. Our approach trains different-sized models together and allows them to distill knowledge from each other. We demonstrate that the proposed co-distillation improves the quality of the teacher model, which in turn improves the quality of the student model. Furthermore, we introduce a Community KD that consists of a single teacher and two students where each student learns from the teacher and the other student. The results of our experiments show that the Community KD is effective at compressing PLMs and improving their quality, outperforming the conventional one-way KD method by 1.2% on the GLUE benchmark. Code is available at https://anonymous.4open.science/r/CTCD-1.

1 Introduction

In recent years, the use of pre-trained language models (PLMs) and rapid adaptation to downstream tasks has been a dominant strategy in the NLP community (Radford et al., 2019; Yang et al., 2019; Dai et al., 2019; Shoeybi et al., 2019; Li et al., 2020; Brown et al., 2020). However, as PLMs become larger to achieve higher performance, their high computational cost due to their large size becomes a constraint in serving resource-limited real-world applications. As a result, there is a need for lightweight, compressed PLMs that are accurate.

One technique that has been used to address the challenge of large PLMs is knowledge distillation (KD), where a smaller student model learns (or distills) knowledge from a larger pre-trained teacher model (Hinton et al., 2015; Romero et al., 2015). There have been several studies on distilling pre-trained large language models into compact models (Turc et al., 2019; Tsai et al., 2019; Tang et al., 2019; Jiao et al., 2020; Sanh et al., 2019; Sun et al., 2019; Wang et al., 2020b,a).

However, in such "*one-way*" distillation, the teacher model plays a crucial role in acquiring knowledge, and the smaller model generally cannot keep up with the performance of the larger models. This raises an interesting question: Is it possible to compress a model via KD without sacrificing performance?

In this work, we propose a novel framework called Co-Training and Co-Distillation (CTCD) for improving the both quality and serving efficiency of a language model. Our approach trains two different-sized models together and allows them to distill knowledge from each other during the pre-training stage. This enables the smaller student model to achieve the same quality as the original teacher model. We focus on task-agnostic distillation, which allows the pre-trained and distilled compact model to be directly fine-tuned on various downstream tasks.

CTCD differs from conventional Knowledge distillation (KD) in that it involves a two-way process of knowledge transfer between two models, rather than simply transferring knowledge from a larger, pre-trained teacher model to a smaller student model. During such a one-way distillation of conventional KD, the pre-trained teacher is no longer trained. Therefore, there is no further improvement in the performance of teacher model quality. However, in co-training and co-distillation, both the teacher and student models are learned together, allowing for further improving the performance of the teacher model, which in turn, benefits the performance of the student model. Specifically,

we have demonstrated that our approach is effective at answering two important questions: 1) Does learning from small student models provide a performance benefit to the teacher model? and 2) Does the performance improvement of the teacher model improve the performance of the student model?

In addition to our CTCD, we introduce a novel Community KD framework that combines co-training and co-distillation with conventional KD. Community KD distills knowledge from two students to each other, as well as from the pre-trained teacher to each student, during the co-training of the students. We show that distilling from the other student as well as the pre-trained teacher is better than only one-way distillation from the pre-trained teacher, as is done in conventional KD. Note that the inference cost on downstream tasks is the same as in conventional KD, as we take one of the students distilled by Community KD and adapt it to downstream tasks after pre-training. Our experiments show that Community KD is effective at improving the performance and efficiency of PLMs.

We validate the effectiveness and efficiency of our CTCD on the GLUE benchmark (Wang et al., 2019), which contains different language understanding tasks. We fine-tune the distilled PLMs on each downstream task and evaluate their performance. In our experiments, the model compressed by CTCD obtained 1.3% higher gain than the original large model, showing that our approach can improve model quality and inference efficiency concurrently. Interestingly, we observe that the larger model benefits from the distillation of the smaller model, and the performance gain of the larger model leads to a further performance gain of the smaller model. Finally, empirical evaluations show that using our proposed Community KD to distill from the pre-trained BERT-base to a 6-layer student model improves student quality by 1.2% compared to using the conventional KD on the GLUE benchmark.

In summary, our contributions are as follows:

- We propose a novel knowledge distillation framework called Co-Training and Co-Distillation (CTCD) in improving the quality of transformer-based models while compressing them, at the pre-training phase.

- In our experiments, we show that the performance improvement of a larger model through distillation leads to a further performance improvement of a smaller model, demonstrating that our CTCD framework effectively improves the performance and efficiency of language models concurrently.

- We provide insights on adjusting loss weights and the length of the training phase for the effective application of CTCD.

- We introduce a novel Community KD method for compressing pre-trained language models, which outperforms the conventional KD method, by 1.2% on the GLUE benchmark.

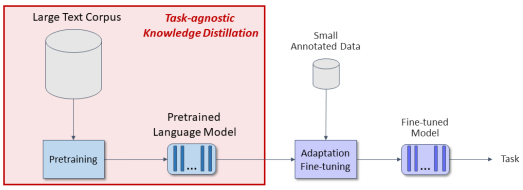


Figure 1: **Task-agnostic KD method** The proposed framework and DistilBERT (Sanh et al., 2019) are task-agnostic KD methods that perform knowledge distillation during the pre-training stage. This allows the distilled model to be deployed and fine-tuned on various downstream tasks.

2 Related Work

One-way Knowledge Distillation Knowledge distillation (KD) (Hinton et al., 2015) is a model compression technique in which a smaller student model distills knowledge from a larger, pre-trained teacher model. Recently, many researchers have explored KD at different training stages of a language model to address its high computational complexity resulting from its increasing size. This includes task-agnostic KD methods for the pre-training stage (Sanh et al., 2019; Wang et al., 2020b,a), task-specific KD method for the fine-tuning stage (Sun et al., 2019), and both of pre-training and fine-tuning stages (Jiao et al., 2020). Additionally, Wang et al. (2020b,a) have redesigned the architectures of student language models. However, the one-way distillation process used in KD can lead to a loss of knowledge, resulting in smaller models that generally have difficulty matching the performance of larger models, leading to performance degradation. In contrast, our CTCD can improve the quality of both teacher and student models, making it possible for the student to achieve the same quality as the original teacher model.

Reversed Knowledge Distillation Recently, researchers (Yuan et al., 2020; Qin et al., 2022) have demonstrated that reversed Knowledge Distillation

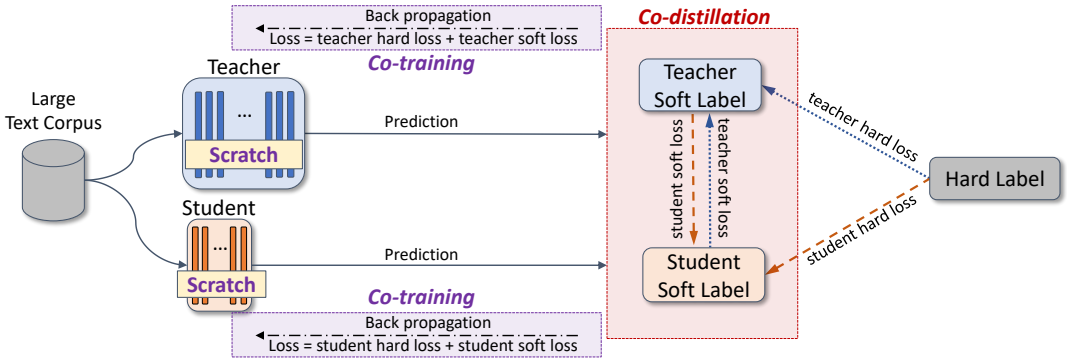


Figure 2: **Co-Training and Co-Distillation (CTCD)** During training, the teacher and student models can learn more effectively by comparing their prediction outputs not only against the ground truth but also against each other's predictions. We refer to the former as a "hard" loss and the latter as a "soft" loss. For example, the student soft loss captures the distance between the student's prediction and the teacher's prediction, and vice versa. This co-training and co-distillation approach improve the performance of the teacher model, which in turn benefits the performance of the student model.

(reversed KD), which transfers knowledge from a smaller or poorer model to a larger model, can improve the performance of the student model. In particular, Qin et al. (2022) investigated the application of reversed KD in the PLMs, showing that a larger model can benefit from a poorer and pre-trained model for a specific downstream task.

Inspired by the success of reversed KD, we design a co-distillation framework that includes reversed KD to improve the performance of the teacher model by distilling knowledge from the smaller student model. Unlike existing reversed KD methods, which are limited to improving the performance of the larger model, our proposed co-distillation framework can achieve both performance improvement and model compression, by showing a better-quality teacher leads to a better-quality student.

3 Co-training and Co-distillation

We first introduce the concepts of co-training and co-distillation briefly:

Co-training trains two (different-sized) models (e.g., a teacher and student) concurrently with the goal of achieving similar model quality.

Co-distillation transfers knowledge in both directions between two models (e.g., a teacher and student), during co-training.

Figure 2 illustrates how co-trained models learn together by comparing their prediction outputs against predictions from each other and to the hard labels (or "ground truth"). We refer to the former as a "soft" loss and the latter as a "hard" loss. For instance, the soft loss of the student model measures the accuracy of the student's prediction by considering the teacher's prediction as a soft label, and vice versa.

Task Formulation Suppose that we are given a classification task with K classes. For each training instance x and its ground truth label y, we denote that the ground truth distribution over the labels is $q(k|x)$ ($q(k)$ for simplicity) where for each label $k \in \{1...K\}$, $q(y) = 1$ and $q(k) = 0$ for all $k \neq y$. For each x, the teacher model $t_{\boldsymbol{\phi}}$ parameterized by $\boldsymbol{\phi}$ and the student model $s_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ predict the probability of each label k as $p^{\tau}_{\boldsymbol{\phi}}(k|x)$ and $p^{\tau}_{\boldsymbol{\theta}}(k|x)$, respectively as follows:

$$p^{\tau}_{\boldsymbol{\phi}}(k|x) = f(\boldsymbol{z}^t) = \frac{\exp(z^t_k/\tau)}{\sum_{i=1}^{K}\exp(z^t_i/\tau)}$$

$$p^{\tau}_{\boldsymbol{\theta}}(k|x) = f(\boldsymbol{z}^s) = \frac{\exp(z^s_k/\tau)}{\sum_{i=1}^{K}\exp(z^s_i/\tau)}$$

where f is the softmax function, $\boldsymbol{z}^t = \{z^t_i\}_{i=1}^{K} = t_{\boldsymbol{\phi}}(x)$ is the output logit of the teacher model, $\boldsymbol{z}^s = \{z^s_i\}_{i=1}^{K} = s_{\boldsymbol{\theta}}(x)$ is the output logit of the student model, and τ is the temperature to soften $p_{\boldsymbol{\phi}}(k)$ and $p_{\boldsymbol{\theta}}(k)$.

The proposed objective $\mathcal{L}_{CTCD}(\boldsymbol{\theta}, \boldsymbol{\phi})$ consists of a normal KD objective $\mathcal{L}_{KD:t\to s}$ to distill knowledge from the teacher model to the student model and a reversed KD objective $\mathcal{L}_{ReKD:s\to t}$ to distill knowledge from the student model to the teacher model, during their co-training.

Teacher $\rightarrow$ Student The normal KD objective $\mathcal{L}_{KD:t\to s}(\boldsymbol{\theta}, \boldsymbol{\phi})$ aims to train the student model by minimizing a weighted sum of the cross-entropy loss $H(q, p_{\boldsymbol{\theta}})$ between the ground truth q and student prediction $p_{\boldsymbol{\theta}}$ and Kullback-Leibler diver-

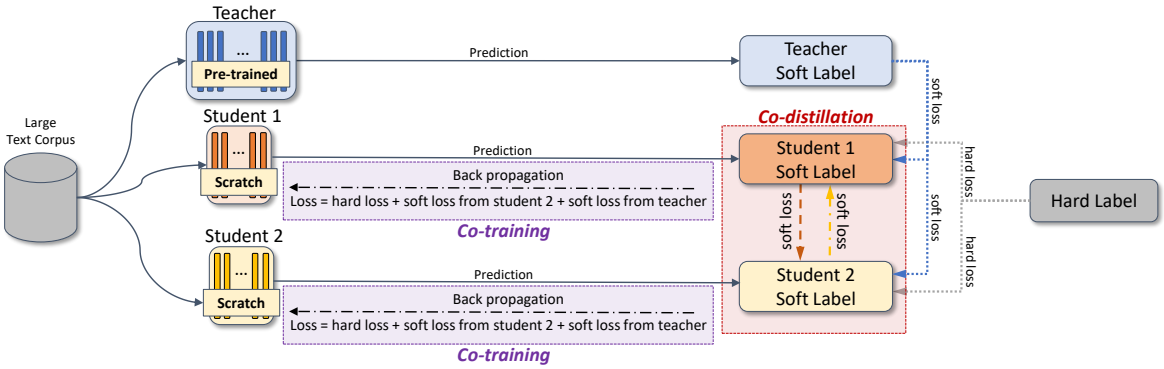


Figure 3: **Community KD** Different from conventional KD, where each student learns from the prediction of the pre-trained teacher only, the proposed approach learns each student from both prediction of the pre-trained teacher and prediction of another student during co-training. Note that since we take one of the pre-trained students to adapt it to downstream tasks, the inference cost is the same as the student training with the conventional KD.

gence (KL divergence) $D(p^\tau_{\boldsymbol{\phi}}, p^\tau_{\boldsymbol{\theta}})$ between the predictions of the student and the teacher as follows:

$$\mathcal{L}_{KD}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \alpha_h \cdot H(q, p_{\boldsymbol{\theta}}) + \alpha_s \cdot D(p^\tau_{\boldsymbol{\phi}}, p^\tau_{\boldsymbol{\theta}}) \tag{1}$$

where

$$H(q, p_{\boldsymbol{\theta}}) = -\sum_{k=1}^{K} q(k) \log(p_{\boldsymbol{\theta}}(k)),$$

$$D(p^\tau_{\boldsymbol{\phi}}, p^\tau_{\boldsymbol{\theta}}) = \sum_{k=1}^{K} p^\tau_{\boldsymbol{\phi}}(k) \cdot \log \frac{p^\tau_{\boldsymbol{\phi}}(k)}{p^\tau_{\boldsymbol{\theta}}(k)},$$

α_h and α_s are weighting hyper-parameter values for the cross-entropy loss and KL divergence, respectively. We regard the cross-entropy loss $H(q, p_{\boldsymbol{\theta}})$ as the hard loss for the student model, KL divergence $D(p^\tau_{\boldsymbol{\phi}}, p^\tau_{\boldsymbol{\theta}})$ as the soft loss for the student model, and following BERT (Devlin et al., 2019), $H(q, p_{\boldsymbol{\theta}})$ denotes the Masked Language Modeling loss (MLM loss). In the KD objective, we consider the teacher parameters $\boldsymbol{\phi}$ as constant since we only train the student parameters $\boldsymbol{\theta}$ while the teacher model $t_{\boldsymbol{\phi}}$ is fixed:

$$\mathcal{L}_{KD:t\to s}(\boldsymbol{\theta}, \text{StopG}(\boldsymbol{\phi})) \tag{2}$$

where $\text{StopG}(x)$ denotes that we do not compute the gradient of x. In the conventional KD method, Equation (2) is the final objective to learn the student model only with the pre-trained teacher model.

Student → Teacher Different from such a one-way KD method, we introduce the reversed KD objective $\mathcal{L}_{ReKD:s\to t}(\text{StopG}(\boldsymbol{\theta}), \boldsymbol{\phi})$ to train the teacher model $t_{\boldsymbol{\phi}}$ as follows:

$$\mathcal{L}_{ReKD:s\to t}(\text{StopG}(\boldsymbol{\theta}), \boldsymbol{\phi}) = \beta_h \cdot H(q, p_{\boldsymbol{\phi}}) + \beta_s \cdot D(p^\tau_{\boldsymbol{\theta}}, p^\tau_{\boldsymbol{\phi}}) \tag{3}$$

where β_h and β_s are weighting hyper-parameter values of the hard loss $H(q, p_{\boldsymbol{\phi}})$ and soft loss $D(p^\tau_{\boldsymbol{\theta}}, p^\tau_{\boldsymbol{\phi}})$ for the teacher model, respectively. By minimizing KL divergence $D(p^\tau_{\boldsymbol{\theta}}, p^\tau_{\boldsymbol{\phi}})$ between the predictions of the student model ($p^\tau_{\boldsymbol{\theta}}$) and the teacher model ($p^\tau_{\boldsymbol{\phi}}$), the teacher model learns from the student model. In the reversed KD objective, we only train the teacher model by applying $\text{StopG}(x)$ to the gradient of the student parameters $\boldsymbol{\theta}$.

Co-training With the Equations (2) and (3), we get the final objective $\mathcal{L}_{CTCD}(\boldsymbol{\theta}, \boldsymbol{\phi})$ as follows:

$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \underset{\boldsymbol{\theta}, \boldsymbol{\phi}}{\text{argmin}}\, \mathcal{L}_{CTCD}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}_{KD}(\boldsymbol{\theta}, \text{StopG}(\boldsymbol{\phi})) + \mathcal{L}_{ReKD}(\text{StopG}(\boldsymbol{\theta}), \boldsymbol{\phi}) \tag{4}$$

Adapting to Downstream Task After model co-training/-distillation, the trained smaller (student) model $s_{\boldsymbol{\theta}^*}$ with trained parameter $\boldsymbol{\theta}^*$ can be deployed for multiple downstream tasks to improve inference efficiency. To fine-tune the model for a specific downstream task, we adapt the trained parameter $\boldsymbol{\theta}^*$ using the dataset for that task.

Discussion with Conventional KD Our `CTCD` involves mutual knowledge distillation between the teacher and student models during training, leading to improved performance for both models. Unlike in conventional knowledge distillation, where the pre-trained teacher does not continue learning, this approach allows for the potential to further enhance the performance of the student model through the improved quality of the teacher model.

4 Community KD

Here we introduce an advanced co-training and co-distillation application named Community KD,

as shown in Figure 3. It consists of a pre-trained teacher t_{ϕ^*} with pre-trained parameters ϕ^* and two students s_{θ_1} and s_{θ_2} parameterized by $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$, respectively. During the co-training of two students, each student learns from the hard labels, soft labels generated from the pre-trained teacher predictions, and soft labels generated from other student predictions. In other words, we conduct one-way knowledge distillation from the pre-trained teacher to each student by minimizing KL divergence between the teacher prediction and predictions of each student $D(p^\tau_{\boldsymbol{\theta_1}}, p^\tau_{\phi^*})$ and $D(p^\tau_{\boldsymbol{\theta_2}}, p^\tau_{\phi^*})$ and co-distillation between students in both directions by minimizing $\mathcal{L}_{CTCD}(\boldsymbol{\theta_1}, \boldsymbol{\theta_2})$. The final objective $\mathcal{L}_{CM}(\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \text{StopG}(\phi^*))$ is as follows:

$$\boldsymbol{\theta_1^*}, \boldsymbol{\theta_2^*} = \underset{\boldsymbol{\theta_1}, \boldsymbol{\theta_2}}{\text{argmin}}\, \mathcal{L}_{CM}(\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \text{StopG}(\phi^*)) = \mathcal{L}_{CTCD}(\boldsymbol{\theta_1}, \boldsymbol{\theta_2}) + D(p^\tau_{\boldsymbol{\theta_1}}, p^\tau_{\phi^*}) + D(p^\tau_{\boldsymbol{\theta_2}}, p^\tau_{\phi^*}) \quad (5)$$

We select **one** of the two students distilled by Community KD ($\boldsymbol{\theta}^* = \boldsymbol{\theta_1^*}$ or $\boldsymbol{\theta}^* = \boldsymbol{\theta_2^*}$) and fine-tune the selected single student $s_{\boldsymbol{\theta}^*}$ for downstream tasks, resulting that the inference cost does not increase compared with the conventional KD method.

5 Experiment

We present a comprehensive analysis of the proposed CTCD method through empirical experiments. In Section 5.1, we validate the proposed CTCD method by comparing the performance of distilled students on the GLUE benchmark (Wang et al., 2019). In Section 5.2, we analyze the impact of co-distillation by adjusting loss weights for the soft losses of a student and a teacher. In Section 5.3, we study the impact of training length on CTCD method, allowing us to determine the optimal training length for CTCD method. In Section 5.4, we demonstrate the efficacy of Community KD by comparing the performance of a model compressed by Community KD to a baseline model compressed by conventional KD on the GLUE benchmark.

Implementation details In our experiments, we use a learning rate of 5e-4, linear warm up of 5%, AdamW optimizer (Loshchilov and Hutter, 2019), and batch size of 128 with A100 GPUs for pre-training. We train the teacher and student models from scratch for 10 epochs in Section 5.1 and Section 5.2. In Section 5.2, we train the models for 10 epochs and 20 epochs to examine the impact of training length on the performance of CTCD method. In Section 5.4, we train the models for 3 epochs after parameter remapping, which is the same as in DistilBERT (Sanh et al., 2019). We use automatic mixed precision (AMP) of PyTorch (Paszke et al., 2019) to accelerate training for all our models. This allows us to train our models more efficiently and reduce training time.

		Performance	GAP w/ Teacher
	Original Teacher	78.06	-
Student	One-way Distil. (10 epoch)	77.46	-0.60
	CTCD (10 epoch)	**77.94**	**-0.12**
	One-way Distil. (20 epoch)	78.39	+0.33
	CTCD (20 epoch)	**79.12**	**+1.66**

Table 1: **Average performance on dev sets of GLUE benchmark** The student distilled by CTCD significantly outperforms the original teacher trained using the stand-alone method, achieving a higher gain of **1.66**.

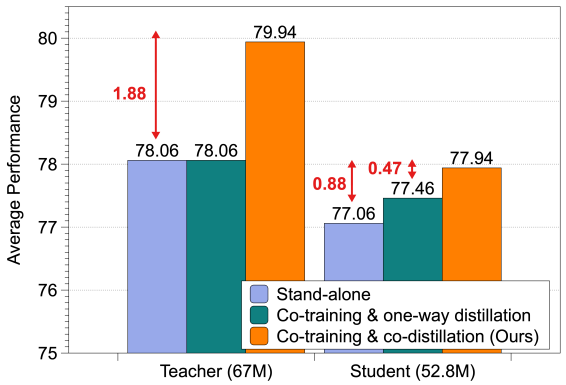
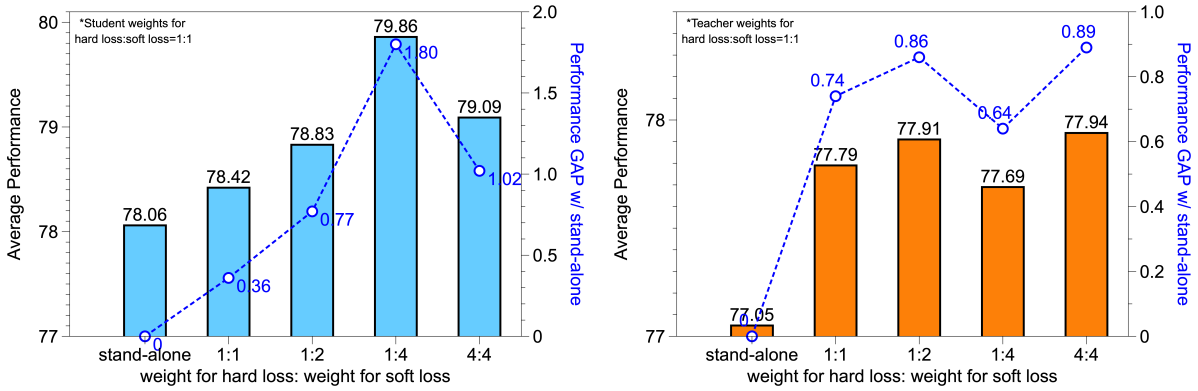


Figure 4: **Average performance on dev sets of GLUE benchmark** Learning from the student improves the performance of the teacher by an average of 1.88. Such improvement in teacher performance leads to improvement in student quality from 77.46 to 77.94.

Training Time For the one-way distillation, we need 1 GPU day to train the teacher for 10 epochs and 1.3 GPU days to distil knowledge from the pre-trained teacher to the student for another 10 epochs. For CTCD (Ours), it takes 3 GPU days to train both teacher and student models from scratch.

Dataset In Section 5.4, we use the original pre-training dataset (BookCorpus (Zhu et al., 2015) + Wikipedia (Foundation)) used in DistilBERT (Sanh et al., 2019) to evaluate the effectiveness of our Community KD method. For the other experiments, we use a reduced dataset (30M) created by uniformly sampling 1 out of every 4 sentences from the original dataset for pre-training. We evaluate our distilled models on the dev sets of the GLUE benchmark (Wang et al., 2019), which consists of nine sentence-level classification tasks.



(a) Teacher Performance (b) Student Performance

Figure 5: **Adjusting Loss Weight** We investigate the impact of the distillation for the teacher model and student model by adjusting loss weights ($\alpha_h, \alpha_s, \beta_h, \beta_s$) for hard loss and soft loss. (a) We (co-)train the teacher model distilling knowledge from the student by fixing $\alpha_h : \alpha_s = 1 : 1$ and varying $\beta_h : \beta_s$ on the large text corpus. (b) We (co-)train the student model distilling knowledge from the teacher by fixing $\beta_h : \beta_s = 1 : 1$ and varying $\alpha_h : \alpha_s$ on the large text corpus. Then we report the average performance of each pre-trained model after fine-tuning it on downstream tasks (dev sets) of GLUE benchmark.

Model Architecture In Section 5.4, we use a pre-trained BERT-base model (Devlin et al., 2019) as the teacher and a 6-layer BERT model as the student, which is the same architecture used in DistilBERT. For the other experiments, we use a 6-layer BERT model as the teacher and a 4-layer BERT model as the student to analyze the effectiveness and efficiency of our CTCD method.

5.1 Could Knowledge Distillation Help Improve Performance?

In Figure 4 and Table 1, we show the average performance of models trained using different methods on a large text corpus and fine-tuned against the GLUE benchmark. **Stand-alone** trains a model without using any knowledge distillation. **Co-training & One-way distillation** trains teacher and student models together from scratch, with knowledge only flowing from the teacher to the student. **Co-training & Co-distillation (Ours)** is our CTCD method, which trains both teacher and student models together from scratch and distills knowledge between each other in both directions. For distillation methods, we set the weights of the hard losses for the teacher and student to 1. The weights of the soft losses are chosen from the set $\{0.5, 1, 2, 4\}$, and the results are reported with the best-performing weights.

1) Overall Results As shown in Table 1, during pre-training, the student model distilled by our CTCD outperforms the student distilled by the one-way distillation method on the average performance of the GLUE benchmark. After training for 10 and 20 epochs, the student distilled by our CTCD consistently has higher gains than the student distilled by one-way distillation, as 77.46 vs. **77.94** and 78.39 vs. **79.12**, respectively. Furthermore, the student distilled by CTCD significantly outperforms the original teacher trained using the stand-alone method, achieving a higher gain of **1.66**. We further analyze these results in Figure 4.

2) Does learning from a small and weak student provide a performance benefit to the teacher? The distillation of knowledge from the student model to the teacher model has been shown to significantly improve the quality of the teacher model, with an average increase of 1.88 compared to teacher training methods that do not incorporate such distillation process (such as Stand-alone and Co-training & one-way distillation).

3) Is the teacher's performance improvement reflected in the student's performance improvement? We find that a better teacher leads to further performance improvement of the student from 77.46 to 77.94. The results demonstrate that the distillation process successfully improves the performance of both the teacher and student. The student trained with our CTCD method achieves better performance than students trained with the Stand-alone or Co-training & one-way distillation methods, with an average improvement of 0.88 and 0.47, respectively.

5.2 In-depth Study 1: Adjusting Loss Weights

In this Section, we investigate the impact of distillation on the performance of both the student and the teacher by adjusting loss weights ($\alpha_h, \alpha_s, \beta_h, \beta_s$)

for hard loss and soft loss. For example, setting $\alpha_h : \alpha_s = 1 : 4$ emphasizes learning from the teacher's knowledge (i.e., the soft loss for the student) 4× more than the ground truth label distribution (i.e., the hard loss for the student), during co-training. This allows us to better understand the effect of distillation on each model and optimize their performance.

Teacher side In Figure 5(a), we co-train the teacher model using distillation to transfer knowledge from the student model. We fix the weighting values for the losses of the student to $\alpha_h : \alpha_s = 1 : 1$ and vary the weighting values for the losses of the teacher, $\beta_h : \beta_s$, while training on a large text corpus. We then evaluate the average performance of the pre-trained teacher models on downstream tasks from the dev sets of GLUE benchmark.

Our results show that co-training the teacher with distillation outperforms training the teacher alone, regardless of the weighting values for the soft loss, by obtaining higher gains of 0.36, 0.77, 1.80, and 1.02 for $\beta_h : \beta_s = 1 : 1, 1 : 2, 1 : 4$, and $4 : 4$, respectively. Additionally, we find that giving more weight to the soft loss of the teacher during training ($\alpha_h : \alpha_s : \beta_h : \beta_\mathbf{s} = 1 : 1 : 1 : \mathbf{1} \rightarrow 1 : 1 : 1 : \mathbf{2} \rightarrow 1 : 1 : 1 : \mathbf{4}$) leads to improved performance, with an average score of $78.42 \rightarrow 78.83 \rightarrow 79.86$. Furthermore, we observe that emphasizing only the soft loss of the teacher ($1 : 1 : 1 : 4$) yields better performance than emphasizing both the hard and soft losses of the teacher ($1 : 1 : 4 : 4$), with an average score of 79.86 vs. 79.09.

Student side We find that the student model's performance is not sensitive to the weighting values for the hard and soft losses of the student, ($\alpha_h : \alpha_s$). Regardless of the chosen values, co-training the student with distillation consistently improves its performance compared to training the student alone. For instance, when we emphasize the soft loss of the student by increasing the weighting value for (α_s) as $1 : \mathbf{1} : 1 : 1 \rightarrow 1 : \mathbf{2} : 1 : 1 \rightarrow 1 : \mathbf{4} : 1 : 1$, we observe similar levels of performance for the student model.

5.3 In-depth Study 2: Length of Training

We studied the impact of co-training length on the effectiveness of the CTCD method (see Figure 6). We find that longer training leads to improved performance, as demonstrated by our experiments using two different training lengths: 10 epochs and 20 epochs. After pre-training the student models

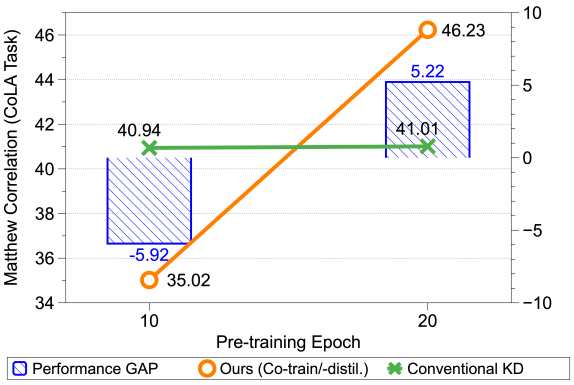


Figure 6: **Length of Training** We pre-trained student models under two different training lengths 10/20 epochs while distilling knowledge from teacher models via ours or the conventional KD method. Then we adapt pre-trained student models on CoLA task. With enough longer training (20 epoch), the student model distilled by ours significantly outperforms the student model distilled by the conventional KD method, with a higher gain of 5.22.

with these different lengths, we adapted them to the CoLA downstream tasks and evaluated their performance using Matthew Correlation.

Results By increasing the (co-)training length from 10 epochs to 20 epochs, CTCD (Ours) significantly improves the performance of the student model from 35.02 to **46.23**, with a gain of 11.21. This outperforms the conventional KD method, which only achieves a gain of 0.07 from 40.94 to **41.01**, despite a longer training time. The conventional KD method relies on a pre-trained teacher model to train the student model, which allows for fast convergence but limits the learning of the student model. In contrast, the CTCD method allows for additional performance gains for both the teacher and student models by enabling them to learn and grow together during co-training. This can provide further benefits to the student model's performance with longer co-training.

5.4 Efficacy of Community KD

We compare the proposed Community KD with the conventional KD method, DistilBERT (Sanh et al., 2019). To ensure a fair comparison, we use the pre-trained BERT-base as the teacher model for both methods and the 6-layer BERT as the student, which is the same architecture used in DistilBERT. As described in Section 4, we train two student models concurrently and they learn from the pre-trained BERT, the ground truth labels, and each other's knowledge. Note that since we fine-tune **one** of the two students distilled by Community KD for downstream tasks, the inference cost is

Downstream Task			MNLI	QQP	QNLI	SST-2	CoLA	STSB		MRPC		RTE	Average
Metric		AMP	Acc.	Acc.	Acc.	Acc.	Matthew.	Pearson.	Spear.	F1	Acc.	Acc.	Acc.
Dataset Size			392.7k	363.8k	104.7k	67.3k	8.5k	5.7k		3.7k		2.5k	
Teacher	BERT (109M)		84.17	90.89	90.68	91.86	57.54	88.84	88.56	89.31	85.04	65.34	83.23
Student (67M)	DistilBERT	FP32	**81.93**	90.05	87.72	90.94	52.03	86.28	86.07	87.94	82.59	57.76	80.33
	Ours: Student 1	FP16	81.88	**90.17**	88.24	**91.51**	54.82	**86.70**	**86.49**	89.76	**85.29**	**59.21**	**81.40**
	Ours: Student 2	FP16	81.34	89.75	**88.37**	90.71	**56.08**	86.42	86.44	**89.80**	**85.29**	59.20	81.34

Table 2: **Efficacy of Community KD** The pre-trained BERT and 6-layer BERT is the teacher model and student architecture, respectively, for both ours and DistilBERT. We fine-tune the distilled students on dev sets of GLUE benchmark. We observe that learning from the soft knowledge of different student model improves performance over DistilBERT on most downstream tasks.

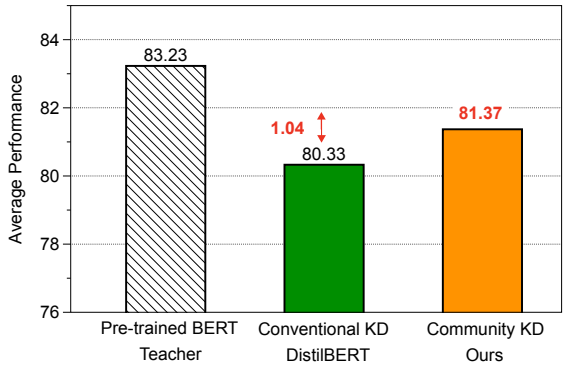


Figure 7: **Comparison with Conventional KD** The student model distilled by ours obtains a higher gain of average performance by 1.04 compared with student model distilled by the conventional KD (DistilBERT).

the same as DistilBERT. In Table 2, we report the results of BERT and DistilBERT using checkpoints provided by Hugging Face (HuggingFace) and both students (Ours: Student 1 and Ours: Student 2) on the dev sets of the GLUE benchmark. We apply Automatic Mixed Precision (AMP) (Paszke et al., 2019) to Community KD, which typically speeds up training but may hurt performance.

Results The results presented in Table 2 and Figure 7 show that Community KD, leads to improved performance on downstream tasks such as QQP, QNLI, SST-2, CoLA, STSB, MRPC, and RTE, even when applying quantization techniques. Specifically, the average performance gain of the student model distilled using our Community KD method is 1.04 (1.2%) higher than that of the student model distilled by the conventional KD (DistilBERT). This suggests that incorporating knowledge distillation from both a student model and a pre-trained teacher model is more effective than only using knowledge distillation from the pre-trained teacher model.

6 Limitations & Future Work

Limitations The proposed method co-train models from scratch and may require a longer pre-training time than the conventional KD method. However, as we described in Section 5.3, when the student model is trained long enough with its teacher, it can outperform the models trained with the conventional KD on the downstream task. The proposed co-training method may increase the overall training cost compared with one-way distillation, and it may become a performance bottleneck depending on training resource constraints. However, note that model co-training and co-distillation can improve model quality while having the same inference cost as the one-way distillation on downstream tasks.

Future Work **1) Co-distillation from the pre-trained models.** If there exists a pre-trained teacher and/or student, using such pre-trained models in co-training can further improve their performance. **2) Architecture Sharing.** Models can share some of their architectures by reusing the output of such architectures and updating them together during back-propagation. This may help reduce the additional computing and memory overhead incurred by model co-training, while improving the model quality, especially for the student model.

7 Conclusion

The increasing size and computational complexity of pre-trained language models (PLMs) can limit their usability in serving online downstream tasks. In this study, we proposed a novel co-training and co-distillation (CTCD) framework that improves both model inference efficiency and quality by training models of different sizes together and extracting inter-model knowledge in both directions. This allowed us to avoid the trade-off between efficiency and performance that is typically associated with KD. We introduced a novel Community KD approach for compressing PLMs, which outperformed the conventional KD method, by 1.2% on the GLUE benchmark.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Wikimedia Foundation. Wikimedia downloads.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

HuggingFace. Huggingface.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Chunyuan Li, Xiang Gao, Yuan Li, Xiujun Li, Baolin Peng, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *EMNLP*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Yujia Qin, Yankai Lin, Jing Yi, Jiajie Zhang, Xu Han, Zhengyan Zhang, Yusheng Su, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Knowledge inheritance for pre-trained language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3921–3937, Seattle, United States. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *In Proceedings of ICLR*.

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

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Henry Tsai, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical BERT models for sequence labeling. 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 3632–3636, Hong Kong, China. Association for Computational Linguistics.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *arXiv preprint arXiv:1908.08962*, 13.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

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. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. 2020. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.