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

Anonymous Author(s)
Affiliation
Address
email

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 refine the SRNets searching process in terms of efficiency and performance, which involves: 1) the appropriate timing to start searching SRNets during full BERT fine-tuning, and 2) how to identify SRNets at high sparsity. Our codes will be released on publication.

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 [6, 25]. On the other hand, despite being pre-trained on large-scale corpus, PLMs still tend to rely on *dataset bias* [8, 18, 31, 24], i.e., the spurious features of input examples that strongly correlate with the 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. [13, 19, 15] compress the fine-tuned PLMs in a post-hoc fashion. [2, 20, 16, 14] extend the *Lottery Ticket Hypothesis* (LTH) [5] to search PLMs subnetworks that can be fine-tuned in isolation. Taking one step further, [32] propose to learn task-specific subnetwork structures via mask training [11, 17], 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.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

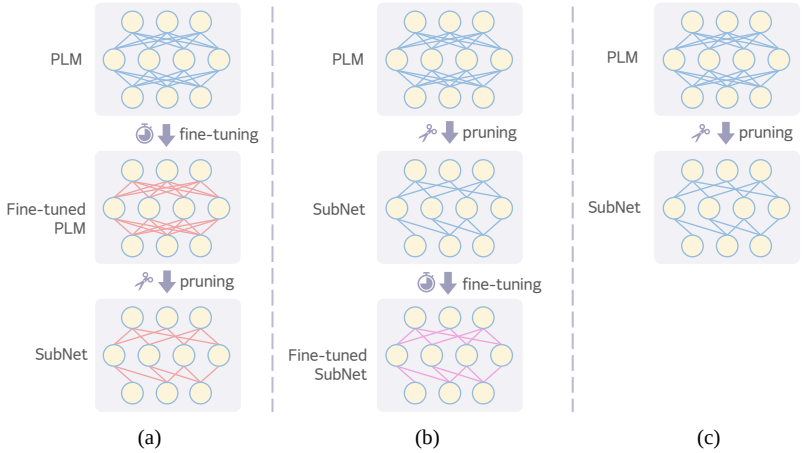


Figure 1: Three kinds of PLM subnetworks obtained from different pruning and fine-tuning paradigms. (a) Pruning a fine-tuned PLM. (b) Pruning the PLM and then fine-tuning the subnetwork. (c) Pruning the PLM without fine-tuning model parameters. The obtained subnetworks are used for testing.

To address the dataset bias problem, numerous debiasing methods have been proposed. A prevailing category of debiasing methods [3, 27, 12, 10, 24, 7, 28] 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 promising progress has been made in both directions, most existing work tackle the two problems independently. In this paper, 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 [4] on three natural language understanding (NLU) tasks, i.e., natural language inference (NLI), paraphrase identification and fact verification. We consider a variety of setups including the three pruning and fine-tuning paradigms, standard and debiasing training objectives, different model compression methods, ID and OOD settings for evaluation. Our results show that **BERT do 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 some room for improvement, based on some observations from the above experiments. Accordingly, we refine the searching process from two aspects: First, we study the appropriate timing to start searching SRNets during full BERT fine-tuning, considering both subnetwork performance and training cost. Second, we ameliorate the mask training method to identify better SRNets at high sparsity, with lower searching cost.

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. By using the OOD information, we further reveal that there exist sparse and almost unbiased BERT subenetworks.
- We refine the SRNets searching process to improve its efficiency and the performance of the obtained subnetworks.

2 Preliminaries

2.1 BERT Architecture and Subnetworks

BERT is composed of an embedding layer, a stack of Transformer layers [29] 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.

2.2 Pruning Methods

In this work, we consider two typical pruning methods, i.e., magnitude-based pruning [9, 5] and mask training [17, 32, 16].

2.2.1 Magnitude-based Pruning

Magnitude-based pruning 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: (1) Training the full model to convergence. (2) Pruning a fraction of parameters with the smallest magnitude. (3) Re-training the pruned subnetwork. (4) Repeat (2)-(3) 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 (3), and (1) can be abandoned. More details about our IMP implementations can be found in Appendix A.

2.2.2 Mask Training

Mask training treats the pruning mask $\mathbf{m}$ as trainable parameters. Following [17, 32, 22, 16], 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* [1] 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 [22, 16], we initialize the real-valued masks according to the magnitude of the original weights. The complete mask training algorithm is summarized in Appendix A.

2.3 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 3.1.3 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) [3, 10, 12], *example reweighting* [24, 7] and *confidence regularization* [27]. In this work, we mainly experiment with the PoE method and the standard cross-entropy (CE) loss.

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 predicted distributions of the main model and the bias model, denoted as $\mathbf{p}_b$ and $\mathbf{p}_m$, into an ensemble distribution, and then computes the training loss as $\mathcal{L}_{\text{poe}} = -\mathbf{y} \cdot \log \text{softmax}\left(\log \mathbf{p}_m + \log \mathbf{p}_b\right)$.

2.4 Notations

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

- $\mathcal{A}^t_{\mathcal{L}}(f(\boldsymbol{\theta}))$: Training $f(\boldsymbol{\theta})$ with loss $\mathcal{L}$ for t steps, where t can be omitted for simplicity.
- $\mathcal{P}^p_{\mathcal{L}}(f(\boldsymbol{\theta}))$: 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 2.2.1. "mask" stands for mask training.
- $\mathcal{E}_d(f(\boldsymbol{\theta}))$: Evaluating $f(\boldsymbol{\theta})$ on the test data with distribution $d \in \{\text{ID}, \text{OOD}\}$.

3 Sparse and Robust BERT Subnetworks

3.1 Experimental Setups

3.1.1 Datasets and Evaluation

Natural Language Inference We use MNLI [30] 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 [18] is built so that such correlation does not hold.

Paraphrase Identification The ID dataset for paraphrase identification is QQP [1], 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 [31] 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 [2] [26] 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) [24] 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 information about the dataset statistics and evaluation details are summarized in Appendix B.

3.1.2 PLM Backbone

We experiment with the BERT-base-uncased model [4]. It has roughly 110M parameters in total, and 84M parameters in the Transformer layers. As described in Section 2.1, we derive the subnetworks from the Transformer layers and the reported sparsity levels are relative to the 84M parameters.

3.1.3 Training Details

Following [3], 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.

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

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

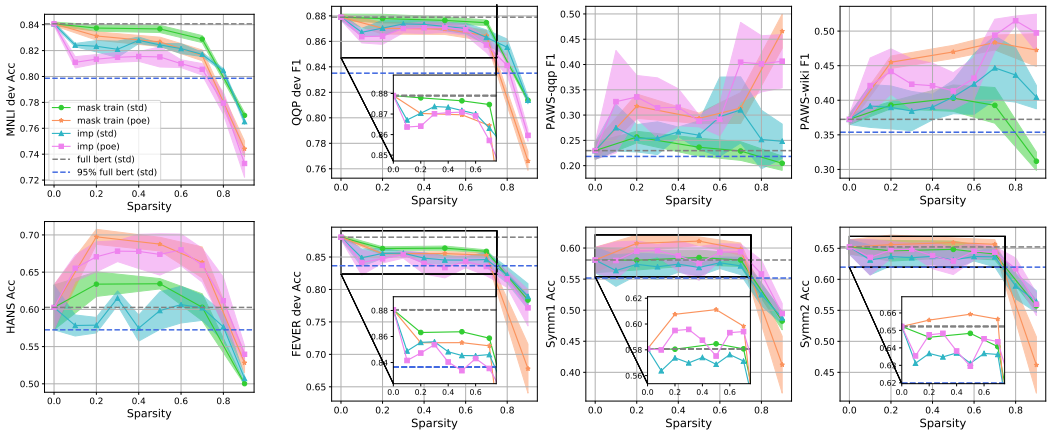


Figure 2: Results of subnetworks pruned from the CE fine-tuned BERT. "std" means standard, and the shadowed areas denote standard deviations, which also apply to the other figures of this paper.

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.

For full BERT fine-tuning, we follow the hyper-parameters suggested by [28]. The hyper-parameters for mask training and IMP are basically the same as full BERT, except for 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 performance on the ID dev set, 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.

3.2 Subnetworks from Fine-tuned BERT

3.2.1 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 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.

3.2.2 Results

Subnetworks from Standard Fine-tuned BERT The results are shown in Fig. 2. We discuss them from three perspectives. For the full BERT, we can see that standard 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

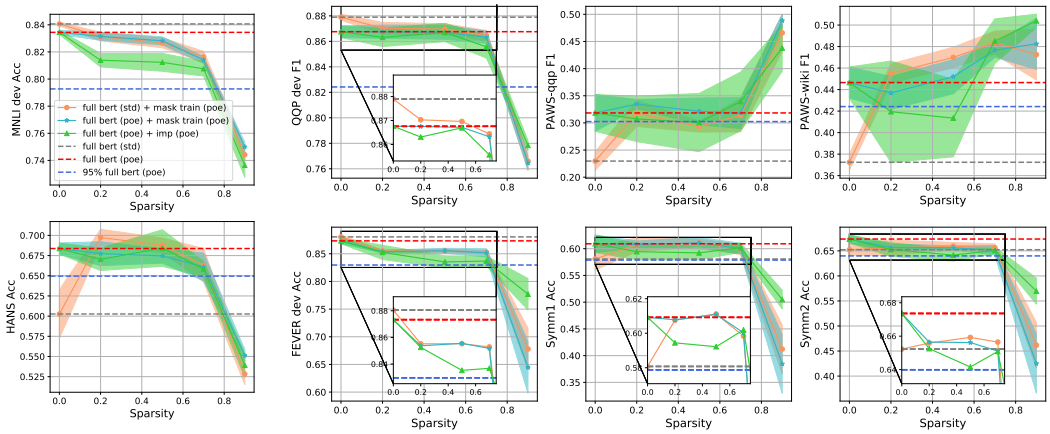


Figure 3: Results of subnetworks pruned from the PoE fine-tuned BERT. Results of the "mask train (poe)" subnetworks from Fig. 2 (the orange line) are also reported for reference.

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), 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.

3.3 BERT Subnetworks Fine-tuned in Isolation

3.3.1 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 $\mathcal{A}_{\mathcal{L}_1}(f(\mathbf{m} \odot \boldsymbol{\theta}_{pt}))$:

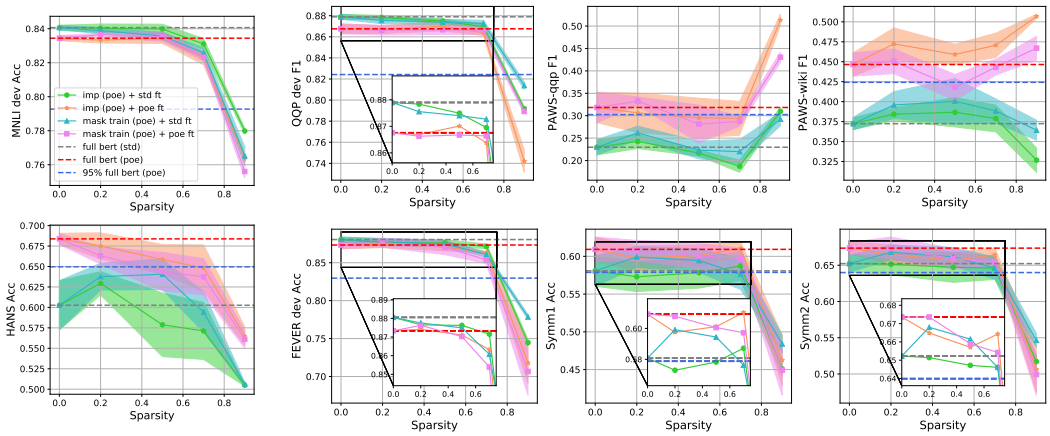


Figure 4: Results of BERT subnetworks fine-tuned in isolation. "ft" is short for fine-tuning.

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

Following the LTH [5], we solve this problem using the train-prune-rewind pipeline. For IMP, the procedure is described in Section 2.2.1 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., $\mathcal{L}_2 = \mathcal{L}_{\text{poe}}$).

3.3.2 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 the first two findings, we can extend the LTH on BERT [2, 20, 14, 16] to the OOD scenario: **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 evaluation settings.**

3.4 BERT Subnetworks Without Fine-tuning

3.4.1 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 [32], 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 $\mathcal{L} \in \{\mathcal{L}_{\text{std}}, \mathcal{L}_{\text{poe}}\}$.

3.4.2 Results

Fig. 5 shows the results of BERT subnetworks without fine-tuning. We can see that: 1) With CE-based mask training, the identified subnetworks (under 50% sparsity) in pre-trained BERT are competitive

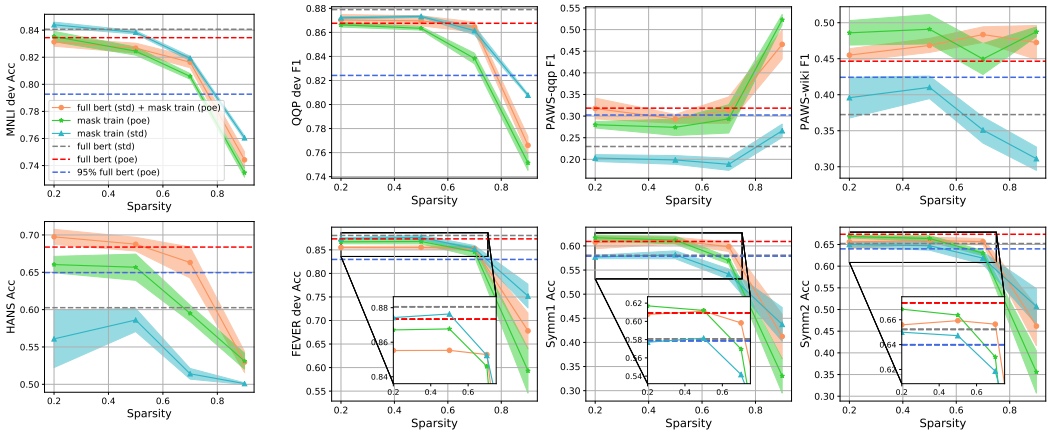


Figure 5: Results of BERT subnetworks without fine-tuning. Results of the "mask train (poe)" subnetworks from Fig. 2 (the orange line) are also reported for reference.

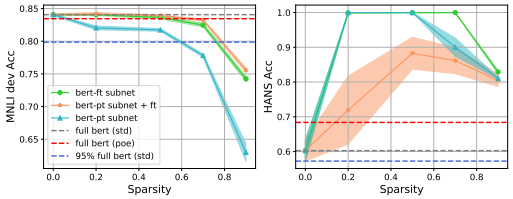


Figure 6: NLI results of BERT subnetworks found using the OOD information. Results of the other two tasks can be found in Appendix C.

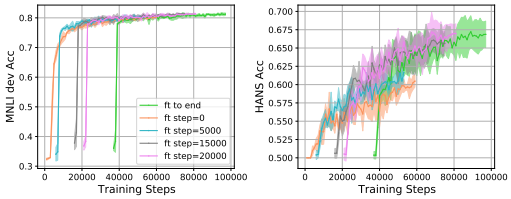


Figure 7: NLI mask training curves (70% sparse), starting from BERT fine-tuned for varied steps. See Appendix C for results of the other two tasks.

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.

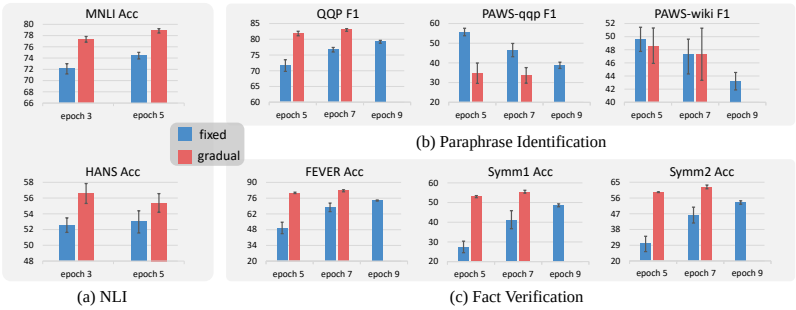
3.5 Sparse and Unbiased BERT Subnetworks

3.5.1 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 summarizes the detailed experimental setups.

3.5.2 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.



(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.

4 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 3.2 and Section 3.3.

4.1 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 3.4. 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.

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.

4.2 SRNets at High Sparsity

As the results of Section 3 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 [33] (relevant ablation studies are shown in Appendix C).

Fig. 8 compares the fixed sparsity used in the previous sections and the gradual sparsity increase schedule, across different 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 3.2.2.

5 Conclusion and Limitation

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 refine the SRNet searching process to improve the subnetwork performance and searching efficiency. However, our experiments are only based on the BERT model and NLU tasks. In the future work, we would like to extend our exploration to other types of PLMs (e.g., GPT [21] and T5 [23]) and other NLP tasks (e.g., language generation).

References

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

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

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

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

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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 5.
   (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 3.1 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 3.1.
   (b) Did you mention the license of the assets? [Yes] Licenses of some dataset we used are mentioned in Section 3.1. 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]