Taking Low-Rank LLM Compression a Step Further: A Global Perspective with Fused Inference

Xinhao Huang, Yuqiang He, Zeyi Wen*
The Hong Kong University of Science and Technology (Guangzhou)
xhuang171@connect.hkust-gz.edu.cn, {yuqiangh, wenzeyi}@hkust-gz.edu.cn

Abstract

Low-rank decomposition has emerged as a promising paradigm for compressing Large Language Models (LLMs). However, current methods universally rely on greedy, layer-wise reconstruction objective. This approach induces cascading error propagation, where localized truncation errors amplify across deep architectures. To mitigate this limitation, we formulate the low-rank decomposition problem from a global perspective using the Alternating Direction Method of Multipliers (ADMM). By decoupling the non-convex rank constraints from the holistic language modeling objective, our method preserves superior model fidelity. Systemically, the standard sequential execution of decomposed matrices often fails to yield practical wall-clock speedups due to kernel launch overheads and activation memory traffic. We bridge the gap by developing a fused GPU operator. Extensive experiments across diverse architectures demonstrate that our method significantly benefits existing methods. For example, at a 30% compression ratio on Qwen-3-8B, our method achieves a perplexity of 10.05—closely approaching the uncompressed model's 9.85—while further improving zero-shot reasoning accuracy. Furthermore, by integrating our custom fused operator into the widely adopted inference ecosystems, we translate theoretical FLOPs reduction into practical inference speedup. The source code is available at this URL.

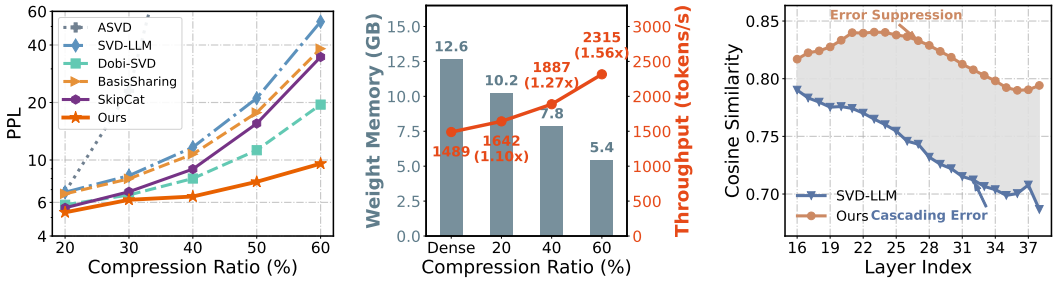
1 Introduction

Large Language Models (LLMs) have demonstrated unprecedented capabilities across diverse natural language processing tasks [1–5]. However, their massive parameter scale imposes prohibitive memory and computational bottlenecks, severely limiting their deployment in resource-constrained environments. To mitigate these overheads, various model compression techniques have been proposed, among which Low-Rank Decomposition (e.g., Singular Value Decomposition, SVD) has emerged as a promising paradigm [6–9]. Unlike unstructured pruning [10–12] or quantization [13, 14], which often require specialized hardware support to realize actual speedups, low-rank decomposition structurally compresses weight matrices, yielding hardware-friendly reductions in both memory footprint and inference latency.

Despite these architectural advantages, the progress of low-rank LLM compression has encountered a stubborn bottleneck. Existing methods, ranging from activation-aware SVD variants (e.g., ASVD [6], SVD-LLM [7]) to advanced structural sharing designs (e.g., Basis Sharing [9] and SkipCat [15]), universally adopt a greedy, layer-wise reconstruction strategy. Specifically, they formulate the compression task as a surrogate objective: minimizing the Frobenius norm of the localized feature discrepancy for each individual layer (*i.e.,* , $\min \|W_l X_l - \hat{W}_l X_l\|_F^2$). While mathematically convenient, this myopic approach harbors a fatal flaw when applied to deep architectures: *Cascading Error Propagation*. Because layer-wise approximations ignore inter-layer dependencies, truncation errors

*Corresponding author.

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



(a) Perplexity-Compression Curves (b) Inference Acceleration (c) Cascading Error Suppression

Figure 1: **(a)** Our method prevents the perplexity explosion observed in greedy, layer-wise methods at high compression ratios. **(b)** The fused GPU operator translates memory reductions into substantial end-to-end throughput acceleration in `vLLM`. **(c)** Error cumulation on LLaMA-2-13B, our method actively suppresses upstream error propagation.

inevitably accumulate and amplify through non-linear activations. This renders the local optima severely sub-optimal for the holistic language modeling objective, causing catastrophic performance degradation at moderate-to-high compression ratios.

To eradicate this local error accumulation, we propose taking low-rank compression a step further by introducing a global perspective via the Alternating Direction Method of Multipliers (ADMM) [16]. Instead of treating decomposition as a disconnected, layer-by-layer sequence, we mathematically formulate the low-rank constraint as a global optimization problem. By introducing auxiliary variables, our method decouples the non-convex rank constraints from the holistic modeling objective. This mechanism enables the network to dynamically coordinate and inherently compensate for representation deviations across all layers simultaneously, actively suppressing upstream error propagation (as illustrated in Figure 1c). Consequently, our global approach gracefully prevents the catastrophic perplexity explosion observed in greedy layer-wise methods at high compression ratios (Figure 1a). Crucially, our method is orthogonal to existing low-rank formulations; it serves as a universal performance amplifier that can be seamlessly integrated with both standard low-rank pipelines and structurally shared topologies.

Systemically, mitigating algorithmic degradation is only the first step toward efficient LLM deployment. We reveal that the standard sequential execution of decomposed matrices ($Y = ABX$) fails to fully translate theoretical FLOP reductions into practical wall-clock speedups on modern GPUs due to severe *CPU-bound kernel launch overhead* caused by operator fragmentation, and the *activation memory wall* induced by redundant Global Memory (HBM) read/write cycles. To bridge the gap between theoretical compression and practical deployment, we develop a hardware-aware GPU kernel-fusion operator. By executing the low-rank projection within a single kernel launch and retaining intermediate states entirely in on-chip SRAM, we eliminate instruction dispatch latency and redundant memory traffic. Successfully integrated into the widely adopted HuggingFace Transformers [17] and the highly optimized `vLLM` [18] engine, our fused operator translates parameter reductions into end-to-end throughput acceleration (Figure 1b). In light of these advancements, the primary contributions of our work are summarized as follows:

- We reveal that current layer-wise LLM compression reconstructs linear layers in isolation, causing localized truncation errors to amplify across deep architectures, thereby establishing the necessity of a global perspective.
- We propose an ADMM-based method that addresses the low-rank constraint from a global perspective. Being intrinsically orthogonal, it serves as a universal catalyst to boost existing low-rank methods. Extensive experiments demonstrate that our method significantly benefits existing methods across diverse models.
- We expose hardware bottlenecks in sequential low-rank execution and develop a fused GPU operator. Integrated with the widely adopted inference ecosystems, this bridges the gap between mathematical FLOPs reduction and practical inference acceleration.

2 Methodology

We propose a method to tackle low-rank compression directly from a global perspective, addressing the critical gap in current LLM compression practices. We ground our approach in principled error correction to ensure the global modeling objective is preserved while mitigating cascading errors.

2.1 Preliminaries and The Layer-Wise Surrogate Flaw

Low-Rank Decomposition. For an L-layer LLM, consider a linear layer i with weight $W_i \in \mathbb{R}^{d_{out} \times d_{in}}$ and input activations $X_i \in \mathbb{R}^{B \times N \times d_{in}}$. The Low-rank compression seeks a factorized approximation $\hat{W}_i = A_i B_i$, where $A_i \in \mathbb{R}^{d_{out} \times r_i}$ and $B_i \in \mathbb{R}^{r_i \times d_{in}}$ with $r_i \ll \min(d_{in}, d_{out})$. Ideally, this is an ideal cross-layer formulation:

$$\mathcal{W}^\star = \arg\min_{\mathcal{W}} \ f(\mathcal{W}) \quad \text{s.t.} \quad \text{rank}(W_i) \leq r_i, \ \forall i = 1, \ldots, L \tag{1}$$

where f denotes the global task loss (e.g., cross-entropy).

The Greedy Surrogate Flaw. Solving Eq. (1) directly is traditionally considered mathematically intractable. This intractability stems from a fundamental conflict: standard decomposition (e.g., SVD) is a purely mathematical operation defined for isolated matrices. It lacks a global perspective, ignoring both the complex inter-layer dependencies and the non-linear dynamics of deep architectures. Consequently, to leverage the closed-form optimality of SVD (via the Eckart-Young-Mirsky theorem [19]), prevailing methods are forced to dismantle the deep architecture into disconnected linear problems. They sidestep the global modeling objective $f(\mathcal{W})$ by adopting a modularized, layer-wise surrogate objective:

$$W_i^\star = \arg\min_{W_i} \mathbb{E}_{X_i \sim \mathcal{D}} \|\bar{W}_i X_i - W_i X_i\|_F^2 \quad \text{s.t.} \quad \text{rank}(W_i) \leq r_i \tag{2}$$

where $\bar{W}_i$ is the original weight and X_i are activations from the model. In essence, they use localized feature reconstruction as a *surrogate* to approximate the holistic semantic loss.

While computationally efficient, this paradigm introduces a fatal *Cascading Error Propagation*. By isolating layers, it suffers from *Activation Drift* (overfitting the obsolete static input X_i while ignoring upstream shifts ΔX_i) and a *Lack of Global Coordination*. Local truncations inevitably interact with non-linearities and compound multiplicatively, stripping the network's capacity for cross-layer compensation. As formally proven in Appendix A, this greedy approach inherently suffers from exponential error amplification.

This theoretical flaw manifests clearly in deep architectures. As visualized in Figure 1c, the representational fidelity of standard layer-wise SVD plunges precipitously in deeper layers. In contrast, our method actively suppresses upstream error propagation, maintaining high similarity. This firmly establishes the necessity of abandoning layer-wise surrogates in favor of a holistic perspective.

2.2 Decoupling Global Objective and Rank Constraints via ADMM

To overcome the aforementioned barriers and eradicate cascading error propagation, we propose to solve the global objective in Eq. (1) directly. We employ the Alternating Direction Method of Multipliers (ADMM) [16], an optimization strategy involving variable splitting. This allows us to decouple the intractable, non-convex rank constraint subspace $\mathcal{R}_i = \{V \in \mathbb{R}^{d_{in} \times d_{out}} \mid \text{rank}(V) \leq r_i\}$ from the continuous, global language modeling objective.

This decoupling is achieved by introducing a set of auxiliary variables $\mathcal{Z} = \{Z_i\}_{i=1}^L$ as follows:

$$\min_{\mathcal{W}, \mathcal{Z}} \ f(\mathcal{W}) + \sum_{i=1}^{L} I_{\mathcal{R}_i}(Z_i) \quad \text{subject to} \quad W_i = Z_i, \quad \forall i \tag{3}$$

where $I_{\mathcal{R}_i}(Z_i)$ is the indicator function for the low-rank representations $\mathcal{R}_i$:

$$I_{\mathcal{R}_i}(Z_i) \coloneqq \begin{cases} 0 & \text{if } Z_i \in \mathcal{R}_i \\ \infty & \text{otherwise.} \end{cases} \tag{4}$$

In turn, we constrain the model weights W_i to be equal to Z_i. This elegant formulation allows us to handle the global language modeling and the rank satisfaction separately, making both tractable.

To solve this constrained formulation, we construct the Augmented Lagrangian:

$$\mathcal{L}_\rho(\mathcal{W}, \mathcal{Z}, \mathcal{U}) = f(\mathcal{W}) + \sum_{i=1}^{L} \left(I_{\mathcal{R}_i}(Z_i) + \frac{\rho_i}{2} \|W_i - Z_i + U_i\|_F^2 - \frac{\rho_i}{2} \|U_i\|_F^2 \right) \tag{5}$$

where $\rho_i > 0$ is a penalty hyperparameter adjusting the strength of the proximal constraint, and $\mathcal{U} = \{U_i\}_{i=1}^L$ are the scaled dual variables (Lagrange multipliers). ADMM solves this by alternately minimizing the Augmented Lagrangian over the primal variables ($\mathcal{W}$, $\mathcal{Z}$) and performing a dual ascent step on $\mathcal{U}$. This decomposes the global challenge into three mathematically tractable subproblems that iterate until convergence at step t:

$$W_i^{t+1} = \arg\min_{W_i} \left(f(\mathcal{W}) + \frac{\rho_i}{2} \|W_i - Z_i^t + U_i^t\|_F^2 \right) \tag{6}$$

$$Z_i^{t+1} = \arg\min_{Z_i \in \mathcal{R}_i} \frac{\rho_i}{2} \|W_i^{t+1} - Z_i + U_i^t\|_F^2 = \Pi_{\mathcal{R}_i}(W_i^{t+1} + U_i^t) \tag{7}$$

$$U_i^{t+1} = U_i^t + W_i^{t+1} - Z_i^{t+1} \tag{8}$$

Intuition: The "Elastic" Decoupling. To intuitively understand why this specific formulation successfully decouples the problem, one can view the proximal penalty term ($\frac{\rho_i}{2} \| \cdot \|_F^2$) as an *elastic band* connecting the continuous weights W_i and the low-rank proxy Z_i as conceptually illustrated in Figure 2. Instead of forcing a single matrix to simultaneously satisfy the complex holistic loss $f(\mathcal{W})$ and the rigid non-convex constraint rank $\leq r_i$, ADMM divides the labor:

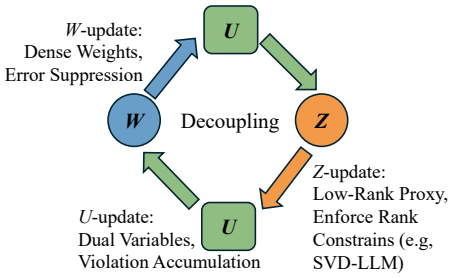


Figure 2: Overview of the ADMM decoupling cycle. It alternates between global error suppression (W), rigid orthogonal projection onto the low-rank representations (Z), and dual violation accumulation (U).

Global Error Suppression ($\mathcal{W}$-update). In Eq. (6), our method identifies the dense weights $\mathcal{W}$ to jointly minimize the holistic language modeling objective $f(\mathcal{W})$. Unlike isolated layer-wise methods, this global formulation enables all layers to simultaneously coordinate and inherently compensate for cascaded representation errors (ΔX_i) from upstream layers. This mechanism successfully decouples the error growth from the network depth. Concurrently, the proximal penalty acts as an "elastic band," gently anchoring W_i to the low-rank proxy Z_i^t to ensure the semantic representations do not drift excessively away from the compressible representations.

Orthogonal Projection Operators ($\mathcal{Z}$-update). In Eq. (7), Z_i operates entirely within the mathematical constraint space. It completely ignores the complex language modeling loss $f(\mathcal{W})$. Its sole objective is to rigidly project the shifted continuous weights ($W_i^{t+1} + U_i^t$) onto the closest exact low-rank representations $\mathcal{R}_i$ via Euclidean distance minimization ($\Pi_{\mathcal{R}_i}$). Crucially, this projection is intrinsically orthogonal to the specific choice of decomposition. Whether using standard activation-aware SVD (e.g., SVD-LLM) or structurally shared SVD (e.g., SkipCat), our method seamlessly accommodates it by substituting $\Pi_{\mathcal{R}_i}$ with the corresponding specific SVD operator.

Violation Accumulation ($\mathcal{U}$-update). Finally, Eq. (8) executes a dual ascent step. The dual variable U_i tracks and accumulates the residual gap between the continuous weights W_i and their low-rank counterparts Z_i. Over successive iterations, U_i acts as a corrective feedback loop, progressively "tightening the elastic band" to force the global solution into the feasible rank-constrained region without abruptly destroying the learned semantic representations.

2.3 Efficient Low-Rank Inference via Hardware-Aware Operator Fusion

While our method effectively restores algorithmic accuracy, translating theoretical FLOP reductions into practical wall-clock acceleration presents a system challenge. A foundational advantage of our

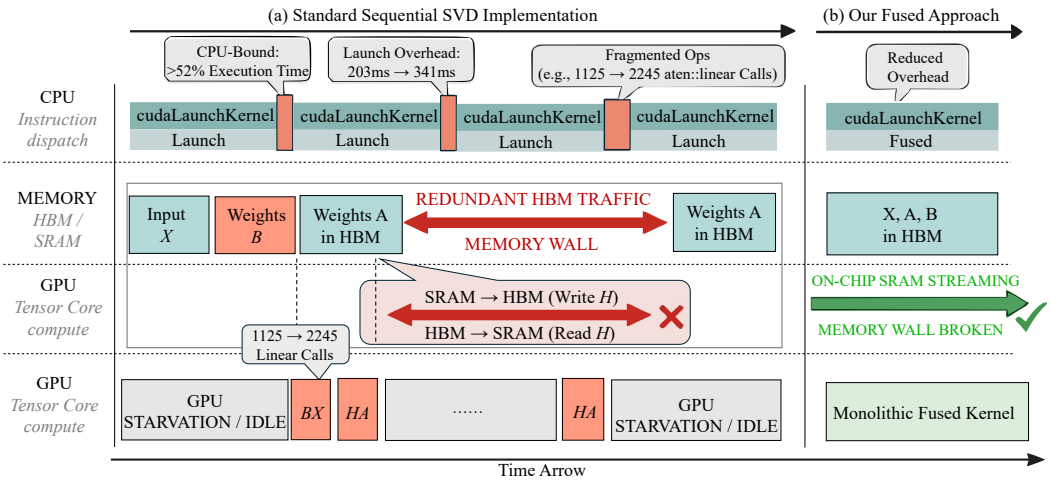


Figure 3: Hardware bottlenecks in standard sequential execution vs. our Fused ABX operator. Standard execution suffers from severe CPU kernel launch overhead and redundant memory accesses. Our monolithic fused operator eradicates these bottlenecks.

ADMM formulation is that it retains the standard decomposed matrix structure. However, evaluating the decomposed layer $Y = ABX$ requires computing an intermediate activation $H = BX$ before the final projection $Y = AH$.

As illustrated in Figure 3, this two-step sequential execution struggles to provide practical acceleration (e.g., yielding negligible improvements at 20% compression in HuggingFace Transformers). Our profiling (Batch Size=128, Prompt=128, Gen=5) on an NVIDIA A800 GPU reveals that theoretical computational savings are cannibalized by two critical hardware bottlenecks. First, *Operator Fragmentation & CPU Overhead*: evaluating two separate linear layers doubles the operator invocations. Since Tensor Cores execute small-rank projections in microseconds, the CPU instruction dispatch (`cudaLaunchKernel`) severely lags, occupying over 52% of total execution time and starving the GPU. Second, *The Activation Memory Wall*: sequential execution forces the explicit writing and reading of the intermediate activation H to and from high-latency Global Memory (HBM), inducing massive redundant memory traffic that negates FLOP reductions. Specifically, for an input sequence length N and hidden dimension d, the intermediate tensor $H \in \mathbb{R}^{N \times r}$ generates $2Nr \times$ sizeof(FP16) bytes of redundant memory traffic. As the target rank r increases, this severe memory-bound overhead directly cannibalizes the $\mathcal{O}(Nd(d-2r))$ theoretical FLOP reductions.

Hardware-Aware Fused ABX Operator. To bypass these bottlenecks, we implement a custom `Fused-ABX` kernel using a hierarchical tiling strategy. Unlike disjoint GEMMs, our kernel re-architects the data flow by co-partitioning the input X and the low-rank weights $\{A, B\}$. Specifically, we decompose the computation into fine-grained thread-block tiles where corresponding chunks of B and X are loaded into on-chip SRAM. The intermediate results of $H = BX$ are maintained as transient register-level accumulators. By carefully tuning the tile size, we ensure the intermediate footprint stays within the physical limits of the Register File, allowing it to be immediately consumed by the subsequent projection with A without any HBM materialization. This unified design not only eliminates the $2Nr$ redundant HBM traffic but also halves the `cudaLaunchKernel` overhead, effectively restoring the compute-bound efficiency of the projection.

3 Experiments

In this section, we conduct extensive empirical evaluations to validate the effectiveness and robustness of our proposed ADMM-based method. Specifically, our experiments are meticulously designed to demonstrate that our method: (i) achieves consistent and superior performance across a diverse range of LLM architectures and parameter scales; (ii) improves the performance of compressed models across various compression ratios; and (iii) serves as a plug-and-play performance amplifier for existing decomposition methods, without altering their native inference formats, fully preserving their

Table 1: Performance comparison of LLaMA-2-13B across different compression ratios (20%, 40%, and 60%). The *Drop* column indicates the absolute decrease in average accuracy compared to the uncompressed Dense model. To demonstrate the plug-and-play nature of our method, we apply it to two representative paradigms: the standard method SVD-LLM (*ADMM-SVD*) and the structurally shared method SkipCat (*ADMM-CatSVD*). Results for intermediate ratios (30% and 50%) are provided in Appendix Table 5.

Ratio	Methods	PPL ↓	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	WinoG.	Avg. ↑	Drop ↓
0%	Dense	4.88	49.15	77.61	80.70	79.35	45.80	80.69	72.06	69.33	-
20%	ASVD	6.80	44.03	72.10	78.56	71.90	**46.00**	76.22	69.85	65.52	3.81
	SVD-LLM	6.78	40.10	67.93	76.70	62.58	40.60	74.16	69.30	61.62	7.71
	Dobi-SVD	5.83	43.00	68.39	75.84	71.37	43.40	76.50	68.35	63.84	5.49
	BasisSharing	6.65	40.44	67.42	72.91	62.68	42.20	73.23	69.61	61.21	8.12
	SkipCat	5.62	**47.01**	**74.49**	77.92	72.75	44.60	77.69	**71.82**	66.61	2.72
	ADMM-SVD	5.64	43.34	68.64	76.02	72.00	42.20	76.88	67.17	63.75	5.58
	ADMM-CatSVD	**5.32**	46.33	73.74	**80.49**	**76.95**	43.60	**78.94**	69.46	**67.07**	**2.26**
40%	ASVD	462	22.35	30.13	57.71	28.26	23.40	54.08	49.64	37.94	31.39
	SVD-LLM	11.66	28.24	49.33	65.90	43.23	34.40	64.91	61.17	49.60	19.73
	Dobi-SVD	8.00	33.02	56.19	66.94	52.33	35.60	67.95	65.19	53.89	15.44
	BasisSharing	10.73	29.18	50.46	60.80	42.82	36.00	62.79	61.96	49.14	20.19
	SkipCat	8.96	30.72	56.27	67.16	49.54	35.20	67.68	62.27	52.69	16.64
	ADMM-SVD	7.34	34.22	58.50	69.48	61.27	39.80	70.18	63.69	56.73	12.60
	ADMM-CatSVD	**6.44**	**36.77**	**64.39**	**73.58**	**67.75**	**41.80**	**74.32**	**66.38**	**60.71**	**8.62**
60%	ASVD	5069	27.47	27.15	37.83	26.15	24.80	50.00	49.17	34.65	34.68
	SVD-LLM	52.97	22.01	30.89	37.86	30.14	27.00	53.37	52.49	36.25	33.08
	Dobi-SVD	19.50	23.38	32.11	39.24	31.53	29.00	54.24	52.33	37.40	31.93
	BasisSharing	38.26	23.89	32.95	37.83	30.27	28.80	52.83	53.51	37.15	32.18
	SkipCat	34.63	22.53	31.99	40.31	31.18	26.80	54.08	54.22	37.30	32.03
	ADMM-SVD	11.83	26.02	41.67	52.84	42.29	29.80	60.34	57.14	44.30	25.03
	ADMM-CatSVD	**9.57**	**27.82**	**46.00**	**63.79**	**48.00**	**32.60**	**64.74**	**58.17**	**48.73**	**20.60**

hardware-friendly inference speedups and memory reductions. Implementation details are provided in Appendix C.

To verify the generalizability of our method, we experiment with a diverse suite of widely adopted LLMs covering varying architectural designs and parameter scales. Our evaluations include the LLaMA-2 family (7B and 13B) [2], LLaMA-3-8B [3], Qwen3-8B [1], and the Gemma-2 family (9B and 27B) [20]. We compare it against a strong repertoire of open-source low-rank decomposition and structured pruning methods. For low-rank decomposition, we evaluate against advanced standard layer-wise and structurally shared methods, including ASVD [6], SVD-LLM [7], Dobi-SVD (without quantization remapping for a fair comparison of low-rank capabilities) [21], BasisSharing [9], and the current state-of-the-art SkipCat [15]. For structured pruning, we include LLM-Pruner [11], FLAP [10], and SliceGPT [22].

3.1 Main Results

Overall Performance Analysis. Our method serves as a plug-and-play amplifier for existing low-rank pipelines. Table 1 demonstrates our peak performance when built upon the strongest structural baseline (SkipCat), while Table 2 explicitly breaks down this amplification effect across different paradigms. The critical *Drop* metric (the absolute accuracy decrease from the uncompressed dense baseline) shows that integrating ADMM consistently rescues models from severe degradation. On Qwen-3-8B, applying ADMM to SVD-LLM (denoted as *ADMM-SVD*) reduces perplexity from an explosive 20.15 down to 12.35, and narrows the accuracy drop by over 5 absolute percentage points. This proves its ability to effectively recover information lost in greedy layer-wise projections. We also evaluate our approach on the knowledge-intensive benchmark MMLU. As detailed in Appendix D.3, ADMM-SVD consistently outperforms the baseline.

Generalization across Model Architectures. Our method generalizes robustly across diverse model families (Table 2). On established models like LLaMA-2, ADMM provides steady improvements. Modern, highly compact (LLaMA-3, Qwen-3) or topologically complex (Gemma-2) architectures often prove to be fragile under the greedy layer-wise truncation method. For example, standalone SkipCat collapses entirely on LLaMA-3-8B (yielding a severe 30.34% accuracy drop), but

Table 2: Performance comparison on various models at 20% and 30% compression ratios. Per-task results are provided in Appendix Table 6.

Ratio	Methods	Qwen-3-8B			LLaMA-2-7B			LLaMA-3-8B		
		PPL ↓	Avg. ↑	Drop ↓	PPL ↓	Avg. ↑	Drop ↓	PPL ↓	Avg. ↑	Drop ↓
0%	Dense	9.85	69.46	-	5.46	66.72	-	6.14	69.98	-
20%	ASVD	21.29	57.94	11.52	9.04	59.71	7.01	1041	40.47	29.51
	Dobi-SVD	14.00	54.70	14.76	9.39	48.75	17.97	14.93	50.59	19.39
	Basis Sharing	14.91	60.62	8.83	7.97	57.83	8.89	14.67	58.70	11.28
	SVD-LLM	13.84	57.01	12.45	8.78	54.31	12.41	19.11	55.82	14.16
	SkipCat	12.12	64.94	4.52	6.30	62.44	4.28	13.00	54.91	15.07
	ADMM-SVD	9.99	58.59	10.87	6.47	58.72	8.00	9.38	59.77	10.21
	ADMM-CatSVD	**8.64**	**64.98**	**4.48**	**5.76**	**62.87**	**3.85**	**8.21**	**62.01**	**7.97**
30%	ASVD	116.71	43.65	25.81	541.87	36.47	30.25	23174	35.15	34.83
	Dobi-SVD	18.25	52.23	17.23	12.38	44.21	22.51	31.13	42.18	27.80
	Basis Sharing	18.42	55.35	14.11	10.32	53.46	13.26	25.95	45.29	24.69
	SVD-LLM	20.15	46.70	22.76	11.66	50.03	16.69	43.91	46.98	23.00
	SkipCat	14.82	57.65	11.81	7.77	56.24	10.48	48.24	39.64	30.34
	ADMM-SVD	12.35	51.89	17.57	7.42	54.77	11.95	10.97	54.59	15.39
	ADMM-CatSVD	**10.05**	**60.66**	**8.80**	**6.58**	**57.89**	**8.83**	**9.69**	**59.70**	**10.28**

our method (*ADMM-CatSVD*) triggers a massive 20% absolute rebound. Furthermore, on Gemma-2 family, whose alternating attention causes standard SVD-LLM to suffer a 12.97% drop at 30% compression, our method (*ADMM-SVD*) drastically restricts the degradation to just 7.05% (PPL 7.97). Detailed per-task metrics for Gemma-2 are provided in Appendix Table 7. This remarkable recovery highlights ADMM's ability to halt cascading errors.

Impact of Compression Ratio. To evaluate robustness at different compression ratios, we scale the compression ratio from 20% to an aggressive 60% on LLaMA-2-13B (Table 1). While all methods naturally decay as the parameter budget shrinks, our approach exhibits uniquely graceful degradation. At 40% compression, baselines already falter (SkipCat's accuracy drops by 16.64%), whereas ADMM restricts the drop to 8.62% and maintains a strong perplexity of 7.65. This divergence is glaring at the extreme 60% level, where baselines suffer catastrophic semantic collapse (perplexities exploding to 34–53, accuracy nearing random guess at ∼36–37%). In stark contrast, ADMM stabilizes perplexity at 11.99 and recovers over 11 absolute percentage points in accuracy (48.73% vs. 37.30%), proving our method is essential for surviving high-sparsity regimes.

3.2 Comparison with Structured Pruning Methods

We further compare our method against three widely adopted structured pruning and slicing techniques, including LLM-Pruner, SliceGPT, and FLAP (omitted for LLaMA-3-8B due to the official repo's lack of support for the Grouped Query Attention). Unlike low-rank decomposition, which compresses models by finding efficient matrix factorizations within a lower-dimensional representations, structured pruning physically removes entire rows, columns, or attention heads. Figure 4 summarizes the performance comparison at a 30% compression ratio. Our approach demonstrates a consistent advantage over structured pruning methods across all tested models, even when those baselines are augmented with post-compression fine-tuning.

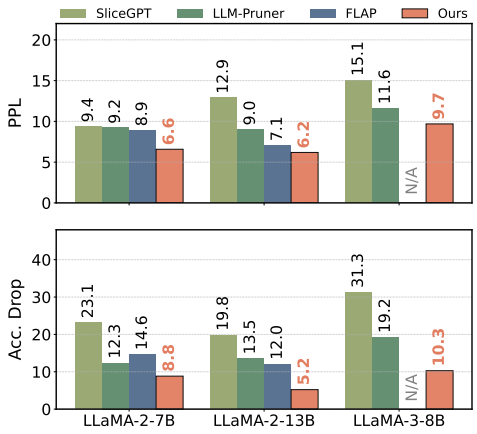


Figure 4: Comparison with structured pruning methods at 30% compression. Detailed metrics are provided in Appendix Table 8.

Physically deleting neurons often causes irreversible semantic collapse. In contrast, our low-rank mechanism preserves critical knowledge by re-projecting weights onto an optimized representations. For instance, on LLaMA-2-13B, our method restricts the average accuracy drop to a mere 5.21%, signif-

Table 3: Impact of ρ on LLaMA-3-8B at 30% compression ratio.

Penalty (ρ)	PPL ↓	Avg. Acc. ↑
0.5	12.26	52.96%
0.1	**10.97**	**54.39%**
0.01	11.42	52.10%
0.001	432.38	34.15%

Table 4: Synergy between decomposition and quantization: Ours (20%) + 4-bit achieves a lower perplexity than aggressive 3-bit quantization at a smaller memory footprint.

Method	Dense	GPTQ-4-bit	GPTQ-3-bit	Ours + 4-bit
		LLaMA-3-8B		
Memory	14.96 GB	5.4 GB	4.6 GB	**4.3 GB**
PPL (↓)	6.14	10.49	16.58	**12.58**

icantly outperforming the best zero-shot baseline, FLAP (12.03% drop). Furthermore, on densely packed models like LLaMA-3-8B, where SliceGPT and LLM-Pruner severely degrade (PPL > 11.6), our method retains a robust perplexity of 9.69 and a 59.70% reasoning average. This highlights that our approach is less destructive than pruning.

3.3 Ablation Study: Sensitivity of the Proximal Penalty (ρ)

In our ADMM formulation, the hyperparameter ρ controls the trade-off between minimizing the holistic language modeling loss and adhering to the low-rank constraint. To evaluate its impact, we aggressively compress LLaMA-3-8B at a 30% ratio under various ρ values.

As shown in Table 3, a very small ρ (e.g., 0.001 or 0.01) provides excessive freedom to the continuous weights, failing to pull them toward the low-rank representations. This results in severe structural violations upon final truncation, leading to catastrophic perplexity explosions (e.g., 432.38) and accuracy dropping to $\sim$34%. Conversely, an overly large ρ (e.g., 0.5) anchors the weights too rigidly to the initial sub-optimal SVD proxy, hindering the network's ability to dynamically compensate for cascading errors (yielding 52.96% average accuracy). We empirically find that $\rho = 0.1$ achieves the optimal balance, effectively recovering the semantics while satisfying the low-rank constraint.

3.4 Integration with Quantization

Quantization (e.g., FP16 to INT4) primarily targets numerical precision, yet realizing high-performance inference with low-bit kernels typically necessitates recent GPU architectures. In contrast, low-rank decomposition provides hardware-agnostic FLOPs reduction, which is orthogonal to the numerical compression of quantization. We demonstrate this synergy in Table 4. Combining 20% Ours with 4-bit quantization achieves a superior trade-off compared to aggressive 3-bit quantization alone. For LLaMA-3-8B, the combined model is 24% better in PPL(12.58 vs 16.58) at less memory size (4.3 GB vs. 4.6 GB).

3.5 Deployment Efficiency

Offline Compression Time. While our method introduces computational overhead compared to layer-wise methods (e.g., SVD-LLM), it remains efficient among optimization-based approaches. By decoupling the orthogonal projection ($\mathcal{Z}$-update) from the loss minimization ($\mathcal{W}$-update), we avoid executing low-rank decomposition at every step. Updating $\mathcal{Z}$ periodically (*e.g.,* every 64 steps) ensures rapid and stable convergence, the detailed dynamics of which are analyzed in Appendix E. Consequently, the additional overhead takes $\sim$2 hours for a 7B model and $\sim$5 hours for a 13B model on NVIDIA H100 (80 GB) GPUs . In contrast, recent methods like Dobi-SVD [21] require tens of GPU hours for similar scales. Given substantial performance recovery and lifelong acceleration of inference, this one-time offline cost is justifiable.

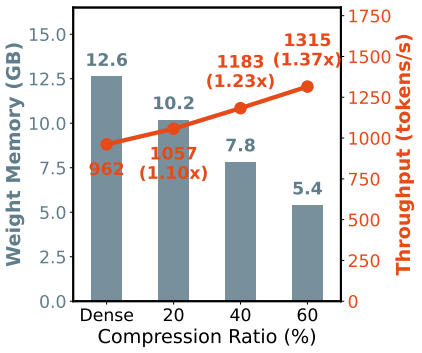


Figure 5: Inference speedup on HuggingFace Transformers.

Inference Speedup and Memory Reduction. To evaluate the practical efficiency, we benchmark the end-to-end inference throughput and memory footprint of LLaMA-2-7B on a single NVIDIA A800 (80GB) GPU. We systematically compare the Dense

FP16 baseline against our compressed models equipped with the `Fused ABX` operator across two distinct inference ecosystems: the standard HuggingFace Transformers (Batch Size=128) and the highly optimized `vLLM` engine (Batch Size=512). All evaluations use a prompt length of 128 and a generation length of 16.

As illustrated in Figure 5, our fused operator ensures that structural memory reductions translate into tangible throughput gains in general-purpose inference libraries. Even at a modest 20% compression, where sequential execution typically fails to yield speedups due to kernel overhead, our approach achieves a 1.10× speedup. As the compression ratio increases to 60%, the throughput reaches 1315.4 tokens/s (1.37×), while the weight memory is reduced significantly from 12.62 GB to 5.40 GB.

We further integrate our method into `vLLM` (Figure 1b). Accelerating a baseline already heavily optimized via PagedAttention is notoriously difficult. Nevertheless, our Fused SVD seamlessly breaks this ceiling. Compared to the Dense baseline (1488.7 tokens/s), our approach achieves 1886.8 tokens/s (1.27× speedup) at 40% compression and 2315.3 tokens/s (1.56×) at the aggressive 60% setting. These results demonstrate that the proposed system-level fusion not only benefits standard implementations but also pushes the limits of inference engines, bringing low-rank compression one step closer to real-world deployment.

4 Related Work

Early low-rank decomposition works primarily focused on minimizing weight reconstruction error in isolation. Atomic Feature Mimicking (AFM) [23] explicitly aligns the feature maps of the compressed model with the original. Building on this, Bolaco [24] employs Bayesian optimization to identify optimal rank truncation thresholds. More efficient activation-aware methods have emerged: ASVD [6] models the influence of input channels via a scaling matrix, while SVD-LLM [7, 25] establishes a direct mapping between singular values and compression loss to guide truncation. SkipCat [15] uses a concat-then-skip strategy to share projections, and Basis Sharing [9] exploits inter-layer redundancy by concatenating adjacent layers to share a common projection basis, thereby amortizing parameter storage. Another works explored dynamic rank allocation. Methods like D-Rank [26], LLRC [27], and ZS-SVD [28] focus on data-aware rank allocation across heterogeneous layers based on activation sensitivity. Other contemporaneous works, such as Swift-SVD [29], aim to provide closed-form optimal local approximations. To mitigate the resulting truncation errors, optimization-based methods like Dobi-SVD [21] further update the decomposed factors. For SVD deployment, FlashSVD [30] mitigates the issue of activation memory walls. However, all these approaches universally rely on local surrogate objectives, inevitably inducing cascading error propagation across deep architectures.

While the Alternating Direction Method of Multipliers (ADMM) [16] is a proven method for weight pruning [31, 32] and vision model tensor decompositions [33], applying ADMM to *low-rank decomposition* remains highly unexplored. By formulating decomposition as a global ADMM problem, we decouple non-convex rank constraints from the holistic modeling objective. This inherently compensates for upstream representational shifts, serving as an orthogonal catalyst for existing methods. Furthermore, we design a custom GPU operator and fuse it to widely adopted inference ecosystems, translating theoretical FLOP reductions into practical inference acceleration.

5 Conclusion

A fundamental limitation of current low-rank LLM compression is its reliance on isolated, layer-wise reconstruction. This independent strategy inevitably induces cascading error accumulation across deep architectures, serving as a critical algorithmic bottleneck that causes severe performance degradation. To address this limitation, we proposed a ADMM-based method that decouples the non-convex rank constraints from the holistic language modeling objective, allowing the network to dynamically compensate for representational shifts. Furthermore, recognizing the hardware inefficiencies of sequential memory execution, we develop a custom GPU Fused ABX operator to eliminate instruction overhead and activation memory walls. By seamlessly co-designing algorithmic global fidelity with systems-level operator fusion, our approach preserves reasoning accuracy while achieving tangible end-to-end acceleration, ultimately bringing low-rank LLM compression one step closer to real-world deployment.

6 Acknowledgement

This work is supported by the NSFC Project (No. 62306256) and the Natural Science Foundation of Guangdong Province (No. 2025A1515010261).

References

[1] An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[2] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[3] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[4] Jared Kaplan, Sam McCandlish, Tom Henighan, et al. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[5] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. Efficient large language models: A survey. *Trans. Mach. Learn. Res.*, 2024, 2024.

[6] 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.

[7] Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: truncation-aware singular value decomposition for large language model compression. *International Conference on Learning Representations*, 2025.

[8] Xinhao Huang, You-Liang Huang, and Zeyi Wen. SoLA: Leveraging soft activation sparsity and low-rank decomposition for large language model compression. In *AAAI Conference on Artificial Intelligence*, pages 17494–17502. AAAI Press, 2025.

[9] Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. Basis sharing: Cross-layer parameter sharing for large language model compression. In *International Conference on Learning Representations*. OpenReview.net, 2025.

[10] Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *AAAI Conference on Artificial Intelligence*, pages 10865–10873. AAAI Press, 2024.

[11] Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, 2023.

[12] Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. Blockpruner: Fine-grained pruning for large language models. In *ACL (Findings)*, Findings of ACL, pages 5065–5080. Association for Computational Linguistics, 2025.

[13] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*. OpenReview.net, 2023.

[14] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, 2023.

[15] Yu-Chen Lu, Sheng-Feng Yu, Hui-Hsien Weng, Pei-Shuo Wang, Yu-Fang Hu, Liang Hung-Chun, Hung-Yueh Chiang, and Kai-Chiang Wu. Skipcat: Rank-maximized low-rank compression of large language models via shared projection and block skipping. In *AAAI Conference on Artificial Intelligence*, pages 24124–24132. AAAI Press, 2026.

[16] Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, 2011.

[17] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *EMNLP (Demos)*, pages 38–45. Association for Computational Linguistics, 2020.

[18] 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 *SOSP*, pages 611–626. ACM, 2023.

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

[20] Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[21] 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*. OpenReview.net, 2025.

[22] Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari Do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*. OpenReview.net, 2024.

[23] Hao Yu and Jianxin Wu. Compressing transformers: Features are low-rank, but weights are not! In *AAAI Conference on Artificial Intelligence*, pages 11007–11015. AAAI Press, 2023.

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

[25] Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. SVD-LLM V2: optimizing singular value truncation for large language model compression. In *NAACL (Long Papers)*, pages 4287–4296. Association for Computational Linguistics, 2025.

[26] Zhendong Mi, Bian Sun, Grace Li Zhang, and Shaoyi Huang. Layer-wise dynamic rank for compressing large language models. *arXiv preprint arXiv:2509.25622*, 2025.

[27] Sidhant Sundrani, Francesco Tudisco, and Pasquale Minervini. Low-rank compression of language models via differentiable rank selection. *arXiv preprint arXiv:2512.13733*, 2025.

[28] Ali Abbasi, Chayne Thrash, Haoran Qin, Shansita Sharma, Sepehr Seifi, and Soheil Kolouri. Zero sum svd: Balancing loss sensitivity for low rank llm compression. *arXiv preprint arXiv:2602.02848*, 2026.

[29] Ruoling Qi et al. Swift-svd: Theoretical optimality meets practical efficiency in low-rank llm compression. *arXiv preprint arXiv:2604.01609*, 2026.

[30] Zishan Shao, Yixiao Wang, Qinsi Wang, Ting Jiang, Zhixu Du, Hancheng Ye, Danyang Zhuo, Yiran Chen, and Hai (Helen) Li. Flashsvd: Memory-efficient inference with streaming for low-rank models. In *AAAI*, pages 25278–25285. AAAI Press, 2026.

[31] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic DNN weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 191–207, 2018.

[32] Kwanhee Lee, Hyeondo Jang, Dongyeop Lee, Dan Alistarh, and Namhoon Lee. The unseen frontier: Pushing the limits of LLM sparsity with surrogate-free ADMM. *arXiv preprint arXiv:2510.01650*, 2025.

[33] Miao Yin, Yang Sui, Siyu Liao, and Bo Yuan. Towards efficient tensor decomposition-based DNN model compression with optimization framework. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10683, 2021.

A Theoretical Analysis of Cascading Error Amplification

In Section 2, we identified that layer-wise surrogate objectives suffer from cascading error propagation. To formalize this flaw, we establish the theoretical upper bound of the representation error induced by layer-wise optimization.

Proposition 1 (Cascading Error Amplification of Layer-wise Methods)**.** *Let X_0 be the initial input to an L-layer architecture. Assume each non-linear layer module function ϕ_i is K-Lipschitz continuous. If a greedy layer-wise decomposition induces a localized truncation error $\|\phi_i(X_i) - \hat{\phi}_i(X_i)\|_F \leq \epsilon$ at each layer i, the final representation deviation at layer L, denoted as $\Delta X_L = \|X_L - \hat{X}_L\|_F$, is bounded by:*

$$\Delta X_L \leq \epsilon \sum_{j=0}^{L-1} K^j = \mathcal{O}(\epsilon K^L) \tag{9}$$

Proof. Let the original uncompressed neural network be denoted by function compositions $X_i = \phi_i(X_{i-1}, W_i)$. Let the compressed network be denoted by $\hat{X}_i = \phi_i(\hat{X}_{i-1}, \hat{W}_i)$. Given the global K-Lipschitz continuity of the activation mapping ϕ_i, for any inputs X and Y:

$$\|\phi_i(X) - \phi_i(Y)\|_F \leq K\|X - Y\|_F \tag{10}$$

In a greedy layer-wise decomposition, the optimal local truncation error is bounded by ϵ assuming unperturbed inputs:

$$\|\phi_i(X_{i-1}, W_i) - \phi_i(X_{i-1}, \hat{W}_i)\|_F \leq \epsilon \tag{11}$$

In actual inference, the compressed layer receives the perturbed input $\hat{X}_{i-1}$. Using the triangle inequality, we bound the deviation:

$$\begin{aligned}
\Delta X_i &= \|X_i - \hat{X}_i\|_F \\
&= \|\phi_i(X_{i-1}, W_i) - \phi_i(\hat{X}_{i-1}, \hat{W}_i)\|_F \\
&\leq \underbrace{\|\phi_i(X_{i-1}, W_i) - \phi_i(X_{i-1}, \hat{W}_i)\|_F}_{\text{Local Truncation Error} \leq \epsilon} + \underbrace{\|\phi_i(X_{i-1}, \hat{W}_i) - \phi_i(\hat{X}_{i-1}, \hat{W}_i)\|_F}_{\text{Cascaded Upstream Error}} \\
&\leq \epsilon + K\|X_{i-1} - \hat{X}_{i-1}\|_F \\
&\leq \epsilon + K\Delta X_{i-1}
\end{aligned} \tag{12}$$

Unrolling this recursive inequality from layer L down to the first layer ($\Delta X_0 = 0$), we obtain $\Delta X_L \leq \epsilon \sum_{j=0}^{L-1} K^j$. In modern deep LLMs, weight matrices typically possess spectral norms (and thus Lipschitz constants) $K > 1$ to maintain signal variance and prevent vanishing representations across extensive layers. Given $K > 1$, this summation forms a geometric series that bounds the terminal error at $\mathcal{O}(\epsilon K^L)$, mathematically demonstrating the fatal exponential amplification inherent to localized surrogate objectives.

Proposition 2 (Decoupling Cascading Error via ADMM)**.** *While layer-wise surrogate decomposition accumulates representation error exponentially bounded by $\mathcal{O}(\epsilon K^L)$, the proposed ADMM formulation actively decouples this error from the network depth L.*

Proof. In our ADMM framework, the continuous weights $\mathcal{W}$ are jointly updated to align with the holistic loss $f(\mathcal{W})$ in the $\mathcal{W}$-update step. Assume the global loss function $f(\mathcal{W})$ explicitly bounds the final representation discrepancy, i.e., there exists a constant $\mu > 0$ such that the loss gap bounds the terminal representation error: $\mu\|X_L - \hat{X}_L\|_F \leq f(\hat{\mathcal{W}}) - f(\mathcal{W}^*)$.

When the $\mathcal{W}$-update converges to a local stationary point with a bounded loss degradation δ (i.e., $f(\hat{\mathcal{W}}) - f(\mathcal{W}^*) \leq \delta$), the final layer representation error is implicitly bounded by the optimization landscape:

$$\Delta X_L = \|X_L - \hat{X}_L\|_F \leq \frac{\delta}{\mu}$$

Crucially, this bound $\frac{\delta}{\mu}$ is determined exclusively by the global convergence quality (δ) and the loss sensitivity (μ), completely bypassing the depth-dependent exponential amplification factor K^L. This guarantees that our method inherently suppresses inter-layer error growth.

B Algorithmic Details and Orthogonal Integrations

In this section, we provide the complete procedural workflow of our ADMM-based optimization and the explicit mathematical formulations for its integration with advanced low-rank techniques.

B.1 The Algorithm Overall Procedure

The overall optimization process is summarized in Algorithm 1. The framework decouples the global objective into three iterative subproblems, where the $\mathcal{Z}$-update functions as a flexible projection operator $\Pi_{\mathcal{R}}$ capable of adapting to different statistical and structural priors.

Algorithm 1 Global Low-Rank Optimization via ADMM

Input: Pre-trained dense LLM $\bar{\mathcal{W}}$, target ranks $\{r_i\}$, calibration data $\mathcal{D}$, penalty ρ.
Initialize: $W_i^0 \leftarrow \bar{W}_i$, $Z_i^0 \leftarrow \text{SVD}(\bar{W}_i, r_i)$, $U_i^0 \leftarrow \mathbf{0}$ for all layers $i \in \{1, \ldots, L\}$.
for $t = 0, 1, \ldots, T-1$ **do**
  1. Global Representation Anchoring ($\mathcal{W}$-update):
  Update W_i^{t+1} by minimizing $\mathcal{L}_\rho(\mathcal{W}, \mathcal{Z}^t, \mathcal{U}^t)$ via backpropagation on $f(\mathcal{W})$.
  2. Orthogonal Projection ($\mathcal{Z}$-update):
  for each group $\mathcal{G}$ or individual layer i **do**
    if Integration == *SVD-LLM* **then** ▷ Activation-Aware Projection
      Compute covariance $H_i = \mathbb{E}[X_i^\top X_i]$.
      $\tilde{M}_i \leftarrow (W_i^{t+1} + U_i^t) H_i^{1/2}$.
      $Z_i^{t+1} \leftarrow \text{TruncatedSVD}(\tilde{M}_i, r_i) H_i^{-1/2}$.
    else if Integration == *SkipCat* **then** ▷ Structural Sharing
      $M_{\text{group}} \leftarrow \text{Concat}_{j \in \mathcal{G}}(W_j^{t+1} + U_j^t)$ along output dimension.
      $U_{\text{shared}} \leftarrow \text{LeftSingularVectors}(\text{SVD}(M_{\text{group}}, r_{\text{shared}}))$.
      $Z_j^{t+1} \leftarrow U_{\text{shared}} U_{\text{shared}}^\top (W_j^{t+1} + U_j^t)$ for each $j \in \mathcal{G}$.
    else ▷ Vanilla SVD
      $Z_i^{t+1} \leftarrow \text{TruncatedSVD}(W_i^{t+1} + U_i^t, r_i)$.
    end if
  end for
  3. Violation Accumulation ($\mathcal{U}$-update):
  $U_i^{t+1} \leftarrow U_i^t + (W_i^{t+1} - Z_i^{t+1})$ for all layers i.
end for
Output: decomposed weights $\hat{W}_i = \text{SVD}(Z_i^T, r_i)$.

B.2 Mathematical Formulation for SVD-LLM Integration

SVD-LLM [7] optimizes the weighted Frobenius norm $\|W - \hat{W}\|_H^2$ to account for outlier features in LLM activations. In our framework, this is equivalent to solving the $\mathcal{Z}$-update in a feature-whitened coordinate system. For each layer i, let $H_i = \mathbb{E}[X_i^\top X_i]$. The projection step is executed as follows:

1. **Whitening:** Transform the shifted matrix: $\tilde{M}_i = (W_i^{t+1} + U_i^t)H_i^{1/2}$.
2. **Geometric Projection:** Perform Truncated SVD on $\tilde{M}_i$ and retain the top-r_i components: $\tilde{Z}_i = P_{r_i}\Sigma_{r_i}Q_{r_i}^\top$.
3. **Un-whitening:** Transform back to the original representations: $Z_i^{t+1} = \tilde{Z}_i H_i^{-1/2}$.

B.3 Mathematical Formulation for SkipCat Integration

SkipCat [15] enforces functional heads (e.g., W_Q, W_K, W_V) sharing the same input stream to reside within a common subspace. We incorporate this via **Grouped Subspace Projection**:

1. **Horizontal Concatenation:** Align shifted weights and dual variables: $M_{\text{group}} = \left[(W_j^{t+1} + U_j^t)\right]_{j\in\mathcal{G}}$.
2. **Joint Subspace Extraction:** Compute shared basis U_{shared} via SVD on M_{group}.
3. **Disentangled Reconstruction:** Project individual heads onto the shared basis: $Z_j^{t+1} = U_{\text{shared}}U_{\text{shared}}^\top(W_j^{t+1} + U_j^t), \forall j \in \mathcal{G}$.

C Implementation Details

Our framework seamlessly bridges algorithmic optimization with systems-level deployment. Below, we detail the configurations for both the offline ADMM compression and the hardware-aware inference evaluation.

Calibration Data and Evaluation. The offline compression procedures utilize a lightweight calibration set consisting of 2,048 sequences randomly sampled from the Wikitext2 and C4 dataset, with a sequence length of 2,048 tokens. The activation-aware baselines (including SVD-LLM, Basis Sharing, and SkipCat) and our proposed method utilize the same calibration dataset. For zero-shot commonsense reasoning evaluations (e.g., ARC, PIQA, HellaSwag), we adopt the standard `lm-evaluation-harness` framework to ensure fair comparisons with all baselines.

ADMM Optimization Configurations. Our method is implemented in PyTorch. Models at the 7B–9B scale are optimized on a single GPU, whereas larger architectures (e.g., 13B) are scaled across multiple GPUs using Fully Sharded Data Parallel (FSDP). The continuous weights are optimized using the 8-bit Adam optimizer (`adam8bit`) to further reduce optimizer state memory. The peak learning rate is set to 2×10^{-4}. For the ADMM-specific hyperparameters, we maintain a constant proximal penalty of $\rho = 0.1$ across all model scales. The orthogonal projection ($\mathcal{Z}$-update) and dual variable accumulation ($\mathcal{U}$-update) are executed periodically every 64 steps.

Hardware and Inference Ecosystem. The offline compression and ADMM-based calibration are executed on a high-performance compute node equipped with NVIDIA H100 (80GB) GPUs. To demonstrate practical deployability, all end-to-end inference throughput benchmarks are conducted on a single NVIDIA A800 (80GB) GPU, representing a standard high-end inference environment. The custom `Fused-ABX` kernel is implemented in CUDA C++ and optimized for Tensor Core execution. For seamless system integration, the kernel is exposed as a PyTorch C++ extension and integrated into both the HuggingFace Transformers pipeline and the `vLLM` inference engine. This allows our low-rank layers to leverage vLLM's continuous batching and PagedAttention mechanisms, ensuring that our operator-level speedups translate into improved system-level throughput.

D Detailed Experimental Results

In this section, we provide the exhaustive, per-task performance breakdowns that complement the aggregated results discussed in the main manuscript.

Table 5: Supplemental performance comparison of Table 1 on LLaMA-2-13B (30% and 50% compression ratios). The *Drop* column highlights the absolute accuracy decrease compared to the Dense baseline. Consistent with the main text, we demonstrate the plug-and-play nature of our framework by applying it to both the standard SVD paradigm (*ADMM-SVD*) and the structurally shared paradigm (*ADMM-CatSVD*).

Ratio	Methods	PPL ↓	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	WinoG.	Avg. ↑	Drop ↓
0%	Dense	4.88	49.15	77.61	80.70	79.35	45.80	80.69	72.06	69.33	-
30%	ASVD	23.03	28.50	48.32	68.59	48.74	35.20	64.42	62.27	50.86	18.47
	SVD-LLM	8.28	33.28	60.56	70.83	53.89	39.60	69.48	66.30	56.28	13.05
	Dobi-SVD	6.56	38.05	64.39	73.15	64.49	41.60	73.29	67.56	60.36	8.97
	BasisSharing	7.97	34.13	62.29	66.42	52.57	39.40	67.57	65.27	55.38	13.95
	SkipCat	6.82	40.10	67.05	68.01	61.64	41.20	73.34	**69.14**	60.07	9.26
	ADMM-SVD	6.50	38.48	65.15	73.70	67.55	40.00	73.94	65.75	60.65	8.68
	ADMM-CatSVD	**6.19**	**43.09**	**69.15**	**76.88**	**73.80**	**44.00**	**75.63**	66.30	**64.12**	**5.21**
50%	ASVD	3021	26.71	26.56	37.95	26.47	26.20	51.41	50.75	35.15	34.18
	SVD-LLM	21.06	24.49	38.26	46.64	35.05	31.20	59.03	56.51	41.60	27.73
	Dobi-SVD	11.25	25.77	42.17	59.24	39.60	31.80	60.99	59.27	45.55	23.78
	BasisSharing	17.71	24.66	39.73	39.30	35.08	28.80	56.75	58.80	40.45	28.88
	SkipCat	15.52	25.00	39.60	60.37	37.58	30.20	60.39	57.46	44.37	24.96
	ADMM-CatSVD	**7.69**	**32.51**	**55.60**	**68.50**	**59.40**	**35.20**	**70.13**	**62.12**	**54.78**	**14.55**

D.1 Comprehensive Results for Low-Rank Decomposition

To complement the compression trend analysis in the main manuscript, Table 5 details the performance of LLaMA-2-13B at intermediate compression ratios of 30% and 50%. Consistent with our primary findings, baseline methods suffer from severe performance cliffs as the compression ratio escalates (e.g., ASVD collapses at 50% with a perplexity of 3021). In contrast, our ADMM framework consistently rescues the model, demonstrating its robust capacity.

Table 6 presents the detailed zero-shot commonsense reasoning metrics for various low-rank decomposition methods (including ASVD, Dobi-SVD, BasisSharing, SVD-LLM, and SkipCat) across three distinct LLM architectures (Qwen-3-8B, LLaMA-2-7B, and LLaMA-3-8B) at both 20% and 30% compression ratios. The results clearly illustrate that integrating our method consistently rescues these baseline methods from severe localized truncation degradation, systematically improving both perplexity and individual reasoning task accuracies across all settings.

To complement the generalization analysis presented in the main text, Table 7 provides the exhaustive, per-task performance breakdown for the Gemma-2 family (9B and 27B) at 20% and 30% compression ratios. The Gemma-2 architecture incorporates complex topological advancements, such as alternating local/global attention mechanisms and logit soft-capping, making it notoriously difficult to compress using standard greedy methodologies. As observed in the detailed metrics, the layer-wise truncation of the standalone SVD-LLM leads to significant performance drops across complex reasoning tasks (e.g., ARC-c and PIQA) at higher sparsity levels. However, by orchestrating the decomposition through ADMM (*ADMM-SVD*), the network effectively coordinates cross-layer compensations. This results in a systematic and robust recovery across almost all individual datasets, further validating the necessity and universality of our method for modern, hard-to-manipulate LLMs.

D.2 Detailed Comparison with Structured Pruning

In Section 3.2 of the main text, we provided a visual summary of the performance comparison between our ADMM-guided low-rank framework and various structured pruning methodologies (SliceGPT, LLM-Pruner, and FLAP) at a 30% compression ratio.

Table 8 presents the exhaustive, per-task breakdown of these zero-shot commonsense reasoning evaluations. As shown in the table, structured pruning methods physically remove network components, which often leads to severe degradation on complex reasoning tasks (e.g., ARC-c and PIQA). In contrast, our approach retains the dense topological skeleton and re-projects weights onto an optimized low-rank representations, thereby uniformly protecting performance across all individual reasoning datasets.

Table 6: Supplemental performance comparison of Table 2. *ADMM-SVD* and *ADMM-CatSVD* represent the application of our ADMM-based method to the SVD-LLM and SkipCat baselines, respectively. This demonstrates the orthogonal and consistent performance gains provided by our approach across diverse architectures.

Ratio	Methods	PPL ↓	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	WinoG.	Avg. ↑	Drop ↓
Qwen-3-8B											
0%	Dense	9.85	56.57	80.81	86.54	74.97	42.00	77.58	67.80	69.46	-
20%	ASVD	21.29	41.89	63.17	71.68	60.20	36.60	72.42	59.59	57.94	11.52
	Dobi-SVD	14.00	37.80	54.30	66.17	56.03	36.00	68.88	63.69	54.70	14.76
	BasisSharing	14.91	42.75	64.31	79.58	61.99	38.80	71.33	65.59	60.62	8.83
	SVD-LLM	13.84	39.76	58.42	79.17	54.80	36.60	69.37	60.93	57.01	12.45
	SkipCat	12.12	**50.17**	**76.43**	**80.80**	65.23	**41.60**	74.27	66.06	64.94	4.52
	ADMM-SVD	9.99	42.06	64.44	71.56	59.75	38.80	70.78	62.75	58.59	10.87
	ADMM-CatSVD	**8.64**	49.74	72.77	76.12	**69.27**	42.40	**75.73**	**68.82**	**64.98**	**4.48**
30%	ASVD	116.71	26.19	40.87	61.62	37.17	25.80	61.70	52.17	43.65	25.81
	Dobi-SVD	18.25	33.79	50.21	71.99	49.73	35.40	63.11	61.40	52.23	17.23
	BasisSharing	18.42	37.20	55.93	77.80	53.21	35.00	66.49	61.80	55.35	14.11
	SVD-LLM	20.15	28.50	45.71	63.49	41.57	30.00	62.57	55.09	46.70	22.76
	SkipCat	14.82	39.68	64.65	**78.59**	55.33	34.80	68.23	62.27	57.65	11.81
	ADMM-SVD	12.35	34.47	52.65	63.27	51.33	34.60	66.87	60.06	51.89	17.57
	ADMM-CatSVD	**10.05**	**45.65**	**69.87**	73.00	**64.25**	**36.00**	**72.91**	**62.98**	**60.66**	**8.80**
LLaMA-2-7B											
0%	Dense	5.46	46.25	74.49	77.77	75.92	44.20	79.05	69.38	66.72	-
20%	ASVD	9.04	36.43	63.51	73.24	65.73	39.00	73.83	66.22	59.71	7.01
	Dobi-SVD	9.39	29.18	47.64	55.11	51.55	33.00	66.27	58.48	48.75	17.97
	BasisSharing	7.97	36.18	64.98	67.16	60.40	37.80	73.78	64.48	57.83	8.89
	SVD-LLM	8.78	33.70	55.39	64.13	56.26	37.60	71.16	61.96	54.31	12.41
	SkipCat	6.30	**42.15**	**70.50**	**71.53**	69.31	40.60	76.39	**66.61**	62.44	4.28
	ADMM-SVD	6.47	37.20	60.44	69.48	65.32	40.60	74.21	63.77	58.72	8.00
	ADMM-CatSVD	**5.76**	41.98	69.23	71.56	**72.39**	**43.00**	**76.99**	64.96	**62.87**	**3.85**
30%	ASVD	541.87	25.34	27.57	43.82	28.47	25.80	51.20	53.12	36.47	30.25
	Dobi-SVD	12.38	26.88	39.65	44.22	42.57	31.00	66.27	58.88	44.21	22.51
	BasisSharing	10.32	32.00	58.71	63.67	51.75	36.20	70.02	61.88	53.46	13.26
	SVD-LLM	11.66	27.99	47.94	63.58	46.87	35.20	66.10	62.59	50.03	16.69
	SkipCat	7.77	35.67	63.13	63.09	58.08	38.40	72.58	62.75	56.24	10.48
	ADMM-SVD	7.42	33.62	57.66	68.17	57.20	37.40	67.95	61.40	54.77	11.95
	ADMM-CatSVD	**6.58**	**37.20**	**61.49**	**67.13**	**63.23**	**40.00**	**72.20**	**64.01**	**57.89**	**8.83**
LLaMA-3-8B											
0%	Dense	6.14	53.33	77.61	80.95	79.13	45.00	80.85	73.01	69.98	-
20%	ASVD	1041	24.15	35.65	52.81	33.27	27.80	56.58	53.04	40.47	29.51
	Dobi-SVD	14.93	30.38	49.12	63.39	49.39	34.40	64.42	63.06	50.59	19.39
	BasisSharing	14.67	38.74	64.90	69.66	58.80	38.40	71.11	69.30	58.70	11.28
	SVD-LLM	19.11	38.23	59.60	66.76	54.36	37.40	68.77	65.59	55.82	14.16
	SkipCat	13.00	32.08	57.49	65.78	57.10	36.60	69.80	65.51	54.91	15.07
	ADMM-SVD	9.38	38.74	63.30	74.31	63.79	38.20	74.05	65.98	59.77	10.21
	ADMM-CatSVD	**8.21**	**42.49**	**65.15**	**71.10**	**69.72**	**41.20**	**76.55**	**67.88**	**62.01**	**7.97**
30%	ASVD	23174	24.83	26.26	40.21	26.34	27.40	50.76	50.28	35.15	34.83
	Dobi-SVD	31.13	26.37	36.49	49.91	37.27	29.80	55.88	59.51	42.18	27.80
	BasisSharing	25.95	26.79	46.72	49.60	41.43	31.00	61.37	60.14	45.29	24.69
	SVD-LLM	43.91	29.61	46.09	63.21	39.35	29.40	61.75	59.43	46.98	23.00
	SkipCat	48.24	22.01	35.52	39.14	33.96	31.60	57.34	57.93	39.64	30.34
	ADMM-SVD	10.97	33.19	53.54	71.93	55.99	35.00	69.10	63.38	54.59	15.39
	ADMM-CatSVD	**9.69**	**37.54**	**62.54**	**75.29**	**65.14**	**39.60**	**71.06**	**66.77**	**59.70**	**10.28**

D.3 Evaluation on MMLU Benchmark

In addition to zero-shot commonsense reasoning tasks, we further evaluate the performance of our compressed models on the Massive Multitask Language Understanding (MMLU) benchmark. MMLU covers 57 diverse subjects (e.g., STEM, humanities, and social sciences). We compare our *ADMM-SVD* framework against the standard *SVD-LLM* baseline at a 30% compression ratio on both the LLaMA-2-7B and LLaMA-2-13B architectures. As shown in Table 9, our global optimization framework consistently yields superior knowledge retention.

Specifically, on LLaMA-2-7B, ADMM-SVD achieves an MMLU score of 29.2%, mitigating the degradation observed in the baseline (27.6%). This improvement is more pronounced on the larger LLaMA-2-13B model, where ADMM-SVD achieves 36.7%, outperforming the baseline's 31.2% by a substantial absolute margin of 5.5%. This performance gap further underscores that decoupling the rank constraints from the holistic modeling objective via ADMM effectively safeguards the deep

Table 7: Detailed metrics for the Gemma-2 family (9B and 27B) at 20% and 30% compression ratios.

Ratio	Methods	PPL ↓	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	WinoG.	Avg. ↑	Drop ↓
Gemma-2-9B											
0%	Dense	6.85	65.96	88.09	84.13	79.92	47.00	82.92	73.64	74.52	-
20%	SVD-LLM	11.18	48.98	77.36	79.66	60.72	42.00	74.54	68.35	64.52	10.00
	ADMM-SVD	**8.56**	**52.47**	**80.35**	**81.77**	**71.40**	**43.00**	**77.42**	**70.24**	**68.09**	**6.43**
30%	SVD-LLM	15.67	40.10	67.42	69.02	50.11	35.40	69.86	63.69	60.80	13.72
	ADMM-SVD	**9.42**	**48.55**	**76.89**	**76.91**	**66.48**	**40.20**	**75.41**	**67.56**	**64.57**	**9.95**
Gemma-2-27B											
0%	Dense	5.69	65.96	88.13	84.68	84.89	47.60	84.49	79.40	76.45	-
20%	SVD-LLM	8.42	56.06	81.40	76.97	73.79	44.20	78.40	75.37	69.46	6.99
	ADMM-SVD	**7.47**	**58.87**	**83.04**	**81.77**	**78.15**	**43.00**	**80.20**	**73.64**	**71.24**	**5.21**
30%	SVD-LLM	10.81	48.55	77.15	65.35	65.13	42.40	75.14	70.64	63.48	12.97
	ADMM-SVD	**7.97**	**54.86**	**81.40**	**80.24**	**76.01**	**42.80**	**79.22**	**71.27**	**69.40**	**7.05**

Table 8: Performance comparison of structural pruning methods.

Ratio	Methods	PPL ↓	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	WinoG.	Avg. ↑	Drop ↓
LLaMA-2-7B											
0%	Dense	5.46	46.25	74.49	77.77	75.92	44.20	79.05	69.38	66.72	-
30%	SliceGPT	9.39	28.84	43.43	47.43	40.72	29.20	58.00	57.54	43.59	23.13
	LLM-Pruner	9.21	32.76	56.69	54.53	63.53	37.00	74.70	61.48	54.38	12.34
	FLAP	8.91	32.25	55.18	52.20	56.58	38.20	70.29	60.06	52.11	14.61
	ADMM-CatSVD	**6.58**	**37.20**	**61.49**	**67.13**	**63.23**	**40.00**	**72.20**	**64.01**	**57.89**	**8.83**
LLaMA-2-13B											
0%	Dense	4.88	49.15	77.61	80.70	79.35	45.80	80.69	72.06	69.33	-
30%	SliceGPT	12.90	31.23	49.79	55.54	50.32	36.80	62.02	61.09	49.54	19.79
	LLM-Pruner	9.01	33.96	55.68	64.01	64.14	40.20	75.57	57.38	55.85	13.48
	FLAP	7.08	37.29	61.45	64.37	62.44	39.20	72.42	63.93	57.30	12.03
	ADMM-CatSVD	**6.19**	**43.09**	**69.15**	**76.88**	**73.80**	**44.00**	**75.63**	**66.30**	**64.12**	**5.21**
LLaMA-3-8B											
0%	Dense	6.14	53.33	77.61	80.95	79.13	45.00	80.85	73.01	69.98	-
30%	SliceGPT	15.08	24.57	35.52	38.07	36.44	28.60	53.81	53.99	38.71	31.27
	LLM-Pruner	11.62	30.12	50.67	55.02	56.36	32.60	71.22	59.59	50.80	19.18
	ADMM-CatSVD	**9.69**	**37.54**	**62.54**	**75.29**	**65.14**	**39.60**	**71.06**	**66.77**	**59.70**	**10.28**

factual knowledge embedded within the dense network, which is otherwise irreversibly destroyed by greedy layer-wise truncation.

Table 9: Performance comparison on the MMLU benchmark at a 30% compression ratio. The results demonstrate that our global optimization approach better preserves factual knowledge compared to standard layer-wise truncation.

Model	SVD-LLM	ADMM-SVD
LLaMA-2-7B (30%)	27.6%	**29.2%**
LLaMA-2-13B (30%)	31.2%	**36.7%**

E Optimization Dynamics

While our method naturally introduce computational overhead compared to one-shot layer-wise methods, our ADMM formulation is specifically designed to be stable and computationally tractable.

A common concern with alternating optimization on non-convex deep architectures is the risk of divergence or severe oscillation. However, our empirical tracking of the low-rank proxy ($\mathcal{Z}$) demonstrates stable convergence. Over the 1,024-step optimization across diverse models, we observe two distinct phenomena (Figure 6). First, the primal residual ($||\mathcal{W} - \mathcal{Z}||_F$) decays exponentially, plunging from massive initial discrepancies down to single-digit values. Second, the validation loss of the low-rank proxy smoothly converges to a competitive level. This demonstrates that our "elastic decoupling" strategy successfully guides the network to a high-quality semantic minimum.

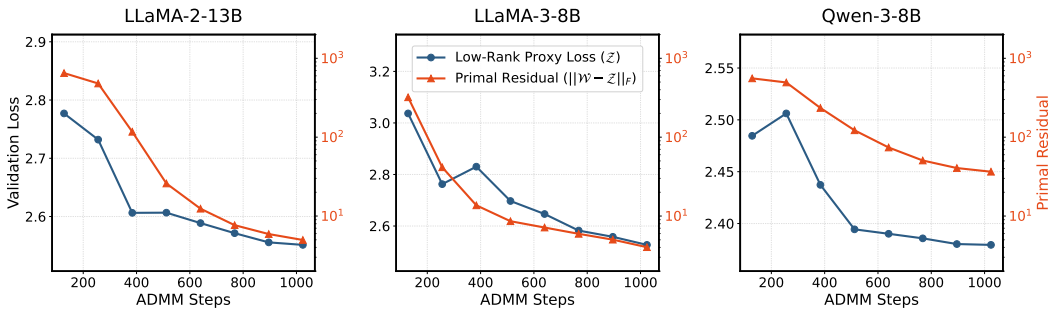


Figure 6: Convergence trajectories across diverse LLM architectures over 1,024 ADMM steps. The left axis (linear) tracks the validation loss of the low-rank proxy (blue). The right axis (logarithmic) tracks the primal residual (orange). As the optimization proceeds, the primal residual decays exponentially, while the proxy losses prove stable convergence to a high-fidelity low-rank consensus.

F Limitations and Future Work

While our ADMM-based framework significantly mitigates cascading error propagation and bridges the gap between theoretical compression and practical acceleration, it is not without limitations.

Limitations. First, low-rank decomposition inherently reduces the model's parameter capacity. Although our global perspective recovers substantial representational fidelity, aggressive compression ratios (e.g., beyond 50%) inevitably lead to noticeable performance degradation. Second, formulating the decomposition as a global ADMM optimization introduces a higher offline calibration cost compared to one-shot, layer-wise SVD methods. While this one-time cost is entirely amortized by the lifelong efficiency gains during deployment, reducing this overhead remains an area for improvement.

Future Work. Low-rank decomposition mathematically reduces structural parameters, making it fundamentally orthogonal to numerical precision compression techniques. A promising direction for future work is to explore joint optimization strategies that seamlessly integrate our global ADMM low-rank formulation with ultra-low-bit quantization (e.g., 2-bit or 3-bit quantization). We hypothesize that such a synergistic approach could yield compounded efficiency gains, pushing the boundaries of deploying massive LLMs on heavily resource-constrained edge devices. Additionally, extending our uniform rank constraint to a dynamic, activation-aware rank allocation strategy under the ADMM framework could further enhance the parameter-to-performance efficiency.

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 clearly state our core contributions: identifying the cascading error flaw in layer-wise compression, proposing the global ADMM framework, and developing the fused GPU operator. These claims are supported by theoretical proofs in Appendix A and empirical evaluations in Section 3.

   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 limitations of our approach, such as the additional offline time introduced by the iterative ADMM optimization process.

   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 theoretical bound for cascading error amplification and its associated proofs are fully detailed in Appendix A.

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: We detail the calibration datasets, sequence lengths, evaluation frameworks (`lm-evaluation-harness`), and optimization hyperparameters (e.g., $\rho = 0.1$, $T = 1024$) in Section 3 and Appendix B to ensure reproducibility.

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 anonymized source code for both the ADMM optimization framework and the custom CUDA kernel for the Fused ABX operator is included in the supplementary material.

   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: Full experimental details, including calibration data sampling (Wikitext2, C4), optimizer choice (`adam8bit`), and hyperparameters are provided in Appendix B.

   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: [No]

   Justification: We report perplexity and zero-shot accuracy on standard deterministic LLM benchmarks using a fixed random seed. Under a fixed seed, the results across runs are identical, effectively eliminating cross-run variance; thus, error bars are not explicitly reported, which aligns with standard practices in the LLM compression literature.

   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: The hardware specifications for both the offline ADMM optimization (NVIDIA H100) and the end-to-end inference latency benchmarking (NVIDIA A800) are explicitly documented in Appendix B.

   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: The research conforms to the NeurIPS Code of Ethics. Our work focuses on improving the computational efficiency of existing language models.

   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: We briefly discuss the broader impacts, noting that while our method democratizes LLM deployment by significantly reducing hardware requirements, compressed models inherently inherit the societal biases of their uncompressed foundational models.

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: This work focuses on an algorithmic compression framework and a custom GPU operator. We do not release new high-risk foundational models or newly scraped datasets.

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: We utilize publicly available models (e.g., LLaMA-2, Qwen) and calibration datasets. We properly cite these assets and respect their respective open-source licenses.

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: [Yes]

Justification: The newly introduced code assets (the ADMM compression logic and the Fused ABX kernel) are documented in the supplementary material with clear instructions for compilation and integration.

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: This research does not involve crowdsourcing or 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: This research does not involve human subjects.

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 strictly for editing and formatting text.

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.