LatentLLM: Activation-Aware Transform to Multi-Head Latent Attention

Toshiaki Koike-Akino[1], Xiangyu Chen[2*], Jing Liu[1], Ye Wang[1], Pu (Perry) Wang[1], Mathew Brand[1]

[1]Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139, USA.
[2]Sony Electronics Inc.
{koike, jiliu, yewang, pwang, brand}@merl.com

Abstract

Modern foundation models such as large language models (LLMs) require a massive amount of computational and memory resources. We propose a new framework to convert such LLMs into a reduced-dimension latent structure. Our method extends a local activation-aware tensor decomposition to a global attention-aware joint tensor decomposition. Our framework can significantly improve the model accuracy over the existing model compression methods when reducing the latent dimension to realize computationally/memory-efficient LLMs. We show the benefit on several benchmark including multi-modal reasoning tasks.

Introduction

Large language models (LLMs) (Touvron et al. 2023; Achiam et al. 2023) and large multi-modal models (LMMs) (Liu et al. 2023) have shown excellent performance across a variety of general tasks (Wei et al. 2022; Katz et al. 2024; Bubeck et al. 2023). Nonetheless, these models having billions of parameters demand significant computational resources (Schwartz et al. 2020). Towards increasing the accessibility and sustainability of LLMs/LMMs, extensive efforts have been devoted to model compression (Xu and McAuley 2023; Zhu et al. 2024; Bai et al. 2024a): e.g., partial activation (Jiang et al. 2024; Lin et al. 2024a), pruning (Frantar and Alistarh 2023; Sun et al. 2023; Bai et al. 2024b; Hassibi, Stork, and Wolff 1993), quantization (Frantar et al. 2022; Lin et al. 2024b; Wang et al. 2024a), knowledge distillation (Hsieh et al. 2023; DeepSeek-AI 2025; Hwang et al. 2024), and low-rank factorization (Yuan et al. 2023; Liu et al. 2024; Hwang et al. 2024; Saxena et al. 2024).

More recently, the reduced-dimension LLM DeepSeek-V3 (Liu et al. 2024) has attracted much attention for its high efficiency and performance. It employs a low-rank architecture called multi-head latent attention (MLA) to compress the standard multi-head attention (MHA), realizing an efficient KV cache (Chang et al. 2024; Saxena et al. 2024), accelerated training, and high-performance inference. In this paper, we provide a novel solution to convert a pretrained LLM/LMM built with MHA into a compressed LLM/LMM with a type of MLA. Our approach is motivated by a global compression framework introduced in SparseLLM (Bai et al. 2024b) and Q-VLM (Wang et al. 2024a). Although the original method was designed for pruning/quantization, we adopt it for tensor rank reduction. We further extended it to the joint compression of MHA, while the original SparseLLM was for compressing the multi-layer perceptron (MLP) part. Our derived solution is based on a high-order tensor-rank decomposition to jointly factorize multiple linear layers.

The contributions of our paper are summarized below.

- We propose a novel low-rank decomposition approach called LatentLLM to compress LLMs/LMMs.
- We discuss an optimal pre-conditioning for activation-aware SVD.
- We reveal that a choice of junction matrix can significantly reduce the model size.
- We then introduce an attention-aware joint SVD framework to compress multiple weights at the same time.
- Several experiments validate that our LatentLLM approach can improve the performance of LLM/LMM compression over existing methods.
- The LLaVA/Qwen-VL compressed with LatentLLM offer a significant advantage in multi-modal reasoning.

Related Work

Model Compression The field of model compression for LLMs/LMMs has seen a surge of innovative techniques aimed at mitigating the substantial computation and memory requirements (Zhu et al. 2024; Yuan et al. 2024). Various methods have emerged to address this challenge, each taking a unique approach to reduce the memory footprint of LLMs. These methods primarily fall into four categories: weight quantization (Lin et al. 2024b; Frantar et al. 2022; Wang et al. 2024a), network pruning (LeCun, Denker, and Solla 1989; Hassibi, Stork, and Wolff 1993; Frantar and Alistarh 2023; Bai et al. 2024b), knowledge distillation (Hsieh et al. 2023; DeepSeek-AI 2025; Hwang et al. 2024), and low-rank factorization (Yuan et al. 2023; Liu et al. 2024; Hwang et al. 2024; Saxena et al. 2024; Saha et al. 2024).

Among them, weight quantization has gained significant traction in the context of large foundation models due to its effectiveness. However, all four compression techniques are

*This work was done when X. Chen was an intern at MERL.

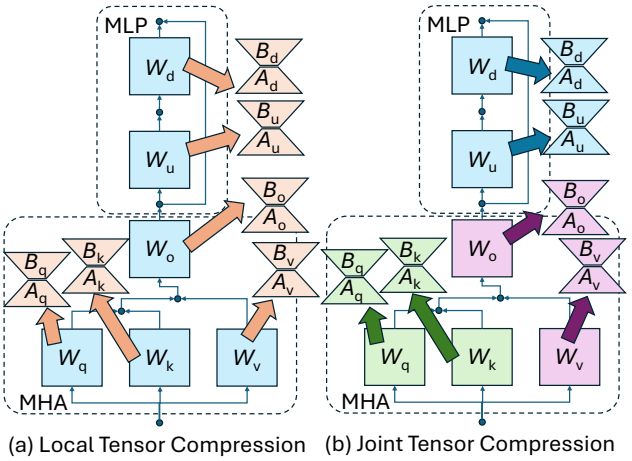

Figure 1: Reduced-dimension LLM/LMM with low-rank tensor decomposition. (a) each linear modules are locally compressed by activation-aware tensor decomposition. (b) multiple linear modules are globally compressed by attention-aware tensor decomposition.

orthogonal and can be applied together. We hence introduce a novel low-rank decomposition method which jointly compresses multiple layers of an LLM in a training-free manner.

Low-Rank Decomposition In the realm of low-rank decomposition (Schotthöfer et al. 2022) for neural network compression, existing methods typically involve decomposing weight matrices of pre-trained networks using techniques like Singular Value Decomposition (SVD) or tensor decomposition, followed by fine-tuning the factorized network (Denton et al. 2014; Sainath et al. 2013). LoSparse (Li et al. 2023) uses low-rank approximation plus a sparse matrix to compress the weight matrix in transformers. Similarly, CALDERA (Saha et al. 2024) uses low-rank approximation plus a quantized matrix. ASVD (Yuan et al. 2023) significantly improves the low-rank decomposition by dealing with activation statistics. It was applied to SVD-LLM (Wang et al. 2024b) and Palu (Chang et al. 2024). DeepSeek-V3 (Liu et al. 2024) employs the similar latent reduction via MLA to make MHA efficient and capable. Eigen attention (Saxena et al. 2024) is highly related to MLA.

LatentLLM: Tensor Compression

Reduced-Dimension LLM/LMM

Figure 1 illustrates the basic transformer architecture consisting of MHA and MLP, used in some LLMs/LMMs. For MLP, there are up and down projections, whereas MHA has query/key/value/output projections. By transforming those dense weight matrices into low-rank decompositions, we can realize an efficient latent LLM/LMM having potential benefits: (i) fewer-parameter model size; (ii) KV cache reduction; (iii) accelerated processing; (iv) lower-power consumption. In fact, some recent LLM models such as DeepSeek-V3 (Liu et al. 2024) demonstrated efficiency and high-performance with MLA. We focus on compressing a pre-trained LLM/LMM by converting MHA into a type of MLA in a zero-shot fashion, i.e., without any fine-tuning.

Most compression methods are based on a local loss minimization to approximate each weight individually. Motivated by recent work towards global optimization (Bai et al. 2024b; Wang et al. 2024a), we propose a joint tensor compression framework that we call "LatentLLM." Specifically, we derive a mathematical solution to jointly decompose a pair of query and value projections, a pair of value and output projections, and a pair of up and down projections to compress LLMs. We first address activation-aware compression to provide some new insights on the choice of pre-conditioner and junction matrix below.

Activation-Aware SVD: Pre-Conditioning

A pioneering work by ASVD (Yuan et al. 2023) introduced a way to compress a layer depending on the activation statistics. Consider a pretrained-weight $W \in \mathbb{R}^{d' \times d}$ to compress with a lower-rank decomposition $\hat{W} = BA$ for compression matrix $A \in \mathbb{R}^{r \times d}$ and decompression matrix $B \in \mathbb{R}^{d' \times r}$. Using the input activation $X \in \mathbb{R}^{d \times l}$ (l is the calibration sample length), ASVD aims to minimize the activation loss:

$$\mathcal{L}_1 = \mathbb{E}_X \big\| WX - \hat{W}X \big\|^2 = \mathbb{E}_X \big\| WX - BAX \big\|^2, \quad (1)$$

instead of the naïve weight-based loss:

$$\mathcal{L}_0 = \big\| W - \hat{W} \big\|^2 = \big\| W - BA \big\|^2. \quad (2)$$

It is well-known that the optimal solution to minimize $\mathcal{L}_0$ can be given by the plain SVD of W. To minimize $\mathcal{L}_1$, ASVD introduced a pre-conditioner $P \in \mathbb{R}^{d \times d}$ to whiten the statistical impact of the activation X. Specifically, ASVD uses the low-rank matrices given by whitened SVD:

$$BAP = \mathsf{svd}_r[WP], \quad (3)$$

where $\mathsf{svd}_r[\cdot]$ denotes the rank-r truncated SVD.

Although ASVD originally suggested a diagonal ℓ_1-norm pre-conditioning, the optimal pre-conditioning matrix P can be given by reformulating $\mathcal{L}_1$ as follows:

$$\mathcal{L}_1 = \mathrm{tr}\big[(W - BA)\mathbb{E}_X[XX^\top](W - BA)^\top\big] \quad (4)$$

$$= \big\| (W - BA)C^{\frac{1}{2}} \big\|^2 = \big\| WC^{\frac{1}{2}} - BAC^{\frac{1}{2}} \big\|^2, \quad (5)$$

where $C = \mathbb{E}_X[XX^\top] \in \mathbb{R}^{d \times d}$ is a covariance (precisely, auto-correlation) of input activation. Hence, the above loss can be minimized by the SVD: $BAC^{\frac{1}{2}} = \mathsf{svd}_r[WC^{\frac{1}{2}}]$. Accordingly, it is found that the optimal pre-conditioner is the square-root covariance: $P = C^{\frac{1}{2}}$. Given the finite calibration data X, we can estimate the covariance as $C = XX^\top + \lambda I$, where the damping factor $\lambda \in \mathbb{R}_+$ corresponds to the shrunk estimator (Ledoit and Wolf 2004).

Remark 1 *Different pre-conditioning methods were introduced in several techniques including pruning and quantization, as listed in Table 1. As those variants are sub-optimal, we use the optimal root covariance: $P = C^{\frac{1}{2}}$. See more discussion in Appendix.*

Conditioning P	Expression	Reference
Identity	I	Plain SVD (Sainath et al. 2013; Denton et al. 2014)
Hessian	$\mathsf{diag}[(XX^\top + \lambda I)^{-1}]^{\frac{-1}{2}}$	OBS (Hassibi et al.); GPTQ (Frantar et al.); SparseGPT (Frantar and Alistarh)
ℓ_1-norm	$\mathsf{diag}\left[\sum_j \lvert X_{1,j}\rvert, \ldots, \sum_j \lvert X_{d,j}\rvert\right]^\alpha$	ASVD (Yuan et al. 2023); AWQ (Lin et al. 2024b)
ℓ_2-norm	$\mathsf{diag}[XX^\top]^{\frac{1}{2}}$	WandA (Sun et al. 2023)
Covariance	$XX^\top + \lambda I$	CorDA (Yang et al. 2024)
Root-Covariance	$(XX^\top + \lambda I)^{\frac{1}{2}}$	LatentLLM (Ours)

Table 1: Variants of pre-conditioning matrices P for activation-aware distillation.

Junction Matrix for Model Compression

In fact, the solution of (3) does not have a unique decomposition into low-rank matrices B and A. The truncated SVD is written as

$$USV = \mathsf{svd}_r[WP], \tag{6}$$

where $U \in \mathbb{R}^{d' \times r}$, $S \in \mathbb{R}^{r \times r}$, and $V \in \mathbb{R}^{r \times d}$ are the left singular unitary matrix, singular-value diagonal matrix, and right singular unitary matrix, respectively. The decompression and compression matrices B and A can be expressed:

$$B = USJ, \qquad A = J^+VP^+, \tag{7}$$

where $J \in \mathbb{R}^{r \times r}$ is a junction matrix and $[\cdot]^+$ denotes the pseudo inverse. Choosing any junction matrix that satisfies $SJJ^+ = S$ has no impact on the loss. Hence, there is few literature discussing the choice of J. Typically, one may use $J = I$ to put singular-values into the decompression matrix; $J = S^+$ to put it into the compression matrix; or $J = [S^{\frac{1}{2}}]^+$ to split it across both matrices equally.

However, a certain choice of J has a noticeable advantage to reduce the number of parameters and floating-point operations (FLOPs). We can write the whitened right-singular matrix VP^+ as two sub-blocks:

$$VP^+ = [V_1 \quad V_2], \tag{8}$$

for $V_1 \in \mathbb{R}^{r \times r}$ and $V_2 \in \mathbb{R}^{r \times (d-r)}$. When we use $J = V_1$, the compression matrix A will contain an identity block as long as V_1 is non-singular:

$$A = J^+VP^+ = V_1^+ [V_1 \quad V_2] = \left[I \quad V_1^+V_2\right]. \tag{9}$$

This can greatly reduce the number of parameters from $r(d' + d)$ to $r(d' + d) - r^2$, as well as the FLOPs, because no computation is needed for the identity projection.

For example, when the hidden size is $d = d'$, even if we compress it by 25%, i.e., the latent size is $r = 0.75d$, the total number of parameters will be $r(d'+d) = 1.5d^2$, which is 50% more than the original d^2. This increased FLOPs hinders the low-rank compression of LLMs, even with the KV cache benefit (Liu et al. 2024; Yuan et al. 2023; Chang et al. 2024). Nevertheless, with our identity block form, we can always reduce the number of parameters regardless of the latent size, i.e., $r(d' + d) - r^2 < d'd$ for $r < \min(d', d)$. For the above example of 25% latent compression, we can achieve $r(d' + d) - r^2 = (15/16)d^2 < d^2$. Figure 2 depicts the role of the pre-conditionning and junction matrices for the activation-aware compression. We also illustrate the tensor diagrams to understand the flexibility of tensor mapping.

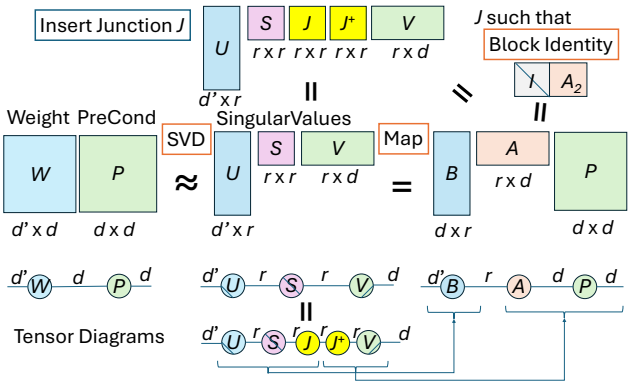

Figure 2: Activation-aware compression with preconditioning and junction matrix. The junction matrix J can be adjusted such that A or B is block identity to save the number of parameters and inference computation.

Remark 2 *Pivoting columns can solve the case when the left-most sub-block V_1 is singular. The pivoting does not require any FLOPs in inference while additional memory is required to record the permutation index.*

LatentLLM: Joint Tensor Compression

The SVD described above is optimal in the sense that the local error is minimized for the single tensor compression, whereas it does not guarantee global optimality. Motivated by the global loss minimization framework introduced by SparseLLM (Bai et al. 2024b), we propose a joint tensor compression technique which factorizes multiple tensors in adjacent modules concurrently.

Multi-Head Latent Attention: Joint QK SVD

First, we consider a joint compression of query (Q) and key (K) projections in MHA to convert into MLA. The attention map is the dot product of query and key features:

$$M_i = X^\top W_{\mathrm{q},i}^\top W_{\mathrm{k},i} X, \tag{10}$$

where $M_i \in \mathbb{R}^{l \times l}$ is the ith head attention map before softmax operation, $W_{\mathrm{q},i} \in \mathbb{R}^{d_\mathrm{h} \times d}$ is the ith head query projection matrix, and $W_{\mathrm{k},i} \in \mathbb{R}^{d_\mathrm{h} \times d}$ is the ith head key projection matrix. Here, d_h is the head dimension, which is often $d_\mathrm{h} = d/h$ for the number of heads h.

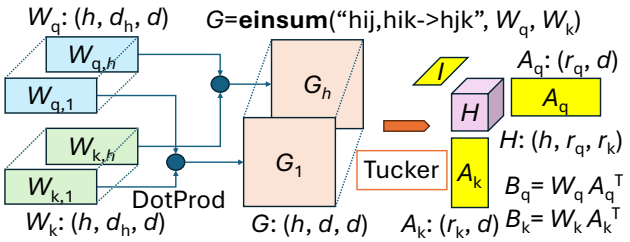

Figure 3: Tucker decomposition for joint QK compression. The compression matrices A_{q} and A_{k} correspond to the Tucker tensor planes, while $H = A_{\mathrm{q}} G A_{\mathrm{k}}^{\top}$ is the Tucker tensor core. For simplicity, we omit junction matrices and pre-conditioning matrix.

Rather than individually compressing Q and K projections, we consider minimizing the attention map error:

$$\mathcal{L}_2 = \sum_{i=1}^{h} \left\| M_i - \hat{M}_i \right\|^2, \tag{11}$$

where $\hat{M}_i$ is the ith head latent attention with the low-rank compression:

$$\hat{M}_i = X^{\top} A_{\mathrm{q}}^{\top} B_{\mathrm{q},i}^{\top} B_{\mathrm{k},i} A_{\mathrm{k}} X, \tag{12}$$

where $A_{\mathrm{q}} \in \mathbb{R}^{r_{\mathrm{q}} \times d}$ is the Q compression matrix, $A_{\mathrm{k}} \in \mathbb{R}^{r_{\mathrm{k}} \times d}$ is the K compression matrix, $B_{\mathrm{q},i} \in \mathbb{R}^{d_{\mathrm{h}} \times r_{\mathrm{q}}}$ is the ith head Q decompression matrix, and $B_{\mathrm{k},i} \in \mathbb{R}^{d_{\mathrm{h}} \times r_{\mathrm{k}}}$ is the ith head K decompression matrix, respectively. Here, r_{q} and r_{k} are the latent dimensions for Q and K.

Similar to (5), we can write

$$\mathcal{L}_2 = \sum_{i=1}^{h} \Big\| \underbrace{C^{\frac{1}{2}} W_{\mathrm{q},i}^{\top} W_{\mathrm{k},i} C^{\frac{1}{2}}}_{G_i \in \mathbb{R}^{d \times d}} - \underbrace{C^{\frac{1}{2}} A_{\mathrm{q}}^{\top}}_{A_{\mathrm{q}}'^{\top}} \underbrace{B_{\mathrm{q},i}^{\top} B_{\mathrm{k},i}}_{H_i \in \mathbb{R}^{r_{\mathrm{q}} \times r_{\mathrm{k}}}} \underbrace{A_{\mathrm{k}} C^{\frac{1}{2}}}_{A_{\mathrm{k}}'} \Big\|^2. \tag{13}$$

This is known as a high-order SVD (HOSVD) problem to decompose for the 3-mode tensor $G \in \mathbb{R}^{h \times d \times d}$, whose ith slice is G_i. A_{q}' corresponds to the 2nd tensor plane, A_{k}' is the 3rd tensor plane, and $H \in \mathbb{R}^{h \times r_{\mathrm{q}} \times r_{\mathrm{k}}}$, whose ith slice is H_i, is the tensor core. This is illustrated in Figure 3.

This Tucker tensor decomposition is typically solved by alternating SVD over each slice sequentially. Algorithm 1 shows the pseudo-code of the joint SVD compression for QK latent projections. See the detailed derivations of the joint SVD algorithm in Appendix . Here, we generalize the pre-conditioning matrix P, as not necessarily the optimal $C^{\frac{1}{2}}$. In addition, we explicitly denoted any arbitrary junction matrices that do not change the error. Note that there are additional junction matrices per heads $J_i \in \mathbb{R}^{d_{\mathrm{h}} \times d_{\mathrm{h}}}$ as well as individual Q/K junctions $J_{\mathrm{q}} \in \mathbb{R}^{r_{\mathrm{k}} \times r_{\mathrm{k}}}$ and $J_{\mathrm{k}} \in \mathbb{R}^{r_{\mathrm{k}} \times r_{\mathrm{k}}}$. This suggests that we can further reduce the number of parameters by transforming into the block identity form per head. The total number of parameters will be $(r_{\mathrm{q}}+r_{\mathrm{k}})(d+d_{\mathrm{h}}h) - r_{\mathrm{q}}^2 - r_{\mathrm{k}}^2 - d_{\mathrm{h}}^2 h$, reduced from the original weights $2dd_{\mathrm{h}}h$.

Algorithm 1: Joint SVD to Transform MHA to MLA

Input: Pre-conditioning $P \in \mathbb{R}^{d \times d}$, query projection heads $W_{\mathrm{q},i} \in \mathbb{R}^{d_{\mathrm{h}} \times d}$, key projection heads $W_{\mathrm{k},i} \in \mathbb{R}^{d_{\mathrm{h}} \times d}$, number of heads h, rank $r_{\mathrm{q}}, r_{\mathrm{k}}$, iteration N
Initialize:
$W_{\mathrm{q},i} = W_{\mathrm{q},i} P$ for $i \in \{1, \ldots, h\}$
$W_{\mathrm{k},i} = W_{\mathrm{k},i} P$ for $i \in \{1, \ldots, h\}$
$G_i = W_{\mathrm{q},i}^{\top} W_{\mathrm{k},i}$ for $i \in \{1, \ldots, h\}$
$A_{\mathrm{q}} = \mathsf{RightSingular}_{r_{\mathrm{q}}} \left[\sum_{i=1}^{h} G_i G_i^{\top} \right]$
for $n = 1$ **to** N **do**
  $A_{\mathrm{k}} = \mathsf{RightSingular}_{r_{\mathrm{k}}} \left[\sum_{i=1}^{h} G_i^{\top} A_{\mathrm{q}}^{\top} A_{\mathrm{q}} G_i \right]$
  $A_{\mathrm{q}} = \mathsf{RightSingular}_{r_{\mathrm{q}}} \left[\sum_{i=1}^{h} G_i A_{\mathrm{k}}^{\top} A_{\mathrm{k}} G_i^{\top} \right]$
end for
Output:
$B_{\mathrm{q},i} = J_i^{\top} W_{\mathrm{q},i} A_{\mathrm{q}}^{\top} J_{\mathrm{q}}$ for $i \in \{1, \ldots, h\}$
$B_{\mathrm{k},i} = J_i^{+} W_{\mathrm{k},i} A_{\mathrm{k}}^{\top} J_{\mathrm{k}}$ for $i \in \{1, \ldots, h\}$
$A_{\mathrm{q}} = J_{\mathrm{q}}^{+} A_{\mathrm{q}} P^{+}$
$A_{\mathrm{k}} = J_{\mathrm{k}}^{+} A_{\mathrm{k}} P^{+}$

Remark 3 *Our joint QK SVD can be extended with most positional encoding methods. See Appendix.*

Remark 4 *Attention-aware pruning (Liang et al. 2024) is related to our method, while our derivation provides an optimal tensor rank decomposition and only requires preconditioning matrices.*

Multi-Head Latent Attention: Joint VO SVD

Next, we discuss the joint SVD for value (V) and output (O) projections in MHA. The MHA output can be written as

$$Y' = \sum_{i=1}^{h} W_{\mathrm{o},i} W_{\mathrm{v},i} X \, \mathsf{softmax}[M_i^{\top}], \tag{14}$$

where $W_{\mathrm{o},i} \in \mathbb{R}^{d' \times d_{\mathrm{h}}}$ is the ith head output projection, and $W_{\mathrm{v},i} \in \mathbb{R}^{d_{\mathrm{h}} \times d}$ is the ith head value value projection. We may consider minimizing the loss:

$$\mathcal{L}_3 = \sum_{i=1}^{h} \left\| W_{\mathrm{o},i} W_{\mathrm{v},i} X - \hat{W}_{\mathrm{o},i} \hat{W}_{\mathrm{v},i} X \right\|^2, \tag{15}$$

for the low-rank compression: $\hat{W}_{\mathrm{o},i} = B_{\mathrm{o}} A_{\mathrm{o},i} \in \mathbb{R}^{d' \times d_{\mathrm{h}}}$ and $\hat{W}_{\mathrm{v},i} = B_{\mathrm{v},i} A_{\mathrm{v}} \in \mathbb{R}^{d_{\mathrm{h}} \times d}$ with $B_{\mathrm{o}} \in \mathbb{R}^{d' \times r_{\mathrm{o}}}$, $A_{\mathrm{o},i} \in \mathbb{R}^{r_{\mathrm{o}} \times d_{\mathrm{h}}}$, $B_{\mathrm{v},i} \in \mathbb{R}^{d_{\mathrm{h}} \times r_{\mathrm{v}}}$, and $A_{\mathrm{v}} \in \mathbb{R}^{r_{\mathrm{v}} \times d}$. The MLA output is thus given as

$$\hat{Y}' = \sum_{i=1}^{h} B_{\mathrm{o}} A_{\mathrm{o},i} B_{\mathrm{v},i} A_{\mathrm{v}} X \, \mathsf{softmax}[M_i]. \tag{16}$$

Interestingly, this is also formulated in a similar manner of (13), and it can be solved by the joint SVD algorithm.

Latent MLP: Joint UD SVD

Finally, we address the joint compression of MLP layers which consists of up (U) projection and down (D) projection in typical LLMs/LMMs. Although the global optimization is generally difficult due to the nonlinear activations in

Compression	FLOPs	MACs	Parameters (byte)	Speed (token/sec)	KV Cache (byte)
0%	109.0T	54.5T	13.32G	6.72k	5.37G
20%	87.2T	43.6T	11.06G	7.11k	2.97G
40%	65.4T	32.7T	8.40G	8.35k	1.98G
60%	43.6T	21.8T	5.74G	11.48k	1.21G
80%	21.8T	10.9T	3.08G	16.02k	0.57G

Table 2: Computational complexity and memory requirements of OPT-6.7B models compressed by LatentLLM.

Compression	10%			20%			30%			40%		
Dataset	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4	WT2	PTB	C4
OPT-1.3B (WT2: 14.6, PTB: 20.3, C4: 16.1)												
Plain SVD (Identity)	9428.1	10670.8	4865.4	16461.2	20589.0	11039.8	18105.3	17360.8	12565.2	22155.9	15820.3	16566.2
ASVD (Hessian)	23.8	40.6	24.9	63.0	173.7	52.8	825.8	927.9	385.0	4912.3	3086.3	2138.9
ASVD (ℓ_2-norm)	20.3	32.3	21.6	28.7	60.2	27.7	74.5	217.4	58.5	592.4	1072.0	336.7
ASVD (Cov)	29750.9	31499.1	18646.3	19716.9	21757.2	14967.2	21738.3	24300.2	16428.7	22776.5	23591.7	14922.1
ASVD (RootCov)	17.7	27.9	18.9	21.9	35.3	22.2	33.9	55.8	29.7	75.0	107.9	51.1
LatentLLM (RootCov)	***14.5**	**21.5**	**16.6**	**15.8**	**24.3**	**17.8**	**20.2**	**31.6**	**21.3**	**34.1**	**58.1**	**30.6**
Qwen3-1.7B (WT2: 16.7, PTB: 33.8, C4: 22.4)												
Plain SVD (Identity)	1.8e7	1.6e7	1.1e7	1.3e7	1.1e7	1.1e7	1.0e7	1.0e7	6.5e6	1.9e7	1.7e7	1.5e7
ASVD (Hessian)	1.1e5	6.8e5	3.4e5	5.2e6	8.0e6	4.6e6	4.4e6	6.7e6	4.0e6	3.0e6	1.6e7	3.2e6
ASVD (ℓ_2-norm)	72.5	138.4	102.8	1679.1	2719.6	1639.8	4842.6	1.2e4	3960.3	2.8e5	2.8e5	9.8e4
ASVD (Cov)	860.6	3378.6	338.3	1989.8	1.0e4	516.1	6645.8	2.4e4	906.1	1.2e4	4.6e4	1796.1
ASVD (RootCov)	37.5	63.2	43.9	66.3	114.8	62.5	147.8	287.6	100.0	387.2	1066.1	193.7
LatentLLM (RootCov)	**22.3**	**47.8**	**28.4**	**27.9**	**51.5**	**35.3**	**48.8**	**81.4**	**53.3**	**137.5**	**264.9**	**98.6**
Qwen3-8B (WT2: 9.7, PTB: 17.2, C4: 15.4)												
Plain SVD (Identity)	2.4e5	8.7e4	4.5e4	9.0e6	1.9e6	8.8e5	2.8e7	3.8e7	1.8e7	5.3e7	1.0e8	5.2e8
ASVD (Hessian)	33.6	78.3	40.7	90.8	573.7	115.2	1250.8	1.3e4	854.4	5324.6	3.1e4	4872.0
ASVD (ℓ_2-norm)	18.8	32.2	25.1	26.0	43.9	32.0	40.6	71.8	47.7	98.6	171.1	92.1
ASVD (Cov)	1.3e5	4.7e5	4.4e4	1.2e5	3.1e5	4.4e4	8.3e4	2.3e5	3.4e4	6.1e4	1.2e5	2.7e4
ASVD (RootCov)	16.7	25.0	21.8	26.0	32.6	26.3	49.3	60.5	38.6	119.2	136.9	71.1
LatentLLM (RootCov)	**11.8**	**21.2**	**17.9**	**14.2**	**23.1**	**19.9**	**22.4**	**29.5**	**24.8**	**53.9**	**68.5**	**40.8**

Table 3: Perplexity (↓) of OPT/Qwen3 models with different SVD compression methods for 10–40% size reduction. Asterisk "*" indicates the better performance than the original un-compressed LLM.

the MLP layer, SparseLLM (Bai et al. 2024b) provides an elegant way to approximate MLP layer. The key idea is to minimize the MLP loss in a decoupled manner by introducing auxiliary variables. Our LatentLLM exploits the same philosophy to compress MLP layers.

Consider a 2-layer MLP:

$$Z = W_{\rm u}X, \quad Z' = \sigma(Z), \quad Y = W_{\rm d}Z', \tag{17}$$

where $W_{\rm u} \in \mathbb{R}^{d_{\rm i}\times d}$ is the up projection matrix, $W_{\rm d} \in \mathbb{R}^{d\times d_{\rm i}}$ is the down projection matrix, and $d_{\rm i}$ is the intermediate size which is typically four-fold of hidden size: $d_{\rm i} = 4d$. The intermediate variables $Z, Z' \in \mathbb{R}^{d_{\rm i}\times l}$ are auxiliary factors to be optimized.

Specifically, we consider minimizing the decoupled loss:

$$\mathcal{L}_4 = \alpha\|W_{\rm u}X - Z\|^2 + \beta\|Z' - \sigma(Z)\|^2 + \gamma\|W_{\rm d}Z' - Y\|^2, \tag{18}$$

for auxiliary variables Z and Z', given calibration input X and output Y.

Following SparseLLM (Bai et al. 2024b), the optimal Z' can be obtained given the other parameters fixed:

$$Z' = \big(\gamma W_{\rm d}^\top W_{\rm d} + \beta I\big)^+ \big(\beta\sigma(Z) + \gamma W_{\rm d}^\top Y\big). \tag{19}$$

The optimal closed-form Z can be obtained for ReLU:

$$Z_- = W_{\rm u}X, \quad Z_+ = \frac{1}{\alpha+\beta}(\alpha Z_- + \beta Z'), \tag{20}$$

depending on Z's element-wise sign.

This approach can be used for low-rank approximation. Given Z, we can optimize low-rank matrix $\hat{W}_{\rm u} = B_{\rm u}A_{\rm u}$ by SVD of $ZX^+C^{\frac{1}{2}}$, where ZX^+ corresponds to the effective weight matrix to map X onto Z. Given Z', we approximate $\hat{W}_{\rm d} = B_{\rm d}A_{\rm d}$ by SVD of $YZ'^+C_{\rm d}^{\frac{1}{2}}$, given correlation $C_{\rm d} = Z'Z'^\top$. This alternating solution is iterated over a few rounds. For detail, see Appendix.

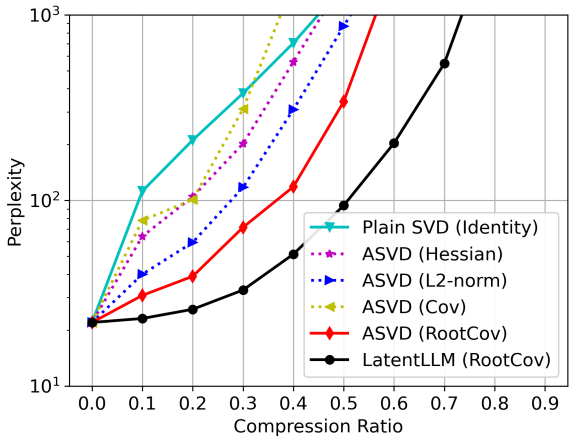

Figure 4: WT2 perplexity over compression ratio for OPT-350M model.

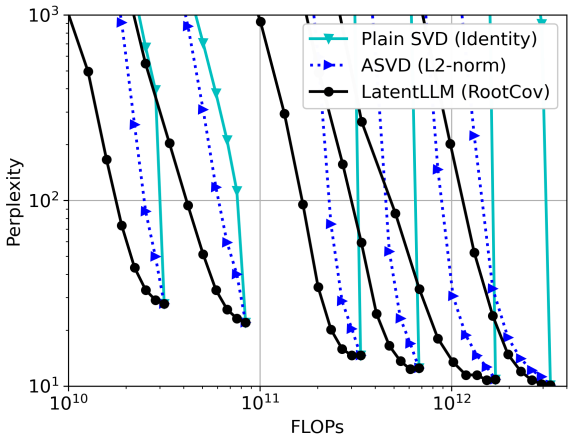

Figure 5: WT2 perplexity vs. FLOPs of six OPT models from 125M to 13B scales with varying compression ratios.

Experiments

Experiments Setup We conduct experiments for LLM and LMM benchmarks to evaluate the effectiveness of our method. Our experiments are based on the same setting of SparseLLM (Bai et al. 2024b) and their code base[1]. We implemented LatentLLM in PyTorch and used the HuggingFace transformers library for handling models and datasets. All experiments are conducted on NVIDIA A40 GPUs.

For LLM benchmark, we follow the same setup of (Frantar and Alistarh 2023) and use 64 samples of 2048-token segments, randomly chosen from the first shard of the C4 (Raffel et al. 2020) dataset. This dataset represents generic text data crawled from the internet and ensures our experiments are zero-shot as no task-specific data is seen during compression. We followed existing work (Sun et al. 2023) and compressed all linear layers in MLP and MHA in LLMs to the target compression ratio. For LMM benchmark, we use 64 samples, randomly chosen from the train split of the multi-modal question answering dataset. We consider two benchmarks: ScienceQA (Lu et al. 2022); and TextVQA (Singh et al. 2019).

For LLM experiments, we consider the OPT (Zhang et al. 2022) and Qwen3 (Yang et al. 2025) models as they provide a wide range of model sizes. We show results on different sizes of models to provide a broader picture for the performance of LatentLLM. We mainly focus on perplexity, which is known to be a stable metric for evaluating the accuracy of compression methods (Yao et al. 2022; Dettmers and Zettlemoyer 2023). We consider the test sets of raw-WikiText2 (WT2) (Merity et al. 2016) and the Penn Treebank (PTB) (Marcus et al. 1994) as well as a subset of the C4 validation data, all popular benchmarks in the LLM compression literature (Frantar and Alistarh 2023; Frantar et al. 2022; Sun et al. 2023).

For LMM experiments, we use LLaVA (Liu et al. 2023) and Qwen2.5-VL (Bai et al. 2025). We evaluate the capability of the multi-modal answer reasoning based on the ScienceQA dataset, which contains 21K vision-language multi-choice questions for three subjects: natural, social, and language science. Some fractions of questions have image and/or text contexts, and the problem levels range from grade 1 to 12. In addition, we also evaluate TextVQA, which makes LMMs to read and reason about text in images to answer visual reasoning questions for 28K images.

Computational Complexity When all linear modules are compressed with LatentLLM, the inference complexity is expected to be reduced with the compression ratio almost linearly. Nevertheless LLMs/VLMs have extra complexity other than linear affine transforms, the inference complexity is not precisely proportional to the compression factor. We show the complexity analysis in Table 2 for the compressed OPT-6.7B models, based on the `calflops` library. We assume the token length of 2048 at 4 batches. We found that the FLOPs, multiply-accumulation operations (MACs), and parameters are almost linearly reduced with the compression factor. The inference throughput on A40 GPU can be also monotonically increased by compressing LLMs. While we used `torch.compile(mode="max-autotune")`, it was not a perfectly linear speedup due to the sub-optimal GPU kernel. The reduction of KV cache memory is significant because the latent dimension has quadratic relation to the sparsity: $r(d' + d) - r^2 = \rho dd'$.

Compression over Model Size We first look into the compression capabilities of our LatentLLM across various model sizes in comparison to baseline methods. Some results are shown for a size reduction over 10–40% in Table 3. The perplexity results of the original un-compressed LLM models are reported next to the names of the models in the table.

We can see that the conventional plain SVD has a poor performance, and that ASVD with a proper pre-conditioning can significantly improve the perplexity. It was found that the diagonal Hessian is worse than the diagonal ℓ_2-norm, whereas covariance pre-conditioning can be terrible in low compression regimes for larger LLMs. In contrast, the superiority of root covariance is remarkable. In addition, the joint

[1]https://github.com/BaiTheBest/SparseLLM

Method	Compression	Subject			Context Modality			Grades		Avg
		NAT	SOC	LAN	TXT	IMG	NO	G1-6	G7-12	
Original un-compressed	0%	72.47	69.18	65.73	73.51	68.82	65.99	72.72	65.19	70.03
Plain SVD (Identity)	10%	5.33	1.35	0.27	5.77	6.69	0.00	3.30	2.97	3.18
ASVD (Hessian)	10%	17.23	24.97	3.18	18.43	29.55	2.16	17.40	11.27	15.21
ASVD (ℓ_2-norm)	10%	16.70	18.34	2.55	17.89	24.34	2.23	16.04	8.57	13.37
ASVD (Cov)	10%	41.21	27.22	37.91	41.30	35.15	38.33	38.62	35.27	37.42
ASVD (RootCov)	10%	64.08	56.13	57.36	64.03	60.98	57.35	62.70	57.02	60.67
LatentLLM (RootCov)	10%	**68.52**	**64.23**	**61.36**	**69.06**	**65.20**	**61.53**	**68.72**	**60.45**	**65.76**
Plain SVD (Identity)	30%	0.13	0.00	0.00	0.10	0.00	0.07	0.11	0.00	0.07
ASVD (Hessian)	30%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ASVD (ℓ_2-norm)	30%	0.09	0.00	0.00	0.10	0.10	0.00	0.04	0.07	0.05
ASVD (Cov)	30%	41.25	27.33	37.36	41.40	35.25	37.84	38.47	35.27	37.33
ASVD (RootCov)	30%	56.66	51.18	52.27	56.74	56.27	51.99	55.73	51.94	54.37
LatentLLM (RootCov)	30%	**64.03**	**56.24**	**55.27**	**64.47**	**61.77**	**55.40**	**62.78**	**55.37**	**60.13**
Plain SVD (Identity)	50%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ASVD (Hessian)	50%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ASVD (ℓ_2-norm)	50%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ASVD (Cov)	50%	40.94	26.88	36.91	41.20	35.10	37.28	38.18	34.74	36.95
ASVD (RootCov)	50%	52.58	45.11	46.00	52.93	50.07	45.99	51.28	45.75	49.30
LatentLLM (RootCov)	50%	**55.55**	**47.24**	**49.55**	**56.01**	**54.09**	**48.78**	**54.55**	**48.12**	**52.25**

Table 4: Accuracy in percent (↑) on ScienceQA dataset of LLaVA-7B model with different compression methods for 10%–50% size reduction. Question subjects: natural science (NAT); social science (SOC); language science (LAN). Context modality: text (TXT); image (IMG); or no context (NO). Grades: 1–6 (G1-6); 7–12 (G7-12).

Compression	10%	20%	30%	40%	50%
LLaVA-7B: Uncompressed Acc 61.32					
Plain SVD (identity)	2.36	0.48	0.35	0.34	0.36
ASVD (Hessian)	23.88	9.60	1.24	0.21	0.31
ASVD (ℓ_2-norm)	24.41	9.53	2.77	0.82	0.75
ASVD (Cov)	0.38	0.36	0.40	0.33	0.35
ASVD (RootCov)	52.51	49.91	45.53	38.47	27.36
LatentLLM (RootCov)	**60.06**	**57.65**	**52.63**	**46.90**	**35.94**
Qwen2.5-VL-7B-Instruct: Uncompressed Acc 82.11					
Plain SVD (identity)	0.02	0.47	0.32	0.05	0.11
ASVD (Hessian)	58.76	7.03	0.23	0.45	0.41
ASVD (ℓ_2-norm)	77.84	73.92	57.13	18.79	0.41
ASVD (Cov)	0.41	0.41	0.41	0.41	0.41
ASVD (RootCov)	79.46	74.76	66.31	51.80	34.91
LatentLLM (RootCov)	**80.85**	**79.30**	**73.90**	**62.11**	**42.53**

Table 5: Accuracy in percent (↑) on TextVQA dataset for compressed LLaVA-7B and Qwen2.5-VL-7B.

SVD used for LatentLLM offers an additional improvement consistently across different model sizes and families. Notice that our methods can also achieve slightly better performance than the original un-compressed LLMs for some cases. Figure 4 shows the plot at a wider range of compression ratios for OPT-350M model, and Figure 5 plots the performance of all six models in OPT family across FLOPs when varying the compression ratios.

Multi-Modal Reasoning Capability We show the accuracy of latent LLaVA models for ScienceQA multi-modal reasoning benchmark in Table 4. It is verified that our LatentLLM can significantly outperform other low-rank compression methods across diverse reasoning problems over different subjects/contexts/grades, approaching the performance of the original un-compressed LLaVA model. It is seen that ASVD without using proper pre-conditioning matrix degrades the performance quickly with higher compression ratios, while our LatentLLM keeps relatively higher performance across all cases. Another benchmark on TextVQA shown in Table 5 validates the clear superiority of our LatentLLM over baselines for both LLaVA and Qwen-VL models.

Discussion Our framework with optimal pre-conditioning and joint tensor distillations can be readily applied to pruning and quantization as well. See some results in Appendix. Further fine-tuning is expected to be able to compensate for the performance degradation by the latent reduction.

Summary

We introduced LatentLLM which jointly compresses multiple tensors through the use of high-order tensor-rank decomposition. We also provided some new perspectives for activation-aware compression when choosing the preconditioner and junction matrix. With a proper selection, we demonstrated that the model compression performance can be significantly improved. Our latent LMMs showed a significant advantage in multi-modal reasoning tasks compared to other baseline methods.

References

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

Bai, G.; Chai, Z.; Ling, C.; Wang, S.; Lu, J.; Zhang, N.; Shi, T.; Yu, Z.; Zhu, M.; Zhang, Y.; et al. 2024a. Beyond efficiency: A systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625*.

Bai, G.; Li, Y.; Ling, C.; Kim, K.; and Zhao, L. 2024b. SparseLLM: Towards global pruning of pre-trained language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Bai, S.; Chen, K.; Liu, X.; Wang, J.; Ge, W.; Song, S.; Dang, K.; Wang, P.; Wang, S.; Tang, J.; et al. 2025. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; et al. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*.

Chang, C.-C.; Lin, W.-C.; Lin, C.-Y.; Chen, C.-Y.; Hu, Y.-F.; Wang, P.-S.; Huang, N.-C.; Ceze, L.; Abdelfattah, M. S.; and Wu, K.-C. 2024. Palu: Compressing KV-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q. V.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948.

Denton, E. L.; Zaremba, W.; Bruna, J.; LeCun, Y.; and Fergus, R. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27.

Dettmers, T.; and Zettlemoyer, L. 2023. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, 7750–7774. PMLR.

Frantar, E.; and Alistarh, D. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, 10323–10337. PMLR.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2022. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Hassibi, B.; Stork, D. G.; and Wolff, G. J. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, 293–299. IEEE.

Hsieh, C.-Y.; Li, C.-L.; Yeh, C.-K.; Nakhost, H.; Fujii, Y.; Ratner, A.; Krishna, R.; Lee, C.-Y.; and Pfister, T. 2023. Distilling step-by-step! Outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*.

Hwang, I.; Park, H.; Lee, Y.; Yang, J.; and Maeng, S. 2024. PC-LoRA: Low-Rank Adaptation for Progressive Model Compression with Knowledge Distillation. *arXiv preprint arXiv:2406.09117*.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Hanna, E. B.; Bressand, F.; et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Katz, D. M.; Bommarito, M. J.; Gao, S.; and Arredondo, P. 2024. GPT-4 passes the bar exam. *Philosophical Transactions of the Royal Society A*, 382(2270): 20230254.

Kenton, J. D. M.-W. C.; and Toutanova, L. K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, 2. Minneapolis, Minnesota.

LeCun, Y.; Denker, J.; and Solla, S. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Ledoit, O.; and Wolf, M. 2004. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2): 365–411.

Li, Y.; Yu, Y.; Zhang, Q.; Liang, C.; He, P.; Chen, W.; and Zhao, T. 2023. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In *International Conference on Machine Learning*, 20336–20350. PMLR.

Liang, Y.; Long, J.; Shi, Z.; Song, Z.; and Zhou, Y. 2024. Beyond linear approximations: A novel pruning approach for attention matrix. *arXiv preprint arXiv:2410.11261*.

Lin, B.; Tang, Z.; Ye, Y.; Cui, J.; Zhu, B.; Jin, P.; Zhang, J.; Ning, M.; and Yuan, L. 2024a. MoE-LlaVa: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.-M.; Wang, W.-C.; Xiao, G.; Dang, X.; Gan, C.; and Han, S. 2024b. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. *Proceedings of Machine Learning and Systems*, 6: 87–100.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024. DeepSeek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023. Visual Instruction Tuning.

Lu, P.; Mishra, S.; Xia, T.; Qiu, L.; Chang, K.-W.; Zhu, S.-C.; Tafjord, O.; Clark, P.; and Kalyan, A. 2022. Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*.

Marcus, M.; Kim, G.; Marcinkiewicz, M. A.; MacIntyre, R.; Bies, A.; Ferguson, M.; Katz, K.; and Schasberger, B. 1994. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

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

Radford, A. 2018. Improving language understanding by generative pre-training. *Preprint*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Saha, R.; Sagan, N.; Srivastava, V.; Goldsmith, A.; and Pilanci, M. 2024. Compressing large language models using low rank and low precision decomposition. *Advances in Neural Information Processing Systems*, 37: 88981–89018.

Sainath, T. N.; Kingsbury, B.; Sindhwani, V.; Arisoy, E.; and Ramabhadran, B. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6655–6659. IEEE.

Saxena, U.; Saha, G.; Choudhary, S.; and Roy, K. 2024. Eigen attention: Attention in low-rank space for KV cache compression. *arXiv preprint arXiv:2408.05646*.

Schotthöfer, S.; Zangrando, E.; Kusch, J.; Ceruti, G.; and Tudisco, F. 2022. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. *Advances in Neural Information Processing Systems*, 35: 20051–20063.

Schwartz, R.; Dodge, J.; Smith, N. A.; and Etzioni, O. 2020. Green AI. *Communications of the ACM*, 63(12): 54–63.

Singh, A.; Natarjan, V.; Shah, M.; Jiang, Y.; Chen, X.; Parikh, D.; and Rohrbach, M. 2019. Towards VQA Models That Can Read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8317–8326.

Su, J.; Ahmed, M.; Lu, Y.; Pan, S.; Bo, W.; and Liu, Y. 2024. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568: 127063.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

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

Wang, C.; Wang, Z.; Xu, X.; Tang, Y.; Zhou, J.; and Lu, J. 2024a. Q-VLM: Post-training Quantization for Large Vision-Language Models. *arXiv preprint arXiv:2410.08119*.

Wang, X.; Zheng, Y.; Wan, Z.; and Zhang, M. 2024b. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Xu, C.; and McAuley, J. 2023. A survey on model compression and acceleration for pretrained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 10566–10575.

Yang, A.; Li, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Gao, C.; Huang, C.; Lv, C.; et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang, Y.; Li, X.; Zhou, Z.; Song, S. L.; Wu, J.; Nie, L.; and Ghanem, B. 2024. CorDA: Context-Oriented Decomposition Adaptation of Large Language Models. *arXiv preprint arXiv:2406.05223*.

Yao, Z.; Yazdani Aminabadi, R.; Zhang, M.; Wu, X.; Li, C.; and He, Y. 2022. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35: 27168–27183.

Yuan, Z.; Shang, Y.; Song, Y.; Wu, Q.; Yan, Y.; and Sun, G. 2023. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*.

Yuan, Z.; Shang, Y.; Zhou, Y.; Dong, Z.; Zhou, Z.; Xue, C.; Wu, B.; Li, Z.; Gu, Q.; Lee, Y. J.; et al. 2024. LLM inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhu, X.; Li, J.; Liu, Y.; Ma, C.; and Wang, W. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12: 1556–1577.