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 growing 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 5 orders of magnitude.

<table>
<thead>
<tr>
<th>Model</th>
<th>Organization</th>
<th>Date</th>
<th>Size</th>
<th>[parameters]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELMo</td>
<td>AI2</td>
<td>Feb 2018</td>
<td>94,000,000</td>
<td></td>
</tr>
<tr>
<td>GPT</td>
<td>OpenAI</td>
<td>June 2018</td>
<td>130,000,000</td>
<td></td>
</tr>
<tr>
<td>BERT</td>
<td>Google</td>
<td>Oct 2018</td>
<td>340,000,000</td>
<td></td>
</tr>
<tr>
<td>GPT-2</td>
<td>OpenAI</td>
<td>Mar 2019</td>
<td>1,500,000,000</td>
<td></td>
</tr>
<tr>
<td>Megatron-LM</td>
<td>NVIDIA</td>
<td>Sep 2019</td>
<td>6,300,000,000</td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>Google</td>
<td>Oct 2019</td>
<td>11,000,000,000</td>
<td></td>
</tr>
<tr>
<td>GPT-3</td>
<td>OpenAI</td>
<td>May 2020</td>
<td>175,000,000,000</td>
<td></td>
</tr>
<tr>
<td>Megatron-GPT-3 [SFG]</td>
<td>NVIDIA</td>
<td>Oct 2021</td>
<td>550,000,000,000</td>
<td></td>
</tr>
</tbody>
</table>

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 Kroner decomposition, with full integration with HuggingFace API;
- created a generalized training procedure for running all three methods;
- achieved post-training compressibility comparable to GPT-2 Medium (355M) with only 45M parameters (8x compression);
- derived theoretical intuition supporting the combination of the methods outlined.

References


Related Work

There currently exists a wide variety of compression methods, e.g., structured and unstructured pruning [8], progressive low-rank decomposition [3], untangled attention [1], and weight quantization [17, 18]. As part of our project, we surveyed the current standing of these techniques.

![Figure 1: Model Distillation](image1.png)

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

![Figure 2: Model Quantization](image2.png)

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 impacted. 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 if running it is less computationally expensive. It does so in 2 steps: (1) pruning weights (making them sparse) and performing knowledge distillation (matching outputs of the smaller pruned model with the outputs of the original) (2) fine-tuning while keeping pruned weights at 0. 

Kroner Decomposition

Recall the definition of Kroner product in eq. 1.

\[ A \otimes B = \begin{pmatrix} a_{11}b_{11} & \cdots & a_{1n}b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & \cdots & a_{mn}b_{mn} \end{pmatrix} \] (1)

where \( a \in \mathbb{R}^{m \times n} \) and \( b \in \mathbb{R}^{n \times r} \). This technique [5, 6] computes and applies the Kroner product to all weight matrices of the model. The Kroner product is calculated for a weight matrix \( W \) by first expanding the Kroner factors, \( A \otimes B \), and then by solving the linear Kroner product problem (2), which can be solved via SVD.

Progressive Low Rank Decomposition

Recall the definition of Singular Value Decomposition for a weight matrix \( W \in \mathbb{R}^{m \times n} \):

\[ W = U \Sigma V^T = \sum_{i=1}^{\min(m,n)} \sigma_i u_i v_i^T \] (2)

This method [10] progressively applies a low rank estimation \( W_k = U_k \Sigma_k V_k^T \) where \( W_k = \sigma_i u_i v_i^T \) and \( W_0 = \sum_{i=1}^{\min(m,n)} \sigma_i u_i v_i^T \) to the weight matrices of a transformer model, truncating smallest eigenvalues, and only keeping 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 skslGPT-2 model as a starter model for all of our experiments. The skslGPT-2 model was pretrained on the WikiText-103 corpus, and it has 16B 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 distillGPT-2) on the standard metric for decoder-only models, perplexity.

\[ P(y|x) = \exp \left( -\sum \log p(x_i|y) \right) \] (3)

Given that pruning directly decreases the FLOPs 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 some compression cases, low-rank decomposition and pruning would start interfering with one another, once the minimal rank is achieved.

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

Lastly, examining recent results for generalization bounds [3], 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 resources con-straining, 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 0.1 meaning that the weights are 90% sparsified and achieved a perplexity of 43.69%, which is comparable to GPT-2 Medium, which has 3.3B parameters. In contrast, our pruned distillGPT-2 models has only 43M non-zero parameters, therefore we have a compression \( > 8x \). This result is surprising, especially provided a training time less than 24 hours on a single GPU, and shows that LLMs are vastly overparameterized.

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.