Combining Improvements in the Compression of Large Language Models

Stanford CS224N Custom Project

Victor Kolev
Stanford University
vkolev@stanford.edu

Daniel Longo
Stanford University
ddl@stanford.edu

Siddharth Sharma
Stanford University
sidshr@stanford.edu

Abstract

While overparametrized large language models achieve state-of-the-art NLP performance, they raise concern about environmental impact, training cost, and feasibility of deployment in edge devices. Model compression promises to alleviate these issues, and multiple approaches have recently been successful in decreasing the number of parameters. We seek to determine whether these methods are complementary, and can be used in conjunction. Starting with a pretrained distilGPT-2 model, we apply progressive low-rank decomposition and weight pruning. Our engineering contributions include the implementation of weight pruning for decoder-based models, and the first architecture-agnostic implementation of low rank decomposition. We develop theoretical intuition for the proposed combinations of compression methods, revealing a deeper connection through matrix rank, and an impact on generalization error. We achieve a perplexity exceeding GPT-2 Medium (355M params) with only 37M parameters ($\approx 10\times$ compression).

1 Key Information to include

- Mentor: Eric Anthony Mitchell

2 Introduction

Transformer-based language foundation models 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 (Table 1), these massive language models are only growing larger in size. In fact, in just four years, model sizes have increased by four orders of magnitude.

It is because of this rapid growth in parameter count that compression of large language models has become a particularly relevant field of study. In this paper, we examine the interaction between existing techniques for the compression of large language models, and evaluate how they can most optimally be combined. There currently exists a wide variety of methods, e.g. Kronecker decomposition [Edalati et al., 2021, Tahaei et al., 2021], progressive low-rank decomposition [Hajimolahoseini et al.], undivided attention [Sridhar and Sarah, 2020], and weight quantization [Yang et al., 2019]. Specifically, we aim to investigate the question:

> What combination of approaches minimizes the performance degradation after compressing up to a certain ratio?

Starting with a pre-trained `distilGPT-2`, we applied various compression and sparsity techniques by first independently implemented the methods and interpolating the combinations of interest. We motivate the particular combinations of methods we chose in the approach section. Following the

Model	Organization	Date	Size (params)
ELMo	AI2	Feb 2018	94,000,000
GPT	OpenAI	June 2018	110,000,000
BERT	Googled	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.

application of compression methods, we evaluate the models for perplexity via the `HuggingFace` API. The perplexity is calculated via a sliding-window strategy.

Our contributions can be summarized in two categories:

Research

- *Quantitative compression results.* We show that our pruned distilGPT-2 achieves perplexity on par with GPT-2 Medium, while having $8\times$ less parameters
- *Theoretical Intuition.* We provide mathematical intuition and show that the compression techniques are complementary and achieve tighter bounds on generalization error, indicating better generalization performance.

Engineering

- *Pruning for GPT-2.* We provide the first pruning method for autoregressive, decoder-based models.
- *Architecture-agnostic Progressive Low Rank implementation.* We provide generalized implementations of Progressive Low Rank Decomposition with full integration with the HuggingFace API.

3 Related Work

The field of language model compression has seen greater research attention as models continue to grow in size. Given their resource-intensiveness, compression is a valuable asset in terms of model deployment. Li et al. [2020] showed that training a large model first then compressing to a smaller model is advantageous to training a smaller model when working with transformers.

Quantization It has been shown [Hubara et al., 2017] that the full 32-bit float precision in NNs is not always necessary. By applying quantization on weights of the model, float precision is reduced, which improves memory footprint and execution speed. An example of quantization in the context of large language models is with BERT: Fan et al. [2020] show that truncating the bits in the BERT weights to a certain bitwidth yields better compression at the expense of a drop in accuracy, which is referred to quantization noise.

Quantization-aware training (QAT) involves more steps to adjust weights when training with quantization. Figure (1) below summarizes naive quantization and the need for QAT.

Other forms of quantization dependent on QAT include fixed-length integer quantization [Zafrir et al., 2021, Boo et al., 2021], Hessian-based mixed-precision quantization [Shen et al., 2020], adaptive floating-point quantization [Tambe et al., 2020], and noise-based quantization [Fan et al., 2020].

Weight Pruning Pruning involves progressively setting the least meaningful weights in a model to zero, hence making the weight matrix sparse. This has many advantages from a computational perspective, since it allows the use of highly-efficient sparse matrix multiplication algorithms.

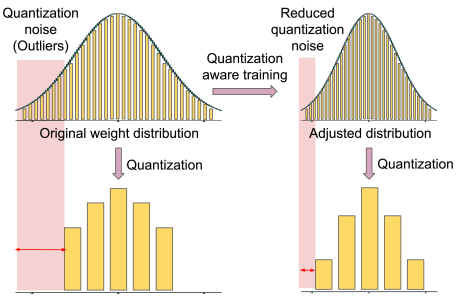

Figure 1: Quantization-aware training treats each matrix of weights as a Gaussian distribution and decreases the variance such that quantization noise is reduced. Ganesh et al. [2021]

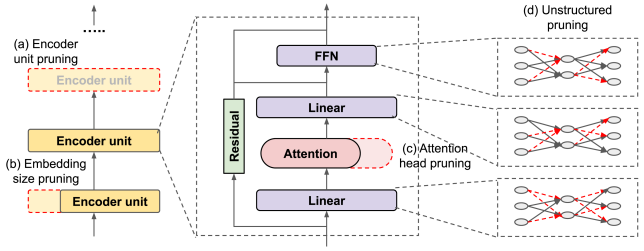

Figure 2: Structured pruning removes chunks of the weights, while unstructured pruning reduces weights via sparsity. Examples of structured pruning include embedding size pruning, attention head pruning, and encoder unit pruning. Ganesh et al. [2021]

Curiously, pruning is the foundation of the famous lottery ticket hypothesis [Frankle and Carbin, 2018].

Formally, pruning can be divided into two classes: structured and unstructured pruning. Structured pruning ablates structured blocks of weights. [Anwar et al., 2017] Unstructured pruning (also known as sparse pruning) identifies the set of the least important weights (smallest gradient) as proposed by Gordon et al. [2020]. A comparison of pruning methods can be found in the diagram on Figure (2).

Knowledge distillation Distillation involves transfer of knowledge from a large "teacher" model to a more compact "student" model. The student model is "fine-tuned" to match the internal representations and outputs of the teacher model on a particular task, so that both models have the same "knowledge". For example, DistilBERT is a smaller, faster, and cheaper version of $\text{BERT}_{\text{Base}}$ that runs 60% faster with with 40% less parameters while preserving 95% of performance on the GLUE benchmark.

Distillation during both pre-training and fine-tuning can help recover lost performance after some weight ablation. The compression methods we use in this work all share a knowledge distillation step.

Kronecker Decomposition Tahaei et al. [2021], Edalati et al. [2021]

Recall the definition of Kronecker product in eq. 1,

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

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{B} \in \mathbb{R}^{p \times q}$, and $\boldsymbol{A} \otimes \boldsymbol{B} \in \mathbb{R}^{mp \times n}$. This technique computes and applies Kronecker decomposition to all weight matrices of the model, which for a matrix W is computed by estimating the Kronecker factors $\hat{A}$ and $\hat{B}$ via the solution to the nearest Kronecker problem 2.

Upon some rearrangement of items in W, we can derive the solution to the nearest Kronecker problem using a rank-1 SVD approximation (view Van Loan and Pitsianis [1993] for additional details).

Kronecker decomposition can be used on a pretrained model by decomposing all weight matrices of a network, and then performing knowledge distillation to recover performance, propagating gradients to the Kronecker factors instead of the original matrix. For compression of LLMs, Kronecker decomposition uses a small matrix size (i.e. 2×1, 2×2) to compress matrices 2-4 times and impose explicit block structure in the weight matrices.

4 Methods

We have identified two 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 two methods below.

Pruning once and for all Zafrir et al. [2021]

This technique 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 and performing knowledge distillation; (ii) fine-tuning while keeping pruned weights at 0. Importantly, as the title suggests, the pruning step is done only once, which greatly alleviates resource cost for adapting to downstream tasks. Prune once for all also aims to keep the sparsity pattern of the sparse pre-trained model in place while fine-tuning. The authors employ a method known as pattern-lock: this technique prevents the zeros in the weights from changing while training the model.

Progressive Low Rank SVD decomposition Hajimolahoseini et al.

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

$$\boldsymbol{W} = \boldsymbol{U}\Sigma\boldsymbol{V}^\top = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$$

This method progressively applies a low rank estimation ($\boldsymbol{W}' = \boldsymbol{W}_0\boldsymbol{W}_1$ where $\boldsymbol{W}_0 = \boldsymbol{U}'\sqrt{\Sigma'}$ and $\boldsymbol{W}_1 = \sqrt{\Sigma'}\boldsymbol{V}'^\top$) to the weight matrices of a transformer model, truncating smallest eigenvalues. Knowledge distillation is applied after each rank reduction, such that model performance recovers.

5 Experiments & Results

5.1 Data and Evaluation

There are two datasets that we are applying in this work. The first is the WikiText corpus (2,500 million words). We use this dataset for both the pretraining and distillation processes. The second dataset that we use is wikitext-2 (100 million words). This corpus is used for evaluation, specifically to measure perplexity of our compressed decoder-based models. These datasets were chosen because they are standardized for use by HuggingFace and these integrate well with our distilGPT-2 model. For background, HuggingFace is the API that provides infrastructure and scripts to train and evaluate large language models.

5.2 Implementation

We build on an open-source framework for compression of language models [Zafrir et al., 2021], which implements Knowledge Distillation, Prune-once-and-for-all, and Quantization. However, the framework was designed for encoder-only BERT-style models.

As we are working with GPT-2, we reverse-engineered the framework and modified it to support pre-training with Causal Language Modelling, which is what decoder-based models use. Additionally, we created two add-ons to the framework, namely an implementation for Kronecker Decomposition and Progressive Low Rank Decomposition. It is worth noting that, to the best of our knowledge, this is the first public open-source implementation of the latter two methods. Our implementation further supports full `Huggingface` integration by using decorators to modify the behaviour of existing PyTorch and `Huggingface` methods.

Model	#parameters	Perplexity
GPT-2 Medium	355M	26.37
DistilGPT-2	82M	43.69
Pruning	41M	26.44
Low Rank	31M	22.00
Pruning + Low Rank	37M	24.56

Table 2: Perplexity is calculated from the test evaluation loss on the `wikitext-103` benchmark. The result for Low Rank is taken from the original paper, which did not provide code, training times, or hyperparameters.

We maintained a constant set of hyperparamters, which we set to match the ones in [Zafrir et al., 2021]. Pruning ratios and ranks for Low Rank were set according to the desired compression ratio. Models were deployed for training on a Azure NC6 Virtual Machine with a Nvidia Tesla V100. Training times for pre-training were highest for pruning, which took ≈ 30 hours to complete with the `DistilGPT-2` model. For the other Low Rank, training takes approximately 8 hours.[1]

5.3 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 $\mathrm{rank}(A)$.

$$\mathrm{rank_s}(A) = \frac{\|A\|_F^2}{\|A\|_2^2} \tag{3}$$

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

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. Additionally, this indicates that there exists some equilibrium state between sparsity and low-rank decomposition.

Lastly, examining recent results for generalization bounds Arora et al. [2018], 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.

5.4 Results & Discussion

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 2, 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 $\approx 8\times$.

When combined with Low Rank, we managed to improve the result both in terms of perplexity and parameter count, which supports the hypothesis that there exists an equilibrium state between the two. We were however still not able to match the performance of Low Rank from the original paper, however this could also be due to hyperparameter optimizations and longer training times.

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 is true in practice as well, since we are able to surpass GPT-2 Medium performance.

[1] Repository accessible at `https://github.com/victorkolev/gpt2-compression`

6 Conclusion & Future work

In conclusion, we have found that the combination of weight pruning and low rank decomposition can be advantageous in creating more compressed models that generalize better. More specifically, our work focused on decoder-based models such as GPT-2 and we were able to attain quantitative results despite high costs of training and limited computational resources. Additionally, we revealed a deeper connection between the two methods through matrix rank and stable rank, as detailed in our theoretical analysis section. The field of large language models is actively developing and we believe compression methods will be a key aspect of the deployment of such models.

Future work A future direction of development for this project is the integration of Kronecker Decomposition and combining it with the methods we investigated. Considering pruning with Kronecker decomposition, we hypothesize that this is a favorable compression method since we can impose explicit block structure (as in Narang et al. [2017]) while pruning. This structure would result in a sparse Kronecker matrix as well.

The second method combination of interest is progressive low rank decomposition with Kronecker decomposition – because Kronecker decomposition is set up to break the original matrix into a small (usually 2×2) matrix and a larger matrix, the latter is a prime candidate for low-rank decomposition. On the other hand, combining Kronecker with Low Rank is of interest, because there are deeper linear-algebraic relationships between Kronecker product and SVD Schacke [2004]. Concretely, Kronecker product is multiplicative with regards to eigenvectors and eigenvalues, and:

$$\boldsymbol{A} \otimes \boldsymbol{B} = (\boldsymbol{U}_A \Sigma_A \boldsymbol{V}_A^\top) \otimes (\boldsymbol{U}_B \Sigma_B \boldsymbol{V}_B^\top) = (\boldsymbol{U}_A \otimes U_B)(\Sigma_A \otimes \Sigma_B)(\boldsymbol{V}_A \otimes \boldsymbol{V}_B)^\top. \tag{5}$$

Note that the right-hand side is itself the singular value decomposition of $\boldsymbol{A} \otimes \boldsymbol{B}$, hence SVD can be computed much more efficiently on the smaller matrices.

References

Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *J. Emerg. Technol. Comput. Syst.*, 13(3), feb 2017. ISSN 1550-4832. doi: 10.1145/3005348. URL https://doi.org/10.1145/3005348.

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.

Yoonho Boo, Sungho Shin, Jungwook Choi, and Wonyong Sung. Stochastic precision ensemble: self-knowledge distillation for quantized deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6794–6802, 2021.

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.

Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. *arXiv preprint arXiv:2004.07320*, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. Compressing Large-Scale Transformer-Based Models: A Case Study on BERT. *Transactions of the Association for Computational Linguistics*, 9:1061–1080, 09 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00413. URL https://doi.org/10.1162/tacl_a_00413.

Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020.

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

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

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.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E Gonzalez. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *arXiv preprint arXiv:2002.11794*, 2020.

John X Morris, Eli Lifland, Jin Yong Yoo, and Yanjun Qi. Textattack: A framework for adversarial attacks in natural language processing. *Proceedings of the 2020 EMNLP, Arvix*, 2020.

Sharan Narang, Eric Undersander, and Gregory Diamos. Block-sparse recurrent neural networks. *arXiv preprint arXiv:1711.02782*, 2017.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Kathrin Schacke. On the kronecker product. *Master's thesis, University of Waterloo*, 2004.

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

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

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.

Thierry Tambe, En-Yu Yang, Zishen Wan, Yuntian Deng, Vijay Janapa Reddi, Alexander Rush, David Brooks, and Gu-Yeon Wei. Algorithm-hardware co-design of adaptive floating-point encodings for resilient deep learning inference. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

Charles F Van Loan and Nikos Pitsianis. Approximation with kronecker products. In *Linear algebra for large scale and real-time applications*, pages 293–314. Springer, 1993.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

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

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.