IO-SVD: INPUT-OUTPUT WHITENED SVD FOR ADAPTIVE-RANK LLM COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models deliver strong performance across language and reasoning tasks, but their storage and compute costs remain major barriers to deployment in resource-constrained and latency-sensitive settings. SVD-based post-training compression offers a hardware-agnostic way to reduce model size and improve inference efficiency through low-rank factorization. However, existing methods often rely on input-only whitening spaces, homogeneous rank allocation, or loss-agnostic allocation heuristics, limiting their ability to preserve model quality under aggressive compression. We propose Input-Output Whitened SVD (IO-SVD), a post-training compression method that forms a KL-aware double-sided whitening space for model weights. Using a second-order expansion of the KL loss over the top-K token probabilities, IO-SVD constructs an output-side metric that captures predictive sensitivity, while input whitening captures activation statistics. We further introduce an efficient heterogeneous rank-allocation strategy that scores whitened singular components using first-order calibration loss estimates and prunes the least sensitive components under a global budget. Inspired by prior work that combines SVD truncation with quantization, we improve hybrid SVD-quantization compression through loss-aware remapping, which selects low-rank factor rows for 8-bit quantization based on the predicted loss change incurred by quantizing them. Across LLM and VLM families, we show that double-sided whitening preserves the model's predictive output distribution more closely than input-only whitening at matched ranks. This improvement is also reflected in stronger free-form generation and higher speculative-decoding acceptance, while our inference analysis characterizes the memory and throughput benefits of low-rank compression across different serving settings.

1 INTRODUCTION

Large language models (LLMs) have achieved strong performance across natural language understanding, reasoning, and code generation, but their storage and computational costs limit deployment in latency-sensitive settings such as robotics, edge computing, and interactive systems. These constraints motivate post-training compression methods that improve inference efficiency without costly retraining or changes to pretraining pipelines.

Existing compression methods include quantization, pruning, and knowledge distillation. Quantization is effective in practice (Dettmers et al., 2022; Frantar et al., 2023; Xiao et al., 2023; Chee et al., 2023; Tseng et al., 2024; Shao et al., 2024), but its efficiency often depends on specialized hardware support and optimized low-bit kernels (Park et al., 2022; Zhao et al., 2024; Lin et al., 2024; Frantar et al., 2025). Pruning removes parameters based on importance metrics (Frantar et al., 2022; Frantar & Alistarh, 2023; Ma et al., 2023; Sun et al., 2024), but can struggle to match other methods at high compression ratios (Cheng et al., 2024). Knowledge distillation transfers knowledge from a large teacher to a smaller student (Hinton et al., 2015; Gu et al., 2024; Agarwal et al., 2024), but typically requires costly gradient-based training and may degrade under large teacher–student architectural or tokenizer gaps (Rao et al., 2023; Nguyen et al., 2026; Shin et al., 2025). These limitations motivate post-training alternatives that are broadly deployable and require no retraining.

SVD-based compression offers a post-training alternative to quantization for LLMs (Hsu et al., 2022; Yuan et al., 2025; Wang et al., 2025c;b;a; Chekalina et al., 2025; Abbasi et al., 2026; Li et al., 2026). By replacing dense weight matrices with low-rank factors, it can reduce floating-point operations,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

while remaining hardware-agnostic and avoiding specialized kernels. Prior methods have progressed from weight-space or layer-output reconstruction (Hsu et al., 2022) to input-aware whitening based on activation statistics (Yuan et al., 2025; Wang et al., 2025c;b;a). However, these objectives remain local: they do not directly model how layer perturbations affect the model's predictive distribution. Recent and concurrent works on double-sided whitening incorporate both input and output statistics (Chekalina et al., 2025; Li et al., 2026); we build on this direction by deriving the output-side metric from a KL divergence to the uncompressed model's predictive distribution.

In addition to the choice of truncation space, rank allocation constitutes an orthogonal design dimension in SVD-based compression. Several approaches adopt simple or predefined allocation strategies across layers (Wang et al., 2025c; Hsu et al., 2022), without fully accounting for the heterogeneous sensitivity of different components. This can limit the effectiveness of compression, particularly at high compression ratios where careful distribution of capacity becomes critical. More recently, Dobi-SVD (Wang et al., 2025a) proposes an explicit optimization framework to determine the optimal truncation point for each linear layer. While effective, this approach relies on gradient-based optimization and can be computationally expensive and time-consuming to solve in practice. As a result, there remains a need for efficient and scalable surrogates of layer sensitivity that can guide rank allocation without incurring significant computational overhead.

Pure low-rank compression becomes increasingly lossy at aggressive compression ratios because it must discard much of the singular spectrum. Dobi-SVD mitigates this issue by combining truncation with quantization through a remapping strategy (Wang et al., 2025a). However, its heuristic selection of components for quantization is agnostic to quantization-induced error and its impact on the task objective. We therefore introduce a loss-aware remapping strategy for hybrid SVD compression.

We make three contributions:

- **KL-aware input-output whitening.** We derive a double-sided whitening objective from a second-order KL approximation, capturing both input activation geometry and output-side predictive sensitivity.
- **Loss-aware adaptive compression.** We introduce an efficient heterogeneous rank-allocation strategy that greedily removes whitened singular components with the smallest predicted calibration-loss impact under a global budget, and extend this loss-aware criterion to hybrid SVD–quantization by selecting low-rank factor rows for 8-bit quantization based on predicted quantization-induced loss.
- **Evaluation of output-distribution preservation.** We show that double-sided whitening better preserves the predictive distribution than input-only whitening at matched ranks, and demonstrate that this translates to stronger free-form generation and higher speculative-decoding acceptance. We further evaluate IO-SVD across multiple LLM and VLM families and analyze the inference behavior of low-rank compression across different serving settings.

2 Related Work

Compression of Large Language Models. Post-training LLM compression broadly includes pruning, quantization, and low-rank factorization. Pruning methods range from second-order compression in OBC (Frantar et al., 2022) and its scalable blockwise extension SparseGPT (Frantar & Alistarh, 2023) to structured LLM-Pruner (Ma et al., 2023) and activation-aware Wanda (Sun et al., 2024). Quantization methods include mixed-precision outlier handling in LLM.int8() (Dettmers et al., 2022), activation rescaling in SmoothQuant (Xiao et al., 2023), second-order weight updates in GPTQ/OPTQ (Frantar et al., 2023), and incoherence processing in QuIP (Chee et al., 2023; Tseng et al., 2024). While pruning and quantization can achieve strong compression, their speedups may depend on hardware support or specialized kernels. Low-rank factorization instead replaces dense linear layers with products of smaller matrices, reducing arithmetic cost using standard matrix multiplication kernels.

SVD-based Compression of Large Language Models. SVD-based compression is attractive because explicit low-rank factorization can reduce both storage and arithmetic cost. FWSVD (Hsu et al., 2022) uses Fisher information to weight parameters before factorization, while ASVD (Yuan et al., 2025) rescales weights using input activation statistics. SVD-LLM (Wang et al., 2025c) whitens weights using the input activation covariance and further uses LoRA-based residual recovery (Hu et al., 2022; Wang et al., 2024). SVD-LLMv2 (Wang et al., 2025b) adds heterogeneous rank allocation based on truncation-induced loss, Dobi-SVD (Wang et al., 2025a) optimizes layerwise ranks through

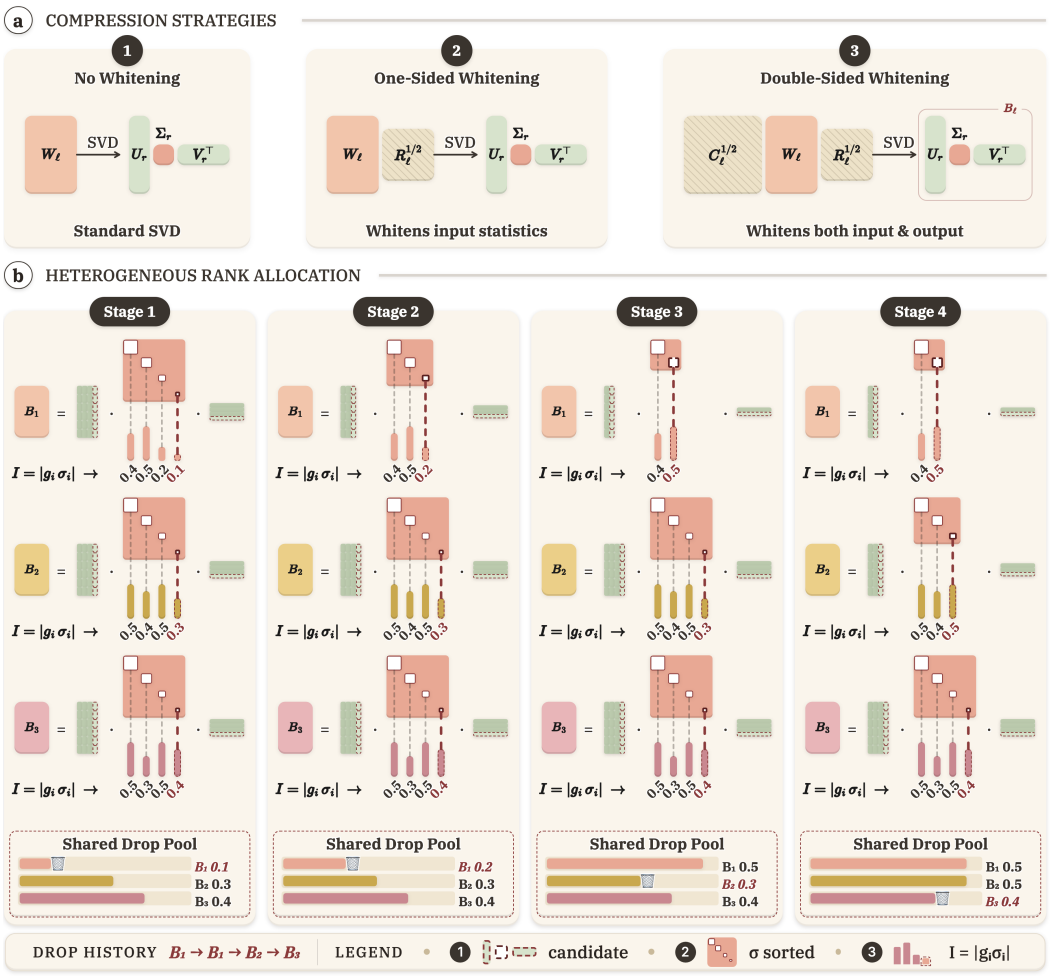


Figure 1: Overview of IO-SVD. **(a)** Comparison of whitening strategies: standard SVD reconstructs the weight directly, one-sided whitening incorporates input activation statistics, and double-sided whitening incorporates both input statistics and output-side sensitivity before SVD. **(b)** Heterogeneous rank allocation. For each whitened matrix, the smallest remaining singular component becomes its drop candidate. These candidates are placed in a shared pool and ranked by the loss-sensitivity score $I = |g_i \sigma_i|$. After a component is removed, the corresponding matrix exposes its next smallest remaining singular component. Repeating this process preserves the spectral order within each matrix while producing adaptive ranks across layers.

backpropagation, and Zero-sum SVD (Abbasi et al., 2026) removes singular components while balancing their predicted loss changes.

Most SVD-based LLM compression methods therefore use one-sided, input-aware objectives and do not explicitly model sensitivity to errors in different output directions. Double-sided objectives can instead capture both input and output-side sensitivity through second-order curvature approximations. Prior work applies this idea to SVD compression using Kronecker-factored Hessian terms (Chekalina et al., 2025), with related Kronecker-factored compression objectives studied elsewhere (Tseng et al., 2025). These approaches motivate accounting for how approximation error affects the model beyond local activation reconstruction.

IO-SVD follows this double-sided perspective. Like SVD-LLM (Wang et al., 2025c;b), it performs activation-aware post-training low-rank compression, but augments input-side whitening with output-side predictive sensitivity derived from the model's predictive distribution.

3 METHOD

IO-SVD compresses each target linear layer using a KL-aware, input-output whitened SVD objective. The method has three components: (i) a double-sided whitening space derived from the predictive KL divergence between the original and compressed models, (ii) global rank allocation based on singular-component loss sensitivity, and (iii) loss-aware remapping for hybrid SVD–quantization compression.

3.1 KL-AWARE INPUT-OUTPUT WHITENED SVD

For a target linear layer W_ℓ, let $h_t = W_\ell x_t$, $\Delta W_\ell = W_\ell - \hat{W}_\ell$, and $\delta h_t = \Delta W_\ell x_t$. Let $z_t \in \mathbb{R}^V$ denote the uncompressed logits at token t, and let $p_t = \mathrm{softmax}(z_t)$. We seek compressed parameters $\hat{\theta}$ that preserve the predictive distribution of the original model:

$$\mathcal{J}(\hat{\theta}) = \mathbb{E}_{(x,y)} \left[\sum_{t=1}^{T} \mathrm{KL}\big(p_\theta(\cdot \mid x, y_{<t}) \parallel p_{\hat{\theta}}(\cdot \mid x, y_{<t})\big) \right]. \tag{1}$$

To obtain a tractable layerwise objective, we approximate the token-level KL around the uncompressed logits with its second order Taylor expansion:

$$\mathrm{KL}(p_t \parallel \mathrm{softmax}(z_t + \delta z_t)) = \frac{1}{2} \delta z_t^\top H_t \delta z_t + O(\|\delta z_t\|^3), \qquad H_t = \mathrm{Diag}(p_t) - p_t p_t^\top. \tag{2}$$

The derivation is given in Appendix B.1. Let $J_t = \partial z_t / \partial h_t$. Using the linearization $\delta z_t \approx J_t \Delta W_\ell x_t$, we obtain

$$\Delta \mathcal{J}_{\ell,t} \approx \frac{1}{2} x_t^\top \Delta W_\ell^\top C_{\mathrm{token},t} \Delta W_\ell x_t, \qquad C_{\mathrm{token},t} = J_t^\top H_t J_t. \tag{3}$$

Averaging over calibration tokens and applying a moment-decoupling approximation yields

$$\Delta \mathcal{J}_\ell \approx \frac{1}{2} \left\| C_\ell^{1/2} (W_\ell - \hat{W}_\ell) R_\ell^{1/2} \right\|_F^2, \qquad R_\ell = \mathbb{E}_t[x_t x_t^\top], \quad C_\ell = \mathbb{E}_t[C_{\mathrm{token},t}]. \tag{4}$$

Here, R_ℓ captures input activation geometry, while C_ℓ captures output-side predictive sensitivity. The trace-to-Frobenius reduction is shown in Appendix B.2. In practice, we use damped estimates $\bar{R}_\ell = R_\ell + \lambda_R I$ and $\bar{C}_\ell = C_\ell + \lambda_C I$ when computing matrix square roots and inverse square roots; for notational simplicity, we omit the bars below.

Define the doubly whitened matrix

$$B_\ell = C_\ell^{1/2} W_\ell R_\ell^{1/2}. \tag{5}$$

With damped positive-definite estimates of R_ℓ and C_ℓ, the whitening maps are invertible and preserve rank, so the layerwise compression problem becomes

$$\min_{\mathrm{rank}(\hat{B}_\ell) \le r_\ell} \|B_\ell - \hat{B}_\ell\|_F^2, \tag{6}$$

where $\hat{B}_\ell = C_\ell^{1/2} \hat{W}_\ell R_\ell^{1/2}$. By the Eckart–Young–Mirsky theorem (Eckart & Young, 1936; Mirsky, 1960), the solution is the rank-r_ℓ truncated SVD

$$\hat{B}_\ell^\star = U_{\ell,r} \Sigma_{\ell,r} V_{\ell,r}^\top. \tag{7}$$

Undoing the whitening gives

$$\hat{W}_\ell^\star = C_\ell^{-1/2} U_{\ell,r} \Sigma_{\ell,r} \left(R_\ell^{-1/2} V_{\ell,r} \right)^\top. \tag{8}$$

Efficiently computing C_ℓ. Directly forming $C_{\mathrm{token},t} = J_t^\top H_t J_t$ is impractical because H_t and J_t scale with the vocabulary size. We therefore approximate the KL curvature on the top-K support of the uncompressed model, renormalize probabilities on this support, and accumulate C_ℓ using vector-Jacobian products with backward hooks. This avoids materializing H_t, J_t, or $J_t^\top H_t J_t$. The full top-K factorization and implementation details are given in Appendix C, and an ablation over the choice of k is given in Section 4.2.

3.2 ADAPTIVE RANK ALLOCATION

The SVD solution above assumes fixed per-layer ranks. Under a global compression budget, we instead allocate ranks by estimating the calibration-loss impact of removing individual whitened singular components.

Let $B_\ell = U_\ell \Sigma_\ell V_\ell^\top = \sum_i \sigma_{\ell,i} u_{\ell,i} v_{\ell,i}^\top$, and $\mathcal{L}$ be the calibration loss, and $G_\ell = \partial \mathcal{L} / \partial W_\ell$. The corresponding whitened gradient is

$$\widetilde{G}_\ell = C_\ell^{-1/2} G_\ell R_\ell^{-1/2}. \tag{9}$$

The first-order derivative with respect to the i-th singular value is $g_{\ell,i} = u_{\ell,i}^\top \widetilde{G}_\ell v_{\ell,i}$. Dropping this component sets $\Delta\sigma_{\ell,i} = -\sigma_{\ell,i}$, so we score its importance by

$$I_{\ell,i} = |g_{\ell,i} \sigma_{\ell,i}|. \tag{10}$$

The derivation of this score is provided in Appendix D.

We greedily remove the lowest-scoring eligible tail component across all layers until the target global parameter budget is reached. After a component is removed from a layer, the next-smallest retained component from that layer becomes eligible, preserving the truncated-SVD structure. The budget is tracked using the layer-specific storage gain defined in Appendix A. This procedure induces heterogeneous ranks $\{r_\ell\}$ without iterative rank optimization.

3.3 LOSS-AWARE REMAPPING FOR HYBRID COMPRESSION

Standard SVD compression ties storage reduction to rank reduction: a rank-k factorization of $W \in \mathbb{R}^{m \times n}$ stores $k(m+n)$ parameters, so aggressive parameter budgets force aggressive spectral truncation. Dobi-SVD (Wang et al., 2025a) mitigates this issue by remapping low-rank factors and storing selected rows in low precision. However, its row-selection rule is fixed and structural, rather than based on quantization error or loss impact.

We replace this fixed rule with a loss-aware row-selection criterion. After SVD truncation, write each compressed module as

$$\hat{W}_\ell = A_\ell D_\ell^\top, \tag{11}$$

where A_ℓ and D_ℓ are the retained low-rank factors. Let $r_{\ell,i}$ denote a candidate row from either factor, and let $\mathcal{Q}_8(\cdot)$ be the 8-bit quantize-dequantize operator. Quantizing this row induces

$$\Delta r_{\ell,i} = \mathcal{Q}_8(r_{\ell,i}) - r_{\ell,i}. \tag{12}$$

Using calibration data, we compute the row gradient $\gamma_{\ell,i} = \frac{\partial \mathcal{L}_{\text{cal}}}{\partial r_{\ell,i}}$, and score the predicted quantization-induced loss by $s_{\ell,i} = |\langle \gamma_{\ell,i}, \mathcal{Q}_8(r_{\ell,i}) - r_{\ell,i} \rangle|$.

Given the remaining compression budget after SVD truncation, we place all candidate rows into a shared pool and greedily select rows with the smallest predicted loss cost, or cost per saved parameter when row sizes differ: $\sum_{(\ell,i)\in\mathcal{S}} c_{\ell,i} \geq C_{\text{rem}}$, and $C_{\text{rem}} = \max\{0, C_{\text{target}} - C_{\text{svd}}\}$.

Rows in $\mathcal{S}$ are stored in 8-bit precision, along with their indices. This preserves the storage benefit of Dobi-SVD-style remapping while making row selection sensitive to both quantization error and task-loss impact. Additional details are given in Appendix E. As in Dobi-SVD and ZS-SVD, remapping reduces checkpoint size only: the factors are restored to FP16 once at load time, so it does not reduce inference-time GPU memory or computation, and the one-time loading overhead is at most 3.4% (Appendix E.1).

4 EXPERIMENTS

To evaluate both generative modeling quality and task-level behavior, we report perplexity on WikiText2 (Merity et al., 2017), Penn Treebank (Marcus et al., 1993), and C4 (Raffel et al., 2020), as well as zero-shot accuracy on OpenBookQA (Mihaylov et al., 2018), ARC-Easy, ARC-Challenge (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), and MathQA (Amini et al., 2019). Following the standard setup in prior SVD-based LLM compression work, we apply compression to the primary transformer linear projections, including the query, key, value, and output attention projections, as well as the MLP layers. For calibration, we use 256 randomly sampled WikiText2 sequences with sequence length 2048, consistent with prior studies (Wang et al., 2025c;a; Abbasi et al., 2026). A calibration time analysis is provided in Appendix H.

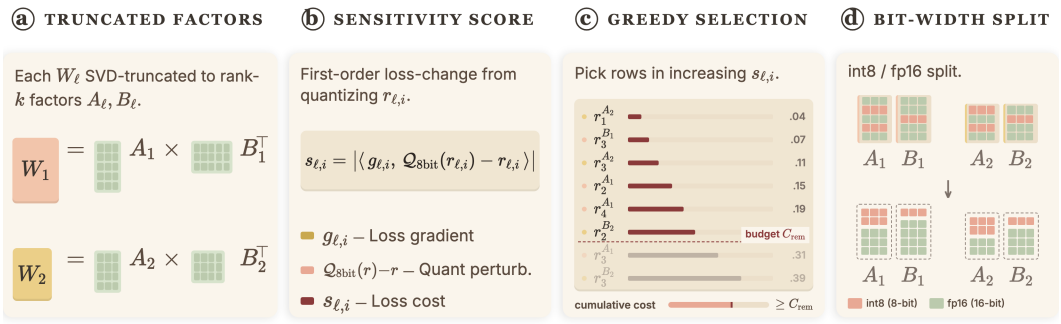


Figure 2: Loss-aware remapping: **(a)** SVD-truncate each weight to rank k; **(b)** score factor rows by first-order calibration-loss change under int8 quantization; **(c)** greedily keep low-score rows in int8 until meeting C_{rem}; **(d)** assign the remaining rows to fp16.

Table 1: IO-SVD vs. SVD-based compression methods on LLaMA-7B across maintenance ratios (0.8/0.6/0.4). Lower is better for PPL; best is in bold. (*) indicates results with Dobi-SVD-style remapping enabled. (‡) denotes our loss-aware remapping. (†) denotes HQ (Half-prune+Quant), which we use in place of remapping at pruning $\geq 50\%$.

Ratio	Method	PPL ↓			Acc ↑							Avg. ↑
		Wiki2	PTB	C4	Openb.	ARC_e	ARC_c	WinoG.	HellaS.	PIQA	MathQA	
1.0	Baseline	5.68	8.35	7.34	0.34	0.75	0.42	0.70	0.57	0.79	0.28	0.55
0.8	ASVD	11.14	16.55	15.93	0.25	0.53	0.27	0.64	0.41	0.68	0.24	0.43
	SVD-LLM	7.94	16.22	15.84	0.22	0.58	0.29	0.63	0.43	0.69	0.24	0.44
	Dobi-SVD	8.54	14.83	10.01	0.26	0.59	0.31	0.66	0.44	0.70	0.23	0.46
	ZS-SVD	6.74	11.87	10.74	0.31	0.70	0.37	0.68	0.49	0.74	0.24	0.50
	IO-SVD	6.41	10.93	9.82	0.30	0.71	0.35	0.69	0.50	0.74	0.24	0.50
	Dobi-SVD*	6.08	15.39	7.83	0.27	0.65	0.37	0.68	0.54	0.77	0.27	0.51
	ZS-SVD*	5.90	8.81	7.95	0.35	0.74	0.41	0.70	0.56	0.78	0.26	0.54
	IO-SVD‡	5.59	8.56	7.62	0.34	0.75	0.40	0.70	0.56	0.78	0.26	0.54
0.6	ASVD	1407	3292	1109	0.13	0.28	0.22	0.48	0.26	0.55	0.19	0.30
	SVD-LLM	13.11	63.75	49.83	0.19	0.42	0.25	0.58	0.33	0.60	0.21	0.37
	Dobi-SVD	13.54	46.38	23.54	0.22	0.41	0.27	0.58	0.34	0.61	0.23	0.38
	ZS-SVD	11.44	43.19	34.13	0.23	0.52	0.26	0.62	0.35	0.64	0.22	0.41
	IO-SVD	9.84	28.84	27.15	0.22	0.55	0.25	0.62	0.37	0.64	0.22	0.41
	Dobi-SVD*	8.12	43.85	12.63	0.28	0.65	0.32	0.62	0.45	0.72	0.25	0.47
	ZS-SVD*	6.96	12.72	11.52	0.32	0.71	0.36	0.68	0.48	0.74	0.24	0.50
	IO-SVD‡	6.27	10.89	10.15	0.31	0.69	0.35	0.68	0.50	0.74	0.26	0.50
0.4	ASVD	57057	45218	43036	0.12	0.26	0.21	0.49	0.26	0.53	0.18	0.29
	SVD-LLM	53.74	438.58	345.49	0.14	0.28	0.22	0.50	0.27	0.55	0.21	0.31
	Dobi-SVD	46.18	238.91	190.62	0.15	0.31	0.20	0.52	0.28	0.54	0.22	0.32
	ZS-SVD	45.17	334.85	212.57	0.15	0.31	0.21	0.54	0.28	0.55	0.21	0.32
	IO-SVD	27.70	189.53	136.42	0.13	0.31	0.19	0.53	0.28	0.55	0.21	0.31
	Dobi-SVD*	9.95	67.62	17.94	0.23	0.52	0.24	0.56	0.38	0.65	0.23	0.40
	ZS-SVD†	6.73	11.78	10.69	0.29	0.72	0.38	0.68	0.49	0.75	0.25	0.51
	IO-SVD†	6.41	10.82	9.79	0.30	0.70	0.36	0.67	0.50	0.75	0.25	0.51

4.1 RESULTS

Table 1 compares IO-SVD against SVD-based compression baselines, including ASVD (Yuan et al., 2025), SVD-LLM (Wang et al., 2025c), Dobi-SVD (Wang et al., 2025a), and ZS-SVD (Abbasi et al., 2026), under maintenance ratios of 80%, 60%, and 40%. Across these settings, IO-SVD consistently improves over the baselines on perplexity, both on in-distribution evaluation and out-of-distribution datasets. On downstream zero-shot commonsense reasoning tasks, IO-SVD also achieves comparable or better accuracy than prior methods.

The results below the horizontal bar in Table 1 report remapping comparisons, where vanilla remapped variants of existing methods are compared against our loss-aware remapping strategy (denoted by markers in the table). Consistent with the observation in ZS-SVD (Abbasi et al., 2026), for pruning ratios above 50% (0.4 ratio in Table 1 $\rightarrow$ 60% pruning), we use half-prune quantization (HQ): first

Table 2: Equal-rank whitening, LLaMA-7B.

Ratio	Whitening	Output KL ↓	Mean rel-MSE ↓
0.8	Input-only	0.303	**0.0371**
	Double-sided	**0.226**	0.0396
0.6	Input-only	0.827	**0.0674**
	Double-sided	**0.619**	0.0718

Table 3: Free-form generation on the full GSM8K test set (pass@k, %, ↑) for Llama-3.1-8B-Instruct at a retained ratio of 0.8, calibrated on 1,024 OpenOrca sequences.

Variant	Method	pass@1	pass@2	pass@4	pass@8
Uncompressed	N/A	78.05	86.73	92.00	94.92
Non-remap	IO-SVD	**20.59**	**28.73**	**37.31**	**45.41**
	ZS-SVD	5.20	8.74	13.45	19.11
Vanilla remap	IO-SVD	**60.28**	**70.94**	**78.71**	**84.23**
	ZS-SVD	45.89	58.41	68.41	76.12
Loss-aware remap	IO-SVD	**59.51**	**71.24**	**79.62**	**85.67**
	ZS-SVD	48.55	60.80	70.49	78.09

Table 4: Speculative-decoding acceptance.

Ratio	IO-SVD	ZS-SVD
0.8	**0.8700**	0.8344
0.6	**0.7025**	0.6362

applying SVD truncation at twice the target maintenance ratio, then using 8-bit quantization to reach the final budget. A more detailed ablation over remapping strategies is provided in Section 4.2.

We further evaluate IO-SVD on vision-language models (LLaVA-1.5 7B/13B and SmolVLM 2B); these results are reported in Appendix F.

4.1.1 Distinction from ZS-SVD and output distribution preservation

Among the SVD baselines, ZS-SVD (Abbasi et al., 2026) is the closest comparison to IO-SVD. A key distinction is that IO-SVD introduces output sensitivity through KL-based double-sided whitening. To isolate this effect from rank allocation, we assign identical homogeneous ranks to input-only and double-sided whitening. As shown in Table 2, double-sided whitening reduces output KL by approximately 25% at both compression ratios, while incurring slightly higher local reconstruction error. This shows that IO-SVD prioritizes preserving the predictive distribution rather than minimizing local activation reconstruction alone.

Perplexity and multiple-choice accuracy do not directly evaluate free-form generation. Perplexity measures next-token likelihood, while multiple-choice benchmarks select among fixed candidate answers. We therefore evaluate Llama-3.1-8B-Instruct (Grattafiori et al., 2024) on the full GSM8K test set (Cobbe et al., 2021), using 1,024 OpenOrca (Lian et al., 2023) calibration sequences. Each problem is sampled eight times and evaluated using pass@k.

As shown in Table 3, the gap between IO-SVD and ZS-SVD is substantially larger on free-form generation than on the multiple-choice benchmarks in Table 1. Without remapping, IO-SVD improves pass@1 from 5.20 to 20.59 and pass@8 from 19.11 to 45.41, with consistent gains under both vanilla and loss-aware remapping. These results suggest that lower output KL leads to better preservation of generative behavior during autoregressive decoding.

We further evaluate this effect through speculative decoding (Leviathan et al., 2023; Chen et al., 2023), using WikiText-2-calibrated LLaMA-7B as the compressed draft model and the uncompressed model as the verifier over 204,700 held-out token positions. IO-SVD achieves higher acceptance rates at both compression ratios, reaching 0.8700 versus 0.8344 at ratio 0.8 and 0.7025 versus 0.6362 at ratio 0.6 (Table 4). Together, these results show that the output-aware whitening objective preserves the predictive distribution more effectively than is reflected by perplexity and multiple-choice accuracy alone.

Tables 5 and 7 further evaluate IO-SVD across architectures and compression settings. At 20% pruning, IO-SVD achieves the lowest perplexity on OPT-6.7B, Vicuna-7B, and LLaMA-13B while maintaining competitive average commonsense accuracy. On LLaMA-2-7B, vanilla IO-SVD remains competitive with prior SVD methods, while loss-aware remapping improves average accuracy over pruning baselines.

4.2 Ablation studies

Effect of top-k: Figure 3 evaluates the top-K value used to construct the output-side KL curvature. We select K on WikiText-2 and evaluate the resulting models on WikiText-2, PTB, and C4 using normalized perplexity. The best K selected on WikiText-2 transfers well across datasets, indicating that the choice generalizes across evaluation sets.

Table 5: Perplexity (PPL, ↓) on WikiText-2 and average accuracy (Acc, ↑) on six commonsense reasoning datasets (excluding arc_c) for OPT-6.7B, Vicuna-7B, and LLaMA-13B at 20% pruning.

	OPT-6.7B		Vicuna-7B		LLaMA-13B	
Method	**PPL**	**Acc**	**PPL**	**Acc**	**PPL**	**Acc**
Original	10.86	0.52	6.78	0.56	5.09	0.59
SVD	66275	0.03	18644	0.05	946.31	0.21
FWSVD	14559	0.06	2758	0.09	15.98	0.43
ASVD	82.00	0.32	16.23	0.33	6.74	0.54
SVDLLM	16.04	0.41	8.41	0.51	6.43	0.55
ZS-SVD	11.40	0.51	8.08	**0.54**	5.84	0.56
IO-SVD	**11.10**	**0.51**	**7.36**	0.53	**5.60**	**0.56**

Table 6: Ablation over whitening curvature and rank allocation on LLaMA-7B, calibrated on WikiText2. WikiText2 PPL across pruning ratios; lower is better.

Curvature	**Het. rank?**	**Ratio 0.8**	**Ratio 0.6**	**Ratio 0.4**
Input-only (SVD-LLM)	✗	7.95	13.11	53.74
Double-sided (OBD-LLM)	✗	7.36	11.34	32.95
Double-sided (Ours)	✗	7.31	11.20	32.09
Input-only (SVD-LLM)	✓	6.72	11.65	62.76
Double-sided (OBD-LLM)	✓	6.45	9.90	28.19
Double-sided (Ours)	✓	**6.41**	**9.84**	**27.70**

Table 7: Commonsense reasoning results on LLaMA-2-7B. IO-SVD and its remapped variants are compared with pruning and SVD-based baselines at compression ratios 0.6 and 0.4, higher is better.

Ratio	**Method**	**Acc ↑**					**Avg ↑**
		PIQA	**Hell**	**Win**	**ARC_e**	**ARC_c**	
1.0	Baseline	0.78	0.57	0.69	0.76	0.43	0.65
0.6	LLM-Pruner	0.70	0.41	0.53	0.53	0.27	0.48
	SliceGPT	0.65	0.57	0.60	0.43	0.32	0.51
	Bonsai	0.72	0.45	0.58	0.59	0.30	**0.53**
	Wanda-sp	0.70	0.42	0.53	0.57	0.29	0.50
	SVD-LLM	0.56	0.30	0.57	0.39	0.21	0.41
	ZS-SVD	0.63	0.34	0.60	0.46	0.25	0.45
	IO-SVD	0.61	0.33	0.59	0.51	0.23	0.45
	Dobi-SVD*	0.72	0.45	0.64	0.67	0.31	0.56
	ZS-SVD*	0.72	0.46	0.67	0.66	0.33	0.57
	IO-SVD‡	0.74	0.47	0.67	0.73	0.38	**0.60**
0.4	SVD-LLM	0.54	0.27	0.48	0.26	0.20	0.35
	ZS-SVD	0.54	0.27	0.52	0.29	0.19	0.36
	IO-SVD	0.54	0.28	0.52	0.30	0.19	**0.37**
	Dobi-SVD*	0.67	0.38	0.57	0.55	0.26	0.49
	ZS-SVD†	0.73	0.48	0.68	0.70	0.36	**0.59**
	IO-SVD†	0.73	0.48	0.65	0.71	0.37	**0.59**

Whitening space and rank allocation: Table 6 compares input-only, double-sided Kronecker, and double-sided KL whitening under homogeneous and heterogeneous rank allocation. Heterogeneous allocation improves performance in most settings, while double-sided whitening consistently outperforms input-only whitening. Combining heterogeneous allocation with our KL whitening achieves the lowest perplexity across all reported compression settings.

Remapping applied to our method: Table 8 compares vanilla compression, standard remapping, and our loss-aware remapping across SVD-LLM, ZS-SVD, and IO-SVD. Consistent with prior work (Wang et al., 2025a; Abbasi et al., 2026), remapping improves over vanilla compression, while loss-aware remapping further improves performance in most settings by selecting rows according to predicted calibration-loss impact.

4.2.1 Inference speed and memory

We evaluate inference efficiency using FP16 weights and caches on a single NVIDIA RTX PRO 6000 Blackwell Max-Q GPU. Remapping and int8 quantization are disabled, so the reported results reflect only low-rank weight factorization and, when enabled, compressed K/V caching. Remapping serves only as a compact checkpoint format: remapped factors are restored to FP16 once at load time, adding at most 3.4% to the cold-load time (Table 10 in Appendix E.1).

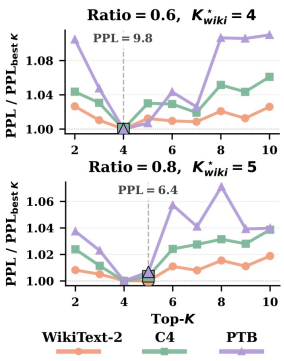


Figure 3: Top-K ablation for output-side KL curvature. Normalized perplexity on wiki2, C4, PTB

Table 8: Effect of standard and loss-aware remapping on LLaMA-7B. Perplexity on wiki2, C4, and PTB at compression ratios 0.8 and 0.6, lower is better.

Method	**Quant mode**	**Ratio 0.8**			**Ratio 0.6**		
		Wiki↓	**C4↓**	**PTB↓**	**Wiki↓**	**C4↓**	**PTB↓**
SVD-LLM	compressed	7.94	15.84	16.22	13.11	49.83	63.75
	+ remap*	5.86	7.82	8.82	6.98	11.59	12.88
	+ loss-aware‡	**5.66**	**7.78**	**8.71**	**6.69**	**11.39**	**12.46**
ZS-SVD	compressed	6.74	10.74	11.87	11.44	34.13	43.19
	+ remap*	5.90	7.95	8.81	6.96	11.52	**12.72**
	+ loss-aware‡	**5.69**	**7.92**	**8.78**	**6.69**	**11.46**	12.80
IO-SVD (Ours)	compressed	6.41	9.82	10.93	9.84	27.15	28.84
	+ remap*	5.76	**7.61**	8.59	6.48	10.24	10.95
	+ loss-aware‡	**5.59**	7.62	**8.56**	**6.27**	**10.15**	**10.89**

For compressed K/V projections, we cache their lower-dimensional latent representations rather than the full keys and values. This reduces KV-cache memory and the amount of data read during decoding, but requires additional computation to operate on the compressed representation at each decoding step. Full details of the latent projections and RoPE handling are provided in Appendix G.

This tradeoff depends strongly on the serving setting (Table 9). At batch size 1, weight compression improves throughput, while the additional computation from compressed K/V caching removes this gain. In contrast, at $B = 64$ with 1024-token prompt and generation lengths, decoding becomes more bandwidth-bound, and the reduction in cache traffic outweighs the added computation. Throughput then increases from 473.3 tokens/s for the dense model to 2041.2 tokens/s with compressed K/V caching. In the same setting, KV-cache memory decreases from 64.0 GB to 11.6 GB and peak GPU memory from 77.5 GB to 23.0 GB (Figure 4).

Table 9: Decode throughput (tokens/s) on LLaMA-2-7B compressed with IO-SVD at 60% pruning (no remapping). B: batch size; $a \rightarrow b$: prompt and generation lengths.

Setting	Dense FP16	Weight only	Weight + K/V cache
B=1, 128 → 2048	70.3	**96.9**	68.4
B=1, 4096 → 128	53.2	**70.5**	51.9
B=64, 1024 → 1024	473.3	483.6	**2041.2**

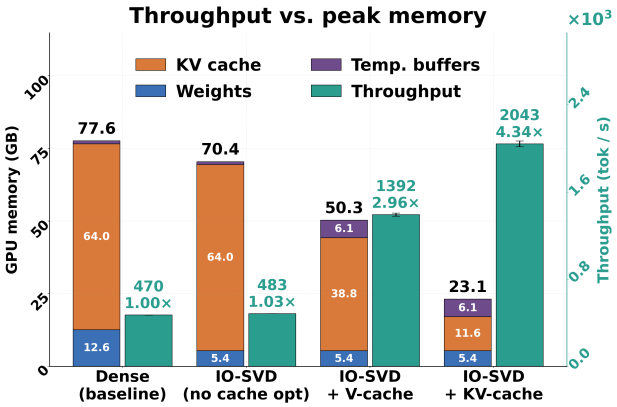


Figure 4: Peak GPU memory and decode throughput on LLaMA-2-7B at B=64, 1024 → 1024.

We emphasize that compressed K/V caching is not specific to IO-SVD and can be applied to any method with low-rank K/V projections. Our analysis therefore studies a general inference benefit of low-rank compression, while IO-SVD focuses on preserving model quality at lower ranks.

5 Conclusion and Limitations

We presented IO-SVD, a post-training compression method that uses KL-aware double-sided whitening, adaptive rank allocation, and loss-aware remapping to improve SVD-based LLM and VLM compression. Experiments show that IO-SVD better preserves perplexity and downstream accuracy than prior SVD-based methods, especially under aggressive compression. Future work should study calibration sensitivity, improve curvature-estimation efficiency, and evaluate IO-SVD across broader models, tasks, and deployment settings.

IO-SVD has several limitations that suggest useful future directions. First, the method estimates output-side curvature using a top-K token approximation, which may not fully capture sensitivity from the long tail of the vocabulary. Second, the adaptive rank allocation is based on a greedy component-removal strategy; while efficient, it may not always find the globally optimal rank distribution. Finally, IO-SVD is evaluated up to 13B-parameter models, so further validation on larger-scale models would strengthen the evidence for scalability.

AI use statement

Generative AI tools were used during manuscript preparation for drafting and revising text, improving clarity and organization, assisting with code debugging and LaTeX formatting, generating visualization scripts, organizing experimental observations, and refining figure and table descriptions. They were not used to generate experimental data, alter reported numerical results, perform the primary literature review, or determine scientific conclusions. Human-subject interaction, participant communication, and qualitative interview analysis are not applicable to this work. All AI-assisted material was reviewed by the authors and treated as draft content. The authors take full responsibility for the final manuscript, including its claims, results, and accompanying artifacts.

Reproducibility statement

We provide detailed descriptions of the proposed method, experimental protocol, and implementation in the main text and appendix. These include the evaluated model checkpoints and datasets, baseline configurations, calibration procedure, compression settings, inference setup and hardware, throughput and memory measurements, and ablation studies. The models and benchmarks used in our experiments are publicly available. Our implementation and evaluation scripts will be released publicly upon acceptance.

References

Ali Abbasi, Chayne Thrash, Haoran Qin, Shansita Sharma, Sepehr Seifi, and Soheil Kolouri. Zero sum svd: Balancing loss sensitivity for low rank llm compression, 2026. URL https://arxiv.org/abs/2602.02848.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2357–2367, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1245. URL https://aclanthology.org/N19-1245/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xrk9g5vcXR.

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, 2025. URL https://arxiv.org/abs/2505.17974.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10558–10578, 2024.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

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

Elias Frantar and Dan Alistarh. Sparsegpt: massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: a framework for accurate post-training quantization and pruning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Elias Frantar, Roberto L Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. Marlin: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pp. 239–251, 2025.

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.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uPv9Y3gmAI5.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023.

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*, pp. 13299–13308, 2024.

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

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. *Hugging Face dataset repository*, 2023.

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. *arXiv preprint arXiv:2405.04532*, 2024.

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.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. *Advances in Neural Information Processing Systems*, 36:21702–21720, 2023.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, 1960.

Truong Nguyen, Phi Van Dat, Ngan Nguyen, Linh Ngo Van, Trung Le, and Thanh Hong Nguyen. Ctpd: Cross tokenizer preference distillation. *arXiv preprint arXiv:2601.11865*, 2026.

Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. *arXiv preprint arXiv:2206.09557*, 2022.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.

Jun Rao, Xv Meng, Liang Ding, Shuhan Qi, Xuebo Liu, Min Zhang, and Dacheng Tao. Parameter-efficient and student-friendly knowledge distillation. *IEEE Transactions on Multimedia*, 26: 4230–4241, 2023.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

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. In *The Twelfth International Conference on Learning Representations*, 2024.

Haebin Shin, Lei Ji, Xiao Liu, and Yeyun Gong. Overcoming vocabulary mismatch: Vocabulary-agnostic teacher guided language modeling. In *Forty-second International Conference on Machine Learning*, 2025.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: even better llm quantization with hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Albert Tseng, Zhaofeng Sun, and Christopher De Sa. Model-preserving adaptive rounding, 2025. URL https://arxiv.org/abs/2505.22988.

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.

Qinsi Wang, Jinghan Ke, Masayoshi Tomizuka, Kurt Keutzer, and Chenfeng Xu. Dobi-svd: Differentiable svd for llm compression and some new perspectives. In *International Conference on Learning Representations*, volume 2025, pp. 12561–12590, 2025a.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. SVD-LLM v2: Optimizing singular value truncation for large language model compression. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4287–4296, Albuquerque, New Mexico, April 2025b. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.217. URL https://aclanthology.org/2025.naacl-long.217/.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=LNYIUouhdt.

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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025d.

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 *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models, 2025. URL https://arxiv.org/abs/2312.05821.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209, 2024.

APPENDIX

A ADAPTIVE-RANK IO-SVD ALGORITHM

Algorithm 2 summarizes the adaptive-rank IO-SVD truncation procedure. For each target layer, IO-SVD constructs the doubly whitened matrix

$$B_\ell = C_\ell^{1/2} W_\ell R_\ell^{1/2},$$

computes its SVD, and scores the first-order calibration-loss sensitivity of each whitened singular component. The algorithm maintains a shared pool containing one candidate from each layer: the smallest currently retained singular component of that layer. At each step, it removes the candidate with the smallest predicted loss impact while preserving the per-layer spectral order. This induces heterogeneous ranks across layers under a single global storage budget.

A practical detail is that low-rank factorization is storage-efficient only below a layer-dependent threshold. For a dense weight $W_\ell \in \mathbb{R}^{m_\ell \times n_\ell}$, the dense representation uses $m_\ell n_\ell$ parameters, while a rank-r factorization uses $r(m_\ell + n_\ell)$ parameters. We define

$$r_\ell^\star := \left\lfloor \frac{m_\ell n_\ell}{m_\ell + n_\ell} \right\rfloor, \tag{13}$$

the largest rank at which the low-rank representation is no more expensive than the dense weight. Above this threshold, storing two low-rank factors would require more parameters than keeping the original dense module.

Let r denote the current rank before removing the tail singular component. The storage gain from dropping this component is

$$\text{storage_gain}_\ell(r) = \begin{cases} 0, & r > r_\ell^\star + 1, \\ m_\ell n_\ell - r_\ell^\star(m_\ell + n_\ell), & r = r_\ell^\star + 1, \\ m_\ell + n_\ell, & r \le r_\ell^\star. \end{cases} \tag{14}$$

When $r > r_\ell^\star + 1$, the layer remains cheaper to store densely even after the drop, so the realized storage gain is zero. When $r = r_\ell^\star + 1$, the next drop moves the layer to rank $r_\ell^\star$, allowing the dense weight to be replaced by low-rank IO-SVD factors and yielding the one-time gain $m_\ell n_\ell - r_\ell^\star(m_\ell + n_\ell)$. Once the layer is already below the threshold, each additional rank reduction removes one column-row pair from the two factors and saves $m_\ell + n_\ell$ parameters.

At the end of truncation, if a layer remains above the threshold rank, we keep the original dense module, i.e., $\hat{W}_\ell \leftarrow W_\ell$. This avoids introducing approximation error when the low-rank representation would not provide storage savings.

Algorithm 1 IO-SVD initialization

Input: weights $\{W_\ell\}_{\ell=1}^{L}$, calibration set $\mathcal{D}_{\mathrm{cal}}$, pruning ratio π, minimum-rank ratio η
Output: per-layer state $\mathcal{S}$, min-heap $\mathcal{Q}$, target removed budget B_{rm}, removed budget b
Set $B_{\mathrm{rm}} \leftarrow \pi \sum_{\ell=1}^{L} m_\ell n_\ell$, $b \leftarrow 0$, and initialize $\mathcal{Q} \leftarrow \emptyset$
for $\ell = 1$ **to** L **do**
  Estimate R_ℓ and C_ℓ on $\mathcal{D}_{\mathrm{cal}}$
  Form $B_\ell \leftarrow C_\ell^{1/2} W_\ell R_\ell^{1/2}$ and compute $B_\ell = U_\ell \Sigma_\ell V_\ell^\top$
  Sort singular components in descending order of $\sigma_{\ell,i}$, with corresponding columns of U_ℓ and V_ℓ
  Compute $G_\ell = \partial \mathcal{L}_{\mathrm{cal}} / \partial W_\ell$ and $\widetilde{G}_\ell \leftarrow C_\ell^{-1/2} G_\ell R_\ell^{-1/2}$
  Set $r_\ell \leftarrow r_\ell^{\max} = \min(m_\ell, n_\ell)$
  Set $r_\ell^\star \leftarrow \lfloor m_\ell n_\ell / (m_\ell + n_\ell) \rfloor$
  Set $r_\ell^{\min} \leftarrow \lceil \eta r_\ell^\star \rceil$
  for $i = 1$ **to** $r_\ell^{\max}$ **do**
    $g_{\ell,i} \leftarrow u_{\ell,i}^\top \widetilde{G}_\ell v_{\ell,i}$
    $I_{\ell,i} \leftarrow |g_{\ell,i} \sigma_{\ell,i}|$
  end for
  if $r_\ell > r_\ell^{\min}$ **then**
    $\Delta b_{\ell,r_\ell} \leftarrow \mathrm{storage_gain}_\ell(r_\ell)$
    Push $(I_{\ell,r_\ell}, \ell, r_\ell, \Delta b_{\ell,r_\ell})$ into $\mathcal{Q}$
  end if
end for

Algorithm 2 Adaptive-rank IO-SVD truncation

Input: per-layer state $\mathcal{S}$, min-heap $\mathcal{Q}$, target removed budget B_{rm}, removed budget b
Output: compressed weights $\{\hat{W}_\ell\}_{\ell=1}^{L}$
while $b < B_{\mathrm{rm}}$ **and** $\mathcal{Q} \neq \emptyset$ **do**
  Pop $(I_{\ell,i}, \ell, i, \Delta b_{\ell,i})$ with the smallest score from $\mathcal{Q}$
  Drop the current tail singular component of layer ℓ
  Update $b \leftarrow b + \Delta b_{\ell,i}$ and $r_\ell \leftarrow r_\ell - 1$
  if $r_\ell > r_\ell^{\min}$ **then**
    $\Delta b_{\ell,r_\ell} \leftarrow \mathrm{storage_gain}_\ell(r_\ell)$
    Push $(I_{\ell,r_\ell}, \ell, r_\ell, \Delta b_{\ell,r_\ell})$ into $\mathcal{Q}$
  end if
end while
for $\ell = 1$ **to** L **do**
  if $r_\ell > r_\ell^\star$ **then**
    Keep dense module: $\hat{W}_\ell \leftarrow W_\ell$
  else
    $\hat{B}_\ell \leftarrow U_{\ell,1:r_\ell} \Sigma_{\ell,1:r_\ell} V_{\ell,1:r_\ell}^\top$
    $\hat{W}_\ell \leftarrow C_\ell^{-1/2} \hat{B}_\ell R_\ell^{-1/2}$
  end if
end for

B Objective derivation

B.1 KL expansion and output-side curvature

We derive the second-order approximation used in Eq. (2). For a fixed token t, let

$$p_t = \mathrm{softmax}(z_t)$$

be the predictive distribution of the uncompressed model, and define

$$f_t(z) := \mathrm{KL}(p_t \,\|\, \mathrm{softmax}(z)) \,. \tag{15}$$

Let $q(z) = \mathrm{softmax}(z)$. Then

$$f_t(z) = \sum_{i=1}^{V} p_{t,i} \log p_{t,i} - \sum_{i=1}^{V} p_{t,i} \log q_i(z). \tag{16}$$

Using

$$\log q_i(z) = z_i - \log \sum_{j=1}^{V} e^{z_j},$$

we obtain

$$f_t(z) = \sum_{i=1}^{V} p_{t,i} \log p_{t,i} - p_t^\top z + \log \sum_{j=1}^{V} e^{z_j}. \tag{17}$$

Therefore,

$$\nabla_z f_t(z) = q(z) - p_t. \tag{18}$$

At $z = z_t$, $q(z_t) = p_t$, so

$$\nabla_z f_t(z_t) = 0. \tag{19}$$

Since $f_t(z_t) = \mathrm{KL}(p_t \,\|\, p_t) = 0$, both the zeroth- and first-order terms vanish.

The Hessian is the softmax Jacobian:

$$\nabla_z^2 f_t(z) = \mathrm{Diag}(q(z)) - q(z)q(z)^\top. \tag{20}$$

Evaluating at $z = z_t$ gives

$$H_t = \nabla_z^2 f_t(z_t) = \mathrm{Diag}(p_t) - p_t p_t^\top. \tag{21}$$

Thus, for a logit perturbation δz_t,

$$\mathrm{KL}(p_t \,\|\, \mathrm{softmax}(z_t + \delta z_t)) = \frac{1}{2} \delta z_t^\top H_t \delta z_t + O(\|\delta z_t\|^3). \tag{22}$$

The matrix H_t is symmetric positive semidefinite. For any $v \in \mathbb{R}^V$,

$$v^\top H_t v = \sum_{i=1}^{V} p_{t,i} v_i^2 - \left(\sum_{i=1}^{V} p_{t,i} v_i \right)^2 \geq 0, \tag{23}$$

which is the variance of v_i under p_t. Also, $H_t \mathbf{1} = 0$, reflecting the invariance of the softmax distribution to adding a constant to all logits.

B.2 Trace-to-Frobenius reduction

Starting from Eq. (3),

$$\Delta \mathcal{J}_{\ell,t} \approx \frac{1}{2} x_t^\top \Delta W_\ell^\top C_{\mathrm{token},t} \Delta W_\ell x_t, \tag{24}$$

we use $a^\top M a = \mathrm{tr}(M a a^\top)$ to write

$$x_t^\top \Delta W_\ell^\top C_{\mathrm{token},t} \Delta W_\ell x_t = \mathrm{tr}\left(\Delta W_\ell x_t x_t^\top \Delta W_\ell^\top C_{\mathrm{token},t} \right). \tag{25}$$

Averaging over calibration tokens gives

$$\Delta \mathcal{J}_\ell \approx \frac{1}{2} \mathbb{E}_t \left[\mathrm{tr}\left(\Delta W_\ell x_t x_t^\top \Delta W_\ell^\top C_{\mathrm{token},t} \right) \right]. \tag{26}$$

Using the moment-decoupling approximation,

$$R_\ell = \mathbb{E}_t[x_t x_t^\top], \qquad C_\ell = \mathbb{E}_t[C_{\mathrm{token},t}], \tag{27}$$

we obtain

$$\Delta \mathcal{J}_\ell \approx \frac{1}{2} \mathrm{tr}\left(\Delta W_\ell R_\ell \Delta W_\ell^\top C_\ell \right). \tag{28}$$

Let

$$M = C_\ell^{1/2} \Delta W_\ell R_\ell^{1/2}.$$

Then

$$\|M\|_F^2 = \mathrm{tr}(MM^\top) \tag{29}$$

$$= \mathrm{tr}\left(C_\ell^{1/2} \Delta W_\ell R_\ell \Delta W_\ell^\top C_\ell^{1/2}\right) \tag{30}$$

$$= \mathrm{tr}\left(\Delta W_\ell R_\ell \Delta W_\ell^\top C_\ell\right), \tag{31}$$

where the last equality uses cyclic invariance of trace. Therefore,

$$\Delta \mathcal{J}_\ell \approx \frac{1}{2} \left\| C_\ell^{1/2} \Delta W_\ell R_\ell^{1/2} \right\|_F^2. \tag{32}$$

Substituting $\Delta W_\ell = W_\ell - \hat{W}_\ell$ yields Eq. (4).

C Efficient computation of $J^\top H J$

For a target module ℓ, let $h_{\ell,t} \in \mathbb{R}^{d_{\text{out}}}$ denote its output at token t. Let $z_{t,K} \in \mathbb{R}^K$ denote the logits restricted to the top-K coordinates of the uncompressed model, and let $p_{t,K} \in \mathbb{R}^K$ be the corresponding probabilities renormalized on this support. The reduced token-level curvature is

$$C_{\text{token},t} \approx \left(J_{t,K}^{(\ell)}\right)^\top H_{t,K} J_{t,K}^{(\ell)}, \qquad J_{t,K}^{(\ell)} = \frac{\partial z_{t,K}}{\partial h_{\ell,t}}, \tag{33}$$

where

$$H_{t,K} = \mathrm{Diag}(p_{t,K}) - p_{t,K} p_{t,K}^\top. \tag{34}$$

Directly forming $J_{t,K}^{(\ell)}$ for all tokens and modules is expensive. Instead, we exploit the structure of $H_{t,K}$. Define

$$s_t = \sqrt{p_{t,K}}, \qquad D_t = \mathrm{Diag}(s_t), \qquad \Omega_t = I - s_t s_t^\top. \tag{35}$$

Since $p_{t,K}$ is renormalized, $\|s_t\|_2 = 1$, and hence

$$\Omega_t^2 = (I - s_t s_t^\top)^2 = I - s_t s_t^\top = \Omega_t. \tag{36}$$

Thus, Ω_t is an orthogonal projector onto the subspace orthogonal to s_t.

Now define

$$A_t = D_t \Omega_t. \tag{37}$$

Then

$$A_t A_t^\top = D_t \Omega_t^2 D_t \tag{38}$$

$$= D_t \Omega_t D_t \tag{39}$$

$$= D_t (I - s_t s_t^\top) D_t \tag{40}$$

$$= \mathrm{Diag}(p_{t,K}) - p_{t,K} p_{t,K}^\top = H_{t,K}. \tag{41}$$

Therefore,

$$H_{t,K} = A_t A_t^\top. \tag{42}$$

Let e_j be the j-th canonical basis vector in $\mathbb{R}^K$, and define the deterministic probe

$$v_{t,j} = A_t e_j. \tag{43}$$

Sweeping over all K basis directions recovers the reduced Hessian:

$$\sum_{j=1}^{K} v_{t,j} v_{t,j}^\top = A_t \left(\sum_{j=1}^{K} e_j e_j^\top \right) A_t^\top = A_t A_t^\top = H_{t,K}. \tag{44}$$

For each probe $v_{t,j}$, define the scalar

$$\phi_{t,j} = v_{t,j}^\top z_{t,K}. \tag{45}$$

Its gradient with respect to the module output is

$$\nabla_{h_{\ell,t}} \phi_{t,j} = \left(J_{t,K}^{(\ell)} \right)^\top v_{t,j}. \tag{46}$$

This is a vector-Jacobian product and can be computed by reverse-mode automatic differentiation without materializing $J_{t,K}^{(\ell)}$. Denote

$$g_{\ell,t,j} = \nabla_{h_{\ell,t}} \phi_{t,j} = \left(J_{t,K}^{(\ell)} \right)^\top v_{t,j}. \tag{47}$$

Then

$$\sum_{j=1}^{K} g_{\ell,t,j} g_{\ell,t,j}^\top = \left(J_{t,K}^{(\ell)} \right)^\top \left(\sum_{j=1}^{K} v_{t,j} v_{t,j}^\top \right) J_{t,K}^{(\ell)} \tag{48}$$

$$= \left(J_{t,K}^{(\ell)} \right)^\top H_{t,K} J_{t,K}^{(\ell)}. \tag{49}$$

Thus, the curvature is recovered exactly on the top-K subspace; the only approximation is the restriction from the full vocabulary to the top-K support.

In practice, for each calibration batch, we compute the top-K logits and renormalized probabilities, form the probes $\{v_{t,j}\}_{j=1}^{K}$, and backpropagate each scalar $\phi_{t,j}$. Backward hooks on each target linear module collect the vectors $g_{\ell,t,j}$ and accumulate

$$\widehat{C}_\ell = \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{K} g_{\ell,t,j} g_{\ell,t,j}^\top \approx \mathbb{E}_t \left[J_t^\top H_t J_t \right]. \tag{50}$$

A small damping term may be added to $\widehat{C}_\ell$ before taking matrix square roots or inverse square roots.

D Derivation of the singular-component score

The adaptive rank-allocation score in Eq. (10) follows from a first-order Taylor approximation in the whitened SVD coordinates. Recall that

$$B_\ell = C_\ell^{1/2} W_\ell R_\ell^{1/2}. \tag{51}$$

A perturbation ΔW_ℓ induces

$$\Delta B_\ell = C_\ell^{1/2} \Delta W_\ell R_\ell^{1/2}, \qquad \Delta W_\ell = C_\ell^{-1/2} \Delta B_\ell R_\ell^{-1/2}. \tag{52}$$

Let $G_\ell = \partial \mathcal{L} / \partial W_\ell$. The first-order loss change is

$$\Delta \mathcal{L} \approx \langle G_\ell, \Delta W_\ell \rangle \tag{53}$$

$$= \operatorname{tr} \left(G_\ell^\top C_\ell^{-1/2} \Delta B_\ell R_\ell^{-1/2} \right) \tag{54}$$

$$= \operatorname{tr} \left(\left(C_\ell^{-1/2} G_\ell R_\ell^{-1/2} \right)^\top \Delta B_\ell \right) \tag{55}$$

$$= \langle \widetilde{G}_\ell, \Delta B_\ell \rangle, \tag{56}$$

where

$$\widetilde{G}_\ell = C_\ell^{-1/2} G_\ell R_\ell^{-1/2} = \frac{\partial \mathcal{L}}{\partial B_\ell}. \tag{57}$$

Let

$$B_\ell = \sum_i \sigma_{\ell,i} u_{\ell,i} v_{\ell,i}^\top. \tag{58}$$

Holding $u_{\ell,i}$ and $v_{\ell,i}$ fixed, we have

$$\frac{\partial B_\ell}{\partial \sigma_{\ell,i}} = u_{\ell,i} v_{\ell,i}^\top. \tag{59}$$

Therefore,

$$g_{\ell,i} = \frac{\partial \mathcal{L}}{\partial \sigma_{\ell,i}} = \left\langle \widetilde{G}_\ell, u_{\ell,i} v_{\ell,i}^\top \right\rangle = u_{\ell,i}^\top \widetilde{G}_\ell v_{\ell,i}. \tag{60}$$

Removing the i-th singular component sets

$$\Delta\sigma_{\ell,i} = -\sigma_{\ell,i}.$$

The predicted first-order loss change is

$$\Delta\mathcal{L}_{\ell,i} \approx g_{\ell,i}\Delta\sigma_{\ell,i} = -g_{\ell,i}\sigma_{\ell,i}. \tag{61}$$

We use its magnitude as the component importance score:

$$I_{\ell,i} = |g_{\ell,i}\sigma_{\ell,i}|. \tag{62}$$

E Additional details for loss-aware remapping

For hybrid SVD–quantization compression, we write each truncated module as

$$\hat{W}_\ell = A_\ell D_\ell^\top, \tag{63}$$

where $A_\ell \in \mathbb{R}^{d_{\text{out}} \times r_\ell}$ and $D_\ell \in \mathbb{R}^{d_{\text{in}} \times r_\ell}$. Candidate rows are drawn from both low-rank factors. For a candidate row $r_{\ell,i}$, the 8-bit quantize-dequantize perturbation is

$$\Delta r_{\ell,i} = \mathcal{Q}_8(r_{\ell,i}) - r_{\ell,i}. \tag{64}$$

Let

$$\gamma_{\ell,i} = \frac{\partial \mathcal{L}_{\text{cal}}}{\partial r_{\ell,i}} \tag{65}$$

be the row gradient computed on calibration data after SVD truncation. A first-order Taylor approximation gives

$$\Delta\mathcal{L}_{\ell,i} \approx \langle \gamma_{\ell,i}, \Delta r_{\ell,i} \rangle = \langle \gamma_{\ell,i}, \mathcal{Q}_8(r_{\ell,i}) - r_{\ell,i} \rangle. \tag{66}$$

We define the remapping sensitivity score as

$$s_{\ell,i} = |\langle \gamma_{\ell,i}, \mathcal{Q}_8(r_{\ell,i}) - r_{\ell,i} \rangle|. \tag{67}$$

This score is small when either the row is quantized accurately or the quantization error lies in a direction that has little effect on the calibration loss.

Let C_{svd} denote the storage saved by SVD truncation and C_{target} denote the desired total storage saving. The remaining saving required from quantization is

$$C_{\text{rem}} = \max\{0, C_{\text{target}} - C_{\text{svd}}\}. \tag{68}$$

Each candidate row has a storage saving $c_{\ell,i}$, which may depend on the row length and quantization format. We select rows by solving the budgeted selection problem

$$\min_{\mathcal{S}} \sum_{(\ell,i)\in\mathcal{S}} s_{\ell,i} \quad \text{subject to} \quad \sum_{(\ell,i)\in\mathcal{S}} c_{\ell,i} \geq C_{\text{rem}}. \tag{69}$$

In practice, we use a greedy approximation. When all candidate rows have the same storage saving, rows are selected in increasing order of $s_{\ell,i}$. When row sizes differ, rows are selected by increasing cost per saved parameter:

$$\frac{s_{\ell,i}}{c_{\ell,i}}. \tag{70}$$

The selected rows are stored in 8-bit precision, and their row indices are stored as metadata. This index overhead is negligible relative to the compressed model size.

Table 10: Cold-load time of FP16 and remapped checkpoints. Ratio denotes the retained parameter ratio.

Ratio	FP16 cold load	Remapped cold load	Overhead
0.8	5.723 s	5.919 s	3.4%
0.6	4.603 s	4.662 s	1.3%

E.1 Remapped checkpoints at inference time

As in Dobi-SVD (Wang et al., 2025a) and ZS-SVD (Abbasi et al., 2026), remapping is used only as a mixed-bit checkpoint storage format. When the model is loaded, the 8-bit rows are dequantized and the low-rank factors are restored to FP16 once, after which inference uses standard FP16 matrix multiplication. Consequently, there is no recurring gather/scatter or layout-conversion overhead during inference, and no custom fused kernels are required.

Table 10 compares the cold-load time of FP16 checkpoints with that of remapped checkpoints. Restoring the remapped factors adds a one-time overhead of 3.4% at a retained ratio of 0.8 and 1.3% at a retained ratio of 0.6. This cost is incurred once at load time and is amortized across all subsequent inference, making it negligible in practice.

Because the factors are restored to FP16, remapping reduces checkpoint size but not inference-time GPU memory or computation. Accordingly, all runtime and GPU-memory results in this paper (Section 4.2.1 and Appendix G) use FP16 weights and caches without remapping, and remapping should be viewed as an optional module that improves the accuracy-versus-checkpoint-size tradeoff. Realizing inference-time memory savings would require mixed-precision kernels that operate directly on the packed int8/FP16 factors, casting only within each matrix multiplication instead of dequantizing the entire model at load time. Developing such kernels is a substantial systems contribution and a natural direction for future work. It is orthogonal to our core contribution, the top-K KL-curvature estimation that defines an output-whitened space preserving the model's predictive distribution.

F Vision-language model results

Table 11 reports VLM compression results on LLaVA-1.5 7B, LLaVA-1.5 13B, and SmolVLM 2B. We use 256 randomly sampled ScienceQA-IMG sequences with length 2048 as calibration data, and evaluate the compressed models on ScienceQA-IMG (Lu et al., 2022) and SEED-Bench (Li et al., 2024). We vary the parameter retention ratio across 90%, 80%, 70%, 60%, and 50% where applicable, and compare against ASVD (Yuan et al., 2025), SVD-LLM (Wang et al., 2025c), QSVD (Wang et al., 2025d), and WSVD (Wang et al., 2026). Consistent with QSVD and WSVD, which are VLM-specific compression methods, we apply truncation only to the query, key, and value attention projections. Across models and retention ratios, IO-SVD achieves the best average accuracy among compressed methods, with particularly strong gains on SmolVLM 2B, where it remains close to the FP16 model while substantially outperforming prior SVD-based baselines.

G Compressed K/V cache implementation and analysis

G.1 Evaluation setup

Unless stated otherwise, efficiency measurements use LLaMA-2-7B compressed with IO-SVD at 60% pruning (40% parameter retention). Remapping and 8-bit quantization are not used, and all weights and caches remain in FP16, so the measured gains come solely from low-rank weight factorization and, when enabled, compressed K/V caching. All experiments run on a single NVIDIA RTX PRO 6000 Blackwell Max-Q GPU with 96 GB of memory, and throughput is reported in generated tokens per second. In the high-load setting, we use batch size $B{=}64$ with 1024-token prompts and 1024 generated tokens, one warm-up batch, and three measured batches. In this setting, throughput increases from 473.3 to 2041.2 tokens/s, peak GPU memory decreases from 77.5 to 23.0 GB, and KV-cache memory decreases from 64.0 to 11.6 GB (Table 9 and Figure 4).

Table 11: Accuracy evaluation of different methods under FP16.

Model.	Method	ScienceQA-IMG ↑					SEED-Bench ↑					Avg. ↑
		ratio: 90%	ratio: 80%	ratio: 70%	ratio: 60%	ratio: 50%	ratio: 90%	ratio: 80%	ratio: 70%	ratio: 60%	ratio: 50%	
LLaVA1.5 7B	ASVD	49.93	50.12	47.10	36.69	19.19	54.27	53.53	48.35	37.17	24.17	42.05
	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	67.72	**68.22**	67.08	65.05	62.37	59.84	59.07	59.78	59.00	58.23	62.64
	WSVD	**68.17**	67.72	**67.28**	**65.89**	**65.49**	**60.10**	**60.17**	**59.89**	**60.18**	**60.46**	**63.54**
	Ours	**67.92**	**68.07**	**68.17**	**66.73**	**65.99**	**60.29**	**60.32**	**60.22**	**60.15**	**60.64**	**63.85**
	FP16	Accuracy: 68.01					Accuracy: 60.18					64.10
LLaVA1.5 13B	ASVD	71.39	71.59	70.00	70.25	69.51	61.92	61.91	61.54	61.51	60.71	66.03
	SVD-LLM	71.05	70.85	70.30	70.35	70.30	62.28	62.34	62.25	62.08	**63.01**	66.48
	QSVD	**71.89**	**71.99**	71.49	71.54	71.39	**62.61**	**62.64**	**62.82**	**62.63**	**62.52**	67.15
	WSVD	**71.99**	71.84	**72.53**	**71.59**	**71.44**	62.52	**62.68**	62.38	62.37	62.37	**67.17**
	Ours	71.79	**72.14**	**72.29**	**72.14**	**72.68**	**62.64**	62.49	**62.46**	**62.45**	62.29	**67.34**
	FP16	Accuracy: 71.83					Accuracy: 62.53					67.18
		ratio: 90%	ratio: 80%		ratio: 70%		ratio: 90%	ratio: 80%		ratio: 70%		
SmolVLM 2B	ASVD	29.30	3.97		0.20		17.85	1.50		0.95		8.96
	SVD-LLM	40.06	17.20		3.82		32.49	15.89		4.60		19.01
	QSVD	**77.00**	62.77		42.59		64.80	50.46		36.24		55.64
	WSVD	76.30	**71.74**		**60.93**		**65.78**	**63.29**		**54.45**		**65.42**
	Ours	**83.34**	**82.65**		**82.30**		**68.36**	**68.65**		**68.15**		**75.58**
	FP16	Accuracy: 84.53					Accuracy: 68.47					76.53

Table 12: Decode throughput (tokens/s) across pruning ratios on LLaMA-2-7B at B=64, $1024 \rightarrow 1024$ (no remapping). Speedups over the dense model (473.3 tokens/s) are shown in parentheses.

Pruning	Weight only	Weight + K/V cache
20%	467.3 (0.99×)	**1433.9** (3.03×)
40%	473.0 (1.00×)	**1623.0** (3.43×)
60%	483.6 (1.02×)	**2041.2** (4.31×)

G.2 Latent K/V caching

We drop the layer index and write the compressed key and value projections as $\hat{W}_{\mathrm{k}} = A_{\mathrm{k}} D_{\mathrm{k}}^{\top}$ and $\hat{W}_{\mathrm{v}} = A_{\mathrm{v}} D_{\mathrm{v}}^{\top}$ (Eq. (11)), with retained ranks r_{k} and r_{v}. Each full projection (k_{proj} or v_{proj}) is factorized as a whole rather than per attention head. For each token x_j, we cache only the latent codes

$$z_j^{\mathrm{k}} = D_{\mathrm{k}}^{\top} x_j \in \mathbb{R}^{r_{\mathrm{k}}}, \qquad z_j^{\mathrm{v}} = D_{\mathrm{v}}^{\top} x_j \in \mathbb{R}^{r_{\mathrm{v}}}, \tag{71}$$

computed separately for keys and values, so each layer stores $r_{\mathrm{k}} + r_{\mathrm{v}}$ values per token instead of the full key and value vectors. Attention remains exact with respect to the compressed model; the cache introduces no additional approximation.

Values. Let $A_{\mathrm{v},h}$ denote the rows of A_{v} that produce head h, so that $v_{j,h} = A_{\mathrm{v},h} z_j^{\mathrm{v}}$. Because the attention output is linear in the values,

$$o_{i,h} = \sum_{j \le i} \alpha_{ij,h}\, v_{j,h} = A_{\mathrm{v},h} \Big(\sum_{j \le i} \alpha_{ij,h}\, z_j^{\mathrm{v}} \Big), \tag{72}$$

where $\alpha_{ij,h}$ are the attention weights of head h. We therefore aggregate the cached codes directly in the rank-r_{v} space and apply $A_{\mathrm{v},h}$ once per head and query.

Keys and RoPE. RoPE is not applied to the cached codes. With $k_{j,h} = A_{\mathrm{k},h} z_j^{\mathrm{k}}$ and R_i the RoPE rotation at position i, the attention logit between query position i and key position j satisfies

$$(R_i q_{i,h})^{\top} (R_j A_{\mathrm{k},h} z_j^{\mathrm{k}}) = (A_{\mathrm{k},h}^{\top} R_j^{\top} R_i q_{i,h})^{\top} z_j^{\mathrm{k}} = (A_{\mathrm{k},h}^{\top} R_{i-j}\, q_{i,h})^{\top} z_j^{\mathrm{k}}, \tag{73}$$

using $R_j^{\top} R_i = R_{i-j}$. During decoding, we therefore apply the relative RoPE transformation to the query, project it through the corresponding block of $A_{\mathrm{k}}^{\top}$, and compute its inner product with the cached code z_j^{k}. This is mathematically equivalent to reconstructing and rotating the full key, but never materializes it. Because the projected query depends on the relative position $i - j$, this projection is

computed for every cached position at each decoding step, which constitutes the additional online computation of the compressed path. During prefill, keys are reconstructed temporarily and standard attention is used.

Implementation. The implementation uses standard PyTorch operations and no custom fused kernels. The mechanism applies to any method with low-rank K/V projections and is not specific to IO-SVD; IO-SVD determines how much model quality is preserved at a given K/V rank.

G.3 Cost model and serving regimes

The compressed key path trades memory traffic for computation: it replaces reads of full keys and values from the cache with batched matrix multiplications on the latent codes. Under the roofline model (Pope et al., 2023; Kwon et al., 2023), runtime can be approximated as

$$T \approx \max\left(\frac{\text{FLOPs}}{\text{compute throughput}}, \frac{\text{bytes transferred}}{\text{memory bandwidth}}\right), \tag{74}$$

and the arithmetic intensity $I = \text{FLOPs}/\text{bytes transferred}$ measures how much computation is performed per byte moved from memory. When decoding is bandwidth-bound, the second term dominates, so additional FLOPs are worthwhile if they sufficiently reduce memory traffic and increase arithmetic intensity (Dao et al., 2022). The roofline model therefore predicts the regime dependence observed in Table 9.

At small batch sizes, where memory traffic is not the main bottleneck, the extra computation of the compressed key path increases per-token decoding latency. At $B{=}64$, a regime commonly targeted by large-scale inference serving systems (Kwon et al., 2023), KV-cache reads dominate: the compressed path removes about 27 GB of memory traffic per decoding step and reduces per-token decoding latency. The compressed path is therefore beneficial in bandwidth-bound serving settings and can be disabled when the workload is not bandwidth-limited. This is controlled by a runtime flag and requires no recompression of the model.

G.4 Throughput across compression ratios

Table 12 reports throughput at 20%, 40%, and 60% pruning in the high-load setting ($B{=}64$, $1024 \rightarrow 1024$). As pruning increases, the retained ranks of the K/V projections decrease, reducing KV-cache memory and traffic. Relative to the dense model at 473.3 tokens/s, in this KV-cache-dominated regime, latent K/V caching increases throughput by $3.03\times$ at 20% pruning and by $4.31\times$ at 60% pruning, whereas weight-only compression leaves throughput essentially unchanged.

H Calibration time analysis

We measure only the compression procedure, excluding model loading, evaluation, and checkpoint writing. All methods use LLaMA-7B, 256 WikiText-2 calibration samples of length 2048, and identical timing instrumentation, and run on NVIDIA RTX PRO 6000 Blackwell Max-Q GPUs with 96 GB of memory.

Table 13: Compression time (minutes) on LLaMA-7B.

Method	40% pruning	60% pruning
SVD-LLM	8.68	8.60
ZS-SVD	9.09	9.05
IO-SVD	15.03	18.03
IO-SVD + loss-aware remapping	40.77	33.58

Table 13 compares IO-SVD with SVD-LLM and ZS-SVD. IO-SVD takes 15.03 and 18.03 minutes at 40% and 60% pruning, respectively, compared with roughly 9 minutes for SVD-LLM and ZS-SVD. We also attempted Dobi-SVD (Wang et al., 2025a), an optimization-based rank-allocation method.

Its rank-allocation stage alone took approximately 19 hours, well beyond the 9–18 minute range of the non-iterative methods.

Table 14: Compression time (minutes) of each IO-SVD stage on LLaMA-7B.

IO-SVD stage	40% pruning	60% pruning
Input whitening	3.47	3.47
Top-K output-curvature estimation	5.24	8.40
Computing the whitened-space SVD	5.27	5.20
Adaptive rank allocation	0.518	0.480
Low-rank factorization	0.525	0.478

Table 14 breaks down the compression time of IO-SVD by stage. Most of the additional overhead relative to prior SVD-based methods comes from top-K output-curvature estimation, which estimates output-side predictive sensitivity and is performed once during compression. Loss-aware remapping requires an additional post-truncation gradient pass over the calibration samples, which adds 25.74 minutes at 40% pruning and 15.55 minutes at 60% pruning (Table 13).

Overall, IO-SVD has a higher compression-time overhead than non-iterative SVD baselines, but remains far more efficient than optimization-based rank allocation. This one-time cost comes with the quality gains reported in Section 4.1.1, including substantially better free-form generation and speculative-decoding acceptance.

Table 15 summarizes which stages are gradient-free and which require backward passes. Stages 4–6 use autograd to estimate gradient or curvature statistics; the model weights remain frozen, no optimizer is used, and no parameters are updated. These statistics are then used by the closed-form whitening, SVD, and rank-allocation procedures.

Table 15: Gradient requirements of each IO-SVD stage.

#	Stage	What it computes	Category
1	Input-side whitening	Calibration input covariance $XX^\top$ and its Cholesky factor for each module	Gradient-free
2	SVD and low-rank factor formation	Whitened SVD, rank-based truncation, and low-rank factor formation	Gradient-free
3	Vanilla remapping	Storage-cost model and stopping rule	Gradient-free
4	Top-K output-side sensitivity	$C \approx \mathbb{E}[J^\top H J]$ using top-K vector–Jacobian products	Backward passes on the frozen model
5	First-order importance	Per-module gradient scores used for adaptive rank allocation	Backward passes on the frozen model
6	Loss-aware remapping (optional)	Gradient scores for the formed low-rank factors	Backward passes with no parameter updates