On the Compression of Natural Language Models

Saeed Damadi
Department of Computer Science and Electrical Engineering Department
University of Maryland, Baltimore County
sdamadi1@umbc.edu

Abstract
Deep neural networks are effective feature extractors but they are prohibitively large for deployment scenarios. Due to the huge number of parameters, interpretability of parameters in different layers is not straightforward. This is why neural networks are sometimes considered black boxes. Although simpler models are easier to explain, finding them is not easy. If found, a sparse network that can fit to a data from scratch would help to interpret parameters of a neural network. To this end, (Frankle and Carbin, 2018) showed that typical dense neural networks contain a small sparse sub-network that can be trained to a reach similar test accuracy in an equal number of steps. The goal of this work is to assess whether such a trainable subnetwork exists for natural language models (NLM)s. To achieve this goal we will review state-of-the-art compression techniques such as quantization, knowledge distillation, and pruning.

1 Introduction
Although deep neural networks (DNNs) are over-parameterized (Denil et al., 2013), they have been able to solve many difficult problems. For example, pre-trained feature extractors, such as BERT (Devlin et al., 2018) for natural language processing and residual networks for computer vision tasks, have become effective methods for improving deep learning models without requiring more labeled data. However, they are sizable and cannot be deployed on devices with small memories like smart phones. This has had researchers to compress DNNs. To compress DNNs there are three main ways: 1) quantization, 2) knowledge distillation, and 3) pruning where the last one is the main focus of this manuscript. The less parameters network has, the less computational cost would be incurred. This cost may be associated with training or testing a DNN. However, most of compression techniques try to reduce computational cost associated with running queries (testing). This is justified by stating that “training is done once, and testing is done infinitely many times”. In addition to reducing the number of parameters, one may reduce the accuracy of the parameters in order not to occupy the memory. This is the most straightforward way for reducing computational cost of testing which is called quantization and will be explained briefly as follows.

1.1 Quantization
Quantization techniques try to store model parameters from 32- or 16-bit floating number to 8-bit or even lower. This is a very simple method for reducing the computational cost of queries. Although, from implementation perspective is very effective, reducing the bit representation of parameters may cause significant accuracy loss. To avoid that, quantization-aware training has been developed to maintain similar accuracy to the original model (Shen et al., 2020; Zafrir et al., 2019). Quantization is applied on a fixed DNN after training. On the other hand, there are techniques looking for a different smaller structures that can learn the same task. One of these method is knowledge distillation which is explained as follows.

1.2 Knowledge distillation
In essence, knowledge distillation tries to use a large trained network to train a smaller network. In this method, a small network is forced to have an output that is similar to a large trained network. The large trained network is called teacher and the small network is called student. Therefore, a com-
pressed and shallow student network under the guidance of a complicated larger teacher network is trained. The trained compressed student network can be directly deployed in real-life applications. Existing methods can be generally divided into two categories: 1) task-specific, and 2) task-agnostic. Task-specific methods require that the teacher be trained for each downstream task. Distilled bidirectional long short-term memory network (Distilled BiLSTM) (Tang et al., 2019), Patient Knowledge Distillation for a BERT model (BERT-PKD) (Sun et al., 2019), and Stacked Internal Distillation (SID) (Aguilar et al., 2020) are all considered as task-specific methods. On the other hand, task-agnostic method uses one teacher for several downstream tasks. In effect, one teacher can train multiple students. Methods such as DistilBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2019) and MobileBERT (Sun et al., 2020) are task-specific methods. Note that process of task-agnostic BERT distillation is computationally expensive (McCarley et al., 2019) because the corpus used in the distillation is sizable and for each training step a forward process of teacher model and a forward-backward process of student model should be performed.

1.2.1 The larger the teacher, the better

The more information the teacher can provide with the the student, the better the performance of the student model would be. To this end, TinyBERT, MobileBERT, and SID all try to improve BERT-PKD by distilling more internal representations to the student, such as embedding layers and attention weights. TinyBERT (Jiao et al., 2019) and MobileBERT (Sun et al., 2020) are small student models distilled from larger pre-trained transformers and can achieve good GLUE (Wang et al., 2018) scores. However, these models require spending substantial compute to pre-train the larger teacher model. To address this issue ELECTRA-Small (Clark et al., 2020) focus more on pre-training speed rather than inference speed.

1.3 Pruning

The main focus of this paper is pruning as a method for compressing neural networks. Pruning refers to identifying and removing redundant or less important parameters. There are two approaches for pruning NLMs: 1) structured, and 2) unstructured. The main difference between the two is that the former drops network block(s), while the latter zeros out individual parameters across the entire network.

1.3.1 Structured Pruning

In structured pruning, architecture blocks (layers) are pruned. Structured pruning first utilized for convolutional neural networks but it has recently been applied to NLMs. Structure pruning can be done in two directions, depth (layers) or width (headers) of the network. In the depth direction, pruning transformer layers is proposed in LayerDrop (Fan et al., 2019) via structured dropout. (Sajjad et al., 2020) prunes the network in the depth direction as well. In the width direction (Michel et al., 2019; Voita et al., 2019; McCarley et al., 2019) retain the performance after pruning a large percentage of attention heads in a structured manner.

1.3.2 Unstructured Pruning

In unstructured pruning, parameters with the least importance are removed from the network. The importance of the weights can be judged by their absolute values, the gradients, or by some custom-designed measurement. Since unstructured pruning considers each parameter individually, the set of pruned ones can be arbitrary and irregular, which in turn might decrease the model size, but with negligible improvement in runtime memory or inference speedup (Sanh et al., 2020), unless executed on specialized hardware or with specialized processing libraries (Han et al., 2016; Chen, 2018). Unstructured pruning could be effective for BERT, given the massive amount of fully-connected layers.

1.4 State-of-the-art-pruning methods

In this subsection we will go over state-of-the-art-pruning methods. As the goal is to explore as many as methods, we briefly introduce each method and do not dive into them.

1.4.1 Pruning of question answering models

To create small task-specific language representations (McCarley et al., 2019) tries to compress BERT- and RoBERTa (Liu et al., 2019) based question answering systems by combining task-specific knowledge distillation and task-specific structured pruning. The intuition is based on two observations: 1) a general-purpose language representation requires expensive pretraining distillation so the number of tasks should be limited,
and 2) BERT models are robust to attention heads pruning.

1.5 LayerDrop: Structured dropout

Dropping an entire layer which is considered as structured pruning is studied in (Fan et al., 2019). They show that it is possible to extract shallower subnetworks from large networks without having to finetune. The performance of these subnetworks is close to the dense network. DropLayer (Fan et al., 2019) uses group regularization to enable structured pruning at inference time.

1.5.1 Head pruning

The original BERT model has 16 attention heads and (Michel et al., 2019) questions the number of heads in the BERT model. (Michel et al., 2019) shows that many of BERT’s attention heads can be pruned, while high accuracy during test time is possible with only 1–2 attention heads per encoder unit. This fact shows that there is a high redundancy in the learned parameters of BERT model. Pruning these many attention heads improve memory efficiency in transformers.

1.5.2 Factorized low-rank Pruning

By combining pruning and matrix factorization for model compression (Wang et al., 2019) proposes a method to structurally prune BERT models using low-rank factorization and augmented Lagrangian $L_0$ norm regularization. The method is called Factorized Low-rank Pruning (FLOP). Given $W$ as a weight matrix, structured pruning can be achieved by replacing the computation of $Wx$ by $WGx$ where diagonal sparsity inducing matrix $G$ is learned using $L_0$ regularization over $WG$ along with the supervised loss. Next, $W$ is decomposed to two smaller matrix $P$ and $Q$ such that $W = PGQ$. In other words (Wang et al., 2019) reparameterizes weight matrices using low-rank factorization and removes rank-1 components during training. One limitation of this method is that this structured pruning method tends to produce lower performance than its unstructured counterpart.

1.5.3 Reweighted proximal pruning

Observing the fact that larger weights are penalized more heavily than smaller ones when $L_1$ regularization is utilized, (Guo et al., 2019) proposes reweighted $L_1$ minimization. This observation is the intrinsic artifact of $L_1$ minimization. Once it happens it defeats the purpose of weight pruning which is “removing the unimportant connections”. (Guo et al., 2019) observes that direct optimization of a regularization penalty term causes divergence from the original loss function and has negative effect on the effectiveness of gradient-based update. To avoid aforementioned problems they solve the following optimization problem:

$$\min_w f(w) + \sum_i \alpha_i w_i$$

where $f(w)$ is the original loss function, $w$ is a parameter (weight) and $0 < \alpha_i$’s are inversely proportional to magnitude of corresponding weights $w_i$. This optimization is solved using a reweighted proximal pruning (RPP) method (which depends on proximal operators). RPP decouples the goal of high sparsity from minimizing loss, and hence leads to improved accuracy even with high levels of pruning for BERT models.

1.5.4 Compressing pretrained BERT

Using pretrained BERT as feature extractor is widespread for natural language processing tasks. Because their learned parameters can be transfer to other tasks. (Gordon et al., 2020) explores the effects of unstructured pruning on transfer learning. (Gordon et al., 2020) observes that low level of pruning (30-40%) let learned parameters to be transformed to downstream tasks without any increase in the loss. As opposed to low level pruning, medium and high levels of pruning prevent useful pretraining information be transferred to downstream tasks. For example, Multi-Genre Natural Language Inference (MNLI) is the task that is most sensitive to pruning under their method. The main observation of (Gordon et al., 2020) is that BERT can be pruned during pretraining rather than separately for each task without affecting performance.

1.5.5 Movement pruning

(Sanh et al., 2020) argues that for transfer learning, what matters is not the magnitude of a parameter. What matters is that whether that parameter is important for the downstream task. They introduce movement pruning which is a first-order method. This method prunes parameters shrinking during fine-tuning regardless of their magnitude. Movement pruning is able to achieve significantly higher performance than magnitude- or $L_0$ based pruning for very high levels of sparsity (e.g., 97%
sparse). It also can be combined with distillation to improve the performance further.

1.5.6 Know what you don’t need

(Zhang et al., 2021) shows that many attention heads can be removed without a significant impact on the performance. To remove those attention heads a single-shot meta-pruner is introduced. This single-shot meta-pruner is a small convolutional neural network that is trained to select heads that contribute to maintaining the attention distribution.

1.6 The Lottery Ticket Hypothesis

The Lottery Ticket Hypothesis (LTH) states that typical dense neural networks contain a small sparse sub-network that can be trained to reach similar test accuracy in an equal number of steps (Frankle and Carbin, 2018). In view of that, follow-up works reveal that sparsity patterns might emerge at the initialization (Lee et al., 2018), the early stage of training (You et al., 2019) and (Chen et al., 2020b), or in dynamic forms throughout training (Evci et al., 2020) by updating model parameters and architecture typologies simultaneously. Some of the recent findings are that the lottery ticket hypothesis holds for BERT models, i.e., largest weights of the original network do form subnetworks that can be retrained alone to reach the performance close to that of the full model (Prasanna et al., 2020; Chen et al., 2020a).

1.6.1 LTH for BERT

(Chen et al., 2020a) applies the LTH to identify matching subnetworks in pretrained BERT models to enforce sparsity in models trained for different downstream tasks. They show that assuming pretrained parameters as the initialization for BERT models, one can find sparse subnetworks that can be trained for downstream NLP tasks. They also show that using unstructured magnitude pruning, one can find matching subnetworks at between 40% and 90% sparsity in BERT models on standard GLUE and SQuAD downstream tasks. While this approach retains accuracy and can lead to sparser models, it does not lead to improvements in training speed without dedicated hardware or libraries (Han et al., 2016; Chen, 2018). On the other hand, in structured pruning, the best subnetworks of BERT’s heads do not quite reach the full model performance. (Chen et al., 2020a) shows for a range of downstream tasks, matching subnetworks at 40% to 90% sparsity exist and they are found at pretrained phase (initialization). This is dissimilar to the prior NLP research where subnetworks emerge only after some amount of training. Subnetworks that are found on the masked language modeling task transfer universally; those found on other tasks transfer in a limited fashion if at all.

1.6.2 EarlyBERT

(Chen et al., 2020b) introduces EarlyBERT which extends the work done on finding lottery-tickets in CNNs (You et al., 2019) to speedup both pre-training and fine-tuning for BERT models. (You et al., 2019) realized that sparsity patterns might emerge at the initialization. Experimental evaluation of EarlyBERT shows some degradation in accuracy for fine-tuning.

1.7 Conclusion

(Prasanna et al., 2020) and (Chen et al., 2020a) explore BERT models from the perspective of the lottery ticket hypothesis (Frankle and Carbin, 2018), looking specifically at the “winning” subnetworks in pre-trained BERT. They find that such subnetworks do exist, and that transferability between subnetworks for different tasks varies. The two papers provide complementary results for magnitude pruning and use a task specific classifier which is randomly initialized.

1.7.1 Issues

The issue with pruning is that existing works on pruning of BERT yields inferior results than its small-dense counterparts such as TinyBERT (Jiao et al., 2019). While unstructured pruning may need dedicated hard-ware or libraries (Han et al., 2016; Chen, 2018), one may prefer them over structured pruning because they maintain accuracy and can lead to sparser models.

1.7.2 An open question

There is an open question as follows. Is there a sparse task-specific classifier that can be combined with a pruned pretrained BERT model to get the accuracy on a par with a dense trained network? If such a task-specific classifier exists, can we use it for other tasks?

1.7.3 Research directions

In computer vision, (Kusupati et al., 2020) sparsifies the network with reparameterization technique that uses one extra single parameters for
each layer. This method can be applied to a pre-trained BERT to see how well it performs.

References
Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. 2020. Knowledge distillation from internal representations. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 7350–7357.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020a. The lottery ticket hypothesis for pre-trained bert networks. arXiv preprint arXiv:2007.12223.

Xiaohan Chen, Yu Cheng, Shuohang Wang, Zhe Gan, Zhangyang Wang, and Jingjing Liu. 2020b. Early-bert: Efficient bert training via early-bird lottery tickets. arXiv preprint arXiv:2101.00063.

Xuhao Chen. 2018. Escoin: Efficient sparse convolutional neural network inference on gpus. arXiv preprint arXiv:1802.10280.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv:2003.10555.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc’Aurelio Ranzato, and Nando De Freitas. 2013. Predicting parameters in deep learning. In Advances in neural info. processing systems, pages 2148–2156.

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

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In International Conference on Machine Learning, pages 2943–2952. PMLR.

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. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635.

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, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. 2016. Eie: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 44(3):243–254.
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. arXiv preprint arXiv:1909.10351.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. 2020. Soft threshold weight reparameterization for learnable sparsity. In International Conference on Machine Learning, pages 5544–5555. PMLR.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:1810.02340.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

JS McCarley, Rishav Chakravarti, and Avirup Sil. 2019. Structured pruning of a bert-based question answering model. arXiv preprint arXiv:1910.06360.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? arXiv preprint arXiv:1905.10650.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When bert plays the lottery, all tickets are winning. arXiv preprint arXiv:2005.00561.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man’s bert: Smaller and faster transformer models. arXiv e-prints, pages arXiv--2004.

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.

Victor Sanh, Thomas Wolf, and Alexander M Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. arXiv preprint arXiv:2005.07683.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8815–8821.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. arXiv preprint arXiv:1908.09355.

Zhiquing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. arXiv preprint arXiv:2004.02984.