ProcrustesGPT: Compressing LLMs with Structured Matrices and Orthogonal Transformations

Ekaterina Grishina, Mikhail Gorbunov, Maxim Rakhuba
HSE University
Correspondence: er.grishina@yandex.ru

Abstract

Large language models (LLMs) demonstrate impressive results in natural language processing tasks but require a significant amount of computational and memory resources. Structured matrix representations are a promising way for reducing the number of parameters of these models. However, it seems unrealistic to expect that weight matrices of pretrained models can be accurately represented by structured matrices without any fine-tuning. To overcome this issue, we utilize the fact that LLM output is invariant under certain orthogonal transformations of weight matrices. This insight can be leveraged to identify transformations that significantly improve the compressibility of weights within structured classes. The proposed approach is applicable to various types of structured matrices that support efficient projection operations. Code is available at: https://github.com/GrishKate/ProcrustesGPT.

1 Introduction

Large language models have achieved remarkable success in language processing and are widely used in a variety of applications, but their deployment is still challenging, as these models hardly fit into a single GPU and require much computational resources during the inference and training processes. The research community is actively seeking efficient algorithms to reduce the size of pretrained models without sacrificing accuracy.

One approach that has not been fully explored is the use of structured matrices, which can potentially not only reduce the number of parameters in the model but also speed up computations. Low-parametric matrix decomposition can be applied directly to the weight matrices, minimizing the difference between the original and decomposed weights, i.e., $\|W - W'\| \to \min_{W' \in \mathcal{S}}$, where $\mathcal{S}$ is the low-parametric matrix class. Such a factorization may appear to be a reasonable initial guess for further fine-tuning. Unfortunately, it often leads to high approximation errors for any $\mathcal{S}$ when no additional training is performed, as there are no restrictions on the structure of the weight matrices during pretraining.

To overcome this issue, we utilize invariance of the network output under certain orthogonal transformations, which was first observed in (Ashkboos et al., 2024). In particular, for each layer, we aim to find such transformations that lead to the best compressibility within the chosen matrix class $\mathcal{S}$. In our paper, we focus on the sum of Kronecker product representation and $\mathcal{GS}$-matrices (Gorbunov et al., 2024; Dao et al., 2022), but other representations are also possible. Finding an optimal orthogonal transformation is a known linear algebra problem and is called the *orthogonal Procrustes problem*. The resulting framework is formulated as an optimization problem on the weights of the pretrained network, is free from the need for fine-tuning and applicable for different structured matrix representations. In Figure 1, we present relative errors in the Frobenius norm for different layers with and without using orthogonal transformations. We observe a noticeable increase in compression ability thanks to optimally chosen orthogonal transformations.

The main contributions of our work:

- We propose a new fine-tuning free framework for compressing LLM weights. The framework utilizes orthogonal transformations of the network weight matrices to ensure their compressibility using a low-parametric representation of choice.
- We formulate this framework as an optimization problem and develop efficient numerical methods to solve it.
- We conduct experiments with OPT (Zhang et al., 2022) and Llama2 (Touvron et al., 2023) models. We show that in most scenarios, our

Findings of the Association for Computational Linguistics: ACL 2025, pages 26937–26949
July 27 - August 1, 2025

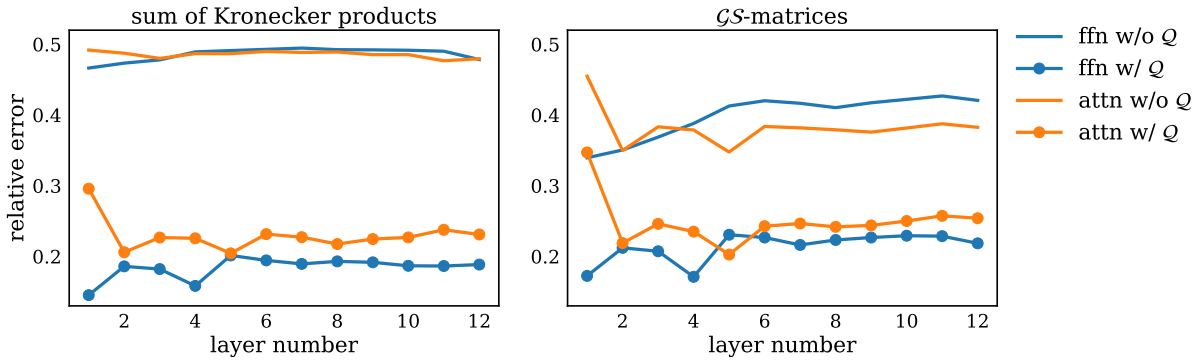


Figure 1: Illustration of compressibility of different layers of OPT-125m with and without applying orthogonal transformations $\mathcal{Q}$. We consider two types of structured representations: sum of Kronecker products (Section 4.1) and $\mathcal{GS}$-matrices (Section 4.2). Both representations result in approximately 25% compression for each compressed matrix within the layers.

approach yields more accurate results than alternative fine-tuning-free methods at comparable compression rates (in the range from 14% to 36%).

2 Related work

Existing approaches to neural network compression can be divided into four categories: quantization, distillation, pruning, and matrix factorization. Matrix factorization is a promising and relatively underexplored technique, and the most widely used approach within this category is a low-rank approximation. The work (Sharma et al., 2023) has shown that by carefully selecting ranks for each of the weights, the performance of LLMs can be improved on some tasks. However, direct application of SVD to uniform compression of weights leads to poor performance because weights usually have high ranks (Yu and Wu, 2023). Instead, the authors (Wang et al., 2024; Hsu et al., 2022; Yuan et al., 2023; Yu and Wu, 2023; Ji et al., 2024; Chen et al., 2021) use calibration datasets or Fisher information matrix to approximate activations. Other works explore low-rank plus sparse (Li et al., 2023) and low-rank plus quantized decompositions (Saha et al., 2024).

The works (Tahaei et al., 2021; Edalati et al., 2021) have been among the first to apply the Kronecker-product decomposition for the compression of BERT and GPT2. The paper (Abronin et al., 2024) proposes to enhance the performance of the Kronecker decomposition by adding permutations. These works compress the weight matrices directly and require knowledge distillation or fine-tuning in order to recover performance.

The authors of ModeGPT (Lin et al., 2024) propose to split the weight matrices of the transformer into pairs and jointly compress them using SVD, CR or Nyström approximation. This method is training-free and efficiently preserves model performance; however, it does not address compression of embedding and head matrices.

Many studies have investigated embedding compression, but most of the proposed algorithms require additional training. The works (Xu et al., 2023; Hrinchuk et al., 2020) apply the Tensor Train decomposition to the embedding layer, which offers strong compression, but requires training from scratch or hinders model performance. Other methods (Lioutas et al., 2020; Acharya et al., 2019) approximate embedding with low-rank decomposition, but these methods require fine-tuning. GroupReduce (Chen et al., 2018) utilizes knowledge about words occurrence by weighing tokens with their frequencies and applies block low-rank approximation.

Our goal in this study is to maintain model performance after weight factorization without additional training. The authors of the prunning approach SliceGPT (Ashkboos et al., 2024) have introduced the concept of computational invariance, meaning that orthogonal transformations can be applied to transformer weights without changing the output of the model. SliceGPT uses invariance to project layer activations onto their principal components and remove columns or rows from weight matrices. In this work, we utilize computational invariance to find a better approximation of the weights with structured representations.

3 Our approach

Our approach consists in iteratively finding orthogonal transformations $\mathcal{Q}$ to obtain optimal compression properties of the matrix weights. For brevity, we call these orthogonal transformations "rotations", although they do not necessarily have determinants equal to one. In Section 3.1, we present the concept of rotational invariance. Then, in Section 3.2, we present our approach as an optimization formulation to be solved. Notably, SliceGPT (Ashkboos et al., 2024) appears to be a particular instance of this formulation, which we also discuss further in Section 4.3.

3.1 Rotational invariance

In this section, we introduce notation and explain the concept of rotational invariance (Ashkboos et al., 2024). The transformer architecture consists of the repeated series of multi-head self-attention (MHSA) blocks and feed-forward network (FFN) blocks. Between these blocks, there is the LayerNorm or RMSNorm block. The RMSNorm normalizes the input vector:

$$\mathrm{RMSNorm}(x) = \frac{x}{\|x\|_2}.$$

LayerNorm is a linear transformation of RMSNorm and a network with LayerNorm can be easily transformed to a network with RMSNorm (Ashkboos et al., 2024).

Each of the MHSA and FFN blocks consists of input linear mappings, an activation function, and an output linear mapping. For example, the MHSA block first obtains queries, keys and values (XW_q, XW_k, XW_v) through a linear projection of the input X. Then these matrices are nonlinearly transformed and multiplied by an output weight matrix (W_o). If we denote the stacked weight matrices of the input linear mappings of each block as W_{in} and the weight matrix of the output linear mapping as W_{out}, then we can write:

$$\begin{aligned}\mathrm{MHSA}(X) &= \sigma(X[W_q, W_k, W_v])W_o + b = \\ &= \sigma(XW_{in})W_{out} + b,\end{aligned}$$

where σ is multi-head attention operation. Similarly, the FFN block can be written as

$$\mathrm{FFN}(X) = \sigma(XW_{in} + b_{in})W_{out} + b_{out},$$

where σ denotes an element-wise activation function, e.g. ReLU.

Let Q be an orthogonal matrix, which is a square matrix satisfying $Q^TQ = I$. It is well known that the Frobenius norm of a matrix $\|X\|_F^2 = \sum_{i,j} x_{ij}^2$ is invariant under orthogonal transformations of X, i.e., for any orthogonal Q_1 and Q_2:

$$\|Q_1XQ_2\|_F = \|X\|_F. \tag{1}$$

Using this invariance and $Q^TQ = I$, we may write

$$\begin{aligned}&\left(\frac{XW_{out} + X_{skip}}{\|XW_{out} + X_{skip}\|_F}\right)W_{in} = \\ &= \left(\frac{XW_{out}Q + X_{skip}Q}{\|XW_{out}Q + X_{skip}Q\|_F}\right)(Q^TW_{in}),\end{aligned} \tag{2}$$

where X_{skip} comes from the skip connection. As a result, we modify the skip connections to apply Q to the input of the RMSNorm (X_{skip}) and Q^T to the output of RMSNorm (the part of (2) in brackets) to keep the model unchanged.

3.2 Optimization problem formulation

We aim to improve the compression of the model weights via structured matrices by utilizing rotational invariance. In essence, for a given structure of layers, we want to find the rotations that complement well with the chosen structure, then rotate the network and project rotated weights on the structured matrix layers, with as little degradation of performance as possible. For each layer, our objective is thus minimizing the L2-difference between the outputs of the rotated network and the outputs of the compressed network on the calibration dataset.

For the ease of presentation and prior to delving into a detailed motivation, let us first articulate the final formulation of the optimization problem for the ℓ-th layer:

$$\begin{aligned}&\|X_{out}^\ell(W_{out}^\ell Q_\ell - \widehat{W}_{out}^\ell)\|_F^2 + \\ &+ \lambda_{in}\|X_{in}^\ell(W_{in}^\ell - Q_\ell\widehat{W}_{in}^\ell)\|_F^2 \to \min_{\substack{Q_\ell^TQ_\ell = I, \widehat{W}_\alpha^\ell \in \mathcal{S}_\alpha, \\ \alpha \in \{in, out\},}}\end{aligned} \tag{3}$$

where $\mathcal{S}_{in}$ and $\mathcal{S}_{out}$ are structured matrix classes that are utilized for compression. Despite the seeming simplicity, it is a nontrivial nonconvex optimization problem. We propose to approach it by alternatingly optimizing between Q_ℓ and $\widehat{W}_{in}^\ell, \widehat{W}_{out}^\ell$. Individual optimization problems are respectively a Procrustes problem for finding optimal orthogonal matrix and a projection step in a weighted norm for low-parametric representations. Importantly, problems for each layer are independent from each other and can be solved in parallel. We further discuss how we tackle this optimization problem in Section 5.

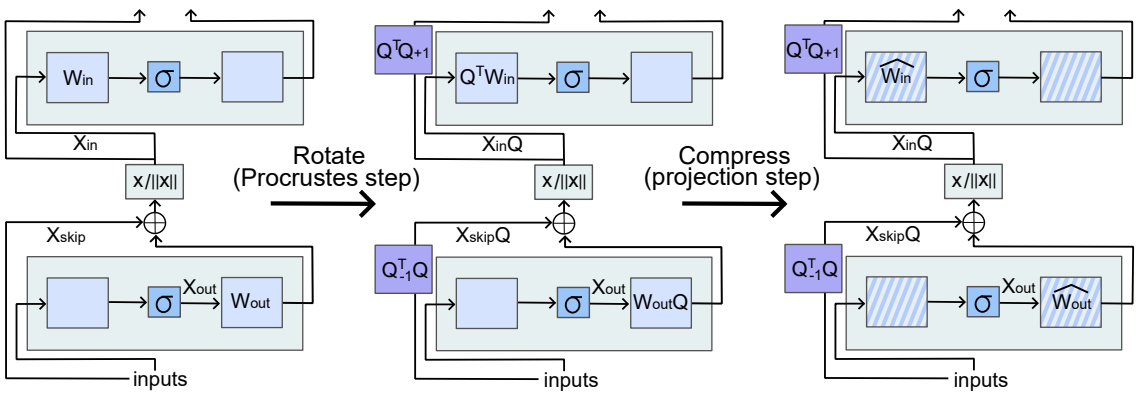


Figure 2: Illustration of the process of compression of a single transformer layer.

3.3 Motivating the optimization formulation

Let us discuss the motivation for (3) with a more general approach. Besides giving us motivation, it also uncovers the connection to the pruning approach of (Ashkboos et al., 2024). As discussed in Section 3.1, by utilizing rotation invariance, one can transform different layers of the network with different rotations without changing the network outputs. However, orthogonal matrices arise in skip connections after the "rotation" step (see Figure 2), and it can be fruitful for future work to additionally compress them.

We denote the weights of the network with the applied set of rotations $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_L\}$ as $W_{in}^{\mathcal{Q},\ell}, W_{out}^{\mathcal{Q},\ell}$ and the intermediate corresponding inputs as $X_{in}^{\mathcal{Q},\ell}, X_{out}^{\mathcal{Q},\ell}, X_{skip}^{\mathcal{Q},\ell}$ (see Figure 2). Let us denote the input of RMSNorm as

$$f_{out}^{\mathcal{Q},\ell}(X_{skip}^{\mathcal{Q},\ell}) = X_{out}^{\mathcal{Q},\ell} W_{out}^{\mathcal{Q},\ell} + X_{skip}^{\mathcal{Q},\ell} Q_{\ell-1}^T Q_\ell,$$

and the output of the linear mapping following RMSNorm as

$$f_{in}^{\mathcal{Q},\ell}(X_{in}^{\mathcal{Q},\ell}) = X_{in}^{\mathcal{Q},\ell} W_{in}^{\mathcal{Q},\ell}.$$

For the l-th rotation, objective can be written as

$$\|f_{out}^{\mathcal{Q},\ell}(X_{skip}^{\mathcal{Q},\ell}) - \widehat{f}_{out}^{\ell}(X_{skip}^{\mathcal{Q},\ell})\|_F^2 + \\ + \lambda_{in}\|f_{in}^{\mathcal{Q},\ell}(X_{in}^{\mathcal{Q},\ell}) - \widehat{f}_{in}^{\ell}(X_{in}^{\mathcal{Q},\ell})\|_F^2 \rightarrow \min_{\substack{\widehat{f}_{in}^{\ell}, \widehat{f}_{out}^{\ell}, \\ Q_\ell^T Q_\ell = I}}$$

Utilization of rotations affects matrices $W_{in}^{\ell}, W_{out}^{\ell}$, while also adding extra matrix $Q_{\ell-1}^T Q_\ell$ in the skip connection. We aim to compress rotated weights $W_{in}^{\mathcal{Q},\ell}, W_{out}^{\mathcal{Q},\ell}$ while also having the possibility of compressing matrices $Q_{\ell-1}^T Q_\ell$ into a separate structured matrix $\widehat{W}_{skip}^{\ell}$. Then, our objective becomes:

$$\lambda_{in}\|X_{in}^{\mathcal{Q},\ell} W_{in}^{\mathcal{Q},\ell} - X_{in}^{\mathcal{Q},\ell} \widehat{W}_{in}^{\ell}\|_F^2 + \\ \|(X_{out}^{\mathcal{Q},\ell} W_{out}^{\mathcal{Q},\ell} + X_{skip}^{\mathcal{Q},\ell} Q_{\ell-1}^T Q_\ell) - \\ (X_{out}^{\mathcal{Q},\ell} \widehat{W}_{out}^{\ell} + X_{skip}^{\mathcal{Q},\ell} \widehat{W}_{skip}^{\ell})\|_F^2 \rightarrow \min_{\substack{Q_\ell^T Q_\ell = I, \widehat{W}_\alpha^\ell \in \mathcal{S}_\alpha, \\ \alpha \in \{in, out, skip\}}}$$

Which can be rewritten as

$$\|X_{out}^{\ell}(W_{out}^{\ell} Q_\ell - \widehat{W}_{out}^{\ell}) + \\ X_{skip}^{\ell}(Q_\ell - Q_{\ell-1}\widehat{W}_{skip}^{\ell})\|_F^2 + \\ \lambda_{in}\|X_{in}^{\ell}(W_{in}^{\ell} - Q_\ell \widehat{W}_{in}^{\ell})\|_F^2 \rightarrow \min_{\substack{Q_\ell^T Q_\ell = I, \widehat{W}_\alpha^\ell \in \mathcal{S}_\alpha, \\ \alpha \in \{in, out, skip\}}}$$

We can approximate this further by abandoning the compression of $\widehat{W}_{skip}^{\ell}$, setting it to be equal to $Q_{\ell-1}^T Q_\ell$. This way we arrive at (3), and optimization problem for the ℓ-th layer becomes independent of the solution for the previous layers. This allows solving problems for different layers in parallel. Additionally, experiments have shown that balancing the terms in (3) using $\lambda_{in} = \frac{\|X_{out} W_{out}\|^2}{\|X_{in} W_{in}\|^2}$ improves the quality compared to $\lambda_{in} = 1$ (see Table 1).

4 Structured matrix representations

In this section, we present different structured matrix representations on which we focus in our work. The sum of Kronecker products yields the most consistent accuracy gain within different models. Although the $\mathcal{GS}$-matrix representation resulted in slightly lower accuracy overall, its computationally efficient structure offers significant potential to accelerate inference. Finally, we examine matrices with zero blocks, revealing connections with pruning techniques (Ashkboos et al., 2024).

4.1 Kronecker products

Definition 4.1. Given matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, the Kronecker product $A \otimes B$ is the $pm \times qn$ block matrix:

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}.$$

One Kronecker product is a very restrictive structure, and one usually considers the sum of $r > 1$ Kronecker products for more accurate results. Fortunately, the problem of obtaining the best approximation within such a structure (projection operation):

$$\left\| W - \sum_{i=1}^{r} A_i \otimes B_i \right\|_F^2 \to \min_{A_i, B_i}, \tag{4}$$

has an analytical solution that can be obtained using SVD (Golub and Van Loan, 2013), see Algorithm 1. For better results, we need to use the weighted Frobenius norm (3). The Kronecker product approximation problem with the weight matrix X reads as:

$$\left\| X \left(W - \sum_{i=1}^{r} A_i \otimes B_i \right) \right\|_F^2 \to \min_{A_i, B_i}. \tag{5}$$

Although it does not admit a simple solution, the SVD-based solution of (4) can be used for initialization for the iterative process. As an iterative procedure, we optimize (5) alternatively with respect to $\{A_i\}$ and $\{B_i\}$. Each of the alternating subproblems is solved exactly, details are presented in Appendix A.

Algorithm 1 SVD-based solution to (4).

Input: $W \in \mathbb{R}^{mp\times nq}$, rank r.
Output: $\{A_i\}$, $\{B_i\}$ from (4).
$W_r = W$.`rearrange((m p) (n q) →`
`(m n) (p q))`
$USV^T = \texttt{SVD}(W_r)$
$A' = U[:, :r]S[:r, :r]^{1/2}$
$B' = S[:r, :r]^{1/2}V[:, :r]^T$
$A = A'$.`rearrange('(m p) r → r m p')`
$B = B'$.`rearrange('(n q) r → r n q')`
return A, B

4.2 $\mathcal{GS}$matrices

Definition 4.2. $\mathcal{GS}$-matrices are matrices that can be represented in the form $P_L(LPR)P_R$, where L, R are block-diagonal matrices and P_L, P, P_R are permutation matrices.

This class of matrices (Gorbunov et al., 2024) generalizes Monarch (Dao et al., 2022) matrices and describes matrices with low-rank blocks up to a permutation of rows and columns. Thanks to this property, the projection step

$$\|W - P_L(LPR)P_R\|_F^2 \to \min_{L,R}$$

can be performed efficiently using an SVD-based procedure described in (Gorbunov et al., 2024). Likewise for the Kronecker decomposition, the weighted approximation problem

$$\|X(W - P_L(LPR)P_R)\|_F^2 \to \min_{L,R}$$

does not admit a simple solution. Nevertheless, it can still be solved numerically by alternating iterations with respect to L and R, see Appendix B.

4.3 Matrices with zero blocks and relation to SliceGPT

Another structured class one could consider is block-sparse matrices, which include matrices with a single nonzero block. For example, this includes matrices of the forms:

$$\begin{pmatrix} W & 0 \end{pmatrix}, \quad \begin{pmatrix} W \\ 0 \end{pmatrix}, \quad \begin{pmatrix} W & 0 \\ 0 & 0 \end{pmatrix}.$$

Such structures are not frequently used due to their simplicity and poor expressivity. Nevertheless, when paired with rotations, they can become a useful representation. Interestingly enough, we find that utilizing these classes and solving our objective have some relation to the SliceGPT method.

Proposition 4.3. *Let $\lambda_{in} = 0$. Let $\mathcal{S}_{\text{in}}$ and $\mathcal{S}_{\text{skip}}$ be matrices with zero columns. Then, solving our objective is equivalent to finding the rotation of the SliceGPT method and column slicing of W_{out} and $Q_{\ell-1}^T Q_\ell$. Row slicing of W_{in} and $Q_{\ell-1}^T Q_\ell$ arises naturally due to the sparse structure of inputs.*

Proof. See Appendix C. □

5 Optimization algorithm

5.1 Orthogonal Procrustes problem

The problem of finding an orthogonal matrix Q, which most closely fits a given matrix A to B, is called an orthogonal Procrustes problem (OPP):

$$\|QA - B\|_F \to \min_{Q^TQ=I}.$$

It was first solved in the work (Schönemann, 1966). The solution is $Q = UV^T$, where U and V^T are from the SVD: $BA^T = U\Sigma V^T$.

The extension of the Procrustes problem, where an orthogonal matrix Q is multiplied from two sides, is called the weighted orthogonal Procrustes problem (WOPP):

$$\|CQA - B\|_F \to \min_{Q^TQ=I}.$$

Unfortunately, WOPP does not have a simple analytical solution (Lissitz et al., 1976). We solve it by parametrizing Q using the so-called Cayley transform (Golub and Van Loan, 2013) and using conjugate-gradients (see Algorithm 3 and (6)).

5.2 Efficient initialization

As we mentioned above, the optimization problem (3) does not admit a simple solution. Therefore, we suggest finding a proper initialization for the arising matrices first. As a good initial point we can use optimal solution in the Frobenius norm:

$$\|W_{out}Q - \widehat{W}_{out}\|_F^2 + \|Q^TW_{in} - \widehat{W}_{in}\|_F^2 \to \min_{\substack{\widehat{W}_{out}, \widehat{W}_{in}, \\ Q^TQ=I}}$$

To solve this problem, we can rewrite it as:

$$\left\|[W_{out}, W_{in}^T]Q - [\widehat{W}_{out}, \widehat{W}_{in}^T]\right\|_F^2 \to \min_{\substack{\widehat{W}_{out}, \widehat{W}_{in}, \\ Q^TQ=I}},$$

which is solved using the alternating scheme called alternating least squares (ALS), see Algorithm 2. In particular, we first find optimal $\widehat{W}_{out}, \widehat{W}_{in}$ for the fixed Q, and then optimal orthogonal matrix Q for the fixed $\widehat{W}_{out}, \widehat{W}_{in}$ (orthogonal Procrustes problem). The process is repeated until the maximum number of iterations is reached.

5.3 Alternating iteration

After obtaining initializations for orthogonal layers from ALS in Frobenius norm, we proceed with more computationally challenging weighted optimization scheme, described in Algorithm 3. Even though we only do a handful of steps of the algorithm, this step is crucial and noticeably improves results, as is shown in Table 1.

Algorithm 2 ALS in the Frobenius norm

Input: W_{out}, W_{in}.
Set $Q = I$.
for 1 . . . n_iters **do**
  ▷ Projection step (Section 4):
  $\widehat{W}_{in} = \arg\min_{W \in \mathcal{S}_{\text{in}}} \|Q^TW_{in} - W\|_F^2$
  $\widehat{W}_{out} = \arg\min_{W \in \mathcal{S}_{\text{out}}} \|W_{out}Q - W\|_F^2$
  $W_{appr} = [\widehat{W}_{out}, \widehat{W}_{in}^T]$
  $W = [W_{out}, W_{in}^T]$
  ▷ Solve OPP, using SVD (Section 5.1)
  $Q = \arg\min_{Q^TQ=I} \|WQ - W_{appr}\|_F^2$
end for
return $Q, \widehat{W}_{in}, \widehat{W}_{out}$

Algorithm 3 ALS in the weighted norm

Before compression in weighted norm, rotate the network with Q_{init} from Algorithm 2.
Input $W_{out}, W_{in}, X_{out}, X_{in}$.
Set $Q = I$
for 1 . . . n_iters **do**
  ▷ Weighted norm projection (Appx. A, B):
  $\widehat{W}_{out} = \arg\min_{W \in \mathcal{S}_{\text{in}}} \|X_{out}(W_{out}Q - W)\|_F^2$
  $\widehat{W}_{in} = \arg\min_{W \in \mathcal{S}_{\text{out}}} \|X_{in}(W_{in} - QW)\|_F^2$
  ▷ Solve WOPP by parametrizing Q with Cayley transform (6) and using conjugate gradients:
  $Q = \arg\min_{Q^TQ=I} \|X_{out}(W_{out}Q - \widehat{W}_{out})\|_F^2 +$
  $+ \|X_{in}(W_{in} - Q\widehat{W}_{in})\|_F^2$
end for
return $Q, \widehat{W}_{in}, \widehat{W}_{out}$

5.4 Practical aspects

Computing inputs. The input of the linear layer X is a matrix of the shape $bs \times n$, where b, s and n are respectively the sequence length, the number of calibration samples and the hidden dimension. Typically, b and s are large, so making computations with X is challenging. However, we can use square root of the smaller $n \times n$ correlation matrix $X^TX \in \mathbb{R}^{n\times n}$ instead of X to solve the optimization problem. Indeed,

$$\|(X^TX)^{1/2}(\dots)\|_F = \|X(\dots)\|_F.$$

To efficiently compute the correlation matrix, we divide $X \in \mathbb{R}^{bs \times n}$ into smaller matrices (batches) $X_i \in \mathbb{R}^{b_i s \times n}$, which fit into memory, and compute $X^T X = \sum_i X_i^T X_i$.

Embedding and head layers. For the embedding layer the inputs X_{in} are one-hot vectors with ones standing in the position of the chosen tokens. Therefore, the correlation matrix $X_{in}^T X_{in}$ is equal to the diagonal matrix D, where D_{ii} is the number of times the i-th token appears in the calibration dataset. This simplifies the weighted problem to:

$$\|\sqrt{D}(W_{emb}Q - \widehat{W}_{emb})\|_F^2 + \| \dots \|_F^2 \to \min_Q,$$

where W_{emb} is the weight of embedding, $\widehat{W}_{emb}$ is its approximation with the matrix decomposition. This prompted us to also experiment with different functions of D and we found that $\sqrt{D+1}$ gave the best results, although $\log(D+1)$ also worked well (see Tables 1, 2). We have also discovered that weighting the model's head with the same diagonal matrix as embedding during compression in Frobenius norm additionally improves the performance.

Orthogonal parametrization. When rotating the network with the set of weights $\mathcal{Q} = \{Q_1, Q_2, \dots Q_L\}$, one downside is additional weights $Q_{\ell-1}^T Q_\ell$ that arise in skip connections and should also be stored as the weights. This negatively affects the compression ratio. One trick to reduce the number of parameters is an observation that $Q_{\ell-1}^T Q_\ell$ is an orthogonal matrix.

Orthogonal matrices (except for those that have an eigenvalue exactly equal to -1) of size $d \times d$ can be represented through the Cayley transform:

$$Q = (I + K)(I - K)^{-1}, \tag{6}$$

or through the matrix exponential (if $\det Q = 1$):

$$Q = \sum_{n=0}^{\infty} \frac{K^n}{n!} \tag{7}$$

where K is skew-symmetric: $K = -K^T$. This allows us to only store the upper triangular part of the matrix K, which reduces the number of parameters from d^2 to $\frac{d(d-1)}{2}$. If $\det Q = -1$, any row of Q can be multiplied by -1 to change $\det$ to 1. For the Cayley transform, if there exist eigenvectors u_i corresponding to -1 eigenvalues, we multiply Q by Householder matrices $I - 2vv^T/\|v\|_2^2$, where $v = Re(u_i)$ or $v = Im(u_i)$, to eliminate all -1 from the spectrum.

6 Experiments

6.1 Setup

We implement our method for OPT (Zhang et al., 2022) and Llama2 (Touvron et al., 2023) models using Hugging Face Transformers (Wolf, 2020). As the calibration data we use 128 sequences of length 2048 from WiKiText2 dataset (Merity et al., 2016). Experiments were run on a single V100 GPU with 32GB memory or A100 GPU with 80GB memory. We evaluate zero-shot performance using LM Evaluation Harness (Gao et al., 2024a) across ARC-e, ARC-c (Clark et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), and HellaSwag (Zellers et al., 2019) datasets.

6.2 Details of implementation

Every pair of the blocks W_{out}, W_{in} is rotated independently with its own orthogonal matrix, which allows us to parallelize the computation of the orthogonal matrices. We have implemented parallelization of ALS in Frobenius norm (Algorithm 2). The speedup depends on the number of processes that can be run simultaneously in GPU memory.

The compression process consists of two stages. Firstly, we compute optimal orthogonal matrices in the Frobenius norm (Algorithm 2), then we rotate the network using them and run compression in the weighted norm (Algorithm 3). We use 50 ALS iterations in Frobenius norm for small models and 25 iterations for 13b models. We use 1 iteration of ALS in the weighted norm for all models. We parametrize orthogonal matrix with the Cayley transform and apply 500 iterations of conjugate gradients to find optimal orthogonal matrices in the weighted norm. We do not compress the Values matrix at all, as it noticeably degrades the results.

6.3 Results

Generation performance. We assess the generation performance of the compressed models using the WikiText2 test set. Table 1 compares our method against SliceGPT at approximately 25% compression of the weight matrices. For the details on choice of matrix sizes in decompositions, see Appendix D. The "F" row demonstrates the perplexity achieved by finding optimal orthogonal matrices in the Frobenius norm (see Section 5.2). We observe that compression using the Frobenius norm alone is not sufficient to preserve model performance, particularly for small models. However, further compression in the weighted norm helps to

Method	Norm	Struct.	Coef.	OPT 125m ppl	OPT 125m %	OPT 2.7b ppl	OPT 2.7b %	OPT 13b ppl	OPT 13b %	Llama2 7b ppl	Llama2 7b %	Llama2 13b ppl	Llama2 13b %
Dense				27.65	0	12.47	0	10.13	0	5.47	0	4.88	0
SliceGPT				38.65	20.12	14.84	16.51	11.12	16.00	7.60	16.04	6.60	15.94
ProcrustesGPT	F	Kron.	$\log(D+1)$	58.17	19.59	82.47	15.57	15.20	15.00	16.37	15.04	8.43	14.93
ProcrustesGPT	W	Kron.	$\log(D+1)$	38.48	19.59	14.17	15.57	10.87	15.00	8.48	15.04	5.72	14.93
ProcrustesGPT	F	Kron.	$\sqrt{D+1}$	55.91	19.59	78.97	15.57	15.38	15.00	11.98	15.04	8.39	14.93
ProcrustesGPT	W	Kron.	$\sqrt{D+1}$	**36.08**	19.59	13.95	15.57	**10.67**	15.00	**6.54**	15.04	**5.71**	14.93
ProcrustesGPT	F	GS	$\sqrt{D+1}$	154.09	19.45	152.51	15.57	261.57	15.00	11.65	15.05	8.80	14.93
ProcrustesGPT	W	GS	$\sqrt{D+1}$	39.58	19.45	**13.81**	15.57	10.68	15.00	6.76	15.05	5.96	14.93

Table 1: Perplexity results on WikiText2. The calibration dataset size is 128 sequences of 2048 tokens. "Ppl" denotes perplexity, the columns with % show the percentage of parameters compressed. "F" stands for the optimization in Frobenius norm, which is used before optimization in weighted norm, denoted as "W"."Coef." stands for the diagonal matrix, which weighs embedding and head. In the rows with $\log(D+1)$, $\lambda_{in} = 1$; in the rows with $\sqrt{D+1}$, $\lambda_{in} = \frac{\|X_{out}W_{out}\|^2}{\|X_{in}W_{in}\|^2}$.

Model	Method	Struct.	Coef.	ARC-c	ARC-e	HellaS.	PIQA	WinoG.	Average
OPT-13b	Dense			35.75	61.83	69.88	76.82	65.19	61.89
	SliceGPT			33.62	61.95	62.99	73.67	63.30	59.10
	ProcrustesGPT	Kron.	$\log(D+1)$	36.52	58.71	68.27	76.22	63.61	60.67
	ProcrustesGPT	Kron.	$\sqrt{D+1}$	**36.60**	**62.25**	**68.41**	**76.44**	64.96	**61.73**
	ProcrustesGPT	GS	$\sqrt{D+1}$	35.07	59.09	67.14	76.06	65.59	60.59
Llama2-7b	Dense			46.25	74.58	75.99	79.11	68.82	68.95
	SliceGPT			35.15	56.10	53.04	65.78	62.98	54.61
	ProcrustesGPT	Kron.	$\log(D+1)$	41.98	**68.35**	69.72	73.94	**67.40**	64.28
	ProcrustesGPT	Kron.	$\sqrt{D+1}$	**42.32**	68.01	**70.20**	**76.39**	66.30	**64.64**
	ProcrustesGPT	GS	$\sqrt{D+1}$	36.69	62.88	66.91	74.37	**67.40**	61.65
Llama2-13b	Dense			49.23	77.53	79.36	80.52	72.30	71.79
	SliceGPT			39.51	62.92	56.98	67.25	67.64	58.86
	ProcrustesGPT	Kron.	$\log(D+1)$	**44.97**	73.19	73.43	**77.58**	70.48	67.93
	ProcrustesGPT	Kron.	$\sqrt{D+1}$	44.80	73.23	**73.94**	77.53	**70.88**	**68.07**
	ProcrustesGPT	GS	$\sqrt{D+1}$	44.45	**73.32**	71.19	76.50	70.32	67.16

Table 2: Zero-shot task performance of compressed OPT-13b, Llama2-7b and Llama2-13b. ProcrustesGPT compresses the weights with Kronecker product. The compression ratio is the same as in Table 1.

regain perplexity, which is shown in the "W" row. We observe that $\mathcal{GS}$ performs consistently worse than Kronecker products in the Frobenius norm for OPT models, and produces approximately the same results in the weighted norm. Our method surpasses SliceGPT at a similar level of compression. The compression ratios in the tables are presented with respect to the full model size.

Table 3 compares ProcrustesGPT to the methods that do not compress embedding and model's head. SVD-LLM (Wang et al., 2024) applies weighted low-rank approximation, while DISP-LLM (Gao et al., 2024b) and SLEB (Song et al., 2024) are pruning methods. ProcrustesGPT outperforms other baselines at lower compression rates, but its performance starts deteriorating at 36% compression of parameters.

Zero-shot tasks. We evaluate our models compressed in the weighted norm on five zero-shot tasks. Our method consistently outperforms SliceGPT, as shown in Table 2. The difference is more pronounced for the Llama2 models, where our method surpasses SliceGPT by an average of 9-10%. We notice that compression with sum of Kronecker products better maintains model quality than $\mathcal{GS}$. Table 4 shows that on average ProcrustesGPT achieves better zero-shot performance than other baselines.

Method	Llama2-7b						Llama2-13b					
	ppl	%	ppl	%	ppl	%	ppl	%	ppl	%	ppl	%
Dense	5.47	0	5.47	0	5.47	0	4.88	0	4.88	0	4.88	0
SVD-LLM	7.86	14.44	9.73	25.00	14.39	35.58	6.34	14.64	7.53	25.36	10.08	36.09
DISP-LLM	6.80	14.31	8.52	25.02	**10.92**	35.60	6.23	14.60	7.90	25.36	**10.05**	36.13
SLEB	6.95	12.01	10.39	24.03	22.76	36.04	5.85	12.19	7.73	24.37	11.36	36.56
ProcrustesGPT (Kron)	**6.43**	14.07	8.19	25.09	19.55	36.11	**5.68**	14.30	**6.95**	25.48	16.88	36.66
ProcrustesGPT (GS)	6.65	14.08	**7.97**	25.08	14.20	36.12	5.94	14.30	7.02	25.48	10.85	36.66

Table 3: Perplexity of compressed Llama2 on WikiText2. Embedding and head are not compressed for all methods. The calibration dataset size is 128 sequences of 2048 tokens. % shows the percentage of parameters compressed.

Method	%	ARC-c	ARC-e	HellaS.	PIQA	WinoG.	Average
Dense	0	49.23	77.53	79.36	80.52	72.30	71.79
SVD-LLM	14.64	39.25	65.61	63.92	73.83	68.35	62.19
DISP-LLM	14.60	**47.61**	70.12	**74.77**	76.93	69.61	67.81
SLEB	12.19	46.33	72.77	74.11	**78.18**	69.85	68.25
ProcrustesGPT (Kron)	14.30	45.56	**74.16**	74.29	77.58	70.24	**68.37**
ProcrustesGPT (GS)	14.30	45.05	73.40	71.71	76.77	**70.56**	67.50
SVD-LLM	25.36	32.76	54.34	54.19	68.12	65.98	55.08
DISP-LLM	25.36	40.27	61.83	**69.40**	73.56	63.30	61.67
SLEB	24.37	38.14	63.47	66.78	**76.39**	60.70	61.10
ProcrustesGPT (Kron)	25.48	38.48	70.03	66.42	75.03	66.06	63.20
ProcrustesGPT (GS)	25.48	**41.04**	**71.38**	66.12	74.71	**67.17**	**64.10**
SVD-LLM	36.09	26.96	43.22	43.35	61.53	60.54	47.12
DISP-LLM	36.13	30.72	53.03	**60.65**	68.72	58.64	54.35
SLEB	36.56	33.62	52.15	58.42	**71.27**	59.67	55.03
ProcrustesGPT (Kron)	36.66	29.69	52.95	48.56	64.80	59.98	51.20
ProcrustesGPT (GS)	36.66	**33.70**	**61.87**	52.58	67.85	**62.43**	**55.69**

Table 4: Zero-shot performance of compressed Llama2-13b. Embedding and head are not compressed for all methods. % shows the percentage of model parameters compressed.

7 Conclusion

This paper presents an approach to LLM compression with structured matrix factorizations, which is suitable for compression with various types of decompositions, including Kronecker products and $\mathcal{GS}$ matrices. Our method maintains performance in generation and zero-shot tasks, and does not require recovery fine-tuning. We hope this work will inspire further research on training-free compression with structured representations.

8 Limitations

A natural question may arise if the models compressed using ProcrustesGPT can be fine-tuned with standard methods such as LoRA. The structured weights may not align well with the low-rank nature of adapters. As a result, after the fine-tuning we will need to store the structured representation of initial weights and the LoRA adapters separately, which may be inconvenient. Therefore, models compressed using structured factorizations require the development of PEFT methods that are better suited to these structures.

9 Acknowledgments

Support from the Basic Research Program of HSE University is gratefully acknowledged. The calculations were performed in part through the computational resources of HPC facilities at HSE University (Kostenetskiy et al., 2021).

References

V Abronin, A Naumov, D Mazur, D Bystrov, K Tsarova, Ar Melnikov, I Oseledets, Sergey Dolgov, R Brasher, and Michael Perelshtein. 2024. Tqcompressor: improving tensor decomposition methods in neural networks via permutations. *arXiv preprint arXiv:2401.16367*.

Anish Acharya, Rahul Goel, Angeliki Metallinou, and Inderjit Dhillon. 2019. Online embedding compression for text classification using low rank matrix factorization. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 6196–6203.

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Patrick Chen, Si Si, Yang Li, Ciprian Chelba, and Cho-Jui Hsieh. 2018. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. *Advances in Neural Information Processing Systems*, 31.

Patrick Chen, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. 2021. Drone: Data-aware low-rank compression for large nlp models. *Advances in neural information processing systems*, 34:29321–29334.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. 2022. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pages 4690–4721. PMLR.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024a. A framework for few-shot language model evaluation.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Zheng Tang, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2024b. Disp-llm: Dimension-independent structural pruning for large language models. *Advances in Neural Information Processing Systems*, 37:72219–72244.

Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. JHU press.

Mikhail Gorbunov, Nikolay Yudin, Vera Soboleva, Aibek Alanov, Alexey Naumov, and Maxim Rakhuba. 2024. Group and shuffle: Efficient structured orthogonal parametrization. In *Advances in Neural Information Processing Systems*, volume 37, pages 68713–68739. Curran Associates, Inc.

Oleksii Hrinchuk, Valentin Khrulkov, Leyla Mirvakhabova, Elena Orlova, and Ivan Oseledets. 2020. Tensorized embedding layers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4847–4860.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Language model compression with weighted low-rank factorization. *arXiv preprint arXiv:2207.00112*.

Yixin Ji, Yang Xiang, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, Kehai Chen, and Min Zhang. 2024. Adaptive feature-based low-rank compression of large language models via bayesian optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4152–4168.

PS Kostenetskiy, RA Chulkevich, and VI Kozyrev. 2021. HPC resources of the higher school of economics. In *Journal of Physics: Conference Series*, volume 1740, page 012050.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. Losparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, pages 20336–20350. PMLR.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2024. Modegpt: Modular decomposition for large language model compression. *arXiv preprint arXiv:2408.09632*.

Vasileios Lioutas, Ahmad Rashid, Krtin Kumar, Md Akmal Haidar, and Mehdi Rezagholizadeh. 2020. Improving word embedding factorization for compression using distilled nonlinear neural decomposition. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2774–2784.

Robert W Lissitz, Peter H Schönemann, and James C Lingoes. 1976. A solution to the weighted procrustes problem in which the transformation is in agreement with the loss function. *Psychometrika*, 41:547–550.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea J Goldsmith, and Mert Pilanci. 2024. Compressing large language models using low rank and low precision decomposition. *arXiv preprint arXiv:2405.18886*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Pratyusha Sharma, Jordan T Ash, and Dipendra Misra. 2023. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. *arXiv preprint arXiv:2312.13558*.

Jiwon Song, Kyungseok Oh, Taesu Kim, Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Sleb: Streamlining llms through redundancy verification and elimination of transformer blocks. *arXiv preprint arXiv:2402.09025*.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2024. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*.

Thomas Wolf. 2020. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Mingxue Xu, Yao Lei Xu, and Danilo P Mandic. 2023. Tensorgpt: Efficient compression of the embedding layer in llms based on the tensor-train decomposition. *arXiv preprint arXiv:2307.00526*.

Hao Yu and Jianxin Wu. 2023. Compressing transformers: features are low-rank, but weights are not! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11007–11015.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. 2023. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Kronecker approximation in weighted norm

Let $A \in \mathbb{R}^{r \times m_1 \times n_1}, B \in \mathbb{R}^{r \times m_2 \times n_2}$. The problem of Kronecker approximation in the weighted norm can be written as

$$g(A, B) = \left\| Y - X \sum_{i}^{r} (A_i \otimes B_i) \right\|_F^2 \to \min_{A,B}.$$

We can not solve this problem explicitly, so we will iteratively optimize this expression with respect to A for the fixed B and with respect to B for the fixed A. We can rewrite it as

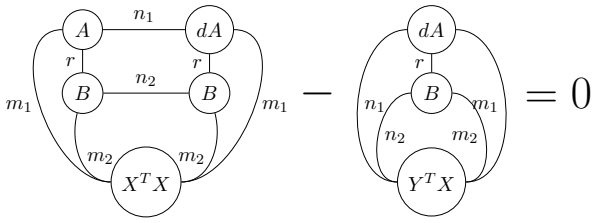
$$g(A, B) = tr \left(Y^T Y - 2Y^T X \left(\sum_{i}^{r} A_i \otimes B_i \right) + \right.$$
$$\left. + \left(\sum_{i}^{r} A_i \otimes B_i \right)^T X^T X \left(\sum_{j}^{r} A_j \otimes B_j \right) \right).$$

Now we take the differential with respect to A

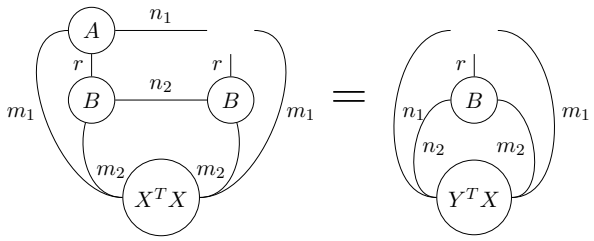
$$dg = tr \left(2Y^T X \left(\sum_{i}^{r} dA_i \otimes B_i \right) + \right.$$
$$\left. +2 \left(\sum_{i}^{r} A_i \otimes B_i \right)^T X^T X \left(\sum_{j}^{r} dA_j \otimes B_j \right) \right).$$

To find the optimal A we should equate the gradient with respect to A to zero. It is complicated to write the solution using formulas, so we will use tensor diagrams instead. In tensor diagrams, the circles denote multidimensional arrays, the connections between them denote the summation by dimensions. Let us equate the dg to zero and illustrate it as follows:

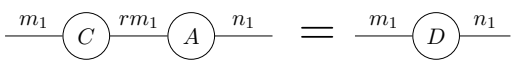
$$tr \left(\left(\sum_{i}^{r} A_i \otimes B_i \right)^T X^T X \left(\sum_{j}^{r} dA_j \otimes B_j \right) \right) -$$
$$- tr \left(Y^T X \left(\sum_{i}^{r} dA_i \otimes B_i \right) \right) = 0$$



Now we equate the gradient with respect to A to zero:



We can reshape the tensors in the left and right parts into matrices. Let us denote the matrix in the right part by D. The left part is the tensor A reshaped into matrix of size $rm_1 \times n_1$ multiplied by a matrix C:



Then the solution is

$$A = (C^+D).reshape(r, m_1, n_1).$$

The pseudo-code is shown in Algorithm 4.

Algorithm 4 Kronecker appr. in weighted norm

Input $B \in \mathbb{R}^{r\times m_2\times n_2}, Y \in \mathbb{R}^{m_1m_2\times n_1n_2}, X \in \mathbb{R}^{m_1m_2\times m_1m_2}$.
Solve $\|Y - X\sum_i^r (A_i \otimes B_i)\|_F^2 \to \min_A$
$Z = X^TX$.reshape(m_1, m_2, m_1, m_2)
$C =$ einsum('rij,nkj,aibk→ranb', B, B, Z)
$C = C$.reshape(rm_1, rm_1)
$T = X^TY$.reshape(m_1, m_2, n_1, n_2)
$D =$ einsum('rij,aibj→rab', B, T)
$D = D$.reshape(rm_1, n_1)
$A = (C^+D)$.reshape(r, m_1, n_1)
return A

B $\mathcal{GS}$pproximation in weighted norm

Let L be block-diagonal matrix with kl blocks of size $L_i \in \mathbb{R}^{bl_1\times bl_2}$, R be block-diagonal matrix with kr blocks of size $R_i \in \mathbb{R}^{br_1\times br_2}$. Let us solve the approximation with $\mathcal{GS}$ matrices with fixed permutations P_L, P, P_R in the weighted norm:

$$\|Y' - X'P_LLPRP_R\|_F^2 \to \min_{L,R}.$$

Due to the unitary invariance of Frobenius norm and orthogonality of permutation matrices, we can rewrite it as

$$\|Y - XLPR\|_F^2 \to \min_{L,R},$$

where $X = X'P_L, Y = Y'P_R^T$. We can not solve this problem analytically, so we will optimize iteratively with respect to L and R.

To find optimal R for the fixed L we have to solve the least squares problem for each of the blocks R_i. Let us divide Y and XLP by columns into kr blocks $Y_i \in \mathbb{R}^{kl\cdot bl1\times bl2}$ and $(XLP)_i \in \mathbb{R}^{kl\cdot bl1\times bl2}$. Then we can find the optimal R as

$$R_i = (XLP)_i^+Y_i.$$

To solve the optimization problem with respect to L for the fixed R, we use built-in iterative algorithm for least squares problem. However, to avoid poorly conditioned systems, we first divide X and PR into kl blocks by columns and rows respectively, and make QR decomposition of X_i and $(PR)_i$. The result is shown in Algorithm 5 below.

Algorithm 5 Find optimal L

Input $X_i \in \mathbb{R}^{kl\cdot bl_1\times bl_1}, Y \in \mathbb{R}^{kl\cdot bl_1\times kr\cdot br_2}, (PR)_i \in \mathbb{R}^{kl\times kr\cdot br_2}$.
Solve $\|Y - \sum_i^{kl} X_iL_i(PR)_i\|_F^2 \to \min_L$
$Q_{X_i}, R_{X_i} = qr(X_i)$
$Q_{(PR)_i^T}, R_{(PR)_i^T} = qr((PR)_i^T)$
$L = \arg\min_L \left(Y - \sum_i^{kl} Q_{X_i}L_iQ_{(PR)_i^T}^T\right)$
$L_i = R_{X_i}^+L_i(R_{(PR)_i^T}^T)^+$
return L

C Proof of Proposition 4.3

Firstly, let us write

$$\begin{aligned}&\|X_{out}(W_{out}Q - \widehat{W}_{out})+\\&+ X_{skip}(Q - Q_{\ell-1}\widehat{W}_{skip})\|_F^2 =\\&= \|(X_{out}W_{out} + X_{skip})-\\&- (X_{out}\widehat{W}_{out} + X_{skip}Q_{\ell-1}\widehat{W}_{skip})Q^T\|_F^2\end{aligned}.$$

Note that slicing of the columns of matrices $\widehat{W}_{skip}$ and $\widehat{W}_{out}$ make matrix $X_{out}\widehat{W}_{out}Q^T + X_{skip}Q_{\ell-1}\widehat{W}_{skip}Q^T$ at most of rank d, where d is the dimension after slicing. If we now show, that we can choose appropriate matrices $Q, \widehat{W}_{out}, \widehat{W}_{skip}$ such that they correspond to best

low-rank approximation of $X_{out}W_{out} + X_{skip}$, we will show that these matrices are the analytical solutions to our problem. Let P equal to the following projector:

$$P = \begin{pmatrix} I_d & 0 \\ 0 & 0 \end{pmatrix}$$

Consider following matrices for $\widehat{W}_{out}, \widehat{W}_{skip}$:

$$\widehat{W}_{out} = W_{out}QP; \quad \widehat{W}_{skip} = Q_{\ell-1}^T QP$$

Substituting, we get:

$$\|(X_{out}W_{out} + X_{skip})Q(I - P)\|_F^2 \to \min_{Q^TQ=I}$$

Solution to which is well-known and revolves around computing SVD of $(X_{out}W_{out} + X_{skip})$ (taking principal components). This also corresponds to the best low-rank approximation and therefore found solution is optimal.

Up to a normalization layer, this corresponds to the rotations and slices of W_{out} and $Q_{\ell-1}^T Q_\ell$ layers applied in SliceGPT. Sparse block structure that arises in inputs after the first layers allows to also slice W_{in} and $Q_{\ell-1}^T Q_\ell$ along other dimension, making it equivalent to full SliceGPT scheme.

D Choice of sizes for factorizations.

To compress a matrix M of size $n \times m$ at approximately p/q of its parameters with the sum of r Kronecker products $\sum_i^r A_i \otimes B_i$, a natural choice is to set $r = p$, A_i of size $q \times 1$ and B_i of size $n/q \times m$. We assume n is divisible by q. Then the number of parameters in the sum $\sum_i^{r=p} A_i \otimes B_i$ equals $p(q + (n/q)m) = qp + nm(p/q)$, which is approximately p/q of initial matrix size. In our case, we use $q = 4$ and $r = 3$, yielding $\sim 25\%$ compression rate for layers. We use $q = 5, r = 8$ for $\sim 37.5\%$ compression, $q = 1, r = 2$ for $\sim 50\%$ compression.

It is important to note, that splitting by q should be applied on the side that is multiplied by an orthogonal matrix, so that it has an effect on approximation error. For example, if the matrix W_{in} is multiplied by orthogonal matrix from the left, then decomposition should be $A_i \in \mathbb{R}^{q\times 1}, B_i \in \mathbb{R}^{n/q\times m}$ or $A_i \in \mathbb{R}^{n/q\times m}, B_i \in \mathbb{R}^{q\times 1}$. If W_{out} is multiplied from the right, then $A_i \in \mathbb{R}^{1\times q}, B_i \in \mathbb{R}^{n\times m/q}$ or $A_i \in \mathbb{R}^{n\times m/q}, B_i \in \mathbb{R}^{1\times q}$.

When compressing the matrix $M \in \mathbb{R}^{kl*bl_1\times kr*br_2}$ with GS decomposition using L with kl blocks of the size $bl_1 \times bl_2$ and R with kr blocks of size $br_1 \times br_2$, the compression ratio equals $c = \frac{kl*bl_1*bl_2+kr*br_1*br_2}{kl*kr*bl_1*br_2}$. From the definition of GS matrices, $kl * bl_2 = kr * br_1$, which means that it is easy to find bl_2 and br_1 if c, kl, bl_1, kr, br_2 are known. To compress square matrices at $c = 3/4$, we choose $kl, kr = 4, 2$. For rectangular matrices, $kl, kr = 4, 8$. For embedding and head $kl, kr = 1, 4$.