Sparse Progressive Distillation: Resolving Overfitting under Pretrain-and-Finetune Paradigm

Anonymous ACL submission

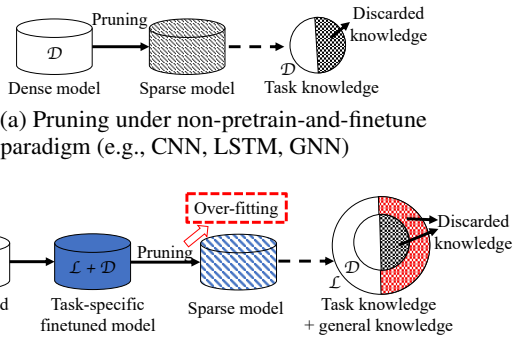
Abstract

Conventional wisdom in pruning Transformer-based language models is that pruning reduces the model expressiveness and thus is more likely to underfit rather than overfit. However, under the trending pretrain-and-finetune paradigm, we postulate a counter-traditional hypothesis, that is: pruning increases the risk of overfitting when performed at the fine-tuning phase. In this paper, we aim to address the overfitting problem and improve pruning performance via progressive knowledge distillation with error-bound properties. We show for the first time that reducing the risk of overfitting can help the effectiveness of pruning under the pretrain-and-finetune paradigm. Ablation studies and experiments on the GLUE benchmark show that our method outperforms the leading competitors across different tasks.

1 Introduction

Recently, the emergence of Transformer-based language models (using pretrain-and-finetune paradigm) such as BERT (Devlin et al., 2018) and GPT-3 (Brown et al., 2020) have revolutionized and established state-of-the-art (SOTA) records (beyond human-level) on various natural language (NLP) processing tasks. These models are first pre-trained in a self-supervised fashion on a large corpus and fine-tuned for specific downstream tasks (Wang et al., 2018). While effective and prevalent, they suffer from redundant computation due to the heavy model size, which hinders their popularity on resource-constrained devices, e.g., mobile phones, smart cameras, and autonomous driving (Li et al., 2021; Choi and Baek, 2020).

Various weight pruning approaches (zeroing out certain weights and then optimizing the rest) have been proposed to reduce the footprint requirements of Transformers (Zhu and Gupta, 2018; Blalock et al., 2020; Gordon et al., 2020; Xu et al., 2021; Li et al., 2021). Conventional wisdom in pruning states that pruning reduces the overfitting risk since the compressed model structures have fewer parameters and are believed to be less prone to overfit (Gerum et al., 2020). However, under the pretrain-and-finetune paradigm, most pruning methods understate the overfitting problem.



(a) Pruning under non-pretrain-and-finetune paradigm (e.g., CNN, LSTM, GNN)

(b) Pruning under pretrain-and-finetune paradigm

Figure 1: Pruning under non-pretrain-and-finetune vs. pruning under pretrain-and-finetune. In the subfigures, the cylinders on the left describe the pruning process, and the circles on the right represent the knowledge analysis of the sparse model.

In this paper, we postulate a counter-traditional hypothesis, that is: model pruning increases the risk of overfitting if pruning is performed at the fine-tuning phase. As shown in Figure 1b, the pretrain-and-finetune paradigm contains two types of knowledge, the general-purpose language knowledge learned during pre-training ($\mathcal{L}$) and the task-specific knowledge from the downstream task data ($\mathcal{D}$). Compared to conventional pruning that only discards task-specific knowledge (Figure 1a), pruning under pretrain-and-finetune (Figure 1b) discards extra knowledge (red area) learned in pre-training phase. Thus, to recover both the extra discarded general-purpose knowledge and the discarded task-specific knowledge, it increases the amount of information a model needs, which results in relative data deficiency, leading to a higher risk of overfitting. To empirically verify the overfitting problem, we visualize the training and evaluation performance on a real-world task data of

(a) Sparsity=0 (b) Sparsity=0.8 (c) Sparsity=0.95

Figure 2: Visualization of the overfitting problem when pruning weight matrices of $\text{BERT}_{\text{BASE}}$ on MRPC at the fine-tuning phase. The overfitting problem becomes more severe with the increasing of sparsity.

MRPC (Devlin et al., 2018) in Figure 2. From Figure 2 (b), it is observed that the evaluation accuracy on the training dataset remains improved while it keeps the same for the validation set through the training process. From Figure 2 (c), the difference in performance becomes more significant when the pruning rate becomes higher and the performance on the validation set even becomes worse after 2,000 training steps. All these observations verify our hypothesis.

The main question this paper attempts to answer is: how to reduce the risk of overfitting of pre-trained language models caused by pruning? However, answering this question is challenging. First, under the pretrain-and-finetune paradigm, both the general-purpose language knowledges and the task-specific knowledge are learned. It is non-trivial to keep the model parameters related to both knowledges when pruning. Second, the amount of data for downstream tasks can be small, such as the data with privacy. Thus, the overfitting problem can easily arise, especially in the face of high pruning rate requirements. A little recent progress has been made on addressing overfitting associated with model compression. However, their results are not remarkable and most of them focus on the vision domain (Bai et al., 2020; Shen et al., 2021).

To address these challenges, we propose SPD, a sparse progressive distillation method, for pruning pre-trained language models. We prune and optimize the weight duplicates of the backbone of the teacher model (a.k.a., student modules). Each student module shares the same architecture (e.g., the number of weights, the dimension of each weight) as the duplicate. We replace the corresponding layer(s) of the duplicated teacher model with the pruned sparse student module(s) in a progressive way and name the new model as a grafted model. We validate our proposed method through the ablation studies and the GLUE benchmark. Experimental results show that our method outperforms the leading competitors.

We summarize our contributions as follows:

- We postulate, analyze, and empirically verify a counter-traditional hypothesis: pruning increases the risk of overfitting under the pretrain-and-finetune paradigm.
- We propose a sparse progressive pruning method and show for the first time that reducing the risk of overfitting can help the effectiveness of pruning.
- Moreover, we theoretically prove that our pruning method can obtain a sub-network from the student model that has similar accuracy as the teacher, and the accuracy gap is bounded.
- Last but not least, we study and minimize the interference between different hyperparameter strategies, including pruning rate, learning rate, and grafting probability, to further improve performance.

2 Related Work

To summarize, our contribution is determining the overfitting problem of pruning under the pretrain-and-finetune paradigm and proposing the sparse progressive distillation method to address it. We demonstrate the benefits of the proposed framework through the ablation studies. We validate our method on six datasets from the GLUE benchmark. To test if our method is applicable across tasks, we include the tasks of both single sentence and sentence-pair classification. Experimental results show that our method outperforms the leading competitors by a large margin.

Network Pruning. Common wisdom has shown that weight parameters of deep learning models can

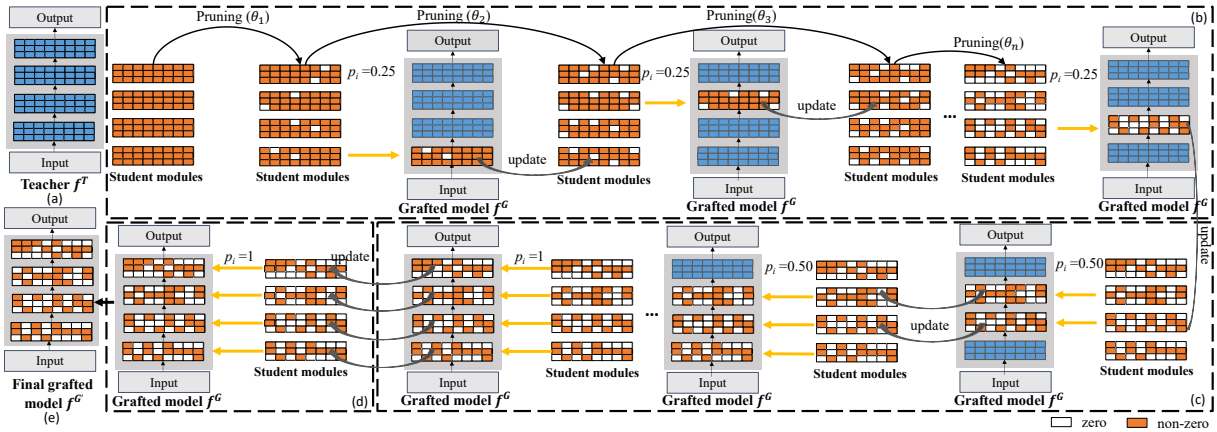


Figure 3: An overview of our sparse progressive distillation method. (a) Teacher model. (b) Pruning to target sparsity. (c) Module grafting with increasing probability. (d) Fine-tuning. (e) Final grafted model.

be reduced without sacrificing accuracy loss, such as magnitude-based pruning (Han et al., 2015) and ottery ticket hypothesis (Frankle and Carbin, 2018). (Zhu and Gupta, 2018) compared small-dense models and large-sparse models with the same parameters and showed that the latter outperforms the former, showing the large-sparse models have better expressive power than their small-dense counterparts. However, under the pretrain-and-finetune paradigm, pruning leads to overfitting as discussed.

Knowledge Distillation (KD). As a common method in reducing the number of parameters, the main idea of KD is that the small student model mimics the behaviour of the large teacher model and achieves a comparable performance (Hinton et al., 2015; Mirzadeh et al., 2020). (Sanh et al., 2019; Jiao et al., 2020; Sun et al., 2020) utilized KD to learn universal language representations from large corpus. However, current SOTA knowledge distillation methods are not able to achieve a high model compression rate (less than 10% remaining weights) while achieving an insignificant performance decrease.

Progressive Learning. The key idea of progressive learning is that student learns to update module by module with the teacher. (Shen et al., 2021) utilized a dual-stage distillation scheme where student modules are progressively grafted onto the teacher network, it targets the few-shot scenario and uses only a few unlabeled samples to achieve comparable results on CIFAR-10 and CIFAR-100. (Xu et al., 2020) gradually increased the probability of replacing each teacher module with their corresponding student module and trained the student to reproduce the behavior of the teacher. However, the performance on Transformer-based models of the aforementioned first method is unknown while the second method has an obvious performance drop with a low sparsity (50%).

3 Methodology

We propose to use a new knowledge distillation framework that utilizes error-bound provable pruning and progressive module grafting.

3.1 Problem Formulation

The teacher model and the grafted model (shown in Figure 3) are denoted as f^S and f^G, respectively. Both models have $N+1$ layers (i.e., the first N layers are encoder layers, and the $(N+1)$-th layer is the output layer). Denote $f_i^{\mathrm{T}}(\cdot)$, $f_i^{\mathrm{G}}(\cdot)$ as the behaviour function induced from the i-th encoder of the teacher model, and the grafted model, respectively. As shown in Figure 4, we utilize layer-wise knowledge distillation (KD), where we aim to bridge the gap between $f_i^{\mathrm{T}}(\cdot)$ and $f_i^{\mathrm{G}}(\cdot)$.

The grafted model is trained to mimic the behavior of the teacher model. During training, we minimize the summation loss $\mathcal{L}$:

$$\mathcal{L} = \sum_{x \in \mathcal{X}} \sum_{i=1}^{N+1} \lambda_i \mathcal{L}_{\mathrm{KD}}(f_i^T(x) f_i^G(x)), \quad (1)$$

where $\mathcal{X}$ denotes the training dataset, λ_i is coefficient of i-th layer loss, $\mathcal{L}_{\mathcal{D}}$ is the distillation loss of the layer pair, x_i is the input of the i-th layer.

During KD, each student module mimics the behavior of the corresponding teacher layer. Similar to (Jiao et al., 2020), we take the advantage of abundant knowledge in self-attention distribution, hidden states of each Transformer layer, and

the final output layer's soft logits of teacher model to help train the student model. Specifically, we design the KD loss as follows

$$\mathcal{L}_{\text{KD}} = \begin{cases} \mathcal{L}_{hidn} + \mathcal{L}_{attn} & 1 \leq i \leq N \\ \mathcal{L}_{pred} & i = N+1 \end{cases} \quad (2)$$

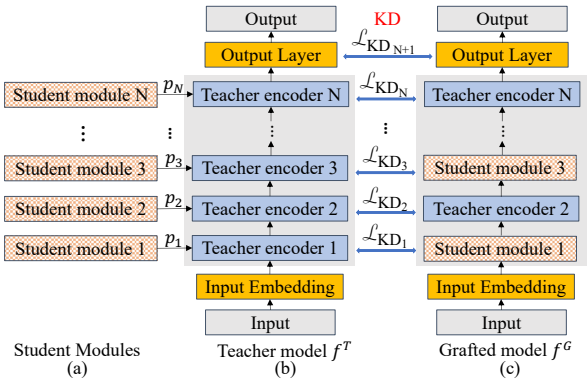


Figure 4: An overview of the layer-wise KD in SPD. (a) N sparse student modules have probabilities of p_1, p_2, p_3, ..., p_N to substitute the corresponding teacher layers separately. (b) Teacher model. (c) Grafted model. L_{KD_i} denotes the distillation loss between the i-th layer of the teacher and i-th layer of the grafted model.

where $\mathcal{L}_{hidn}$ = MSE(H_i^T, H_i^S) ($1 \leq i \leq N$) indicates the difference between hidden states, $\mathcal{L}_{attn}$ = MSE(A_i^T, A_i^S) indicates the difference between attention matrices. MSE($\cdot$) is the mean square error loss function and i is the index of Transformer layer. $\mathcal{L}_{pred}$ = -softmax(z^T) $\cdot$ log_softmax(z^S / $temp$) indicates the difference of soft cross-entropy loss, where z^T and z^S are the soft logits of teacher and student model, respectively. T is the temperature hyper-parameter.

We further reduce the number of non-zero parameters in the weight matrix while maintaining accuracy. We denote $\{W_j\}_{j=1}^{j=i}$ as the collection of weights in the first i layers, θ_j as the sparsity of the j-th layer. Then, the loss function of sparse knowledge distillation becomes

$$\mathcal{L} = \sum_{x \in \mathcal{X}} \sum_{i=1}^{N+1} \lambda_i \mathcal{L}_{\text{KD}}(f_i^T(x, \{\mathbf{W}_j\}_{j=1}^{j=i}), f_i^G(x, \{\mathbf{W}_j\}_{j=1}^{j=i}))$$
$$s.t.\ \ sparsity(\mathbf{W}_j) \leq \theta_j \text{ for } j = 1, ..., N \quad (3)$$

After training, we find the sparse weight matrix W_j^* using

$$\mathbf{W}_j^* = \Pi_{S_j}(\mathbf{W}_j) \text{ for } j = 1, ..., N, \quad (4)$$

where $\Pi_{S_j}(\cdot)$ denotes the Euclidean projection onto the set $S_j = \{\mathbf{W}_j \mid sparsity(\mathbf{W}_j) \leq \theta_j\}$.

3.2 The Proposed

3.2.1 Error-bound Provable Pruning

Our pruning method is similar to finding matching subnetworks using the lottery ticket hypothesis methodology.

Analysis on Feed-forward linear Network. Consider a linear network $f(x) = \mathbf{w} \cdot x$, and $g(x) = (\sum_{i=1}^{n} \mathbf{w}_i)\, x$. Lueker et al.(Lueker, 1998) shows that existing a subset of $\mathbf{w}_i$, such that the corresponding value of $g(x)$ is very close to $f(x)$.

Corollary: When $\mathbf{w}_1^*, ..., \mathbf{w}_n^*$ belongs to i.i.d. uniform distribution over [-1,1], where $n \geq C \log \frac{2}{\delta}$, $\delta \leq \min\{1, \epsilon\}$. Then, with probability at least 1-δ, we have

$$\exists G_{\textbf{spd}} \subset \{1, 2, ..., n\}, \forall\, \mathbf{W} \in [-0.5, 0.5],$$
$$s.t \left| \mathbf{w} - \sum_{i \in G_{\textbf{spd}}} \mathbf{w}_i^* \right| \leq \epsilon \quad (5)$$

Analysis on self-attention layer. The self-attention layer can be present as:

$$\mathbf{Z} = \text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}}) \cdot \mathbf{V}. \quad (6)$$

Consider a model $f(x)$ with only one self-attention layer, when the token size of input x is 1, softmax($\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}}$) = 1, we have $\mathbf{Z}$ = $\mathbf{V}$, where $\mathbf{V} = \mathbf{W}^{\mathbf{V}} x$.

Consider $f^G(x) = \left(\sum_{i=1}^{d} \mathbf{w}_i^G\right) x$ and a pruning sparsity θ, base on ***Corollary***, when $d \geq C \log 4/\epsilon$, there exists a pattern of $\mathbf{w}_i^G$, such that, with probability $1 - \epsilon$,

$$\forall\, \mathbf{w} \in [-1, 1], \exists \theta_i \in \{0, 1\},$$
$$s.t. \left| \mathbf{w} - (\sum_{i \in [1,d]} \mathbf{w}_i^G I(\theta_i)) \right| < \epsilon \quad (7)$$

In general, let the token size x be n. so $\mathbf{x} = (x_1, x_2, ..., x_n)$. Consider a teacher model $f^T(x)$ with a self-attention layer, then

$$f^T(x_i) = \text{softmax}(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{(d_k)}}) \cdot \mathbf{V}_i = (\frac{\sum_j e^{c_{ij}}}{\sum_i \sum_j (e^{c_{ij}})}) \cdot \mathbf{V}_i$$
$$= (\frac{\sum_j e^{c_{ij}}}{\sum_i \sum_j (e^{c_{ij}})}) \mathbf{W}^{\mathbf{V}_i} x_i$$
$$= \mathbf{W}^{c_{i\cdot}} x_i \quad (8)$$

Base on ***Corollary***, when $d \geq C \log 4/\epsilon$, there exists a pattern of $\mathbf{W}_i^G$, such that, with probability $1 - \epsilon$,

$$\forall \mathbf{W}^{c_{i\cdot}} \in [-1,1], \exists \theta_k \in \{0,1\},$$
$$s.t. \left| \mathbf{W}^{c_{i\cdot}} - \left(\sum_{k \in [1,d]} \mathbf{W}_k^G I(\theta_k) \right) < \epsilon \right. \quad (9)$$

In summary:

$$\forall i \in \{1,2,...,n\}, \left| f^T(x_i) - f^G(x_i) \right| < \epsilon \quad (10)$$

Progressive Module Grafting. To avoid overfitting in the training process for the sparse Transformer model, we further graft student modules (scion) onto the teacher model duplicates (rootstock). For the i-th student module, we use an independent Bernoulli random variable r_i to indicate where it will be grafted on the rootstock. To be more specific, r_i has a probability of p (grafting probability) to be set as 1 (i.e., student module substitutes the corresponding teacher layer). Otherwise, the latter will keep unchanged. Once the target pruning rate is achieved, we apply linear increasing probability to graft student modules which enable the student modules to orchestrate with each other.

Different from the model compression methods that update all model parameters at once, such as TinyBERT (Jiao et al., 2020) and DistilBERT (Sanh et al., 2019), SPD only updates the student modules on the grafted model. It reduces the complexity of network optimization, which mitigates the overfitting problem and enables the student modules to learn deeper knowledge from the teacher model. The overview is described in Algorithm 1. We will further demonstrate the effectiveness of progressive student module grafting in 4.2.

Algorithm 1 Sparse Progressive Distillation

Input: Teacher model f^T (fine-tuned BERT$_{\text{BASE}}$); grafted model f^G: duplicates of teacher model.
Set t_1, t_2, t_3 as the final number of training steps of pruning, progressive module grafting, and finetuning, respectively.
Set p as the grafting probability
Output: Student model
$p \leftarrow p_0$
for t = 0 to t_3 **do**
  if $0 \leq$ t $< t_1$ **then**
    Prune student modules and generate mask M
    Graft student modules with p_0
  end if
  if $t_1 \leq$ t $< t_2$ **then**
    Graft student modules with $p \leftarrow k(t - t_1) + p_0$
  end if
  Calculate distillation loss $\mathcal{L}$ in Eqn. (3)
  For f^G, update sparse weights $w\prime \leftarrow w \cdot M$
  Duplicate sparse weight(s) on f^G to corresponding student module(s)
end for
return f^G

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate SPD on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) and report the metrics i.e., accuracy scores for SST-2, QNLI, RTE, and WNLI, Matthews Correlation Coefficient (MCC) for CoLA, F1 scores for QQP and MRPC, Spearman correlations for STS-B.

Baselines. We first use 50% sparsity (a widely adopted sparsity ratio among SOTA), and compare SPD against two types of baselines – non-progressive and progressive. For the former, we select BERT-PKD (Sun et al., 2019), DistilBERT (Sanh et al., 2019), MiniLM (Wang et al., 2020), TinyBERT (Jiao et al., 2020), SparseBERT (Xu et al., 2021) and E.T. (Chen et al., 2021), while for the latter, we choose Theseus (Xu et al., 2020). We further compare SPD against other existing works under higher sparsity, e.g., TinyBERT (Jiao et al., 2020), SparseBERT (Xu et al., 2021) and RPP (Guo et al., 2019).

SPD Settings. We use official BERT$_{\text{BASE}}$, uncased model as the pre-train model and the fine-tuned pre-train model as our teacher. Both BERT$_{\text{BASE}}$ and teacher model have the same architecture (i.e., 12 encoder layers (L = 12; embedding dimension d_{model} = 768; self-attention heads H = 12)). We finetune BERT$_{\text{BASE}}$ using 32 as batch size, 128 as max sequence length, best performance from $\{2e^-5, 3e^-5, 4e^-5, 5e^-5\}$ as the learning rate. For SPD model training, the number of pruning epochs, linear increasing module grafting epochs, finetuning epochs vary from [10, 30], [5, 20], [5, 10], respectively. For pruning, we use AdamW (Loshchilov and Hutter, 2018) as the optimizer and run the experiments with an initial grafting probability from {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}. The probability with the best performance will be adopted. After pruning, we adjust the slope of the grafting probability curve so that the grafting probability equals 1 at the end of module grafting. For module grafting and finetuning, an AdamW optimizer is used with learning rate chosen from $\{3e^-5,\ 1e^-4,\ 3.2e^-4,\ 5e^-4,\ 6.4e^-4\}$. The model training and evaluation are performed with Python 3.6.8, torch 1.8.0, and CUDA 11.1 on Quadro RTX6000 GPU226and Intel(R) Xeon(R) Gold 6244 @ 3.60GHz CPU.

Model	#Param	MNLI (393k) Acc	QQP (364k) F1	QNLI (105k) Acc	SST-2 (67k) Acc	CoLA (8.5k) Mcc	STS-B (5.7k) Spea	MRPC (3.7k) F1	RTE (2.5k) Acc	#Avg.
BERT$_{\text{BASE}}$ (Devlin et al., 2018)	109M	84.6	91.2	90.5	93.5	52.1	85.8	88.9	66.4	81.6
BERT$_{\text{BASE}}$ (ours)	109M	83.9	91.4	91.1	92.7	53.4	85.8	89.8	66.4	81.8
Fine-tuned BERT$_{\text{BASE}}$ (teacher)	109M	84.0	91.4	91.6	92.9	57.9	89.1	90.2	72.2	83.7
non-progressive										
BERT$_6$-PKD (Sun et al., 2019)	67M	81.5	88.9	88.4	91.0	45.5	86.2	85.7	66.5	79.2
DistilBERT (Sanh et al., 2019)	67M	82.2	88.5	89.2	92.7	51.3	86.9	87.5	59.9	79.8
MiniLM$_6$ (Wang et al., 2020)	67M	84.0	91.0	91.0	92.0	49.2	-	88.4	71.5	-
TinyBERT$_6$ (Jiao et al., 2020)	67M	84.5	91.1	91.1	93.0	54.0	90.1	90.6	**73.4**	83.5
SparseBERT (Xu et al., 2021)	67M	84.2	91.1	91.5	92.1	57.1	89.4	89.5	70.0	83.1
E.T. (Chen et al., 2021)	67M	83.7	86.5	88.9	90.8	55.6	87.6	88.7	69.5	81.4
progressive										
Theseus (Xu et al., 2020)	67M	82.3	89.6	89.5	91.5	51.1	88.7	89.0	68.2	81.2
SPD (ours)	67M	**85.0**	**91.4**	**92.0**	**93.0**	**61.4**	**90.1**	**90.7**	72.2	**84.5**

Table 1: Results on the dev set of the GLUE benchmark. The results of DistilBERT and TinyBERT$_6$ are taken from (Jiao et al., 2020). Mcc refers to Matthews correlation, and Spea refers to Spearman.

Model	Sparsity	CoLA (Mcc)	STS-B (Spea)	MRPC (F1)	RTE (Acc)	Avg.
Teacher	100%	57.9	89.1	90.2	72.2	77.4
TinyBERT$_4$	82%	29.8	-	82.4	-	-
RPP	88.4%	-	-	81.9	67.5	-
SparseBERT	95%	18.1	32.2	81.5	47.3	44.8
SPD (**ours**)	66.6%	50.7	88.9	90.4	69.7	74.9
SPD (**ours**)	75%	50.0	88.3	90.2	67.9	74.1
SPD (**ours**)	87.5%	49.9	87.8	89.9	67.9	73.9
SPD (**ours**)	90%	48.7	87.8	89.9	69.0	73.9
SPD (**ours**)	95%	42.1	86.9	88.7	56.7	68.2

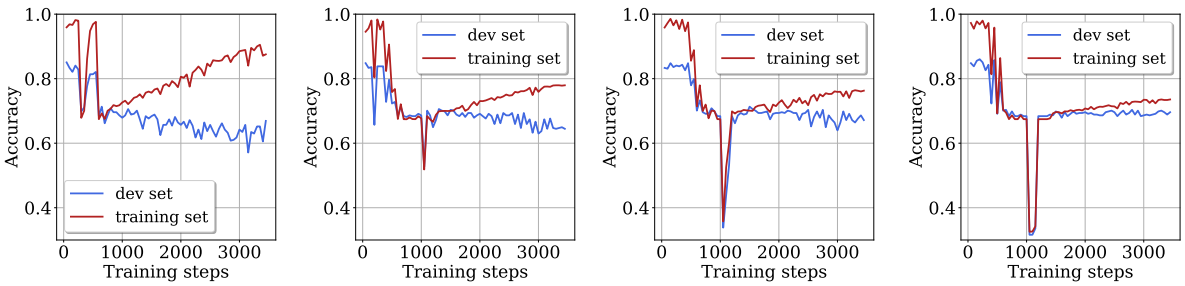
Table 2: Results on the dev set of the GLUE benchmark at higher pruning rates.

4.2 Experimental Results

Accuracy vs. Sparsity. We do experiments on eight GLUE benchmark tasks (Table 1). For non-progressive baselines, SPD exceeds all of them on QNLI, SST-2, CoLA, STS-B, and MRPC. For RTE, TinyBERT$_6$ has a 1.6% higher accuracy than SPD. However, TinyBERT$_6$ used augmented data while SPD does not use data augmentation to generate the results in Table 1. On average, SPD has 6.3%, 5.6%, 1.2%, 1.7%, 3.7% improvement in performance than BERT$_6$-PKD, DistilBERT, TinyBERT$_6$, SparseBERT, E.T. respectively. Furthermore, on CoLA, SPA achieves up to 25.9% higher performance compared to all non-progressive baselines. For the progressive baseline, we compare SPD with BERT-of-Theseus. Experimental results show that SPD exceeds the latter on all tasks. SPD has a 3.9% increase on average. Among all the tasks, CoLA and RTE have 20.2% and 5.9% gain respectively. For the comparison with sparse and non-progressive baseline, SPD has an improvement of 16.8%, 5.5%, 3.2%, 2.7%, 2.0%, 1.9%, 1.6%, 1.6% on CoLA, RTE, MNLI, QNLI, QQP, MRPC, STS-B, SST-2, respectively.

On all listed tasks, SPD even outperforms the teacher model except for RTE. On RTE, SPD retains exactly the full accuracy of the teacher model. On average, the proposed SPD achieves a 1.1% higher accuracy/score than the teacher model. We conclude the reason for the outstanding performance from three respects: 1) There is redundancy in the original dense BERT model. Thus, pruning the model with a low pruning rate (e.g., 50%) will not lead to a significant performance drop. 2) SPD decreases the overfitting risk which helps the student model learn better. 3) The interference between different hyperparameter strategies is mitigated, which enables SPD to obtain a better student model.

We also compare SPD with other baselines (i.e., 4-layer TinyBERT (Jiao et al., 2020), RPP (Guo et al., 2019), and SparseBERT (Xu et al., 2021)) under higher pruning rates. Results are summarized in Table 2. For the fairness of comparison, we remove data augmentation from the above methods. We mainly compare the aforementioned baselines with very high sparsity (e.g., 90%, 95%) SPD. For the comparison with TinyBERT$_4$, both SPD (90% sparsity) and SPD (95% sparsity) win. SPD (90% sparsity) has 63.4% and 9% higher evaluation score than TinyBERT$_4$ on CoLA and MRPC, respectively. For the setting of 95% sparsity, SPD outperforms TinyBERT$_4$ with 41.3% and



(a) No progressive, no KD (b) Progressive, no KD (c) No progressive, KD (d) Progressive, KD (ours)

Figure 5: Comparison of four strategies to deal with the overfitting problem on MRPC.

7.6% higher performance, respectively. Compared to RPP, both SPD (90% sparsity) and SPD (95% sparsity) show higher performance on MRPC, with 9.8% and 8.3% higher F1 score, respectively. For SparseBERT, SPD exceeds it on all tasks in Table 2. Especially on CoLA, SPD (90% sparsity) and SPD (95% sparsity) have 2.69× and 2.33× higher Mcc score on CoLA, respectively. SparseBERT has competitive performance with SOTA when using data augmentation. The reason for the performance drop for SparseBERT may because its deficiency of ability in mitigating overfitting problems.

Overfitting Mitigation. We explore the effectiveness of SPD to mitigate the overfitting problem. Depending on whether progressive, grafting, or KD is used, we compare 4 strategies: (a) no progressive, no KD; (b) progressive, no KD; (c) no progressive, KD; (d) progressive, KD (ours). We evaluate these strategies on both training and validation sets of MRPC. The results are summarized in Figure 5. From (a) to (d), the gap between the evaluation results of the training set and the dev set is reduced, which strongly suggests that the strategy adopted by SPD, i.e., progressive + KD, outperforms other strategies in mitigating the overfitting problem. Figure 5 (a), (b), and (c) indicate that compared to progressive only, KD has a bigger impact on mitigating overfitting, as the performance gap between the training set and the dev set decreases more from (a) to (c) than from (a) to (b). From Figure 5 (a), (b) and (c), we also observe that compared to no progressive, no KD, either using progressive (Figure 5 (b)) or KD (Figure 5 (c)) is very obvious to help mitigate the overfitting problem. Figures 5 (b), (c) and (d) indicate that the combination of progressive and KD brings more benefits than only using progressive or KD as Figure 5 (d) has the smallest performance gap between the training set and the dev set. Combined with Table 1 and Table 2, Figure 5 shows that SPD mitigates overfitting and leads to higher performance.

4.3 Ablation Studies

In this section, we justify the three schedulers used in our method (i.e., grafting probability, pruning rate, and learning rate), and study the sensitivity of our method with respect to each of them.

Study on Components of SPD. The proposed SPD consists of three components (i.e., sparse, knowledge distillation, and progressive module grafting). We conduct experiments to study the importance of each component on GLUE benchmark tasks with the sparsity of 50% and results are shown in Table 3. Compared to sparse (vanilla pruning), both KD and progressive achieve gains on performance among all tasks. The combination of all components has the highest average score, which is 0.96% higher than vanilla pruning.

Effects of Grafting Probability Strategy. In our method, we set the grafting probability greater than 0 during pruning, to allow student modules to learn deeper knowledge from the teacher model. To verify the benefit of this design, we change the grafting probability to zero and compare it with our method. The result on RTE is shown in Figure 6. Pruning with grafting (the red curve) shows better performance than pruning without grafting, which justifies the existence of grafting during pruning. In addition, we study the sensitivity of our method to grafting probability (Figure 7). It is observed that $p_0 = 0.6$ achieves the best performance, and the progressive design is better than the non-progressive.

Effects of Pruning Rate Strategy. For the pruning rate scheduler, we compare the strategies with different pruning ending steps. The results are shown in Figure 8. It is observed that the pruning during when grafting probability $p = p_0$ has a higher F1 score than other strategies on MRPC.

Effects of Optimizer Strategy. We also compare

Model	#Param	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg.
Fine-tuned BERT$_{\text{BASE}}$ (teacher)	109M	84.0	91.4	91.6	92.9	57.9	89.1	90.2	72.2	83.7
Sparse	67M	83.7	86.5	88.9	90.8	55.6	87.6	88.7	69.5	81.4
Sparse + KD	67M	84.2	91.1	91.5	92.1	57.1	89.4	89.5	70.0	83.1
Sparse + Progressive	67M	83.9	91.2	91.5	92.3	57.4	89.6	89.6	71.4	83.4
SPD (ours)	67M	**85.0**	**91.4**	**92.0**	**93.0**	**61.4**	**90.1**	**90.7**	**72.2**	**84.5**

Table 3: The performance comparison of different strategies on the dev set of GLUE. The results of DistilBERT and TinyBERT$_6$ are taken from (Jiao et al., 2020). Mcc refers to Matthews correlation and Spea refers to Spearman.

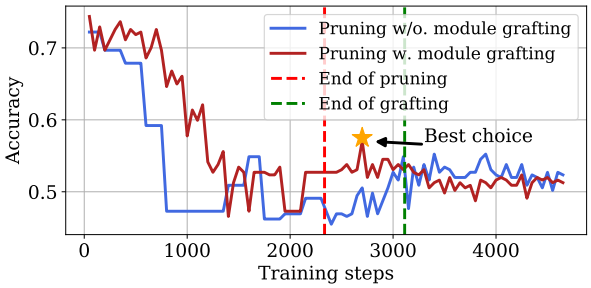


Figure 6: Pruning w/ module grafting vs. Pruning w/o. module grafting on RTE (dev set).

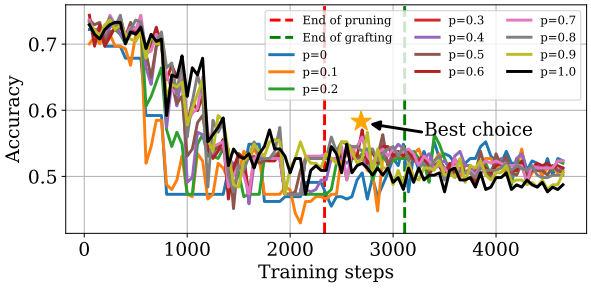


Figure 7: Sensitivity analysis of grafting probability on RTE (dev set).

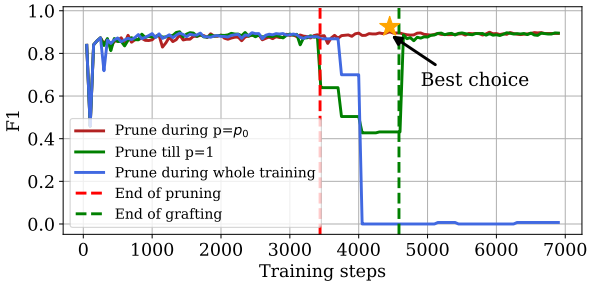


Figure 8: Effects of different pruning ending strategies on MRPC (dev set).

our strategy with the strategy that only has one learning rate scheduler. The results (Figure 9) indicate that our strategy (i.e., two independent optimizers) is better. We also evaluate different learning rates with the pruning rate of 0.9 and the grafting probability of 0.8.

5 Conclusion

In this paper, we postulate a counter-traditional hypothesis that pruning increases the risk of over-

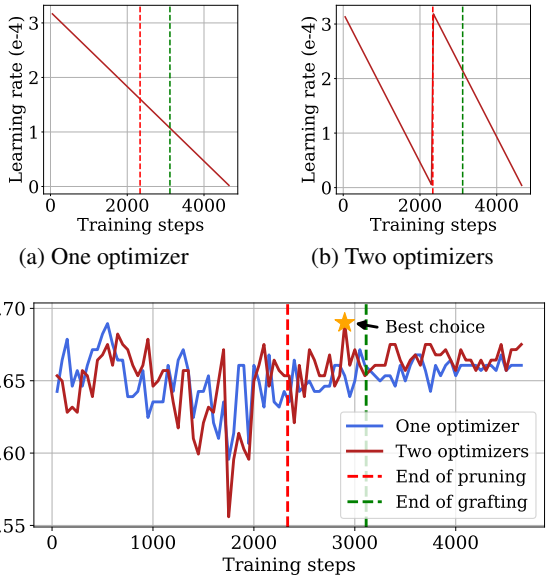


Figure 9: (a) The learning rate curve of one AdamW optimizer in training. (b) The learning rate of two AdamW optimizers in training. (c) Performance comparison of the above two settings.

fitting under the pretrain-and-finetune paradigm. We analyze and empirically verify this hypothesis, and propose a sparse progressive pruning method to address the overfitting problem. We theoretically prove that our pruning method can obtain a sub-network from the student model that has a similar accuracy as the teacher and the accuracy gap is bounded. We study and minimize the interference between different hyperparameter strategies, including pruning rate, learning rate, and grafting probability. A number of ablation studies and experimental results on eight tasks from the GLUE benchmark demonstrate the superiority of our method over the leading competitors.

References

Haoli Bai, Jiaxiang Wu, Irwin King, and Michael Lyu. 2020. Few shot network compression via cross distillation. In *AAAI*, volume 34, pages 3203–3210.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. 2020. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*.

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

Shiyang Chen, Shaoyi Huang, Santosh Pandey, Bingbing Li, Guang R Gao, Long Zheng, Caiwen Ding, and Hang Liu. 2021. Et: re-thinking self-attention for transformer models on gpus. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–18.

Yun Won Choi and Jang Woon Baek. 2020. Edge camera system using dee p learning method with model compression on embedded applications. In *2020 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4. IEEE.

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

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Richard C Gerum, André Erpenbeck, Patrick Krauss, and Achim Schilling. 2020. Sparsity through evolutionary pruning prevents neuronal networks from overfitting. *Neural Networks*, 128:305–312.

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

Fu-Ming Guo, Sijia Liu, Finlay S Mungall, Xue Lin, and Yanzhi Wang. 2019. Reweighted proximal pruning for large-scale language representation. *arXiv preprint arXiv:1909.12486*.

Song Han et al. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1135–1143.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Advances in Neural Information Processing Systems (NIPS)*.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4163–4174.

Zhengang Li, Geng Yuan, Wei Niu, Pu Zhao, Yanyu Li, Yuxuan Cai, Xuan Shen, Zheng Zhan, Zhenglun Kong, Qing Jin, et al. 2021. Npas: A compiler-aware framework of unified network pruning and architecture search for beyond real-time mobile acceleration. In *CVPR*, pages 14255–14266.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

George S Lueker. 1998. Exponentially small bounds on the expected optimum of the partition and subset sum problems. *Random Structures & Algorithms*, 12(1):51–62.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5191–5198.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Advances in Neural Information Processing Systems (NIPS)*.

Chengchao Shen, Xinchao Wang, Youtan Yin, Jie Song, Sihui Luo, and Mingli Song. 2021. Progressive network grafting for few-shot knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2541–2549.

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.

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.

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.

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. *Advances in Neural Information Processing Systems(NIPS)*.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. BERT-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7859–7869, Online. Association for Computational Linguistics.

Dongkuan Xu, Ian EH Yen, Jinxi Zhao, and Zhibin Xiao. 2021. Rethinking network pruning–under the pre-train and fine-tune paradigm.

Michael H Zhu and Suyog Gupta. 2018. To prune, or not to prune: Exploring the efficacy of pruning for model compression. *The International Conference on Learning Representations*.

Appendix

We provide the sensitivity analysis of learning rate on RTE and STS-B (dev set) and the evaluation curves of four tasks (CoLA, STS-B, MRPC, and RTE) with the target pruning rate of 0.95.

Sensitivity Analysis of Learning Rate. The analysis results on RTE and STS-B are shown in Figure 10 and Figure 11, respectively. Results vary with different learning rate settings. Among the eight learning rates listed in the legend of Figure 10, $3.2 \times e^{-4}$ achieves the best performance. For STS-B, $4.0 \times e^{-4}$ gives the best performance among the learning rate choices in Figures 11.

Evaluation Curves of Four Tasks at Target Pruning rate of 0.95. We plot the evaluation curves of CoLA (Figure 12), STS-B (Figure 13), MRPC (Figure 14), RTE (Figure 15) to further demonstrate the advantages of our proposed method SPD. In each figure, the x-axis is the training steps while the y-axis represents evaluation metrics. To obtain the curves, we use the same settings as Table 2.

Moreover, we describe the hyper-parameters settings in detail. For CoLA, we set the max sequence length as 128, the learning rate as $5.0e^{-}4$, the grafting probability during pruning as 0.8, the number of training epochs as 60, and the number of pruning epochs as 30. For STS-B, we use the same setting as CoLA. For MRPC, we set the max sequence length as 128, the learning rate as $6.4 \times e^{-4}$, the grafting probability during pruning as 0.8, the number of training epochs as 60, and the number of pruning epochs as 30. For RTE, we set the max sequence length as 128, the learning rate as $3.0 \times e^{-5}$, the grafting probability during pruning as 0.6, the number of training epochs as 60, and the number of pruning epochs as 30.

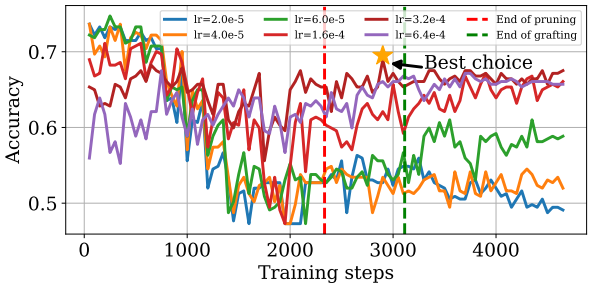


Figure 10: Sensitivity analysis of learning rate on RTE (dev set).

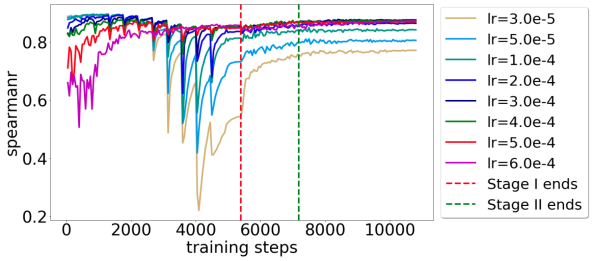


Figure 11: Sensitivity analysis of learning rate on STS-B (dev set).

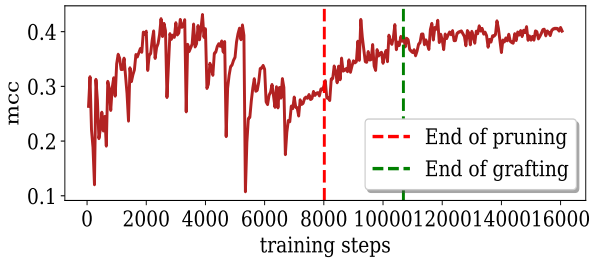


Figure 12: Evaluation on CoLA (dev set). Target pruning rate is 0.95.

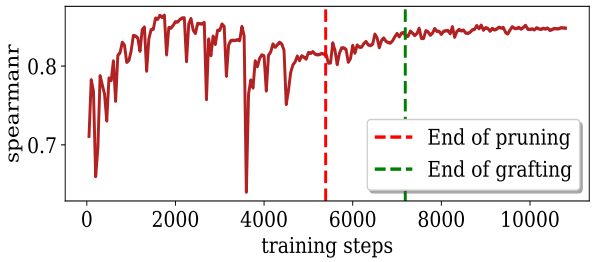


Figure 13: Evaluation on STS-B (dev set). Target pruning rate is 0.95.

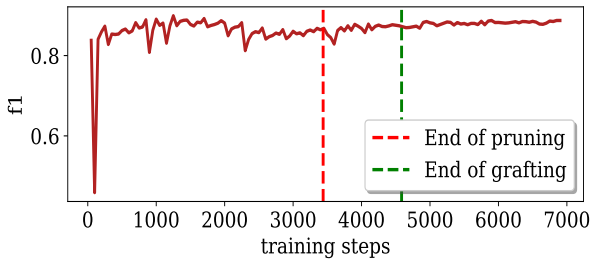


Figure 14: Evaluation on MRPC (dev set). Target pruning rate is 0.95.

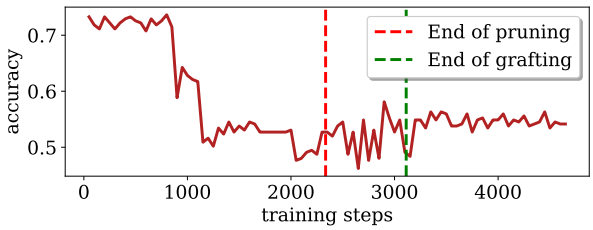


Figure 15: Evaluation on RTE (dev set). Target pruning rate is 0.95.