Pruning a BERT-based Question Answering Model

J.S. McCarley
IBM T.J. Watson Research Center
Yorktown Heights, NY
jsmc@us.ibm.com

Abstract
We investigate compressing a BERT-based question answering system by pruning parameters from the underlying BERT model. We start from models trained for SQuAD 2.0 and introduce gates that allow selected parts of transformers to be individually eliminated. Specifically, we investigate (1) reducing the number of attention heads in each transformer, (2) reducing the intermediate width of the feed-forward sublayer of each transformer, and (3) reducing the embedding dimension. We compare several approaches for determining the values of these gates. We find that a combination of pruning attention heads and the feed-forward layer almost doubles the decoding speed, with only a 1.5 f-point loss in accuracy.

1 Introduction
The recent surge in NLP model complexity has outstripped Moore’s law. \(^1\) (Peters et al., 2018; Devlin et al., 2018; Narasimhan, 2019) Deeply stacked layers of transformers (including BERT, RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019b), and ALBERT (Lan et al., 2019)) have greatly improved state-of-the-art accuracies across a variety of NLP tasks, but the computational intensity raises concerns in the cloud-computing economy. Numerous techniques developed to shrink neural networks including distillation, quantization, and pruning are now being applied to transformers.

Question answering, in particular, has immediate applications in real-time systems. Question answering has seen striking gains in accuracy due to transformers, as measured on the SQuAD (Rajpurkar et al., 2016) and SQuAD 2.0 (Rajpurkar et al., 2018) leaderboards. SQuAD is seen as a worst-case performance loss, for speed up techniques based on quantization, (Shen et al., 2019) while the difficulty of distilling a SQuAD model (compared to sentence-level GLUE tasks) is acknowledged in (Jiao et al., 2019). We speculate that these difficulties are because answer selection requires token level rather than passage level annotation, and the need for long range attention between query and passage.

In this paper we investigate pruning three aspects of BERT:
(1) the number of attention heads \(n_H\),
(2) the intermediate size \(d_I\),
(3) the embedding or hidden dimension \(d_E\).

The contributions of this paper are (1) application of structured pruning techniques to the feed-forward layer and the hidden dimension of the transformers, not just the attention heads, (2) thereby significantly pruning BERT with minimal loss of accuracy on a question answering task, considerable speedup, all without the expense of revisiting pretraining, and (3) surveying multiple pruning techniques (both heuristic and trainable) and providing recommendations specific to transformer-based question answering models.

Widely distributed pre-trained models consist of typically 12-24 layers of identically sized transformers. We will see that an optimal pruning yields non-identical transformers, namely lightweight transformers near the top and bottom while retaining more complexity in the intermediate layers.

2 Related work
While distillation (student-teacher) of BERT has produced notably smaller models, (Tang et al., 2019; Turc et al., 2019; Tsai et al., 2019; Yang et al., 2019a), the focus has been on sentence level annotation tasks that do not require long-range

\(^1\) ELMO: \(93 \times 10^6\), BERT: \(340 \times 10^6\), Megatron: \(8300 \times 10^6\) parameters
attention. Revisiting the pretraining phase during distillation is often a significant requirement. DistilBERT (Sanh et al., 2019) reports modest speedup and small performance loss on SQuAD 1.1. TinyBERT (Jiao et al., 2019) restricts SQuAD evaluation to using BERT-base as a teacher, and defers deeper investigation to future work.

Our work is perhaps most similar to (Fan et al., 2019), an exploration of pruning as a form of dropout. They prune entire layers of BERT, but suggest that smaller structures could also be pruned. They evaluate on MT, language modeling, and generation-like tasks, but not SQuAD. $L_0$ regularization was combined with matrix factorization to prune transformers in (Wang et al., 2019). Gale et al. (2019) induced unstructured sparsity on a transformer-based MT model, but did not report speedups. Voita et al. (2019) focused on linguistic interpretability of attention heads and introduced $L_0$ regularization to BERT, but did not report speedups. Kovaleva et al. (2019) also focused on interpreting attention, and achieved small accuracy gains on GLUE tasks by disabling (but not pruning) certain attention heads. Michel et al. (2019) achieved speedups on MT and MNLI by gating only the attention with simple heuristics.

3 Pruning transformers

3.1 Notation

The size of a BERT model is characterized by the values in table 1.

| notation | dimension | base | large |
|----------|-----------|------|-------|
| $n_L$    | layers    | 12   | 24    |
| $d_E$    | embeddings| 768  | 1024  |
| $n_H$    | attention heads | 12 | 16 |
| $d_I$    | intermediate size | 3072 | 4096 |

Figure 1: Notation: important dimensions of a BERT model

In each self-attention sublayer, we place a mask, $\Gamma_{\text{attn}}$ of size $n_H$ which selects attention heads to remain active. (section 3.2.2)

In each feed-forward sublayer, we place a mask, $\Gamma_{\text{ff}}$ of size $d_I$ which selects ReLU/GeLU activations to remain active. (section 3.3)

The final mask, $\Gamma_{\text{emb}}$, of size $d_E$, selects which embedding dimensions, (section 3.4) remain active. This gate is applied identically to both input and residual connections in each transformer.

3.3 Determining Gate Values

We investigate four approaches to determining the gate values.

(1) “random:” each $\gamma_i$ is sampled from a Bernoulli distribution of parameter $p$, where $p$ is manually adjusted to control the sparsity

(2) ”gain:” We follow the method of (Michel et al., 2019) and estimate the influence of each gate $\gamma_i$ on the training set likelihood $\mathcal{L}$ by computing the mean value of

$$g_i = \left| \frac{\partial \mathcal{L}}{\partial \gamma_i} \right|$$  \hspace{1cm} (1)

(“head importance score”) during one pass over the training data. We threshold $g_i$ to determine which transformer slices to retain.

(3) ”leave-one-out:” We again follow the method of (Michel et al., 2019) and evaluate the impact on devset $f$ score of a system with exactly one gate set to zero: Note that this procedure requires $n_L \times n_H$ passes through the data. We control the sparsity during decoding by retaining those gates for which $\delta f_i$ is large.

(4) “$L_0$ regularization:” Following the method described in (Louizos et al., 2017), during training time the gate variables $\gamma_i$ are sampled from a hard-concrete distribution (Maddison et al., 2017) parameterized by a corresponding variable $\alpha_i \in \mathbb{R}$. The task-specific objective function is penalized in proportion to the expected number instances of $\gamma = 1$. Proportionality constants $\lambda_{\text{attn}}, \lambda_{\text{ff}}, \lambda_{\text{emb}}$ in the penalty terms are manually adjusted to control the sparsity. We resample the $\gamma_i$ with each minibatch. We note that the full objective function is differentiable with respect to the $\alpha_i$ because of the reparameterization trick. (Kingma and Welling, 2014; Rezende et al., 2014) The $\alpha_i$ are updated by backpropagation for one training epoch with the SQuAD training data, with all other parameters held fixed. The final values for the gates $\gamma_i$ are obtained by thresholding the $\alpha_i$. 

| $n_L$ | layers | 12 | 24 |
| $d_E$ | embeddings | 768 | 1024 |
| $n_H$ | attention heads | 12 | 16 |
| $d_I$ | intermediate size | 3072 | 4096 |
3.4 Pruning

After the values of the $\gamma_i$ have been determined by one of the above methods, the model is pruned. Attention heads corresponding to $\gamma_{\text{attn}} = 0$ are removed. Slices of the feed forward linear transformations corresponding to $\gamma_{\text{ff}} = 0$ are removed. The pruned model no longer needs masks, and now consists of transformers of varying, non-identical sizes.

We note that task-specific training of all BERT parameters may be continued further with the pruned model.

4 Experiments

For development experiments (learning rate penalty weight exploration), and in order to minimize overuse of the official dev-set, we use 90% of the official SQuAD 2.0 training data for training gates, and report results on the remaining 10%. Our development experiments (base-qa) are all initialized from a SQuAD 2.0 system initialized from bert-base-uncased and trained on the 90% that provides a baseline performance of $75.0 f_1$ on the 10% dataset.  

Our validation experiments use the standard training/dev configuration of SQuAD 2.0. All are initialized from system that has an accuracy of $84.6 f_1$ on the official dev set. (Glass et al., 2019) (This model was initialized from bert-large-uncased.)

The gate parameters of the $L_0$ regularization experiments are trained for one epoch starting from the models above, with all transformer and embedding parameters fixed. The cost of training the gate parameters is comparable to extending fine tuning for an additional epoch. We investigated learning rates of $10^{-3}$, $10^{-2}$, and $10^{-1}$ on base-qa, and chose the latter for presentation and results on large-qa. This is notably larger than typical learning rates to tune BERT parameters. We used a minibatch size of 24, otherwise default hy-
Table 1: Decoding times, accuracies, and space savings achieved by two sample operating points on large-qa

| model       | time (sec) | f1    | attn-prune | ff-prune | size (MiB) |
|-------------|------------|-------|------------|----------|------------|
| no pruning  | 2605       | 84.6  | 0          | 0        | 1278       |
| attn₁       | 2253       | 84.2  | 44.3       | 0        | 1110       |
| ff₁         | 2078       | 83.2  | 0          | 47.7     | 909        |
| ff₁ + attn₁ | 1631       | 82.6  | 44.3       | 47.7     | 741        |
| ff₂ + attn₂ | 1359       | 80.9  | 52.6       | 65.2     | 575        |
| ff₂ + attn₂ + retrain | 1349   | 83.2  | 52.6       | 65.2     | 575        |

4.1 Accuracy as function of pruning

In figure 2 we plot the accuracy of base-qa $f_1$ accuracy as a function of the percentage of heads removed. As expected, the performance of "random" decays most abruptly. "Leave-one-out" and "Gain" are better, but substantially similar. "L₀ regularization" is best, allowing 50% pruning at a cost of 5 f-points.

Also in figure 3 we plot the accuracy $f_1$ accuracy of removing activations. We see broadly similar trends as above, except that the performance is robust to even larger pruning. "Leave-one-out" require a prohibitive number of passes ($n_L \times d_I$) through the data.

In figure 4 we plot the accuracy for removing embedding dimensions. We see that performance falls much more steeply with the removal of embedding dimensions. Attempts to train “L₀ regularization” were unsuccessfully - we speculate that the strong cross-layer coupling may necessitate a different learning rate schedule.

4.2 Validating these results

On the basis of the development experiments, we select operating points (values of $\lambda_{\text{attn}}$ and $\lambda_{\text{ff}}$) and train the gates of large-qa with these penalties. The decoding times, accuracies, and model sizes are summarized in table 1. Models in which both attention and feed forward components are pruned were produced by combining the independently trained gate configurations of attention and feed forward. For the same parameters values, the large model is pruned somewhat less than the small model. We also note that the $f_1$ loss due to pruning is somewhat smaller, for the same parameter values. We note that much of the performance loss can be recovered by continuing the training for an additional epoch after the pruning.

The speedup in decoding due to pruning the model is not simply proportional to the amount pruned. There are computations in both the attention and feed-forward part of each transformer layer that necessarily remain unpruned, for example layer normalization.

4.3 Impact of pruning each layer

In Fig. 5 we show the percentage of attention heads and feed forward activations remaining after pruning, by layer. We see that intermediate layers retained more, while layers close to the embedding and close to the answer were pruned more heavily.

5 Conclusion

We investigate various methods to prune transformer-based models, and evaluate the accuracy-speed tradeoff for this pruning. We find that both the attention heads and especially the feed forward layer can be pruned considerably with minimal loss of accuracy, while pruning the embedding/hidden dimension is ineffective because of a loss in accuracy. We find that $L₀$ regularization pruning, when successful, is considerably more effective than heuristic methods. We also find that pruning the feed-forward layer and the attention heads can be easily combined, and, especially after retraining, yield a considerably faster question answering model with minimal loss in accuracy.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805.
Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. CoRR, abs/1902.09574.

Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, G P Shrivatsa Bhargav, Dinsh Garg, and Avirup Sil. 2019. Span selection pre-training for question answering.

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.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. International Conference on Learning Representations.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. CoRR.

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.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. International Conference on Learning Representations.

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. CoRR, abs/1907.11692.

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

Sharp Narasimhan. 2019. Nvidia clocks worlds fastest bert training time and largest transformer based model, paving path for advanced conversational ai.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. CoRR, abs/1606.05250.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In Proceedings of the 31st International Conference on Machine Learning, volume 32 of Proceedings of Machine Learning Research, pages 1278–1286, Beijing, China. PMLR.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. CoRR, abs/1606.05250.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In Proceedings of the 31st International Conference on Machine Learning, volume 32 of Proceedings of Machine Learning Research, pages 1278–1286, Beijing, China. PMLR.

A Supplemental Material