Kronecker Products and GPT2

Ayoub @ Uni Passau

March 10, 2024

Abstract

What do we do differently:

- We put distillation to test, and see if does improve improve the learning quality over vanilla supervised learning. **Spoiler alert**: NOPE. (Others, all use distillation)
- We don't compress the attention matrices.
- We try different compression schemes and see how far we can push down the size.
- We use multiple Kronecker Factors.

Contents

1 Resutls:

Early results on wikitext103, wikitext2 [1] and Lambada [2]. **KronyPT** is our model, and the suffixes 1350 and 3950 refer to different checkpoints. Both models are already outperforming distilGPT2 ([3]) on the 3 datasets. And the 3950 checkpoint is outperforming **KnGPT2** ([4]) on Lambada, while slightly still behind **TQCompressedGPT2** [5]. We measure perplexity (duh) using the Hugging Face interface that was used by the other papers.

		Datasets		
# Params	Model	wikitext-103	wikitext-2	Lambada
124M	GPT2	29.16	24.67	45.28
82M	DistilGPT2	44.53	36.48	76.00
81M	**KronyPT-81M-1350**	41.98	34.99	-
81M	**KronyPT-81M-3950**	-	-	64.92
81M	TQCompressedGPT2	40.28	32.25	64.72
81M	KnGPT-2 (Huawei)	40.97	32.81	67.62

Table 1: Comparison of different models on various datasets.

2 Introduction

A lot of work has been done on the compression of LLMs using factorization methods, including Kronecker Products. Most notably for encoder based based, and rarely applied to decoder based methods ([4], [6], [5]).

We study how far we can push the compression ratio, and record then impact of down-scaling on the quality of the model. We also investigate adding multiple Kronecker factors.

3 Training Set-up

Add here the basic set-up that I used to train the latest models, basically Chinchilla recommendations. (for later)

4 Kronecker Decomposition

In this section, we study different Kronecker decomposition setups, and the percentage of compression it would lead to. So far we only decompose the weights `c_fc.weight` and `c_proj.weight` (each has $2.3M$ in the original GPT2-small architecture.). Each transformer layer (there are 12 in total) has two of these weights. They count to $56.6M$ in total (45% of GPT2 $124M$). Hence any significant reduction to these matrices would lead to a remarkable compression ratio of the whole model. We choose not to compress the other weights, namely, attention weights and embedding matrix for various reasons that we will expose later on.

4.1 The 95M model

The most basic strategy is to divide one of the dimensions of each W by 2, this would lead to a 95M model. The parameter $p_1 =$ `c_fc.weight` (resp. $p_2 =$ `c_proj.weight`) has a shape of $(3072, 768)$ (resp. $(768, 3072)$). We first try the following decomposition: $p_1 = \underbrace{W_{11}}_{(3072,384)} \otimes \underbrace{W_{12}}_{(1,2)}$ and $p_2 = \underbrace{W_{21}}_{(384,3072)} \otimes \underbrace{W_{22}}_{(2,1)}$

This decomposition would lead to to reduction of $28M$ (50%). The new network would have approx $95M$. Our goal is to eventually reach the $82M$ mark, similar to DistilGPT2, and other Factorized models (inserts other refs here).

4.2 Different decomposition schemes:

It is reasonable to aim for a decomposition that guarantees the maximum rank we can get. Since the Rank of Kronecker products is multiplicative, i.e., $\text{rank}(A \otimes B) = \text{rank}(A) \cdot \text{rank}(B)$ [1], we can easily compute the rank of each possible decomposition. In our case, we have $W \in R^{(m,n)}$ where $m = 3072$ and $n = 768$. Hence, for each layer of GPT2, we aim to find the "optimal" $A \in R^{(m_1,n_1)}$, and $B \in R^{(m_2,n_2)}$, i.e.,:

$$W \approx A \otimes B, \qquad m = m_1 m_2, n = n_1 n_2$$

.

W.l.o.g, for each decomposition (A, B) the maximum rank we can reach is $\min(m_1, n_1) \cdot \min(m_2, n_2)$. And each of the reduced decompositions would have exactly $m_1 n_1 + m_2 n_2$ parameters. Hence, theoritically, the maximum rank we can get is 768 of a $(3072, 768)$ matrix. The following table summarizes some possible combinations, alongside the reduction it would lead to per layer, and the total number of parameters in GPT2, for only those decompostions of maximal attainable rank. We are particularly interested in 3 class of models, the 67M, the 81M and the 96M. (Need to add this) Furthermore, we add multiple factors to the models labeled with **MF** (second table / a few decompositions are missing, check this out. e.g., 3072, 384).

[1] Link to proof: https://math.stackexchange.com/questions/707091/elementary-proof-for-textrank-lefta-otimes-b-right-textranka-cdot

Name	Dimension	params	Model size
67M	(64, 32)	3200	67,893,504
	(64, 48)	3840	67,908,864
	(96, 32)	3840	67,908,864
	(64, 64)	4672	67,928,832
	(128, 32)	4672	67,928,832
	(96, 48)	5120	67,939,584
	(96, 64)	6528	67,973,376
	(128, 48)	6528	67,973,376
	(128, 64)	8480	68,020,224
	(96, 96)	9472	68,044,032
MF1	(192, 48)	9472	68,044,032
	(128, 96)	12480	68,116,224
	(192, 64)	12480	68,116,224
	(128, 128)	16528	68,213,376
MF2	(256, 64)	16528	68,213,376
	...	...	...
MF3	(1024, 256)	262153	74,108,376
	(768, 384)	294920	74,894,784
	(1024, 384)	393222	77,254,032
81M	(768, 768)	589828	81,972,576
	(1536, 384)	589828	81,972,576
	(1024, 768)	786435	86,691,144
96M	(1536, 768)	1179650	96,128,304
GPT2	(3072, 768)	2359297	124,439,832

Table 2: Different compression schemes

Name	Dimension	params	1 factor	2 factors	3 factors
MF1	(192, 48)	9472	68,044,032	68,044,032	68,044,032
MF2	(256, 64)	16528	68,213,376	68,213,376	68,213,376
MF3	(1024, 256)	262153	74,108,376	74,108,376	74,108,376

Table 3: Adding multiple Kronecker Factors

5 How small can we go?

In this section, we study the impact of down-scaling on the compressed models, we train 4 different architectures, with variant dimensions:

- 67M: diverges // try with higher batch size (the least we can get)
- 81M: Super.
- 81M: with a Variant scheme (67M with multiple factors / 1025-256 with two factors)

- 95M: duh duh duh. (the highest we can get)

Keep in mind:

- You can probably stabilize training using various tricks. It's just an endless loop. We are not going to play the What-IF game. One single config, that's it.
- One could try to have a mixed strategy:
  - higher levels of compression in the early layers.
  - higher levels of compression in the late layers.
  - Every odd layer
  - In the middle

6 Initialization

Since we inherit a GPT2 checkpoint that was trained for multiple epochs on the Open Web Text (OWT) (cite here), we want to initiliaze our weights in a way that leverages the old pre-training as much as possible. This of course obvious for the parameters that are common between **GPT2** and **KronyPT** (i.e., exact match). But more tricky for the weights that are decomposed into Kronecker Factors. In our work, we try two different approaches, Van Loan decomposition (cite here), and a Pruning, exclusively for the 95M model.

6.1 Van Loan Method

Given matrices $A \in R^{m \times m}$ and $B \in R^{n \times n}$, the Kronecker product is defined as $A \otimes B = [a_{ij}B] \in R^{mn \times mn}$. The Van Loan method computes a matrix function, denoted as $f(A \otimes B)$, by exploiting eigenvalue or singular value decompositions. Specifically, if f is a matrix exponential, then $e^{A \otimes B} = e^A \otimes e^B$. This method reduces computational complexity and is especially advantageous in large-scale matrix computations.

6.2 Pruning based Initialization

XX

References

[1] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.

[2] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández, "The lambada dataset: Word prediction requiring a broad discourse context," *arXiv preprint arXiv:1606.06031*, 2016.

[3] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," in *NeurIPS* $EMC^2 Workshop$, 2019.

[4] M. Tahaei, E. Charlaix, V. Nia, A. Ghodsi, and M. Rezagholizadeh, "Kroneckerbert: Significant compression of pre-trained language models through kronecker decomposition and knowledge distillation," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2116–2127, 2022.

[5] V. Abronin, A. Naumov, D. Mazur, D. Bystrov, K. Tsarova, A. Melnikov, I. Oseledets, S. Dolgov, R. Brasher, and M. Perelshtein, "Tqcompressor: improving tensor decomposition methods in neural networks via permutations," *arXiv preprint arXiv:2401.16367*, 2024.

[6] A. Edalati, M. Tahaei, A. Rashid, V. P. Nia, J. J. Clark, and M. Rezagholizadeh, "Kronecker decomposition for gpt compression," *arXiv preprint arXiv:2110.08152*, 2021.