Compressing Pre-trained Language Models by Matrix Decomposition

Matan Ben Noach[†] and **Yoav Goldberg**[*‡]
[*]Computer Science Department, Bar-Ilan University, Ramat-Gan Israel
[†]Intel AI Lab, Petah-Tikva Israel
[‡]Allen Institute for Artificial Intelligence
matan.ben.noach@intel.com, yoav.goldberg@gmail.com

Abstract

Large pre-trained language models reach state-of-the-art results on many different NLP tasks when fine-tuned individually; They also come with a significant memory and computational requirements, calling for methods to reduce model sizes (green AI). We propose a two-stage model-compression method to reduce a model's inference time cost. We first decompose the matrices in the model into smaller matrices and then perform feature distillation on the internal representation to recover from the decomposition. This approach has the benefit of reducing the number of parameters while preserving much of the information within the model. We experimented on BERT-base model with the GLUE benchmark dataset and show that we can reduce the number of parameters by a factor of 0.4x, and increase inference speed by a factor of 1.45x, while maintaining a minimal loss in metric performance.

1 Introduction

Deep learning models have been demonstrated to achieve state-of-the-art results, but require large parameter storage and computation. It's estimated that training a Transformer model with a neural architecture search has a CO_2 emissions equivalent to nearly five times the lifetime emissions of the average U.S. car, including its manufacturing (Strubell et al., 2019). Alongside the increase in deep learning models complexity, in the NLP domain, there has been a shift in the NLP modeling paradigm from training a randomly initialized model to fine-tuning a large and computational heavy pre-trained language model (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2018; Radford, 2018; Radford et al., 2019; Dai et al., 2019; Yang et al., 2019; Lample and Conneau, 2019; Liu et al., 2019b; Raffel et al., 2019; Lan et al., 2019; Lewis et al., 2019).

While re-using pre-trained models offsets the training costs, inference time costs of the fine-tuned models remain significant, and are show-stoppers in many applications. The main challenge with pre-trained models is how can we reduce their size while saving the information contained within them. Recent work, approached this by keeping some of the layers while removing others (Sanh et al., 2019; Sun et al., 2019; Xu et al., 2020). A main drawback of such approach is in its coarse-grained nature: removing entire layers might discard important information contained within the model, and working at the granularity of layers makes the trade-off between compression and accuracy of a model hard to control. Motivated by this, in this work we suggest a more fine-grained approach which decomposes each matrix to two smaller matrices and then perform feature distillation on the internal representation to recover from the decomposition. This approach has the benefit of preserving much of the information while reducing the number of parameters. Alongside the advantage of preserving the information within each layer, there is also a memory flexibility advantage compared to removing entire layers; As a result of decomposing each matrix to two smaller matrices, we can store each of the two matrices in two different memory blocks. This has the benefit of distributing the model matrices in many small memory blocks, which is useful when working in shared CPU-based environments.

We evaluated our approach on the General Language Understanding Evaluation (GLUE) benchmark dataset (Wang et al., 2018) and show that our approach is superior or competitive in the different GLUE tasks to previous approaches which remove entire layers. Furthermore, we study the effects of different base models to decompose and show the superiority of decomposing a fine-tuned model compared to a pre-trained model or a ran-

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 884–889
December 4 - 7, 2020.

domly initialized model. Finally, we demonstrate the trade-off between compression and accuracy of a model.

2 Related Work

In the past year, there have been many attempts to compress transformer models involving pruning (McCarley, 2019; Guo et al., 2019; Wang et al., 2019; Michel et al., 2019; Voita et al., 2019; Gordon et al., 2020), quantization (Zafrir et al., 2019; Shen et al., 2019) and distillation (Sanh et al., 2019; Zhao et al., 2019; Tang et al., 2019; Mukherjee and Awadallah, 2019; Sun et al., 2019; Liu et al., 2019a; Jiao et al., 2019; Izsak et al., 2019). Specifically, works on compressing pre-trained transformer language models focused on pruning layers. Sun et al. (2019) suggested to prune layers while distilling information from the unpruned model layers. Xu et al. (2020) proposed to gradually remove layers during training.

We also note that very recently a work similar to ours was uploaded to arxiv (Mao et al., 2020). There are a few differences from their work to ours. Firstly, we distill different parts of the model (see Section 3 for details). Secondly, we focus on training the decomposed model and do not prune the model parameters. Thirdly, our base model, which is used for decomposition and as a teacher, is a fine-tuned model; This has the benefit of task-specific information as we show in our experiments in Section 4.2.

3 Method

Our goal is to decompose each matrix $W \in R^{n\times d}$ as two smaller matrices, obtaining an approximated matrix $W' = AB$, $A \in R^{n\times r}$, $B \in R^{r\times d}$, where $r < \frac{nd}{n+d}$. We seek a decomposition s.t. W' is close to W in the sense that $d(Wx, W'x)$ is small for all x, where d is a distance metric between vectors. In practice, we require the condition to hold not for all x, but for vectors seen in a finite relevant sample (in our case, the training data). While one could start with random matrices and optimize the objective using gradient descent, we show that a two-staged approach performs better: we first decompose the matrices using SVD, obtaining A', B' s.t. $||A'B' - W||_2^2$ is small (SVD is guaranteed to produce the best rank-r approximation to W, (Stewart, 1991)). We then use these matrices as initialization and optimize $d(Wx, W'x)$ (feature distillation), while also optimizing for task loss. We show that this process works substantially better in practice. Our loss function is thus composed of three different objectives:

Cross Entropy Loss The cross entropy loss over an example x with label y is defined likewise: $L_{CE} = -\log p_s(y|x)$, where p_s is the probability for label y given by the decomposed student model.

Knowledge Distillation Loss The goal of knowledge distillation is to imitate the output layer of a teacher model by a student model. The Knowledge Distillation Loss is defined likewise: $L_{KD} = \left\| \frac{\mathbf{z_s} - \mathbf{z_t}}{T} \right\|_2$, where $\mathbf{z_s}$ and $\mathbf{z_t}$ are the logits of the decomposed and original models respectively and T is a temperature hyper-parameter.

Feature Distillation Loss The goal of feature distillation is to imitate the intermediate layers of a teacher model by a student model. we use the following intermediate representations to distill the knowledge from[1]:

- Query, Key and Value Layers - The dot product of a matrix of concatenated tokens representation vectors X by the query, key and value parameter matrices, $Z_q = X \cdot W^Q$, $Z_k = X \cdot W^K$, $Z_v = X \cdot W^V$
- Attention Matrix - The attention matrix probabilities. $Z_{att} = softmax(Z_q \cdot Z_k^T)$
- Attention Heads - The output of the attention heads. $Z_H = Z_{att} \cdot Z_v$
- The Multihead Attention Layer Output - The dot product of the attention heads by the matrix W^O. $Z_{MH} = Z_H \cdot W^O$
- The first feed forward layer - The dot product of the multihead attention layer by the first feed forward layer. $Z_{f1} = Z_{MH} \cdot W_1$
- The second feed forward layer - The dot product of the first feed forward layer by the second feed forward layer. $Z_{f2} = Z_{f1} \cdot W_2$

We denote S_z^i and T_z^i as the intermediate representations which were described above of layer i for

[1] We follow the notations of Vaswani et al. (2017) for the transformer parameters and omit biases for notation convenience.

the decomposed student and original teacher models respectively. Our loss function then is defined by: $L_{FD} = \sum_i \sum_{T_z,S_z}^{T_z^i,S_z^i} \|T_z - S_z\|_2$

Full Objective Our loss function is then defined by a weighted combination of these three loss functions likewise: $L = \alpha L_{CE} + (1 - \alpha)L_{KD} + L_{FD}$ where $\alpha \in [0, 1]$ is a chosen hyper-parameter.

4 Experiments

We compare various variants of our compression method, corresponding to different subsets of our loss. All variants decompose the matrices using SVD, but differ in their objective functions. These correspond to the four last lines in Table 1. Low Rank BERT Fine-tuning (LRBF) corresponds to $L = L_{CE}$. LRBF+KD corresponds to $L = \alpha L_{CE} + (1 - \alpha)L_{KD}$. LRBF+FD corresponds to $L = L_{CE} + L_{FD}$, while LRBF+FD+KD corresponds to the complete objective.

The other lines in the table correspond to uncompressed model (first line) and to baselines which prune layers and distill. Fine-tuning fine-tunes a six layered BERT model. Vanilla KD trains a six-layered BERT model with $L = \alpha L_{CE} + (1-\alpha)L_{KD}$. BERT-PKD trains a six layered BERT model with $L = \alpha L_{CE} + (1-\alpha)L_{KD}$ while also adding an L_{FD} objective, but on the hidden states between every consecutive layer. BERT-of-Theseus fine-tunes BERT model while gradually pruning half of the layers. We chose this baselines for several reasons: like our method they result in a practical reduction of parameters;[2], they are task-specific;[3] and they do not require the pre-training stage, which is expensive and not practical for most practitioners.

Datasets We evaluate our proposed approach on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), a collection of diverse NLP tasks.

Training Details We fine-tune a pre-trained BERT model (Devlin et al., 2018) for each task with a batch size of 8 and a learning rate of $2e{-}5$ for 3 epochs with an early stop mechanism according to the validation set. We perform the matrix decomposition on every parametric weight matrix of the encoder (excluding the embedding matrix) in a fine-tuned model and train the decomposed model as the student model and the original fine-tuned model as the teacher. For each task we train for 3 epochs with an early stopping mechanism according to the task validation set, the maximum sequence length is 128 and we perform a grid search over the learning rates $\{2e^{-6}, 5e^{-6}, 2e^{-5}, 5e^{-5}, 2e^{-4}, 5e^{-4}\}$ and 5 different seeds and choose the best model according to the validation set of each task.[4] For knowledge distillation hyper-parameters we used a temperature hyper-parameter $T = 10$ and $\alpha = 0.7$.[5]

4.1 Main Results

Table 1 compares the results for validation and test of other compression approaches which prune layers, along with low rank models which were fine-tuned and trained with one or more of the distillation objectives described in Section 3. As can be seen, Low Rank BERT Feature Distillation + KD and Low Rank BERT Feature Distillation surpass all of results of all methods in both validation and test sets except BERT-of-Theseus method in the test set, in which Low Rank BERT Feature Distillation + KD surpasses the results in 5 of the tasks and reach comparable results in 2 of the tasks. Also, as can be seen knowledge distillation alone is not sufficient to compensate for the decomposition, but it slightly improves the results when incorporating feature distillation alone.

4.2 Further Analysis

Effect of Base Model and Decomposition In this experiment we test the importance of the base model we use to decompose and use as a teacher. We compared between three types of distillation sources: fine-tuned teacher, pre-trained teacher and no teacher. Furthermore, we compared between three types of model initializations: a decomposed fine-tuned model, a decomposed pre-trained model and a randomly initialized model with the same architecture as the decomposed models. The results are shown in Table 2, on all tasks when training with no teacher distilla-

[2]Unlike, e.g., pruning, which sets parameters to zero and requires specialized hardware to fully take advantage of.

[3]Unlike, e.g., DistillBERT which is meant to be run before fine-tuning.

[4]We detailed the changes we made to the original fine-tuning procedure, every other hyper-parameters which were not mentioned, is set as described in (Devlin et al., 2018).

[5]We chose those hyper-parameters from a grid search over $T = \{5, 10, 20\}$ and $\alpha = \{0.2, 0.5, 0.7\}$ on the MRPC validation set.

Method	CoLA		MNLI		MRPC		QNLI		QQP		RTE		SST-2		STS-B		Macro Score	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
Uncompressed Models - 110M Parameters																		
BERT-base (uncompressed)	59.9	53.9	84.6	83.9	89.0	85.6	91.6	90.9	88.1	80.2	71.5	67.2	93.5	93.6	89.8	84.8	83.5	80.0
6 Layers Transformer Models - 66M Parameters																		
Fine-tuning	43.4	41.5	80.1	80.1	86.0	83.1	86.9	86.7	87.8	78.7	62.1	63.6	89.6	90.7	81.9	81.1	77.2	75.7
Vanilla KD (Hinton et al., 2015)	45.1	42.9	80.1	80.0	86.2	83.4	88.0	88.3	88.1	79.5	64.9	64.7	90.5	91.5	84.9	81.2	78.5	76.4
BERT-PKD (Sun et al., 2019)	45.5	43.5	81.3	81.3	85.7	82.5	88.4	89.0	88.4	79.8	66.5	65.5	91.3	92.0	86.2	82.5	79.2	77.0
BERT-of-Theseus (Xu et al., 2020)	51.1	**47.8**	82.3	82.3	89.0	85.4	89.5	89.6	89.6	**80.5**	68.2	66.2	91.5	92.2	88.7	**84.9**	81.2	78.6
Low Rank Approximated Models - 65.2M Parameters (This Work)																		
Low Rank BERT Fine-tuning	41.0	40.5	82.9	82.3	82.4	79.8	89.4	88.8	89.0	79.5	65.0	60.4	91.3	92.0	87.0	81.2	78.5	75.6
Low Rank BERT + KD	44.7	34.0	83.1	82.4	83.4	80.4	89.1	88.7	89.0	79.9	64.3	60.6	91.3	91.5	86.6	80.9	78.9	74.8
Low Rank BERT Feature Distillation	**51.2**	43.4	**84.9**	**83.8**	**89.4**	**86.1**	**91.4**	**90.7**	**89.8**	**80.5**	**70.8**	66.0	**92.2**	**92.9**	**89.3**	84.2	**82.4**	78.4
Low Rank BERT Feature Distillation + KD	**53.0**	42.9	**84.8**	**83.7**	**90.4**	**86.2**	**91.4**	**90.8**	**89.7**	**80.5**	**71.1**	**67.8**	**92.4**	**92.9**	**89.4**	84.6	**82.8**	**78.7**

Table 1: Results on GLUE dev and test sets. Metrics are *Accuracy* (MNLI (average of MNLI match and MNLI mis-match), QNLI, RTE, SST-2), *Avg of Accuracy and F1* (MRPC, QQP), *Matthew's correlation* (CoLA), *Avg of Pearson and Spearman correlations* (STS-B). BERT-base (Teacher) is our fine-tuned BERT model. The numbers for the 6 layered models are taken from (Xu et al., 2020), Best results are indicated in Bold.

Base Model/Teacher Model	CoLA			MRPC			SST-2		
	Fine-tuned	Pre-trained	None	Fine-tuned	Pre-trained	None	Fine-tuned	Pre-trained	None
Fine-tuned	48.7 ± 2.4	47.5 ± 0.7	40.1 ± 0.6	88.5 ± 0.5	85.8 ± 0.5	81.6 ± 1.3	91.8 ± 0.4	91.3 ± 0.5	90.9 ± 0.4
Pre-trained	49.4 ± 1.7	44.8 ± 2.1	10.8 ± 2.6	89.2 ± 0.4	86.3 ± 1.0	77.1 ± 0.6	91.7 ± 0.2	91.2 ± 0.4	89.6 ± 1.1
Random	3.6 ± 5.1	0.0 ± 0.0	0.6 ± 0.6	75.9 ± 1.1	75.3 ± 0.7	75.0 ± 0.4	88.2 ± 0.5	87.2 ± 0.7	81.2 ± 0.5

Table 2: Results on the dev set of CoLA, MRPC and SST-2 tasks with different initializations and different teachers. The results are averages and standard deviations of five runs with different seeds.

tion, the results are best when decomposing a fine-tuned model and decomposing a pre-trained model is better than randomly initializing a model; This indicates that the decomposition saves the information within the model and when decomposing a fine-tuned model it saves some of the more task specific information. Furthermore, on all tasks and all initialization the best results are when using a fine-tuned model as a teacher.

Rank (Parameter Count)	CoLA	MRPC	SST-2
Full Rank (110M)	58.4 ± 1.2	88.3 ± 0.7	92.8 ± 0.5
350 (82.6M)	57.7 ± 0.9	88.9 ± 0.7	92.0 ± 0.5
245 (65.2M)	48.7 ± 2.4	88.5 ± 0.5	91.8 ± 0.4
150 (49.4M)	38.7 ± 1.6	87.8 ± 0.6	91.3 ± 0.4

Table 3: Results on the dev set of CoLA, MRPC and SST-2 tasks with different ranks. The results are averages and standard deviations of five runs with different seeds.

Compression vs. Performance Trade-off Our method requires to determine a rank for the compression. But can we achieve better results when choosing a higher rank? Can we choose a lower rank for smaller models and still achieve satisfactory results? To determine this we experimented on three different ranks. As shown in Table 3, higher ranks achieve better results, while lower ranks achieve satisfactory results while compromising metric performance.

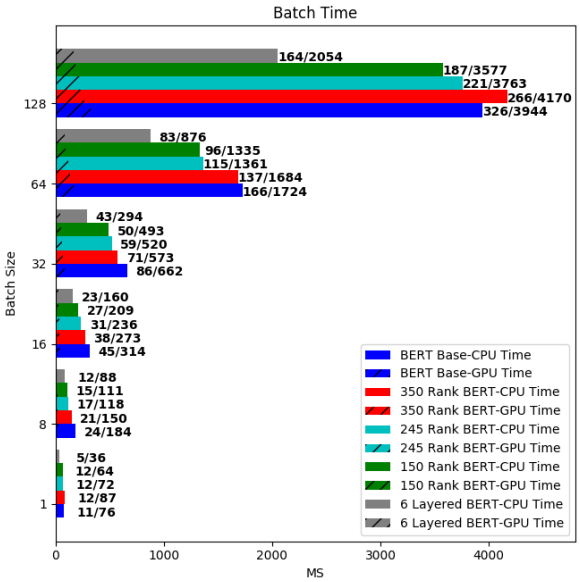

Figure 1: Average time in milliseconds to run a batch of samples from all of the GLUE tasks, when running on a Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz and on a single TITAN V 12GB GPU.

Run-time Savings In this experiment we measured the average time in milliseconds it takes for BERT-base compared to its decomposed and six-layered counterparts to output predictions for a batch of samples with varying batch sizes. As shown in Figure 1, we still gain a significant time performance improvement when running on both CPU and GPU architectures over a BERT-base

model. Models that are decomposed to a rank $r = 245$ are about 1.45 faster than their uncompressed counterpart for batches larger than one when running on a GPU and around $1.2 - 1.55$ faster for batches $8, 16, 32, 64$ when running on a CPU. Furthermore, higher ranks still benefit running time and lower ranks improve the running time further. Also, we note that although a six-layered BERT does achieve faster inference time, due to the coarse-grained compression, it losses more information contained within it and thus achieves inferior results; As shown in the results in Table 1, a six-layered model trained with distillation (e.g. BERT-PKD (Sun et al., 2019)) achieves significantly lower results and the BERT-of-Theseus model, which does improve upon BERT-PKD, requires many training iterations to achieve this to overcome the loss of information when gradually removing entire layers, which result in higher training times.

5 Conclusions

We presented a way to compress pre-trained large language models fine-tuned for specific tasks, while preserving much of the information contained within them, by using matrix decomposition to two small matrices. For future work it might be interesting to combine this approach with another approach such as pruning or quantization to achieve smaller models.

Acknowledgements

This project has received funding from the Europoean Research Council (ERC) under the Europoean Union's Horizon 2020 research and innovation programme, grant agreement No. 802774 (iEXTRACT) and was sponsored in part by an Intel AI grant to the Bar-Ilan University NLP lab.

References

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988.

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

Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: studying the effects of weight pruning on transfer learning. *CoRR*, abs/2002.08307.

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

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Peter Izsak, Shira Guskin, and Moshe Wasserblat. 2019. Training compact models for low resource entity tagging using pre-trained language models. *CoRR*, abs/1910.06294.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Linqing Liu, Huan Wang, Jimmy Lin, Richard Socher, and Caiming Xiong. 2019a. Attentive student meets multi-task teacher: Improved knowledge distillation for pretrained models. *CoRR*, abs/1911.03588.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Yaming Yang, Quanlu Zhang, Yunhai Tong, and Jing Bai. 2020. Ladabert: Lightweight adaptation of BERT through hybrid model compression. *CoRR*, abs/2004.04124.

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

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *CoRR*, abs/1905.10650.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2019. Distilling transformers into simple neural networks with unlabeled transfer data. *CoRR*, abs/1910.01769.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. *ArXiv*, abs/1802.05365.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

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

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2019. Q-BERT: hessian based ultra low precision quantization of BERT. *CoRR*, abs/1909.05840.

G. W. Stewart. 1991. Perturbation theory for the singular value decomposition. *SVD and Signal Processing, II: Algorithms, Analysis and Applications*, pages 99–109.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from BERT into simple neural networks. *CoRR*, abs/1903.12136.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

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. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5797–5808.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019. Structured pruning of large language models. *CoRR*, abs/1910.04732.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. Bert-of-theseus: Compressing BERT by progressive module replacing. *CoRR*, abs/2002.02925.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8BERT: quantized 8bit BERT. *CoRR*, abs/1910.06188.

Sanqiang Zhao, Raghav Gupta, Yang Song, and Denny Zhou. 2019. Extreme language model compression with optimal subwords and shared projections. *CoRR*, abs/1909.11687.