Combining Improvements in the Compression of Large Language Models

Victor Kolev Daniel Longo Siddharth Sharma

Introduction

Large language models trained on massive text corpora have achieved state-of-the-art performance on a variety of NLP tasks. However, this comes at the cost of exponentially increasing their size. This raises several concerns, including their environmental impact, the engineering challenge and cost of training them, and the impracticality of their deployment in edge devices and other production environments. As seen in the table below, these massive language models are only growing larger in size. In fact, in just four years, model sizes have increased by 3 orders of magnitude.

Model	Organization	Date	Size (params)
ELMo	AI2	Feb 2018	94,000,000
GPT	OpenAI	June 2018	110,000,000
BERT	Google	Oct 2018	340,000,000
GPT-2	OpenAI	Mar 2019	1,500,000,000
Megatron-LM	NVIDIA	Sep 2019	8,300,000,000
T5	Google	Oct 2019	11,000,000,000
GPT-3	OpenAI	May 2020	175,000,000,000
Megatron-Turing NLG	Microsoft, NVIDIA	Oct 2021	530,000,000,000

Table 1. Increasing Model Sizes.

It is because of this that model compression for large language models has become a particularly relevant field of study.

Contributions

We summarize our contributions as follows:

- implemented weight pruning for decoder-only GPT-style models, pretrained with causal language modeling;
- implemented an architecture-agnostic implementation of Kronecker decomposition, with full integration with Huggingface API;
- created a generalized training procedure for running all three methods
- achieved a perplexity measure comparable to GPT-2 Medium (355M) with only 41M parameters ($> 8\times$ compression);
- derived theoretical intuition supporting the combination of the methods outlined.

References

[1] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263. PMLR, 2018.

[2] Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Kronecker decomposition for gpt compression. *arXiv preprint arXiv:2110.08152*, 2021.

[3] Habib Hajimolahoseini, Mehdi Rezagholizadeh, Vahid Partovinia, Marzieh Tahaei, Omar Mohamed Awad, and Yang Liu. Compressing pre-trained language models using progressive low rank decomposition.

[4] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[5] Sharath Nittur Sridhar and Anthony Sarah. Undivided attention: Are intermediate layers necessary for bert? *arXiv preprint arXiv:2012.11881*, 2020.

[6] Marzieh S Tahaei, Ella Charlaix, Vahid Partovi Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. Kroneckerbert: Learning kronecker decomposition for pre-trained language models via knowledge distillation. *arXiv preprint arXiv:2109.06243*, 2021.

[7] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7308–7316, 2019.

[8] Ofir Zafrir, Ariel Larey, Guy Boudoukh, Haihao Shen, and Moshe Wasserblat. Prune once for all: Sparse pre-trained language models. *arXiv preprint arXiv:2111.05754*, 2021.

Related Work

There currently exists a wide variety of compression methods, e.g. structured and unstructured pruning [8], progressive low-rank decomposition [3], undivided attention [5], and weight quantization [7, 4]. As part of our project, we surveyed the current standing of these techniques:

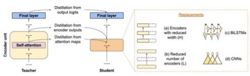

Figure 1. Model Distillation

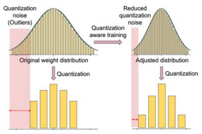

Figure 2. Model Quantization

Matrix decomposition has also been proposed as a technique for reducing attention computations. Compression techniques should aim to reduce model size while preserving accuracy.

Approach

We have identified 3 techniques which all, in their own way, decrease the number of model parameters in a way, such that performance is not significantly impaired. We seek to combine the improvements in a way that maximizes compression while maintaining good performance and high-quality internal representations. We outline the 4 methods below.

Pruning once and for all

This technique ([8]) introduces sparsity in the weight matrices of the model, so that it running it is less computationally expensive. It does so in 2 steps: (i) *pruning* weights (making them sparse) and performing knowledge distillation (matching outputs of the smaller pruned model with the outputs of the original); (ii) fine-tuning while keeping pruned weights at 0.

Kronecker Decomposition

Recall the definition of Kronecker product in eq. 1,

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix} \qquad (1) \qquad\qquad (\hat{A}, \hat{B}) = \underset{(A,B)}{\operatorname{argmin}} ||W - A \otimes B||_2^2 \qquad (2)$$

where $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, and $A \otimes B \in \mathbb{R}^{mp\times n}$. This technique ([6, 2]) computes and applies the Kronecker product to all weight matrices of the model. The Kronecker product is calculated for a weight matrix W by estimating the Kronecker factors $\hat{A}$ and $\hat{B}$ via the solution to the nearest Kronecker problem (eq. 2), which can be solved via SVD.

Progressive Low Rank Decomposition

Recall the definition of Singular Value Decomposition for a weight matrix $W \in \mathbb{R}^{C\times S}$,

$$W = U\Sigma V^\top = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$$

This method ([3]) progressively applies a low rank estimation ($W' = W_0 W_1$ where $W_0 = U'\sqrt{\Sigma'}$ and $W_1 = \sqrt{\Sigma'}V'^\top$) to the weight matrices of a transformer model, truncating smallest eigenvalues, and using knowledge distillation to restore performance losses.

Experiments

To facilitate combining the techniques above, we used the HuggingFace API to access and modify models. We specifically used the **distilGPT-2** model as a starter model for all of our experiments. The distilGPT-2 model was pretrained on the WikiText-103 corpus, and has ≈ 82M parameters.

Since, unlike other approaches, we start with an already compressed model, it cannot be expected to match the compression ratio that studies show with full-scale models. This, however, improved our training times and made training feasible on a single GPU with limited resources.

We evaluated our compression models (based on `distilGPT-2`) on the standard metric for decoder-only models, perplexity:

$$PP(p) := 2^{H(p)} = 2^{-\sum_x p(x)\log_2 p(x)} = \prod_x p(x)^{-p(x)}.$$

Theoretical Analysis

Recall that the stable rank of a matrix A, $\mathrm{rank_s}(A)$, is defined as the ratio in eq. 3, where the numerator is the Frobenius norm of A, and the denominator is the spectral norm. Note further that $\mathrm{rank_s}(A)$ is at most the rank of A, and hence the stable rank is intuitively understood as a continuous proxy to $\mathrm{rank}(A)$.

$$\mathrm{rank_s}(A) = \frac{\|A\|_F^2}{\|A\|_2^2} \qquad (3) \qquad\qquad \mathcal{O}\left(\prod_{i=1}^{d} \|W_i\|_2^2 \sum_{i=1}^{d} \mathrm{rank_s}(W_i)\right) \qquad (4)$$

Given that pruning directly decreases the Frobenius norm of the weights, we can infer that it decreases the stable rank as well (the spectral norm, i.e. largest eigenvalue, should not be changing under pruning, since knowledge distillation ensures that model outputs stay the same). Therefore, under extreme compression cases, low-rank decomposition and pruning would start interfering with one another, once the minimal rank is achieved.

In contrast with low-rank decomposition, Kronecker product is multiplicative with regards to the rank, and hence rank remains constant after Kronecker decomposition. Therefore, Kronecker should be able to fully integrate with both methods, and would aid computation in low-rank decompostition, as we would be calculating SVD on a much smaller matrix.

Lastly, examining recent results for generalization bounds [1], we see generalization error asymptotically bounded by the expression in eq. 4, which indicates that low-rank decomposition and pruning, by explicitly decreasing the stable rank, would yield better generalization results.

Results and Analysis

While the full evaluation of all methods is scope of future work (due to time and resource constrains), we have demonstrated theoretical intuition for the success of the methods in combination, and the improved generalization capabilities of the compressed models.

We evaluated the pruning method on GPT-2 with a pruning factor of 0.5 (meaning that the weights are 50% sparse) and achieved a **perplexity of 43.69%, which is comparable to GPT-2 Medium**, which has 335M parameters. In contrast, our pruned `distilGPT-2` model has only 41M non-zero parameters, therefore we have **a compression $> 8\times$**. This result is surprising, especially provided a training time less than 24 hours on a single GPU, and shows that LLMs are vastly overparametrized.

Looking more broadly, our theoretical analysis of pruning and low-rank estimators indicates that these methods provide models with tighter bounds on generalization error, which indicates better de facto generalization performance.

{sidshr, vkolev, ddl} @ stanford.edu