# ERNIE-Tiny : A Progressive Distillation Framework for Pretrained Transformer Compression

Anonymous ACL submission


## Abstract

Pretrained language models (PLMs) such as BERT adopt a training paradigm that first pretrains the model in general data and then finetunes the model on task-specific data, and have recently achieved great success. However, PLMs are notorious for their enormous parameters and hard to be deployed on real-life applications. Knowledge distillation has been prevailing to address this problem by transferring knowledge from a large teacher to a much smaller student over a set of data. We argue that the selection of three key components, namely teacher, training data, and learning objective, is crucial to the effectiveness of distillation. We, therefore, propose a four-stage progressive distillation framework ERNIE-Tiny to compress PLM, which varies the three components gradually from general level to task-specific level. Specifically, the first stage, **General Distillation**, performs distillation with guidance from pretrained teacher, general data, and latent distillation loss. Then, **General-Enhanced Distillation** changes teacher model from pretrained teacher to finetuned teacher. After that, **Task-Adaptive Distillation** shifts training data from general data to task-specific data. In the end, **Task-Specific Distillation** adds two additional losses, namely Soft-Label and Hard-Label loss onto the last stage. Empirical results demonstrate the effectiveness of our framework and generalization gain brought by ERNIE-Tiny. In particular, experiments show that a 4-layer ERNIE-Tiny maintains over 98.0% performance of its 12-layer teacher BERT$_{Base}$ on GLUE benchmark, surpassing state-of-the-art (SOTA) by 1.0% GLUE score with the same amount of parameters. Moreover, ERNIE-Tiny achieves a new compression SOTA on five Chinese NLP tasks, outperforming BERT$_{Base}$ by 0.4% accuracy with 7.5x fewer parameters and 9.4x faster inference speed.


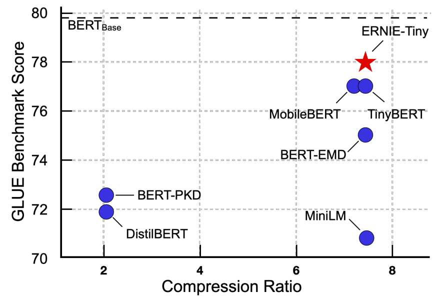


Figure 1: GLUE score of different distillation methods. Performance of the teacher, BERT$_{base}$, is shown in dash line.

## 1 Introduction

Transformer-based pretrained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020; Sun et al., 2019b; Lewis et al., 2020; Lample and Conneau, 2019) have brought significant improvements to the field of Natural Language Processing (NLP). Their training process that first pretrains model on general data and then finetunes on task-specific data has set up a new training paradigm for NLP. However, the performance gains come with the massive growth in model sizes (Brown et al., 2020; Raffel et al., 2019; Fedus et al., 2021; Shoeybi et al., 2019) which causes high inference time and storage cost. It becomes the main obstacle for industrial application, especially for deploying on edge devices.

There are some recent efforts such as Knowledge Distillation (KD) (Hinton et al., 2015; Urban et al., 2016; Ba and Caruana, 2013), quantization (Kim et al., 2019; Shin et al., 1909; Wei et al., 2018), and weights pruning (Wang et al., 2018b; Han et al., 2015; Sindhwani et al., 2015) trying to tackle this problem. KD, in particular, aims to transfer knowledge from one network called teacher model to another called student model by training student under the guidance of teacher. Typically, teacher is a model with more parameters and capable of achieving high accuracy, whereas student is a model with significantly fewer parameters and requires much less computation. Once trained, the student model maintains teacher's performance while massively reducing inference time and storage demand, and

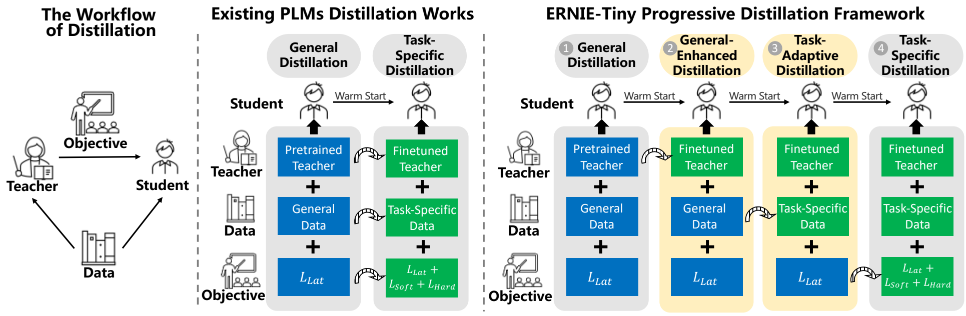


Figure 2: The comparison between existing works and ERNIE-Tiny. The curly shaded arrow indicates the change of the three key components (i.e. Teacher, Data, and Objective). **Left: Workflow of Distillation.** Teacher transfers its knowledge to student through data and objective. **Middle: Workflow of Existing Works.** All of the three components shift between the two stages. **Right: Workflow of ERNIE-Tiny**. ERNIE-Tiny carefully designs the distillation framework such that only one component is changed between any two consecutive stages.

can be deployed in real-life applications. KD can be applied on either or both of pretrain and finetune stages. For example, MiniLM (Wang et al., 2020) and MobileBert (Sun et al., 2020) apply KD on pretrain stage while (Sun et al., 2019a) applies KD on finetune stage. Moreover, TinyBERT (Jiao et al., 2020) and DistilBERT (Ren et al., 2020) perform KD on both pretrain and finetune stages. In particular, they employ pretrained teacher to provide guidance during pretrain stage and choose finetuned teacher during finetune stage, where pretrained teacher is the teacher model trained on general data and finetuned teacher is obtained by finetuning pretrained teacher on task-specific data.

However, existing works suffer from pretrain-finetune distillation discrepancy consisting of the difference of training data, teacher model, and learning objective between pretrain phase and finetune phase. Specifically, training data is shifted from general data to task-specific data, teacher is changed from pretrained teacher to finetuned teacher, and learning objective is altered differently according to their own decisions. We argue that this sudden transition hurts the effectiveness of distillation. We, therefore, propose a four-stage progressive distillation framework ERNIE-Tiny to alleviate this problem, and our method outperforms several baselines as shown in Figure 1. ERNIE-Tiny attempts to smooth this pretrain-finetune transition by gradually altering teacher, learning objective, and training data from general level to task-specific level.

Akin to pretrain distillation at existing works, **General distillation (GD)** performs distillation with pretrained teacher on general data. Following previous works (Jiao et al., 2020; Sun et al., 2019a, 2020; Ren et al., 2020), we utilize latent distillation ($\mathcal{L}_{Lat}$) as our learning objective. Then, by altering teacher from pretrained teacher to finetuned teacher, ERNIE-Tiny introduces **General-Enhanced Distillation (GED)** which distills with finetuned teacher and $\mathcal{L}_{Lat}$ on general data. After that, through changing training data from general data to task-specific data, ERNIE-Tiny presents **Task-Adaptive Distillation (TAD)** which distills with finetuned teacher and $\mathcal{L}_{Lat}$ on task-specific data. Finally, ERNIE-Tiny concludes the training process with **Task-Specific Distillation (TSD)** through adding new learning objectives, namely Soft-Label Distillation ($\mathcal{L}_{Soft}$) and Hard-Label loss ($\mathcal{L}_{Hard}$) which represents the task-specific finetune loss such as cross-entropy for classification downstream task. Note that TSD is similar to the finetune distillation at existing works. Figure 2 compares the workflow of existing works and ERNIE-Tiny.

Notably, general-enhanced distillation provides finetuned teacher's guidance not through task-specific data as what existing works do, but through general data. Compared with existing works, general-enhanced distillation allows student to absorb task-specific knowledge through general data, improving the effectiveness of distillation and generalization of student model (Laine and Aila, 2016; Sajjadi et al., 2016; Miyato et al., 2018; Goodfellow et al., 2014). Empirical results show that with general-enhanced distillation, ERNIE-Tiny outperforms the baseline on out-of-domain datasets, demonstrating the generalization gain brought by general-enhanced distillation. In addition, general data can be regarded as additional data to task-specific data. We conduct experiments to show that the effect of general-enhanced distillation is more significant on low-resource tasks. Moreover, task-adaptive distillation is introduced between general-enhanced distillation and task-specific distillation, serving as a bridge to smooth the transition between those two stages. We conduct experiments to show the performance gain brought by this stage.

The **main contributions** of this work are as follows: **1)** We propose a novel four-stage progressive learning framework for language model compression called ERNIE-Tiny to smooth the distillation process by gradually altering teacher, training data, and learning objective. **2)** To our knowledge, leveraging finetuned teacher with general data is the first time introduced in PLM distillation, helping student capture task-specific knowledge from finetuned teacher and improving generalization of student. **3)** ERNIE-Tiny achieves 9.4x speedup keeps over 98.0% performance of its 12-layer teacher $\text{BERT}_{base}$ on GLUE benchmark and exceeds state-of-the-art (SOTA) by 1.0% GLUE score. In Chinese datasets, 4-layer ERNIE-Tiny, harnessed with a better teacher, outperforms $\text{BERT}_{base}$ by 0.4% accuracy with 7.5x fewer parameters and 9.4x faster inference speed.

## 2 Related Work

**Pretrained Language Models** Pretrained language models are learned on large amounts of text data and then finetuned to adapt to specific tasks. BERT (Devlin et al., 2019) proposes to pretrain a deep bidirectional Transformer. RoBERTa (Liu et al., 2019) achieves strong performance by training longer steps using large batch size and more text data. ERNIE (Sun et al., 2019b) (Sun et al., 2019c) proposes to pretrain the language model on an enhanced mask whole word objective and further employs continue learning strategy. Recent works (Shoeybi et al., 2019; Brown et al., 2020; Kaplan et al., 2020) observe the trend that increasing model size also leads to lower perplexity. Switch-transformer (Fedus et al., 2021) simplifies and improves over Mixture of Experts (Shazeer et al., 2017) and trains a trillion parameters language model. However, (Kovaleva et al., 2019) shows the parameters are redundant in those models and the performance can be kept even when the computational overhead and model storage is reduced. Moreover, the training cost of those models also raises serious environmental concerns (Strubell et al., 2019).

**Knowledge Distillation** Knowledge distillation (Hinton et al., 2015; Wang et al., 2020) aims to train a small student model with soft labels and intermediate representations provided by the large teacher model. (Jiao et al., 2020) proposes TinyBERT on the general distillation and task-specific distillation stages. (Ren et al., 2020) proposes DistilBERT, which successfully halves the depth of BERT model by knowledge distillation in the pretrain stage and an optional finetune stage. (Sun et al., 2019a) distills BERT into a shallower student through knowledge distillation only in the finetune stage. (Wang et al., 2020) proposes to compress teacher by mimicking self-attention and value relation in the pretrain stage. In contrast to these existing literature, we argue that the pretrain-finetune distillation discrepancy exists. Specifically, the pretrain-finetune distillation discrepancy is caused by training data shift, teacher model alteration and learning objective change. Therefore, we propose a progressive distillation framework ERNIE-Tiny to compress PLM. Through this progressive distillation framework, the discrepancy of distillation can be alleviated and the performance of the distilled student can be improved. Table 1 summarizes the differences between our framework and previous works.

## 3 Proposed Framework

Distillation aims to use the pretrained teacher $T$ to teach a student model $S$ that is usually much smaller as shown in the left part of Figure 2. In our setting, besides the labeled task-specific data $D_t$, we also have large-scale unlabeled data which we call general data $D_g$ from which the teacher is pretrained. To combine those data and teacher knowledge smoothly, we devise a four-stage progressive distillation framework. Those four stages vary the three key distillation components, namely training data, teacher model and learning objective gradually from general level to task-specific level as shown in Figure 2. To better explain those methods, we first show the background and discuss the distillation framework in detail.

### 3.1 Background: Transformer Backbone

The Transformer architecture (Vaswani et al., 2017) is a highly modularized neural network, where each Transformer layer consists of two sub-modules, namely the multi-head self-attention

| Stage | Teacher | Data | ERNIE-Tiny | BERT-EMD | TinyBERT | DistilBERT | BERT-PKD | MiniLM | MobileBert |
|---|---|---|---|---|---|---|---|---|---|
| GD | pretrained | General | $\mathcal{L}_{Lat}$ | $\mathcal{L}_{Lat}$ | $\mathcal{L}_{Lat}$ | $\mathcal{L}_{Lat}$+$\mathcal{L}_{Soft}$ | - | $\mathcal{L}_{Lat}$ | $\mathcal{L}_{Lat}$+$\mathcal{L}_{Soft}$ |
| GED | finetuned | General | $\mathcal{L}_{Lat}$ | - | - | - | - | - | - |
| TAD | finetuned | Task-Specific | $\mathcal{L}_{Lat}$ | - | - | - | - | - | - |
| TSD | finetuned | Task-Specific | $\mathcal{L}_{L+S+H}$ | $\mathcal{L}_{L+S+H}$ | $\mathcal{L}_{L+S+H}$ | $\mathcal{L}_{L+S+H}$ | $\mathcal{L}_{L+S+H}$ | $\mathcal{L}_{Hard}$ | $\mathcal{L}_{Hard}$ |

Table 1: Comparison with previous PLM distillation approaches. Latent Distillation ($\mathcal{L}_{Lat}$) represents distillation loss on the attributes at intermediate layers and it varies on different methods (e.g hidden states and attention distribution in TinyBERT and BERT-EMD; attention distribution and attention value relation in MiniLM). Soft-Label Distillation ($\mathcal{L}_{Soft}$) denotes distillation on soft target probabilities from the teacher model. As all methods adopt Hard-Label loss ($\mathcal{L}_{Hard}$) in TSD, for simplicity, we denote $\mathcal{L}_{L+S+H} = \mathcal{L}_{Lat} + \mathcal{L}_{Soft} + \mathcal{L}_{Hard}$.

(MHA) and position-wise feed-forward network (FFN). Transformer encodes contextual information for input tokens. The input embeddings $\{\mathbf{x}\}_{i=1}^{s}$ for sample $x$ are packed together into $\mathbf{H}_0 = [\mathbf{x}_1, \cdots, \mathbf{x}_s]$, where $s$ denotes the input sequence length. Then stacked Transformer blocks iteratively compute the encoding vectors as $\mathbf{H}_l = \text{Transformer}_l(\mathbf{H}_{l-1}), l \in [1, L]$, and the Transformer is computed as:

$$\begin{aligned}\mathbf{A}_{l,a} &= \text{MHA}_{l,a}(\mathbf{H}_{l-1}\mathbf{W}^{Q}_{l,a}, \mathbf{H}_{l-1}\mathbf{W}^{K}_{l,a}),\\ \mathbf{H}'_{l-1} &= \text{LN}(\mathbf{H}_{l-1} + (\overset{h}{\underset{a=1}{\Vert}} \mathbf{A}_{l,a}(\mathbf{H}_{l-1}\mathbf{W}^{V}_{l,a}))\mathbf{W}^{O}_{l}),\\ \mathbf{H}_l &= \text{LN}\left(\mathbf{H}'_{l-1} + \text{FFN}\left(\mathbf{H}'_{l-1}\right)\right),\end{aligned} \tag{1}$$

where the previous layer's output $\mathbf{H}_{l-1} \in \mathbb{R}^{s\times d}$ is linearly projected to a triple of queries, keys and values using parameter matrices $\mathbf{W}^{Q}_{l,a}, \mathbf{W}^{K}_{l,a}, \mathbf{W}^{V}_{l,a} \in \mathbb{R}^{d\times d'}$, where $d$ denotes the hidden size of $\mathbf{H}_l$ and $d'$ denotes the hidden size of each head's dimension. $\mathbf{A}_{l,a} \in \mathbb{R}^{s\times s}$ indicates the attention distributions for the $a$-th head in layer $l$, which is computed by the scaled dot-product of queries and keys respectively. $h$ represents the number of self-attention heads. $\Vert$ denotes concatenate operator along the head dimension. $\mathbf{W}^{O}_{l} \in \mathbb{R}^{d\times d}$ denotes the linear transformer for the output of the attention module. LN denotes the layer normalization operation (Ba et al., 2016). FFN is composed of two linear transformation function including mapping the hidden size of $\mathbf{H}'_{l-1}$ to $d_{ff}$ and then mapping it back to $d$.

### 3.2 General Distillation and General-Enhanced Distillation

**General Distillation** As shown in Figure 2, ERNIE-Tiny employs general distillation and general-enhanced distillation sequentially. In the general distillation stage, the pretrained teacher helps the student learn knowledge on the massive unlabeled general data with the intermediate representation. The loss is computed as follows:

$$\begin{aligned}\mathcal{L}^{T}_{Lat}(x) &= \sum_{l=1}^{L_S}\sum_{a=1}^{h} F(\mathbf{A}^{T}_{k,a}(x), \mathbf{M}_{l,a}\,\mathbf{A}^{S}_{l,a}(x))\\ &\quad + \sum_{l=1}^{L_S} F(\mathbf{H}^{T}_{k}(x), \mathbf{H}^{S}_{l}(x)\mathbf{N}_l),\\ \mathcal{L}_{GD} &= \mathop{\mathbb{E}}_{x\sim D_g} \mathcal{L}^{T_g}_{Lat}(x),\end{aligned} \tag{2}$$

where $k = l \times c$, $\mathcal{L}_{GD}$ denotes the loss for general distillation on the general data $D_g$ . $L_S$ denotes the number of layers of student model. Considering the number of layers of pretrained teacher $L_T$ and student model $L_S$ may not be the same, we set student layers to mimic the representation of every $c$ layers of pretrained teacher model, where $c = L_T \,/\, L_S$. We introduce a mapping matrix $\mathbf{M}_{l,a} \in \mathbb{R}^{h\times h'}$ to align the number of attention heads for teacher and student's attention heads, $h$ and $h'$, when they do not match. Similarly, a linear transformation $\mathbf{N}_l \in \mathbb{R}^{d\times d'}$ is used when the hidden size $d$ and $d'$ of $\mathbf{H}^{T}_{l} \in \mathbb{R}^{s\times d}$ and $\mathbf{H}^{S}_{l} \in \mathbb{R}^{s\times d'}$ does not match. A metric function $F$ is utilized to measure the distance between teacher and student's representation and guide the distillation process. We choose mean square error as $F$ for our experiment. Put it together, we call the right hand side of Eq. (2) latent distillation and denotes it as $\mathcal{L}^{T_g}_{Lat}$ where $T_g$ indicates the pretrained teacher (i.e. the guidance $\mathbf{A}^{T_g}$ and $\mathbf{H}^{T_g}$ come from pretrained teacher).

**General-Enhanced Distillation** To further exploit the general data, we propose to use the fine-tuned teacher as a surrogate for task-specific knowledge and perform distillation over general data. And the training loss of general-enhanced distillation is defined as follows:

$$\mathcal{L}_{GED} = \mathop{\mathbb{E}}_{x\sim D_g} \mathcal{L}^{T_f}_{Lat}(x), \tag{3}$$

where $\mathcal{L}_{Lat}^{T_f}$ indicates that the guidance involved in latent distillation loss comes from finetuned teacher. During general-enhanced distillation, the student is optimized by minimizing the $\mathcal{L}_{GED}$ on general data.

One benefit of this stage is that the distillation process becomes much smoother. Comparing Eq. (3) with Eq. (2), the only change between general distillation and general-enhanced distillation is that we only replace the teacher $T_g$ with $T_f$ among the three components (i.e. teacher, training data, learning objective) while existing works change all of them together at the same time as shown in Figure 2.

Another benefit is that introducing finetuned teacher on general data improves the generalization of student model. As the number of task-specific samples is usually much smaller than general data, having the finetuned teacher generating hidden representations on general data can be used to compensate for the task-specific data sparsity. Those hidden representations extracted from $D_g$ can be regarded as feature augmentation. Although there may be no task-related label information on $D_g$, the hidden representation from finetuned teacher still contains task-specific information. Several works (Laine and Aila, 2016; Sajjadi et al., 2016; Miyato et al., 2018; Goodfellow et al., 2014) succeed in using the random image augmentation to improve generalization performance for semi-supervised tasks. The empirical results on generalization gains led by general-enhanced distillation are shown in Section 4.3.

### 3.3 Task-Adaptive Distillation and Task-Specific Distillation

**Task-Adaptive Distillation** Task-adaptive distillation is introduced after general-enhanced distillation to start distillation on task-specific data. The task-adaptive distillation loss is devised as following:

$$\mathcal{L}_{TAD} = \mathop{\mathbb{E}}_{x \sim D_t} \mathcal{L}_{Lat}^{T_f}(x), \tag{4}$$

where $D_t$ is the task-specific data. Student model is trained by minimizing $\mathcal{L}_{TAD}$. Comparing Eq.(4) with Eq.(3), we see that the difference between general-enhanced distillation and task-adaptive distillation is that the training data is changed from general data to task-specific data.

The advantage of proposing the task-specific stage is two-fold. First, continuing with the philosophy of progressive distillation and pretrain-then-finetune paradigm, only the dataset is changed in this stage to smoothen the distillation. Second, as recent work (Raffel et al., 2019) shows that unsupervised learning on the task-specific data before applying the supervised signal leads to improvement on downstream performance, distillation of hidden representations on task-specific data paves the way for the upcoming task-specific objective learning.

**Task-Specific Distillation** Task-specific distillation is presented to finish the whole distillation process. Compared with the last stage, this stage includes soft-label and hard-label learning objectives. Specifically, the loss is computed as follows:

$$\begin{aligned}
&\mathcal{L}_{TSD} = \mathop{\mathbb{E}}_{(x,y) \sim D_t} \mathcal{L}_{Lat}^{T_f}(x) + \mathcal{L}_{Soft}^{T_f}(x) + \mathcal{L}_{Hard}(x, y), \\
&\mathcal{L}_{Soft}^{T_f}(x) = F_1(z^{T_f}(x), z^S(x)), \\
&\mathcal{L}_{Hard}(x, y) = F_2(y, z^S(x)),
\end{aligned} \tag{5}$$

where $\mathcal{L}_{TSD}$ contains three losses for distillation ($L_{Lat}^{T_f}$), soft-label ($L_{Soft}^{T_f}$) and hard-label ($L_{Hard}$). $z^{T_f}$ and $z^S$ denotes the logit of finetuned teacher and student respectively. $y$ represents the ground-truth label from task-specific data. For supervised classification problems, we choose Kullback-Leibler Divergence (Kullback and Leibler, 1951) for $F_1$ and cross entropy for $F_2$. For regression task, we choose mean square error for both $F_1$ and $F_2$.

### 3.4 Progressive Distillation Framework

The key technique for ERNIE-Tiny is to change the teacher, training data and learning objective carefully and smoothly. Overall, the student is trained using following four losses:

$$\begin{aligned}
\mathcal{L}_{\{T,D,\alpha\}} &= \mathop{\mathbb{E}}_{(x,y) \sim D} \mathcal{L}_{Lat}^{T}(x) + \alpha(\mathcal{L}_{Soft}^{T}(x) + \mathcal{L}_{Hard}(x, y)) \\
&= \begin{cases} \mathcal{L}_{GD}, & T = T_g, D = D_g, \alpha = 0 \\ \mathcal{L}_{GED}, & T = T_f, D = D_g, \alpha = 0 \\ \mathcal{L}_{TAD}, & T = T_f, D = D_t, \alpha = 0 \\ \mathcal{L}_{TSD}, & T = T_f, D = D_t, \alpha = 1 \end{cases}
\end{aligned} \tag{6}$$

where $T \in \{T_g, T_f\}$, $D \in \{D_g, D_t\}$ and $\alpha \in \{0, 1\}$. The overall algorithm is shown in Appendix A.4. Put them together, ERNIE-Tiny presents a smoothly transited distillation framework to effectively compress a large teacher model into a significantly smaller student model. The advantage of each stage is shown in the ablation studies.

| Method | Params | Speedup | MNLIm | MNLImm | QQP | SST-2 | QNLI | MRPC | RTE | CoLA | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BERT$_{Base}$(T.) | 109M | 1x | 84.6 | 83.4 | 71.2 | 93.5 | 90.5 | 88.9 | 66.4 | 52.1 | 85.8 | 79.6 |
| DistilBERT | 52.2M | 3x | 78.9 | 78.0 | 68.5 | 91.4 | 85.2 | 82.4 | 54.1 | 32.8 | 76.1 | 71.9 |
| BERT-PKD | 52.2M | 3x | 79.9 | 79.3 | 70.2 | 89.4 | 85.1 | 82.6 | 62.3 | 24.8 | 79.8 | 72.6 |
| BERT-EMD | 14.5M | 9.4x | 82.1 | 80.6 | 69.3 | 91.0 | 87.2 | 87.6 | 66.2 | 25.6 | **82.3** | 74.7 |
| MobileBERT* | 15.1M | 8.6x | 81.5 | 81.6 | 68.9 | 91.7 | **89.5** | 87.9 | 65.1 | 46.7 | 80.1 | 77.0 |
| MiniLM(re.) | 14.5M | 9.4x | 77.9 | 77.6 | 67.5 | 88.0 | 86.5 | 81.4 | 62.0 | 13.7 | 79.4 | 70.4 |
| TinyBERT | 14.5M | 9.4x | 82.5 | **81.8** | **71.3** | 92.6 | 87.7 | 86.4 | **66.6** | 44.1 | 80.4 | 77.0 |
| ERNIE-Tiny | **14.5M** | **9.4x** | **83.0** | **81.8** | **71.3** | **93.3** | 88.3 | **88.4** | **66.6** | **47.4** | **82.3** | **78.0** |

Table 2: GLUE test results that are scored by GLUE evaluation server. The state-of-the-art results are in bold. All methods adopt BERT$_{Base}$ as teacher model, excluding MobileBERT. MobileBERT* is distilled from IB-BERT, which has the same amount of parameters with BERT$_{Large}$. The architecture of ERNIE-Tiny, BERT-EMD, MiniLM and TinyBERT is ($L$=4, $d$=312, $d_{ff}$=1200). MiniLM on this table is reproducted by us. BERT-PKD and DistilBERT is ($L$=4, $d$=768, $d_{ff}$=3072). MobileBERT is ($L$=24, $d$=128, $d_{ff}$=512) with different transformer architecture design. Please refer to Appendix A.6 for how the speedup is calculated.

# 4 Experiment

In this section, we first evaluate ERNIE-Tiny on English datasets and compare it with existing works. Then, we evaluate ERNIE-Tiny on Chinese datasets. After that, ablation studies and discussions are presented to analyze the contribution of each stage.

## 4.1 Evaluation on English Datasets

### 4.1.1 Downstream Tasks

General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) is chosen to evaluate ERNIE-Tiny. It is a well-studied collection of NLP tasks, including textual entailment, emotion detection, etc. Please refer to Appendix A.2 for details.

### 4.1.2 Experiment Setup

For a fair comparison, we adopt pretrained BERT$_{Base}$ checkpoint released by the author (Devlin et al., 2019) as pretrained teacher. BERT$_{Base}$ is a 12-layer transformer-based model with hidden size of 768 and intermediate size of 3072, accounting for 109M parameters in total, pretrained on English Wikipedia and BooksCorpus (Zhu et al., 2015a). To obtain finetuned teachers, we finetune pretrained BERT$_{Base}$ on each task as the finetuned teachers. Following existing works (Jiao et al., 2020), we adopt a 4-layer model with hidden size of 312 and intermediate hidden size of 1200 as our student. The hyper-parameters for each task are in Appendix A.9. We use GLUE as the task-specific data which is the training data in TAD and TD. We also adopt English Wikipedia and BooksCorpus on which BERT$_{Base}$ is pretrained as the general data which is the training data in GD and GED. This ensures that no additional resources or knowledge are involved. Recall that a finetuned teacher and general data are combined to perform distillation during GED.

### 4.1.3 Results on English Datasets

We compare ERNIE-Tiny with several baselines. The results of MobileBERT (Sun et al., 2020), TinyBERT (Jiao et al., 2020) and BERT-EMD (Li et al., 2020) are quoted from their paper. As BERT-PKD (Sun et al., 2019a) and DistilBERT (Ren et al., 2020) do not experiment with 4-layer model, we quote the results from the TinyBERT's implementation (Jiao et al., 2020). We report test set results evaluated by the official GLUE server, summarized in Table 2. Since MiniLM (Wang et al., 2020) do not report test results on GLUE, We reproduce a 4-layer MiniLM for comparison. ERNIE-Tiny outperforms TinyBERT, DistilBERT, BERT-PKD, MiniLM and BERT-EMD across most tasks and exceeds SOTA by 1.0% GLUE score. Compared with its teacher BERT$_{Base}$, ERNIE-Tiny retains 98.0% performance while is 7.5x smaller and 9.4x faster for inference.

## 4.2 Evaluation on Chinese Datasets

We have also conducted experiments on 5 Chinese datasets, and ERNIE-Tiny outperforms baseline models. Particularly, with equipping a strong teacher ERNIE2.0$_{Base}$ (Sun et al., 2019c), ERNIE-Tiny even outperforms a 12-layer BERT$_{Base}$. Please refer to Appendix A.5 for details.

## 4.3 Ablation Studies

We perform ablation studies on each stage involved in ERNIE-Tiny. To better illustrate the contribution of each stage, we divide them into two categories

| Method | MNLIm | MNLImm | MRPC | CoLA | Avg. |
|---|---|---|---|---|---|
| BERT$_{Base}$ (T.) | 84.5 | 84.6 | 86.8 | 61.3 | 79.3 |
| ERNIE-Tiny | 83.0 | 83.0 | 86.9 | 50.0 | 75.7 |
| w/o GED | 80.7 | 80.8 | 85.0 | 44.9 | 72.3 |

Table 3: Ablation study on distillation with general data. (T.) denotes the teacher model. The model with only GD employ the same training computations with ERNIE-Tiny.

| Method | MNLIm | MNLImm | MRPC | CoLA | Avg. |
|---|---|---|---|---|---|
| BERT$_{Base}$ (T.) | 84.5 | 84.6 | 86.8 | 61.3 | 79.3 |
| ERNIE-Tiny | 83.0 | 83.0 | 86.9 | 50.0 | 75.7 |
| w/o TAD | 80.3 | 80.7 | 86.5 | 39.3 | 71.7 |
| w/o TAD&TSD w/ FT | 81.4 | 81.9 | 83.5 | 20.8 | 66.9 |

Table 4: Ablation study on distillation with task-specific data. FT denotes finetuning model directly. Three experiments used the same number of training computations.

based on the training data used: GD and GED as general data based distillation; TAD and TSD as task-specific data based distillation. Experiments in this section follow the experiment setup in Section 4.1.2. All results in this section are obtained by taking the average on the dev set result of 5 runs.

**Effect of General Data Based Distillation** To analyze the contribution of general data, we perform ablation studies on 2 low-resource tasks MRPC and CoLA, and 1 high-resource task MNLI. As general data is utilized in GD and GED, we construct 2 different settings of ERNIE-Tiny to demonstrate the effect of distilling with general data by removing GED. For a fair comparison, we have increased the training steps of GD to keep the number of training computations the same. It means that we increase the training steps of GD in experiment w/o GED in Table 3 such that the total number of training steps of this experiment equals that of ERNIE-Tiny (i.e., the former has GD with 1000k steps while the latter has GD with 500k steps and GED with 500k steps). This setting aims to remove GED and leave all other settings, including the number of training steps the same to show the performance gain comes from GED strategy, rather than the additional computation. As shown in Table 3, removing GED significantly worsens the performance of distilled student, suggesting that general data plays an important role in distillation. Recall that the only difference between GED and GD is that GED equips a finetuned teacher model. Compared with pretrained treacher, finetuned teacher captures task-specific information and is able to extract task-specific knowledge from general data. The results show that ERNIE-Tiny exceeds the one without GED by 3.4% average score, indicating that GED has a more significant contribution than GD on distillation.

**Effect of Task-specific Data Based Distillation** To demonstrate the effectiveness of distillation on task-specific data, we vary the training process when performing distillation on task-specific data and summarize the results in Table 4. We keep the training computations of three experiments the same by increasing the steps of TAD and finetuning. The results show that solely removing TAD consistently leads to a performance drop across all tasks. Note that although TAD only differs from TSD in that TAD has only $\mathcal{L}_{Lat}$ involved while the loss in TSD comprises $\mathcal{L}_{Lat}$, $\mathcal{L}_{Soft}$ and $\mathcal{L}_{Hard}$. Table 4 shows that without the task-adaptive distillation step, the average score dropped from 75.7 to 71.7, verifying that TAD is essential. The results verify that the transition smoothing brought by TAD is crucial to the effectiveness of distillation. We then remove distillation on task-specific data entirely (i.e. TAD and TSD) and only finetune student of task-specific data, and find significant performance degradation. This indicates that distillation on task-specific data is non-negligible.

**Effect of Student Capacity** To illustrate the effect of the student model size, we enlarge the size of the student model to have the same size as the teacher model. As shown in Table 5, an ERNIE-Tiny with the original model size can exceed the teacher by 0.4% average score.

### 4.4 Discussion

In this section, we analyze how general-enhanced distillation benefits the effectiveness of distillation. Experiments in this section follows the setup in Section 4.1.2.

**General Data as Supplement to Task-specific Data** ERNIE-Tiny transfers task-specific knowledge from finetuned teacher over *general data* to student model in GED, while it transfers task-specific knowledge over *task-specific data* in TAD and TSD. General data in GED can be regarded as a supplement to task-specific data. The effect of additional data should be more significant on low-resource tasks. To illustrate this, we select the relatively large datasets MNLI, QNLI and QQP from GLUE and vary them to 1%, 10%, and 50% of the original size to simulate low-resource tasks.

| Method | MNLIm | MNLImm | MRPC | CoLA | Avg. |
|---|---|---|---|---|---|
| BERT$_{Base}$ ($L$=12;$d$=768;$d_{ff}$=3072) (T.) | 84.5 | 84.6 | 86.8 | 61.3 | 79.3 |
| ERNIE-Tiny ($L$=4;$d$=312;$d_{ff}$=1200) | 83.0 | 83.0 | 86.9 | 50.0 | 75.7 (-3.6) |
| ERNIE-Tiny ($L$=12;$d$=768;$d_{ff}$=3072) | 84.6 | 84.9 | 87.3 | 62.1 | 79.7 (+0.4) |

Table 5: Ablation study on student capacity. (T.) is the teacher model.

| Method | MNLIm | MNLImm | QNLI | QQP | Avg. |
|---|---|---|---|---|---|
| 1% of labeled data | | | | | |
| BERT$_{Base}$ (T.) | 67.0 | 69.3 | 78.4 | 71.3 | 71.5 |
| ERNIE-Tiny | 65.2 | 67.4 | 75.4 | 70.8 | 69.7 |
| w/o GED | 57.7 | 60.5 | 75.4 | 69.4 | 65.8 |
| gain of GED | +7.5 | +6.9 | +0.0 | +1.4 | +4.0 |
| 10% of labeled data | | | | | |
| BERT$_{Base}$ (T.) | 76.4 | 77.3 | 86.9 | 79.7 | 80.1 |
| ERNIE-Tiny | 74.5 | 75.0 | 82.4 | 78.1 | 77.5 |
| w/o GED | 69.1 | 69.8 | 82.4 | 78.2 | 74.9 |
| gain of GED | +5.4 | +5.2 | +0.0 | -0.1 | +2.6 |
| 50% of labeled data | | | | | |
| BERT$_{Base}$ (T.) | 80.5 | 81.9 | 90.1 | 84.2 | 84.2 |
| ERNIE-Tiny | 79.3 | 80.1 | 84.2 | 83.5 | 81.8 |
| w/o GED | 75.3 | 76.4 | 83.5 | 83.3 | 79.6 |
| gain of GED | +4.0 | +3.7 | +0.7 | +0.2 | +2.2 |

Table 6: Ablation study on labeled data size.

| Method | MNLIm | SNLI | RTE |
|---|---|---|---|
| GD+GED+TSD | 81.2 | 75.9 | 65.7 |
| GD+TAD+TSD | 82.4 | 70.9 | 52.8 |
| GD+TSD | 80.8 | 63.6 | 47.3 |

Table 7: Accuracy on out-of-domain datasets.

The resulting data sizes are listed in Appendix A.3. Then we finetune BERT$_{Base}$ to obtain finetuned teacher and perform distillation on student model with the finetuned teacher for each configuration. Results are presented in Table 6, from which we can see that the gain from GED is impressive when less task-specific data is used, especially when only 1% of the dataset can be used, the gain of GED can reach 4%, showing the importance of our method.

**Generalization Gain by GED** Besides its benefits on low-resource tasks, GED can also be considered as a stage to improve the generalization of the student, as it allows the student to capture task-specific knowledge on a much larger dataset. Several works (Laine and Aila, 2016; Sajjadi et al., 2016; Miyato et al., 2018; Goodfellow et al., 2014) succeeded in using random image augmentation to improve generalization performance for semi-supervised tasks. Similarly, at this stage, the hidden representation information still contains task-specific data distribution information, which can be used to compensate for the sparse task data and augment the feature representations. This leads to improving the generalization of the student model. To show that, we first distill ERNIE-Tiny on MNLI and then evaluate it on out-of-domain datasets including SNLI (Bowman et al., 2015) and RTE. As RTE is a 2-class classification task while MNLI is a 3-class classification task, we simply drop the "neural" and take argmax of "entailment" and "not entailment" when calculating accuracy on RTE. As shown in Table 7, experiment with GED exceed those without GED by a large margin. Specifically, with GED and TAD, the out-of-domain SNLI and RTE can improve 12 and 18.4 percent points respectively. In particular, although removing one of GED or TAD results in similar MNLI accuracy, the experiment with GED significantly outperforms the one without GED on all out-of-domain datasets, demonstrating the generalization benefit led by GED. Another interesting observation is that adding TAD can also be beneficial to the generalization of the student.

## 5 Conclusion

In this paper, we propose a progressive distillation framework ERNIE-Tiny to compress PLMs. Four-stage distillation is introduced to smooth the transition from pretrain distillation to finetune distillation. In particular, general-enhanced distillation employs finetuned teacher to deliver enhanced knowledge over general data to student model, boosting the generalization of student model. Task-adaptive distillation further smooths transition via carefully designed learning objectives. ERNIE-Tiny distilled from BERT$_{Base}$ retains 98% performance with 9.4x faster inference speed, achieving SOTA on GLUE benchmark with the same amount of parameters. Our 4-layer ERNIE-Tiny distilled from Chinese ERNIE2.0$_{Base}$ also outperforms 12-layer Chinese BERT$_{Base}$. Our work didn't apply larger unlabeled general data such as C4 (Raffel et al., 2019). More efficient data utilization is left for future work.

## 6 Broader Impact

As we have introduced four stages in our framework, it naturally causes concerns about the computation cost brought by our method. One metric to measure the computation cost is carbon footprint introduced in (Patterson et al., 2021), and the calculation equation is shown as following:

$$F = (\mathcal{E}_{\text{train}} + q \times \mathcal{E}_{\text{inference}}) \times CO_2/KWh, \qquad (7)$$

where $\mathcal{E}_{\text{train}}$ and $\mathcal{E}_{\text{inference}}$ is the energy for training and inference respectively. $q$ is the number of calling for model inference, $CO_2/KWh$ is the emission of $CO_2$ per $KWh$. Assume the hardware and software environment are the same for teacher and student, the only factors affects equation 7 are the computation FLOPS and number of $q$. Figure 3, shows the total cost for ERNIE-Tiny and BERT, including the total training cost and inference cost. Please refer to A.8 for the calculation details. It can be seen that at a certain point of $q$, the computation for ERNIE-Tiny is much lower than BERT with nearly 10x.

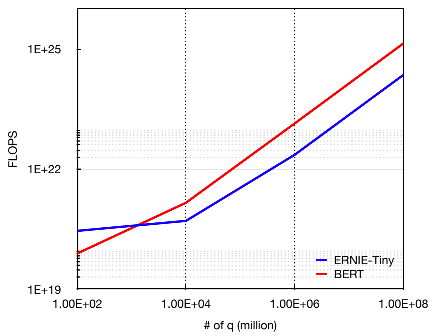


Figure 3: Cost Comparison Between ERNIE-Tiny and BERT with number of queries. The axes are shown in log scale.

ERNIE-Tiny distillation framework might seem expensive at first glance as it brings additional computation requirements compared to other existing works. However, as shown in Figure 3, when the number of model inferences is large enough, ERNIE-Tiny requires less computational resources than directly inferring with BERT. That being said, ERNIE-Tiny is more suitable for the scenarios where the number of model inferences is large such as real-life servers, and less suitable for those with small number of model inferences required.

Applying ERNIE-Tiny on real-life application with large number of requests, ERNIE-Tiny can significantly reduce carbon emission by 10x comparing to inferring with BERT. Furthermore, we have also discussed some interesting research questions in Appendix **??**.

## References


Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

Lei Jimmy Ba and Rich Caruana. 2013. Do deep nets really need to be deep? arXiv preprint arXiv:1312.6184.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In TAC.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. arXiv preprint arXiv:1708.00055.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2475–2485, Brussels, Belgium. 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, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. arXiv preprint arXiv:2101.03961.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

Song Han, Jeff Pool, John Tran, and William J Dally. 2015. Learning both weights and connections for efficient neural networks. arXiv preprint arXiv:1506.02626.

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

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.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361.

Jangho Kim, Yash Bhalgat, Jinwon Lee, Chirag Patel, and Nojun Kwak. 2019. Qkd: Quantization-aware knowledge distillation. arXiv preprint arXiv:1911.12491.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. arXiv preprint arXiv:1908.08593.

S. Kullback and R. A. Leibler. 1951. On information and sufficiency. Ann. Math. Statist., 22(1):79–86.

Samuli Laine and Timo Aila. 2016. Temporal ensembling for semi-supervised learning. arXiv preprint arXiv:1610.02242.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. Advances in Neural Information Processing Systems (NeurIPS).

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880, Online. Association for Computational Linguistics.

Jianquan Li, Xiaokang Liu, Honghong Zhao, Ruifeng Xu, Min Yang, and Yaohong Jin. 2020. BERT-EMD: Many-to-many layer mapping for BERT compression with earth mover's distance. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 3009–3018, Online. Association for Computational Linguistics.

Xin Liu, Qingcai Chen, Chong Deng, Huajun Zeng, Jing Chen, Dongfang Li, and Buzhou Tang. 2018. LCQMC:a large-scale Chinese question matching corpus. In Proceedings of the 27th International Conference on Computational Linguistics, pages 1952–1962, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE transactions on pattern analysis and machine intelligence, 41(8):1979–1993.

David A. Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. CoRR, abs/2104.10350.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.

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

Xingkai Ren, Ronghua Shi, and Fangfang Li. 2020. Distill bert to traditional models in chinese machine reading comprehension (student abstract). In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 13901–13902.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. arXiv preprint arXiv:1606.04586.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538.

Sungho Shin, Yoonho Boo, and Wonyong Sung. 1909. Empirical analysis of knowledge distillation technique for optimization of quantized deep neural networks. arXiv preprint arXiv.

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.

Vikas Sindhwani, Tara N Sainath, and Sanjiv Kumar. 2015. Structured transforms for small-footprint deep learning. arXiv preprint arXiv:1510.01722.

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

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019a. 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.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019b. Ernie: Enhanced representation through knowledge integration. arXiv preprint arXiv:1904.09223.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019c. Ernie 2.0: A continual pre-training framework for language understanding. arXiv preprint arXiv:1907.12412.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2158–2170, Online. Association for Computational Linguistics.

Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. 2016. Do deep convolutional nets really need to be deep and convolutional? arXiv preprint arXiv:1603.05691.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.

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 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

Wei Wang, Ming Yan, and Chen Wu. 2018a. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. arXiv preprint arXiv:1811.11934.

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

Yunhe Wang, Chang Xu, Chao Xu, and Dacheng Tao. 2018b. Packing convolutional neural networks in the frequency domain. IEEE transactions on pattern analysis and machine intelligence, 41(10):2495–2510.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. Transactions of the Association for Computational Linguistics, 7:625–641.

Yi Wei, Xinyu Pan, Hongwei Qin, Wanli Ouyang, and Junjie Yan. 2018. Quantization mimic: Towards very tiny cnn for object detection. In Proceedings of the European conference on computer vision (ECCV), pages 267–283.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. arXiv preprint arXiv:1704.05426.

Suxiang Zhang, Ying Qin, Wen-Juan Hou, and Xiaojie Wang. 2006. Word segmentation and named entity recognition for sighan bakeoff3. In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, pages 158–161.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015a. 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.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015b. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pages 19–27. IEEE Computer Society.

# A Appendix

## A.1 Pretraining Dataset Details

For ERNIE-Tiny, **GD** and **GED** are trained on pretraining dataset. Specifically, we use Wikipedia (English Wikipedia dump[1]; 12GB), BookCorpus (Zhu et al., 2015b) (4.6GB) for those two steps. Table 8 shows statistics of the pretraining data.

| Source | Tokens | Avg doc len |
|---|---|---|
| Wikipedia | 3.0B | 515 |
| BookCorpus | 1.2B | 23K |

Table 8: Pretraining data statistics.

## A.2 Task Dataset Details

**GLUE** The General Language Understanding Evaluation (GLUE) benchmark is a well-studied collections of nine natural language understanding tasks, including:

- **CoLA**: The Corpus of Linguistic Acceptability (CoLA)(Warstadt et al., 2019) is commonly used to judge whether a sentence conforms to the syntax specification, consisting of 10657 sentences from 23 linguistics, annotated for acceptability (grammatically) by their original authors.
- **SST-2**: The Stanford Sentiment Treebank (SST-2)(Socher et al., 2013) is a sentiment analysis task consisting of 9645 movie reviews.
- **MNLI**: Multi-genre Natural Language Inference (MNLI)(Williams et al., 2017) is a textual inference task, including 433k sentence pairs annotated with textual entailment information.
- **RTE**: Recognizing Textual Entailment (RTE)(Bentivogli et al., 2009) is a Natural Language Inference task, similar to MNLI.
- **WNLI**: Winograd Natural Language Inference (WNLI)(Levesque et al., 2012) is a task that needs capturing the coreference information between two paragraphs.
- **QQP**: Quora Question Pairs (QQP)[2] is a task for detecting whether the question pairs are duplicates or not, consisting of over 400,000 sentence pairs with data extracted from Quora QA community.
- **MRPC**: Microsoft Research Paraphrase Corpus (MRPC)(Dolan and Brockett, 2005) is a task that requires the model to capture the paraphrase or semantic relationship between a pair of sentences. it contains 5800 pairs of sentences extracted from web-crawled news.
- **STS-B**: The Semantic Textual Similarity Benchmark (STS-B)(Cer et al., 2017) contains a selection of English datasets containing texts from image captions, news headlines, and user forums.
- **QNLI**: Question Natural Language Inference (QNLI)(Rajpurkar et al., 2016; Wang et al., 2018a) is a task that requires the mode to classify if the given premise is the answer to the hypothesis.

**Chinese Datasets** We have chosen the following 5 Chinese NLP datasets to evaluate ERNIE-Tiny. Like GLUE, the collections of Chinese datasets also covers various NLP tasks. The details of the chosen Datsets are listed below:

- **XNLI (Conneau et al., 2018)**: The Cross-lingual Natural Language Inference (XNLI) is the extension of MNLI to multiple languages. The train set of XNLI is translated by machines, and the dev set is translated by human experts. We took the Chinese version of XNLI.
- **ChnSentiCorp** [3]: ChnSentiCorp consists of 9600 samples collected from hotel reviews and is annotated for sentiment analysis.
- **MSRA-NER (SIGHAN 2006) (Zhang et al., 2006)**: MSRA-NER is a named entity recognition task containing 21000 examples annotated into three types: people, location, and organization.
- **LCQMC (Liu et al., 2018)**: LCQMC is a text similarity task consisting of 260,068 query-paragraph pairs collected from search engine logs. The similarity between question and paragraph is annotated by human experts.
- **NLPCC-DBQA** [4]: DBQA is a QA task

[1] https://dumps.wikimedia.org/enwiki/

[2] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

[3] https://github.com/pengming617/bert_classification

[4] http://tcci.ccf.org.cn/conference/2016/dldoc/evagline2.pdf

| Corpus | Task | #Train | #Dev | #Test | #Label | Metrics |
|---|---|---|---|---|---|---|
| CoLA | Acceptability | 8.5k | 1k | 1k | 2 | Matthews corr |
| SST-2 | Sentiment | 67k | 872 | 1.8k | 2 | Accuracy |
| MNLI | NLI | 393k | 20k | 20k | 3 | Accuracy |
| RTE | NLI | 2.5k | 276 | 3k | 2 | Accuracy |
| WNLI | NLI | 634 | 71 | 146 | 2 | Accuracy |
| QQP | Paraphrase | 364k | 40k | 391k | 2 | Accuracy/F1 |
| MRPC | Paraphrase | 3.7k | 408 | 1.7k | 2 | Accuracy/F1 |
| STS-B | Similarity | 7k | 1.5k | 1.4k | 1 | Pearson/Spearman corr |
| QNLI | QA/NLI | 108k | 5.7k | 5.7k | 2 | Accuracy |

Table 9: The details of GLUE benchmark. The #Train, #Dev and #Test denote the size of the training set, development set and test set of corresponding corpus respectively. The #label denotes the size of the label set of the corresponding corpus.

consisting of 182K question-document pairs. Though one-to-many relation is presented in training data, we cast this task into a sentence-pair classification problem.

### A.3 Size of Resulted Low-source Datasets

We vary the task-specific dataset size of MNLI, QNLI, and QQP tasks to 1%, 10%, and 50% of the original size, the resulting data sizes are listed in Table 12.

### A.4 Algorithm

**Algorithm 1** ERNIE-Tiny Progressive Distillation Framework

$step \leftarrow 0$
**while** $step \leq E_{GD}$ **do**
  $\theta_{step+1} \leftarrow \theta_{step} - \beta_{GD} \nabla_\theta \mathcal{L}_{GD}$ ▷ GD
**end while**
$step \leftarrow 0$
**while** $step \leq E_{GED}$ **do**
  $\theta_{step+1} \leftarrow \theta_{step} - \beta_{GED} \nabla_\theta \mathcal{L}_{GED}$ ▷ GED
**end while**
$step \leftarrow 0$
**while** $step \leq E_{TAD}$ **do**
  $\theta_{step+1} \leftarrow \theta_{step} - \beta_{TAD} \nabla_\theta \mathcal{L}_{TAD}$ ▷ TAD
**end while**
$step \leftarrow 0$
**while** $step \leq E_{TD}$ **do**
  $\theta_{step+1} \leftarrow \theta_{step} - \beta_{TD} \nabla_\theta \mathcal{L}_{TD}$ ▷ TD
**end while**

Algorithm 1 shows the overall procedure of ERNIE-Tiny. $E_{GD}$, $E_{GED}$, $E_{TAD}$, $E_{TD}$, $\beta_{GD}$, $\beta_{GED}$, $\beta_{TAD}$ and $\beta_{TD}$ are the training steps and learning rate of these four stage respectively. As shown in this algorithm, the student resulted from each stage are used as initialization for next stage.

### A.5 Evaluation on Chinese Datasets

**Dataset** 5 Chinese NLP datasets are chosen for evaluating ERNIE-Tiny, including XNLI (Conneau et al., 2018) for natural language inference, LCQMC (Liu et al., 2018) for semantic similarity, ChnSentiCorp[5] for sentiment analysis, NLPCC-DBQA[6] for question answering and MSRA-NER (Zhang et al., 2006) for named entity recognition. All results reported in this section are calculated by taking the average on the dev set result of 5 runs. Please refer to Appendix A.9 for details.

**Result** Since most of the compression models do not experiment on Chinese tasks, we reproduce TinyBERT for comparison. Both TinyBERT and ERNIE-Tiny are distilled from an strong teacher Chinese ERNIE2.0$_{Base}$ (Sun et al., 2019c) instead of Chinese BERT$_{Base}$. It can be seen in Table 11 that ERNIE-Tiny outperforms Chinese BERT$_{Base}$[7] on XNLI, LCQMC, ChnSentiCorp and NLPCC-DBQA, and exceeds it by 0.4% average score over the five datasets, while being 7.5x smaller and 9.4x faster in inference time.

### A.6 Speed Up Calculation

We followed how TinyBERT (Jiao et al., 2020) evaluates inference speedup (i.e. evaluating the inference time on a single NVIDIA K80 GPU) and obtained the same result with TinyBERT as our

[5] https://github.com/pengming617/bert_classification
[6] http://tcci.ccf.org.cn/conference/2016/dldoc/evagline2.pdf
[7] https://github.com/google-research/bert

| Corpus | Task | #Train | #Dev | #Test | #Label | Metrics |
|---|---|---|---|---|---|---|
| XNLI | NLI | 392k | 2.5k | 2.5k | 3 | Accuracy |
| ChnSentiCorp | SA | 9.6k | 1.2k | 1.2k | 2 | Accuracy |
| MSRA-NER | NER | 21k | 2.3k | 4.6k | 7 | F1 |
| LCQMC | SS | 240k | 8.8k | 12.5k | 2 | Accuracy |
| NLPCC-DBQA | QA | 182k | 41k | 82k | 2 | mrr/F1 |

Table 10: The details of Chinese NLP datasets. The #Train, #Dev and #Test denote the size of the training set, development set and test set of corresponding corpus respectively. The #label denotes the size of the label set of the corresponding corpus.

| Method | Params | Speedup | XNLI | LCQMC | ChnSentiCorp | NLPCC-DBQA | MSRA-NER | Avg. |
|---|---|---|---|---|---|---|---|---|
| ERNIE2.0$_{Base}$ (T.) | 109M | 1x | 79.8 | 87.5 | 95.5 | 84.4 | 95.0 | 88.4 |
| BERT$_{Base}$ | 109M | 1x | 77.2 | 87.0 | 94.3 | 80.9 | **92.3** | 86.3 |
| TinyBERT (re.) | 14.5M | 9.4x | 76.3 | 86.8 | 94.2 | 81.8 | 87.3 | 85.3 |
| ERNIE-Tiny | **14.5M** | **9.4x** | **77.6** | **88.0** | **94.9** | **82.2** | 90.8 | **86.7** |

Table 11: Test Results of Chinese Tasks. TinyBERT on this table is reproducted by us. The teacher of TinyBERT and ERNIE-Tiny($L$=4, $d$=312, $d_{ff}$=1200) are set to Chinese ERNIE2.0$_{Base}$. Both BERT$_{Base}$ and ERNIE2.0$_{Base}$ is ($L$=12, $d$=768, $d_{ff}$=3072).

| proportion | MNLI | QNLI | QQP |
|---|---|---|---|
| 1% | 3927 | 1047 | 3638 |
| 10% | 39270 | 10474 | 36384 |
| 50% | 196351 | 52371 | 181925 |

Table 12: Number of labeled data.

model has the same architecture and parameters as TinyBERT

### A.7 Sensitive To Hyperparameters

We empirically found that the final performance is insensitive to most hyper-parameters and most hyper-parameters in our experiment can be adopted in practice, except for the number of training steps in TAD which requires adjustment based on the size of the task datasets (e.g. reduce it for large task datasets). Take the hyper-parameters used in our experiment as examples, the hyper-parameters for experiments with Chinese datasets are mostly the same as for GLUE.

### A.8 Details on Computation Cost

| Arch. | Train | Inference |
|---|---|---|
| BERT$_{Base}$ | 6.43E+19 | 1.37E+11 |
| ERNIE-Tiny | 5.79E+19 | 2.24E+10 |

Table 13: Computation FLOPS for both training and inference.

In this section, we will describe the calculation details for Figure 3. As shown in Table 13, we list the training and inference FLOPS for both BERT$_{Base}$ and ERNIE-Tiny respectively [8]. The training computation cost is recorded for total training process, and the inference computation cost is recorded for one sample feedfoward. So the total distillation cost for ERNIE-Tiny can be calculated as

$$C_T = T_{student} + I_{teacher} \times S \times Batchsize,$$

where $T_{student}$ denotes that FLOPS caused by student training such as forward/backward propagation and updating parameters which can be found in Table 13. $I_{teacher}$ denotes the FLOPS for a single inference or forward propagation on Teacher. $S$ denotes the total training steps.

The inference cost for ERNIE-Tiny can be calculated as

$$C_{Infer} = I_{student} \times S \times Batchsize,$$

where $I_{student}$ denotes the FLOPS for a single inference or forward propagation on ERNIE-Tiny.

### A.9 Implementation Details

Table 14 gives the detailed hyper-parameters used in our ablation experiment on GLUE tasks. Note

[8]Those result can be calculated with the opensourced scripts at https://tinyurl.com/47ercmu9

that dropout intervenes in hidden states and hurts distillation performance, thus all dropout rates were set to 0 in our experiment setting. Also, we find that using AMP hurts distillation performance due to inaccuracy of hidden state representation, so we use pure FP32 training in all experiments. The GED stage takes about 2 days on 4 16g-V100, while the computation cost for other tasks no more than 1 day. Hyper-parameters for our Chinese results are listed in Table 15.

| Hyper parameters | MNLI | QQP | QNLI | RTE | SST-2 | STS-B | MRPC | CoLA |
|---|---|---|---|---|---|---|---|---|
| GD | | | | | | | | |
| batch size | 2000 | 2000 | 2000 | 2000 | 2000 | 2000 | 2000 | 2000 |
| learning rate | 4e-4 | 4e-4 | 4e-4 | 4e-4 | 4e-4 | 4e-4 | 4e-4 | 4e-4 |
| training steps | 500K | 500K | 500K | 500K | 500K | 500K | 500K | 500K |
| optimizer | Adam | Adam | Adam | Adam | Adam | Adam | Adam | Adam |
| warmup steps | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 | 5000 |
| dropout rate | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| GED | | | | | | | | |
| batch size | 1280 | 1280 | 1280 | 1280 | 1280 | 1280 | 1280 | 1280 |
| learning rate | 2e-4 | 2e-4 | 2e-4 | 2e-4 | 2e-4 | 2e-4 | 2e-4 | 2e-4 |
| training steps | 500K | 500K | 500K | 500K | 500K | 500K | 500K | 500K |
| optimizer | Adam | Adam | Adam | Adam | Adam | Adam | Adam | Adam |
| warmup steps | 500 | 500 | 500 | 500 | 500 | 500 | 500 | 500 |
| dropout rate | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| TAD | | | | | | | | |
| batch size | 256 | 256 | 256 | 128 | 128 | 128 | 128 | 128 |
| learning rate | 5e-5 | 5e-5 | 5e-5 | 5e-5 | 5e-5 | 5e-5 | 5e-5 | 5e-5 |
| epoch | 5 | 5 | 5 | 10 | 10 | 10 | 10 | 50 |
| optimizer | Adam | Adam | Adam | Adam | Adam | Adam | Adam | Adam |
| warmup proportion | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| dropout rate | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| TSD | | | | | | | | |
| batch size | 256 | 256 | 256 | 128 | 128 | 128 | 128 | 128 |
| learning rate | 1e-5 | 1e-5 | 1e-5 | 3e-5 | 3e-5 | 3e-5 | 3e-5 | 3e-5 |
| epoch | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| optimizer | Adam | Adam | Adam | Adam | Adam | Adam | Adam | Adam |
| warmup proportion | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| dropout rate | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

Table 14: Hyper-parameters for ablation studies on GLUE tasks.

| Hyper parameter | XNLI | ChnSentiCorp | MSRA-NER | LCQMC | NLPCC-DBQA |
|---|---|---|---|---|---|
| GD | | | | | |
| batch size | 2000 | 2000 | 2000 | 2000 | 2000 |
| learning rate | 4e-4 | 4e-4 | 4e-4 | 4e-4 | 4e-4 |
| training steps | 500K | 500K | 500K | 500K | 500K |
| optimizer | Adam | Adam | Adam | Adam | Adam |
| warmup steps | 5000 | 5000 | 5000 | 5000 | 5000 |
| dropout rate | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| GED | | | | | |
| batch size | 1280 | 1280 | 1280 | 1280 | 1280 |
| learning rate | 2e-4 | 2e-4 | 2e-4 | 2e-4 | 2e-4 |
| training steps | 500K | 500K | 500K | 500K | 500K |
| optimizer | Adam | Adam | Adam | Adam | Adam |
| warmup steps | 500 | 500 | 500 | 500 | 500 |
| dropout rate | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| TAD | | | | | |
| batch size | 256 | 128 | 256 | 128 | 128 |
| learning rate | 5e-5 | 5e-5 | 5e-5 | 5e-5 | 5e-5 |
| epoch | 5 | 10 | 5 | 10 | 50 |
| optimizer | Adam | Adam | Adam | Adam | Adam |
| warmup proportion | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| dropout rate | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| TSD | | | | | |
| batch size | 256 | 128 | 256 | 128 | 128 |
| learning rate | 1e-5 | 3e-5 | 1e-5 | 3e-5 | 3e-5 |
| epoch | 3 | 3 | 3 | 3 | 3 |
| optimizer | Adam | Adam | Adam | Adam | Adam |
| warmup proportion | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| dropout rate | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

Table 15: Hyper-parameters for evaluation on Chinese Datasets.