TensorLLM: Tensorising Multi-Head Attention for Enhanced Reasoning and Compression in LLMs

Yuxuan Gu, Wuyang Zhou, Giorgos Iacovides, Danilo Mandic
Department of Electrical and Electronic Engineering
Imperial College London, United Kingdom
{yuxuan.gu21, wuyang.zhou19, giorgos.iacovides20, d.mandic}@imperial.ac.uk

***Abstract*—The reasoning abilities of Large Language Models (LLMs) can be improved by structurally denoising their weights, yet existing techniques primarily focus on denoising the feed-forward network (FFN) of the transformer block, and can not efficiently utilise the Multi-head Attention (MHA) block, which is the core of transformer architectures. To address this issue, we propose a novel intuitive framework that, at its very core, performs MHA compression through a multi-head tensorisation process and the Tucker decomposition. This enables both higher-dimensional structured denoising and compression of the MHA weights, by enforcing a shared higher-dimensional subspace across the weights of the multiple attention heads. We demonstrate that this approach consistently enhances the reasoning capabilities of LLMs across multiple benchmark datasets, and for both encoder-only and decoder-only architectures, while achieving compression rates of up to ~ 250 times in the MHA weights, all without requiring any additional data, training, or fine-tuning. Furthermore, we show that the proposed method can be seamlessly combined with existing FFN-only-based denoising techniques to achieve further improvements in LLM reasoning performance.** [1]

***Index Terms*—Large Language Models, Multi-head Attention, Tensorisation, Tucker Decomposition, Reasoning, Compression**

I. Introduction

Large Language Models (LLM) based on the transformer architecture, such as those belonging to the GPT-series [1], [2] and LLaMA-series [3]–[5], have demonstrated enormous success across diverse applications in natural language processing (NLP) [6]–[8]. This is attributed to the exceedingly large size of these models and the vast amount of data for their training. Indeed, transformer models with more parameters or larger training datasets tend to comprehensively outperform their smaller-scale predecessors [1], [3], owing to their ability to capture more complex patterns.

The success of LLMs has demonstrated that massive over-parameterization is beneficial during training [9], however, a growing body of research suggests that transformer-based models, and neural networks (NN) in general, do not require all learnt parameters to maintain their performance [10], [11]. This has led to the exploration of post-training compression techniques, which aim to make LLMs more efficient and practical for real-world applications, particularly in resource-constrained environments. This should be achieved without largely compromising their inference performance and generative capabilities [12]–[15].

[1]The code is available at https://github.com/guyuxuan9/TensorLLM.

Given that LLMs are grossly overparameterised, it comes as no surprise that the reasoning abilities of LLMs can be maintained, or even improved, when structurally compressing their parameters. For instance, in the LAyer-SElective Rank reduction (LASER) model [12], the authors demonstrated that by applying singular value decomposition (SVD) to the individual matrices, and subsequently removing the factors corresponding to the smallest singular values, one can improve the reasoning performance of LLMs. This is particularly the case when compressing the feed-forward network (FFN) weight matrices. The authors argued that by structurally compressing the individual weight matrices, they were essentially removing the weight noise caused by the randomness introduced during training. However, LASER applies SVD to only individual weight matrices, and is unable to exploit the information shared between the weight matrices.

To overcome this limitation, authors in [16] introduced the Tensor Reduced and Approximated Weights (TRAWL) model, which adopts a tensor-based approach by naively stacking weight matrices from either the multi-head attention (MHA) block or the FFN block into a higher-dimensional tensor, before applying tensor decompositions. While TRAWL can leverage the inherent tensor structure within the weights to exploit the inter-matrix correlations and can even outperform LASER in some experiments, it was found to be only effective when denoising the FFN blocks and not the MHA blocks.

The success of the existing approaches for denoising FFN blocks [12], [16] has also highlighted the open problem of exploiting similar denoising benefits in the MHA blocks. Indeed, existing approaches [12], [16] usually achieve their best performance gains when denoising the weight matrices in the FFN blocks. However, when applied to the denoising of weights in MHA blocks, similar performance increases were not observed. This discrepancy is significant and somewhat surprising, as the MHA mechanism is widely regarded as the very core of the transformer architecture and LLMs in general. To this end, in this work, we show that the reason why prior works were not effective when applied to MHA is that they do not leverage domain knowledge — the multiple attention heads in each layer should operate in a coherent way rather than independently.

Both design intuitions and current literature [17]–[19] on MHA suggest that:

1) Attention heads within the same layer capture the same

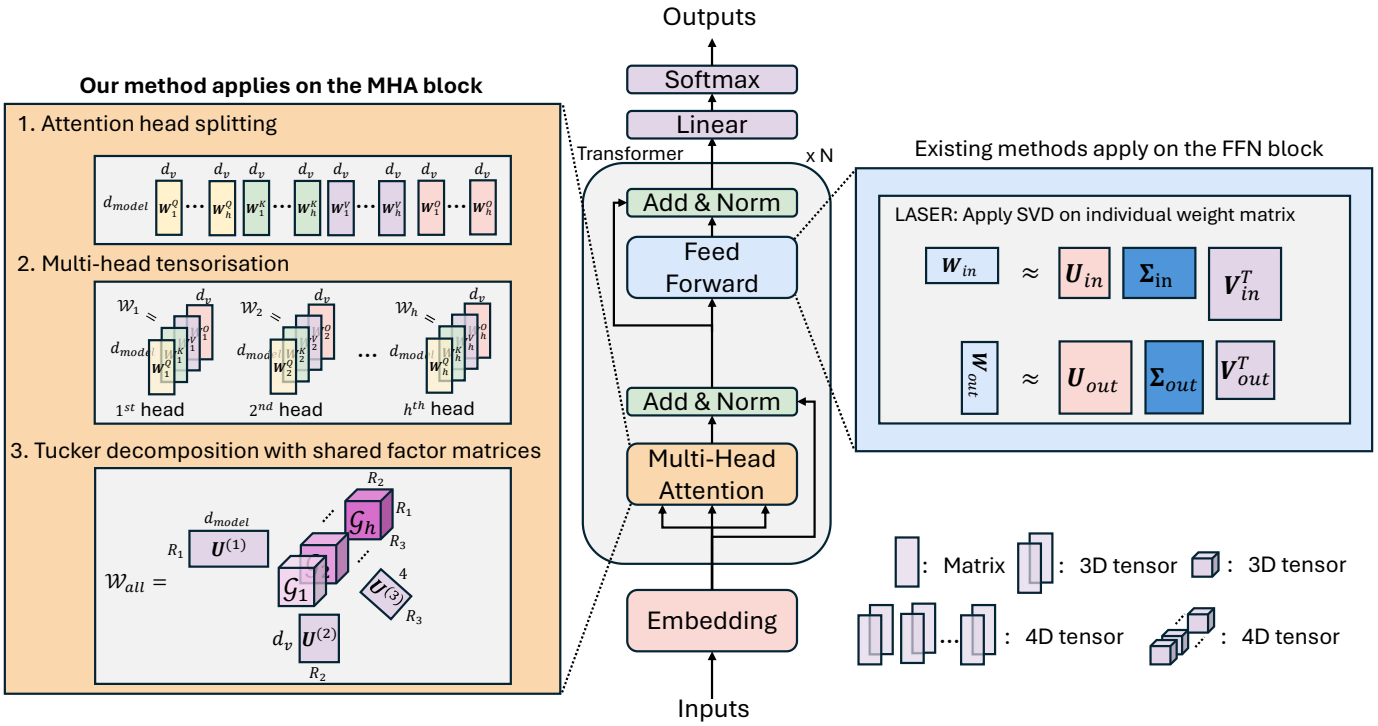

Fig. 1. Structures of our proposed framework for the compression of the MHA block and the existing methods (such as LASER) which apply to the FFN block. **Left**: The three-step denoising process applied to the MHA block (**our method**): (1) split the weight matrices into multiple heads, (2) tensorise the matrices of each attention head into a 3D tensor, and (3) apply Tucker decomposition to this set of 3D tensors with a common set of factor matrices to perform denoising. **Middle**: A standard (vanilla) decoder-only or encoder-only transformer architecture. **Right**: Existing methods applied to the FFN block; an illustration of LASER.

level of patterns;

2) Different attention heads within the same layer learn different specialized knowledge.

More specifically, the authors in [17] provided empirical evidence to support these intuitions by analyzing the Jensen-Shannon divergence between the outputs of the corresponding attention heads. Their analysis revealed distinct clusters of attention heads within transformer layers, indicating that heads in the same layer tend to focus on similar type of information, albeit with some degree of variation. Furthermore, the authors in [18], [19] visualized attention on multiple scales and demonstrated the specialised functions of different heads, such as those capturing positional and lexical patterns, detecting named entities, and identifying syntactic and semantic relations.

By leveraging on those intuitions and domain knowledge about MHA, we conjecture that the weights of MHA in a single transformer layer contain reasoning-related information in a subspace which is shared across multiple attention heads. We explore and elaborate this conjecture to help mitigate the limitations of existing works by answering the following question:

- Can we improve the reasoning capabilities of LLMs by enforcing a shared higher-dimensional subspace among the weights of multiple attention heads within a single transformer layer?

To answer this question, we propose a novel and intuitive framework based on the multi-head tensorisation and a special variant of the Tucker decomposition, which denoises the original MHA weights according to a shared higher-dimensional low-rank structure across multiple attention heads. By enforcing the weights of each attention head to be in a common higher-dimensional subspace characterised by a common set of Tucker factor matrices, this makes each attention head contain different information in the same subspace. In contrast to existing approaches that focus on the FFN weights only, we show that this improves the reasoning capabilities of LLMs by structurally denoising and compressing the MHA weights in a higher-dimensional format.

The main contributions of this work are threefold:

- A novel post-training weight-denoising framework is proposed based on prior knowledge, intuition, and empirical evidence about MHA, in order to improve the reasoning abilities of LLMs while simultaneously performing parameter compression. This is achieved through a unique multi-head tensorisation technique and a special variant of the Tucker decomposition applied on the MHA weight matrices.
- We demonstrate that the proposed framework can be used in conjunction with existing methods that denoise the FFN layers, to achieve even greater gains in the reasoning abilities of LLMs.
- Extensive experiments are conducted on four benchmark datasets across three well-established LLMs, ranging

from multiple millions to billions of parameters and from encoder-only to decoder-only architectures. Our method is found to improve the reasoning abilities of both the uncompressed LLMs and existing methods that focus only on denoising the FFN layers, all without requiring any additional data, training or fine-tuning.

II. Background

A. Mathematical Notations

The mathematical notations used in this paper are summarized in Table I. We adopt the same notation conventions as those in [21].

TABLE I
MATHEMATICAL NOTATIONS

Symbol	Meaning
a, $\mathbf{a}$, $\mathbf{A}$, $\mathcal{A}$	Scalar, vector, matrix, tensor
$(\cdot)^T$	Matrix Transpose
$\|\cdot\|_F$	Frobenius Norm
$\mathcal{A}_{[i_1,i_2,\ldots,i_N]}$	The $(i_1, i_2, \ldots, i_N)$-th element in an N-D tensor
$\mathbf{a} \circ \mathbf{b}$	Outer product between two vectors
$\mathbf{a} \cdot \mathbf{b}$	Inner product between two vectors
$\mathcal{A} \times_n \mathbf{B}$	Mode-n product between a tensor and a matrix
$\text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_R)$	A diagonal matrix
$\text{diag}_N(\lambda_1, \lambda_2, \ldots, \lambda_R)$	A diagonal tensor of N-dimensions

B. Tensor Preliminaries

a) Definition: A tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a multi-dimensional array. The number of modes (dimensions) in this tensor is its order, N. Its n-th mode has a size of I_n, where $n \in [1, N]$, along the n^{th} dimension [22].

For example, a scalar a is a 0-dimensional tensor. By stacking multiple scalars together, we form a 1-dimensional tensor $\mathbf{a}$, commonly referred to as a vector. Extending this further, stacking multiple vectors together results in a 2-dimensional tensor $\mathbf{A}$, which is a matrix. Finally, stacking multiple matrices together produces a 3-dimensional tensor $\mathcal{A}$. This process can be generalized to construct higher-dimensional tensors by iteratively stacking lower-dimensional ones.

b) Mode-n product: Mode-n product of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\mathbf{B} \in \mathbb{R}^{J_n \times I_n}$ yields a tensor $\mathcal{C} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times I_N}$. This is defined mathematically as

$$\mathcal{C} = \mathcal{A} \times_n \mathbf{B}, \tag{1}$$

with the element-wise definition of $\mathcal{C}$ as

$$\mathcal{C}_{[i_1,\ldots,i_{n-1},j_n,i_{n+1},i_N]} = \sum_{i_n=1}^{I_n} \mathcal{A}_{[i_1,\ldots,i_{n-1},i_n,i_{n+1},i_N]} \mathbf{B}_{[j_n,i_n]}. \tag{2}$$

C. Singular Value Decomposition (SVD)

In linear algebra, SVD factorizes any matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ into a sum of rank-1 matrices, and can be expressed as

$$\begin{aligned} \mathbf{A} &= \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sum_{i=1}^{r} \sigma_i \left(\mathbf{u}_i \circ \mathbf{v}_i\right) \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \end{aligned} \tag{3}$$

where $\mathbf{u}_i \in \mathbb{R}^m$ is an eigenvector of $\mathbf{A}\mathbf{A}^T$, and $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m] \in \mathbb{R}^{m \times m}$ is a matrix consisting of a set of orthonormal eigenvectors in $\mathbb{R}^m$. Similarly, $\mathbf{v}_i \in \mathbb{R}^n$ is an eigenvector of $\mathbf{A}^T\mathbf{A}$, and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n] \in \mathbb{R}^{n \times n}$ is a matrix consisting of a set of orthonormal eigenvectors in $\mathbb{R}^n$. Moreover, $\mathbf{u}_i \circ \mathbf{v}_i$ denotes the outer product between two vectors, σ_i are the singular values of $\mathbf{A}$, which are the square root of the eigenvalues of $\mathbf{A}^T\mathbf{A}$, and $\mathbf{\Sigma} = \text{diag}\left(\sigma_1, \sigma_2, \ldots, \sigma_r\right) \in \mathbb{R}^{m \times n}$ is a diagonal matrix, where r is the rank of matrix $\mathbf{A}$. The SVD performs low rank approximation by discarding the rank-1 factors associated with the smallest singular values, which likely contain the noisy component of the data and less useful information, thus performing denoising.

D. Tucker decomposition

The Tucker decomposition [23] is a generalisation of SVD [24], [25], which decomposes the original tensor into a product of factor matrices and a smaller core tensor, as shown in Eq. (4). Thus, Tucker decomposition enables the denoising of the original tensor by approximating the original tensor using a higher-dimensional low-rank structure, and is defined as

$$\begin{aligned} \mathcal{T} = &\sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} \mathcal{G}_{[r_1,r_2,\ldots,r_N]} \\ &\left(\mathbf{u}_{r_1}^{(1)} \circ \mathbf{u}_{r_2}^{(2)} \circ \cdots \circ \mathbf{u}_{r_N}^{(N)}\right) \\ &= \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)}. \end{aligned} \tag{4}$$

In this formulation, $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is an N-dimensional tensor. The core tensor, $\mathcal{G}_{[r_1,r_2,\ldots,r_N]} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$, represents a scaling coefficient, and $\mathbf{u}_{r_n}^{(n)} \in \mathbb{R}^{I_n}$ denotes a factor vector for the n-th dimension, while the factor matrix for the n-th dimension is defined as $\mathbf{U}^{(n)} = \left[\mathbf{u}_1^{(n)}, \mathbf{u}_2^{(n)}, \ldots \mathbf{u}_{R_n}^{(n)}\right] \in \mathbb{R}^{I_n \times R_n}$ for $1 \leq n \leq N$. The factor matrices in the Tucker decomposition characterise the projection of the original tensor onto a subspace. The N-tuple $(R_1, R_2, \ldots, R_N)$, commonly referred to as the multilinear ranks, is a generalization of matrix rank to higher-dimensions and is usually treated as hyperparameters. Identifying the optimal set of ranks efficiently is an active area of research, with numerous recent studies focusing on advanced methods for tensor rank search [26], [27]. Tucker decomposition enables compression in terms of the parameter count by setting $R_n \ll I_n$ for $1 \leq n \leq N$.

E. Transformer Architecture

Current transformer architectures [28] used in LLMs can be categorized into the encoder-only and decoder-only categories and often comprise N identical transformer layers, as illustrated in the middle section of Fig. 1. Each layer consists of two primary components: an MHA block and an FFN block. To enhance stability and facilitate learning, layer normalization [29] and residual connections are also applied between these blocks.

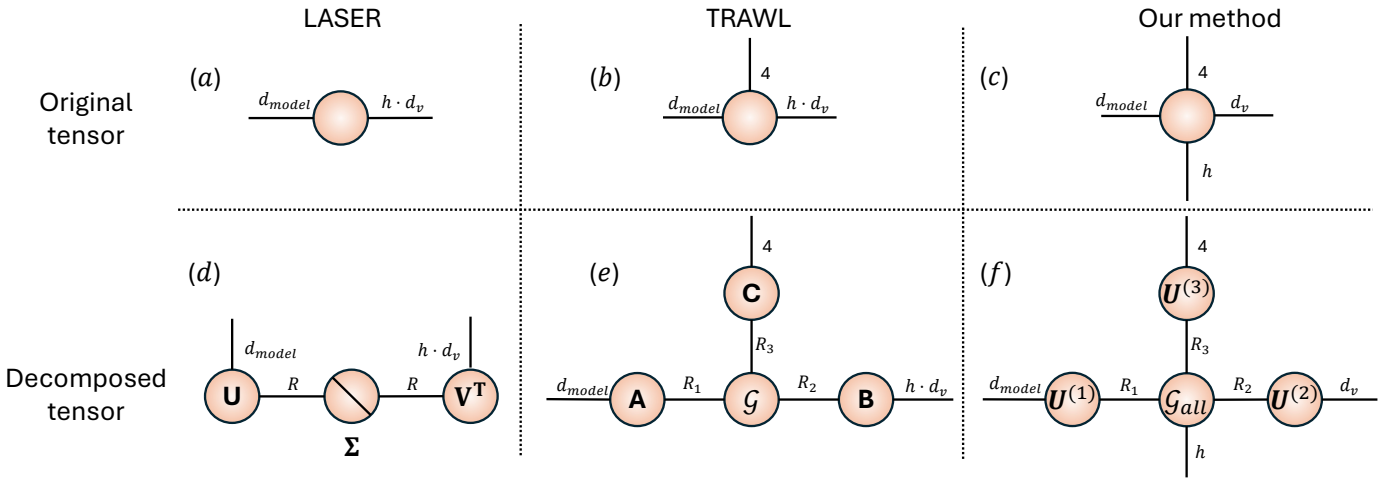

Fig. 2. Tensor network [20] topologies of different decomposition methods applied to the MHA weights in a single transformer layer. Note that while both LASER and TRAWL reported performances when only applied to the FFN blocks, they could also be applied to the MHA weights. The symbol d_{model} represents the embedding dimension, h is the number of attention heads, while $d_v = \frac{d_{model}}{h}$ stands for the head dimension. $(\mathbf{a}, \mathbf{d})$ LASER [12]: decomposition of a single weight matrix into $\mathbf{U} \in \mathbb{R}^{d_{model} \times R}$, $\mathbf{V} \in \mathbb{R}^{h \cdot d_v \times R}$, and a diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{R \times R}$. $(\mathbf{b}, \mathbf{e})$ TRAWL [16]: decomposition of a 3D tensor into the factor matrices $\mathbf{A} \in \mathbb{R}^{4 \times R_3}$, $\mathbf{B} \in \mathbb{R}^{h \cdot d_v \times R_2}$, and $\mathbf{C} \in \mathbb{R}^{d_{model} \times R_1}$, along with a core $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$. $(\mathbf{c}, \mathbf{f})$ **Our method** – Tucker decomposition with shared factor matrices: decomposing a set of 3D tensors using Tucker decomposition while having a common set of shared factor matrices $\mathbf{U}^{(1)} \in \mathbb{R}^{d_{model} \times R_1}$, $\mathbf{U}^{(2)} \in \mathbb{R}^{d_v \times R_2}$, $\mathbf{U}^{(3)} \in \mathbb{R}^{4 \times R_3}$, with $\mathcal{G}_{all} \in \mathbb{R}^{R_1 \times R_2 \times R_3 \times h}$ as the core tensor.

a) Multi-Head Attention (MHA): Single-head learnable self-attention is computed with the input query ($\mathbf{Q} \in \mathbb{R}^{L \times d_{model}}$), key ($\mathbf{K} \in \mathbb{R}^{L \times d_{model}}$), and value ($\mathbf{V} \in \mathbb{R}^{L \times d_{model}}$) matrices, defined as [28]:

$$\text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) = \text{softmax}\left(\frac{\left(\mathbf{Q}\mathbf{W}_i^Q\right)\left(\mathbf{K}\mathbf{W}_i^K\right)^T}{\sqrt{d_v}}\right)\left(\mathbf{V}\mathbf{W}_i^V\right), \quad (5)$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_{model} \times d_v}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{model} \times d_v}$ and $\mathbf{W}_i^V \in \mathbb{R}^{d_{model} \times d_v}$, with $1 \leq i \leq h$, are learnable projection matrices for a single attention head. Here, L represents the sequence length, d_{model} denotes the dimensionality of the embeddings of the model, h is the number of heads, and $d_v = \frac{d_{model}}{h}$ designates the dimensionality of each head.

The primary role of multi-head attention [28] is to enhance the ability of the model to capture complex patterns in the latent space, whereby each head independently learns a distinct representation of the input data. In the case of a h-head attention block, the outputs from all attention heads are concatenated and projected back onto the original latent space by an output projection matrix $\mathbf{W}^O$, as

$$\begin{aligned} &\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O, \\ &\text{where head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V). \end{aligned} \quad (6)$$

Therefore, the four weight matrices in the Attention block are respectively the query projection weights $\mathbf{W}^Q = \left[\mathbf{W}_1^Q, \mathbf{W}_2^Q, \ldots, \mathbf{W}_h^Q\right] \in \mathbb{R}^{d_{model} \times h \cdot d_v}$, the key projection weights $\mathbf{W}^K = \left[\mathbf{W}_1^K, \mathbf{W}_2^K, \ldots, \mathbf{W}_h^K\right] \in \mathbb{R}^{d_{model} \times h \cdot d_v}$, the value projection weights $\mathbf{W}^V = \left[\mathbf{W}_1^V, \mathbf{W}_2^V, \ldots, \mathbf{W}_h^V\right] \in \mathbb{R}^{d_{model} \times h \cdot d_v}$, and the output projection weights $\mathbf{W}^O = \left[\mathbf{W}_1^O, \mathbf{W}_2^O, \ldots, \mathbf{W}_h^O\right] \in \mathbb{R}^{h \cdot d_v \times d_{model}}$. We refer to $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$, and $\mathbf{W}^O$ as the MHA weight matrices.

By representing the MHA weights as tensor network diagrams [20], Fig. 2 illustrates the difference between LASER, TRAWL and the Tucker decomposition with shared factor matrices used in our proposed method (See section III), when applied to these weights. Note that in these diagrams, a tensor is denoted by a circle, with each line emanating from the circle corresponding to a tensor dimension index. Also, connecting two index lines implies a summation over the connected indices.

III. Proposed Methodology

By exploiting the intuitions behind MHA [17]–[19], we propose a novel and intuitive multi-head tensorisation method. It first tensorises the MHA weights into a set of 3D tensors, each corresponding to an attention head. Then, Tucker decomposition is applied to the tensorised weights of each attention head in a single transformer layer, while sharing a common set of factor matrices across such decompositions. Uniquely, this ensures that the weights of the different attention heads are in a shared subspace characterised by a common set of factor matrices. By structurally denoising the attention weights using a shared higher-dimensional low-rank structure, our proposed framework is found to both improve LLM reasoning and simultaneously achieve compression in the MHA blocks of LLMs.

A. Multi-head Tensorisation

Despite their multi-head nature, the MHA weight matrices are usually stored in 2D formats to accelerate computation. To build a tensor from those matrices, we tensorise the model by first folding the weights into higher-dimensional

formats, before applying tensor decompositions to compress the resulting weight tensor. To this end, we develop a multi-head tensorisation technique based on the intuitions about MHA, in order to naturally tensorise the original 2D query, key, value, and output projection weight matrices into a set of 3D tensors. More specifically, as shown in Step 1 and 2 in the left part of Fig. 1, the tensorisation process starts by splitting the four global weight matrices in a single transformer layer, $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$, and $\mathbf{W}^{O^T} \in \mathbb{R}^{d_{model} \times h \cdot d_v}$, into the local sub-matrices, $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V, \mathbf{W}_i^{O^T} \in \mathbb{R}^{d_{model} \times d_v}$, belonging to each attention head i, where $1 \leq i \leq h$. Next, for each attention head, the four 2D sub-matrices $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V, \mathbf{W}_i^{O^T}$ are stacked into a 3D tensor, $\mathcal{W}_i \in \mathbb{R}^{d_{model} \times d_v \times 4}$, as

$$\mathcal{W}_{i[:,:,j]} = \begin{cases} \mathbf{W}_i^Q & \text{if } j = 1, \\ \mathbf{W}_i^K & \text{if } j = 2, \\ \mathbf{W}_i^V & \text{if } j = 3, \\ \mathbf{W}_i^{O^T} & \text{if } j = 4. \end{cases} \tag{7}$$

This process can then repeated for all h heads before stacking all tensors $\{\mathcal{W}_i\}_{i=1}^h$ together in order to tensorise all MHA weight matrices of a single transformer layer into a 4D tensor, $\mathcal{W}_{all} \in \mathbb{R}^{d_{model} \times d_v \times 4 \times h}$, as

$$\mathcal{W}_{all[:,:,:,i]} = \mathcal{W}_i, \text{ for } 1 \leq i \leq h. \tag{8}$$

Such a multi-head tensorisation process converts the attention weight matrices into a higher-dimensional format to prepare for the utilisation of Tucker decomposition.

B. Tucker Decomposition with Shared Factor Matrices

As shown in Fig. 1, Tucker decomposition decomposes a higher-dimensional tensor into a small-scale core tensor, which is of the same order as the original large-scale tensor, and multiple factor matrices. Physically, the core tensor represents the variability information in a subspace designated by the factor matrices. This makes it possible to apply Tucker decomposition to multiple attention weight tensors, to enable sharing the same set of factor matrices in order to enforce different information to reside within the same subspace.

To this end, in our approach, Tucker decomposition is applied to each of the h 3D tensors, $\{\mathcal{W}_i\}_{i=1}^h$, defined in Eq. (7), while sharing a common set of factor matrices, $\mathbf{U}^{(1)} \in \mathbb{R}^{d_{model} \times R_1}, \mathbf{U}^{(2)} \in \mathbb{R}^{d_v \times R_2}, \mathbf{U}^{(3)} \in \mathbb{R}^{4 \times R_3}$. With the variability information of the i-th attention head weights after Tucker decomposition being contained in a 3D tensor, $\mathcal{G}_i \in \mathbb{R}^{R_1 \times R_2 \times R_3}$, the weights of each attention head, $\mathcal{W}_i$, can be expressed as

$$\mathcal{W}_i = \mathcal{G}_i \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}, \text{ for } 1 \leq i \leq h. \tag{9}$$

Notice that this expression can be conveniently written as a special variant of the Tucker decomposition of a 4D tensor, in the form

$$\mathcal{W}_{all} = \mathcal{G}_{all} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \times_4 \mathbf{I}, \tag{10}$$

or in its element-wise definition form as

$$\mathcal{W}_{all[i_1,i_2,i_3,i_4]} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \mathcal{G}_{all[r_1,r_2,r_3,i_4]} \mathbf{U}^{(1)}_{[i_1,r_1]} \mathbf{U}^{(2)}_{[i_2,r_2]} \mathbf{U}^{(3)}_{[i_3,r_3]}, \tag{11}$$

where $\mathcal{W}_{all} \in \mathbb{R}^{d_{model} \times d_v \times 4 \times h}$ is defined in Eq. (8) and represents the tensor containing all attention weights in a single transformer layer, while $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}$, and $\mathbf{U}^{(3)}$ are the shared factor matrices. The term $\mathbf{I} \in \mathbb{R}^{h \times h}$ is an identity matrix which can be omitted, and the 4D core tensor, $\mathcal{G}_{all} \in \mathbb{R}^{R_1 \times R_2 \times R_3 \times h}$, is defined as

$$\mathcal{G}_{all[:,:,:,i]} = \mathcal{G}_i, \text{ for } 1 \leq i \leq h. \tag{12}$$

The denoising process is performed by approximating the original weight tensors of each attention head using the Tucker decomposition; in other words, given a set of multilinear ranks, we denoise the tensorised MHA weights by minimizing

$$\frac{1}{2} \left\| \sum_{i=1}^{h} \mathcal{W}_i - \sum_{i=1}^{h} \mathcal{G}_i \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \right\|_F^2 \tag{13}$$

or in an equivalent format

$$\frac{1}{2} \left\| \mathcal{W}_{all} - \mathcal{G}_{all} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \right\|_F^2. \tag{14}$$

In practice, we utilised the TensorLy library [30] to implement this special variant of Tucker decomposition based on the Higher Order Orthogonal Iterations (HOOI) algorithm [31].

Remark 1. *Observe from Eq. (13) that the weights of each attention head are sharing a common set of factor matrices $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}$ and $\mathbf{U}^{(3)}$. At the same time, each attention head is assigned its own Tucker core tensor. We conjecture that this design aligns with the intuition that attention heads within a single transformer layer capture patterns at similar abstraction levels with different specialisations.*

Remark 2. *The tensor decomposition process in Eqs. (10) - (14) allows us to structurally denoise the attention weight matrices according to a shared higher-dimensional low-rank structure. Additionally, this also allows for parameter compression, by representing the original weight tensor through smaller-sized factors.*

IV. Experiments

Comprehensive experiments were conducted to verify the performance of our proposed framework on four benchmark reasoning datasets and three LLMs; this includes both the encoder-only and decoder-only architectures. In the experiments, our framework was applied in a layer-selective fashion similar to [12], to enable fair comparisons. For example, we applied the proposed method in Section III to only a single transformer layer at a time. The results demonstrate that our model is capable of significantly improving the reasoning capabilities of the original LLM while enabling their parameter compression. Furthermore, experimental results show that our method can be used in conjunction with existing FFN-only compression methods, such as LASER [12], which improved

the reasoning abilities of LLMs by denoising the FFN weights. Finally, we validated our framework with an ablation study which compares the proposed approach against that of denoising separate tensors for each query, key, value and output weight matrices in a transfomer layer. The experiments were conducted using NVIDIA A100 GPUs.

TABLE II
PERFORMANCE COMPARISON OF ROBERTA, GPT-J, AND LLAMA2, WITH AND WITHOUT APPLYING OUR METHOD, TESTED ON FOUR BENCHMARK DATASETS. THE HIGHEST ACCURACY ↑ AND THE LOWEST LOSS ↓ FOR EACH MODEL AND DATASET COMBINATION ARE DENOTED IN BOLD. CR ↑ STANDS FOR COMPRESSION RATE OF THE MHA PARAMETERS IN A TRANSFORMER LAYER. "-" INDICATES NO COMPRESSION.

Dataset		Model Name					
		RoBERTa		GPT-J		LLaMA2	
		Original	Ours	Original	Ours	Original	Ours
HotPotQA	Acc	6.1	**7.33**	19.6	**20.15**	16.5	**18.44**
	Loss	10.99	**10.00**	**3.40**	4.49	**3.15**	9.80
	CR	-	1.12	-	247.30	-	3.54
FEVER	Acc	50.0	**50.45**	50.2	**58.94**	59.3	**66.75**
	Loss	2.5	**1.47**	1.24	**1.02**	1.02	**1.01**
	CR	-	3.74	-	14.69	-	3.54
Bios Profession	Acc	64.5	**72.57**	75.6	**81.18**	85.0	**86.61**
	Loss	**4.91**	6.64	4.64	**4.57**	**4.19**	4.54
	CR	-	8.78	-	74.68	-	3.54
BigBench-WikidataQA	Acc	28.0	**32.72**	51.8	**68.81**	59.5	**60.37**
	Loss	9.07	**8.72**	3.52	**2.63**	4.19	**2.38**
	CR	-	2.52	-	46.77	-	5.81

A. LLM Models

We tested our proposed method on three LLM models: RoBERTa 125M [32], GPT-J 6B [33], and LLaMA 2 7B [4].

- RoBERTa has an encoder-only architecture and predicts missing tokens within a given context. Thus, we appended five <mask> tokens to each question before inputting it to the RoBERTa model.
- Both GPT-J and LLaMA 2 are decoder-only models and can directly generate tokens autoregressively, by predicting the next token based on the preceding tokens.

B. Datasets

We adopted the same data pre-processing processes as in LASER [12] for the 4 benchmark reasoning datasets, *HotPotQA* [34], *FEVER* [35], *Bios Profession* [36], and *BigBench-WikidataQA* [37].

a) HotPotQA: The *HotPotQA* is a large-scale question-answering dataset. Question-answer pairs were extracted from its training and validation sets. The input texts were tokenized using the LLaMA 2 [4] tokenizer with samples exceeding 15 tokens being discarded. Prompts were formatted as "<question> The answer is", where <question> was replaced by the actual question. The model was allowed to generate up to max_len tokens, and if the answer appearred in the generated text, it was considered correct.

b) FEVER: The Fact Extraction and VERification (*FEVER*) dataset was developed to evaluate fact-checking systems. It contains binary labels: 0 (false) and 1 (true). Claims with conflicting labels were filtered to ensure consistency. Question-answer pairs were extracted from its validation and test sets. Prompts were structured as: "Consider the following claim: <claim>. Is this claim true or false? The claim is", where <claim> was replaced by the actual statement. If the probability of a claim being true exceeded that of it being false, the claim was classified as true and otherwise false.

c) Bios Profession: The *Bios Profession* dataset is a benchmark for analyzing gender biases in occupational and professional contexts. The task involves predicting the profession given a biography. Question-answer pairs were derived from the validation set, focusing on the same 10 professions used in the LASER paper. Prompts were structured as: "Consider the following text: <bio>. What is the profession of the person in this text? The profession of this person is" where <bio> was replaced by the actual biography. The model was tasked to output the profession with the highest probability among the 10 possible professions.

d) BigBench-WikidataQA: In Beyond the Imitation Game Benchmark (BIG-Bench) dataset, the WikidataQA subset focuses on question answering (QA) using structured knowledge from Wikidata. Question-answer pairs were derived from the train and validation set. The model was allowed to generate up to max_len tokens, and if the answer appearred in the generated text, it was considered correct.

C. Experimental Results

To assess the enhancement in the reasoning capabilities of LLMs with our proposed framework, we applied it to three LLMs and evaluated their performance on the four reasoning datasets mentioned in Section IV-B. Table II shows the accuracy, loss, and compression ratio of the LLMs with and without our method applied to the MHA block. Accuracy is the key evaluation metric for model reasoning performances and is measured by the percentage of correctly predicted instances in the test set. Test accuracy was evaluated using the last 20% of data in each dataset. Loss is an indirect performance measure and represents the uncertainty of the model, i.e., the deviation from the true target distribution. The loss is included for completeness and measured by the negative log-likelihood of the ground truth token. The Compression Ratio (CR) quantifies the reduction in model size and is computed as $\text{CR} = \frac{N_{original}}{N_{compressed}}$, where $N_{original}$ and $N_{compressed}$ represent the total number of parameters of the MHA weights in an original single transformer layer and a single compressed transformer layer, respectively. Table II shows that our proposed method consistently improved the performance of the three original models on all four reasoning datasets, in terms of test accuracy, and achieved compression rates of the MHA weights up to 247.3 times.

Remark 3. *Our method is applied to the MHA blocks so that it can be used in conjunction with existing methods to further improve the reasoning capabilities of LLMs. For*

TABLE III

PERFORMANCE OF STAND-ALONE AND HYBRID METHODS FOR LLM COMPRESSION AND REASONING. CASE 1: LASER APPLIED ON ONE WEIGHT MATRIX OF THE FFN BLOCK; CASE 2: LASER APPLIED ON BOTH THE MHA AND FFN BLOCKS; CASE 3: OUR METHOD APPLIED ON THE MHA BLOCK; AND LASER APPLIED ON THE FFN BLOCK. THE HIGHEST ACCURACY ↑ AND THE LOWEST LOSS ↓ OF EACH MODEL AND DATASET COMBINATION ARE DENOTED IN BOLD.

Dataset		Model Name: RoBERTa Case 1	RoBERTa Case 2	RoBERTa Case 3 (Ours)	GPT-J Case 1	GPT-J Case 2	GPT-J Case 3 (Ours)	LLaMA2 Case 1	LLaMA2 Case 2	LLaMA2 Case 3 (Ours)
HotPotQA	Acc	6.7	5.24	**7.05**	19.5	19.62	**19.91**	17.2	18.88	**19.22**
	Loss	10.53	**8.60**	9.87	**3.39**	5.08	5.07	**2.97**	9.33	9.99
FEVER	Acc	52.3	53.6	**55.23**	56.2	55.59	**58.98**	64.5	65.13	**66.39**
	Loss	1.76	**1.18**	2.61	**1.27**	1.28	1.39	**0.91**	1.11	1.33
Bios Profession	Acc	72.5	71.14	**72.51**	82.1	81.28	**82.52**	86.7	86.07	**87.07**
	Loss	**6.44**	6.62	7.42	4.91	4.61	**4.52**	**4.05**	4.20	**4.05**
BigBench-WikidataQA	Acc	30.7	34.49	**37.40**	65.9	65.68	**68.20**	**62.0**	61.21	61.78
	Loss	**7.69**	8.25	7.86	2.86	2.89	**2.59**	**2.31**	2.35	2.34

example, in combination with LASER [12] which achieves best performance when applied to the FFN blocks.

To evaluate such "hybrid" scenarios as mentioned in Remark 3, we investigated the following three cases:

- Case 1: LASER [12] was applied to one matrix in the FFN block;
- Case 2: LASER [12] was applied to all matrices in the FFN and MHA blocks;
- Case 3: Our method was applied to the MHA block; LASER [12] was applied to matrices in the FFN block.

The results for Case 1 are directly quoted from the best performances reported in [12]. Furthermore, as pointed out by the authors in [12], applying their method multiple times to many weight matrices can yield further improvements. To this end, in Case 2, we applied the LASER method to both the FFN block and the MHA block to obtain the best performance of LASER and allow for a fair comparison with Case 3, where our method was applied to the MHA block while LASER was applied to the FFN block. Table III presents the results obtained for the above three cases. Case 3 achieves the highest test accuracies across the three models and four datasets, except with LLaMA2 on the *BigBench-WikidataQA* dataset. This further highlights the intuition behind our approach which focuses on the MHA weights. Through a careful design of the tensorisation process and the utilisation of current domain knowledge about MHA, our method was capable of structurally denoising the MHA weights according to the intuition of having a shared high-dimensional subspace among the attention heads in a single transformer layer. This not only enhances the interpretability of our approach, but also demonstrates that our framework can be used as a versatile module, in conjunction with methods designed for the FFN weights only, in order to achieve enhanced reasoning and compression in LLMs.

D. Ablation study

To evaluate the impact of stacking the query, key, value, and output weight matrices into a tensor, we compared our framework to the scenario where the same methodology was applied to one of the four types of weight matrices in MHA – query, key, value, and output. Table IV shows that our proposed method was able to consistently achieve the best performance across all four reasoning datasets. This validates our underpinning conjecture of tensorising together all MHA weights in a transformer layer.

TABLE IV

THE IMPACT OF COMPRESSING THE $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$, AND $\mathbf{W}^O$ SEPARATELY AND TOGETHER. OUR PROPOSED METHOD COMPRESSES SIMULTANEOUSLY ALL MHA WEIGHTS IN A TRANSFORMER LAYER. THE HIGHEST ACCURACY ↑ AND THE LOWEST LOSS ↓ OF EACH DATASET ARE DENOTED IN BOLD.

Dataset		GPT-J Original	$\mathbf{W}^Q$	$\mathbf{W}^K$	$\mathbf{W}^V$	$\mathbf{W}^O$	Ours
HotPotQA	Acc	19.6	19.19	19.25	19.70	19.62	**20.15**
	Loss	**3.4**	4.45	4.45	4.43	4.44	4.49
FEVER	Acc	50.2	54.41	53.40	55.86	56.07	**58.94**
	Loss	1.24	1.22	1.22	1.23	1.15	**1.02**
Bios Profession	Acc	75.6	76.06	74.97	79.39	79.71	**81.18**
	Loss	4.64	4.54	4.59	4.46	**4.41**	4.57
BigBench-WikidataQA	Acc	51.8	49.72	51.01	48.82	48.87	**68.81**
	Loss	3.52	3.66	3.58	3.69	3.69	**2.63**

V. CONCLUSION

We have proposed a novel framework for the enhancement of the reasoning abilities of LLMs while simultaneously preforming parameter compression. This has been achieved by exploiting the domain knowledge and empirical evidence about multi-head attention in LLMs, along with a unique multi-head tensorisation and a special variant of the Tucker decomposition. In this way, our framework has explored structurally denoising the weights of each attention head in a transformer layer, according to a shared higher-dimensional low-rank structure among the attention heads. Consequently, this has ensured that the weights of each attention head encode different information within a common higher-dimensional subspace characterised by the common Tucker factor matrices. Our method has been shown to enable parameter compression and enhanced reasoning in both encoder-only and decoder-only LLMs, of which the parameter complexity ranges from hundreds of millions to billions of parameters. Additionally,

our framework can be used in conjunction with existing methods that improve LLM reasoning, such as those based on denoising FFN weights, to yield further performance gains. Through an ablation study, we have validated the advantage and performance gain obtained by our proposed multi-head tensorisation process.

a) Limitations: Similar to other existing weight denoising methods, we have found that for different datasets, our method achieves the best results under different hyperparameter settings. Our future work will focus on finding unified and generalisable hyperparameters settings for both our proposed method and other existing methods.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[3] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[4] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[5] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, *et al.*, "The Llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[6] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao, "Learning Deep Transformer Models for Machine Translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1810–1822, July 2019.

[7] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language Modeling with Deep Transformers," in *Proceedings of Interspeech 2019*, pp. 3905–3909, 2019.

[8] Y. Liu and M. Lapata, "Text Summarization with Pretrained Encoders," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3730–3740, 2019.

[9] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *ArXiv*, vol. abs/1503.02531, 2015.

[10] H. Sajjad, F. Dalvi, N. Durrani, and P. Nakov, "On the effect of dropping layers of pre-trained transformer models," *Computer Speech & Language*, vol. 77, p. 101429, 2023.

[11] A. Fan, E. Grave, and A. Joulin, "Reducing Transformer Depth on Demand with Structured Dropout," in *Proceedings of the International Conference on Learning Representations*, 2020.

[12] P. Sharma, J. T. Ash, and D. Misra, "The Truth is in There: Improving Reasoning in Language Models with Layer-Selective Rank Reduction," in *Proceedings of The Twelfth International Conference on Learning Representations*, 2024.

[13] R. Saha, N. Sagan, V. Srivastava, A. Goldsmith, and M. Pilanci, "Compressing Large Language Models using Low Rank and Low Precision Decomposition," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[14] G. G. Calvi, A. Moniri, M. Mahfouz, Q. Zhao, and D. P. Mandic, "Compression and Interpretability of Deep Neural Networks Via Tucker Tensor Layer: From First Principles to Tensor Valued Back-Propagation," *arXiv preprint arXiv:1903.06133*, 2019.

[15] M. Xu, Y. L. Xu, and D. P. Mandic, "TensorGPT: Efficient compression of the embedding layer in LLMs based on the tensor-train decomposition," *arXiv preprint arXiv:2307.00526*, 2023.

[16] Y. Luo, H. Patel, Y. Fu, D. Ahn, J. Chen, Y. Dong, and E. E. Papalexakis, "TRAWL: Tensor Reduced and Approximated Weights for Large Language Models," *arXiv preprint arXiv:2406.17261*, 2024.

[17] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does BERT look at? an analysis of BERT's attention," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, Aug. 2019.

[18] J. Vig, "A Multiscale Visualization of Attention in the Transformer Model," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 37–42, July 2019.

[19] J. Vig and Y. Belinkov, "Analyzing the Structure of Attention in a Transformer Language Model," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 63–76, Aug. 2019.

[20] R. Orús, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states," *Annals of Physics*, vol. 349, pp. 117–158, 2014.

[21] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[22] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[23] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[24] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[25] M. A. O. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: Tensorfaces," in *Proceedings of Computer Vision—ECCV 2002: 7th European Conference on Computer Vision Copenhagen, Denmark, May 28–31, 2002 Proceedings, Part I 7*, pp. 447–460, 2002.

[26] G. Iacovides, W. Zhou, and D. Mandic, "Towards LLM-guided Efficient and Interpretable Multi-linear Tensor Network Rank Selection," *arXiv preprint arXiv:2410.10728*, 2024.

[27] Y.-B. Zheng, X.-L. Zhao, J. Zeng, C. Li, Q. Zhao, H. C. Li, and T.-Z. Huang, "SVDinsTN: A Tensor Network Paradigm for Efficient Structure Search from Regularized Modeling Perspective," in *Proceedings of 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22413–22422, 2024.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[29] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv e-prints*, pp. arXiv–1607, 2016.

[30] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "TensorLy: Tensor Learning in Python," *Journal of Machine Learning Research*, vol. 20, no. 26, pp. 1–6, 2019.

[31] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-$(r_1, r_2, ..., r_n)$ approximation of higher-order tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.

[32] Y. Liu, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv preprint arXiv:1907.11692*, vol. 364, 2019.

[33] B. Wang and A. Komatsuzaki, "GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model." https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[34] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Oct.-Nov. 2018.

[35] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: a Large-scale Dataset for Fact Extraction and VERification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, June 2018.

[36] M. De-Arteaga, A. Romanov, H. Wallach, J. Chayes, C. Borgs, A. Chouldechova, S. Geyik, K. Kenthapadi, and A. T. Kalai, "Bias in bios: A case study of semantic representation bias in a high-stakes setting," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 120–128, 2019.

[37] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, Sept. 2015.