A Win-win Deal: Towards Sparse and Robust Pre-trained Language Models

Yuanxin Liu[1,2,3]*, Fandong Meng[5], Zheng Lin[1,4]†, Jiangnan Li[1,4], Peng Fu[1], Yanan Cao[1,4], Weiping Wang[1], Jie Zhou[5]

[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]MOE Key Laboratory of Computational Linguistics, Peking University
[3]School of Computer Science, Peking University
[4]School of Cyber Security, University of Chinese Academy of Sciences
[5]Pattern Recognition Center, WeChat AI, Tencent Inc, China
liuyuanxin@stu.pku.edu.cn, {fandongmeng,withtomzhou}@tencent.com
{linzheng,lijiangnan,fupeng,caoyanan,wangweiping}@iie.ac.cn

Abstract

Despite the remarkable success of pre-trained language models (PLMs), they still face two challenges: First, large-scale PLMs are inefficient in terms of memory footprint and computation. Second, on the downstream tasks, PLMs tend to rely on the dataset bias and struggle to generalize to out-of-distribution (OOD) data. In response to the efficiency problem, recent studies show that dense PLMs can be replaced with sparse subnetworks without hurting the performance. Such subnetworks can be found in three scenarios: 1) the fine-tuned PLMs, 2) the raw PLMs and then fine-tuned in isolation, and even inside 3) PLMs without any parameter fine-tuning. However, these results are only obtained in the in-distribution (ID) setting. In this paper, we extend the study on PLMs subnetworks to the OOD setting, investigating whether sparsity and robustness to dataset bias can be achieved simultaneously. To this end, we conduct extensive experiments with the pre-trained BERT model on three natural language understanding (NLU) tasks. Our results demonstrate that **sparse and robust subnetworks (SRNets) can consistently be found in BERT**, across the aforementioned three scenarios, using different training and compression methods. Furthermore, we explore the upper bound of SRNets using the OOD information and show that **there exist sparse and almost unbiased BERT subnetworks**. Finally, we present 1) an analytical study that provides insights on how to promote the efficiency of SRNets searching process and 2) a solution to improve subnetworks' performance at high sparsity. The code is available at `https://github.com/llyx97/sparse-and-robust-PLM`.

1 Introduction

Pre-trained language models (PLMs) have enjoyed impressive success in natural language processing (NLP) tasks. However, they still face two major problems. On the one hand, the prohibitive model size of PLMs leads to poor efficiency in terms of memory footprint and computational cost [12, 49]. On the other hand, despite being pre-trained on large-scale corpus, PLMs still tend to rely on *dataset bias* [18, 37, 65, 46], i.e., the spurious features of input examples that strongly correlate with the

*Work was done when Yuanxin Liu was a graduate student of IIE, CAS.
†Corresponding author: Zheng Lin.
Joint work with Pattern Recognition Center, WeChat AI, Tencent Inc, China.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

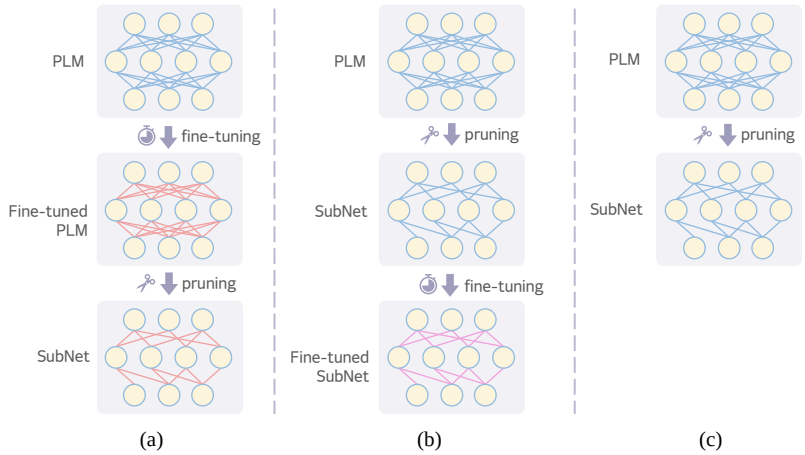


Figure 1: Three kinds of PLM subnetworks obtained from different pruning and fine-tuning paradigms. (a) Pruning a fine-tuned PLM. (b) Pruning the PLM and then fine-tuning the subnetwork. (c) Pruning the PLM without fine-tuning model parameters. The obtained subnetworks are used for testing.

label, during downstream fine-tuning. These two problems pose great challenge to the real-world deployment of PLMs, and they have triggered two separate lines of works.

In terms of the efficiency problem, some recent studies resort to sparse subnetworks as alternatives to the dense PLMs. [27, 38, 30] compress the fine-tuned PLMs in a post-hoc fashion. [4, 40, 32, 28] extend the *Lottery Ticket Hypothesis* (LTH) [9] to search PLMs subnetworks that can be fine-tuned in isolation. Taking one step further, [66] propose to learn task-specific subnetwork structures via mask training [23, 35], without fine-tuning any pre-trained parameter. Fig. 1 illustrates these three paradigms. Encouragingly, the empirical evidences suggest that PLMs can indeed be replaced with sparse subnetworks without compromising the in-distribution (ID) performance.

To address the dataset bias problem, numerous debiasing methods have been proposed. A prevailing category of debiasing methods [5, 54, 25, 20, 46, 13, 55] adjust the importance of training examples, in terms of training loss, according to their bias degree, so as to reduce the impact of biased examples (examples that can be correctly classified based on the spurious features). As a result, the model is forced to rely less on the dataset bias during training and generalizes better to OOD situations.

Although progress has been made in both directions, most existing work tackle the two problems independently. To facilitate real-world application of PLMs, the problems of robustness and efficiency should be addressed simultaneously. Motivated by this, we extend the study on PLM subnetwork to the OOD scenario, investigating **whether there exist PLM subnetworks that are both sparse and robust against dataset bias?** To answer this question, we conduct large-scale experiments with the pre-trained BERT model [6] on three natural language understanding (NLU) tasks that are widely-studied in the question of dataset bias. We consider a variety of setups including the three pruning and fine-tuning paradigms, standard and debiasing training objectives, different model pruning methods, and different variants of PLMs from the BERT family. Our results show that **BERT does contain sparse and robust subnetworks (SRNets)** within certain sparsity constraint (e.g., less than 70%), giving affirmative answer to the above question. Compared with a standard fine-tuned BERT, SRNets exhibit comparable ID performance and remarkable OOD improvement. When it comes to BERT model fine-tuned with debiasing method, SRNets can preserve the full model's ID and OOD performance with much fewer parameters. On this basis, we further explore the upper bound of SRNets by making use of the OOD information, which reveals that **there exist sparse and almost unbiased subnetworks, even in a standard fine-tuned BERT that is biased**.

Regardless of the intriguing properties of SRNets, we find that the subnetwork searching process still have room for improvement, based on some observations from the above experiments. First, we study the timing to start searching SRNets during full BERT fine-tuning, and find that the entire training and searching cost can be reduced from this perspective. Second, we refine the mask training method with gradual sparsity increase, which is quite effective in identifying SRNets at high sparsity.

Our main contributions are summarized as follows:

- We extend the study on PLMs subnetworks to the OOD scenario. To our knowledge, this paper presents the first systematic study on sparsity and dataset bias robustness for PLMs.
- We conduct extensive experiments to demonstrate the existence of sparse and robust BERT subnetworks, across different pruning and fine-tuning setups. By using the OOD information, we further reveal that there exist sparse and almost unbiased BERT subenetworks.
- We present analytical studies and solutions that can help further refine the SRNets searching process in terms of efficiency and the performance of subnetworks at high sparsity.

2 Related Work

2.1 BERT Compression

Studies on BERT compression can be divided into two classes. The first one focuses on the design of model compression techniques, which include pruning [15, 38, 11], knowledge distillation [44, 50, 24, 31], parameter sharing [26], quantization [61, 64], and combining multiple techniques [51, 36, 30]. The second one, which is based on the lottery ticket hypothesis [9], investigates the compressibility of BERT on different phases of the pre-training and fine-tuning paradigm. It has been shown that BERT can be pruned to a sparse subnetwork after [11] and before fine-tuning [4, 40, 28, 32, 15], without hurting the accuracy. Moreover, [66] show that directly learning subnetwork structures on the pre-trained weights can match fine-tuning the full BERT. In this paper, we follow the second branch of works, and extend the evaluation of BERT subnetworks to the OOD scenario.

2.2 Dataset Bias in NLP Tasks

To facilitate the development of NLP systems that truly learn the intended task solution, instead of relying on dataset bias, many efforts have been made recently. On the one hand, challenging OOD test sets are constructed [18, 37, 65, 46, 1] by eliminating the spurious correlations in the training sets, in order to establish more strict evaluation. On the other hand, numerous debiasing methods [5, 54, 25, 20, 46, 13, 55] are proposed to discourage the model from learning dataset bias during training. However, few attention has been paid to the influence of pruning on the OOD generalization ability of PLMs. This work presents a systematic study on this question.

2.3 Model Compression and Robustness

Some pioneer attempts have also been made to obtain models that are both compact and robust to adversarial attacks [16, 60, 48, 10, 59] and spurious correlations [62, 8]. Specially, [59, 8] study the compression and robustness question on PLM. Different from [59], which is based on adversarial robustness, we focus on the spurious correlations, which is more common than the worst-case adversarial attack. Compared with [8], which focus on post-hoc pruning of the standard fine-tuned BERT, we thoroughly investigate different fine-tuning methods (standard and debiasing) and subnetworks obtained from the three pruning and fine-tuning paradigms. A more detailed discussion of the relation and difference between our work and previous studies on model compression and robustness is provided in Appendix D.

3 Preliminaries

3.1 BERT Architecture and Subnetworks

BERT is composed of an embedding layer, a stack of Transformer layers [56] and a task-specific classifier. Each Transformer layer has a multi-head self-attention (MHAtt) module and a feed-forward network (FFN). MHAtt has four kinds of weight matrices, i.e., the query, key and value matrices $\mathbf{W}_{Q,K,V} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, and the output matrix $\mathbf{W}_{AO} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$. FFN consits of two linear layers $\mathbf{W}_{\text{in}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{FFN}}}$, $\mathbf{W}_{\text{out}} \in \mathbb{R}^{d_{\text{FFN}} \times d_{\text{model}}}$, where d_{FFN} is the hidden dimension of FFN.

To obtain the subnetwork of a model $f(\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$, we apply a binary pruning mask $\mathbf{m} \in \{0,1\}^{|\boldsymbol{\theta}|}$ to its weight matrices, which produces $f(\mathbf{m} \odot \boldsymbol{\theta})$, where $\odot$ is the Hadamard product.

For BERT, we focus on the L Transformer layers and the classifier. The parameters to be pruned are $\boldsymbol{\theta}_{pr} = \{\mathbf{W}_{\text{cls}}\} \cup \left\{\mathbf{W}_Q^l, \mathbf{W}_K^l, \mathbf{W}_V^l, \mathbf{W}_{AO}^l, \mathbf{W}_{\text{in}}^l, \mathbf{W}_{\text{out}}^l\right\}_{l=1}^{L}$, where $\mathbf{W}_{\text{cls}}$ is the classifier weights.

3.2 Pruning Methods

3.2.1 Magnitude-based Pruning

Magnitude-based pruning [19, 9] zeros-out parameters with low absolute values. It is usually realized in an iterative manner, namely, iterative magnitude pruning (IMP). IMP alternates between pruning and training and gradually increases the sparsity of subnetworks. Specifically, a typical IMP algorithm consists of four steps: (i) Training the full model to convergence. (ii) Pruning a fraction of parameters with the smallest magnitude. (iii) Re-training the pruned subnetwork. (iv) Repeat (ii)-(iii) until reaching the target sparsity. To obtain subnetworks from the pre-trained BERT, i.e., (b) and (c) in Fig. 1, the subnetwork parameters are rewound to the pre-trained values after (iii), and (i) can be abandoned. More details about our IMP implementations can be found in Appendix A.1.1.

3.2.2 Mask Training

Mask training treats the pruning mask $\mathbf{m}$ as trainable parameters. Following [35, 66, 42, 32], we achieve this through binarization in forward pass and gradient estimation in backward pass.

Each weight matrix $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$, which is frozen during mask training, is associated with a bianry mask $\mathbf{m} \in \{0,1\}^{d_1 \times d_2}$, and a real-valued mask $\hat{\mathbf{m}} \in \mathbb{R}^{d_1 \times d_2}$. In the forward pass, $\mathbf{W}$ is replaced with $\mathbf{m} \odot \mathbf{W}$, where $\mathbf{m}$ is derived from $\hat{\mathbf{m}}$ through binarization:

$$\mathbf{m}_{i,j} = \begin{cases} 1 & \text{if } \hat{\mathbf{m}}_{i,j} \geq \phi \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where ϕ is the threshold. In the backward pass, since the binarization operation is not differentiable, we use the *straight-through estimator* [3] to compute the gradients for $\hat{\mathbf{m}}$ using the gradients of $\mathbf{m}$, i.e., $\frac{\partial \mathcal{L}}{\partial \mathbf{m}}$, where $\mathcal{L}$ is the loss. Then, $\hat{\mathbf{m}}$ is updated as $\hat{\mathbf{m}} \leftarrow \hat{\mathbf{m}} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{m}}$, where η is the learning rate.

Following [42, 32], we initialize the real-valued masks according to the magnitude of the original weights. The complete mask training algorithm is summarized in Appendix A.1.2.

3.3 Debiasing Methods

As described in the Introduction, the debiasing methods measure the bias degree of training examples. This is achieved by training a *bias model*. The inputs to the bias model are hand-crafted spurious features based on our prior knowledge of the dataset bias (Section 4.1.3 describes the details). In this way, the bias model mainly relies on the spurious features to make predictions, which can then serve as a measurement of the bias degree. Specifically, given the bias model prediction $\mathbf{p}_b = (\mathbf{p}_b^1, \cdots, \mathbf{p}_b^K)$ over the K classes, the bias degree $\beta = \mathbf{p}_b^c$, i.e., the the probability of the ground-truth class c.

Then, β can be used to adjust the training loss in several ways, including *product-of-experts* (PoE) [5, 20, 25], *example reweighting* [46, 13] and *confidence regularization* [54]. Here we describe the standard cross-entropy and PoE, and the other two methods are introduced in Appendix A.2.

Standard Cross-Entropy computes the cross-entropy between the predicted distribution $\mathbf{p}_m$ and the ground-truth one-hot distribution $\mathbf{y}$ as $\mathcal{L}_{\text{std}} = -\mathbf{y} \cdot \log \mathbf{p}_m$.

Product-of-Experts combines the predictions of main model and bias model, i.e., $\mathbf{p}_b$ and $\mathbf{p}_m$, and then computes the training loss as $\mathcal{L}_{\text{poe}} = -\mathbf{y} \cdot \log \text{softmax}\left(\log \mathbf{p}_m + \log \mathbf{p}_b\right)$.

3.4 Notations

Here we define some notations, which will be used in the following sections.

- $\mathcal{A}_{\mathcal{L}}^t(f(\boldsymbol{\theta}))$: Training $f(\boldsymbol{\theta})$ with loss $\mathcal{L}$ for t steps, where t can be omitted for simplicity.
- $\mathcal{P}_{\mathcal{L}}^p(f(\boldsymbol{\theta}))$: Pruning $f(\boldsymbol{\theta})$ using pruning method p and training loss $\mathcal{L}$.
- $\mathcal{M}(f(\mathbf{m}\boldsymbol{\theta}))$: Extracting the pruning mask of $f(\mathbf{m}\boldsymbol{\theta})$, i.e., $\mathcal{M}(f(\mathbf{m}\boldsymbol{\theta})) = \mathbf{m}$.

- $\mathcal{L} \in \{\mathcal{L}_{\text{std}}, \mathcal{L}_{\text{poe}}, \mathcal{L}_{\text{reweight}}, \mathcal{L}_{\text{confreg}}\}$ and $p \in \{\text{imp}, \text{imp-rw}, \text{mask}\}$, where "imp" and "imp-rw"denote the standard IMP and IMP with weight rewinding, as described in Section 3.2.1. "mask" stands for mask training.
- $\mathcal{E}_d(f(\boldsymbol{\theta}))$: Evaluating $f(\boldsymbol{\theta})$ on the test data with distribution $d \in \{\text{ID}, \text{OOD}\}$.

4 Sparse and Robust BERT Subnetworks

4.1 Experimental Setups

4.1.1 Datasets and Evaluation

Natural Language Inference We use MNLI [57] as the ID dataset for NLI. MNLI is comprised of premise-hypothesis pairs, whose relationship may be *entailment*, *contradiction*, or *neutral*. In MNLI the word overlap between premise and hypothesis is strongly correlated with the *entailment* class. To solve this problem, the OOD HANS dataset [37] is built so that such correlation does not hold.

Paraphrase Identification The ID dataset for paraphrase identification is QQP [3], which contains question pairs that are labelled as either *duplicate* or *non-duplicate*. In QQP, high lexical overlap is also strongly associated with the *duplicate* class. The OOD datasets PAWS-qqp and PAWS-wiki [65] are built from sentences in Quora and Wikipedia respectively. In PAWS sentence pairs with high word overlap have a balanced distribution over *duplicate* and *non-duplicate*.

Fact Verification FEVER [4] [52] is adopted as the ID dataset of fact verification, where the task is to assess whether a given evidence *supports* or *refutes* the claim, or whether there is *not-enough-info* to reach a conclusion. The OOD dataset Fever-Symmetric (v1 and v2) [46] is proposed to evaluate the influence of the claim-only bias (the label can be predicted correctly without the evidence).

For NLI and fact verification, we use Accuracy as the evaluation metric. For paraphrase identification, we evaluate using the F1 score. More details of datasets and evaluation are shown in Appendix B.1.

4.1.2 PLM Backbone

We mainly experiment with the BERT-base-uncased model [6]. It has roughly 110M parameters in total, and 84M parameters in the Transformer layers. As described in Section 3.1, we derive the subnetworks from the Transformer layers and report sparsity levels relative to the 84M parameters. To generalize our conclusions to other PLMs, we also consider two variants of the BERT family, namely RoBERTa-base and BERT-large, the results of which can be found in Appendix C.5.

4.1.3 Training Details

Following [5], we use a simple linear classifier as the bias model. For HANS and PAWS, the spurious features are based on the the word overlapping information between the two input text sequences. For Fever-Symmetric, the spurious features are max-pooled word embeddings of the claim sentence. More details about the bias model and the spurious features are presented in Appendix B.3.1.

Mask training and IMP basically use the same hyper-parameters (adopting from [55]) as full BERT. An exception is longer training, because we find that good subnetworks at high sparsity levels require more training to be found. Unless otherwise specified, we select the best checkpoints based on the ID dev performance, without using OOD information. All the reported results are averaged over 4 runs. We defer training details about each dataset, and each training and pruning setup, to Appendix B.3.

4.2 Subnetworks from Fine-tuned BERT

4.2.1 Problem Formulation and Experimental Setups

Given the fine-tuned full BERT $f(\boldsymbol{\theta}_{ft}) = \mathcal{A}_{\mathcal{L}_1}(f(\boldsymbol{\theta}_{pt}))$, where $\boldsymbol{\theta}_{pt}$ and $\boldsymbol{\theta}_{ft}$ are the pre-trained and fine-tuned parameters respectively, the goal is to find a subnetwork $f(\mathbf{m} \odot \boldsymbol{\theta}_{ft}^{'}) = \mathcal{P}^{p}_{\mathcal{L}_2}(f(\boldsymbol{\theta}_{ft}))$ that

[3] `https://www.kaggle.com/c/quora-question-pairs`

[4] See the licence information at `https://fever.ai/download/fever/license.html`

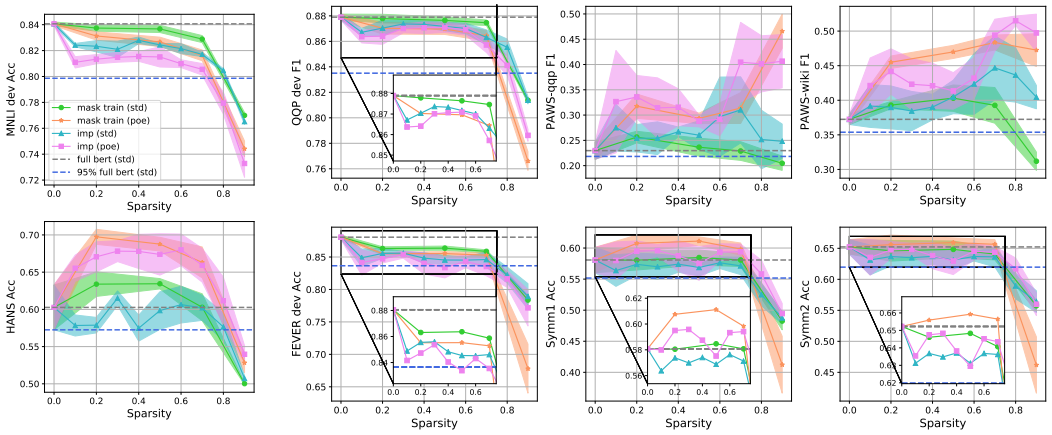


Figure 2: Results of subnetworks pruned from the CE fine-tuned BERT. "std" means standard, and the shadowed areas denote standard deviations, which also apply to the other figures of this paper.

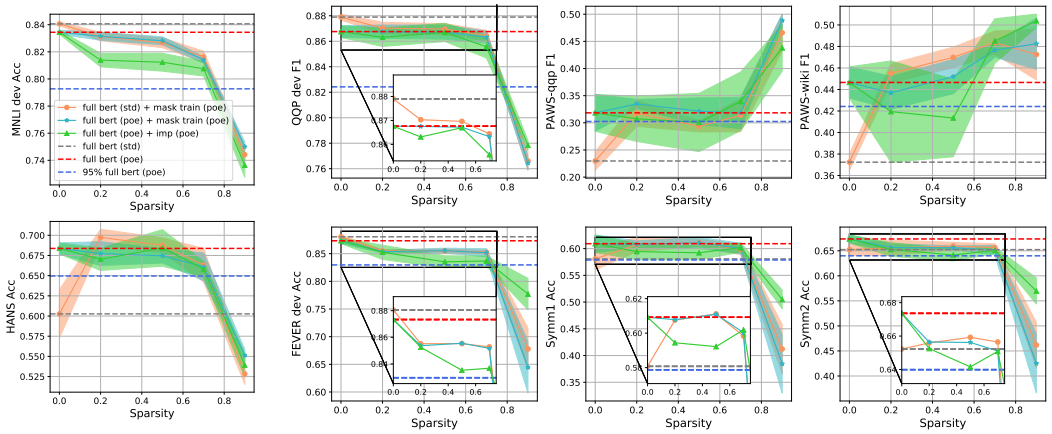


Figure 3: Results of subnetworks pruned from the PoE fine-tuned BERT. Results of the "mask train (poe)" subnetworks from Fig. 2 (the orange line) are also reported for reference.

satisfies a target sparsity level s and maximize the ID and OOD performance.

$$\max_{\mathbf{m},\boldsymbol{\theta}'_{ft}} \left(\mathcal{E}_{\text{ID}}\left(f\left(\mathbf{m}\odot\boldsymbol{\theta}'_{ft}\right)\right)+\mathcal{E}_{\text{OOD}}\left(f\left(\mathbf{m}\odot\boldsymbol{\theta}'_{ft}\right)\right)\right),\ \text{s.t.}\ \frac{\|\mathbf{m}\|_0}{|\boldsymbol{\theta}_{pr}|}=(1-s) \tag{2}$$

where $\|\|_0$ is the L_0 norm and $|\boldsymbol{\theta}_{pr}|$ is the total number of parameters to be pruned. In practice, the above optimization problem is achieved via $\mathcal{P}^{p}_{\mathcal{L}_2}()$, which minimizes the loss $\mathcal{L}_2$ on the ID training set. When the pruning method is IMP, the subnetwork parameters will be further fine-tuned and $\boldsymbol{\theta}'_{ft} \neq \boldsymbol{\theta}_{ft}$. For mask training, only the subnetwork structure is updated and $\boldsymbol{\theta}'_{ft} = \boldsymbol{\theta}_{ft}$.

We consider two kinds of fine-tuned full BERT, which utilize the standard CE loss and PoE loss respectively (i.e., $\mathcal{L}_1 \in \{\mathcal{L}_{\text{std}}, \mathcal{L}_{\text{poe}}\}$). IMP and mask training are used as the pruning methods (i.e., $p \in \{\text{imp}, \text{mask}\}$). For the standard fine-tuned BERT, both $\mathcal{L}_{\text{std}}$ and $\mathcal{L}_{\text{poe}}$ are examined in the pruning process. For the PoE fine-tuned BERT, we only use $\mathcal{L}_{\text{poe}}$ during pruning. Note that in this work, we mainly experiment with $\mathcal{L}_{\text{std}}$ and $\mathcal{L}_{\text{poe}}$. $\mathcal{L}_{\text{reweight}}$ and $\mathcal{L}_{\text{confreg}}$ are also examined for subnetworks from fine-tuned BERT, the results of which can be found in Appendix C.1.

4.2.2 Results

Subnetworks from Standard Fine-tuned BERT The results are shown in Fig. 2 (In this paper, we present most results in figures for clear comparisons. Actual values of the results can be found in the code link.). We discuss them from three perspectives. For the full BERT, we can see that standard

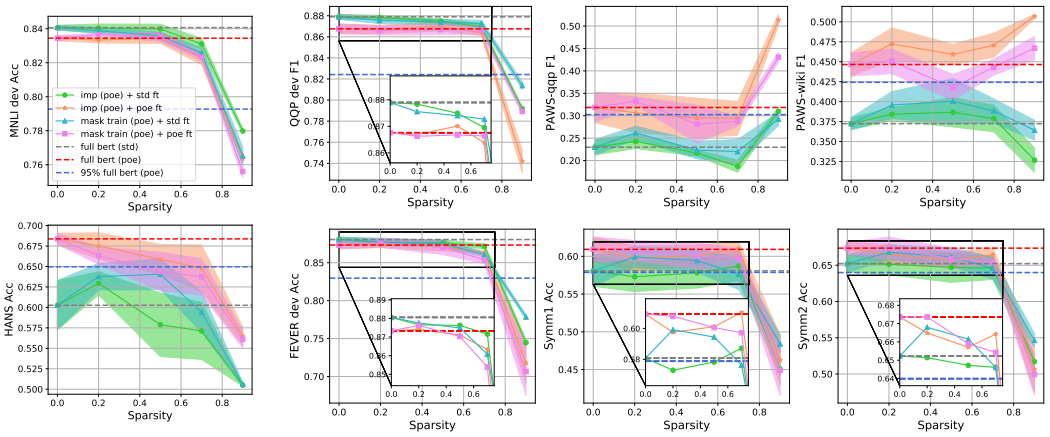


Figure 4: Results of BERT subnetworks fine-tuned in isolation. "ft" is short for fine-tuning.

CE fine-tuning, which achieves good results on the ID dev sets, performs significantly worse on the OOD test sets. This demonstrates that the ID performance of BERT depends, to a large extent, on memorizing the dataset bias.

In terms of the subnetworks, we can derive the following observations: (1) Using any of the four pruning methods, we can compress a large proportion of the BERT parameters (up to 70% sparsity) and still preserve 95% of the full model's ID performance. (2) With standard pruning, i.e., "mask train (std)" or "imp (std)", we can observe small but perceivable improvement over the full BERT on the HANS and PAWS datasets. This suggests that pruning may remove some parameters related to the bias features. (3) The OOD performance of "mask train (poe)" and "imp (poe)" subnetworks is even better, and the ID performance degrades slightly but is still above 95% of the full BERT. This shows that introducing the debiasing objective in the pruning process is beneficial. Specially, as mask training does not change the model parameters, the results of "mask train (poe)" implicates that the biased "full bert (std)" contains sparse and robust subnetworks (SRNets) that already encode a less biased solution to the task. (4) SRNets can be identified across a wide range of sparsity levels (from $20\% \sim 70\%$). However at higher sparsity of 90%, the performance of the subnetworks is not desirable. (5) We also find that there is an abnormal increase of the PAWS F1 score at $70\% \sim 90\%$ sparsity for some pruning methods, when the corresponding ID performance drops sharply. This is because the class distribution of PAWS is imbalanced (see Appendix B.1), and thus even a naive random-guessing model can outperform the biased full model on PAWS. Therefore, the OOD improvement should only be acceptable when there is no large ID performance decline.

Comparing IMP and mask training, the latter performs better in general, except for "mask train (poe)" at 90% sparsity on QQP and FEVER. This suggests that directly optimizing the subnetwork structure is a better choice than using the magnitude heuristic as the pruning metric.

Subnetworks from PoE Fine-tuned BERT Fig. 3 presents the results. We can find that: (1) For the full BERT, the OOD performance is obviously promoted with the PoE debiasing method, while the ID performance is sacrificed slightly. (2) Unlike the subnetworks from the standard fine-tuned BERT, the subnetworks of PoE fine-tuned BERT (the green and blue lines) cannot outperform the full model. However, these subnetworks maintain comparable performance at up to 70% sparsity, on both the ID and OOD settings, making them desirable alternatives to the full model in resource-constraint scenarios. Moreover, this phenomenon suggests that there is a great redundancy of BERT parameters, even when OOD generalization is taken into account. (3) With PoE-based pruning, subnetworks from the standard fine-tuned BERT (the orange line) is comparable with subnetworks from the PoE fine-tuned BERT (the blue line). This means we do not have to fine-tune a debiased BERT before searching for the SRNets. (4) IMP, again, slightly underperforms mask training at moderate sparsity levels, while it is better at 90% sparsity on the fact verification task.

4.3 BERT Subnetworks Fine-tuned in Isolation

4.3.1 Problem Formulation and Experimental Setups

Given the pre-trained BERT $f(\boldsymbol{\theta}_{pt})$, a subnetwork $f(\mathbf{m} \odot \boldsymbol{\theta}_{pt})$ is obtained before downstream fine-tuning. The goal is to maximize the performance of the fine-tuned subnetwork $\mathcal{A}_{\mathcal{L}_1}(f(\mathbf{m} \odot \boldsymbol{\theta}_{pt}))$:

$$\max_{\mathbf{m}} \left(\mathcal{E}_{\text{ID}}\left(\mathcal{A}_{\mathcal{L}_1}(f(\mathbf{m} \odot \boldsymbol{\theta}_{pt}))\right) + \mathcal{E}_{\text{OOD}}\left(\mathcal{A}_{\mathcal{L}_1}(f(\mathbf{m} \odot \boldsymbol{\theta}_{pt}))\right)\right), \text{ s.t. } \frac{\|\mathbf{m}\|_0}{|\boldsymbol{\theta}_{pr}|} = (1-s) \tag{3}$$

Following the LTH [9], we solve this problem using the train-prune-rewind pipeline. For IMP, the procedure is described in Section 3.2.1 and $\mathbf{m} = \mathcal{M}(\mathcal{P}_{\mathcal{L}_2}^{\text{imp-rw}}(f(\boldsymbol{\theta}_{pt})))$. For mask training, the subnetwork structure is learned from $f(\boldsymbol{\theta}_{ft})$ (same as the previous section) and $\mathbf{m} = \mathcal{M}(\mathcal{P}_{\mathcal{L}_2}^{\text{mask}}(f(\boldsymbol{\theta}_{ft})))$.

We employ CE and PoE loss for model fine-tuning (i.e., $\mathcal{L}_1 \in \{\mathcal{L}_{\text{std}}, \mathcal{L}_{\text{poe}}\}$). Since we have shown that using the debiasing loss in pruning is conducive, the CE loss is not considered (i.e., $\mathcal{L}_2 = \mathcal{L}_{\text{poe}}$).

4.3.2 Results

The results of subnetworks fine-tuned in isolation are presented in Fig. 4. It can be found that: (1) For standard CE fine-tuning, the "mask train (poe)" subnetworks are superior to "full bert (std)" on the OOD test data, i.e., the subnetworks are less susceptible to the dataset bias during training. (2) In terms of the PoE-based fine-tuning, the "imp (poe)" and "mask train (poe)" subnetworks are generally comparable to "full bert (poe)". (3) For most of the subnetworks, "poe ft" clearly outperforms "std ft" in the OOD setting, which suggests that it is important to use the debiasing method in fine-tuning, even if the BERT subnetwork structure has already encoded some unbiased information.

Moreover, based on (1) and (2), we can extend the LTH on BERT [4, 40, 28, 32]: **The pre-trained BERT contains SRNets that can be fine-tuned in isolation, using either standard or debiasing method, and match or even outperform the full model in both the ID and OOD evaluations.**

4.4 BERT Subnetworks Without Fine-tuning

4.4.1 Problem Formulation and Experimental Setups

This setup aims at finding a subnetwork $f(\mathbf{m} \odot \boldsymbol{\theta}_{pt})$ inside the pre-trained BERT, which can be directly employed to a task. The problem is formulated as:

$$\max_{\mathbf{m}} \left(\mathcal{E}_{\text{ID}}\left(f(\mathbf{m} \odot \boldsymbol{\theta}_{pt})\right) + \mathcal{E}_{\text{OOD}}\left(f(\mathbf{m} \odot \boldsymbol{\theta}_{pt})\right)\right), \text{ s.t. } \frac{\|\mathbf{m}\|_0}{|\boldsymbol{\theta}_{pr}|} = (1-s) \tag{4}$$

Following [66], we fix the pre-trained parameters $\boldsymbol{\theta}_{pt}$ and optimize the mask variables $\mathbf{m}$. This process can be represented as $\mathcal{P}_{\mathcal{L}}^{\text{mask}}(f(\boldsymbol{\theta}_{pt}))$, where $\mathcal{L} \in \{\mathcal{L}_{\text{std}}, \mathcal{L}_{\text{poe}}\}$.

4.4.2 Results

As we can see in Fig. 5: (1) With CE-based mask training, the identified subnetworks (under 50% sparsity) in pre-trained BERT are competitive with the CE fine-tuned full BERT. (2) Similarly, using PoE-based mask training, the subnetworks under 50% sparsity are comparable to the PoE fine-tuned full BERT, which demonstrates that SRNets for a particular downstream task already exist in the pre-trained BERT. (3) "mask train (poe)" subnetworks in pre-trained BERT can even match the subnetworks found in the fine-tuned BERT (the orange lines) in some cases (e.g., on PAWS and on FEVER under 50% sparsity). Nonetheless, the latter exhibits a better overall performance.

4.5 Sparse and Unbiased BERT Subnetworks

4.5.1 Problem Formulation and Experimental Setups

To explore the upper bound of BERT subnetworks in terms of OOD generalization, we include the OOD training data in mask training, and use the OOD test sets for evaluation. Like the previous sections, we investigate three pruning and fine-tuning paradigms, as formulated by Eq. 2, 3 and 4 respectively. We only consider the standard CE for subnetwork and full BERT fine-tuning, which is more vulnerable to the dataset bias. Appendix B.3.3 summarizes the detailed experimental setups.

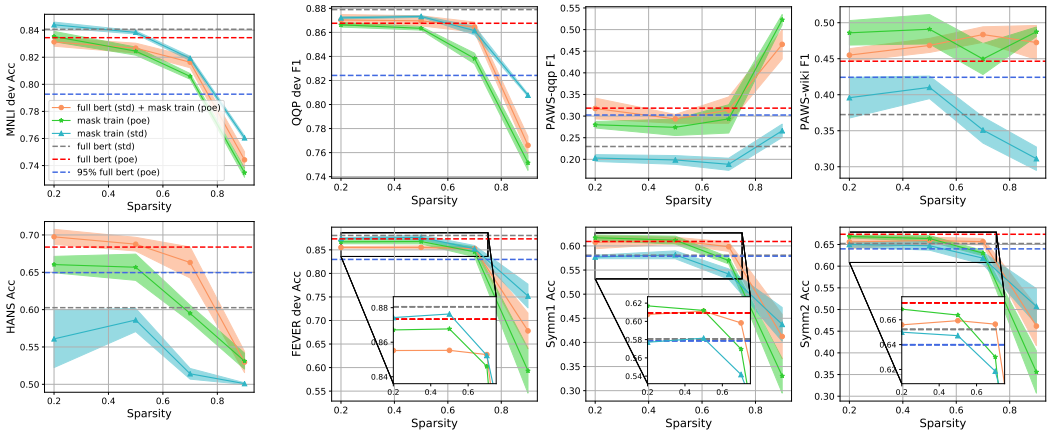


Figure 5: Results of BERT subnetworks without fine-tuning. Results of the "mask train (poe)" subnetworks from Fig. 2 (the orange line) are also reported for reference.

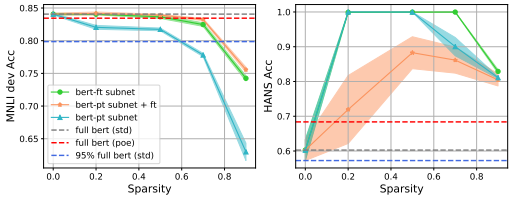


Figure 6: NLI results of BERT subnetworks found using the OOD information. Results of the other two tasks can be found in Appendix C.2.

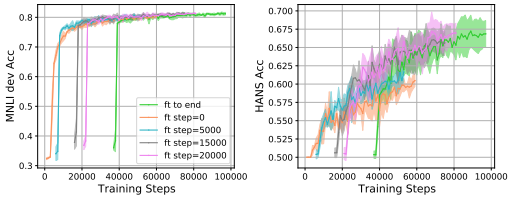


Figure 7: NLI mask training curves (70% sparse), starting from BERT fine-tuned for varied steps. Appendix C.3 shows results of the other two tasks.

4.5.2 Results

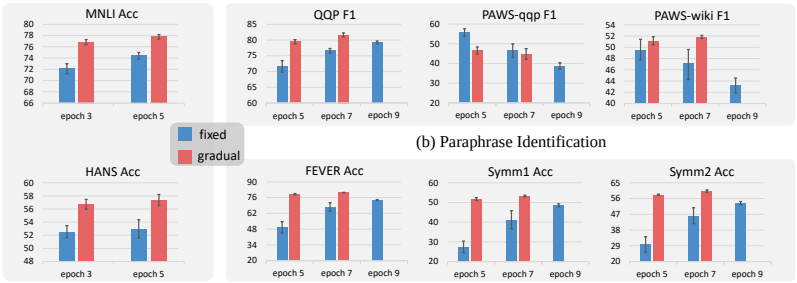
From Fig. 6 we can observe that: (1) The subnetworks from fine-tuned BERT ("bert-ft subnet") at $20\% \sim 70\%$ sparsity achieve nearly 100% accuracy on HANS, and their ID performance is also close to the full BERT. (2) The subnetworks in the pre-trained BERT ("bert-pt subnet") also have very high OOD accuracy, while they perform worse than "bert-ft subnet" in the ID setting. (3) "bert-pt subnet + ft" subnetworks, which are fine-tuned in isolation with CE loss, exhibits the best ID performance, and the poorest OOD performance. However, compared to the full BERT, these subnetworks still rely much less on the dataset bias, reaching nearly 90% HANS accuracy at 50% sparsity. Jointly, these results show that there consistently exist BERT subnetworks that are almost unbiased towards the MNLI training set bias, under the three kinds of pruning and fine-tuning paradigms.

5 Refining the SRNets Searching Process

In this section, we study how to further improve the SRNets searching process based on mask training, which generally performs better than IMP, as shown in Section 4.2 and Section 4.3.

5.1 The Timing to Start Searching SRNets

Compared with searching subnetworks from the fine-tuned BERT, directly searching from the pre-trained BERT is more efficient in that it dispenses with fine-tuning the full model. However, the former has a better overall performance, as we have shown in Section 4.4. This induces a question: **At which point of the BERT fine-tuning process, can we find subnetworks comparable to those found after the end of fine-tuning using mask training?** To answer this question, we perform mask training on the model checkpoints $f(\boldsymbol{\theta}_t) = \mathcal{A}^t_{\mathcal{L}_{\text{std}}}(f(\boldsymbol{\theta}_{pt}))$ from different steps t of BERT fine-tuning.



(a) NLI (b) Paraphrase Identification (c) Fact Verification

Figure 8: Comparison between fixed sparsity and gradual sparsity increase for mask training with the standard fine-tuned full BERT. The subnetworks are at 90% sparsity.

Fig. 7 shows the mask training curves, which start from different $f(\boldsymbol{\theta}_t)$. We can see that "ft step=0" converges slower and to a worse final accuracy, as compared with "ft to end", especially on the HANS dataset. However, with 20,000 steps of full BERT fine-tuning, which is roughly 55% of the "ft to end", the mask training performance is very competitive. This suggests that the total training cost of SRNet searching can be reduced, by a large amount, in the full model training stage.

To actually reduce the training cost, we need to predict the exact timing to start mask training. This is intractable without information of all the training curves in Fig. 7. A feasible solution is adopting the idea of early-stopping (see Appendix E.1 for detailed discussions). However, accurately predicting the optimal timing (with the least amount of fine-tuning and comparable subnetwork performance to fully fine-tuning) is indeed difficult and we invite follow-up studies to investigate this question.

5.2 SRNets at High Sparsity

As the results of Section 4 demonstrate, there is a sharp decline of the subnetworks' performance from $70\% \sim 90\%$ sparsity. We conjecture that this is because directly initializing mask training to 90% reduces the model's capacity too drastically, and thus causes some difficulties in optimization. Therefore, we gradually increase the sparsity from $70\% \sim 90\%$ during mask training, using the cubic sparsity schedule [67] (see Appendix C.4 for ablation studies). Fig. 8 compares the fixed sparsity used in the previous sections and the gradual sparsity increase, across varied mask training epochs. We find that while simply extending the training process is conducive, gradual sparsity increase achieves better results. In particular, "gradual" outperforms "fixed" with lower training cost on all the three tasks, except for the PAWS dataset, A similar phenomenon is explained in Section 4.2.2.

6 Conclusions and Limitations

In this paper, we investigate whether sparsity and robustness to dataset bias can be achieved simultaneously for PLM subnetworks. Through extensive experiments, we demonstrate that BERT indeed contains sparse and robust subnetworks (SRNets) across a variety of NLU tasks and training and pruning setups. We further use the OOD information to reveal that there exist sparse and almost unbiased BERT subnetworks. Finally, we present analysis and solutions to refine the SRNet searching process in terms of subnetwork performance and searching efficiency.

The limitations of this work is twofold. First, we focus on BERT-like PLMs and NLU tasks, while dataset biases are also common in other scenarios. For example, gender and racial biases exist in dialogue generation systems [7] and PLMs [17]. In the future work, we would like to extend our exploration to other types of PLMs and NLP tasks (see Appendix E.2 for a discussion). Second, as we discussed in Section 5.1, our analysis on "the timing to start searching SRNets" mainly serves as a proof-of-concept, and actually reducing the training cost requires predicting the exact timing.

Acknowledgments and Disclosure of Funding

This work was supported by National Natural Science Foundation of China (61976207 and 61906187).

References

[1] A. Agrawal, D. Batra, D. Parikh, and A. Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering. In *CVPR*, pages 4971–4980. Computer Vision Foundation / IEEE Computer Society, 2018.

[2] S. Beery, G. V. Horn, and P. Perona. Recognition in terra incognita. In *ECCV (16)*, volume 11220 of *Lecture Notes in Computer Science*, pages 472–489. Springer, 2018.

[3] Y. Bengio, N. Léonard, and A. C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[4] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, Z. Wang, and M. Carbin. The lottery ticket hypothesis for pre-trained BERT networks. In *NeurIPS*, pages 15834–15846, 2020.

[5] C. Clark, M. Yatskar, and L. Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *EMNLP/IJCNLP*, pages 4069–4082. Association for Computational Linguistics, 2019.

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

[7] E. Dinan, A. Fan, A. Williams, J. Urbanek, D. Kiela, and J. Weston. Queens are powerful too: Mitigating gender bias in dialogue generation. In *EMNLP (1)*, pages 8173–8188. Association for Computational Linguistics, 2020.

[8] M. Du, S. Mukherjee, Y. Cheng, M. Shokouhi, X. Hu, and A. H. Awadallah. What do compressed large language models forget? robustness challenges in model compression. *CoRR*, abs/2110.08419, 2021.

[9] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*. OpenReview.net, 2019.

[10] Y. Fu, Q. Yu, Y. Zhang, S. Wu, X. Ouyang, D. D. Cox, and Y. Lin. Drawing robust scratch tickets: Subnetworks with inborn robustness are found within randomly initialized networks. In *NeurIPS*, pages 13059–13072, 2021.

[11] T. Gale, E. Elsen, and S. Hooker. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574, 2019.

[12] P. Ganesh, Y. Chen, X. Lou, M. A. Khan, Y. Yang, H. Sajjad, P. Nakov, D. Chen, and M. Winslett. Compressing large-scale transformer-based models: A case study on BERT. *Transactions of the Association for Computational Linguistics*, 9:1061–1080, 2021.

[13] A. Ghaddar, P. Langlais, M. Rezagholizadeh, and A. Rashid. End-to-end self-debiasing framework for robust NLU training. In *ACL/IJCNLP (Findings)*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1923–1929. Association for Computational Linguistics, 2021.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR (Poster)*, 2015.

[15] M. A. Gordon, K. Duh, and N. Andrews. Compressing BERT: studying the effects of weight pruning on transfer learning. In *RepL4NLP@ACL*, pages 143–155, 2020.

[16] S. Gui, H. Wang, H. Yang, C. Yu, Z. Wang, and J. Liu. Model compression with adversarial robustness: A unified optimization framework. In *NeurIPS*, pages 1283–1294, 2019.

[17] Y. Guo, Y. Yang, and A. Abbasi. Auto-debias: Debiasing masked language models with automated biased prompts. In *ACL*, pages 1012–1023. Association for Computational Linguistics, 2022.

[18] S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. R. Bowman, and N. A. Smith. Annotation artifacts in natural language inference data. In *NAACL-HLT*, pages 107–112. Association for Computational Linguistics, 2018.

[19] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems 28*, pages 1135–1143. Curran Associates, Inc., 2015.

[20] H. He, S. Zha, and H. Wang. Unlearn dataset bias in natural language inference by fitting the residual. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 132–142. Association for Computational Linguistics, 2019.

[21] D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan, and D. Song. Pretrained transformers improve out-of-distribution robustness. In *ACL*, pages 2744–2751. Association for Computational Linguistics, 2020.

[22] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[23] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *NIPS*, pages 4107–4115, 2016.

[24] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. Tinybert: Distilling BERT for natural language understanding. In *EMNLP (Findings)*, pages 4163–4174, 2020.

[25] R. Karimi Mahabadi, Y. Belinkov, and J. Henderson. End-to-end bias mitigation by modelling biases in corpora. In *ACL*, pages 8706–8716. Association for Computational Linguistics, 2020.

[26] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*. OpenReview.net, 2020.

[27] Z. Li, E. Wallace, S. Shen, K. Lin, K. Keutzer, D. Klein, and J. E. Gonzalez. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *CoRR*, abs/2002.11794, 2020.

[28] C. Liang, S. Zuo, M. Chen, H. Jiang, X. Liu, P. He, T. Zhao, and W. Chen. Super tickets in pre-trained language models: From model compression to improving generalization. In *ACL/IJCNLP*, pages 6524–6538. Association for Computational Linguistics, 2021.

[29] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[30] Y. Liu, Z. Lin, and F. Yuan. ROSITA: refined BERT compression with integrated techniques. In *AAAI*, pages 8715–8722. AAAI Press, 2021.

[31] Y. Liu, F. Meng, Z. Lin, W. Wang, and J. Zhou. Marginal utility diminishes: Exploring the minimum knowledge for BERT knowledge distillation. In *ACL/IJCNLP*, pages 2928–2941, 2021.

[32] Y. Liu, F. Meng, Z. Lin, P. Fu, Y. Cao, W. Wang, and J. Zhou. Learning to win lottery tickets in BERT transfer via task-agnostic mask training. *CoRR*, abs/2204.11218, 2022.

[33] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR (Poster)*. OpenReview.net, 2019.

[34] D. Madaan, J. Shin, and S. J. Hwang. Adversarial neural pruning with latent vulnerability suppression. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 6575–6585. PMLR, 2020.

[35] A. Mallya, D. Davis, and S. Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, volume 11208 of *Lecture Notes in Computer Science*, pages 72–88. Springer, 2018.

[36] Y. Mao, Y. Wang, C. Wu, C. Zhang, Y. Wang, Q. Zhang, Y. Yang, Y. Tong, and J. Bai. Ladabert: Lightweight adaptation of BERT through hybrid model compression. In *COLING*, pages 3225–3234, 2020.

[37] T. McCoy, E. Pavlick, and T. Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *ACL*, pages 3428–3448. Association for Computational Linguistics, 2019.

[38] P. Michel, O. Levy, and G. Neubig. Are sixteen heads really better than one? In *NeurIPS*, pages 14014–14024, 2019.

[39] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. In *LREC*. European Language Resources Association (ELRA), 2018.

[40] S. Prasanna, A. Rogers, and A. Rumshisky. When BERT plays the lottery, all tickets are winning. In *EMNLP*, pages 3208–3229, 2020.

[41] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding with unsupervised learning. In *Technical report, OpenAI*, 2018.

[42] E. Radiya-Dixit and X. Wang. How fine can fine-tuning be? learning efficient language models. In *AISTATS*, volume 108 of *Proceedings of Machine Learning Research*, pages 2435–2443. PMLR, 2020.

[43] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[44] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[45] V. Sanh, T. Wolf, and A. M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *NeurIPS*, pages 20378–20389, 2020.

[46] T. Schuster, D. J. Shah, Y. J. S. Yeo, D. Filizzola, E. Santus, and R. Barzilay. Towards debiasing fact verification models. In *EMNLP/IJCNLP*, pages 3417–3423. Association for Computational Linguistics, 2019.

[47] V. Sehwag, S. Wang, P. Mittal, and S. Jana. Towards compact and robust deep neural networks. *CoRR*, abs/1906.06110, 2019.

[48] V. Sehwag, S. Wang, P. Mittal, and S. Jana. HYDRA: pruning adversarially robust neural networks. In *NeurIPS*, 2020.

[49] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. In *ACL*, pages 3645–3650. Association for Computational Linguistics, 2019.

[50] S. Sun, Y. Cheng, Z. Gan, and J. Liu. Patient knowledge distillation for BERT model compression. In *EMNLP/IJCNLP*, pages 4322–4331, 2019.

[51] T. Tambe, C. Hooper, L. Pentecost, E. Yang, M. Donato, V. Sanh, A. M. Rush, D. Brooks, and G. Wei. Edgebert: Optimizing on-chip inference for multi-task NLP. *CoRR*, abs/2011.14203, 2020.

[52] J. Thorne, A. Vlachos, O. Cocarascu, C. Christodoulopoulos, and A. Mittal. The fact extraction and verification (FEVER) shared task. *CoRR*, abs/1811.10971, 2018.

[53] L. Tu, G. Lalwani, S. Gella, and H. He. An empirical study on robustness to spurious correlations using pre-trained language models. *Trans. Assoc. Comput. Linguistics*, 8:621–633, 2020.

[54] P. A. Utama, N. S. Moosavi, and I. Gurevych. Mind the trade-off: Debiasing NLU models without degrading the in-distribution performance. In *ACL*, pages 8717–8729. Association for Computational Linguistics, 2020.

[55] P. A. Utama, N. S. Moosavi, and I. Gurevych. Towards debiasing NLU models from unknown biases. In *EMNLP*, pages 7597–7610. Association for Computational Linguistics, 2020.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[57] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122. Association for Computational Linguistics, 2018.

[58] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. 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, Oct. 2020.

[59] C. Xu, W. Zhou, T. Ge, K. Xu, J. J. McAuley, and F. Wei. Beyond preserved accuracy: Evaluating loyalty and robustness of BERT compression. In *EMNLP (1)*, pages 10653–10659. Association for Computational Linguistics, 2021.

[60] S. Ye, X. Lin, K. Xu, S. Liu, H. Cheng, J. Lambrechts, H. Zhang, A. Zhou, K. Ma, and Y. Wang. Adversarial robustness vs. model compression, or both? In *ICCV*, pages 111–120. IEEE, 2019.

[61] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat. Q8BERT: quantized 8bit BERT. In *EMC2@NeurIPS*, pages 36–39. IEEE, 2019.

[62] D. Zhang, K. Ahuja, Y. Xu, Y. Wang, and A. C. Courville. Can subnetwork structure be the key to out-of-distribution generalization? In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 12356–12367. PMLR, 2021.

[63] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang. A systematic DNN weight pruning framework using alternating direction method of multipliers. In *ECCV (8)*, volume 11212 of *Lecture Notes in Computer Science*, pages 191–207. Springer, 2018.

[64] W. Zhang, L. Hou, Y. Yin, L. Shang, X. Chen, X. Jiang, and Q. Liu. Ternarybert: Distillation-aware ultra-low bit BERT. In *EMNLP*, pages 509–521. Association for Computational Linguistics, 2020.

[65] Y. Zhang, J. Baldridge, and L. He. PAWS: paraphrase adversaries from word scrambling. In *NAACL-HLT*, pages 1298–1308. Association for Computational Linguistics, 2019.

[66] M. Zhao, T. Lin, F. Mi, M. Jaggi, and H. Schütze. Masking as an efficient alternative to finetuning for pretrained language models. In *EMNLP*, pages 2226–2241, 2020.

[67] M. Zhu and S. Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *ICLR (Workshop)*. OpenReview.net, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Currently, we think there are no apparent negative societal impacts related to our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will release the codes and reproduction instructions upon publication.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.1 and Appendix B.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See all the figures of our experiments.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Section 4.1.

(b) Did you mention the license of the assets? [Yes] Licenses of some dataset we used are mentioned in Section 4.1. However, for the other datasets, we were unable to find the licenses.

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A More Information of Pruning and Debiasing Methods

A.1 Pruning Methods

A.1.1 Iterative Magnitude Pruning

Algo. 1 summarizes our implementation of IMP and IMP with weight rewinding. In practice, we set the per time pruning ratio $\Delta s = 10\%$ and the pruning interval $\Delta t = 0.1 \cdot t_{\text{max}}$.

A.1.2 Mask Training

As we described in Section 3.2.2 of the main paper, we realize mask training via binarization in forward pass and gradient estimation in backward pass. Following [42, 32], we adopt a magnitude-based strategy to initialize the real-valued masks. Specially, we consider two variants: The first one (hard variant) identifies the weights in matrix $\mathbf{W}$ with the smallest magnitudes, and sets the corresponding elements in $\hat{\mathbf{m}}$ to zero, and the remaining elements to a fixed value:

$$\hat{\mathbf{m}}_{i,j} = \begin{cases} 0 & \text{if } \mathbf{W}_{i,j} \in \text{Min}_s(\text{abs}(\mathbf{W})) \\ \alpha \times \phi & \text{otherwise} \end{cases} \tag{5}$$

where $\text{Min}_s(\text{abs}(\mathbf{W}))$ extracts the weights with the lowest absolute value, according to sparsity level s. $\alpha \geq 1$ is a hyper-parameter. The second one (soft variant) directly utilizes the absolute values of the weights for mask initialization:

$$\hat{\mathbf{m}}_{i,j} = \text{abs}(\mathbf{W}_{i,j}) \tag{6}$$

To control the sparsity of the model, the threshold ϕ is adjusted dynamically at a frequency of Δt_ϕ training steps. In practice, we control the sparsity in a local way, i.e., all the weight matrices $\mathbf{W} \in \boldsymbol{\theta}_{pr}$ should satisfy the same sparsity constraint s. Algo. 2 summarizes the entire process of mask training.

Algorithm 1: Iterative Magnitude Pruning (+ weight rewinding)

Input: PLM $f(\boldsymbol{\theta}_0)$ w. $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{ft}$, maximum training steps t_{max}, pruning interval Δt, per time pruning ratio Δs, target sparsity level $s = k \cdot \Delta s$ ($k \in \{1, 2, \cdots\}$), pruning method $p \in \{\text{imp}, \text{imp-rw}\}$

Output: Pruned subentwork $f(\mathbf{m} \odot \boldsymbol{\theta}^{'}_{ft})$

Initialize the pruning mask $\mathbf{m} = 1^{|\boldsymbol{\theta}_0|}$ and the number of pruning $n = 0$
while $t < t_{max}$ **do**
  if *(t mod* Δt*)* == 0 **then**
    `# For imp, return the subnetwork after some further training`
    if $n \cdot \Delta s$ == s *and* p==imp **then**
      return $f(\mathbf{m} \odot \boldsymbol{\theta}_t)$
    end
    Prune $\Delta s \cdot |\boldsymbol{\theta}_0|$ from the remaining parameters $\mathbf{m} \odot \boldsymbol{\theta}_t$ based on the magnitudes, and update $\mathbf{m}$ accordingly
    $n \leftarrow n + 1$
    `# For imp-rw, return the subnetwork directly after pruning`
    if $n \cdot \Delta s$ == s *and* p==imp-rw **then**
      return $f(\mathbf{m} \odot \boldsymbol{\theta}_0)$
    end
  end
  Update the remaining model parameters $\mathbf{m} \odot \boldsymbol{\theta}_t$ via AdamW [33];
end

A.2 Debiasing Methods

We have introduced the PoE method in Section 3.3. Here we provide descriptions of the other two debiasing methods, i.e., example reweighting and confidence regularization.

Example Reweighting directly assigns an importance weight to the standard CE training loss, according to the bias degree β:

$$\mathcal{L}_{\text{reweight}} = -\left(1-\beta\right) \mathbf{y} \cdot \log \mathbf{p}_m \tag{7}$$

Confidence Regularization is based on knowledge distillation [22]. It involves a teacher model trained with the standard CE loss. The teacher model's prediction $\mathbf{p}_t$ is used as a supervision signal to train the main model. To account for the bias degree of training examples, $\mathbf{p}_t$ is smoothed using a scaling function $\mathrm{S}\left(\mathbf{p}_t, \beta\right)$, and the final loss is computed as:

$$\begin{aligned} \mathcal{L}_{\text{confreg}} &= -\mathrm{S}\left(\mathbf{p}_t, \beta\right) \cdot \log \mathbf{p}_m \\ \mathrm{S}\left(\mathbf{p}_t, \beta\right) &= \frac{(\mathbf{p}_t^j)^{(1-\beta)}}{\sum_{k=1}^{K} (\mathbf{p}_t^k)^{(1-\beta)}} \end{aligned} \tag{8}$$

B More Experimental Setups

B.1 Datasets and Evaluations

We utilize eight datasets from three NLU tasks. The statistics of different dataset splits are summarized in Tab. 1. If one dataset has a test set, we use it for evaluation, and otherwise we report results on the dev set. For MNLI and QQP, since the official test server [5] only allows two submissions a day, we instead evaluate on the dev sets, following [4, 32, 45]. For FEVER, we use the training and evaluation data processed by [46] [6].

Tab. 2 shows the distribution of examples over classes. We can see that the distributions of the QQP and PAWS$_{qqp}$ evaluation sets are imbalanced. Specially, in the OOD PAWS$_{qqp}$, where a biased model

[5] `https://gluebenchmark.com/`

[6] `https://github.com/TalSchuster/FeverSymmetric`

Algorithm 2: Mask Training

Input: PLM $f(\boldsymbol{\theta}_0)$ w. $\boldsymbol{\theta}_0 \in \{\boldsymbol{\theta}_{pt}, \boldsymbol{\theta}_{ft}\}$, maximum training steps t_{max}, frequency Δt_ϕ, target sparsity level s, threshold ϕ, hyper-parameter α, initialization method $init \in \{\text{hard}, \text{soft}\}$

Output: Pruned subentwork $f(\mathbf{m} \odot \boldsymbol{\theta}_0)$

if *init* == hard **then**
  Initialize the real-valued mask $\hat{\mathbf{m}}$ according to Eq. 5
  Set threshold $\phi = 0.01$
else
  Initialize the real-valued mask $\hat{\mathbf{m}}$ according to Eq. 6
  Set threshold ϕ according to the sparsity constraint
end
while $t < t_{max}$ **do**
  Get a mini-batch of B examples $\{(\mathbf{x}_b, y_b)\}_{b=1}^{B}$
  Forward pass through binarization:
    $\mathcal{L}(f(\mathbf{x}_b, \mathbf{m} \odot \boldsymbol{\theta}_0), y_b)$, where $\mathbf{m}_{i,j} = \begin{cases} 1 & \text{if } \hat{\mathbf{m}}_{i,j} \geq \phi \\ 0 & \text{otherwise} \end{cases}$
  Backward pass through gradient estimation:
    $\hat{\mathbf{m}} \leftarrow \hat{\mathbf{m}} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{m}}$
  if *(t mod* Δt_ϕ*)* == 0 **then**
    Update the threshold ϕ to satisfy the sparsity constraint
  end
end
return $f(\mathbf{m} \odot \boldsymbol{\theta}_0)$

Table 1: The number of examples in different dataset splits. The splits used for evaluation are highlighted with red color. The dev set for MNLI is MNLI-m.

	NLI		Paraphrase Identification			Fact Verification		
	MNLI	HANS	QQP	PAWS-qqp	PAWS-wiki	FEVER	FEVER-Symm1	FEVER-Symm2
Train	392,702	30,000	363,849	11,988	49,401	242,911	-	-
Dev	9,815	30,000	40,432	677	8,000	16,664	-	708
Test	-	-	-	-	8,000	-	717	712

tends to predict most examples to the *duplicate* class, simply classifying all examples as *non-duplicate* can achieve substantial improvement in accuracy (from 28.2% to 71.8%). To account for this, we use the F1 score to evaluate the performance on the three paraphrase identification datasets. Specifically, we calculate the weighted average of the F1 score of each class. However, the class imbalance may still affect the evaluation on PAWS (as we discussed in Section 4.2.2) and therefore the OOD improvement should be assessed by also considering the ID performance.

B.2 Software and Computational Resources

We use two types of GPU, i.e., Nvidia V100 and TITAN RTX. All the experiments are run on a single GPU. Our codes are based on the Pytorch[7] and the huggingface transformers library[8] [58].

B.3 Training Details

B.3.1 Bias Model

As mentioned in Section 4.1.3, we train the bias model with spurious features. For MNLI and QQP, we adopt the hand-crafted word overlapping features proposed by [5], which includes:

- Whether all the hypothesis words also belong to the premise.

[7] https://pytorch.org/

[8] https://github.com/huggingface/transformers

Table 2: Data distribution over classes. The meaning of the abbreviations are: ent (entailment), cont (contradiction), dulp (duplicate), supp (support), not-info (not-enough-info). "Eval" represents the dataset split used for evaluation, as described in Tab. 1

		MNLI	HANS
Train	ent	33.3%	50%
	cont	33.3%	50%
	neutral	33.3%	0%
Eval	ent	35.4%	50%
	cont	32.7%	50%
	neutral	31.8%	0%

		QQP	PAWS$_{qqp}$	PAWS$_{wiki}$
Train	dulp	36.9%	31.5%	44.2%
	non-dulp	63.1%	68.5%	55.8%
Eval	dulp	36.8%	28.2%	44.2%
	non-dulp	63.2%	71.8%	55.8%

		FEVER	Symm1	Symm2
Train	supp	41.4%	-	-
	refute	17.2%	-	-
	not-info	41.4%	-	-
Eval	supp	47.9%	52.9%	50%
	refute	52.1%	47.1%	50%
	not-info	0%	0%	0%

Table 3: Basic training hyper-parameters.

	#Epoch	Learning Rate	Batch Size	Max Length	Eval Interval	Eval Metric	Optimizer
MNLI	3 or 5	5e-5	32	128	1,000	Acc	AdamW
QQP	3	2e-5	32	128	1,000	F1	AdamW
FEVER	3	2e-5	32	128	500	Acc	AdamW

- Whether the hypothesis appears as a continuous subsequence in the premise.
- The percentage of the hypothesis words $\mathbf{w}^h = \{\mathbf{w}^h_1, \mathbf{w}^h_2, \cdots, \mathbf{w}^h_{|\mathbf{w}^h|}\}$ that appear in the premise $\mathbf{w}^p = \{\mathbf{w}^p_1, \mathbf{w}^p_2, \cdots, \mathbf{w}^p_{|\mathbf{w}^p|}\}$. Formally $\frac{|\mathbf{w}^h \cap \mathbf{w}^p|}{|\mathbf{w}^h|}$.
- The average of the maximum similarity between each hypothesis word and all the premise words: $\frac{1}{|\mathbf{w}^h|}\,\mathrm{sum}(\{\max(\{\mathrm{sim}(\mathbf{w}^p_i, \mathbf{w}^h_j)|\forall \mathbf{w}^p_j \in \mathbf{w}^p\})|\forall \mathbf{w}^h_i \in \mathbf{w}^h\})$, where the similarity is computed based on the fastText word vectors [39] and the cosine distance.
- The minimum of the same similarities above: $\min(\{\max(\{\mathrm{sim}(\mathbf{w}^p_i, \mathbf{w}^h_j)|\forall \mathbf{w}^p_j \in \mathbf{w}^p\})|\forall \mathbf{w}^h_i \in \mathbf{w}^h\})$.

For FEVER, we use the max-pooled word embeddings of the claim sentence, which are also based on the fastText word vectors.

B.3.2 Full BERT

The main training hyper-parameters are shown in Tab. 3, which basically follow [55]. Most of the hyper-parameters are the same for different training strategies, except for the number of training epochs (#Epoch) on MNLI. For the standard CE loss and example reweighting, the model is trained for 3 epochs. For PoE and confidence regularization, the model is trained for 5 epochs.

B.3.3 Mask Training and IMP

Mask training and IMP basically use the same set of hyper-parameters as full BERT, except for longer training. The number of training epochs for mask training and IMP is 5 on MNLI, and 7 on QQP and FEVER. The hyper-parameters specific to mask training or IMP are summarized in Tab. 4. Unless otherwise specified, we adopt the hard-variant of mask initialization (Eq. 5) and fix the subnetwork sparsity to target sparsity s throughout the process of mask training. Some special experimental setups are described as follows:

Subnetworks from Fine-tuned BERT When we search for subnetworks at low sparsity (e.g., 20%) from a fine-tuned BERT, we find that mask training (with debiasing loss) stably improves the OOD performance, while the ID performance peaks at an early point of training and then slightly drops and recovers later. Therefore, the ID performance favors the early checkpoints, which are not good at the OOD generalization. To address this problem, we select the best checkpoint after $0.7 \cdot t_{\text{max}}$ of training, but still according to the performance on the ID dev set. This strategy is only adopted for mask training on fine-tuned BERT (for all sparsity levels), and in other cases we select the best checkpoint across training based on ID performance.

Table 4: Basic hyper-parameters related to pruning methods. t_{max} is the number of optimization steps by training #Epoch epochs.

Mask Training					IMP	
Mask Init	Sparsity Schedule	ϕ	α	Δt_ϕ	Δs	Δt
magnitude (hard)	fixed to s	0.01	2	equal to Eval Interval	10%	$0.1 \cdot t_{\text{max}}$

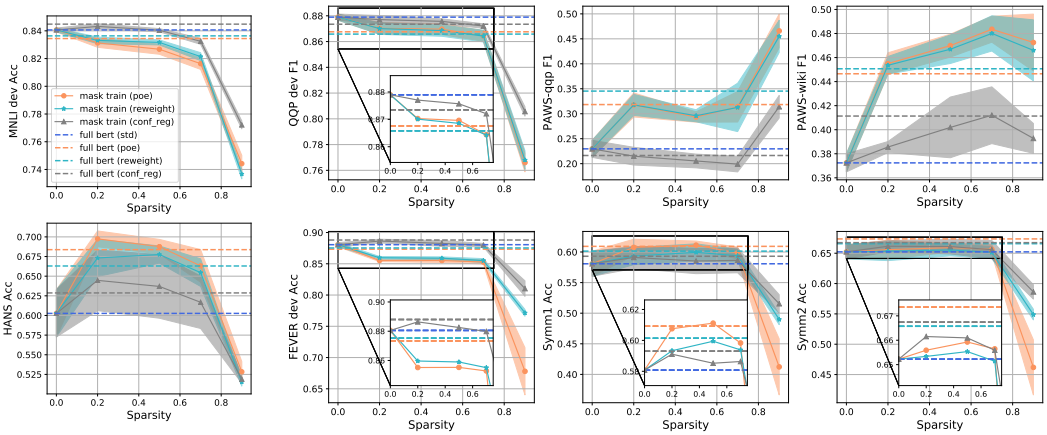


Figure 9: Results of subnetworks pruned from the CE fine-tuned BERT, with different debiasing methods in pruning.

BERT Subnetworks Fine-tuned in Isolation When fine-tuning the searched subnetworks (with their weights rewound to pre-trained values) in isolation, we use the same set of hyper-parameters as full BERT fine-tuning.

Sparse and Unbiased BERT Subnetworks The OOD data is used in this setup. Specifically, we utilize the training data of HANS and PAWS for NLI and paraphrase identification respectively. In terms of the FEVER-Symmetric dataset, which does not provide a training set (see Tab. 1), we use the dev set of FEVER-Symm2 and copy the data 10 times to construct the OOD training data. The OOD and ID training data are then combined to form the final training set. Note that the evaluation sets are the same as the other setups, and **NO** test data is used in mask training.

Gradual Sparsity Increase We mainly experiment with the gradual sparsity increase schedule for subnetworks at 90% sparsity. Concretely, we increase the sparsity from 70% to 90% during the process of mask training. The real-valued mask is initialized using the soft-variant (Eq. 6). This is because we find that the hard-variant is difficult to optimize with sparsity increase.

C More Results and Analysis

C.1 More Debiasing Methods

In Section 4, we mainly experiment with the PoE debiasing method. Here, we combine mask training with the other two debiasing methods, namely example reweighting and confidence regularization, and search for SRNets from the CE fine-tuned BERT. Fig. 9 presents the results. As we can see: (1) Pruning with different debiasing methods almost consistently improves the OOD performance over the CE fine-tuned BERT. (2) The confidence regularization method (the grey lines) only achieves mild OOD improvement over the full BERT, while it preserves more ID performance compared with the other two methods. This phenomenon is in accordance with the results from [54], which propose the confidence regularization method to achieve a better trade-off between the ID and OOD performance.

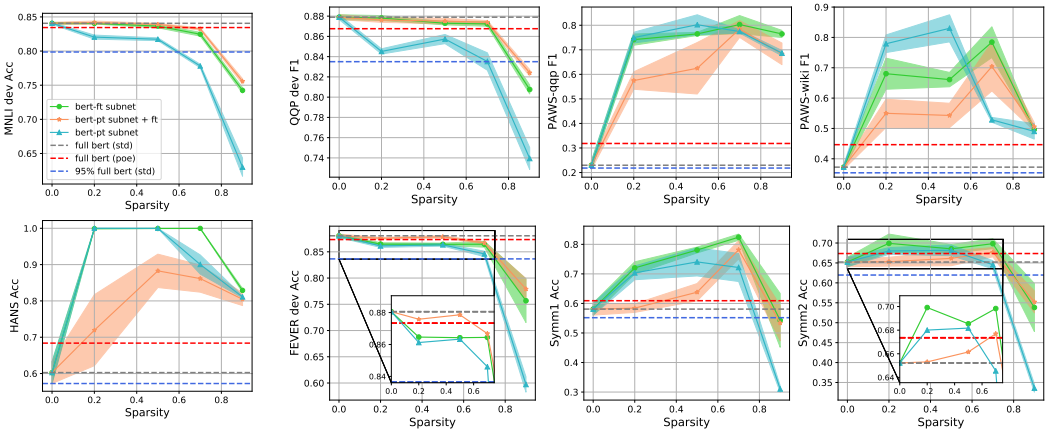


Figure 10: Results of subnetworks found using the OOD information.

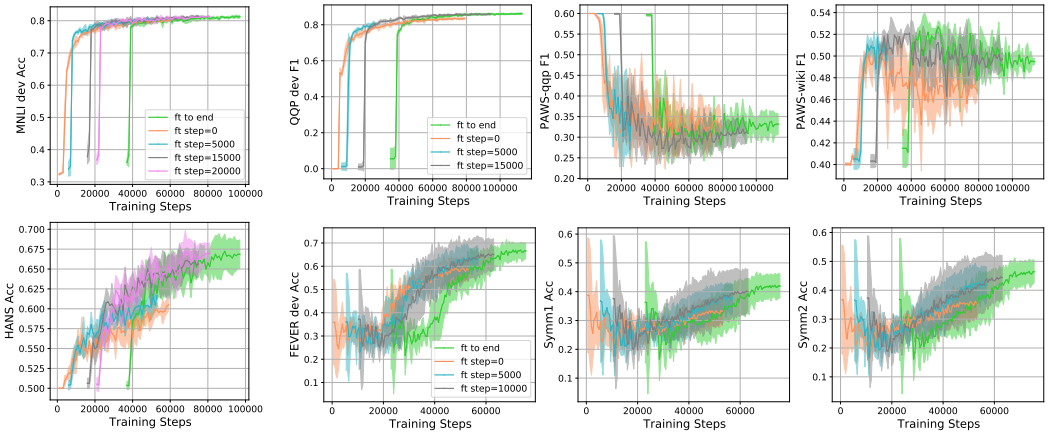


Figure 11: Mask training curves starting from full BERT checkpoints fine-tuned for varied steps. The sparsity levels are 70%, 70% and 90% for MNLI, QQP and FEVER respectively. At these sparsity levels, the gap between "ft step=0" and "ft to end" is the largest, according to Fig. 5.

C.2 Sparse and Unbiased Subnetworks

Fig. 10 shows the results of mask training with the OOD training data. We can see that the general patterns in paraphrase identification and fact verification datasets are basically the same as the NLI datasets. Although the identified subnetworks cannot achieve 100% accuracy on PAWS and FEVER-Symmetric as on HANS, they substantially narrow the gap between OOD and ID performance, as compared with the full BERT. An exception is on the Symm2, where the upper bound of SRNets seems not very high. This is probably because we do not have enough examples (708 in total) to represent the data distribution of the FEVER-Symmetric dataset. Therefore, we conjecture that the existence of sparse and unbiased subnetworks might be ubiquitous.

C.3 The Timing to Start Searching SRNets

Fig. 11 shows the mask training curves on all the 8 datasets. Similar to the NLI datasets, mask training on the other two tasks can achieve comparable results as "ft to end" by starting from an intermediate checkpoint of BERT fine-tuning. For QQP, we can start from 15,000 steps of full BERT fine-tuning (44% of t_{max}). For FEVER, we can start from 10,000 steps (44% of t_{max}).

Table 5: Ablation studies of the gradual sparsity increase schedule. The number of training epochs are 3, 5 and 5 for MNLI, QQP and FEVER respectively. The subnetworks are at 90% sparsity. The numbers in the subscripts are standard deviations.

		MNLI	HANS
fixed	hard	$72.09_{0.92}$	$52.56_{0.92}$
	soft	$72.63_{0.31}$	$52.82_{0.47}$
gradual	0.2∼0.9	$73.61_{0.28}$	$53.90_{0.87}$
	0.5∼0.9	$75.06_{0.31}$	$54.99_{1.28}$
	0.7∼0.9	$\mathbf{76.84}_{0.46}$	$\mathbf{56.72}_{0.75}$

		QQP	PAWS_{qqp}	PAWS_{qqp}
fixed	hard	$71.64_{1.85}$	$\mathbf{55.70}_{1.92}$	$49.59_{1.84}$
	soft	$77.08_{0.66}$	$46.48_{3.55}$	$49.38_{0.98}$
gradual	0.2∼0.9	$75.79_{0.39}$	$51.57_{0.69}$	$47.94_{0.98}$
	0.5∼0.9	$77.54_{0.47}$	$50.92_{0.97}$	$48.86_{0.89}$
	0.7∼0.9	$\mathbf{79.49}_{0.58}$	$46.59_{1.81}$	$\mathbf{51.15}_{0.73}$

		FEVER	Symm1	Symm2
fixed	hard	$49.56_{5.09}$	$27.45_{2.94}$	$29.75_{4.40}$
	soft	$72.80_{0.95}$	$46.67_{0.73}$	$52.33_{0.75}$
gradual	0.2∼0.9	$73.53_{1.36}$	$46.47_{1.66}$	$52.42_{1.39}$
	0.5∼0.9	$77.01_{0.43}$	$49.87_{0.95}$	$56.57_{0.22}$
	0.7∼0.9	$\mathbf{79.01}_{0.68}$	$\mathbf{51.74}_{0.71}$	$\mathbf{58.17}_{0.33}$

Table 6: Results of RoBERTa-base and BERT-large on the NLI task. We conduct mask training with PoE loss on the standard fine-tuned PLMs. "0.5∼0.7" denotes gradual sparsity increase. The numbers in the subscripts are standard deviations.

RoBERTa-base		MNLI	HANS
full model	std	$87.14_{0.21}$	$68.33_{0.88}$
	poe	$86.56_{0.18}$	$76.15_{1.35}$
mask train	0.5	$85.40_{0.14}$	$75.17_{0.55}$
	0.7	$83.48_{0.29}$	$68.63_{1.33}$
	0.5∼0.7	$84.41_{0.15}$	$71.95_{1.23}$

BERT-large		MNLI	HANS
full model	std	$86.84_{0.13}$	$69.44_{2.39}$
	poe	$86.25_{0.17}$	$76.27_{1.55}$
mask train	0.5	$85.47_{0.28}$	$75.40_{0.64}$
	0.7	$77.54_{6.10}$	$60.19_{7.56}$
	0.5∼0.7	$84.83_{0.26}$	$70.18_{2.24}$

C.4 Ablation Studies on Gradual Sparsity Increase

As we mentioned in Appendix B.3.3, we increase the sparsity from 70% to 90% and adopt the soft variant of mask initialization. To explain the reason for using this specific strategy, we present the ablation study results in Tab. 5. We can observe that: (1) Replacing the hard variant of mask initialization with the soft variant is beneficial, which leads to obvious improvements on the QQP, FEVER, Symm1 and Symm2 datasets. (2) Gradually increasing the sparsity further promotes the performance, with the 0.7∼0.9 strategy achieving the best results on 7 out of the 8 datasets.

C.5 Results on RoBERTa-base and BERT-large

It has been shown by [21, 53] that pre-trained model RoBERTa [29] have better OOD generalization than BERT. [53] also shows that larger PLMs, which are more computationally expensive, are more robust. To examine whether our conclusions can generalize to RoBERTa and larger versions of BERT, we conduct mask training on the standard fine-tuned RoBERTa-base and BERT-large models and use the PoE debiasing loss in the mask training process.

The results are shown in Tab. 6. We can see that, for RoBERTa-base: (1) At 50% sparsity, the searched subnetworks outperform the full RoBERTa (std) by 6.84 points on HANS, with a relative small drop of 1.74 on MNLI, validating that SRNets can be found in RoBERTa. (2) At 70% sparsity, the vanilla mask training produces subnetworks with undesirable ID performance and OOD performance comparable to full model (std). In comparison, when we gradually increase the sparsity level from 50% to 70%, the ID and OOD performance are improved simultaneously, demonstrating that gradual sparsity increase is also effective for RoBERTa.

When it comes to BERT-large, the conclusions are basically the same as BERT-base and RoBERTa-base: (1) We can find 50% sparse SRNets from BERT-large using the original mask training. (2) Gradual sparsity increase is also effective for BERT-large. Additionally, we find that the original mask training exhibits high variance at 70% sparsity because the training fails for some random seeds. In comparison, with gradual sparsity increase, the searched subnetworks have better performance and low variance.

D Related Work on Model Compression and Robustness

Some prior attempts have also been made to obtain compact and robust deep neural networks. We discuss the relationship and difference between these works and our paper from three perspectives:

Robustness Types There are various types of model robustness, including generalization to in-distribution unseen examples, robustness towards dataset bias [2, 37, 65, 46] and adversarial attacks [14], etc. Among the researches on model compression and robustness, adversarial robustness [16, 60, 48, 10, 59] and dataset bias robustness [62, 8] are the most widely studied. In this paper, we focus on the dataset bias problem, which is more common than the worst-case adversarial attack, in terms of real-world application.

Compression Methods A major direction in robust model compression is about the design of compression methods. [47] investigate the effect of magnitude-based pruning on adversarially trained models. [16, 60] treat sparsity and adversarial robustness as a constrained optimization problem, and solve it using the alternating direction method of multipliers (ADMM) framework [63]. [48, 62, 34] combine learnable weight mask (i.e., mask training) and robust training objectives. Our study investigates the use of magnitude-based pruning and mask training, which are also widely employed in the literature of BERT compression.

Application Fields Despite the topic of model compression and robustness has been proposed for years, it is mostly studied in the context of computer vision (CV) tasks and models, and few attention has been paid to the NLP field. Considering the real-world application potential of PLMs, it is critical to study the questions of PLM compression and robustness jointly. To this end, some recent studies extend the evaluation of compressed PLMs to consider adversarial robustness [59] and dataset bias robustness [8].

Although our work shares the same topic with [8], we differ in several aspects. First, the scope and focus of our research questions are different. They aim at analyzing the impact of different compression methods (pruning and knowledge distillation [22]) on the OOD robustness of standard fine-tuned BERT. By contrast, we focus on subnetworks obtained from different pruning and fine-tuning paradigms and consider both standard fine-tuning and debiasing fine-tuning. Second, our conclusions are different. The results of [8] suggest that pruning generally has a negative impact on the robustness of BERT. In comparison, we revel the consistent existence of sparse BERT subnetworks that are more robust to dataset bias than the full model.

E More Discussions

E.1 How to Predict the Timing to Start Searching SRNets?

A feasible way of solution is to stop full BERT fine-tuning when there is no significant improvement across several consecutive evaluation steps. The patience of early-stopping can be determined based on the computational budget. If our resource is limited, we can at least directly training the mask on $\boldsymbol{\theta}_{pt}$, which can still produce SRNets at 50% sparsity (as shown by Section 4.4.2).

E.2 How to Generalize to Other Scenarios?

In this work, we focus on NLU tasks and PLMs from the BERT family. However, the methodology we utilize is agnostic to the type of bias, task and backbone model. Theoretically, it can be flexibly adapted to other scenarios by simply change the spurious features to train the bias model (for the three debiasing methods considered in this paper) or combine the pruning method with another kind of debiasing method that also involves model training. In the future work, we would like to extend our exploration to other types of PLMs (e.g., language generation models like GPT [41] and T5 [43]) and other types of NLP tasks (e.g., dialogue generation).