TinyBERT: Distilling BERT for Natural Language Understanding

Xiaoqi Jiao[1*†], **Yichun Yin**[2*‡], **Lifeng Shang**[2‡], **Xin Jiang**[2]
Xiao Chen[2], **Linlin Li**[3], **Fang Wang**[1‡] **and Qun Liu**[2]
[1]Key Laboratory of Information Storage System, Huazhong University of Science and Technology, Wuhan National Laboratory for Optoelectronics
[2]Huawei Noah's Ark Lab
[3]Huawei Technologies Co., Ltd.
{jiaoxiaoqi,wangfang}@hust.edu.cn
{yinyichun,shang.lifeng,jiang.xin}@huawei.com
{chen.xiao2,lynn.lilinlin,qun.liu}@huawei.com

Abstract

Language model pre-training, such as BERT, has significantly improved the performances of many natural language processing tasks. However, pre-trained language models are usually computationally expensive, so it is difficult to efficiently execute them on resource-restricted devices. To accelerate inference and reduce model size while maintaining accuracy, we first propose a novel *Transformer distillation* method that is specially designed for knowledge distillation (KD) of the Transformer-based models. By leveraging this new KD method, the plenty of knowledge encoded in a large "teacher" BERT can be effectively transferred to a small "student" TinyBERT. Then, we introduce a new *two-stage learning* framework for TinyBERT, which performs Transformer distillation at both the pre-training and task-specific learning stages. This framework ensures that TinyBERT can capture the general-domain as well as the task-specific knowledge in BERT.

TinyBERT$_4$[1] with 4 layers is empirically effective and achieves more than 96.8% the performance of its teacher BERT$_{\text{BASE}}$ on GLUE benchmark, while being **7.5x smaller** and **9.4x faster** on inference. TinyBERT$_4$ is also significantly better than 4-layer state-of-the-art baselines on BERT distillation, with only ∼**28%** parameters and ∼**31%** inference time of them. Moreover, TinyBERT$_6$ with 6 layers performs **on-par with** its teacher BERT$_{\text{BASE}}$.

1 Introduction

Pre-training language models then fine-tuning on downstream tasks has become a new paradigm for natural language processing (NLP). Pre-trained language models (PLMs), such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020), T5 (Raffel et al., 2019) and ELECTRA (Clark et al., 2020), have achieved great success in many NLP tasks (e.g., the GLUE benchmark (Wang et al., 2018) and the challenging multi-hop reasoning task (Ding et al., 2019)). However, PLMs usually have a large number of parameters and take long inference time, which are difficult to be deployed on edge devices such as mobile phones. Recent studies (Kovaleva et al., 2019; Michel et al., 2019; Voita et al., 2019) demonstrate that there is redundancy in PLMs. Therefore, it is crucial and feasible to reduce the computational overhead and model storage of PLMs while retaining their performances.

There have been many model compression techniques (Han et al., 2016) proposed to accelerate deep model inference and reduce model size while maintaining accuracy. The most commonly used techniques include quantization (Gong et al., 2014), weights pruning (Han et al., 2015), and knowledge distillation (KD) (Romero et al., 2014). In this paper, we focus on knowledge distillation, an idea originated from Hinton et al. (2015), in a *teacher-student* framework. KD aims to transfer the knowledge embedded in a large teacher network to a small student network where the student network is trained to reproduce the behaviors of the teacher network. Based on the framework, we propose a novel distillation method specifically for the Transformer-based models (Vaswani et al., 2017), and use BERT as an example to investigate the method for large-scale PLMs.

KD has been extensively studied in NLP (Kim and Rush, 2016; Hu et al., 2018) as well as for pre-trained language models (Sanh et al., 2019; Sun et al., 2019, 2020; Wang et al., 2020). The *pre-training-then-fine-tuning* paradigm firstly pre-

*Authors contribute equally.
†This work is done when Xiaoqi Jiao is an intern at Huawei Noah's Ark Lab.
‡Corresponding authors.
[1]The code and models are publicly available at https://github.com/huawei-noah/Pretrained-Language-Model/tree/master/TinyBERT

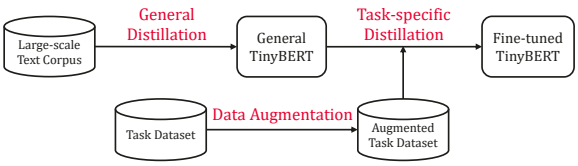

Figure 1: The illustration of TinyBERT learning.

trains BERT on a large-scale unsupervised text corpus, then fine-tunes it on task-specific dataset, which greatly increases the difficulty of BERT distillation. Therefore, it is required to design an effective KD strategy for both training stages.

To build a competitive TinyBERT, we firstly propose a new *Transformer distillation* method to distill the knowledge embedded in teacher BERT. Specifically, we design three types of loss functions to fit different representations from BERT layers: 1) the output of the embedding layer; 2) the hidden states and attention matrices derived from the Transformer layer; 3) the logits output by the prediction layer. The attention based fitting is inspired by the recent findings (Clark et al., 2019) that the attention weights learned by BERT can capture substantial linguistic knowledge, and it thus encourages the linguistic knowledge can be well transferred from teacher BERT to student TinyBERT. Then, we propose a novel *two-stage learning* framework including the *general distillation* and the *task-specific distillation*, as illustrated in Figure 1. At general distillation stage, the original BERT without fine-tuning acts as the teacher model. The student TinyBERT mimics the teacher's behavior through the proposed Transformer distillation on general-domain corpus. After that, we obtain a general TinyBERT that is used as the initialization of student model for the further distillation. At the task-specific distillation stage, we first do the data augmentation, then perform the distillation on the augmented dataset using the fine-tuned BERT as the teacher model. It should be pointed out that both the two stages are essential to improve the performance and generalization capability of TinyBERT.

The main contributions of this work are as follows: 1) We propose a new Transformer distillation method to encourage that the linguistic knowledge encoded in teacher BERT can be adequately transferred to TinyBERT; 2) We propose a novel two-stage learning framework with performing the proposed Transformer distillation at both the pre-training and fine-tuning stages, which ensures that TinyBERT can absorb both the general-domain and task-specific knowledge of the teacher BERT. 3) We show in the experiments that our TinyBERT$_4$ can achieve more than 96.8% the performance of teacher BERT$_{\text{BASE}}$ on GLUE tasks, while having much fewer parameters ($\sim$13.3%) and less inference time ($\sim$10.6%), and significantly outperforms other state-of-the-art baselines with 4 layers on BERT distillation; 4) We also show that a 6-layer TinyBERT$_6$ can perform on-par with the teacher BERT$_{\text{BASE}}$ on GLUE.

2 Preliminaries

In this section, we describe the formulation of Transformer (Vaswani et al., 2017) and Knowledge Distillation (Hinton et al., 2015). Our proposed *Transformer distillation* is a specially designed KD method for Transformer-based models.

2.1 Transformer Layer

Most of the recent pre-trained language models (e.g., BERT, XLNet and RoBERTa) are built with Transformer layers, which can capture long-term dependencies between input tokens by self-attention mechanism. Specifically, a standard Transformer layer includes two main sub-layers: *multi-head attention* (MHA) and *fully connected feed-forward* network (FFN).

Multi-Head Attention (MHA). The calculation of attention function depends on the three components of queries, keys and values, denoted as matrices $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ respectively. The attention function can be formulated as follows:

$$\boldsymbol{A} = \frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}, \quad (1)$$

$$\texttt{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \texttt{softmax}(\boldsymbol{A})\boldsymbol{V}, \quad (2)$$

where d_k is the dimension of keys and acts as a scaling factor, $\boldsymbol{A}$ is the attention matrix calculated from the compatibility of $\boldsymbol{Q}$ and $\boldsymbol{K}$ by dot-product operation. The final function output is calculated as a weighted sum of values $\boldsymbol{V}$, and the weight is computed by applying `softmax()` operation on the each column of matrix $\boldsymbol{A}$. According to Clark et al. (2019), the attention matrices in BERT can capture substantial linguistic knowledge, and thus play an essential role in our proposed distillation method.

Multi-head attention is defined by concatenating the attention heads from different representation

subspaces as follows:

$$\texttt{MHA}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=\texttt{Concat}(\mathrm{h}_1,\ldots,\mathrm{h}_k)\boldsymbol{W}, \quad (3)$$

where k is the number of attention heads, and h_i denotes the i-th attention head, which is calculated by the $\texttt{Attention}()$ function with inputs from different representation subspaces. The matrix $\boldsymbol{W}$ acts as a linear transformation.

Position-wise Feed-Forward Network (FFN). Transformer layer also contains a fully connected feed-forward network, which is formulated as follows:

$$\texttt{FFN}(x) = \max(0, x\boldsymbol{W}_1 + b_1)\boldsymbol{W}_2 + b_2. \quad (4)$$

We can see that the FFN contains two linear transformations and one ReLU activation.

2.2 Knowledge Distillation

KD aims to transfer the knowledge of a large teacher network T to a small student network S. The student network is trained to mimic the behaviors of teacher networks. Let f^T and f^S represent the *behavior* functions of teacher and student networks, respectively. The behavior function targets at transforming network inputs to some informative representations, and it can be defined as the output of any layer in the network. In the context of Transformer distillation, the output of MHA layer or FFN layer, or some intermediate representations (such as the attention matrix $\boldsymbol{A}$) can be used as behavior function. Formally, KD can be modeled as minimizing the following objective function:

$$\mathcal{L}_{\text{KD}} = \sum_{x\in\mathcal{X}} L\big(f^S(x), f^T(x)\big), \quad (5)$$

where $L(\cdot)$ is a loss function that evaluates the difference between teacher and student networks, x is the text input and $\mathcal{X}$ denotes the training dataset. Thus the key research problem becomes how to define effective behavior functions and loss functions. Different from previous KD methods, we also need to consider how to perform KD at the pre-training stage of BERT in addition to the task-specific training stage.

3 Method

In this section, we propose a novel distillation method for Transformer-based models, and present a *two-stage learning* framework for our model distilled from BERT, which is called TinyBERT.

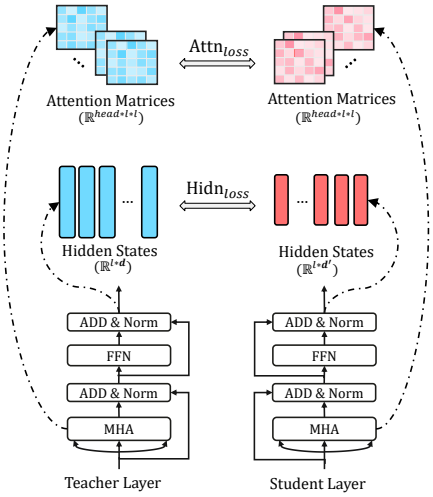

Figure 2: The details of Transformer-layer distillation consisting of Attn$_{loss}$(attention based distillation) and Hidn$_{loss}$(hidden states based distillation).

3.1 Transformer Distillation

The proposed *Transformer distillation* is a specially designed KD method for Transformer networks. In this work, both the student and teacher networks are built with Transformer layers. For a clear illustration, we formulate the problem before introducing our method.

Problem Formulation. Assuming that the student model has M Transformer layers and teacher model has N Transformer layers, we start with choosing M out of N layers from the teacher model for the *Transformer-layer distillation*. Then a function $n = g(m)$ is defined as the mapping function between indices from student layers to teacher layers, which means that the m-th layer of student model learns the information from the $g(m)$-th layer of teacher model. To be precise, we set 0 to be the index of embedding layer and $M+1$ to be the index of prediction layer, and the corresponding layer mappings are defined as $0 = g(0)$ and $N+1 = g(M+1)$ respectively. The effect of the choice of different mapping functions on the performances is studied in the experiment section. Formally, the student can acquire knowledge from the teacher by minimizing the following objective:

$$\mathcal{L}_{\text{model}} = \sum_{x\in\mathcal{X}}\sum_{m=0}^{M+1}\lambda_m\mathcal{L}_{\text{layer}}(f_m^S(x), f_{g(m)}^T(x)), \quad (6)$$

where $\mathcal{L}_{\text{layer}}$ refers to the loss function of a given model layer (e.g., Transformer layer or embedding layer), $f_m(x)$ denotes the behavior function induced from the m-th layers and λ_m is the hyper-

parameter that represents the importance of the m-th layer's distillation.

Transformer-layer Distillation. The proposed Transformer-layer distillation includes the *attention based distillation* and *hidden states based distillation*, which is shown in Figure 2. The attention based distillation is motivated by the recent findings that attention weights learned by BERT can capture rich linguistic knowledge (Clark et al., 2019). This kind of linguistic knowledge includes the syntax and coreference information, which is essential for natural language understanding. Thus we propose the attention based distillation to encourage that the linguistic knowledge can be transferred from teacher (BERT) to student (TinyBERT). Specifically, the student learns to fit the matrices of multi-head attention in the teacher network, and the objective is defined as:

$$\mathcal{L}_{\text{attn}} = \frac{1}{h}\sum_{i=1}^{h} \texttt{MSE}(\boldsymbol{A}_i^S, \boldsymbol{A}_i^T), \tag{7}$$

where h is the number of attention heads, $\boldsymbol{A}_i \in \mathbb{R}^{l\times l}$ refers to the attention matrix corresponding to the i-th head of teacher or student, l is the input text length, and $\texttt{MSE}()$ means the *mean squared error* loss function. In this work, the (unnormalized) attention matrix $\boldsymbol{A}_i$ is used as the fitting target instead of its softmax output $\texttt{softmax}(\boldsymbol{A}_i)$, since our experiments show that the former setting has a faster convergence rate and better performances.

In addition to the attention based distillation, we also distill the knowledge from the output of Transformer layer, and the objective is as follows:

$$\mathcal{L}_{\text{hidn}} = \texttt{MSE}(\boldsymbol{H}^S\boldsymbol{W}_h, \boldsymbol{H}^T), \tag{8}$$

where the matrices $\boldsymbol{H}^S \in \mathbb{R}^{l\times d'}$ and $\boldsymbol{H}^T \in \mathbb{R}^{l\times d}$ refer to the hidden states of student and teacher networks respectively, which are calculated by Equation 4. The scalar values d and d' denote the hidden sizes of teacher and student models, and d' is often smaller than d to obtain a smaller student network. The matrix $\boldsymbol{W}_h \in \mathbb{R}^{d'\times d}$ is a learnable linear transformation, which transforms the hidden states of student network into the same space as the teacher network's states.

Embedding-layer Distillation. Similar to the hidden states based distillation, we also perform embedding-layer distillation and the objective is:

$$\mathcal{L}_{\text{embd}} = \texttt{MSE}(\boldsymbol{E}^S\boldsymbol{W}_e, \boldsymbol{E}^T), \tag{9}$$

where the matrices $\boldsymbol{E}^S$ and $\boldsymbol{H}^T$ refer to the embeddings of student and teacher networks, respectively. In this paper, they have the same shape as the hidden state matrices. The matrix $\boldsymbol{W}_e$ is a linear transformation playing a similar role as $\boldsymbol{W}_h$.

Prediction-layer Distillation. In addition to imitating the behaviors of intermediate layers, we also use the knowledge distillation to fit the predictions of teacher model as in Hinton et al. (2015). Specifically, we penalize the soft cross-entropy loss between the student network's logits against the teacher's logits:

$$\mathcal{L}_{\text{pred}} = \texttt{CE}(\boldsymbol{z}^T/t, \boldsymbol{z}^S/t), \tag{10}$$

where $\boldsymbol{z}^S$ and $\boldsymbol{z}^T$ are the logits vectors predicted by the student and teacher respectively, $\texttt{CE}$ means the *cross entropy* loss, and t means the temperature value. In our experiment, we find that $t = 1$ performs well.

Using the above distillation objectives (i.e. Equations 7, 8, 9 and 10), we can unify the distillation loss of the corresponding layers between the teacher and the student network:

$$\mathcal{L}_{\text{layer}} = \begin{cases} \mathcal{L}_{\text{embd}}, & m = 0 \\ \mathcal{L}_{\text{hidn}} + \mathcal{L}_{\text{attn}}, & M \geq m > 0 \\ \mathcal{L}_{\text{pred}}, & m = M+1 \end{cases} \tag{11}$$

3.2 TinyBERT Learning

The application of BERT usually consists of two learning stages: the pre-training and fine-tuning. The plenty of knowledge learned by BERT in the pre-training stage is of great importance and should be transferred to the compressed model. Therefore, we propose a novel two-stage learning framework including the *general distillation* and the *task-specific distillation*, as illustrated in Figure 1. General distillation helps TinyBERT learn the rich knowledge embedded in pre-trained BERT, which plays an important role in improving the generalization capability of TinyBERT. The task-specific distillation further teaches TinyBERT the knowledge from the fine-tuned BERT. With the two-step distillation, we can substantially reduce the gap between teacher and student models.

General Distillation. We use the original BERT without fine-tuning as the teacher and a large-scale text corpus as the training data. By performing the Transformer distillation [2] on the text from general

[2] In the general distillation, we do not perform prediction-layer distillation as Equation 10. Our motivation is to make

Algorithm 1 Data Augmentation Procedure for Task-specific Distillation

Input: $\mathbf{x}$ is a sequence of words
Params: p_t: the threshold probability
N_a: the number of samples augmented per example
K: the size of candidate set
Output: D': the augmented data

$n \leftarrow 0$; $D' \leftarrow [\,]$
while $n < N_a$ **do**
  $\mathbf{x}_m \leftarrow \mathbf{x}$
  for $i \leftarrow$1 to len($\mathbf{x}$) **do**
    if $\mathbf{x}[i]$ is a single-piece word **then**
      Replace $\mathbf{x}_m[i]$ with `[MASK]`
      $C \leftarrow K$ most probable words of $\texttt{BERT}(\mathbf{x}_m)[i]$
    else
      $C \leftarrow K$ most similar words of $\mathbf{x}[i]$ from GloVe
    end if
    Sample $p \sim$ Uniform(0, 1)
    if $p \leq p_t$ **then**
      Replace $\mathbf{x}_m[i]$ with a word in C randomly
    end if
  end for
  Append $\mathbf{x}_m$ to D'
  $n \leftarrow n + 1$
end while
return D'

domain, we obtain a general TinyBERT that can be fine-tuned for downstream tasks. However, due to the significant reductions of the hidden/embedding size and the layer number, general TinyBERT performs generally worse than BERT.

Task-specific Distillation. Previous studies show that the complex models, such as fine-tuned BERTs, suffer from over-parametrization for domain-specific tasks (Kovaleva et al., 2019). Thus, it is possible for smaller models to achieve comparable performances to the BERTs. To this end, we propose to produce competitive fine-tuned TinyBERTs through the task-specific distillation. In the task-specific distillation, we re-perform the proposed Transformer distillation on an augmented task-specific dataset. Specifically, the fine-tuned BERT is used as the teacher and a data augmentation method is proposed to expand the task-specific training set. Training with more task-related examples, the generalization ability of the student model can be further improved.

Data Augmentation. We combine a pre-trained language model BERT and GloVe (Pennington et al., 2014) word embeddings to do word-level replacement for data augmentation. Specifically, we use the language model to predict word replacements for single-piece words (Wu et al., 2019), and use the word embeddings to retrieve the most similar words as word replacements for multiple-pieces words[3]. Some hyper-parameters are defined to control the replacement ratio of a sentence and the amount of augmented dataset. More details of the data augmentation procedure are shown in Algorithm 1. We set p_t = 0.4, N_a = 20, K = 15 for all our experiments.

The above two learning stages are complementary to each other: the general distillation provides a good initialization for the task-specific distillation, while the task-specific distillation on the augmented data further improves TinyBERT by focusing on learning the task-specific knowledge. Although there is a significant reduction of model size, with the data augmentation and by performing the proposed Transformer distillation method at both the pre-training and fine-tuning stages, TinyBERT can achieve competitive performances in various NLP tasks.

4 Experiments

In this section, we evaluate the effectiveness and efficiency of TinyBERT on a variety of tasks with different model settings.

4.1 Datasets

We evaluate TinyBERT on the General Language Understanding Evaluation (GLUE) (Wang et al., 2018) benchmark, which consists of 2 single-sentence tasks: CoLA (Warstadt et al., 2019), SST-2 (Socher et al., 2013), 3 sentence similarity tasks: MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017), QQP (Chen et al., 2018), and 4 natural language inference tasks: MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Bentivogli et al., 2009) and WNLI (Levesque et al., 2012). The metrics for these tasks can be found in the GLUE paper (Wang et al., 2018).

4.2 TinyBERT Settings

We instantiate a tiny student model (the number of layers M=4, the hidden size d'=312, the feed-forward/filter size d'_i=1200 and the head number h=12) that has a total of 14.5M parameters. This model is referred to as TinyBERT$_4$. The original

the TinyBERT primarily learn the intermediate structures of BERT at pre-training stage. From our preliminary experiments, we also found that conducting prediction-layer distillation at pre-training stage does not bring extra improvements on downstream tasks, when the Transformer-layer distillation (Attn and Hidn distillation) and Embedding-layer distillation have already been performed.

[3] A word is tokenized into multiple word-pieces by the tokenizer of BERT.

System	#Params	#FLOPs	Speedup	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg
$\text{BERT}_{\text{BASE}}$ (Teacher)	109M	22.5B	1.0x	83.9/83.4	71.1	90.9	93.4	52.8	85.2	87.5	67.0	79.5
$\text{BERT}_{\text{TINY}}$	14.5M	1.2B	9.4x	75.4/74.9	66.5	84.8	87.6	19.5	77.1	83.2	62.6	70.2
$\text{BERT}_{\text{SMALL}}$	29.2M	3.4B	5.7x	77.6/77.0	68.1	86.4	89.7	27.8	77.0	83.4	61.8	72.1
BERT_4-PKD	52.2M	7.6B	3.0x	79.9/79.3	70.2	85.1	89.4	24.8	79.8	82.6	62.3	72.6
DistilBERT_4	52.2M	7.6B	3.0x	78.9/78.0	68.5	85.2	91.4	32.8	76.1	82.4	54.1	71.9
$\text{MobileBERT}_{\text{TINY}}$†	15.1M	3.1B	-	81.5/81.6	68.9	**89.5**	91.7	**46.7**	80.1	**87.9**	65.1	**77.0**
TinyBERT_4 (ours)	14.5M	1.2B	9.4x	**82.5/81.8**	**71.3**	87.7	**92.6**	44.1	**80.4**	86.4	**66.6**	**77.0**
BERT_6-PKD	67.0M	11.3B	2.0x	81.5/81.0	70.7	89.0	92.0	-	-	85.0	65.5	-
PD	67.0M	11.3B	2.0x	82.8/82.2	70.4	88.9	91.8	-	-	86.8	65.3	-
DistilBERT_6	67.0M	11.3B	2.0x	82.6/81.3	70.1	88.9	92.5	49.0	81.3	86.9	58.4	76.8
TinyBERT_6 (ours)	67.0M	11.3B	2.0x	**84.6/83.2**	**71.6**	**90.4**	**93.1**	**51.1**	**83.7**	**87.3**	**70.0**	**79.4**

Table 1: Results are evaluated on the test set of GLUE official benchmark. The best results for each group of student models are in-bold. The architecture of TinyBERT_4 and $\text{BERT}_{\text{TINY}}$ is (M=4, d=312, d_i=1200), $\text{BERT}_{\text{SMALL}}$ is (M=4, d=512, d_i=2048), BERT_4-PKD and DistilBERT_4 is (M=4, d=768, d_i=3072) and the architecture of BERT_6-PKD, DistilBERT_6 and TinyBERT_6 is (M=6, d=768, d_i=3072). All models are learned in a single-task manner. The inference speedup is evaluated on a single NVIDIA K80 GPU. † denotes that the comparison between $\text{MobileBERT}_{\text{TINY}}$ and TinyBERT_4 may not be fair since the former has 24 layers and is task-agnosticly distilled from $\text{IB-BERT}_{\text{LARGE}}$ while the later is a 4-layers model task-specifically distilled from $\text{BERT}_{\text{BASE}}$.

$\text{BERT}_{\text{BASE}}$ (N=12, d=768, d_i=3072 and h=12) is used as the teacher model that contains 109M parameters. We use $g(m) = 3 \times m$ as the layer mapping function, so TinyBERT_4 learns from every 3 layers of $\text{BERT}_{\text{BASE}}$. The learning weight λ of each layer is set to 1. Besides, for a direct comparisons with baselines, we also instantiate a TinyBERT_6 (M=6, d'=768, d'_i=3072 and h=12) with the same architecture as BERT_6-PKD (Sun et al., 2019) and DistilBERT_6 (Sanh et al., 2019).

TinyBERT learning includes the general distillation and the task-specific distillation. For the general distillation, we set the maximum sequence length to 128 and use English Wikipedia (2,500M words) as the text corpus and perform the *intermediate layer distillation* for 3 epochs with the supervision from a pre-trained $\text{BERT}_{\text{BASE}}$ and keep other hyper-parameters the same as BERT pre-training (Devlin et al., 2019). For the task-specific distillation, under the supervision of a fine-tuned BERT, we firstly perform *intermediate layer distillation* on the augmented data for 20 epochs[4] with batch size 32 and learning rate 5e-5, and then perform *prediction layer distillation* on the augmented data [5] for 3 epochs with choosing the batch size from {16, 32} and learning rate from {1e-5, 2e-5, 3e-5} on dev set. At task-specific distillation, the maximum sequence length is set to 64 for single-sentence tasks, and 128 for sequence pair tasks.

[4]For large datasets MNLI, QQP, and QNLI, we only perform 10 epochs of the *intermediate layer distillation*, and for the challenging task CoLA, we perform 50 epochs at this step.

[5]For regression task STS-B, the original train set is better.

4.3 Baselines

We compare TinyBERT with $\text{BERT}_{\text{TINY}}$, $\text{BERT}_{\text{SMALL}}$[6] (Turc et al., 2019) and several state-of-the-art KD baselines including BERT-PKD (Sun et al., 2019), PD (Turc et al., 2019), DistilBERT (Sanh et al., 2019) and MobileBERT (Sun et al., 2020). $\text{BERT}_{\text{TINY}}$ means directly pretraining a small BERT, which has the same model architecture as TinyBERT_4. When training $\text{BERT}_{\text{TINY}}$, we follow the same learning strategy as described in the original BERT (Devlin et al., 2019). To make a fair comparison, we use the released code to train a 4-layer BERT_4-PKD[7] and a 4-layer DistilBERT_4[8] and fine-tuning these 4-layer baselines with suggested hyper-paramters. For 6-layer baselines, we use the reported numbers or evaluate the results on the test set of GLUE with released models.

4.4 Experimental Results on GLUE

We submitted our model predictions to the official GLUE evaluation server to obtain results on the test set[9], as summarized in Table 1.

The experiment results from the 4-layer student models demonstrate that: 1) There is a large performance gap between $\text{BERT}_{\text{TINY}}$ (or $\text{BERT}_{\text{SMALL}}$) and $\text{BERT}_{\text{BASE}}$ due to the dramatic reduction in model size. 2) TinyBERT_4 is consistently better than $\text{BERT}_{\text{TINY}}$ on all the GLUE tasks and

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

[7]https://github.com/intersun/PKD-for-BERT-Model-Compression

[8]https://github.com/huggingface/transformers/tree/master/examples/distillation

[9]https://gluebenchmark.com

obtains a large improvement of 6.8% on average. This indicates that the proposed KD learning framework can effectively improve the performances of small models on a variety of downstream tasks. 3) TinyBERT$_4$ significantly outperforms the 4-layer state-of-the-art KD baselines (i.e., BERT$_4$-PKD and DistilBERT$_4$) by a margin of at least 4.4%, with $\sim$28% parameters and 3.1x inference speedup. 4) Compared with the teacher BERT$_{\text{BASE}}$, TinyBERT$_4$ is 7.5x smaller and 9.4x faster in the model efficiency, while maintaining competitive performances. 5) For the challenging CoLA dataset (the task of predicting linguistic acceptability judgments), all the 4-layer distilled models have big performance gaps compared to the teacher model, while TinyBERT$_4$ achieves a significant improvement over the 4-layer baselines. 6) We also compare TinyBERT with the 24-layer MobileBERT$_{\text{TINY}}$, which is distilled from 24-layer IB-BERT$_{\text{LARGE}}$. The results show that TinyBERT$_4$ achieves the same average score as the 24-layer model with only 38.7% FLOPs. 7) When we increase the capacity of our model to TinyBERT$_6$, its performance can be further elevated and outperforms the baselines of the same architecture by a margin of 2.6% on average and achieves comparable results with the teacher. 8) Compared with the other two-stage baseline PD, which first pre-trains a small BERT, then performs distillation on a specific task with this small model, TinyBERT initialize the student in task-specific stage via general distillation. We analyze these two initialization methods in Appendix C.

In addition, BERT-PKD and DistilBERT initialize their student models with some layers of a pre-trained BERT, which makes the student models have to keep the same size settings of Transformer layer (or embedding layer) as their teacher. In our two-stage distillation framework, TinyBERT is initialized through general distillation, making it more flexible in choosing model configuration.

More Comparisons. We demonstrate the effectiveness of TinyBERT by including more baselines such as Poor Man's BERT (Sajjad et al., 2020), BERT-of-Theseus (Xu et al., 2020) and MiniLM (Wang et al., 2020), some of which only report results on the GLUE dev set. In addition, we evaluate TinyBERT on SQuAD v1.1 and v2.0. Due to the space limit, we present our results in the Appendix A and B.

System	MNLI-m	MNLI-mm	MRPC	CoLA	Avg
TinyBERT$_4$	82.8	82.9	85.8	50.8	75.6
w/o GD	82.5	82.6	84.1	40.8	72.5
w/o TD	80.6	81.2	83.8	28.5	68.5
w/o DA	80.5	81.0	82.4	29.8	68.4

Table 2: Ablation studies of different procedures (i.e., TD, GD, and DA) of the two-stage learning framework. The variants are validated on the dev set.

System	MNLI-m	MNLI-mm	MRPC	CoLA	Avg
TinyBERT$_4$	82.8	82.9	85.8	50.8	75.6
w/o Embd	82.3	82.3	85.0	46.7	74.1
w/o Pred	80.5	81.0	84.3	48.2	73.5
w/o Trm	71.7	72.3	70.1	11.2	56.3
w/o Attn	79.9	80.7	82.3	41.1	71.0
w/o Hidn	81.7	82.1	84.1	43.7	72.9

Table 3: Ablation studies of different distillation objectives in the TinyBERT learning. The variants are validated on the dev set.

4.5 Ablation Studies

In this section, we conduct ablation studies to investigate the contributions of : a) different procedures of the proposed two-stage TinyBERT learning framework in Figure 1, and b) different distillation objectives in Equation 11.

4.5.1 Effects of Learning Procedure

The proposed two-stage TinyBERT learning framework consists of three key procedures: GD (General Distillation), TD (Task-specific Distillation) and DA (Data Augmentation). The performances of removing each individual learning procedure are analyzed and presented in Table 2. The results indicate that all of the three procedures are crucial for the proposed method. The TD and DA has comparable effects in all the four tasks. We note that the task-specific procedures (TD and DA) are more helpful than the pre-training procedure (GD) on all of the tasks. Another interesting observation is that GD contribute more on CoLA than on MNLI and MRPC. We conjecture that the ability of linguistic generalization (Warstadt et al., 2019) learned by GD plays an important role in the task of linguistic acceptability judgments.

4.5.2 Effects of Distillation Objective

We investigate the effects of distillation objectives on the TinyBERT learning. Several baselines are proposed including the learning without the Transformer-layer distillation (w/o Trm), the embedding-layer distillation (w/o Emb) or the

prediction-layer distillation (w/o Pred)[10] respectively. The results are illustrated in Table 3 and show that all the proposed distillation objectives are useful. The performance w/o Trm[11] drops significantly from 75.6 to 56.3. The reason for the significant drop lies in the initialization of student model. At the pre-training stage, obtaining a good initialization is crucial for the distillation of transformer-based models, while there is no supervision signal from upper layers to update the parameters of transformer layers at this stage under the w/o Trm setting. Furthermore, we study the contributions of attention (Attn) and hidden states (Hidn) in the Transformer-layer distillation. We can find the attention based distillation has a greater impact than hidden states based distillation. Meanwhile, these two kinds of knowledge distillation are complementary to each other, which makes them the most important distillation techniques for Transformer-based model in our experiments.

4.6 Effects of Mapping Function

We also investigate the effects of different mapping functions $n = g(m)$ on the TinyBERT learning. Our original TinyBERT as described in section 4.2 uses the uniform strategy, and we compare with two typical baselines including top-strategy ($g(m) = m + N - M; 0 < m \leq M$) and bottom-strategy ($g(m) = m; 0 < m \leq M$).

The comparison results are presented in Table 4. We find that the top-strategy performs better than the bottom-strategy on MNLI, while being worse on MRPC and CoLA, which confirms the observations that different tasks depend on the knowledge from different BERT layers. The uniform strategy covers the knowledge from bottom to top layers of $\text{BERT}_{\text{BASE}}$, and it achieves better performances than the other two baselines in all the tasks. Adaptively choosing layers for a specific task is a challenging problem and we leave it as future work.

System	MNLI-m	MNLI-mm	MRPC	CoLA	Avg
Uniform	82.8	82.9	85.8	50.8	75.6
Top	81.7	82.3	83.6	35.9	70.9
Bottom	80.6	81.3	84.6	38.5	71.3

Table 4: Results (dev) of different mapping strategies for TinyBERT_4.

5 Related Work

Pre-trained Language Models Compression Generally, pre-trained language models (PLMs) can be compressed by low-rank approximation (Ma et al., 2019; Lan et al., 2020), weight sharing (Dehghani et al., 2019; Lan et al., 2020), knowledge distillation (Tang et al., 2019; Sanh et al., 2019; Turc et al., 2019; Sun et al., 2020; Liu et al., 2020; Wang et al., 2020), pruning (Cui et al., 2019; McCarley, 2019; F. et al., 2020; Elbayad et al., 2020; Gordon et al., 2020; Hou et al., 2020) or quantization (Shen et al., 2019; Zafrir et al., 2019). In this paper, our focus is on knowledge distillation.

Knowledge Distillation for PLMs There have been some works trying to distill pre-trained language models (PLMs) into smaller models. $\text{BiLSTM}_{\text{SOFT}}$ (Tang et al., 2019) distills task-specific knowledge from BERT into a single-layer BiLSTM. BERT-PKD (Sun et al., 2019) extracts knowledges not only from the last layer of the teacher, but also from intermediate layers at fine-tuning stage. DistilBERT (Sanh et al., 2019) performs distillation at pre-training stage on large-scale corpus. Concurrent works, MobileBERT (Sun et al., 2020) distills a $\text{BERT}_{\text{LARGE}}$ augmented with bottleneck structures into a 24-layer slimmed version by progressive knowledge transfer at pre-training stage. MiniLM (Wang et al., 2020) conducts deep self-attention distillation also at pre-training stage. By contrast, we propose a new *two-stage learning* framework to distill knowledge from BERT at both pre-training and fine-tuning stages by a novel transformer distillation method.

Pretraining Lite PLMs Other related works aim at directly pretraining lite PLMs. Turc et al. (2019) pre-trained 24 miniature BERT models and show that pre-training remains important in the context of smaller architectures, and fine-tuning pre-trained compact models can be competitive. ALBERT (Lan et al., 2020) incorporates embedding factorization and cross-layer parameter sharing to reduce model parameters. Since ALBERT does not reduce hidden size or layers of transformer block, it still has large amount of computations. Another concurrent work, ELECTRA (Clark et al., 2020) proposes a sample-efficient task called replaced token detection to accelerate pre-training, and it also presents a 12-layer $\text{ELECTRA}_{\text{small}}$ that has comparable performance with TinyBERT_4. Different

[10] The prediction-layer distillation performs soft cross-entropy as Equation 10 on the augmented data. "w/o Pred" means performing standard cross-entropy against the ground-truth of original train set.

[11] Under "w/o Trm" setting, we actually 1) conduct embedding-layer distillation at the pre-training stage; 2) perform embedding-layer and prediction-layer distillation at fine-tuning stage.

from these small PLMs, TinyBERT$_4$ is a 4-layer model which can achieve more speedup.

6 Conclusion and Future Work

In this paper, we introduced a new method for Transformer-based distillation, and further proposed a two-stage framework for TinyBERT. Extensive experiments show that TinyBERT achieves competitive performances meanwhile significantly reducing the model size and inference time of BERT$_{\mathrm{BASE}}$, which provides an effective way to deploy BERT-based NLP models on edge devices. In future work, we would study how to effectively transfer the knowledge from wider and deeper teachers (e.g., BERT$_{\mathrm{LARGE}}$) to student TinyBERT. Combining distillation with quantization/pruning would be another promising direction to further compress the pre-trained language models.

Acknowledgements

This work is supported in part by NSFC NO.61832020, No.61821003, 61772216, National Science and Technology Major Project No.2017ZX01032-101, Fundamental Research Funds for the Central Universities.

References

L. Bentivogli, P. Clark, I. Dagan, and D. Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge.

D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation*.

Z. Chen, H. Zhang, X. Zhang, and L. Zhao. 2018. Quora question pairs.

K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. 2019. What does bert look at? an analysis of bert's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

K. Clark, M. Luong, Q. V. Le, and C. D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

B. Cui, Y. Li, M. Chen, and Z. Zhang. 2019. Fine-tune bert with sparse self-attention mechanism. In *EMNLP*.

M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. 2019. Universal transformers. In *ICLR*.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *ACL*.

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

M. Elbayad, J. Gu, E. Grave, and M. Auli. 2020. Depth-adaptive transformer. In *ICLR*.

Angela F., Edouard G., and Armand J. 2020. Reducing transformer depth on demand with structured dropout. In *ICLR*.

Y. Gong, L. Liu, M. Yang, and L. Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

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

S. Han, Mao H., and Dally W. J. 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *ICLR*.

S Han, J. Pool, J. Tran, and W. Dally. 2015. Learning both weights and connections for efficient neural network. In *NIPS*.

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

L. Hou, L. Shang, X. Jiang, and Q. Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *arXiv preprint arXiv:2004.04037*.

M. Hu, Y. Peng, F. Wei, Z. Huang, D. Li, N. Yang, and M. Zhou. 2018. Attention-guided answer distillation for machine reading comprehension. In *EMNLP*.

Y. Kim and A. M. Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky. 2019. Revealing the dark secrets of bert. In *EMNLP*.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

W. Liu, P. Zhou, Z. Zhao, Z. Wang, H. Deng, and Q. Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

X. Ma, P. Zhang, S. Zhang, N. Duan, Y. Hou, M. Zhou, and D. Song. 2019. A tensorized transformer for language modeling. In *NIPS*.

J. S. McCarley. 2019. Pruning a bert-based question answering model. *arXiv preprint arXiv:1910.06360*.

P. Michel, O. Levy, and G. Neubig. 2019. Are sixteen heads really better than one? In *NIPS*.

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

P. Rajpurkar, R. Jia, and P. Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov. 2020. Poor man's bert: Smaller and faster transformer models. *arXiv preprint arXiv:2004.03844*.

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

S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer. 2019. Q-bert: Hessian based ultra low precision quantization of bert. *arXiv preprint arXiv:1909.05840*.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

S. Sun, Y. Cheng, Z. Gan, and J. Liu. 2019. Patient knowledge distillation for bert model compression. In *EMNLP*.

Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, and J. Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

I. Turc, M. Chang, K. Lee, and K. Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *arXiv preprint arXiv:1908.08962*.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *NIPS*.

E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*.

A. Warstadt, A. Singh, and S. R. Bowman. 2019. Neural network acceptability judgments. *TACL*.

A. Williams, N. Nangia, and S. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

X. Wu, S. Lv, L. Zang, J. Han, and S. Hu. 2019. Conditional bert contextual augmentation. In *International Conference on Computational Science*.

C. Xu, W. Zhou, T. Ge, F. Wei, and M. Zhou. 2020. Bert-of-theseus: Compressing bert by progressive module replacing. *arXiv preprint arXiv:2002.02925*.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NIPS*.

O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat. 2019. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*.

Appendix

A More Comparisons on GLUE

Since some prior works on BERT compression only evaluate their models on the GLUE dev set, for an easy and direct comparison, we here compare our TinyBERT$_6$ with the reported results from these prior works. All the compared methods have the

System	CoLA (8.5k) Mcc	MNLI-m (393k) Acc	MNLI-mm (393k) Acc	MRPC (3.7k) F1/Acc	QNLI (105k) Acc	QQP (364k) F1/Acc	RTE (2.5k) Acc	SST-2 (67k) Acc	STS-B (5.7k) Pear/Spea
Same Student Architecture (M=6;d'=768;d'_i=3072)									
DistilBERT$_6$	51.3	82.2	-	87.5/-	89.2	-/88.5	59.9	92.7	-/86.9
Poor Man's BERT$_6$	-	81.1	-	-/80.2	87.6	-/90.4	65.0	90.3	-/88.5
BERT-of-Theseus	51.1	82.3	-	89.0/-	89.5	-/89.6	68.2	91.5	-/88.7
MiniLM$_6$	49.2	84.0	-	88.4/-	91.0	-/91.0	71.5	92.0	-
TinyBERT$_6$	**54.0**	**84.5**	**84.5**	**90.6/86.3**	**91.1**	**88.0/91.1**	**73.4**	**93.0**	**90.1/89.6**

Table 5: Comparisons between TinyBERT with other baselines on the dev set of GLUE tasks. Mcc refers to Matthews correlation and Pear/Spea refer to Pearson/Spearman.

same model architecture as TinyBERT$_6$ (i.e. M=6, d'=768, d'_i=3072).

The direct comparison results are shown in Table 5. We can see the TinyBERT$_6$ outperforms all the baselines under the same settings of architecture and evaluation methods. The effectiveness of TinyBERT is further confirmed.

B Results on SQuAD v1.1 and v2.0

We also demonstrate the effectiveness of TinyBERT on the question answering (QA) tasks: SQuAD v1.1 (Rajpurkar et al., 2016) and SQuAD v2.0 (Rajpurkar et al., 2018). Following the learning procedure in the previous work (Devlin et al., 2019), we treat these two tasks as the problem of sequence labeling which predicts the possibility of each token as the start or end of answer span. One small difference from the GLUE tasks is that we perform the prediction-layer distillation on the original training dataset instead of the augmented dataset, which can bring better performances.

The results show that TinyBERT consistently outperforms both the 4-layer and 6-layer baselines, which indicates that the proposed framework also works for the tasks of token-level labeling. Compared with sequence-level GLUE tasks, the question answering tasks depend on more subtle knowledge to infer the correct answer, which increases the difficulty of knowledge distillation. We leave how to build a better QA-TinyBERT as future work.

C Initializing TinyBERT with BERT$_{\text{TINY}}$

In the proposed two-stage learning framework, to make TinyBERT effectively work for different downstream tasks, we propose the General Distillation (GD) to capture the general domain knowledge, through which the TinyBERT learns the knowledge from intermediate layers of teacher BERT at the pre-training stage. After that, a general TinyBERT is obtained and used as the initialization of student model for Task-specific Distillation (TD) on downstream tasks.

In our preliminary experiments, we have also tried to initialize TinyBERT with the directly pre-trained BERT$_{\text{TINY}}$, and then conduct the TD on downstream tasks. We denote this compression method as BERT$_{\text{TINY}}$(+TD). The results in Table 7 show that BERT$_{\text{TINY}}$(+TD) performs

System	**SQuAD 1.1** EM	F1	**SQuAD 2.0** EM	F1
BERT$_{\text{BASE}}$ (Teacher)	80.7	88.4	74.5	77.7
4-layer student models				
BERT$_4$-PKD	70.1	79.5	60.8	64.6
DistilBERT$_4$	71.8	81.2	60.6	64.1
MiniLM$_4$	-	-	-	69.7
TinyBERT$_4$	**72.7**	**82.1**	**68.2**	**71.8**
6-layer student models				
BERT$_6$-PKD	77.1	85.3	66.3	69.8
DistilBERT$_6$	78.1	86.2	66.0	69.5
MiniLM$_6$	-	-	-	76.4
TinyBERT$_6$	**79.7**	**87.5**	**74.7**	**77.7**

Table 6: Results (dev) of baselines and TinyBERT on question answering tasks. The architecture of MiniLM$_4$ is (M=4, d=384, d_i=1536) which is wider than TinyBERT$_4$, and the architecture of MiniLM$_6$ is the same as TinyBERT$_6$(M=6, d=768, d_i=3072)

System	MNLI-m (392k)	MNLI-mm (392k)	MRPC (3.5k)	CoLA (8.5k)	Avg
BERT$_{\text{TINY}}$	75.9	76.9	83.2	19.5	63.9
BERT$_{\text{TINY}}$(+TD)	79.2	79.7	82.9	12.4	63.6
TinyBERT (GD)	76.6	77.2	82.0	8.7	61.1
TinyBERT (GD+TD)	80.5	81.0	82.4	29.8	68.4

Table 7: Results of different methods at pre-training stage. TD and GD refers to Task-specific Distillation (without data augmentation) and General Distillation, respectively. The results are evaluated on dev set.

even worse than $\text{BERT}_{\text{TINY}}$ on MRPC and CoLA tasks. We conjecture that if without imitating the $\text{BERT}_{\text{BASE}}$'s behaviors at the pre-training stage, $\text{BERT}_{\text{TINY}}$ will derive mismatched distributions in intermediate representations (e.g., attention matrices and hidden states) with the $\text{BERT}_{\text{BASE}}$ model. The following task-specific distillation under the supervision of fine-tuned $\text{BERT}_{\text{BASE}}$ will further disturb the learned distribution/knowledge of $\text{BERT}_{\text{TINY}}$, finally leading to poor performances on some less-data tasks. For the intensive-data task (e.g. MNLI), TD has enough training data to make $\text{BERT}_{\text{TINY}}$ acquire the task-specific knowledge very well, although the pre-trained distributions have already been disturbed.

From the results of Table 7, we find that GD can effectively transfer the knowledge from the teacher BERT to the student TinyBERT and achieve comparable results with $\text{BERT}_{\text{TINY}}$ (61.1 vs. 63.9), even without performing the MLM and NSP tasks. Furthermore, the task-specific distillation boosts the performances of TinyBERT by continuing on learning the task-specific knowledge from fine-tuned teacher $\text{BERT}_{\text{BASE}}$.

D GLUE Details

The GLUE datasets are described as follows:

MNLI. Multi-Genre Natural Language Inference is a large-scale, crowd-sourced entailment classification task (Williams et al., 2018). Given a pair of $\langle premise, hypothesis \rangle$, the goal is to predict whether the $hypothesis$ is an entailment, contradiction, or neutral with respect to the $premise$.

QQP. Quora Question Pairs is a collection of question pairs from the website Quora. The task is to determine whether two questions are semantically equivalent (Chen et al., 2018).

QNLI. Question Natural Language Inference is a version of the Stanford Question Answering Dataset which has been converted to a binary sentence pair classification task by Wang et al. (2018). Given a pair $\langle question, context \rangle$. The task is to determine whether the $context$ contains the answer to the $question$.

SST-2. The Stanford Sentiment Treebank is a binary single-sentence classification task, where the goal is to predict the sentiment of movie reviews (Socher et al., 2013).

CoLA. The Corpus of Linguistic Acceptability is a task to predict whether an English sentence is a grammatically correct one (Warstadt et al., 2019).

STS-B. The Semantic Textual Similarity Benchmark is a collection of sentence pairs drawn from news headlines and many other domains (Cer et al., 2017). The task aims to evaluate how similar two pieces of texts are by a score from 1 to 5.

MRPC. Microsoft Research Paraphrase Corpus is a paraphrase identification dataset where systems aim to identify if two sentences are paraphrases of each other (Dolan and Brockett, 2005).

RTE. Recognizing Textual Entailment is a binary entailment task with a small training dataset (Bentivogli et al., 2009).