Gradient-based Intra-attention Pruning on Pre-trained Language Models

Ziqing Yang[†], Yiming Cui[‡†], Xin Yao[†], Shijin Wang[†§]
[†]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, Beijing, China
[‡]Research Center for SCIR, Harbin Institute of Technology, Harbin, China
[§]iFLYTEK AI Research (Central China), Wuhan, China
[†]`{zqyang5,ymcui,xinyao10,sjwang3}@iflytek.com`
[‡]`ymcui@ir.hit.edu.cn`

Abstract

Pre-trained language models achieve superior performance but are computationally expensive. Techniques such as pruning and knowledge distillation have been developed to reduce their sizes and latencies. In this work, we propose a structured pruning method **GRAIN** (**Gra**dient-based **In**tra-attention pruning), which performs task-specific pruning with knowledge distillation and yields highly effective models. Different from common approaches that prune each attention head as a whole, GRAIN inspects and prunes intra-attention structures, which greatly expands the structure search space and enables more flexible models. We also propose a gradient separation strategy that reduces the interference of distillation on pruning for a better combination of the two approaches. Experiments on GLUE, SQuAD, and CoNLL 2003 show that GRAIN notably outperforms other methods, especially in the high sparsity regime, and achieves $6 \sim 7\times$ speedups while maintaining $93\% \sim 99\%$ performance. Under extreme compression where only 3% transformer weights remain, the pruned model is still competitive compared to larger models.[1]

1 Introduction

Transformer-based (Vaswani et al., 2017) pre-trained language models (PLMs) have achieved great success and become the backbones of various natural language processing tasks. However, PLMs are computationally expensive and slow in inference due to their large sizes, which limits their applications in real-world scenarios. Hence, a growing interest has been in developing compression and acceleration methodologies for PLMs.

A common approach to model compression is structured pruning, which compresses the model by removing groups of consecutive parameters, namely the pruning units. In applying structured pruning on PLMs, recent works have investigated removing units such as hidden dimensions in feed-forward layers, attention heads in the multi-head attention (Michel et al., 2019; Li et al., 2022), and coarse-grained units such as multi-head attention layers and feed-forward layers (Xia et al., 2022).

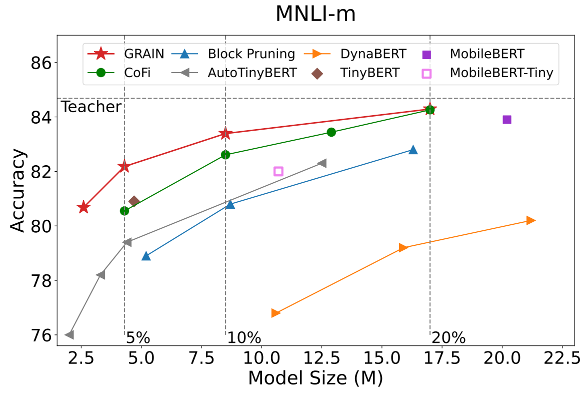

Figure 1: A comparison of GRAIN and other distillation and pruning methods on MNLI-m development set at different model sizes. More details are in Section 5.

However, these pruning units only span a small space of model structures and limit the exploration for better structures. For example, in the pruning of BERT$_{\text{base}}$ (Devlin et al., 2019), which contains 144 attention heads, the possible choices of attention heads for the pruned model are limited. Block Pruning (Lagunas et al., 2021) extends pruning units by considering blocks in the weight matrices, but Block Pruning is not a fully structured pruning method and can not achieve large speedups.

In this work, we propose **GRAIN** (**Gra**dient-based **In**tra-attention pruning), a structured pruning method that prunes PLMs with finer pruning units. In the following, we present the method from three aspects: pruning units, pruning algorithm, and training objectives.

Pruning Units Unlike attention heads pruning where the pruning unit is a single head, we propose intra-attention pruning, which inspects and prunes

[1]Code is available at https://github.com/airaria/GRAIN.

the structures inside attention heads. Intra-attention pruning greatly expands the search space of model structures, making the resulting models more likely to find better structures. However, directly applying intra-attention pruning yields fragmented models, i.e., models with many small heads. The fragmented models have relatively large latencies on devices like GPUs. To overcome the shortcoming, we introduce structure regularization, which encourages prioritizing specific units for pruning. Structure regularization helps generate more regular structures and achieve lower latencies.

Pruning Algorithm Pruning algorithms decide which units to be removed. We adapt the gradient-based pruning algorithm (Michel et al., 2019) for intra-attention pruning. Gradient-based pruning is a light-weighted method that estimates the importance of the pruning units with gradient-based scores and then prunes the least important ones. In addition, we conduct the pruning in an iterative manner (Zhu and Gupta, 2018), i.e., the model is gradually pruned during fine-tuning. The iterative approach has been employed in combination with pruning algorithms such as Movement Pruning (Sanh et al., 2020) and Magnitude Pruning (Zhu and Gupta, 2018), but few works have combined it with gradient-based pruning. We find that iterative gradient-based pruning is especially effective despite its simplicity.

Training Objectives As another common approach to model compression, knowledge distillation offers highly effective training objectives (Jiao et al., 2020). Pruning with distillation objective shows improved performance (Sanh et al., 2020; Xia et al., 2022). However, in gradient-based pruning, the distillation objectives may disturb the estimation of importance scores. We propose a gradient separation strategy that uses different gradients for model optimization and importance score estimation. We show that this method leads to better performance.

GRAIN performs task-specific pruning without additional pre-training or data augmentation. In the experiments, we compare GRAIN with strong pruning and distillation baselines on GLUE, SQuAD, and CoNLL 2003. GRAIN notably outperforms the comparable methods in the high-sparsity regime. A demonstration of the results on MNLI is shown in Figure 1. While keeping 5% parameters in transformers, GRAIN maintains $93\% \sim 99\%$ performance of BERT$_{\text{base}}$ and $6 \sim 7\times$ speedups across different tasks. Furthermore, GRAIN still achieves competitive results even under extreme compression where only 3% transformer weights remain.

2 Related Work

A growing number of works have been devoted to the compression and acceleration of PLMs. Most of the works have combined multiple techniques.

Knowledge Distillation (Hinton et al., 2015) is a training technique that trains a student model to mimic the outputs and intermediate representations of the teacher model (Sun et al., 2019). DistilBERT (Sanh et al., 2019) and TinyBERT (Jiao et al., 2020) are both small BERT-like models distilled with general and task-specific distillation. MobileBERT (Sun et al., 2020) and KroneckerBERT (Tahaei et al., 2022) have designed novel structures for student models. Chen et al. (2021) proposes to extract a subnetwork from the teacher and then perform distillation. AutoTinyBERT (Yin et al., 2021) combine distillation with neural architecture search to find optimal hyperparameters. DynaBERT (Hou et al., 2020) apply task-specific distillation and can flexibly adjust the model size. In this work, we only apply task-specific distillation, which consumes fewer resources.

Structured Pruning on PLMs remove different types of units from the models, like attention heads (Michel et al., 2019), FFN hidden dimensions (Liang et al., 2021), blocks of weights (Lagunas et al., 2021), MHA layers or FFN layers (Xia et al., 2022). Many works combine pruning with other methods. Wang et al. (2020) presents a structured pruning approach with low-rank factorization of weight matrices. McCarley (2019) and Xia et al. (2022) apply pruning with knowledge distillation. In this work, we apply matrix factorization on the embeddings and use distillation and pruning to reduce the size of transformers.

Unstructured Pruning removes each weight individually based on its magnitude (Han et al., 2015; Zhu and Gupta, 2018; Gordon et al., 2020), or the score computed by first-order (Sanh et al., 2020; Louizos et al., 2017) or second-order (Kurtic et al., 2022) method. Unstructured pruning yields higher sparsity models but is hard to speed up without specialized devices for sparse matrix operations. In this work, we only consider structured pruning.

Besides model compression, another group of acceleration methods is dynamic inference, where the computation cost is determined at test time (Fan et al., 2020; Liu et al., 2020; Xin et al., 2020). Liu et al. (2021) and Shen et al. (2022) have proposed to integrate model compression with dynamic inference. We do not consider dynamic inference in this work and leave it for future work.

3 Preliminaries

3.1 Transformers

A Transformer block (Vaswani et al., 2017) is mainly composed of a multi-head attention (MHA) layer and a feed-forward network (FFN) layer.

Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be the input sequence, where n is the length, and d is the hidden size. An attention head is parameterized by the matrices $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V, \boldsymbol{W}_i^O \in \mathbb{R}^{d_h \times d}$. Its output is[2]

$$\mathrm{Att}_i(\boldsymbol{X}) = \mathrm{softmax}\left(\boldsymbol{Q}_i \boldsymbol{K}_i^{\mathsf{T}} / \sqrt{d}\right) \boldsymbol{V}_i \boldsymbol{W}_i^O, \quad (1)$$
$$\boldsymbol{Q}_i = \boldsymbol{X}(\boldsymbol{W}_i^Q)^{\mathsf{T}}, \boldsymbol{K}_i = \boldsymbol{X}(\boldsymbol{W}_i^K)^{\mathsf{T}}, \boldsymbol{V}_i = \boldsymbol{X}(\boldsymbol{W}_i^V)^{\mathsf{T}},$$

where d_h is head size, and i is the head index. An MHA layer contains $N_h = d/d_h$ attention heads

$$\mathrm{MHA}(\boldsymbol{X}) = \sum\nolimits_i^{N_h} \mathrm{Att}_i(\boldsymbol{X}). \quad (2)$$

Following the MHA layer is the feed-forward network layer. It consists of two linear layers and a GeLU activation (Hendrycks and Gimpel, 2016)

$$\mathrm{FFN}(\boldsymbol{X}) = \mathrm{GeLU}(\boldsymbol{X} \cdot \boldsymbol{W}_1) \cdot \boldsymbol{W}_2, \quad (3)$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d \times d_f}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_f \times d}$, and d_f is the intermediate hidden size. Typically $d_f > d$.

A transformer block contains other components, such as LayerNorm and residual connection, but they only take up a few parameters.

3.2 Gradient-based Pruning

Gradient-based pruning (Michel et al., 2019) defines the importance score of a pruning unit w as the variation of the loss with respect to the unit:

$$\mathrm{IS}(w) = \mathbb{E}_{x \sim X} \left| \frac{\partial \mathcal{L}(x)}{\partial w} w \right|, \quad (4)$$

where X is the data distribution. The term in the absolute value is the first-order Taylor approximation of the loss $\mathcal{L}$ around $w = 0$. To apply (4) in PLM pruning, w should be set accordingly. For example, by setting w to $\boldsymbol{W}_i^O$, Equation (4) gives the importance score of the head h_i; by setting w to the i-th row of $\boldsymbol{W}_2$, Equation (4) gives the importance score of the i-th FFN hidden dimension. A lower importance score implies that the loss is less sensitive to the unit. The pruning units are sorted and then pruned in the order of increasing scores.

[2]We omit bias terms throughout for simple presentation.

4 Methodology

GRAIN performs task-specific intra-attention pruning together with knowledge distillation. The overview of GRAIN is depicted in Figure 2. Following previous works, we only include the encoder in counting the model size unless otherwise specified. We refer to the size of the pruned model relative to the unpruned model as *model density*:

$$\text{model density} = \frac{\texttt{SizeOf(pruned model)}}{\texttt{SizeOf(original model)}}.$$

Sparsity is equal to one minus model density.

4.1 Intra-attention Pruning

4.1.1 Intra-attention Pruning Units

FFN hidden dimensions and attention heads are common pruning units in PLM pruning studies. These pruning units have been treated as atomic in structured pruning. However, attention heads include finer pruning units and are not really atomic. Equation (2) shows that the output of an MHA layer is the sum of individual heads, so different heads can be pruned independently. To be specific, We can remove the rows of the matrices $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V, \boldsymbol{W}_i^O$ to reduce head size. Further, from Equation (1), we see that the output dimensions of $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K$ and the input dimensions of $\boldsymbol{W}_i^V, \boldsymbol{W}_i^O$ can be different. It gives another freedom to set the dimensions of attention heads.

Based on the above observation, we introduce two kinds of intra-attention pruning units: query units, namely the rows of $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K$; and value units, namely the rows of $\boldsymbol{W}_i^V, \boldsymbol{W}_i^O$. We keep FFN hidden dimensions but discard attention heads as the pruning units since the intra-attention pruning units are more structurally fundamental. Each pruning unit takes $2d$ parameters. The new set of pruning units greatly expands the structure space.

In the actual implementation (Wolf et al., 2020), the parameters of all heads in an MHA layer are gathered and stored in four large matrices $\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V, \boldsymbol{W}^O \in \mathbb{R}^{d \times d}$. The parameters of the i-th head are stored in the rows $(i, i + d_h)$. We prune query and value units from large matrices by removing corresponding rows. The pruning units are illustrated in the right part of Figure 2.

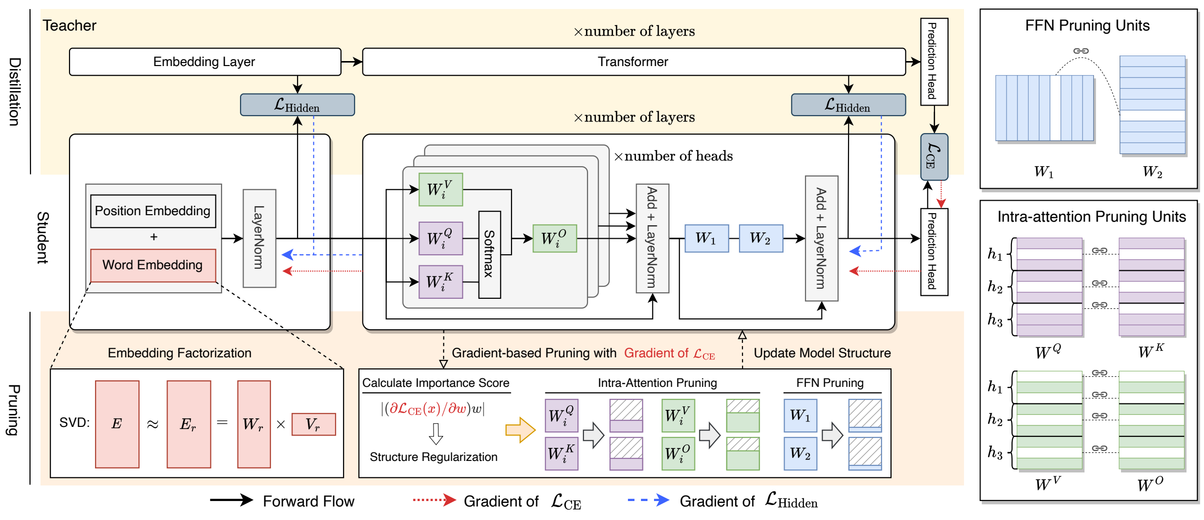

Figure 2: An overview of GRAIN. Left: The distillation and pruning methodology of the student model. Right: The pruning units in intra-attention pruning. We show three heads for illustration. Each head can be pruned to a different size. The pruned units are in white. The dashed lines indicate that the connected rows or columns should be removed together.

4.1.2 Structure Regularization

Since intra-attention pruning removes the units inside attention heads, it tends to generate models with many small heads of different sizes, but the total number of heads can still be large. We refer to this kind of structure as fragmented (see the upper panel in Figure 6 for an example). The fragmented structure has low efficiency on devices like GPUs since there are still many attention modules left in the model, and these heads are hard to parallelize.

To remedy this, we introduce **Structure Regularization** (**StructReg** for short) to encourage generating less fragmented structures. Intuitively, to avoid small heads, the pruning process should first prune the units in the small heads and make them empty, which can then be safely removed.

To be general, we define $D(\mathcal{M}, \mathcal{W})$ as the density of a set of pruning units $\mathcal{W}$ in module $\mathcal{M}$, i.e., the ratio of the remaining units in $\mathcal{M}$. The regularized importance score of a unit $w \in \mathcal{W}$ is:

$$\text{IS}^r(w) = \text{IS}(w) \cdot \tanh(D(\mathcal{M}, \mathcal{W})/\alpha), \qquad (5)$$

where α is the regularization strength. The lower the density of the units in $\mathcal{M}$, the lower the regularized scores of the units. Hence, the units in low-density modules will be pruned with priority until all the units in $\mathcal{M}$ have been pruned, leaving fewer low-density modules in the pruned model.

StructReg can be applied on different levels by choosing different $\mathcal{M}$s and $\mathcal{W}$s. We apply it to intra-attention structures. We set $\mathcal{M}$ to each attention head and $\mathcal{W}$ to the value units in $\mathcal{M}$. Heads with fewer value units will be pruned with priority until empty, resulting in fewer small heads.

4.2 Knowledge Distillation

Distillation Objectives Knowledge distillation provides effective objectives for transferring knowledge from a large model to a small model. The most simple distillation objective involves a cross-entropy loss between the student's and the teacher's prediction probabilities

$$\mathcal{L}_{\text{CE}} = \boldsymbol{p}_\tau^{(T)} \cdot \log \boldsymbol{p}_\tau^{(S)}, \qquad (6)$$

where T and S denote *teacher* and *student* respectively, and $\boldsymbol{p}_\tau = \text{softmax}(\boldsymbol{z}/\tau)$ is the scaled probability with temperature τ and logits $\boldsymbol{z}$. By integrating logits distillation with hidden layer representation distillation (Jiao et al., 2020; Sun et al., 2020), the performance of knowledge distillation can be further improved:

$$\mathcal{L}_{\text{Hidden}} = \sum_{(i,j)\in\mathcal{I}} \text{MSE}(\boldsymbol{H}_i^{(S)}\boldsymbol{W}_i, \boldsymbol{H}_j^{(T)}), \qquad (7)$$

where $\mathcal{I}$ is the set of layer index pairs, $\boldsymbol{H}_i (i > 0)$ is the hidden states from the i-th transformer block ($\boldsymbol{H}_0$ is the output from the embedding layer), and $\boldsymbol{W}_i$ is a trainable linear mapping. We employ the sum of $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{Hidden}}$ as the total loss.

Gradient Separation When applying distillation with gradient-based pruning, the hidden layer matching loss $\mathcal{L}_{\text{Hidden}}$ should be treated carefully.

In gradient-based pruning, the units are pruned based on how significantly they affect the model predictions. Thus, the importance score should be calculated solely from the cross-entropy loss, and we should avoid the gradients from other losses like $\mathcal{L}_{\text{Hidden}}$ affecting the estimation of the importance scores. Therefore, we propose to use the gradient from $\mathcal{L}_{\text{CE}}$ for model optimization and importance score computation, while using the gradient from $\mathcal{L}_{\text{Hidden}}$ only for model optimization. We call this strategy **gradient separation** (GS). The gradient flows of different losses are illustrated in Figure 2.

4.3 Iterative Gradient-based Pruning

Iterative Pruning Similar to Sanh et al. (2020), we take an iterative approach to prune the model, i.e., the model size is gradually reduced during fine-tuning. We denote the total training steps as N and the current step as i. The model is pruned to the density $s(t)$ at every step, where $s(t)$ is the density scheduler as a function of the training percentage $t = i/N \in [0, 1]$. We will give the exact form of $s(t)$ shortly. Notice that in the standard gradient-based pruning, the importance score is estimated from all the examples in the dataset X (see Equation (4)). It would be impractical to estimate the score at every step. Therefore we define an exponentially smoothed importance score $\overline{\text{IS}}_i(w)$ which can be computed efficiently during training and used for pruning at step i:

$$\overline{\text{IS}}_i(w) = \beta \cdot \overline{\text{IS}}_{i-1}(w) + (1-\beta) \cdot \text{IS}_i(w), \quad (8)$$

where $\text{IS}_i(w)$ is the importance score of the pruning unit w calculated with a single batch at step i, and β is the smoothing factor. The smoothed score avoids the large variance and leads to more stability. Equation (8) can also be applied on the regularized score simply by replacing $\text{IS}(w)$ with $\text{IS}^r(w)$.

Scheduling Following Zhu and Gupta (2018), we use a cubic density scheduler $s(t)$

$$\begin{cases} 1 & 0 \le t < p_s \\ s_f + (1-s_f)(1-\frac{t-p_s}{p_e-p_s})^3 & p_s \le t \le p_e \\ s_f & p_e < t \le 1 \end{cases}.$$

The complete process can be divided into three stages, as depicted in Figure 3. The first stage is the warm-up stage. We train the student model for Np_s steps with the distillation objective, where $0 < p_s < 1$ is a hyperparameter. In the second stage, we gradually prune the model with distillation for $N(p_e - p_s)$ steps. The model density s decreases from the initial density (100%) to the target density s_f following the schedule. In the last stage, the model structure is fixed, and we continually train the model with distillation to recover performance (Sanh et al., 2020; Zhu and Gupta, 2018). The three stages take place consecutively, and the whole process is done in a single run of fine-tuning.

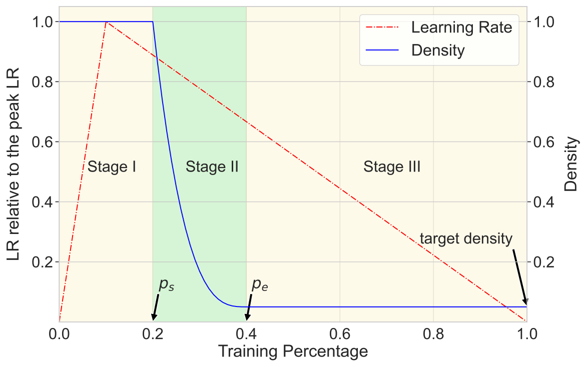

Figure 3: The training stages of GRAIN.

4.4 Embedding Factorization

The pruning mentioned above reduces the parameters in the transformers, while another large fraction of the parameters stored in the word embedding matrix is untouched. We apply singular value decomposition (SVD) to reduce the embedding size. SVD decomposes the word embedding matrix $\boldsymbol{E} \in \mathbb{R}^{q \times d}$ as $\boldsymbol{E} = \boldsymbol{U \Sigma V}$, where q is the vocabulary size and d is the hidden size, $\boldsymbol{U} \in \mathbb{R}^{q \times d}$, $\boldsymbol{V} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{\Sigma}$ is a diagonal matrix composed of singular values. $\boldsymbol{E}$ can be approximated as $\boldsymbol{E}_r$ by selecting top r singular values and corresponding r rows from $\boldsymbol{U}$ and $\boldsymbol{V}$

$$\boldsymbol{E} \approx \boldsymbol{E}_r = \boldsymbol{U}_r \Sigma_r \boldsymbol{V}_r = \boldsymbol{W}_r \boldsymbol{V}_r, \quad (9)$$

where $\boldsymbol{W}_r \in \mathbb{R}^{q \times r}$ and $\boldsymbol{U}_r \in \mathbb{R}^{r \times d}$. The original embedding $\boldsymbol{E}$ is now replaced by $\boldsymbol{W}_r$ and $\boldsymbol{V}_r$. The embedding size is reduced from qd to $(q+d)r$.

Embedding factorization has little effect on latencies but significantly reduces model sizes. Some works (Xia et al., 2022; Lagunas et al., 2021) do not prune embeddings. We also conduct experiments without embedding factorization for comparison. We name this setting as **GRAIN w/o EF**.

5 Experiments

5.1 Experiment Setup

Datasets We evaluate our approach on machine reading comprehension SQuAD 1.1 (Rajpurkar

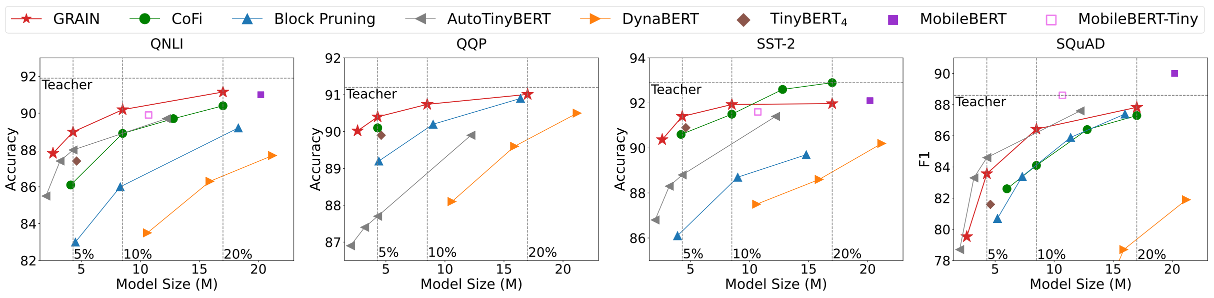

Figure 4: Performance of GRAIN and baseline methods at different model densities. The results of CoFi, AutoTinyBERT, and MobileBERT are taken from the original papers and the CoFi repository. Since the precise sizes of CoFi models around density 10%, 15%, and 20% are unknown, we use 10%, 15%, and 20% as estimates instead.

Model	**QNLI** (Acc)	**MNLI** (m/mm Acc)	**QQP** (Acc)	**SST-2** (Acc)	**SQuAD** (F1 / EM)	**CoNLL-03** (F1)	**Model Size**	**Total Size**
BERT$_{base}$ (teacher)	91.9	84.7 / 85.0	91.2	92.9	88.6 / 81.1	91.2	85.1M	108.9M
5% Model Density								
TinyBERT$_4$†	87.4	80.9 / 81.9	89.9	90.9	81.6 / 71.9	84.9	4.7M (5.5%)	14.6M
AutoTinyBERT§	88.0	79.4 / -	87.7	88.8	**84.6** / -	-	4.3M (5.0%)	14.5M
Block Pruning†	83.0	78.9 / 78.6	89.2	86.1	80.7 / 71.0	84.0	4.6M (5.4%)	28.8M
CoFi (reimpl.)†	85.3	79.8 / 79.6	89.8	89.8	79.0 / 69.2	85.0	4.2M (4.9%)	28.2M
CoFi§	86.1	80.6 / 80.7	90.1	90.6	82.6 / -	-	4.7M (5.5%)‡	29.0M‡
GRAIN	89.0	82.2 / **82.5**	90.4	91.4	83.6 / 73.7	**88.3**	4.3M (5.0%)	10.7M
GRAIN w/o EF	**89.1**	**82.4** / 82.2	**90.5**	**91.6**	83.4 / 73.2	**88.3**	4.3M (5.0%)	28.1M
3% Model Density								
GRAIN	**87.8**	80.7 / 81.1	90.0	90.4	**79.5 / 68.4**	86.8	2.6M (3.0%)	9.0M
GRAIN w/o EF	87.6	**81.0 / 81.2**	**90.2**	**91.0**	79.0 / 67.3	**87.2**	2.6M (3.0%)	26.4M

Table 1: Results of GRAIN and the baselines at model density around 5% and 3%. The best results are in bold. We average the sizes across different tasks in counting the model and total size. §: The results are taken from the original papers. †: The results are obtained by our reimplementation with the released code. ‡: The size is averaged excluding QQP since there are no public CoFi model checkpoints for the QQP task.

et al., 2016), named entity recognition CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003), and four classification tasks (SST-2, QNLI, MNLI, and QQP) that have relative large training data from GLUE benchmark (Wang et al., 2018). Details are summarized in Appendix B. We report the results on the development sets of GLUE and SQuAD and the results on the test set of CoNLL 2003.

Training Settings We use BERT$_{base}$ as the backbone model.[3] We first fine-tune the teachers for each task, then train and prune the students following the procedure in Section 4.3. The target model densities range from 3% to 20%. We list the model size and the total size (with embeddings and classifiers) for reference. We report the mean score of 3 runs with different random seeds. See Appendix A for training details and costs.

Baselines We compare our proposed method with **CoFi** (Xia et al., 2022), **Block Pruning** (Lagunas et al., 2021), **TinyBERT**$_4$ (Jiao et al., 2020) and **DynaBERT** (Hou et al., 2020). We also list the results of **AutoTinyBERT** (Yin et al., 2021) and **MobileBERT** (Sun et al., 2020). However, they are not directly comparable to GRAIN since they have been distilled from different teacher models and pre-trained extensively, consuming much more computation. Following Xia et al. (2022), we re-implement TinyBERT$_4$ and DynaBERT without task-specific data augmentation for a fair comparison. We also re-implement CoFi and Block Pruning with their public code, and choose *Hybrid Filled* approach as the Block Pruning baseline. We use the same teachers in training for GRAIN, TinyBERT$_4$, CoFi, and Block Pruning.

5.2 Main Results

In Figure 1 and Figure 4, we show the scores of GRAIN and the baseline methods on various downstream tasks with model densities ranging

[3]We also experiment with RoBERTa (Liu et al., 2019) and Chinese-RoBERTa-wwm-ext (Cui et al., 2021) on Chinese tasks. See Appendix E for details.

Method	**QNLI**	**SST-2**	**SQuAD**
GRAIN	89.0	91.4	83.6
GRAIN w/o EF	89.1	91.6	83.4
– StructReg	89.4	92.2	83.1
– GradSep	89.3	92.0	82.8
– Hidden Layer Loss	86.1	88.1	80.3
– Importance Scores	82.3	88.0	65.7

Table 2: Ablation results at 5% model density.

Units	**(FFN, Heads)**	**QNLI**	**SQuAD**
Intra+FFN	(3.5%, 7,9%)	**89.0**	
Intra+FFN	(3.5%, 8.0%)		**83.6**
Heads+FFN	(5.0%, 5.0%)	87.3	77.3
Heads+FFN	(3.75%, 7.5%)	88.2	79.2
Heads+FFN	(3.0%, 9.0%)	<u>88.5</u>	<u>81.4</u>
Heads+FFN	(2.5%, 10%)	<u>88.5</u>	80.9
Heads+FFN	(1.5%, 12%)	88.2	80.8

Table 3: Comparison between different pruning units at 5% model density. Heads+FFN denotes pruning with attention heads and FFN hidden dimensions. Intra+FFN denotes pruning with intra-attention units and FFN hidden dimensions. The best results are shown in bold. The best results with Heads+FFN are underlined.

from 3% to 20%. Table 1 summarizes the detailed results at densities 5% and 3%.[4] We see that GRAIN outperforms baselines in the majority of tasks on a wide range of model sizes. GRAIN outperforms TinyBERT$_4$ and Block Pruning on all tasks and outperforms CoFi except on SST-2 at relatively high density. Especially, in the low-density regime, GRAIN exhibits notable advantages over other methods. Under extreme compression at density 3%, GRAIN (2.6M) can match TinyBERT (4.7M) and CoFi (4.7M) on most tasks, despite having fewer parameters. In addition, compared to MobileBERT and AutoTinyBERT, which require general pre-training and use different teachers than GRAIN's, although not directly comparable, GRAIN shows promising results with less computation.

In Table 1, we show the results of GRAIN without embedding factorization (**GRAIN w/o EF**). One can see that the pruned models do not always benefit from having large embeddings. On SQuAD, the factorized embedding leads to improved performance, while on SST-2, a large embedding matrix is better. However, the gaps at model density 5% are closer than those at model density 3%, indicating that embedding factorization has more minor impacts on larger pruned models.

We also measure the latency of GRAIN and find that GRAIN achieves competitive speedups when compared with other methods. Please refer to Appendix D for more details.

To summarize the above, GRAIN is efficient and effective for compressing pre-trained language models on a wide range of downstream tasks.

5.3 Ablation Study

We apply ablations on GRAIN w/o EF to study the effect of each component, as listed in Table 2.

Firstly, The impact of removing StructReg varies depending on the task, with performance either increasing or decreasing. We defer the detailed discussion on StructReg to Section 5.4.

Secondly, we remove gradient separation (GradSep), so the importance scores are influenced by gradients from both $\mathcal{L}_{\text{Hidden}}$ and $\mathcal{L}_{\text{CE}}$. The performance on different tasks drops more or less, and SQuAD is most notably affected. The results indicate that the gradients from the hidden layer loss $\mathcal{L}_{\text{Hidden}}$ have an impact on the pruning process, and it would be more beneficial to exclude it from the estimation of importance scores.

Thirdly, we remove the hidden layer loss $\mathcal{L}_{\text{Hidden}}$, so knowledge distillation only optimizes the cross-entropy objective $\mathcal{L}_{\text{CE}}$. The performance drops significantly, showing the necessity to use both objectives for obtaining effective pruned models.

Lastly, we investigate if gradient-based pruning is necessary and effective. To ablate gradient-based pruning, we generate random scores instead of gradient-based scores at each pruning step and keep all other settings unchanged, so the models are randomly pruned. The results are displayed in the last line in Table 2. The random structures resulted in inferior results, proving the superiority of the structures found by gradient-based pruning. Thus both pruning and distillation are crucial components.

5.4 Analysis

We first compare the effects of different pruning units. Then we look into the structures of pruned models to better understand our method.

Attention Heads Pruning Intra-attention pruning allows larger structure search space and more flexible models, but is intra-attention pruning more

[4] Please refer to Table 7 in Appendix E for detailed results of GRAIN at higher model densities.

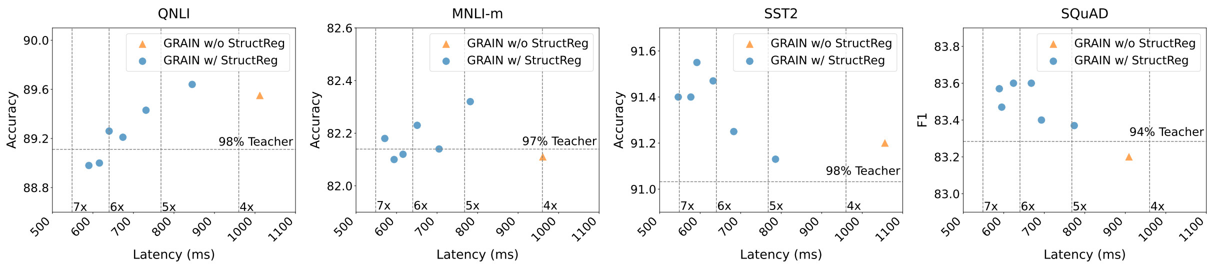

Figure 5: Performance and latencies under different structure regularization strengths at density 5%. The orange marker denotes $\alpha = 0$, and the blue markers denote $\alpha = 0.05, 0.1, 0.15, 0.2, 0.25, 0.3$ from right to left. The vertical lines indicate the speedups relative to BERT$_{\text{base}}$ (teacher). The horizontal line indicates the teacher's performance.

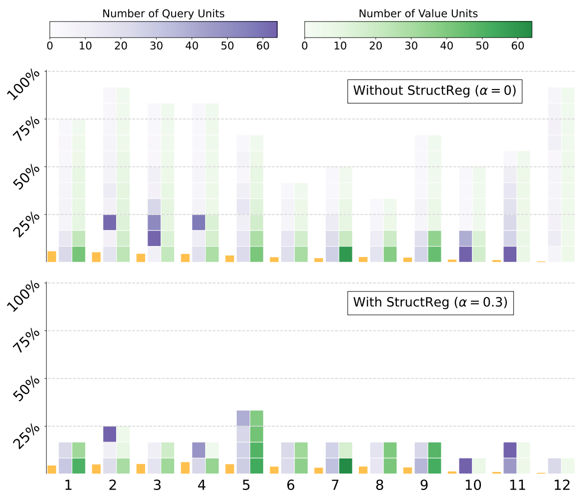

Figure 6: The structures of the pruned models at 5% density on QNLI. Orange bars denote the remaining units in FFN layers. Each purple (green) cell denotes a remaining pair of query/key (value/output) matrices in a head. The darker the cell, the more units remain.

effective compared to attention heads pruning in practice? To answer the question, we conduct comparative attention heads pruning experiments.

We follow the GRAIN procedure, except for setting the pruning units to be attention heads and FFN hidden dimensions. The structure regularization strength is set to 0, and the target model density is set to 5%. Since each attention head has more parameters than each FFN hidden dimension, the importance scores of attention heads and FFN hidden dimensions are not directly comparable, so attention heads and FFN hidden dimensions can not be globally sorted and pruned.[5] Hence, we sort and prune the two kinds of units independently and we have the freedom to set their densities as long as the model density is fixed to 5%.

We experiment with five groups of (FFN, Heads) density,[6] and the results are shown in Table 3. **Intra+FFN** denotes pruning with intra-attention units. **Heads+FFN** denotes pruning with attention heads. Heads+FFN reaches its best performance when its (FFN, Heads) density is close to the (FFN, Heads) density of Intra+FFN, but Intra+FFN still outperforms Heads+FFN at different (FFN, Heads) densities. The results imply that intra-attention pruning is more effective than attention heads pruning.

Model Structures As we stated previously, intra-attention pruning tends to yield fragmented structures, which hinder running efficiency. We apply structure regularization (StructReg) to encourage generating models with less fragmented units. To get an intuitive understanding, Figure 6 shows the structures of the models pruned with and without StructReg at model density 5% on QNLI.[7] We first notice that with intra-attention pruning, attention heads take more diverse structures since the number of query and value units can differ. The model pruned without StructReg holds 95 attention heads, where most heads contain only a few query or value units. The average query and value units per head are 9.8 and 8.2, respectively. With StructReg, the model holds only 25 attention heads, and the average numbers of query and value units per head are 28.6 and 28.5. The number of heads is significantly reduced. We also find FFN layers are more severely pruned than attention heads, consistent with results in Xia et al. (2022).

Speed and Performance We next study the impacts of StructReg on speed and performance. We evaluate the latency with batch size 128 and sequence length 512 on an NVIDIA M40 GPU for all tasks. The results are shown in Figure 5. The

[5]A possible solution is to manually rescale the scores of different units. However, this also introduces additional hyperparameters.

[6]FFN (heads) density is defined as the percentage of the remained parameters in all FFNs (heads).

[7]Structures of models on different tasks are listed in Appendix C.

latency of BERT$_{\text{base}}$ is around 3840ms, far beyond the plots' range. The pruned models without StructReg only achieve about $4\times$ speedup. As the regularization strength α increases from 0 to 0.3, the latency decreases monotonically. At $\alpha = 0.3$ (the leftmost marker in each plot), models achieve $6 \sim 7\times$ speedups, notably faster than the unregularized ones. The task performance is also affected by StructReg. As α increases from 0 to 0.3, the QNLI accuracy drops by 0.6%, while SQuAD F1 increases by 0.4%. There is no uniform trend in performance across different tasks. Nevertheless, compared to the gains in speedups, the variances in performance are marginal.

6 Conclusion

This paper proposes GRAIN, a gradient-based structured pruning method that expands the structure search space by pruning with intra-attention structures. We provide a structure regularization strategy that encourages finding regular structures and helps achieve lower latencies. We also combine pruning with distillation. We propose to separate the gradients from different losses to reduce the interference. GRAIN is computationally efficient since it does not require pre-training or data augmentation. Experiments show that GRAIN achieves impressive high performance and outperforms other methods at different model densities on various natural language understanding tasks and meanwhile maintains competitive speedups.

Limitations

Inference Speed At the same model size, the latencies of GRAIN on different tasks are relatively large compared to the methods like CoFi and TinyBERT. This is because GRAIN generates models with different head size, and the computation of these heads are not parallelized. Thus the resulting models are slower than the models with uniform attention structures. This problem could be relieved by introducing model structure regularization at a higher level or by some engineering techniques, such as merging heads with the same or similar size into a large matrix to increase parallelism.

Backbone Models GRAIN is designed for transformer-based models. Although the transformer is one of the most popular building blocks of NLP models, there are many other promising structures. The effectiveness of GRAIN on model compression is possibly correlated with hardware lottery or software lottery (Hooker, 2020). In addition, we have only tested our method with the standard multi-head attention mechanism. Transplanting GRAIN to other attention mechanisms is possible, but the effectiveness has yet to be tested.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (Grant No. 2022YFC3303504).

References

Cheng Chen, Yichun Yin, Lifeng Shang, Zhi Wang, Xin Jiang, Xiao Chen, and Qun Liu. 2021. Extract then distill: Efficient and effective task-agnostic bert distillation.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-training with whole word masking for chinese bert.

Yiming Cui, Ting Liu, Wanxiang Che, Li Xiao, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2019. A span-extraction dataset for Chinese machine reading comprehension. 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 5886–5891, Hong Kong, China. 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.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155, Online. Association for Computational Linguistics.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Sara Hooker. 2020. The hardware lottery.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic BERT with adaptive width and depth. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence Moss. 2020. OCNLI: Original Chinese Natural Language Inference. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3512–3526, Online. Association for Computational Linguistics.

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.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The optimal BERT surgeon: Scalable and accurate second-order pruning for large language models. *CoRR*, abs/2203.07259.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yanyang Li, Fuli Luo, Runxin Xu, Songfang Huang, Fei Huang, and Liwei Wang. 2022. Probing structured pruning on multilingual pre-trained models: Settings, algorithms, and efficiency. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1852–1865, Dublin, Ireland. Association for Computational Linguistics.

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6524–6538, Online. Association for Computational Linguistics.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. 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.

Zejian Liu, Fanrong Li, Gang Li, and Jian Cheng. 2021. EBERT: Efficient BERT inference with dynamic structured pruning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4814–4823, Online. Association for Computational Linguistics.

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

Christos Louizos, Max Welling, and Diederik P. Kingma. 2017. Learning sparse neural networks through l_0 regularization. *CoRR*, abs/1712.01312.

J. S. McCarley. 2019. Pruning a bert-based question answering model. *CoRR*, abs/1910.06360.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14014–14024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

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

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Chih Chieh Shao, Trois Liu, Yuting Lai, Yiying Tseng, and Sam Tsai. 2019. Drcd: a chinese machine reading comprehension dataset.

Bowen Shen, Zheng Lin, Yuanxin Liu, Zhengxiao Liu, Lei Wang, and Weiping Wang. 2022. Cost-eff: Collaborative optimization of spatial and temporal efficiency with slenderized multi-exit language models.

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

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, ACL 2020, Online, July 5-10, 2020*, pages 2158–2170. Association for Computational Linguistics.

Marzieh Tahaei, Ella Charlaix, Vahid Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. 2022. KroneckerBERT: Significant compression of pre-trained language models through kronecker decomposition and knowledge distillation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2116–2127, Seattle, United States. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355. Association for Computational Linguistics.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528, Dublin, Ireland. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yichun Yin, Cheng Chen, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2021. AutoTinyBERT: Automatic hyper-parameter optimization for efficient pre-trained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5146–5157, Online. Association for Computational Linguistics.

Michael Zhu and Suyog Gupta. 2018. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net.

A Reproducibility and Training Costs

Hyperparameters We summarize the hyperparameters of our experiments in Table 4. We use AdamW optimizer (Loshchilov and Hutter, 2019). The learning rate is scheduled with 10% warm-up steps followed by a linear decay.

Training Environment All the training experiments are conducted on a single NVIDIA V100 GPU. The PyTorch (Paszke et al., 2019) version is 1.8.1, the CUDA version is 10.2, and Transformers (Wolf et al., 2020) version is 4.10.0.

Training Costs It takes about 15 hours to finish training on MNLI and QQP, 11 hours on SQuAD, 5 hours on QNLI, 3 hours on SST-2, and 1 hour on CoNLL 2003.

Hyperparameter	**Value**
peak learning rate	3e-5 (GLUE) 3e-5 (SQuAD) 1e-4 (CoNLL 2003)
number of epochs	20 (GLUE) 20 (SQuAD) 40 (CoNLL 2003)
batch size	32
temperature τ	8
start of pruning p_s	0.2
end of pruning p_e	0.4
smoothing factor β	0.998
regularization strength α	0.3
reduced embedding size r	192

Table 4: Hyperparameters used in the experiments.

B Dataset Statistics

The details of the datasets are shown in Table 5.

C Structures of Pruned Models

Table 6 summarizes the structures of the pruned models on different tasks at model density 5%.

D Inference Speed vs. Performance

Figure 7 shows the latency of GRAIN and other methods on various tasks. All the measurements are conducted under the same environment (see the paragraph **Speed and Performance** in Section 5.4). The structure regularization strength α is 0.3. GRAIN achieves competitive speedups comparable to other methods.

Task	**Train Size**	**Metric**	**# Labels**
English Task			
QNLI	105k	Acc	2
MNLI	393k	Acc	3
QQP	364k	Acc	2
SST-2	67k	Acc	2
SQuAD	88k	F1	N/A
CoNLL 2003	14k	F1	9
Chinese Task			
OCNLI	50k	Acc	3
TNEWS	53k	Acc	15
CMRC 2018	10k	F1	N/A
DRCD	27k	F1	N/A

Table 5: Details of the datasets.

E More Results

E.1 Pruning RoBERTa

We conduct GRAIN with RoBERTa-base (Liu et al., 2019) on the same set of tasks and use the same hyperparameters as those in Table 4. The results of GRAIN with BERT and RoBERTa at different model densities are shown in Table 7. The pruned RoBERTa outperforms pruned BERT at high densities, but at low densities, BERT surpasses RoBERTa on some tasks.

E.2 Experiments on Chinese Tasks

Due to the limited availability of results on model compression methods for Chinese tasks, we present the results of GRAIN on several Chinese tasks, providing a useful reference point for related works.

We evaluate GRAIN on the following Chinese tasks: OCNLI (Hu et al., 2020), an original Chinese natural language inference task; TNEWS (Xu et al., 2020), a short text classification task for news; CMRC 2018 (Cui et al., 2019) and DRCD (Shao et al., 2019), two representative span-extraction Chinese machine reading comprehension tasks. The details of the datasets are shown in Table 5.

The learning rate is 1e-4 for CMRC 2018 and DRCD, 2e-5 for OCNLI and TNEWS; the number of epochs is 40 for CMRC 2018 and DRCD, 20 for OCNLI and TNEWS. Other hyperparameters are the same as those in Table 4. The teacher model is Chinese-RoBERTa-wwm-ext (Cui et al., 2021).

We report the mean score of 3 runs for each task using different random seeds. The results are shown in Table 8.

Datasets	MHA Layers	Total Heads	Query Units / Head	Value Units / Head	FFN Size
QNLI ($\alpha = 0$)	12	95	9.8	8.2	87.9
QNLI ($\alpha = 0.3$)	12	25	28.6	28.5	106.1
MNLI ($\alpha = 0$)	12	86	9.0	8.6	103.9
MNLI ($\alpha = 0.3$)	11	21	28.8	32.9	122.5
QQP ($\alpha = 0$)	12	93	9.8	8.7	87.1
QQP ($\alpha = 0.3$)	12	26	27.5	26.4	113.5
SST-2 ($\alpha = 0$)	12	101	4.2	8.9	120.2
SST-2 ($\alpha = 0.3$)	11	19	20.5	37.7	138.2
SQuAD ($\alpha = 0$)	12	75	12.8	10.1	87.3
SQuAD ($\alpha = 0.3$)	12	23	33.0	30.8	108.0
CoNLL-03 ($\alpha = 0$)	12	91	6.1	9.1	114.5
CoNLL-03 ($\alpha = 0.3$)	9	22	21.4	31.9	132.6

Table 6: Structures of the pruned models on different tasks at model density 5%.

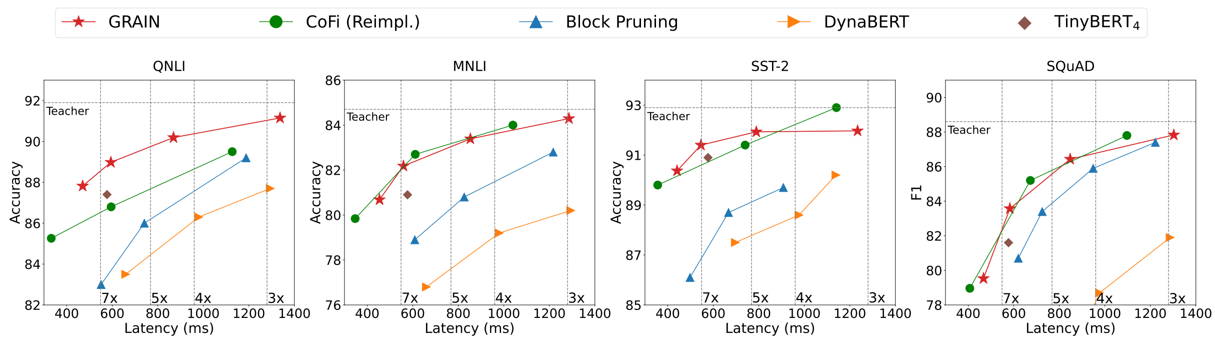

Figure 7: Performance vs. latency of GRAIN and other methods. The dashed vertical lines indicate the speedups relative to BERT$_{\text{base}}$.

Model	QNLI (Acc)	MNLI (m/mm Acc)	QQP (Acc)	SST-2 (Acc)	SQuAD (F1 / EM)	CoNLL-03 (F1)	Model Size	Total Size
BERT$_{\text{base}}$ (teacher)	91.9	84.7 / 85.0	91.2	92.9	88.6 / 81.1	91.2	85.1M	108.9M
RoBERTa$_{\text{base}}$ (teacher)	**93.0**	**87.7 / 87.5**	**91.7**	**94.7**	**91.5 / 84.9**	**92.1**	85.1M	124.0M
20% Model Density								
GRAIN	91.2	84.3 / 84.2	91.0	92.0	87.8 / 79.9	90.4	17M (20%)	23.4M
GRAIN-R	**91.9**	**86.8 / 86.6**	**91.6**	**93.1**	**89.4 / 81.6**	**91.2**	17M (20%)	27.2M
10% Model Density								
GRAIN	90.2	83.4 / 83.5	90.7	91.9	86.4 / **77.7**	89.7	8.5M (10%)	14.9M
GRAIN-R	**90.9**	**{85.0 / 85.0**	**91.0**	**92.2**	**86.5** / 77.6	**90.7**	8.5M (10%)	18.7M
5% Model Density								
GRAIN	89.0	82.2 / 82.5	**90.4**	91.4	**83.6 / 73.7**	88.3	4.3M (5.0%)	10.7M
GRAIN-R	**89.4**	**83.1 / 83.0**	90.3	**91.6**	82.4 / 71.9	**89.7**	4.3M (5.0%)	14.5M
3% Model Density								
GRAIN	87.8	80.7 / 81.1	90.0	90.4	79.5 / 68.4	86.8	2.6M (3.0%)	9.0M

Table 7: Results of GRAIN (pruning BERT) and GRAIN-R (pruning RoBERTa) with model density varying from 3% to 20%.

Model	**OCNLI** (Acc)	**TNEWS** (Acc)	**CMRC 2018** (F1/EM)	**DRCD** (F1/EM)	**Model Size**	**Total Size**
RoBERTa-wwm-ext (teacher)	77.1	57.8	87.3 / 67.7	94.5 / 89.1	85.1M	101.7M
20% Model Density						
GRAIN	75.4	56.9	87.3 / 67.7	93.8 / 88.5	17M (20%)	21.6M
10% Model Density						
GRAIN	73.3	56.2	85.8 / 65.3	92.6 / 86.7	8.5M (10%)	13.1M
5% Model Density						
GRAIN	70.2	55.6	83.5 / 61.1	90.6 / 83.4	4.3M (5.0%)	8.9M

Table 8: Results of GRAIN (pruning Chinese RoBERTa-wwm-ext) on the development sets of Chinese text classification and machine reading comprehension tasks.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
The section after Conclusion.

☒ A2. Did you discuss any potential risks of your work?
This work presents a general compression method, which is not tied to particular applications.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1 Introduction.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 5.1 and Appendix B

☑ B1. Did you cite the creators of artifacts you used?
Section 5.1 and Appendix B

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
The licenses for each artifact can be found in the original paper or the repository on GitHub.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Reader may refer to the original papers of the artifacts.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix B

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5.1 and Appendix A

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix A

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5.1

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.