LSVD: Loss-Aware Low-Rank Approximation for Efficient Low-Precision Vision-Language Models

Haiyu Wang[1] **Yutong Wang**[1*] **Leshu Li**[2*] **Yihui Ren**[3] **Sai Qian Zhang**[1,2]
[1]Tandon School of Engineering, New York University
[2]Courant Institute of Mathematical Sciences, New York University
[3]Brookhaven National Laboratory
{hw3689, yw6594, ll5914}@nyu.edu yren@bnl.gov sai.zhang@nyu.edu

Abstract

Vision-language models (VLMs) deliver strong multimodal reasoning capabilities, but their large computational cost and high parameter counts make deployment challenging on resource-constrained devices. Low-rank decomposition has emerged as a promising compression technique, yet existing methods often optimize local matrix reconstruction error, rely on uniform or heuristic rank allocation, and focus mainly on attention projections while leaving feed-forward networks underexplored. In this paper, we propose *LSVD*, a loss-aware low-rank approximation framework for efficient low-precision VLMs. LSVD derives a curvature-weighted SVD objective from a second-order approximation of the model loss and uses Kronecker-factored Fisher information to guide decomposition toward downstream performance rather than reconstruction alone. We further introduce a loss-aware cross-layer rank allocation strategy based on calibration gradients, enabling more effective parameter budgeting across layers. Finally, we extend low-rank compression to FFN layers through a hybrid scheme that combines SVD with quantization. Experimental results demonstrate that LSVD achieves more than 2.3× decoding speedup over existing methods while preserving strong accuracy under low-precision inference. Code is available at `https://github.com/SAI-Lab-NYU/LSVD`.

1 Introduction

Vision–language models (VLMs) have become an important research direction in modern AI because they bridge visual perception and language understanding. By reasoning over both image and text inputs, VLMs support a wide range of multimodal tasks, such as image captioning [9, 13, 19, 23, 24, 61], visual question answering [3, 4, 7, 49, 60, 65], and multimodal semantic retrieval [22, 43, 44, 66]. Despite their strong performance, these models impose substantial implementation costs. Processing high-dimensional visual features together with textual context leads to intensive computation, while autoregressive

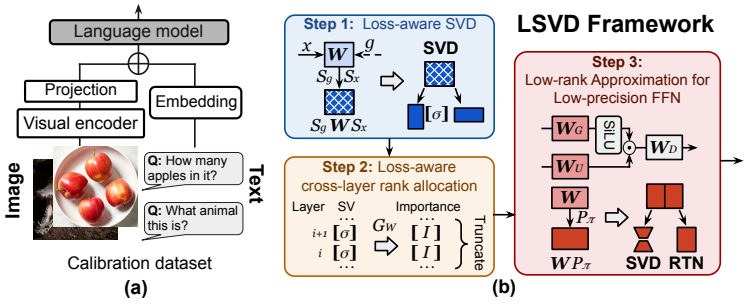

Figure 1: (a) A typical VLM. (b) Overview of LSVD Framework.

*Authors contributed equally; the order of authorship was assigned randomly.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

decoding repeatedly accesses model states and generated tokens, creating severe memory-bandwidth pressure during inference and therefore high inference latency. Moreover, the large size of VLMs leads to substantial storage overhead, limiting their deployment on storage-constrained devices.

While quantization and pruning have been widely explored to reduce the computation and storage costs of large models [2, 29, 45, 47, 56], low-rank decomposition has recently attracted increasing attention as another promising compression approach. By factorizing the query (Q), key (K), and value (V) projection matrices in self-attention blocks into low-rank components, prior studies have demonstrated notable reductions in parameter count and computational cost [6, 25, 28, 50, 52–55, 62, 68]. However, several important directions remain underexplored. First, although individual optimization techniques have shown promising gains, it remains unclear whether combining them can yield further improvements. Second, rank allocation across VLM layers has received limited attention, although different layers may exhibit distinct sensitivity to compression and therefore require non-uniform rank assignments. Finally, most existing efforts focus primarily on attention projections, while the feed-forward network (FFN) layers, which account for a large fraction of model parameters and storage cost, remain a critical but less explored target for low-rank compression.

To address these limitations, we systematically revisit low-rank compression for VLMs along these underexplored directions, and propose *Loss-Aware Low-Rank Approximation for Efficient Low-Precision Vision-Language Models* (LSVD). Our contributions are summarized as follows:

- LSVD derives a loss-aware SVD objective from a second-order approximation of the model loss, using Kronecker-factored Fisher information to align low-rank decomposition with downstream VLM performance rather than only minimizing reconstruction error.
- LSVD provides an analysis of why singular values obtained from loss-aware SVD are not directly comparable across layers and heads, and introduces a calibration-gradient-based importance score for more reliable global rank allocation.
- LSVD extends low-rank compression to FFN layers through a hybrid SVD–quantization scheme that assigns each channel to the compression branch it can better tolerate.
- We develop Triton kernels for the hybrid low-rank FFN and low-precision SA layers. Experimental results demonstrate that LSVD achieves average layer-wise decoding speedups of $4.7\times$ over Flash Decoding and $2.3\times$ over WSVD, while preserving accuracy under low-precision inference.

2 Related Work

Vision-Language Model VLMs extend LLMs to multimodal reasoning by incorporating visual inputs into language-conditioned generation [5, 10, 16, 23, 24, 33, 51, 58, 59]. By aligning visual features with the LLM semantic space, they support image captioning, visual question answering, grounding, document understanding, and video comprehension. Modern VLMs typically consist of a vision encoder, a projection module, and an autoregressive LLM, as seen in BLIP/InstructBLIP [23, 24], LLaVA [33], PaLI-Gemma [5], Qwen-VL [51], and SmolVLM [37]. Despite their strong capabilities, VLMs are expensive to deploy because they inherit LLM computation and memory costs while adding vision encoders and long visual-token sequences. This challenge motivates compact VLMs [37, 67, 69] and compression techniques such as low-rank decomposition and quantization.

Low-Rank Approximation for Large Models Low-rank approximation is a classical approach for reducing the storage and computational cost of large matrices. A standard approach is singular value decomposition (SVD) [20], which factorizes a weight matrix $W \in \mathbb{R}^{m\times n}$ as $W = U\Sigma V^T$, where U and V contain the left and right singular vectors, and Σ is a diagonal matrix of singular values. Retaining only the largest r singular components gives the rank-r approximation: $W \approx U_r\Sigma_r V_r^T$, which can be implemented as two smaller matrices, $W \approx AB$ with $A = U_r\Sigma_r^{1/2} \in \mathbb{R}^{m\times r}$ and $B = \Sigma_r^{1/2}V_r^T \in \mathbb{R}^{r\times n}$. When $r \ll \min(m, n)$, this decomposition reduces both parameter count and arithmetic cost. SVD-based compression has been extensively studied for large language models and vision–language models [6, 18, 25, 28, 41, 52, 53, 55, 68]. While early work directly applied vanilla SVD to pretrained weights [41], later methods improve robustness by incorporating calibration or sensitivity information: FWSVD [18] uses Fisher information, ASVD [68] accounts for activation

outliers, and SVD-LLM [53] and Dobi-SVD [52] reduce layer-wise truncation error. Recent methods further refine rank allocation [28, 55] or combine low-rank decomposition with quantization [25, 50].

Beyond weight compression, low-rank approximation has also been applied to inference-time memory reduction and acceleration, particularly for KV cache compression [6, 50, 64]. By decomposing the key and value projection matrices, for example $W_K \approx A_K B_K$ and $W_V \approx A_V B_V$, the cache can keep lower-dimensional latent representations $C_K = xA_K$ and $C_V = xA_V$, where x denotes the input. The full keys and values are then reconstructed only when needed as $K = C_K B_K$ and $V = C_V B_V$, avoiding storage of the full-dimensional KV states. This reduces KV cache footprint and memory traffic during autoregressive decoding, which can improve decoding throughput, especially in long-context generation and multimodal models with large token budgets.

Quantization for Large Models Quantization reduces the memory and inference cost of large models by mapping weights and activations to low-bit formats, with post-training quantization (PTQ) being especially attractive because it avoids retraining. Representative LLM PTQ methods include GPTQ [15], AWQ [30], and SmoothQuant [57], which respectively use second-order error compensation, activation-aware channel protection, and channel-wise scaling to improve robustness. Recent methods further improve low-bit quantization through learned parameters, rotations, and optimized kernels [56, 63], such as OmniQuant [42], QuaRot [2], SpinQuant [35], and QServe [31]. For VLMs, quantization is more challenging due to modality-dependent activation distributions and different sensitivities between visual and textual tokens; Q-VLM [47] and MBQ [26] address these issues through cross-layer dependency modeling and modality-balanced calibration.

Kronecker-Factored Approximate Curvature Second-order curvature provides a principled way to measure parameter sensitivity, but the full Hessian or Fisher matrix is prohibitively expensive to compute and store for large models. For a linear layer with $d_{\text{in}} d_{\text{out}}$ parameters, the corresponding Fisher block contains $O(d_{\text{in}}^2 d_{\text{out}}^2)$ entries. K-FAC [39] makes second-order curvature tractable by approximating this block with a Kronecker product of activation and gradient covariance factors, reducing the curvature representation to two much smaller covariance matrices. For a linear layer $y = xW$ with input activation x and output gradient g, K-FAC approximates the Fisher information: $F_W \approx \mathbb{E}[g^\top g] \otimes \mathbb{E}[x^\top x]$, thereby retaining structured curvature information at substantially lower cost. Originally proposed for natural-gradient optimization [39], Kronecker-factored curvature has also been used for curvature-aware compression. EigenDamage [48] performs structured pruning in the Kronecker-factored eigenbasis, and the LLM Surgeon [46] scales such approximations to large language models for pruning and weight-update compensation. Recent and concurrent works, including GFWSVD [8] and OBD-LLM [27], also explore two-sided curvature-aware SVD. In this work, we leverage K-FAC to derive a closed-form loss-aware SVD that minimizes the estimated compression-induced model-loss increase rather than merely local reconstruction error. We further introduce empirical-Fisher-based rank allocation to address cross-layer and cross-head scale inconsistencies, and integrate quantization-aware whitening and hybrid FFN compression for efficient low-precision VLM inference.

3 Method

3.1 Preliminary

Conventional SVD-based low-rank approximation is usually formulated as a local matrix reconstruction problem. Given a weight matrix W, truncated SVD solves $\arg\min_{\widehat{W}:\text{rank}(\widehat{W})\le r} \|W - \widehat{W}\|_F^2$, where $\widehat{W}$ denotes the compressed weight with low rank. Since this objective ignores the input distribution, activation-aware methods [52, 53, 68] instead minimize layer-wise output error on calibration activations, e.g., $\|X(W - \widehat{W})\|_F^2$. Other approaches further incorporate gradients or Fisher information to guide truncation and rank allocation [18, 50, 55]. Nevertheless, the decomposition itself remains largely driven by local reconstruction criteria rather than the end-to-end training objective. In the following sections, we therefore derive a loss-aware low-rank objective from a local curvature approximation of the model loss as illustrated in Fig. 2. Following K-FAC [39] and prior curvature-aware compression methods [46, 48], we approximate the loss increase caused by compression using a second-order Kronecker-factored Fisher surrogate. The derivation from a local quadratic loss approximation is provided in Appendix A.1.

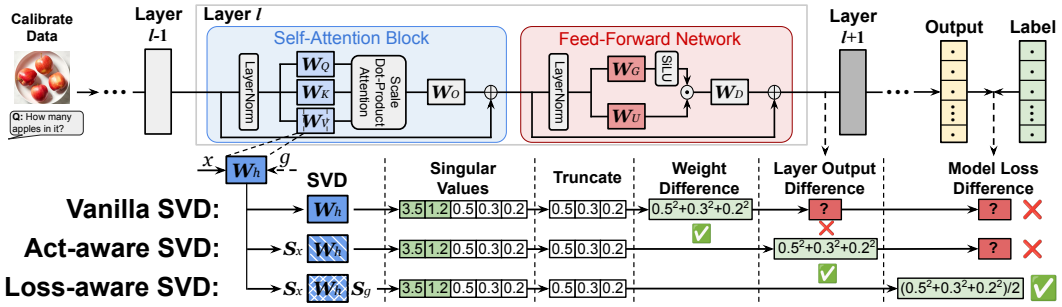

Figure 2: Vanilla SVD minimizes weight reconstruction error. Activation-aware SVD such as SVD-LLM [53] reduces layer-wise output error with input activation statistics. Loss-aware SVD performs SVD in a curvature-weighted weight space to minimize the compression-induced model loss.

Assumption 3.1 (K-FAC loss surrogate for weight compression)**.** Consider a linear layer $y = xW$, and let $\Delta W = \widehat{W} - W$ denote the perturbation introduced by replacing the pretrained weight W with its compressed approximation $\widehat{W}$. The compression-induced loss increase is estimated by:

$$\Delta\mathcal{L} \approx \frac{1}{2}\mathrm{vec}(\Delta W)^\top (G \otimes X)\mathrm{vec}(\Delta W) \tag{1}$$

where x is the input activation, $g = \nabla_y \ell$ is the output gradient of the per-sample loss, and $X = \mathbb{E}[x^\top x]$ and $G = \mathbb{E}[g^\top g]$ are the two K-FAC factors estimated on the calibration set.

3.2 Loss-Aware Singular Value Decomposition

Building on Assumption 3.1, we derive the low-rank approximation of a weight matrix that minimizes the estimated compression-induced loss increase. Consider a candidate linear layer with weight $W \in \mathbb{R}^{m\times n}$ and its rank-r approximation $\widehat{W}_r$. The associated activation and gradient covariances are factorized as $X = S_x^\top S_x$ and $G = S_g S_g^\top$ with Cholesky factorization [17], where S_x and S_g are assumed nonsingular.

Theorem 3.2 (Optimal loss-aware rank-r weight approximation)**.** *Under Assumption 3.1, define the curvature-weighted weight matrix $\widetilde{W} = S_x W S_g$ and let $\widetilde{W} = U\Sigma V^\top$ be its SVD, where $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_q)$ and $q = \min\{m, n\}$. The rank-r approximation of W that minimizes the estimated compression-induced loss increase, and the corresponding truncation loss, are:*

$$\widehat{W}_r^\star = S_x^{-1} U_r \Sigma_r V_r^\top S_g^{-1}, \qquad \Delta\mathcal{L}_r \approx \frac{1}{2}\sum_{i=r+1}^{q} \sigma_i^2 \tag{2}$$

Theorem 3.2 directly leads to an implementable procedure. Given a calibration set, we collect the input activations x and output gradients g for each candidate linear layer, and estimate $X = \mathbb{E}[x^\top x]$ and $G = \mathbb{E}[g^\top g]$. Then we compute factors S_x and S_g using Cholesky factorization [17], such that $X = S_x^\top S_x$ and $G = S_g S_g^\top$. Next, we form the transformed weight $\widetilde{W} = S_x W S_g$, perform truncated SVD to obtain $\widetilde{W}_r^\star = U_r \Sigma_r V_r^\top$, and map it back as:

$$\widehat{W}_r^\star = S_x^{-1} U_r \Sigma_r V_r^\top S_g^{-1} = A_r B_r, \qquad A_r = S_x^{-1} U_r \Sigma_r^{1/2}, \quad B_r = \Sigma_r^{1/2} V_r^\top S_g^{-1} \tag{3}$$

The corresponding truncation loss is: $\Delta\mathcal{L}_r \approx \frac{1}{2}\left\|S_x(W - \widehat{W}_r^\star)S_g\right\|_F^2 = \frac{1}{2}\sum_{i=r+1}^{q}\sigma_i^2$. At inference, we do not materialize $\widehat{W}_r^\star \in \mathbb{R}^{m\times n}$; instead, we replace xW with xA_rB_r, where $A_r \in \mathbb{R}^{m\times r}$ and $B_r \in \mathbb{R}^{r\times n}$. This reduces the parameter count and dominant multiply-accumulation (MAC) cost from mn to $r(m+n)$, giving the compression ratio $\rho_1 = \frac{r(m+n)}{mn}$. Thus, when $\rho_1 < 1$, or equivalently $r < mn/(m+n)$, the low-rank form reduces parameter count and computational cost.

3.3 Loss-Aware Cross-Layer Rank Allocation

Theorem 3.2 shows that the squared singular values of the curvature-weighted matrix estimate the loss increase caused by truncating the corresponding singular components. For layer ℓ, let $\sigma_{\ell,i}$ denote the i-th singular value after loss-aware SVD. This gives a natural intuition for rank allocation: singular

components with larger $\sigma_{\ell,i}^2$ should be preserved first, and layers or heads with more large singular values should receive more ranks. However, using these K-FAC-based singular values for global rank allocation assumes that values computed from different layers or attention heads are directly comparable. In other words, a larger $\sigma_{\ell,i}$ in one layer ℓ should correspond to a larger loss increase than a smaller $\sigma_{\ell',j}$ in another layer ℓ'. This comparability is not guaranteed in practice. K-FAC estimates curvature separately for each layer or attention head, so the loss estimate in Assumption 3.1 may differ from the empirical-Fisher estimate by layer- or head-dependent scaling factors. In this case, the same value of $\sigma_{\ell,i}$ may correspond to different loss increases in different layers or heads. Although $\sigma_{\ell,i}$ is meaningful for local truncation within the same layer or head, it may be unreliable for ranking components globally across layers and heads. To examine this issue, we use the trace ratio η_ℓ between the empirical-Fisher block and its K-FAC approximation for layer ℓ to indicate K-FAC's estimation bias:

$$\eta_\ell \triangleq \frac{\mathrm{Tr}(F_\ell)}{\mathrm{Tr}(K_\ell)} = \frac{\mathrm{Tr}(F_\ell)}{\mathrm{Tr}(G_\ell \otimes X_\ell)} = \frac{\mathrm{Tr}(F_\ell)}{\mathrm{Tr}(G_\ell)\,\mathrm{Tr}(X_\ell)} \tag{4}$$

where F_ℓ is the empirical-Fisher block of layer ℓ, $K_\ell = G_\ell \otimes X_\ell$ is its K-FAC approximation, and X_ℓ and G_ℓ are the activation-side and gradient-side K-FAC factors of layer ℓ. We provide a formal derivation of this trace ratio and its connection to the K-FAC approximation error in Appendix A.3. $\eta_\ell > 1$ means that K-FAC underestimates the average curvature measured by the empirical Fisher for layer ℓ, while $\eta_\ell < 1$ means that it overestimates it. Large variation in η_ℓ indicates that K-FAC-based singular values are not consistently scaled across layers or heads. We use this ratio only to expose this mismatch, rather than to rescale individual singular components.

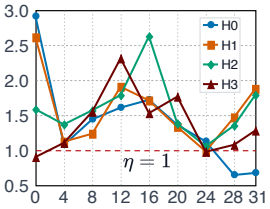

Figure 3: Values of $\eta_{\ell,h}$ across 32 layers (4 heads illustrated).

To empirically examine and quantify this mismatch, we analyze LLaVA-v1.5 7B [33] by computing this ratio for each Q-projection head. For the h-th Q head in layer ℓ, we evaluate $\eta_{\ell,h}$ on the corresponding head-wise weight block. As shown in Fig. 3, $\eta_{\ell,h}$ varies substantially across both layers and heads. Some heads remain close to $\eta = 1$, while others exceed 2 and approach 3 in several layers. Heads within the same layer also show different degrees of mismatch. These results show that K-FAC-based singular values are not on a consistent scale across layers and heads, and therefore may not be directly compared for global rank allocation. This motivates our use of an empirical-Fisher importance score for more accurate component ranking.

Empirical-Fisher Singular-Value Importance. Instead of globally ranking singular components by the K-FAC estimate $\sigma_{\ell,i}^2$, we extend the Importance Score [55] to the loss-aware SVD setting. Specifically, we use calibration gradients to directly estimate the empirical-Fisher-based loss sensitivity of each singular component, providing a better calibrated and more loss-aware criterion for cross-layer rank allocation. Recall that $\widetilde{W} = S_x W S_g = U\Sigma V^\top$. For the i-th singular value of layer ℓ, we define its importance score as:

$$\widehat{I}_{\ell,i} = \frac{1}{N}\sum_{n=1}^{N} \sigma_{\ell,i}^2 \left[U_\ell^\top S_{x,\ell}^{-\top} G_{W_\ell}^{(n)} S_{g,\ell}^{-\top} V_\ell\right]_{i,i}^2 \tag{5}$$

where $G_{W_\ell}^{(n)} = \nabla_{W_\ell}\ell^{(n)}$ denotes the weight gradient of layer ℓ on the n-th calibration sample. This score measures how much each singular component contributes to the empirical-Fisher quadratic loss approximation, rather than relying on the K-FAC surrogate used to compute the SVD basis. It combines the singular value magnitude, captured by $\sigma_{\ell,i}^2$, with its alignment to the calibration gradients after mapping the basis back to the original parameter space. We provide the derivation in Appendix A.4.

Global Rank Allocation. We allocate ranks by selecting singular components under a target parameter budget B. For the ℓ-th candidate matrix $W_\ell \in \mathbb{R}^{m_\ell \times n_\ell}$, where m_ℓ and n_ℓ are its input and output dimensions, preserving one singular component will cost additional $\kappa_\ell = m_\ell + n_\ell$ parameters. Let $\mathcal{S}$ denote the set of selected components, where $(\ell, i) \in \mathcal{S}$ means that the i-th singular component of W_ℓ is retained. The global selection problem is:

$$\mathcal{S}^\star = \arg\max_{\mathcal{S}} \sum_{(\ell,i)\in\mathcal{S}} \widehat{I}_{\ell,i} \quad \text{s.t.} \quad \sum_{(\ell,i)\in\mathcal{S}} \kappa_\ell \le B \tag{6}$$

Algorithm 1 SVD-aware Hybrid FFN Compression

Require: FFN weights $W_G, W_U \in R^{d_{\text{in}} \times d_{\text{ffn}}}, W_D \in R^{d_{\text{ffn}} \times d_{\text{out}}}$; SVD channel ratio γ; SVD retention ratio ρ_{ffn}
Ensure: Quantized hybrid low-rank–dense FFN

$\bar{W}_G, \bar{W}_U \leftarrow \text{LowRank}(W_G; \rho_{\text{ffn}}), \text{LowRank}(W_U; \rho_{\text{ffn}})$ ▷ temporary SVD for channel scoring
for $j = 1, \ldots, d_{\text{ffn}}$ **do**
  $e_j \leftarrow \|W_G[:, j] - \bar{W}_G[:, j]\|_2^2 + \|W_U[:, j] - \bar{W}_U[:, j]\|_2^2$ ▷ SVD error for each hidden channel
end for
$\mathcal{I}_S \leftarrow$ indices of the $\lfloor \gamma d_{\text{ffn}} \rfloor$ smallest scores in $\{e_j\}$ ▷ SVD-friendly channels
$\mathcal{I}_D \leftarrow \{1, \ldots, d_{\text{ffn}}\} \setminus \mathcal{I}_S$ ▷ other channels kept dense
$\pi \leftarrow \text{Concat}(\mathcal{I}_S, \mathcal{I}_D)$ ▷ new channel order with SVD channels first
$P_\pi \leftarrow$ permutation matrix induced by this order
$W'_G, W'_U, W'_D \leftarrow W_G P_\pi,\ W_U P_\pi,\ P_\pi^\top W_D$ ▷ preserve FFN output
$[W'_{G,S}, W'_{G,D}] \leftarrow \text{Split}(W'_G, |\mathcal{I}_S|), \quad [W'_{U,S}, W'_{U,D}] \leftarrow \text{Split}(W'_U, |\mathcal{I}_S|)$ ▷ separate SVD and dense blocks
$(A_G, B_G), (A_U, B_U) \leftarrow \text{LowRank}(W'_{G,S}; \rho_{\text{ffn}}),\ \text{LowRank}(W'_{U,S}; \rho_{\text{ffn}})$ ▷ SVD selected blocks
for $T \in \{G, U\}$ **do**
  $R_T \leftarrow \text{Rotation}(\text{rank}(A_T))$ ▷ orthogonal rotation: $R_T^\top R_T = I$
  $(A_T, B_T) \leftarrow (A_T R_T, R_T^\top B_T)$ ▷ preserve product while smoothing factors
  $(A_{T,q}, B_{T,q}) \leftarrow \text{RTN}(A_T, B_T)$
end for
$W'_{G,D,q}, W'_{U,D,q}, W'_{D,q} \leftarrow \text{RTN}(W'_{G,D}), \text{RTN}(W'_{U,D}), \text{RTN}(W'_D)$ ▷ quantize dense branch and down projection
return $\{A_{G,q}, B_{G,q}, A_{U,q}, B_{U,q}, W'_{G,D,q}, W'_{U,D,q}, W'_{D,q}\}$

The rank assigned to candidate layer ℓ is $r_\ell = |\{i : (\ell, i) \in \mathcal{S}^\star\}|$. If all candidate layer matrices have the same shape, the problem reduces to globally keeping the components with the largest $\widehat{I}_{\ell,i}$. When candidate matrices have different shapes, retaining a singular component can incur different parameter costs. We therefore rank components by importance per parameter, $\widehat{I}_{\ell,i}/\kappa_\ell$, and greedily retain the highest-ranked components until the parameter budget B is reached.

In LSVD, we treat the per-head QKV projections in self-attention layers as candidate matrices. We apply the loss-aware SVD from Sec. 3.2 to each per-head projection and use the proposed rank allocation strategy to assign ranks across layers and heads. This compresses QKV weights and reduces projection cost. For K/V projections, the per-head low-rank form also lowers KV-cache memory traffic during autoregressive decoding. Overall, LSVD accelerates inference while preserving accuracy by retaining the most important singular components.

3.4 Quantization-Aware Whitening for SVD

To further reduce computational cost, we apply linear quantization to both activations and low-rank weights. The loss-aware SVD objective in Sec. 3.2 uses the input-side K-FAC factor $X = \mathbb{E}[x^\top x]$, which corresponds to a full-precision linear layer $y = xW$. In deployment, however, activation quantization replaces the input with $\tilde{x} = Q(x)$, where $Q(\cdot)$ is the activation quantization function with its scale and clipping range fixed after calibration. As a result, substituting W with its low-rank approximation $\widehat{W}$ induces the quantized-output perturbation $\Delta y_q = \tilde{x}(\widehat{W} - W)$, rather than $x(\widehat{W} - W)$. To align the whitening objective with the deployed low-precision layer, we adopt *Quantization-Aware Whitening* (QAW) by estimating the input-side K-FAC factors using quantized calibration activations: $X_q = \mathbb{E}[\tilde{x}^\top \tilde{x}]$. We then factorize it as $X_q = S_{x,q}^\top S_{x,q}$ and construct the transformed weight $\widetilde{W}_q = S_{x,q} W S_g$. The remaining steps follow Sec. 3.2: we perform truncated SVD on $\widetilde{W}_q$ and map the resulting low-rank factors back to the original weight space. After obtaining the low-rank factors, following prior work [55], we insert a rotation matrix [2] between them before weight quantization, which preserves the low-rank product while smoothing outliers and improving quantization robustness. When activation quantization is disabled, Q reduces to the identity map, so $X_q = X$, and this formulation recovers the standard loss-aware SVD in Sec. 3.2.

3.5 Low-Rank Approximation for Low-Precision Feed-Forward Networks

While Sections 3.2 and 3.3 focus on optimizing self-attention (SA) layers, this section extends the discussion to gated feed-forward networks (FFNs). FFN layers account for a large fraction of transformer parameters, but applying the same low-rank compression strategy uniformly to all FFN hidden channels can cause substantial accuracy degradation. Empirically, we find that decomposing the entire FFN weight matrices with SVD often introduces excessive accuracy degradation relative to the original model. To address this, we apply low-rank SVD only to selected FFN submatrices and introduce a SVD-aware procedure to identify SVD-friendly hidden channels. Afterward, we further quantize the resulting matrices to reduce the computational cost and parameter size of the FFN layers.

Algorithm 1 summarizes the proposed hybrid FFN compression procedure, which consists of four steps. First, we estimate the SVD-friendliness of each FFN hidden channel by applying a temporary truncated SVD to the gate and up projections, and select the channels with the lowest reconstruction error for final SVD. Second, we move the selected channels into a contiguous block by applying the same permutation to W_G and W_U, and the inverse permutation to W_D. This preserves the SwiGLU FFN output because each gate/up hidden channel remains aligned with its corresponding row in the down projection. Third, we apply SVD only to the selected gate/up blocks, while keeping the remaining channels dense. Finally, we insert an orthogonal rotation between each pair of SVD factors and apply round-to-nearest (RTN) quantization to all resulting factors and dense blocks.

3.6 System Implementation

To realize the latency benefits of LSVD's low-rank and low-precision design in Sec. 3.5, we implement the hybrid low-rank FFN using fused Triton kernels, as shown in Fig. 4. This reduces the number of kernel launches and intermediate memory traffic across gate/up projections, SwiGLU, and quantization/dequantization operations.

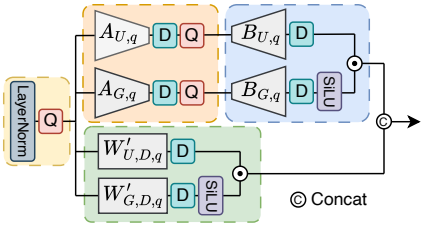

Figure 4: Fused kernels for the hybrid low-rank–dense FFN. $\odot$ denotes the element-wise product in SwiGLU, and Q/D denote quantization/dequantization.

For the SVD-factorized branch in Sec. 3.5, where $W'_{G,S} \approx A_{G,q}B_{G,q}$ and $W'_{U,S} \approx A_{U,q}B_{U,q}$, we first fuse the input-side projections $XA_{G,q}$ and $XA_{U,q}$ into a single kernel to generate the latent activations. The second-stage projections with $B_{G,q}$ and $B_{U,q}$ are fused and then applied to the corresponding latent activations. For the dense branch, we likewise fuse the computation of $XW'_{G,D,q}$ and $XW'_{U,D,q}$ into one kernel. Both branches use the same channel partition defined in Sec. 3.5. Since the gate and up projections share aligned channel assignments, SwiGLU can be applied independently within each branch. We therefore fuse SiLU and element-wise multiplication after the projections, and then concatenate the SVD-factorized and dense outputs to obtain the final FFN activation. For low-precision inference, we further fuse the per-token maximum reduction, quantization, and dequantization steps into the projection and nonlinear LayerNorm kernels, as shown by the Q/D nodes in Fig. 4. This avoids extra passes over intermediate activations and reduces quantization-related overhead. Overall, the fused implementation preserves the original FFN computation while reducing memory movement, kernel launches, and quantization overhead. For SA layers, we extend the fused attention kernel [50] to the low-precision setting to further improve inference efficiency.

4 Evaluation

We conduct experiments on five representative vision–language models: LLaVA-v1.5 7B [33], LLaVA-Next 7B, LLaVA-Next 13B, Qwen2-VL 7B [51], and SmolVLM 2B [37]. For calibration, we use 256 samples randomly drawn from the ScienceQA training split [36], following the procedures described in Sec. 3.2, 3.3 and 3.4. Evaluation is conducted on three widely used benchmarks, ScienceQA-IMG [36], SEED-Bench-IMG [21] and MMBench [34], consistent with prior studies on VLMs such as LLaVA, using the VLMEvalKit [12] tool. For comparison, LSVD is benchmarked against several baselines, including SVD-based approaches (SVD-LLM [53], QSVD [55], WSVD [50]) and quantization-based techniques (DuQuant [29], QVLM [47]). For SVD-LLM, QSVD and WSVD, we follow their official implementations and apply SVD independently to the

Table 1: Accuracy of different methods under FP16 (detailed results in Appendix A.5.1).

Acc.	Method	ScienceQA-IMG ↑					SEED-Bench ↑					Avg. ↑
		$\rho_1:90\%$	$\rho_1:80\%$	$\rho_1:70\%$	$\rho_1:60\%$	$\rho_1:50\%$	$\rho_1:90\%$	$\rho_1:80\%$	$\rho_1:70\%$	$\rho_1:60\%$	$\rho_1:50\%$	
SmolVLM 2B	SVD-LLM	40.06%	17.20%	3.82%	0.64%	0.69%	32.49%	15.89%	4.60%	3.56%	1.23%	12.02%
	QSVD-noQ	77.00%	62.77%	42.59%	9.87%	0.20%	64.80%	50.46%	36.24%	3.60%	2.25%	34.98%
	WSVD-noQ	76.30%	71.74%	60.93%	39.51%	27.27%	65.78%	63.29%	54.45%	29.22%	27.35%	51.58%
	LSVD-noQ	**84.58%**	**83.69%**	**82.00%**	**81.11%**	**78.14%**	**68.54%**	**68.10%**	**67.62%**	**67.12%**	**66.17%**	**74.51%**
	FP16	Accuracy: 84.58%					Accuracy: 68.47%					76.53%
LLaVA-v1.5 7B	SVD-LLM	65.44%	63.71%	61.92%	57.41%	55.53%	57.89%	57.50%	55.33%	54.64%	55.31%	58.47%
	QSVD-noQ	67.72%	**68.22%**	67.08%	65.05%	62.37%	59.84%	59.07%	59.78%	59.00%	58.23%	62.64%
	WSVD-noQ	**68.17%**	67.72%	**67.28%**	65.89%	65.49%	60.10%	**60.17%**	59.89%	**60.18%**	60.46%	63.54%
	LSVD-noQ	68.12%	68.02%	67.03%	**67.08%**	**66.73%**	**60.14%**	60.00%	**60.55%**	59.82%	**60.60%**	**63.81%**
	FP16	Accuracy: 68.01%					Accuracy: 60.18%					64.10%
LLaVA-Next 13B	SVD-LLM	72.53%	72.24%	71.74%	71.15%	70.55%	70.76%	70.63%	70.25%	69.96%	69.58%	70.94%
	QSVD-noQ	71.94%	72.14%	71.74%	72.14%	71.79%	71.23%	71.02%	71.06%	70.92%	70.40%	71.44%
	WSVD-noQ	72.88%	72.98%	**73.57%**	**73.48%**	73.28%	71.29%	71.17%	71.25%	70.95%	70.81%	72.17%
	LSVD-noQ	**73.13%**	**73.03%**	73.38%	73.13%	**73.67%**	**71.39%**	**71.40%**	**71.33%**	**71.37%**	**71.25%**	**72.31%**
	FP16	Accuracy: 73.23%					Accuracy: 71.30%					72.27%

QKV projection matrices to ensure a fair comparison with LSVD, while leaving other linear layers unchanged. In addition, LSVD introduces low-rank approximation for low-precision FFN under quantized settings. For FFN optimization, we set $\gamma = 50\%$ and retain $\rho_{\text{ffn}} = 90\%$ of the selected SVD branch parameters for Algorithm 1. We denote the parameter ratio by ρ_1. For rank-r SVD of a per-head projection $W_h \in \mathbb{R}^{E\times H}$, where E is the hidden dimension and H is the head dimension, $\rho_1 = (E+H)r/(EH)$, while KV-cache size is reduced by the rank ratio r/H. Additional results are provided in Appendix A.5.

To isolate the impact of SVD from quantization, we introduce **LSVD-noQ**, which applies only the loss-aware SVD and rank allocation techniques described in Sec. 3.2 and 3.3. We compare it with SVD-LLM, QSVD-noQ and WSVD-noQ (unquantized version of QSVD/WSVD). We then apply quantization-aware whitening (Sec. 3.4), low-rank approximation for low-precision FFN (Sec. 3.5) and full-model RTN quantization on top of LSVD-noQ, benchmarking against DuQuant, QVLM, QSVD and WSVD. All experiments are conducted on NVIDIA H100 GPUs.

4.1 Results in FP16

We first evaluate the FP16 performance of LSVD-noQ under different rank budgets. To ensure fairness, we align the parameter ratio ρ_1 across all methods and compare their accuracy performance under the same ρ_1. For LSVD, ρ_1 is defined in Sec. 4, while for other SVD-based baselines, ρ_1 is defined as the proportion of parameters relative to the original model after SVD is applied.

The evaluation results are summarized in Tab. 1, with detailed results in Appendix A.5.1. Under the same parameter ratio ρ_1, LSVD-noQ surpasses SVD-LLM, QSVD-noQ, and WSVD-noQ in accuracy in most cases. On large-scale models such as LLaVA-Next 13B, LSVD-noQ incurs less than a 1% accuracy drop on ScienceQA-IMG and SEED-Bench compared to the FP16 baseline. Notably, for LLaVA-Next 13B, when $\rho_1 = 50\%$, LSVD-noQ even outperforms the FP16 model on ScienceQA-IMG, exceeding the FP16 baseline by more than 0.4%. One possible reason is that low-rank approximation may implicitly reduce hallucinations [32], although this hypothesis requires further validation. Across the evaluated models and compression ratios, LSVD-noQ achieves the highest aggregate average accuracy, with particularly pronounced improvements under aggressive compression. For example, on SmolVLM, LSVD-noQ maintains over 78% accuracy on ScienceQA-IMG, while other baselines fail to produce usable results under the same parameter-ratio settings.

4.2 Results in Quantized Settings

We present results under two weight–activation quantization configurations. LSVD and WSVD use W8A8 with $\rho_1 = 50\%$: A8 halves the KV-cache storage, and SVD keeps nearly half of the K/V channels, yielding about 25% of the FP16 full KV-cache size. Other baselines use W8A4 with a full-dimensional KV cache. These config-

Table 2: Accuracy evaluation of different methods under low-precision on LLaVA-v1.5 7B, Next 7B and 13B.

Method	ScienceQA-IMG ↑			SEED-Bench ↑			Avg. ↑
	v1.5 7B	Next 7B	Next 13B	v1.5 7B	Next 7B	Next 13B	
DuQuant	57.36%	66.34%	70.20%	54.11%	63.64%	66.15%	62.97%
QVLM	55.24%	60.60%	65.28%	50.13%	50.38%	65.39%	57.84%
QSVD	65.61%	66.10%	70.43%	58.49%	65.63%	69.21%	65.91%
WSVD	64.25%	66.94%	**73.08%**	60.23%	67.49%	70.67%	67.11%
LSVD	**66.14%**	**68.37%**	72.11%	**61.94%**	**67.49%**	**71.56%**	**67.94%**
FP16	68.10%	69.60%	73.23%	60.18%	69.02%	71.30%	68.57%

urations provide approximately matched KV-cache memory footprints rather than identical quantization precision. In particular, WSVD serves as a directly matched low-rank baseline under the same W8A8 precision and parameter ratio, while comparisons with W8A4 baselines evaluate accuracy under comparable KV-cache memory budgets. LSVD further reduces the parameter footprint through the hybrid FFN compression described in Sec. 3.5.

For activation quantization, we adopt per-token symmetric quantization. For weight quantization, we employ RTN with per-channel symmetric scaling and a learnable clipping ratio, where the clipping value is selected via linear search to minimize squared error, following QuaRot [2]. This quantization scheme is applied to the per-head Q/K/V weight matrices and all remaining attention and feed-forward modules, ensuring that the dominant matrix multiplications in each transformer block are executed in low precision. As shown in Tab. 2, LSVD consistently outperforms the baselines in most cases, despite using a smaller parameter budget and the same cache size. On average across models and datasets, LSVD incurs only a modest accuracy drop of just about 0.5% relative to the FP16 baseline, while reducing cache size to 25% of the FP16 model.

4.3 Ablation Studies

Effectiveness of Loss-Aware SVD. We evaluate loss-aware SVD against a vanilla SVD baseline using the same importance-score-based rank allocation and compression ratio $\rho_1 = 50\%$ under FP16. The *Vanilla* baseline applies SVD directly to the weights, while LSVD performs SVD in the loss-aware transformed space from Sec. 3.2. As shown in Tab. 3, LSVD consistently outperforms vanilla SVD across all evaluated models, improving average accuracy from 51.02% to 72.95%, a 21.93-percentage-point improvement. The improvement is especially large on Qwen2-VL 7B and SmolVLM 2B, where vanilla SVD degrades severely, while LSVD preserves much higher accuracy.

Table 3: Results of loss-aware SVD ablation.

Method	ScienceQA-IMG ↑					Avg. ↑
	v1.5 7B	Next 7B	Next 13B	Q2 7B	S 2B	
FP16	68.10%	69.60%	73.23%	84.38%	84.58%	75.98%
Vanilla	62.87%	67.72%	71.28%	53.25%	0.00%	51.02%
LSVD	**66.73%**	**70.80%**	**73.67%**	**75.41%**	**78.14%**	**72.95%**

Effectiveness of Loss-Aware Cross-Layer Rank Allocation. We evaluate the loss-aware cross-layer rank allocation in Sec. 3.3 by comparing it against two allocation baselines under W8A8. The *Uniform* baseline keeps the same number of top-k singular components in each layer, without performing cross-layer rank allocation. The *SV-based* baseline follows the naive singular-value-based strategy discussed in the beginning of Sec. 3.3, which ranks components globally only by their singular values. As shown in Tab. 4, LSVD consistently outperforms both baselines across all evaluated models. On average, LSVD outperforms uniform and SV-based allocation by 9.35 and 7.89 percentage points. Additional ablations in Appendix A.5.3 further show that QAW improves low-precision SVD, while SVD-aware permutation preserves accuracy under hybrid FFN compression.

Table 4: Ablation of rank allocation.

Method	ScienceQA-IMG ↑			Avg. ↑
	v1.5 7B	Next 7B	Next 13B	
FP16	68.10%	69.60%	73.23%	70.31%
Uniform	58.06%	59.84%	60.63%	59.51%
SV-based	59.05%	55.63%	68.22%	60.97%
LSVD	**66.14%**	**68.34%**	**72.11%**	**68.86%**

4.4 System Evaluation

We evaluate the system-level performance of LSVD under the quantized setting in Sec. 4.2, focusing on decoding acceleration. We measure layer-wise decoding latency of LLaVA-Next 7B across attention and FFN modules using our fused kernels from Sec. 3.6 on RTX 4090 and 5090 GPUs. We compare against Eager Attention without Flash Decoding, Palu [6], Flash Decoding [11], and WSVD [50]. Flash Decoding uses SDPA, while Palu and WSVD use their official implementations. Eager Attention and Flash Decoding operate on the full KV cache. For low-rank methods, Palu and LSVD use $\rho_1 = 50\%$, retaining approximately half of the original K/V dimensions, while WSVD uses $\rho_1 = 60\%$ to provide a

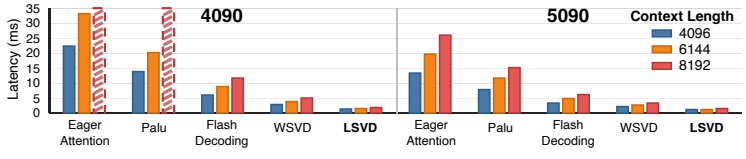

Figure 5: Latency evaluations on RTX 4090 and 5090. Dashed-line columns indicate out-of-memory cases.

closer accuracy-oriented comparison. Since Palu uses FP16 and LSVD uses INT8 KV-cache storage, LSVD further reduces the KV-cache footprint to approximately 25% of the full-dimensional FP16 baseline. Both WSVD and LSVD use INT8 W8A8 quantization, while others operate at FP16, and all measurements use batch size 32. As shown in Fig. 5, LSVD consistently outperforms all baselines on both GPUs, achieving on average a $4.7\times$ speedup over Flash Decoding and a $2.3\times$ speedup over WSVD. The gains come from the fused kernels in Sec. 3.6: the W8A8 low-rank decoding kernel reduces attention-side materialization and KV-cache memory traffic, while the hybrid FFN kernel fuses low-rank reconstruction, SwiGLU, quantization, and accumulation. By reducing memory movement and kernel launch overhead, LSVD delivers consistent decoding latency reduction across context lengths and GPUs while preserving accuracy.

5 Conclusion and Limitations

LSVD provides a loss-aware low-rank compression framework for efficient VLM inference by aligning SVD with model-loss sensitivity, allocating ranks across layers and heads using calibration gradients, and extending compression to FFN layers with a hybrid SVD–quantization design. Currently, LSVD is evaluated on representative VLMs and benchmarks, and broader validation on newer architectures, longer contexts, and more aggressive low-bit settings remains future work.

Acknowledgments and Disclosure of Funding

The authors declare no funding or competing interests to disclose.

References

[1] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[2] Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

[3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization. *Text Reading, and Beyond*, 2(1):1, 2023.

[4] Yakoub Bazi, Mohamad Mahmoud Al Rahhal, Laila Bashmal, and Mansour Zuair. Vision–language model for visual question answering in medical imagery. *Bioengineering*, 10(3):380, 2023.

[5] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *CoRR*, 2024.

[6] Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S Abdelfattah, and Kai-Chiang Wu. Palu: Compressing kv-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

[7] Christel Chappuis, Valérie Zermatten, Sylvain Lobry, Bertrand Le Saux, and Devis Tuia. Prompt-rsvqa: Prompting visual context to a language model for remote sensing visual question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1372–1381, 2022.

[8] Viktoriia Chekalina, Daniil Moskovskiy, Daria Cherniuk, Maxim Kurkin, Andrey Kuznetsov, and Evgeny Frolov. Generalized fisher-weighted svd: Scalable kronecker-factored fisher approximation for compressing large language models. *arXiv preprint arXiv:2505.17974*, 2025.

[9] Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18030–18040, 2022.

[10] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: towards general-purpose vision-language models with instruction tuning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[11] Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. Flash-decoding for long-context inference. `https://pytorch.org/blog/flash-decoding/`, October 2023. Accessed: 2025-09-22.

[12] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11198–11201, 2024.

[13] Maksim Dzabraev, Alexander Kunitsyn, and Andrei Ivaniuta. Vlrm: Vision-language models act as reward models for image captioning. *arXiv preprint arXiv:2404.01911*, 2024.

[14] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1 (3):211–218, 1936.

[15] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[16] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[17] Nicholas J Higham. Analysis of the cholesky decomposition of a semi-definite matrix. 1990.

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

[19] Xiaowei Hu, Zhe Gan, Jianfeng Wang, Zhengyuan Yang, Zicheng Liu, Yumao Lu, and Lijuan Wang. Scaling up vision-language pre-training for image captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17980–17989, 2022.

[20] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374 (2065):20150202, 2016.

[21] Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench: Benchmarking multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13299–13308, 2024.

[22] Chuanhao Li, Zhen Li, Chenchen Jing, Shuo Liu, Wenqi Shao, Yuwei Wu, Ping Luo, Yu Qiao, and Kaipeng Zhang. Searchlvlms: A plug-and-play framework for augmenting large vision-language models by searching up-to-date internet knowledge. *arXiv preprint arXiv:2405.14554*, 2024.

[23] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.

[24] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[25] Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdqunat: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024.

[26] Shiyao Li, Yingchun Hu, Xuefei Ning, Xihui Liu, Ke Hong, Xiaotao Jia, Xiuhong Li, Yaqi Yan, Pei Ran, Guohao Dai, et al. Mbq: Modality-balanced quantization for large vision-language models. *arXiv preprint arXiv:2412.19509*, 2024.

[27] Yuhang Li, Donghyun Lee, Ruokai Yin, and Priyadarshini Panda. Optimal brain decomposition for accurate llm low-rank approximation. *arXiv preprint arXiv:2604.00821*, 2026.

[28] Zhiteng Li, Mingyuan Xia, Jingyuan Zhang, Zheng Hui, Linghe Kong, Yulun Zhang, and Xiaokang Yang. Adasvd: Adaptive singular value decomposition for large language models. *arXiv preprint arXiv:2502.01403*, 2025.

[29] Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems*, 37:87766–87800, 2024.

[30] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

[31] Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *Proceedings of Machine Learning and Systems*, 7, 2025.

[32] Hanchao Liu, Wenyuan Xue, Yifei Chen, Dapeng Chen, Xiutian Zhao, Ke Wang, Liping Hou, Rongjun Li, and Wei Peng. A survey on hallucination in large vision-language models. *arXiv preprint arXiv:2402.00253*, 2024.

[33] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[34] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer, 2024.

[35] Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024.

[36] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[37] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, et al. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.

[38] James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

[39] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[40] Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59, 1960.

[41] Matan Ben Noach and Yoav Goldberg. Compressing pre-trained language models by matrix decomposition. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 884–889, 2020.

[42] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

[43] Yifan Shen, Yuanzhe Liu, Jingyuan Zhu, Xu Cao, Xiaofeng Zhang, Yixiao He, Wenming Ye, James Rehg, and Ismini Lourentzou. Fine-grained preference optimization improves spatial reasoning in vlms. *Advances in Neural Information Processing Systems*, 38:17929–17960, 2026.

[44] Zelong Sun, Dong Jing, Guoxing Yang, Nanyi Fei, and Zhiwu Lu. Leveraging large vision-language model as user intent-aware encoder for composed image retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7149–7157, 2025.

[45] Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024.

[46] Tycho FA van der Ouderaa, Markus Nagel, Mart Van Baalen, Yuki M Asano, and Tijmen Blankevoort. The llm surgeon. *arXiv preprint arXiv:2312.17244*, 2023.

[47] Changyuan Wang, Ziwei Wang, Xiuwei Xu, Yansong Tang, Jie Zhou, and Jiwen Lu. Q-vlm: Post-training quantization for large vision-language models. *arXiv preprint arXiv:2410.08119*, 2024.

[48] Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the kronecker-factored eigenbasis. In *International conference on machine learning*, pages 6566–6575. PMLR, 2019.

[49] Guankun Wang, Long Bai, Wan Jun Nah, Jie Wang, Zhaoxi Zhang, Zhen Chen, Jinlin Wu, Mobarakol Islam, Hongbin Liu, and Hongliang Ren. Surgical-lvlm: Learning to adapt large vision-language model for grounded visual question answering in robotic surgery. *arXiv preprint arXiv:2405.10948*, 2024.

[50] Haiyu Wang, Yutong Wang, Jack Jiang, and Sai Qian Zhang. WSVD: Weighted low-rank approximation for fast and efficient execution of low-precision vision-language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=zrmQ4koOw9`.

[51] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[52] Qinsi Wang, Jinghan Ke, Masayoshi Tomizuka, Yiran Chen, Kurt Keutzer, and Chenfeng Xu. Dobi-svd: Differentiable svd for llm compression and some new perspectives. *arXiv preprint arXiv:2502.02723*, 2025.

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

[54] Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. Svd-llm v2: Optimizing singular value truncation for large language model compression. *arXiv preprint arXiv:2503.12340*, 2025.

[55] Yutong Wang, Haiyu Wang, and Sai Qian Zhang. Qsvd: Efficient low-rank approximation for unified query-key-value weight compression in low-precision vision-language models. *arXiv preprint arXiv:2510.16292*, 2025.

[56] Jingyang Xiang and Sai Qian Zhang. Dfrot: Achieving outlier-free and massive activation-free for rotated llms with refined rotation. *arXiv preprint arXiv:2412.00648*, 2024.

[57] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

[58] Xi Xiao, Yunbei Zhang, Xingjian Li, Tianyang Wang, Xiao Wang, Yuxiang Wei, Jihun Hamm, and Min Xu. Visual instance-aware prompt tuning. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 2880–2889, 2025.

[59] Xi Xiao, Xingjian Li, Cheng Han, Tianyang Wang, Lin Zhao, Yunbei Zhang, Guosheng Hu, Runmin Jiang, Xi Li, Xiao Wang, and Min Xu. Adapting vision foundation models with cascaded semantics. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL `https://openreview.net/forum?id=SSsobNZJPO`.

[60] Xi Xiao, Xingjian Li, Yunbei Zhang, Cheng Han, Tianming Liu, Tianyang Wang, Runmin Jiang, Jihun Hamm, Xiao Wang, and Min Xu. Layer-specific prompt fusion discovery via differentiable search in vision foundation models. In *European Conference on Computer Vision*, pages 391–409. Springer, 2026.

[61] Xi Xiao, Chen Liu, Chih-Ting Liao, Yunbei Zhang, Qizhen Lan, Yuxiang Wei, Lin Zhao, Janet Wang, Jianyang Gu, Muchao Ye, et al. Staying vigil ant: Mitigating visual laziness via counterfactual visual alignment in mllms. In *European Conference on Computer Vision*, pages 287–305. Springer, 2026.

[62] Xi Xiao, Chenrui Ma, Yunbei Zhang, Chen Liu, Zhuxuanzi Wang, Yanshu Li, Lin Zhao, Guosheng Hu, Tianyang Wang, and Hao Xu. Not all directions matter: Towards structured and task-aware low-rank model adaptation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2132–2154, 2026.

[63] Xiangyang Yin, Xingyu Liu, Tianhua Xia, Bo Bao, Vithursan Thangarasa, Valavan Manohararajah, Eric Sather, and Sai Qian Zhang. Codequant: Unified clustering and quantization for enhanced outlier smoothing in low-precision mixture-of-experts. *arXiv preprint arXiv:2604.10496*, 2026.

[64] Hao Yu, Zelan Yang, Shen Li, Yong Li, and Jianxin Wu. Effectively compress kv heads for llm. *arXiv preprint arXiv:2406.07056*, 2024.

[65] Tianjiao Yu, Xinzhuo Li, Yifan Shen, Yuanzhe Liu, and Ismini Lourentzou. Core3d: Collaborative reasoning as a foundation for 3d intelligence. *arXiv preprint arXiv:2512.12768*, 2025.

[66] Tianjiao Yu, Xinzhuo Li, Yifan Shen, Onkar Susladkar, Yuanzhe Liu, Xiaona Zhou, and Ismini Lourentzou. Elsa3d: Elastic semantic anchoring for unified 3d understanding and generation. *arXiv preprint arXiv:2607.06565*, 2026.

[67] Zhengqing Yuan, Zhaoxu Li, Weiran Huang, Yanfang Ye, and Lichao Sun. Tinygpt-v: Efficient multimodal large language model via small backbones. *arXiv preprint arXiv:2312.16862*, 2023.

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

[69] Baichuan Zhou, Ying Hu, Xi Weng, Junlong Jia, Jie Luo, Xien Liu, Ji Wu, and Lei Huang. Tinyllava: A framework of small-scale large multimodal models. *arXiv preprint arXiv:2402.14289*, 2024.

A Technical appendices and supplementary material

A.1 K-FAC Loss Surrogate for Weight Compression

Derivation. By Taylor's theorem, the loss variation induced by ΔW satisfies:

$$\Delta\mathcal{L} = \mathcal{L}(W+\Delta W) - \mathcal{L}(W) = \langle \nabla_W \mathcal{L}(W), \Delta W\rangle + \frac{1}{2}\mathrm{vec}(\Delta W)^\top H_W \mathrm{vec}(\Delta W) + O(\|\Delta W\|_F^3) \quad (7)$$

where $H_W = \nabla_W^2 \mathcal{L}(W)$. For a pretrained model close to a stationary point and a small compression perturbation, the first-order term and higher-order remainder are negligible, giving:

$$\Delta\mathcal{L} \approx \frac{1}{2}\mathrm{vec}(\Delta W)^\top H_W \mathrm{vec}(\Delta W) \quad (8)$$

Since directly forming the Hessian block H_W is prohibitively expensive for large models, following standard natural-gradient and second-order approximations, we replace the Hessian block with the Fisher block [1, 38], and then use K-FAC to obtain a tractable Kronecker-factored curvature surrogate [39]. For the linear layer $y = xW$, this gives:

$$\begin{aligned} H_W \approx F_W &= \mathbb{E}\Big[\mathrm{vec}(\nabla_W \ell)\mathrm{vec}(\nabla_W \ell)^\top\Big], \qquad \nabla_W \ell = x^\top g, \qquad \mathrm{vec}(\nabla_W \ell) = g^\top \otimes x^\top, \\ F_W &= \mathbb{E}\Big[(g^\top \otimes x^\top)(g \otimes x)\Big] = \mathbb{E}\Big[(g^\top g) \otimes (x^\top x)\Big] \approx \mathbb{E}[g^\top g] \otimes \mathbb{E}[x^\top x] = G \otimes X \end{aligned} \quad (9)$$

Here $X = \mathbb{E}[x^\top x]$ and $G = \mathbb{E}[g^\top g]$ are estimated from calibration data. Substituting $H_W \approx G \otimes X$ into Eq. 8 yields Eq. 1.

A.2 Proof of Theorem 3.2

Proof. Let $\Delta W = W - \widehat{W}_r$. From Assumption 3.1, using $X = S_x^\top S_x$ and $G = S_g S_g^\top$, together with the Kronecker-product identities $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ and $\mathrm{vec}(ABC) = (C^\top \otimes A)\mathrm{vec}(B)$, we obtain:

$$\begin{aligned} \Delta\mathcal{L} &\approx \frac{1}{2}\mathrm{vec}(\Delta W)^\top (G \otimes X)\mathrm{vec}(\Delta W) = \frac{1}{2}\mathrm{vec}(\Delta W)^\top (S_g S_g^\top \otimes S_x^\top S_x)\mathrm{vec}(\Delta W) \\ &= \frac{1}{2}\mathrm{vec}(\Delta W)^\top (S_g \otimes S_x^\top)(S_g^\top \otimes S_x)\mathrm{vec}(\Delta W) = \frac{1}{2}\left\|(S_g^\top \otimes S_x)\mathrm{vec}(\Delta W)\right\|_2^2 \\ &= \frac{1}{2}\left\|\mathrm{vec}(S_x \Delta W S_g)\right\|_2^2 = \frac{1}{2}\left\|S_x \Delta W S_g\right\|_F^2 \end{aligned} \quad (10)$$

Therefore, minimizing the estimated loss increase in Assumption 3.1 over rank-r approximations is equivalent to:

$$\min_{\mathrm{rank}(\widehat{W}_r)\le r} \left\|S_x(W - \widehat{W}_r)S_g\right\|_F^2 \quad (11)$$

Define the curvature-weighted weight and rank-r approximation as $\widetilde{W} = S_x W S_g$ and $\widetilde{W}_r = S_x \widehat{W}_r S_g$. Since S_x and S_g are nonsingular, left and right multiplication by them is rank-preserving. Thus Eq. 11 reduces to the standard Frobenius-norm rank-r approximation problem:

$$\min_{\mathrm{rank}(\widetilde{W}_r)\le r} \left\|\widetilde{W} - \widetilde{W}_r\right\|_F^2 \quad (12)$$

By the Eckart–Young–Mirsky theorem [14, 40], the optimal solution is the truncated SVD $\widetilde{W}_r^\star = U_r \Sigma_r V_r^\top$. Mapping it back to the original weight space gives:

$$\widehat{W}_r^\star = S_x^{-1}\widetilde{W}_r^\star S_g^{-1} = S_x^{-1} U_r \Sigma_r V_r^\top S_g^{-1} \quad (13)$$

Moreover, the optimal residual in the transformed space is $\|\widetilde{W} - \widetilde{W}_r^\star\|_F^2 = \sum_{i=r+1}^q \sigma_i^2$. Substituting this residual into Eq. 10 gives: $\Delta\mathcal{L}_r \approx \frac{1}{2}\sum_{i=r+1}^q \sigma_i^2$. This proves Theorem 3.2. □

A.3 K-FAC Approximation Error Analysis

We analyze why the local loss estimate $\frac{1}{2}\sigma_{\ell,i}^2$ obtained in the loss-aware transformed space is useful within one layer or head, but may not be directly comparable across different layers or heads. Let F_ℓ denote the empirical Fisher matrix of layer ℓ, and let

$$K_\ell = G_\ell \otimes X_\ell$$

be the Kronecker-factored approximation used in Assumption 3.1. We write the difference between the empirical Fisher and this K-FAC approximation as:

$$F_\ell = K_\ell + R_\ell, \qquad R_\ell \triangleq F_\ell - K_\ell$$

Here, R_ℓ contains the curvature information that is not captured by the Kronecker factorization.

For the i-th loss-aware singular value of layer ℓ, $\sigma_{\ell,i}$, let:

$$W_{\ell,i}^{\mathrm{svd}} = S_{x,\ell}^{-1}\sigma_{\ell,i}u_{\ell,i}v_{\ell,i}^\top S_{g,\ell}^{-1}$$

denote the corresponding rank-one component mapped back to the original weight space. Dropping this component induces the weight perturbation:

$$\Delta W_{\ell,i}^{\mathrm{drop}} = -W_{\ell,i}^{\mathrm{svd}}, \qquad z_{\ell,i} = \mathrm{vec}(\Delta W_{\ell,i}^{\mathrm{drop}}).$$

Here, $W_{\ell,i}^{\mathrm{svd}}$ denotes the component itself, while $\Delta W_{\ell,i}^{\mathrm{drop}}$ denotes the perturbation caused by removing it. The minus sign does not affect the following quadratic loss estimate.

For this discarded loss-aware singular component $z_{\ell,i} = \mathrm{vec}(\Delta W_{\ell,i}^{\mathrm{drop}})$, the empirical-Fisher second-order loss estimate can be decomposed as:

$$\begin{aligned}\Delta\mathcal{L}_{\ell,i}^F &\approx \frac{1}{2}z_{\ell,i}^\top F_\ell z_{\ell,i} \\ &= \frac{1}{2}z_{\ell,i}^\top K_\ell z_{\ell,i} + \frac{1}{2}z_{\ell,i}^\top (F_\ell - K_\ell) z_{\ell,i} \\ &\triangleq T_{\ell,i}^{\mathrm{KFAC}} + T_{\ell,i}^{\mathrm{res}}\end{aligned} \tag{14}$$

By Theorem 3.2, the K-FAC-based term is:

$$T_{\ell,i}^{\mathrm{KFAC}} = \frac{1}{2}\|S_{x,\ell}\Delta W_{\ell,i}^{\mathrm{drop}} S_{g,\ell}\|_F^2 = \frac{1}{2}\sigma_{\ell,i}^2$$

Thus, $\frac{1}{2}\sigma_{\ell,i}^2$ is the loss increase predicted by the K-FAC surrogate when the i-th loss-aware singular component is removed. This estimate is suitable for comparing components within the same layer or head, because all candidates are evaluated under the same local approximation. However, when components from different layers or heads are ranked together, the K-FAC surrogate in each layer or head may over- or under-estimate the empirical-Fisher loss by a different amount.

To make this difference explicit, we compare the empirical-Fisher and K-FAC approximation using the trace ratio:

$$\eta_\ell = \frac{\mathrm{Tr}(F_\ell)}{\mathrm{Tr}(K_\ell)} = \frac{\mathrm{Tr}(F_\ell)}{\mathrm{Tr}(G_\ell \otimes X_\ell)} = \frac{\mathrm{Tr}(F_\ell)}{\mathrm{Tr}(G_\ell)\,\mathrm{Tr}(X_\ell)} \tag{15}$$

The trace measures the total curvature mass of a positive semi-definite matrix. Therefore, η_ℓ indicates whether the K-FAC surrogate tends to under- or over-estimate the empirical-Fisher loss scale on average. Specifically, $\eta_\ell > 1$ means that K-FAC underestimates the total empirical-Fisher curvature of layer ℓ, while $\eta_\ell < 1$ means that it overestimates it.

Derivation. To interpret this trace ratio, consider an isotropic perturbation $z = au$ in the vectorized weight space of layer ℓ, where $a > 0$ and $\mathbb{E}_u[uu^\top] = I/d_\ell$. For any symmetric curvature matrix C, the quadratic loss surrogate along $z = au$ satisfies:

$$\mathbb{E}_u\left[\frac{1}{2}z^\top C z\right] = \frac{a^2}{2}\mathbb{E}_u[u^\top C u] \tag{16}$$

Using $u^\top C u = \mathrm{Tr}(Cuu^\top)$ and $\mathbb{E}_u[uu^\top] = I/d_\ell$, we have:

$$\mathbb{E}_u[u^\top C u] = \mathrm{Tr}\Big(C\mathbb{E}_u[uu^\top]\Big) = \frac{1}{d_\ell}\,\mathrm{Tr}(C) \tag{17}$$

Therefore,

$$\mathbb{E}_u\left[\frac{1}{2}z^\top C z\right] = \frac{a^2}{2d_\ell}\,\mathrm{Tr}(C)$$

Applying this result to $C = F_\ell$ and $C = K_\ell$ cancels the common factor $a^2/(2d_\ell)$ and gives:

$$\frac{\mathbb{E}_u\left[\frac{1}{2}z^\top F_\ell z\right]}{\mathbb{E}_u\left[\frac{1}{2}z^\top K_\ell z\right]} = \frac{\mathrm{Tr}(F_\ell)}{\mathrm{Tr}(K_\ell)} = \eta_\ell \tag{18}$$

Similarly, applying the same result to $C = F_\ell - K_\ell$ gives the direction-averaged relative error of the K-FAC curvature:

$$\frac{\mathbb{E}_u\left[\frac{1}{2}z^\top(F_\ell - K_\ell)z\right]}{\mathbb{E}_u\left[\frac{1}{2}z^\top K_\ell z\right]} = \eta_\ell - 1 \tag{19}$$

Finally, since $K_\ell = G_\ell \otimes X_\ell$, the Kronecker trace identity gives:

$$\mathrm{Tr}(K_\ell) = \mathrm{Tr}(G_\ell \otimes X_\ell) = \mathrm{Tr}(G_\ell)\,\mathrm{Tr}(X_\ell)$$

which yields Eq. 15.

This derivation shows that η_ℓ is a simple indicator of how the K-FAC-based loss estimate compares with the empirical-Fisher loss estimate on average. It is not a per-component correction, because the exact residual term $T^{\mathrm{res}}_{\ell,i}$ depends on the direction of the discarded component. However, large variation in η_ℓ across layers or heads means that $\frac{1}{2}\sigma^2_{\ell,i}$ may not put all components on the same loss scale. Therefore, directly sorting components from different layers or heads by $\sigma^2_{\ell,i}$ can lead to inaccurate global rank allocation.

The trace-ratio indicator is also easy to compute in practice. Since the trace only depends on diagonal entries, we do not need to materialize the full empirical Fisher. Instead, we only need its diagonal entries, which can be estimated from calibration gradients. For each calibration sample n, we run a forward and backward pass, collect the per-sample gradient $\nabla_{W_\ell}\ell_n$, square it elementwise, and accumulate it over the calibration set:

$$\mathbf{f}^{\mathrm{diag}}_\ell \approx \frac{1}{N}\sum_{n=1}^{N} \mathrm{vec}(\nabla_{W_\ell}\ell_n) \odot \mathrm{vec}(\nabla_{W_\ell}\ell_n)$$

where $\mathbf{f}^{\mathrm{diag}}_\ell$ is a vector that estimates the diagonal entries of the empirical Fisher, and $\odot$ denotes elementwise multiplication. Summing its entries gives the empirical-Fisher trace:

$$\mathrm{Tr}(F_\ell) = \sum_j [F_\ell]_{j,j} \approx \sum_j [\mathbf{f}^{\mathrm{diag}}_\ell]_j = \frac{1}{N}\sum_{n=1}^{N} \|\nabla_{W_\ell}\ell_n\|_F^2 \,. \tag{20}$$

Thus, the practical computation only requires two trace values: $\mathrm{Tr}(F_\ell)$ and $\mathrm{Tr}(K_\ell)$. The empirical-Fisher trace $\mathrm{Tr}(F_\ell)$ is estimated by accumulating squared per-sample calibration gradients, while the K-FAC trace $\mathrm{Tr}(K_\ell)$ is computed directly from the Kronecker factors:

$$\mathrm{Tr}(K_\ell) = \mathrm{Tr}(G_\ell \otimes X_\ell) = \mathrm{Tr}(G_\ell)\,\mathrm{Tr}(X_\ell)$$

We estimate both quantities on the same calibration set used for loss-aware SVD and rank allocation. This analysis supports the observation in Sec. 3.3: the local loss estimate $\frac{1}{2}\sigma^2_{\ell,i}$ is useful for deciding which components to truncate within a layer or head, but it is not always reliable for ranking components globally. This motivates the empirical-Fisher importance score derived in Appendix A.4, which keeps the loss-aware SVD basis while calibrating each singular component on a shared empirical loss scale.

A.4 Loss-Aware Rank Allocation

We provide the derivation of the calibration-gradient-based importance score used for global rank allocation. For a target layer ℓ, let the loss-aware SVD of the curvature-weighted weight be $\widetilde{W}_\ell = S_{x,\ell}W_\ell S_{g,\ell} = U_\ell \Sigma_\ell V_\ell^\top$, where $\Sigma_\ell = \mathrm{diag}(\sigma_{\ell,1}, \ldots, \sigma_{\ell,q_\ell})$. The rank-one component corresponding to the i-th singular value $\sigma_{\ell,i}$ in the original weight space is:

$$W^{\mathrm{svd}}_{\ell,i} = S^{-1}_{x,\ell}\sigma_{\ell,i}u_{\ell,i}v^\top_{\ell,i}S^{-1}_{g,\ell}$$

Under the local K-FAC surrogate, discarding this component gives the loss estimate $\frac{1}{2}\sigma^2_{\ell,i}$. As discussed in Sec. 3.3, this quantity is meaningful for local truncation within the same layer or head, but its scale may not be directly comparable across different layers or heads.

To obtain a shared empirical scale for global rank allocation, we estimate the loss increase of removing each loss-aware singular component under the empirical Fisher. Recall that LSVD performs SVD on

the curvature-whitened weight $\widetilde{W}_\ell = S_{x,\ell} W_\ell S_{g,\ell} = U_\ell \Sigma_\ell V_\ell^\top$. We introduce a binary mask $m_{\ell,i}$ for each singular component in this whitened space:

$$\widetilde{W}_\ell(m) = \sum_{i=1}^{q_\ell} m_{\ell,i} \sigma_{\ell,i} u_{\ell,i} v_{\ell,i}^\top$$

Mapping it back to the original weight space gives:

$$\widehat{W}_\ell(m) = S_{x,\ell}^{-1} \widetilde{W}_\ell(m) S_{g,\ell}^{-1} = \sum_{i=1}^{q_\ell} m_{\ell,i} S_{x,\ell}^{-1} \sigma_{\ell,i} u_{\ell,i} v_{\ell,i}^\top S_{g,\ell}^{-1}$$

Therefore, removing the i-th loss-aware singular component corresponds to the weight perturbation:

$$\Delta W_{\ell,i}^{\mathrm{drop}} = -W_{\ell,i}^{\mathrm{svd}} = -S_{x,\ell}^{-1} \sigma_{\ell,i} u_{\ell,i} v_{\ell,i}^\top S_{g,\ell}^{-1}$$

Using the second-order loss approximation with the empirical Fisher $F_\ell = \mathbb{E}[\mathrm{vec}(\nabla_{W_\ell} \ell)\mathrm{vec}(\nabla_{W_\ell} \ell)^\top]$, the loss increase caused by this perturbation is:

$$\Delta \mathcal{L}_{\ell,i} \approx \frac{1}{2} \mathrm{vec}(\Delta W_{\ell,i}^{\mathrm{drop}})^\top F_\ell \mathrm{vec}(\Delta W_{\ell,i}^{\mathrm{drop}})$$

Substituting the empirical Fisher form gives:

$$\Delta \mathcal{L}_{\ell,i} \approx \frac{1}{2} \mathbb{E}_n \left[\left\langle \nabla_{W_\ell} \ell_n, \Delta W_{\ell,i}^{\mathrm{drop}} \right\rangle^2 \right]$$

where ℓ_n is the calibration loss of the n-th sample. Since the sign of $\Delta W_{\ell,i}^{\mathrm{drop}}$ is removed by the square, we obtain:

$$\Delta \mathcal{L}_{\ell,i} \approx \frac{1}{2} \sigma_{\ell,i}^2 \mathbb{E}_n \left[\left(u_{\ell,i}^\top S_{x,\ell}^{-T} \nabla_{W_\ell} \ell_n S_{g,\ell}^{-T} v_{\ell,i} \right)^2 \right]$$

Equivalently, by defining the whitened-space gradient

$$\nabla_{\widetilde{W}_\ell} \ell_n = S_{x,\ell}^{-T} \nabla_{W_\ell} \ell_n S_{g,\ell}^{-T}$$

the score becomes

$$s_{\ell,i} = \frac{1}{2} \sigma_{\ell,i}^2 \mathbb{E}_n \left[\left(u_{\ell,i}^\top \nabla_{\widetilde{W}_\ell} \ell_n v_{\ell,i} \right)^2 \right]$$

In practice, we estimate the expectation with the calibration set:

$$s_{\ell,i} = \frac{1}{2N} \sigma_{\ell,i}^2 \sum_{n=1}^{N} \left(u_{\ell,i}^\top \nabla_{\widetilde{W}_\ell} \ell_n v_{\ell,i} \right)^2$$

This score measures the empirical-Fisher loss increase of removing each loss-aware singular component. The constant factor $\frac{1}{2}$ is shared by all components and does not affect the global ranking, so we omit it and obtain the importance score used in the main text:

$$I_{\ell,i} = \frac{1}{N} \sigma_{\ell,i}^2 \sum_{n=1}^{N} \left(u_{\ell,i}^\top \nabla_{\widetilde{W}_\ell} \ell_n v_{\ell,i} \right)^2$$

Compared with directly ranking by the K-FAC-predicted local value $\sigma_{\ell,i}^2$, this score keeps the loss-aware SVD basis induced by $S_{x,\ell}$ and $S_{g,\ell}$ while calibrating each component on a shared empirical loss scale across layers and heads.

A.5 Results

A.5.1 More Results in FP16

The FP16 results in Tab. 5 further support the robustness of LSVD under a wide range of low-rank compression ratios. Compared with SVD-LLM, QSVD-noQ, and WSVD-noQ, LSVD-noQ consistently preserves higher accuracy as the parameter-size ratio ρ_1 decreases. The advantage is especially clear under aggressive compression. For example, on SmolVLM-2B, several baselines degrade

Table 5: Accuracy evaluation of different methods under FP16.

Acc.	Method	ScienceQA-IMG ↑					SEED-Bench ↑					Avg. ↑
		ρ_1 : 90%	ρ_1 : 80%	ρ_1 : 70%	ρ_1 : 60%	ρ_1 : 50%	ρ_1 : 90%	ρ_1 : 80%	ρ_1 : 70%	ρ_1 : 60%	ρ_1 : 50%	
SmolVLM 2B	SVD-LLM	40.06%	17.20%	3.82%	0.64%	0.69%	32.49%	15.89%	4.60%	3.56%	1.23%	12.02%
	QSVD-noQ	77.00%	62.77%	42.59%	9.87%	0.20%	64.80%	50.46%	36.24%	3.60%	2.25%	34.98%
	WSVD-noQ	76.30%	71.74%	60.93%	39.51%	27.27%	65.78%	63.29%	54.45%	29.22%	27.35%	51.58%
	LSVD-noQ	**84.58%**	**83.69%**	**82.00%**	**81.11%**	**78.14%**	**68.54%**	**68.10%**	**67.62%**	**67.12%**	**66.17%**	**74.51%**
	FP16	Accuracy: 84.58%					Accuracy: 68.47%					76.53%
Qwen2-VL 7B	SVD-LLM	82.80%	81.11%	80.32%	77.34%	69.31%	74.74%	74.27%	71.85%	60.83%	44.36%	71.69%
	QSVD-noQ	84.09%	83.34%	82.40%	80.57%	**76.38%**	75.06%	74.97%	72.75%	70.63%	65.30%	76.55%
	WSVD-noQ	82.99%	81.90%	67.38%	49.68%	15.87%	75.74%	75.27%	74.53%	72.96%	62.16%	65.85%
	LSVD-noQ	**85.23%**	**84.28%**	**83.74%**	**82.00%**	75.41%	**76.22%**	**76.24%**	**75.96%**	**75.14%**	**69.95%**	**78.42%**
	FP16	Accuracy: 84.38%					Accuracy: 76.29%					80.33%
LLaVA-v1.5 7B	SVD-LLM	65.44%	63.71%	61.92%	57.41%	55.53%	57.89%	57.50%	55.33%	54.64%	55.31%	58.47%
	QSVD-noQ	67.72%	**68.22%**	67.08%	65.05%	62.37%	59.84%	59.07%	59.78%	59.00%	58.23%	62.64%
	WSVD-noQ	**68.17%**	67.72%	**67.28%**	65.89%	65.49%	60.10%	**60.17%**	59.89%	**60.18%**	60.46%	63.54%
	LSVD-noQ	68.12%	68.02%	67.03%	**67.08%**	**66.73%**	**60.14%**	60.00%	**60.55%**	59.82%	**60.60%**	**63.81%**
	FP16	Accuracy: 68.01%					Accuracy: 60.18%					64.10%
LLaVA-Next 7B	SVD-LLM	68.27%	67.92%	66.58%	66.39%	65.54%	68.50%	68.31%	67.65%	67.45%	66.28%	67.29%
	QSVD-noQ	**70.10%**	69.16%	69.01%	68.27%	66.19%	68.86%	68.95%	68.44%	67.98%	67.27%	68.42%
	WSVD-noQ	69.81%	69.56%	69.36%	68.22%	67.87%	69.18%	69.27%	**69.15%**	69.16%	68.59%	69.02%
	LSVD-noQ	69.51%	**69.81%**	**69.86%**	**69.86%**	**70.80%**	**69.25%**	**69.29%**	69.13%	**69.34%**	**68.77%**	**69.45%**
	FP16	Accuracy: 69.60%					Accuracy: 69.02%					69.31%
LLaVA-Next 13B	SVD-LLM	72.53%	72.24%	71.74%	71.15%	70.55%	70.76%	70.63%	70.25%	69.96%	69.58%	70.94%
	QSVD-noQ	71.94%	72.14%	71.74%	72.14%	71.79%	71.23%	71.02%	71.06%	70.92%	70.40%	71.44%
	WSVD-noQ	72.88%	72.98%	**73.57%**	**73.48%**	73.28%	71.29%	71.17%	71.25%	70.95%	70.81%	72.17%
	LSVD-noQ	**73.13%**	**73.03%**	73.38%	73.13%	**73.67%**	**71.39%**	**71.40%**	**71.33%**	**71.37%**	**71.25%**	**72.31%**
	FP16	Accuracy: 73.23%					Accuracy: 71.30%					72.27%

Table 6: Accuracy evaluation of different methods on MMBench under FP16 low-rank compression.

Acc.	Method	MMBench EN V1.1 ↑					Avg. ↑
		ρ_1: 90%	ρ_1: 80%	ρ_1: 70%	ρ_1: 60%	ρ_1: 50%	
SmolVLM 2B	SVD-LLM	44.31%	33.04%	15.74%	0.73%	0.31%	18.83%
	QSVD-noQ	58.39%	52.62%	35.79%	3.38%	0.88%	30.21%
	WSVD-noQ	59.06%	55.90%	51.22%	43.64%	24.52%	46.87%
	LSVD-noQ	**64.31%**	**63.79%**	**62.96%**	**61.56%**	**61.09%**	**62.74%**
	FP16	Accuracy: 64.47%					64.47%
Qwen2-VL 7B	SVD-LLM	79.12%	78.29%	73.92%	50.23%	20.83%	60.48%
	QSVD-noQ	80.16%	79.79%	76.69%	70.06%	64.51%	74.24%
	WSVD-noQ	80.68%	80.00%	76.31%	67.27%	52.81%	71.41%
	LSVD-noQ	**80.68%**	**80.10%**	**79.12%**	**75.43%**	**66.44%**	**76.35%**
	FP16	Accuracy: 80.52%					80.52%
LLaVA-v1.5 7B	SVD-LLM	55.22%	52.52%	47.01%	40.88%	31.06%	45.34%
	QSVD-noQ	61.25%	60.21%	60.17%	58.60%	56.56%	59.36%
	WSVD-noQ	60.82%	60.51%	59.94%	58.64%	56.61%	59.30%
	LSVD-noQ	**61.51%**	**61.45%**	**60.88%**	**60.00%**	**57.87%**	**60.34%**
	FP16	Accuracy: 61.56%					61.56%
LLaVA-Next 7B	SVD-LLM	62.18%	61.04%	58.34%	55.95%	49.82%	57.47%
	QSVD-noQ	64.83%	64.36%	63.64%	62.08%	59.79%	62.94%
	WSVD-noQ	64.84%	64.83%	64.04%	64.10%	63.21%	64.20%
	LSVD-noQ	**64.99%**	**65.09%**	**64.88%**	**64.94%**	**64.26%**	**64.83%**
	FP16	Accuracy: 65.14%					65.14%
LLaVA-Next 13B	SVD-LLM	65.09%	64.52%	61.40%	52.78%	44.05%	57.57%
	QSVD-noQ	67.27%	66.91%	66.86%	66.18%	63.95%	66.23%
	WSVD-noQ	67.48%	66.98%	66.86%	66.08%	65.14%	66.51%
	LSVD-noQ	**67.58%**	**67.38%**	**67.12%**	**66.91%**	**66.44%**	**67.09%**
	FP16	Accuracy: 67.53%					67.53%

sharply at $\rho_1 = 50\%$, while LSVD-noQ still maintains strong performance on both ScienceQA-IMG and SEED-Bench. Similar trends are observed on Qwen2-VL, LLaVA-v1.5, and LLaVA-Next models, showing that the proposed loss-aware decomposition is not specific to a single architecture.

The MMBench EN V1.1 results in Tab. 6 show the same trend on a different evaluation benchmark. LSVD-noQ achieves the best average accuracy among low-rank methods for all evaluated models. This indicates that the gains of LSVD are not limited to ScienceQA-IMG or SEED-Bench, but also transfer to a more general multimodal reasoning benchmark. In particular, LSVD-noQ remains close to the FP16 baseline under mild compression and degrades more gracefully than prior SVD-based methods under stronger compression.

A.5.2 More Results in Quantized Settings

The low-precision results in Tab. 7 demonstrate that LSVD remains effective when low-rank compression is combined with quantization. LSVD achieves the highest average accuracy among all compressed low-precision methods, improving over both quantization-only and SVD-based baselines. Compared with WSVD, LSVD uses a more loss-aware decomposition, which better preserves accuracy after quantization. The remaining gap to FP16 is small, suggesting that LSVD provides a favorable trade-off between compression, low-precision execution, and accuracy.

Table 7: Accuracy evaluation of different methods under low-precision on LLaVA-v1.5 and LLaVA-Next models.

Method	ScienceQA-IMG ↑			SEED-Bench-IMG ↑			MMBench EN V1.1 ↑			Avg. ↑
	v1.5 7B	Next 7B	Next 13B	v1.5 7B	Next 7B	Next 13B	v1.5 7B	Next 7B	Next 13B	
DuQuant	57.36%	66.34%	70.20%	54.11%	63.64%	66.15%	55.91%	**62.78%**	64.31%	62.31%
QVLM	55.24%	60.60%	65.28%	50.13%	50.38%	65.39%	57.15%	60.29%	63.06%	58.61%
QSVD	65.61%	66.10%	70.43%	58.49%	65.63%	69.21%	45.45%	50.96%	45.56%	59.72%
WSVD	64.25%	66.94%	**73.08%**	60.23%	67.49%	70.67%	56.57%	60.56%	64.68%	64.94%
LSVD	**66.14%**	**68.37%**	72.11%	**61.94%**	**67.49%**	**71.56%**	**58.66%**	61.61%	**65.08%**	**65.88%**
FP16	68.10%	69.60%	73.23%	60.18%	69.02%	71.30%	61.56%	65.14%	67.53%	67.30%

A.5.3 More Results for Ablation Studies

Effectiveness of Quantization-aware Whitening We evaluate quantization-aware whitening under the same W8A8 setting and compression ratio. The *W/o QAW* baseline removes this component while keeping the rest of LSVD unchanged. As shown in Tab. 8, QAW consistently improves accuracy across all models, raising the average from 68.52% to 68.86%. This indicates that aligning the SVD transformation with the subsequent quantized execution helps preserve accuracy under low-precision compression.

Table 8: Results of quantization-aware whitening ablation.

Method	ScienceQA-IMG ↑			Avg. ↑
	v1.5 7B	Next 7B	Next 13B	
FP16	68.10%	69.60%	73.23%	70.31%
W/o QAW	65.70%	68.07%	71.79%	68.52%
LSVD	**66.14%**	**68.34%**	**72.11%**	**68.86%**

Effectiveness of SVD-aware Permutation We evaluate the effectiveness of the SVD-aware permutation in the FFN (Sec. 3.5) under the same setting as above. The *W/o Perm.* baseline uses the same FFN SVD ratio as LSVD but does not group SVD-friendly channels before factorization. As shown in Tab. 9, removing this permutation reduces the average accuracy from 68.86% to 59.03%, with a particularly large drop on LLaVA-Next 7B from 68.34% to 50.37%. This shows that SVD-aware permutation effectively groups FFN channels that are more suitable for low-rank factorization, thereby preserving accuracy under FFN compression.

Table 9: Results of SVD-aware permutation ablation.

Method	ScienceQA-IMG ↑			Avg. ↑
	v1.5 7B	Next 7B	Next 13B	
FP16	68.10%	69.60%	73.23%	70.31%
W/o Perm.	58.80%	50.37%	67.92%	59.03%
LSVD	**66.14%**	**68.34%**	**72.11%**	**68.86%**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's contributions and scope as a novel loss-aware low-rank approximation framework for efficient VLM inference. The claimed accuracy and efficiency improvements are supported by the theoretical analysis, main experiments, and ablation studies.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation in the last section of the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper states the assumptions used in the loss-aware lo-rank approximation framework and provides the corresponding derivations and proofs in the main content and appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Yes, the paper discloses sufficient information to reproduce its main results. The methods, implementation details, and evaluation setup, including models, datasets, and calibration procedures, are well-documented.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The experiments use publicly available models and benchmarks, and the paper provides detailed experimental settings and implementation details. The code, run scripts and calibration caches are publicly available at `https://github.com/SAI-Lab-NYU/LSVD`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the models, benchmark datasets, calibration settings, compression ratios, rank-allocation settings, quantization settings, and hardware/kernel configurations used in the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, we report the average results over 5 random seeds in all evaluations. And we follow the open-source VLM evaluation toolkit to report the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All the details, type of compute workers, memory, time of execution, are provided in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and acknowledge the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive impacts such as reducing inference cost and improving accessibility of efficient VLM deployment, as well as potential risks that more efficient VLMs may also make misuse easier.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new high-risk pretrained model, image generator, or scraped dataset. It proposes a compression method evaluated on existing public models, and any use should follow the original models' licenses and usage restrictions.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the existing models, datasets, benchmarks, and codebases used in the experiments, and describes their versions, access sources, and licenses or terms of use where available.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: Our paper does not release new models or datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject studies, crowdsourcing experiments, or collection of data from participants, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only for writing, editing, and formatting assistance, and did not affect the core methodology, experiments, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.