Rethinking Network Pruning—under the Pre-train and Fine-tune Paradigm

Dongkuan Xu[1] and **Ian En-Hsu Yen**[2] and **Jinxi Zhao**[2] and **Zhibin Xiao**[2]
[1]The Pennsylvania State University, State College, PA, USA
[2]Moffett AI, Los Altos, CA, USA
{dux19}@psu.edu, {ian.yan, jx.zhao, zb.xiao}@moffett.ai

Abstract

Transformer-based pre-trained language models have significantly improved the performance of various natural language processing (NLP) tasks in the recent years. While effective and prevalent, these models are usually prohibitively large for resource-limited deployment scenarios. A thread of research has thus been working on applying network pruning techniques under the pretrain-then-finetune paradigm widely adopted in NLP. However, the existing pruning results on benchmark transformers, such as BERT, are not as remarkable as the pruning results in the literature of convolutional neural networks (CNNs). In particular, common wisdom in pruning CNN states that sparse pruning technique compresses a model more than that obtained by reducing number of channels and layers (Elsen et al., 2020; Zhu and Gupta, 2017), while existing works on sparse pruning of BERT yields inferior results than its small-dense counterparts such as TinyBERT (Jiao et al., 2020). In this work, we aim to fill this gap by studying how knowledge are transferred and lost during the pre-train, fine-tune, and pruning process, and proposing a knowledge-aware sparse pruning process that achieves significantly superior results than existing literature. We show for the first time that sparse pruning compresses a BERT model significantly more than reducing its number of channels and layers. Experiments on multiple data sets of GLUE benchmark show that our method outperforms the leading competitors with a 20-times weight/FLOPs compression and neglectable loss in prediction accuracy[1].

1 Introduction

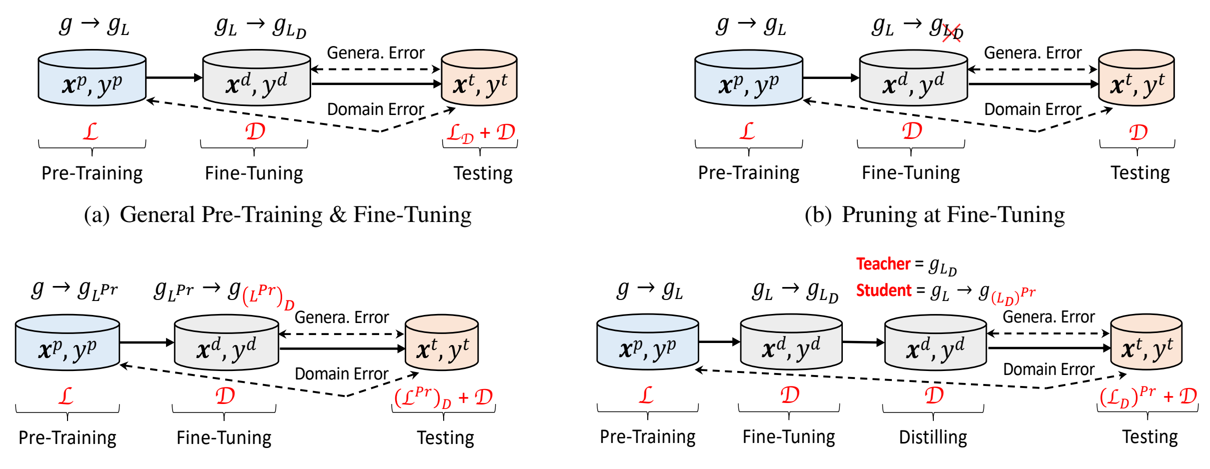
Pre-trained language models, such as BERT (Devlin et al., 2019), become the standard and effective methods for improving the performance of a variety of natural language processing (NLP) tasks. These models are pre-trained in a self-supervised fashion and then fine-tuned for supervised downstream tasks. However, these models suffer from the heavy model size, making them impractical for resource-limited deployment scenarios and incurring cost concerns (Strubell et al., 2019).

In parallel, an emerging subfield has studied the redundancy in deep neural network models (Zhu and Gupta, 2017; Gale et al., 2019) and proposed to prune networks without sacrificing performance, such as the lottery ticket hypothesis (Frankle and Carbin, 2019). Common wisdom in CNN literature shows that sparse pruning leads to more compression rate than structural pruning. For example, for the same number of parameters (0.46M), the sparse MobileNets improve by 11.2% accuracy over the dense ones (Zhu and Gupta, 2017). However, similar conclusions are not observed for pre-trained language models.

The main question this paper attempts to answer is: how to perform sparse pruning under the pre-train and fine-tune paradigm? Answering this question correctly is challenging. First, these models adopt pre-training and fine-tuning procedures, during which the general-purpose language knowledge and the task-specific knowledge are learned respectively. Thus, it is desirable and challenging to keep the weights that are important to both knowledge during pruning. Second, unlike CNNs, pre-trained language models have a complex architecture consisting of embedding, self-attention, and feed-forward layers.

To address these challenges, we propose SparseBERT, a knowledge-aware sparse pruning method for pre-trained language models, with a special focus on the widely used BERT model. SparseBERT is executed in the fine-tuning stage. It preserves both general-purpose and task-specific language knowledge while pruning. To preserve the general-purpose knowledge learned during pre-training,

[1]Codes can be found in the authors' website. Work done during an internship at Moffett AI.



$g \rightarrow g_L$
$g_L \rightarrow g_{L_D}$
x^p, y^p
x^d, y^d
x^t, y^t
Genera. Error
Domain Error
$\mathcal{L}$
$\mathcal{D}$
$\mathcal{L}_D + \mathcal{D}$
Pre-Training
Fine-Tuning
Testing
$g_L \rightarrow \cancel{g_{L_D}}$
$g \rightarrow g_{L^{Pr}}$
$g_{L^{Pr}} \rightarrow g_{(L^{Pr})_D}$
$(\mathcal{L}^{Pr})_D + \mathcal{D}$
Teacher = g_{L_D}
Student = $g_L \rightarrow g_{(L_D)^{Pr}}$
Distilling
$(\mathcal{L}_D)^{Pr} + \mathcal{D}$

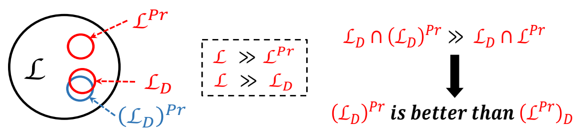
(a) General Pre-Training & Fine-Tuning

(b) Pruning at Fine-Tuning

(c) Pruning at Pre-training

(d) Pruning at Distilling (Proposed)

Figure 1: How knowledge is transferred under different pruning strategies. (a) is the general pre-training and fine-tuning procedure (Section 3.1). g is an encoder. g_L and g_{L_D} are the encoders well-trained on the pre-training and fine-tuning datasets respectively. $\mathcal{L}$ and $\mathcal{D}$ are the general-purpose language knowledge and the task-specific knowledge respectively. There is a domain error between pre-training and testing, and a generalization error between fine-tuning and testing. (b) and (c) are two basic pruning strategies (Section 3.2.1). Both $\mathcal{L}_D$ and $\mathcal{L}^{pr}$ are subsets of knowledge $\mathcal{L}$. $\mathcal{L}_D$ is related to the downstream task. $\mathcal{L}^{pr}$ is preserved in a pruned encoder $g_{L^{pr}}$. (d) is the proposed pruning strategy (Sections 3.2.2-3.2.3). $(\mathcal{L}^{pr})_D$ refers to the knowledge obtained by first pruning and then fine-tuning. $(\mathcal{L}_D)^{pr}$ corresponds to first fine-tuning and then pruning while distilling.



$\mathcal{L}$
$\mathcal{L}^{Pr}$
$\mathcal{L}_D$
$(\mathcal{L}_D)^{Pr}$
$\mathcal{L} \gg \mathcal{L}^{Pr}$
$\mathcal{L} \gg \mathcal{L}_D$
$\mathcal{L}_D \cap (\mathcal{L}_D)^{Pr} \gg \mathcal{L}_D \cap \mathcal{L}^{Pr}$
$(\mathcal{L}_D)^{Pr}$ is better than $(\mathcal{L}^{Pr})_D$

(a) Knowledge Relationship

(b) Why Distilling

Figure 2: Knowledge Analysis.

SparseBERT uses the pre-trained BERT without fine-tuning as the initialized model and prunes the linear transformations in self-attention and feed-forward layers, which is inspired by the recent findings that self-attention and feed-forward layers are overparameterized (Michel et al., 2019; Voita et al., 2019) and are also the most computation consumption parts (Ganesh et al., 2020). To learn the task-specific task knowledge during pruning while preserving the general-purpose knowledge at the same time, we apply knowledge distillation (Hinton et al., 2015). We adopt the task-specific fine-tuned BERT as the teacher network and the pre-trained BERT that is being pruned as the student. We feed the downstream task data into the teacher-student framework to train the student to reproduce the behaviors of the teacher.

We summarize different types of BERT pruning approaches in Figure 1 (see Section 3.2 for detailed discussion) Experimental results on the GLUE benchmark demonstrate that SparseBERT outperforms all the leading competitors and achieves 1.4% averaged loss with down to only 5% remaining weights compared to BERT-base.

2 Related Work

A lot of efforts have been made on studying network redundancy and pruning networks without accuracy loss (Gale et al., 2019; Renda et al., 2020). For example, the work on lottery ticket hypothesis (Frankle and Carbin, 2019) showed that there exist sparse smaller subnetworks capable of training to full accuracy in CNNs. Common wisdom in CNN literature shows that spare pruning leads to much more compression rate than structural pruning (Gale et al., 2019; Elsen et al., 2020). For example, for the same number of parameters (0.46M), the sparse MobileNets achieve 61.8% accuracy while the dense ones achieve 50.6% (Zhu and Gupta, 2017). However, similar observations are not observed in existing approaches for pre-trained language models (Fan et al., 2019; Michel et al., 2019; Chen et al., 2020; McCarley et al., 2020; Jiao et al., 2020). Our method aims to fill the gap and summarize these pruning strategies. There are other compression approaches for pre-trained language models, such as quantization (Zafrir et al., 2019) and weight factorization (Wang et al., 2019), which are out of the scope of this work.

3 SparseBERT

We first formalize the knowledge transfer involved in fine-tuning pre-trained language models. Then, we introduce our SparseBERT.

3.1 Knowledge Transfer under the Pre-train and Fine-tune Paradigm

The practice of fine-tuning pre-trained language models has become prevalent in various NLP tasks. The two-stage procedure is illustrated in Figure 1(a). The language model is denoted by f = $g \circ h$, where g is a text encoder and h is a task predictor head. Text encoders, like Transformers in BERT, are used to map input sentences to hidden representations and task predictors further map the representations to the label space. The pre-trained model is trained on a large amount of data examples $(\mathbf{x}^p, y^p)$ from the pre-training task domain via different tasks that resemble language modeling.

During pre-training, the general-purpose language knowledge, denoted by $\mathcal{L}$, is learned based on $(\mathbf{x}^p, y^p)$. $\mathcal{L}$ contains a subset that is related to the downstream task, denoted by $\mathcal{L}_D$, and the amount of $\mathcal{L}$ is far greater than that of $\mathcal{L}_D$ (see Figure 2(a)). To transfer knowledge $\mathcal{L}$ (especially $\mathcal{L}_D$) from pre-training domain to downstream domain, the well-trained encoder g_L is used to initialize the downstream encoder. In fine-tuning, downstream encoder is trained based on the task-specific knowledge $\mathcal{D}$ preserved in a small amount of data examples $(\mathbf{x}^d, y^d)$ from downstream domain. Finally, the well-trained downstream encoder g_{L_D} is evaluated on test data.

3.2 Knowledge-Aware Compression

3.2.1 Two Basic Pruning Strategies

Intuitively, there are two pruning strategies. One is that pruning is applied to the downstream encoder g_L during fine-tuning (see Figure 1(b)). However, because the loss to update the weights during fine-tuning is exclusively based on the data examples $(\mathbf{x}^d, y^d)$ from the downstream task domain, this pruning strategy might destruct the knowledge $\mathcal{L}_D$, which is learned based on $(\mathbf{x}^p, y^p)$ and encoded in the initialization of g_L.

The other strategy is that pruning is executed during pre-training (see Figure 1(c)). The generated pruned network preserves a subset of knowledge $\mathcal{L}$, denoted by $\mathcal{L}^{pr}$. Unfortunately, because this strategy ignores the downstream task information and the amount of $\mathcal{L}$ is extremely large, i.e., $\mathcal{L} \gg \mathcal{L}^{pr}$, the knowledge $\mathcal{L}^{pr}$ could be much different from $\mathcal{L}_D$ that we hope to preserve (see Figure 2(a)).

3.2.2 The Proposed Pruning Strategy

As shown in Figure 1(d), SparseBERT executes pruning at the distilling stage. It prunes the pre-trained encoder without fine-tuning, g_L, while fine-tuning the pruned encoder based on the downstream dataset $(\mathbf{x}^d, y^d)$. Recent findings indicate that self-attention and feed-forward layers are over-parameterized and are the most computation consumption parts (Michel et al., 2019; Voita et al., 2019; Ganesh et al., 2020). Thus, SparseBERT applies network pruning to the linear transformations matrices in self-attention and feed-forward layers (see Figure 3). The choice of pruning approach is flexible. We choose magnitude weight pruning (Han et al., 2015) in this paper, mainly because it is one of the most effective and popular pruning methods. More details about the pruning strategy used in SparseBERT can be found in the codes.

3.2.3 Knowledge Distillation Helps Pruning Preserve Task-Specific Knowledge

To mitigate the loss of $\mathcal{L}_D$, we propose to utilize knowledge distillation while pruning. We use the task-specific fine-tuned BERT as the teacher network and the pre-trained BERT that is being pruned as the student (see Figure 1(d) and Figure 3). The motivation is that the task-specific fine-tuned BERT preserves $\mathcal{L}_\mathcal{D}$. By feeding downstream task data $(\mathbf{x}^d, y^d)$ into the teacher-student framework, we help the student reproduce the behaviors of the teacher to learn both $\mathcal{L}_d$ and $\mathcal{L}$ as much as possible.

We design the distillation loss as

$$L_{distil} = L_{emb} + L_{att} + L_{hid} + L_{prd}. \quad (1)$$

L_{emb} = MSE($\mathbf{E}^S, \mathbf{E}^T$) is the difference between the embedding layers of student and teacher. L_{att} = $\sum$MSE($\mathbf{A}_i^S, \mathbf{A}_i^T$) is the difference between attention matrices and i is the layer index. L_{hid} = $\sum$MSE($\mathbf{H}_i^S, \mathbf{H}_i^T$) is the difference between hidden representations. L_{prd} = -softmax($\mathbf{z}^T$) $\cdot$ log_softmax($\mathbf{z}^S$/temp) is the soft cross-entropy loss between the logits of student and teacher. temp represents the temperature value. The proposed distillation loss is inspired by (Jiao et al., 2020) and it helps the student imitate the teacher's behavior as much as possible. In addition, we perform the same data augmentation as (Jiao et al., 2020) does to generate more task-specific data for teacher-student

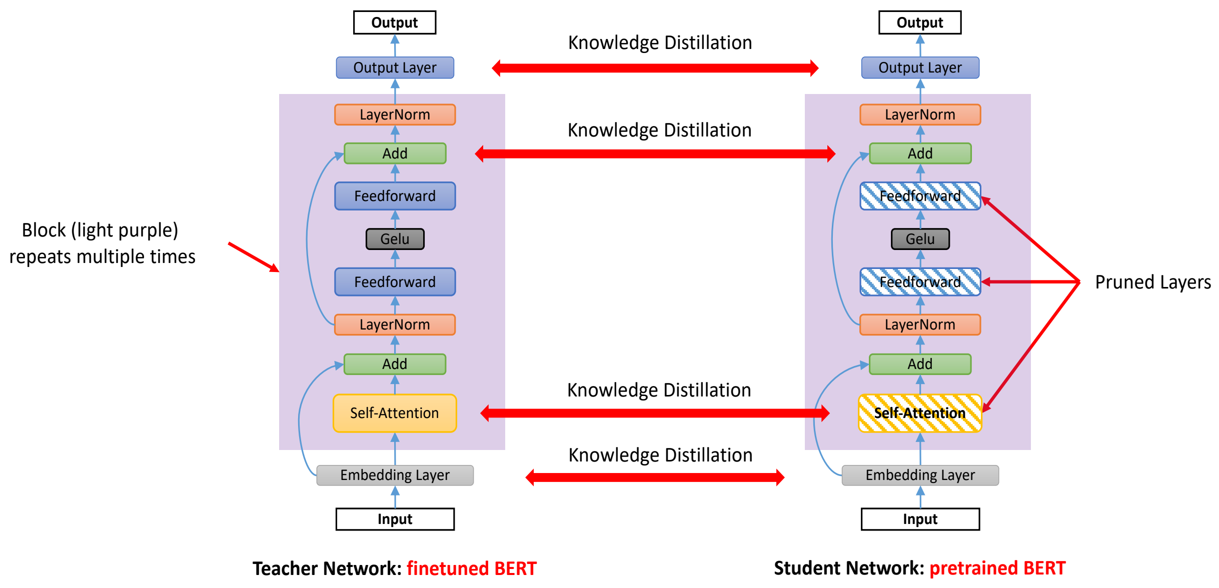


Figure 3: Illustration of the proposed knowledge-aware compression. Pruning is performed in parallel with distillation, based on specific data from downstream tasks.

learning. Notably, the choices of distillation loss and data augmentation method are flexible and we found the ones we adopted worked well in general.

4 Experiments

4.1 GLUE Benchmark

We evaluate SparseBERT on four data sets from the GLUE benchmark (Wang et al., 2018). To test if SparseBERT is applicable across tasks, we include the tasks of both single sentence and sentence-pair classification. We report the results on dev sets. We run 3, 20, 20, 50 epochs for QNLI, MRPC, RTE, CoLA separately. The baselines include BERT-base, ELMo (Peters et al., 2018), BERT-PKD (Sun et al., 2019), Bert-of-Theseus (Xu et al., 2020), DistilBERT (Sanh et al., 2019), MiniLM (Wang et al., 2020), TinyBERT (Jiao et al., 2020), BERT-Tickets (Chen et al., 2020), CompressBERT (Gordon et al., 2020), and RPP (Guo et al., 2019).

The results are shown in Table 1. Compared to BERT-base, SparseBERT achieves 1.4% averaged performance loss with down to 5% weights. In addition, SparseBERT outperforms all leading competitors with the highest sparsity.

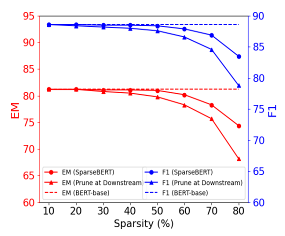
Method	Remain. Weights	QNLI (Acc)	MRPC (F1)	RTE (Acc)	CoLA (Mcc)	Avg.
		Without Pruning				
BERT-base	-	91.8	88.6	69.3	56.3	76.5
ELMo	-	71.1	76.6	53.4	44.1	61.3
		Structural Pruning				
BERT$_6$-PKD	50%	89.0	85.0	65.5	45.5	71.3
BERT-of-Theseus	50%	89.5	89.0	68.2	51.1	74.5
DistilBERT	50%	89.2	87.5	59.9	51.3	72.0
MiniLM$_6$	50%	91.0	88.4	71.5	49.2	75.0
TinyBERT$_6$	50%	90.4	87.3	66.0	54.0	74.4
TinyBERT$_4$	18%	88.7	86.8	66.5	49.7	72.9
		Sparse Pruning				
BERT-Tickets	30-50%	88.9	84.9	66.0	53.8	73.2
CompressBERT	10%	76.8	-	-	-	-
RPP	11.6%	88.0	81.9	67.5	-	-
SparseBERT	5%	90.6	88.5	69.1	52.1	75.1

Table 1: Comparison on the dev sets of GLUE.

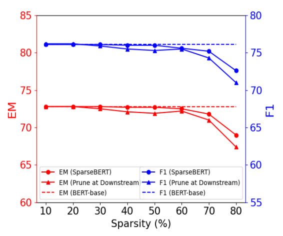
4.2 SparseBERT v.s. Pruning at Downstream

We compare SparseBERT with the pruning described in Figure 1(b) on the question answer tasks of SQuAD v1.1 and v2.0 (Rajpurkar et al., 2016, 2018). Given a question and a passage containing the answer, the two tasks are to predict the answer text span in the passage. The difference between them is that SQuAD v2.0 allows for the possibility that no short answer exists in the passage. We follow the general setting of SparseBERT, except that we only apply the logit distillation, i.e., $L_{distil} = L_{prd}$, and do not perform data augmentation, which are the most common distillation strategies.

The results are shown in Figure 4. It is observed that SparseBERT consistently outperforms the baseline method, especially at high sparsity. The performance gain of SparseBERT decreases on SQuAD v2.0 mainly because SQuAD v2.0 is more challenging than SQuAD v1.1. These observations demonstrate advantage of SparseBERT compared to pruning at downstream.



(a) SQuAD v1.1.



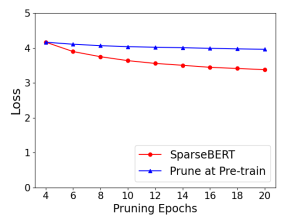
(b) SQuAD v2.0.

Figure 4: Performance comparison of SparseBERT and the pruning approach described in Figure 1(b).

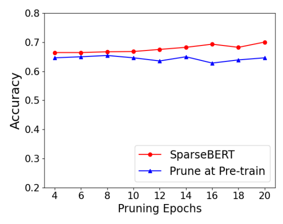
4.3 SparseBERT v.s. Pruning at Pre-Training

To get more insights about the advantage of SparseBERT over the pruning described in Figure 1(c), we compare their fitting abilities. Specifically, we use TinyBERT as an example of the baseline pruning method. We compare SparseBERT with TinyBERT with 4 layers and 312 hidden dimensions, which has a similar number of parameters as SparseBERT (sparsity=95%). SparseBERT only distills knowledge from the same layers as TinyBERT does.

We vary the number of pruning epochs and report the results (loss on training set and accuracy on dev set) on RTE in Figure 5. It is observed that SparseBERT consistently shows smaller training loss while higher evaluation performance, which demonstrates that SparseBERT has a better fitting ability when pruning compared to the baseline.



(a) Loss (training set).



(b) Performance (dev set).

Figure 5: Fitting ability comparison of SparseBERT and the pruning approach described in Figure 1(c).

5 Discussion

5.1 Hardware Performance

Sparse networks were not hardware-friendly in the past. However, hardware platforms with sparse tensor operation support have been rising up. For example, the latest release of Nvidia high-end GPU A100 has native support of sparse tensor operation up to 2x compression rate, while startup company such as Moffett AI has developed computing platform with sparse tensor operation acceleration up to 32x compression rate.

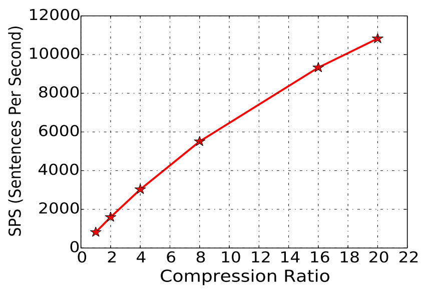


Figure 6: Hardware performance under different compression ratios on the MRPC dataset, with 818, 1594, 3029, 5508, 9326, and 10826 SPS (sentences per second) respectively.

Here we deployed SparseBERT of different sparse compression ratios (1, 2, 4, 8, 16, 20) on Moffett AI's latest hardware platform ANTOM to measure the real inference speedup induced by sparse compression, where '4' indicates the model is compressed by a factor of 4, with 75% of the parameters being zeros. As shown in Figure 6, the sparse compression has almost linear speedup up to 4x and leads to more than 10x speedup when compression rate is 20x.

5.2 Reduction of Parameters and FLOPS

We studied the reduction of parameters and FLOPS. For example, on the MRPC dataset, BERT-base (backbone) vs SparseBERT (backbone) = 85.53 vs 4.84 (#parameters, M) and BERT-base vs SparseBERT = 10.87 vs 0.54 (GFLOPS).

5.3 Inference/Training Time

We studied the time and convergence speed. For example, to get the reported 20x pruned result (Table 1), it needed 12 epochs of fine-tuning on MRPC and each epoch took 1.5 h (two RTX 2080 Ti). The inference time was around 20 s.

6 Conclusion

We introduce SparseBERT, a knowledge-aware sparse pruning method for pre-trained language models, with a focus on BERT. We summarize different types of BERT pruning approaches and compare SparseBERT with leading competitors. Experimental results on GLUE and SQuAD benchmarks demonstrate the superiority of SparseBERT.

7 Acknowledgements

We thank Xiaoqi Jiao for his valuable discussion and feedback on this work.

References

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained bert networks. *NeurIPS*, 33.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. 2020. Fast sparse convnets. In *CVPR*, pages 14629–14638.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2020. Compressing large-scale transformer-based models: A case study on bert. *arXiv preprint arXiv:2002.11985*.

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

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

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *NeurIPS*, pages 1135–1143.

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

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding.

J. S. McCarley, Rishav Chakravarti, and Avirup Sil. 2020. Structured pruning of a bert-based question answering model.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *NeurIPS*, pages 14014–14024.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *ACL*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing rewinding and fine-tuning in neural network pruning. In *ICLR*.

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

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *ACL*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *EMNLP*, pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

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

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. Bert-of-theseus: Compressing bert by progressive module replacing. *arXiv preprint arXiv:2002.02925*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.