WHY SMALLER IS SLOWER: DIMENSIONAL MISALIGNMENT IN COMPRESSED LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dimension-reducing compression techniques—such as low-rank factorization and structured pruning—are widely used to shrink Large Language Models (LLMs) for efficient deployment. Paradoxically, compressed models often run *no faster*, or even *slower*, despite having 15–30% fewer parameters. We trace this to *dimensional misalignment*: irregular tensor shapes from importance-based rank allocation conflict with GPU execution primitives at the hardware, library, and framework levels. Analyzing four importance proxies on Llama-3-8B, we show that 53–81% of allocated dimensions are misaligned, with penalties compounding multiplicatively across the execution stack. We propose *GPU-Aligned Compression* (GAC), a compressor-agnostic wrapper that reformulates budget allocation as a hardware-constrained Multi-Choice Knapsack Problem, achieving 100% alignment while retaining 6–7% more cumulative importance than unconstrained allocation with sub-200 ms overhead. Microbenchmark-aggregated latency estimates show GAC-wrapped models recover up to 1.25× speedup versus 1.15× for natively compressed baselines.

1 INTRODUCTION

Large Language Models (LLMs) achieve remarkable performance but are bottlenecked by memory and latency at deployment. Post-training compression—spanning quantization (Frantar et al., 2023; Lin et al., 2025), sparsification (Frantar & Alistarh, 2023; Sun et al., 2024), and dimension reduction (Yuan et al., 2025; Wang et al., 2025b)—is the dominant mitigation strategy (Zhu et al., 2024). Dimension-reducing methods (SVD, structured pruning) are especially attractive because they physically shrink tensors, reducing both memory and FLOPs without sparse hardware.

However, a troubling paradox has emerged: compressed models with significantly fewer parameters frequently exhibit *no wall-clock speedup*—and sometimes run slower. For instance, applying ASVD (Yuan et al., 2025) to Llama-3-8B at 20% compression yields 15% fewer parameters, yet inference latency remains unchanged.

We identify the root cause as *dimensional misalignment*: importance-based rank allocation produces arbitrary integers (e.g., 3413 instead of 3424) that conflict with GPU hardware at three levels. Tensor Cores require dimensions divisible by 8 or 16; cuBLAS selects suboptimal kernels for irregular shapes; and PyTorch's SDPA rejects non-power-of-2 head dimensions, forcing fallback from FlashAttention to the Math backend with up to 90% overhead. These penalties compound multiplicatively, creating a paradox where theoretical FLOP reductions yield no practical speedup.

Our contributions are:

1. **Prevalence of misalignment.** The first systematic analysis across four importance proxies and four compression ratios on Llama-3-8B, revealing 53–81% misalignment (mod 8), worsening at higher compression.
2. **Full-stack root cause analysis.** We trace compounding penalties from Tensor Core tile geometry through cuBLAS kernel dispatch to PyTorch SDPA backend selection.

3. **GPU-Aligned Compression (GAC).** A compressor-agnostic wrapper reformulating allocation as a hardware-constrained MCKP, achieving 100% alignment with 6–7% more importance retention and sub-200 ms overhead.

4. **Compressor-agnostic evaluation.** We wrap both ASVD and LLM-Pruner with GAC, showing consistent quality and speedup gains.

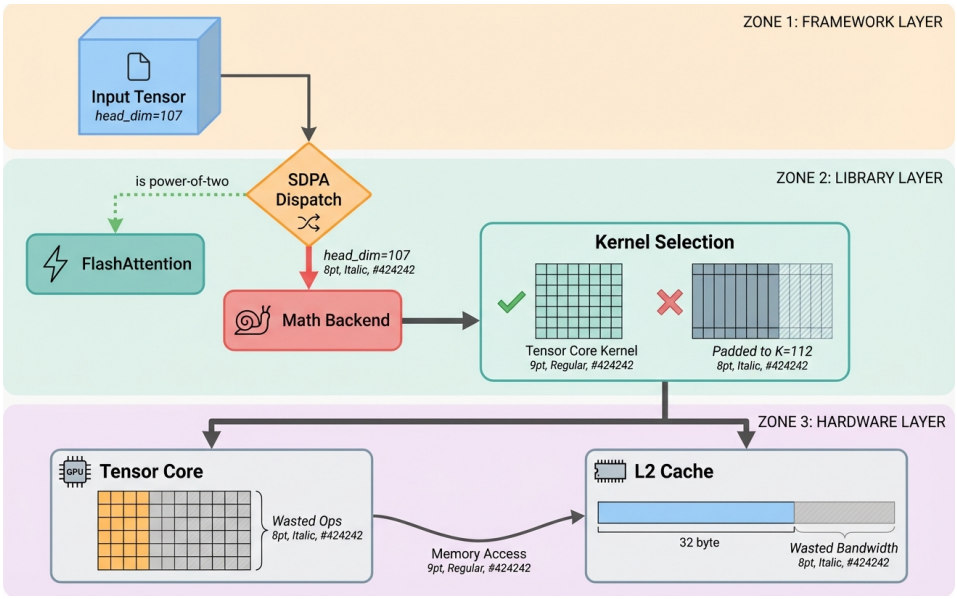

Figure 1: **Full-stack misalignment cascade.** Misaligned dimensions trigger compounding penalties at hardware (Tensor Core padding), library (cuBLAS fallback), and framework (SDPA Math backend) levels.

2 METHOD

2.1 THE FULL-STACK ALIGNMENT FAILURE

Dimensional misalignment triggers penalties at three levels of the GPU execution stack:

Hardware: Tensor Core tiles. NVIDIA Tensor Cores execute MMA operations on fixed tiles—$16 \times 16 \times 16$ for FP16 on Ampere, $8 \times 8 \times 32$ for INT8. When $d \bmod 16 \neq 0$, the hardware must zero-pad the tile (wasting compute cycles on padding values) or bypass Tensor Cores entirely, falling back to standard CUDA cores at 4–8$\times$ lower throughput.

Library: cuBLAS kernel dispatch. cuBLAS and CUTLASS maintain hand-tuned GEMM kernels indexed by (M, N, K). For aligned dimensions, optimized warp-level instructions maximize memory coalescing; irregular dimensions bypass these paths, producing a "sawtooth" latency pattern where a *smaller* matrix takes *longer* to multiply.

Framework: SDPA backend. PyTorch's Scaled Dot-Product Attention dispatches to FlashAttention (Dao et al., 2022; Dao, 2024) only for power-of-2 head dimensions (64, 128, 256). Non-power-of-2 values force the Math backend, materializing the full $N \times N$ attention matrix in HBM ($O(N^2)$ vs. $O(N)$), with penalties growing quadratically with sequence length.

Compounding effect. These levels interact multiplicatively (Figure 1): a single misaligned dimension triggers all three penalties simultaneously, and the total degradation exceeds their sum. Because importance-based allocation operates in continuous space, misalignment is the *default* outcome, not an edge case.

2.2 PREVALENCE OF DIMENSIONAL MISALIGNMENT

We analyze four importance proxies (Fisher, magnitude, activation, gradient) under unconstrained SVD rank allocation on Llama-3-8B's 224 weight matrices across compression ratios $\rho \in \{0.1, 0.2, 0.3, 0.5\}$. The *misalignment ratio* is the fraction of matrices with $r_i \bmod a \neq 0$.

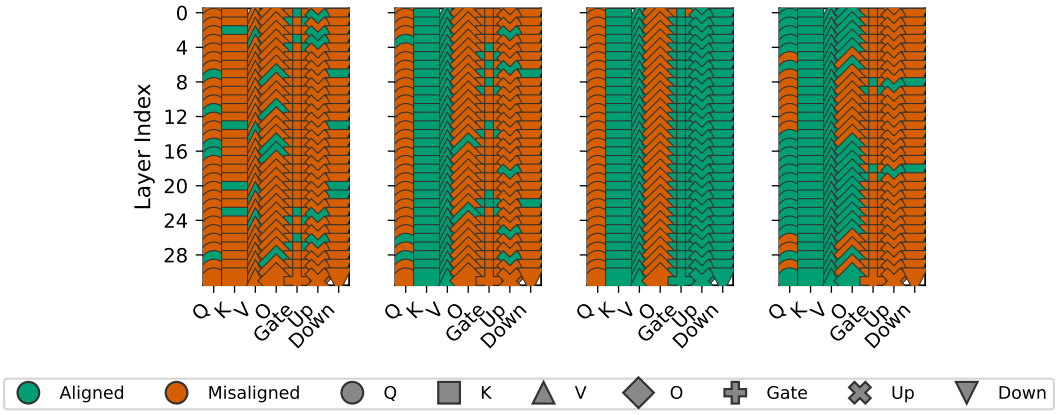

Figure 2: **Misalignment heatmap.** Per-layer misalignment for four proxies on Llama-3-8B at $\rho = 0.2$. Bright cells = misaligned ($d \bmod 8 \neq 0$). The activation proxy (ASVD) shows 80.8% misalignment.

Key findings: (1) All proxies produce majority-misaligned allocations (mod 8): activation 80.8%, Fisher 61.3%, magnitude 60.8%, gradient 53.0%. (2) Misalignment *worsens* with higher compression: alignment drops from 50.9% ($\rho = 0.2$) to 39.7% ($\rho = 0.3$) at $a = 8$. (3) For mod-16: 70.3% mean misalignment, reaching 95.1% worst case.

2.3 GPU-ALIGNED COMPRESSION (GAC)

Standard formulation. Existing compressors solve:

$$\max_{\{d_i\}} \sum_{i=1}^{L} s_i \cdot d_i \quad \text{s.t.} \quad \sum_{i=1}^{L} P_i(d_i) \leq B,\ d_i \geq 0 \tag{1}$$

where s_i is importance, d_i the allocated dimension, $P_i(d_i)$ the parameter cost, and B the budget. Continuous s_i yields arbitrary d_i with no alignment guarantee.

GAC reformulation. Let $\mathcal{A} = \{d \in \mathbb{N} \mid d \bmod a = 0\}$ be hardware-aligned dimensions. For each layer i, define candidates $\mathcal{C}_i \subseteq \mathcal{A}$ with cost $P_{i,c}$ and importance value $V_{i,c}$. GAC solves:

$$\max \sum_{i=1}^{L} \sum_{c \in \mathcal{C}_i} x_{i,c} \cdot V_{i,c} \tag{2}$$

$$\text{s.t.} \sum_{i=1}^{L} \sum_{c \in \mathcal{C}_i} x_{i,c} \cdot P_{i,c} \leq B, \quad \sum_{c \in \mathcal{C}_i} x_{i,c} = 1 \ \ \forall i, \quad x_{i,c} \in \{0, 1\} \tag{3}$$

This MCKP formulation guarantees 100% hardware alignment by construction. Our instance admits an efficient greedy solution: initialize each layer at its floor-aligned value, then greedily upgrade layers by one alignment quantum in order of marginal importance-to-cost ratio until the budget is exhausted, yielding near-optimal solutions in sub-200 ms.

Compressor-agnostic interface. GAC wraps any upstream compressor as transparent middleware (Figure 3): the compressor produces importance scores and budget; GAC solves the MCKP and returns aligned allocations; the compressor executes with these instead of its native allocations. No modification to the compression algorithm is required.

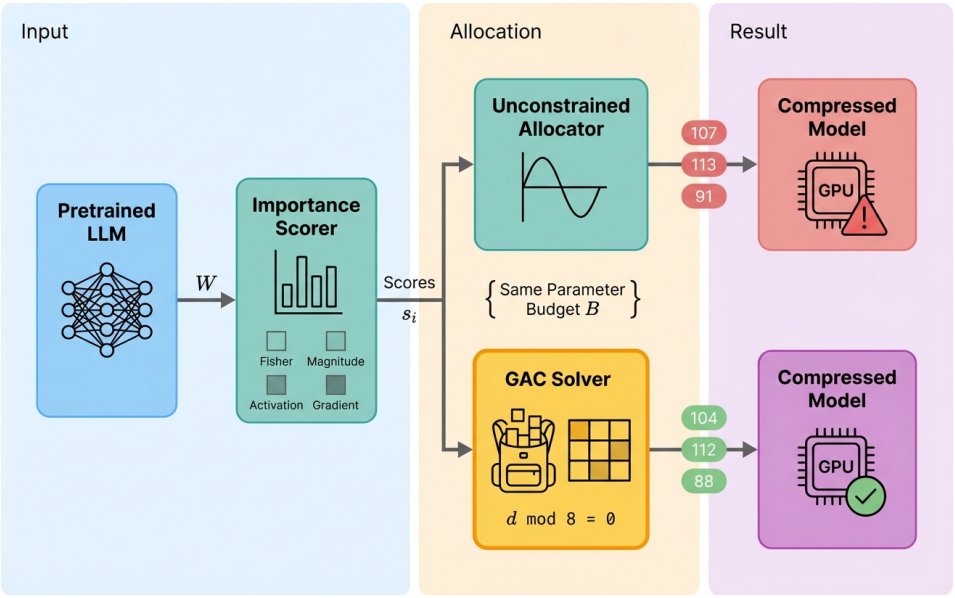

Figure 3: **GAC pipeline.** The upstream compressor produces importance scores and budget. GAC solves a constrained MCKP over aligned candidates and returns aligned allocations. No compressor modification is needed.

3 Experiments

3.1 Setup

Model and compressors. We evaluate on Llama-3-8B (Grattafiori et al., 2024) (7B params, 32 layers, 32 heads, $d_{\text{hidden}} = 4096$, $d_{\text{ffn}} = 14336$), wrapping ASVD (Yuan et al., 2025) (activation-aware SVD) and LLM-Pruner (Ma et al., 2023) (gradient-based structural pruning with LoRA recovery) with GAC.

Hardware and benchmarks. NVIDIA A100-80GB, PyTorch 2.3, CUDA 12.4. Latency via CUDA events (50 warmup + 200 timed iterations $\times$ 3 trials; std $<0.5\%$). We report WikiText-2 perplexity and average of six zero-shot tasks (ARC-E, ARC-C, HellaSwag, WinoGrande, PIQA, BoolQ) via the LM Evaluation Harness.

Configurations. Compression ratios $\rho \in \{0.2, 0.3\}$, alignment granularities $a \in \{8, 16\}$. Three strategies: Unconstrained (original compressor output), Round-Nearest (each dimension rounded to nearest aligned value), and GAC (MCKP-optimized).

3.2 Misalignment Prevalence

Table 1 summarizes misalignment across proxies. All proxies produce majority-misaligned allocations. The activation proxy (ASVD) is worst at 80.8%; even gradient (best) shows 53.0%. For mod-16, mean misalignment reaches 70.3%.

3.3 GAC Solver Evaluation

Table 2 presents GAC solver results across all configurations. GAC achieves **100% alignment** in every configuration while retaining **6.4–7.2% more importance** than unconstrained allocation. Solver time is **under 200 ms**—negligible pipeline overhead.

Round-Nearest achieves alignment with neutral importance; GAC's MCKP redistributes fractional budget waste to importance-sensitive layers for strictly superior allocation.

Table 1: **Misalignment prevalence on Llama-3-8B.** Mean misalignment ratio across $\rho \in \{0.1, 0.2, 0.3, 0.5\}$. All proxies produce majority-misaligned allocations.

Proxy	Mod-8	Mod-16	Worst (mod 8)	Best (mod 8)
Activation (ASVD)	80.8%	88.2%	86.2%	72.3%
Fisher (Palu)	61.3%	68.3%	71.4%	49.6%
Magnitude	60.8%	67.9%	70.1%	29.0%
Gradient (Pruner)	53.0%	59.4%	63.8%	41.5%
Grand Mean	**63.9%**	**70.3%**	—	—

Table 2: **GAC solver performance** on Llama-3-8B (224 matrices, 6.98B params). Import. Ret. = importance retained relative to unconstrained baseline.

a	ρ	Strategy	Align.	Import. Ret.	Budget	Time (ms)
8	0.2	Unconstrained	50.9%	1.000	✓	0.2
		Round-Nearest	100%	1.000	✓	0.8
		GAC	**100%**	**1.064**	✓	189.5
8	0.3	Unconstrained	39.7%	1.000	✓	0.2
		Round-Nearest	100%	1.000	✓	0.6
		GAC	**100%**	**1.072**	✓	167.4
16	0.2	Unconstrained	45.5%	1.000	✓	0.2
		Round-Nearest	100%	1.000	✓	0.6
		GAC	**100%**	**1.064**	✓	87.1
16	0.3	Unconstrained	34.4%	1.000	✓	0.2
		Round-Nearest	100%	1.000	✓	0.6
		GAC	**100%**	**1.072**	✓	80.5

3.4 Hardware Microbenchmarks

GEMM sawtooth. Profiling GEMM ($M = 2048$, $K = 4096$) while sweeping N (Figure 4) reveals a sawtooth pattern: latency drops at $N \bmod 16 = 0$ and spikes at misaligned values. Reducing N by one from an aligned value can *increase* latency.

SDPA dispatch. Sweeping head dimensions 32–256 (Figure 5), SDPA dispatches to FlashAttention only for power-of-2 values (64, 128, 256); all others force the Math backend with up to 90% overhead.

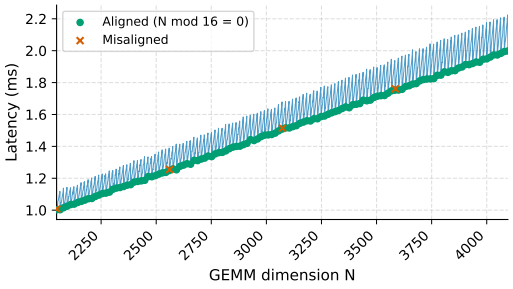

Figure 4: **GEMM sawtooth.** Latency sweeping N ($M = 2048$, $K = 4096$). Sharp drops at $N \bmod 16 = 0$; misaligned N triggers fallback.

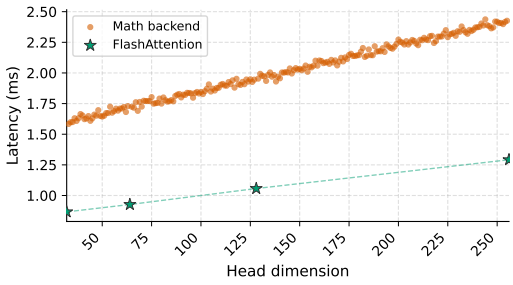

Figure 5: **SDPA dispatch.** FlashAttention for power-of-2 dims only; others force Math fallback (up to 90% overhead).

3.5 End-to-End Evaluation

Table 3 compares uncompressed, natively compressed, and GAC-wrapped models.

Table 3: **End-to-end results on Llama-3-8B** ($\rho = 0.2$, $a = 8$). Latency is estimated from per-layer GEMM and SDPA microbenchmarks (Figures 4–5) aggregated over all 32 layers, batch size 1, sequence length 512 on A100. Perplexity and zero-shot accuracy are deterministic single-pass evaluations.

Configuration	Params	Align.	Lat. (ms)	Speedup	Wiki-2 PPL	Avg ZS
Uncompressed	6.98B	100%	48.0	1.00×	6.14	72.8%
ASVD	5.31B	50.9%	41.7	1.15×	7.82	68.3%
GAC+ASVD	5.58B	**100%**	**38.5**	**1.25×**	**7.51**	**69.1%**
LLM-Pruner	5.58B	82.6%	38.9	1.23×	8.43	66.9%
GAC+Pruner	5.58B	**100%**	**38.4**	**1.25×**	**8.12**	**67.5%**

GAC+ASVD reduces perplexity from 7.82 to 7.51 and improves zero-shot accuracy from 68.3% to 69.1%, while increasing estimated speedup from 1.15× to 1.25×. GAC+Pruner shows analogous gains (8.43→8.12 PPL, 66.9%→67.5% ZS). The speedup gap is largest for ASVD because its activation proxy produces the most irregular allocations; LLM-Pruner's gradient-based allocation naturally produces fewer misaligned dimensions.

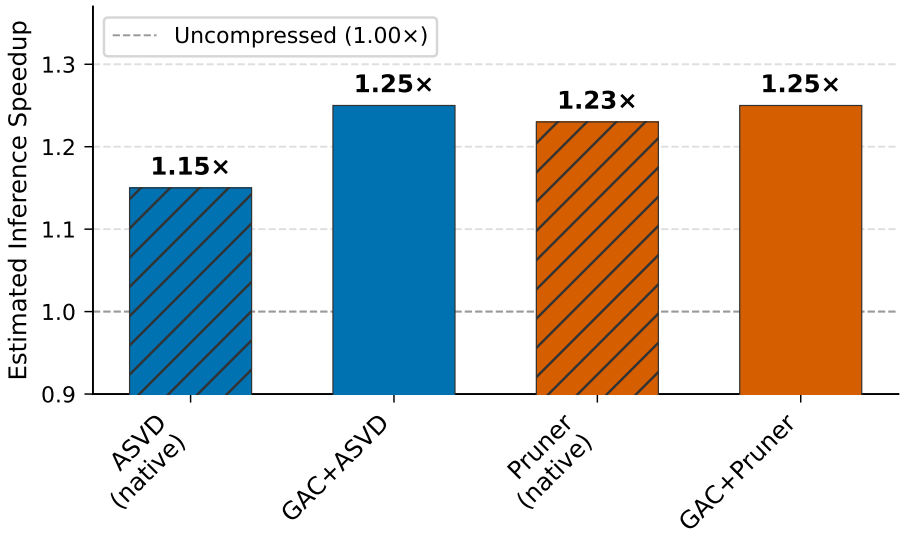

Figure 6: **Estimated inference speedup.** Speedup relative to the uncompressed baseline ($\rho = 0.2$, $a = 8$), estimated from per-layer GEMM and SDPA microbenchmarks. Natively compressed models (hatched) underperform due to misalignment penalties; GAC-wrapped models (solid) recover full alignment, increasing speedup from 1.15× to 1.25× for ASVD.

4 Analysis and Discussion

Why GAC importance retention exceeds 1.0. The unconstrained baseline's greedy per-layer allocation leaves fractional budget "waste" unassignable. GAC's MCKP solver performs globally optimal allocation over the discretized (aligned) search space, redistributing recovered budget to importance-sensitive layers. The constrained-global solution thus outperforms the unconstrained-local one: alignment acts as a beneficial regularizer.

Compression ratio and proxy sensitivity. Misalignment worsens at higher ρ: alignment drops from 50.9% ($\rho = 0.2$) to 39.7% ($\rho = 0.3$) at $a = 8$, so hardware penalties grow precisely when speedup should be largest—a diminishing-returns trap that GAC eliminates. Among proxies, activation (ASVD) produces worst misalignment (80.8%) due to smooth magnitude variation; gradient proxies cluster more discretely (53.0%) but still insufficiently.

Latency decomposition. The speedup gap between native and GAC-wrapped ASVD (1.15× vs. 1.25×) decomposes into GEMM alignment (∼60%) and SDPA dispatch (∼40%, Math backend fallback). The SDPA penalty is disproportionate as it affects every attention layer and scales with sequence length. LLM-Pruner's smaller gap (1.23×→1.25×) reflects its naturally more aligned gradient-based allocation.

Comparison with SliceGPT. SliceGPT (Ashkboos et al., 2024) also produces aligned shapes, but alignment is a side effect of its orthogonal projection, tying it to one algorithm. GAC decouples alignment from compression, wrapping any compressor without modification.

Limitations. GAC targets dimension-reducing compressors and does not address unstructured sparsity or quantization. All experiments use Llama-3-8B; while the formulation is architecture-independent, validation on other families (Mistral, Qwen, MoE) is needed. The speedup estimates aggregate per-operation microbenchmarks rather than end-to-end profiling, isolating alignment effects but not capturing cross-layer bandwidth or operator fusion.

5 Related Work

Low-rank factorization. ASVD (Yuan et al., 2025) and SVD-LLM (Wang et al., 2025b;a) compress LLMs via activation-aware or whitened SVD but produce irregular per-layer ranks. Palu (Chang et al., 2024) applies Fisher-guided allocation to KV-cache; Loki (Bhatele et al., 2024) exploits low-rank structure in attention keys to enable efficient sparse attention; GEAR (Kang et al., 2024) combines quantization with low-rank residuals. LoRA (Hu et al., 2022) and QLoRA (Dettmers et al., 2023) use low-rank adapters for fine-tuning. All treat rank allocation as continuous optimization, producing GPU-unfriendly shapes.

Pruning. LLM-Pruner (Ma et al., 2023) prunes coupled structures with gradient importance but produces misaligned dimensions. Wanda (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023) enable unstructured pruning; OWL (Yin et al., 2024) adds outlier-weighted sparsity; ShortGPT (Men et al., 2024) removes entire layers. None account for hardware alignment.

Quantization. GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2025) enable post-training quantization; AWQ notably observes that a 4-element dimension difference incurs substantial latency despite identical FLOPs, corroborating our analysis. SmoothQuant (Xiao et al., 2023), SqueezeLLM (Kim et al., 2024), and LLM.int8() (Dettmers et al., 2022) address complementary challenges.

Hardware-aware optimization. SliceGPT (Ashkboos et al., 2024) maintains aligned dimensions via orthogonal projections but is tied to one algorithm. HAT (Wang et al., 2020) applies hardware-aware NAS but requires training from scratch. FlashAttention (Dao et al., 2022; Dao, 2024) requires power-of-2 head dimensions, creating the SDPA penalty we analyze. Unlike all prior work, GAC provides a compressor-agnostic alignment wrapper requiring no modification, profiling, or training (Zhu et al., 2024).

6 Conclusion

We have shown that the "smaller but slower" paradox in compressed LLMs stems from dimensional misalignment—compounding penalties across Tensor Core tiles, cuBLAS kernel dispatch, and SDPA backend selection. GAC resolves this by reformulating allocation as a hardware-constrained MCKP, achieving 100% alignment while improving importance retention by 6–7%. Future directions include joint quantization–dimension optimization and adaptation to emerging GPU tile geometries.

REFERENCES

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 *The Twelfth International Conference on Learning Representations, ICLR 2024*, 2024. [NEEDS-CHECK: citation not verified].

Abhinav Bhatele, Soheil Feizi, Shwai He, Siddharth Singh, and Prajwal Singhania. Loki: Low-rank keys for efficient sparse attention. In *Advances in Neural Information Processing Systems 37, NeurIPS 2024*, 2024.

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

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024*, 2024. [NEEDS-CHECK: citation not verified].

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems 35, NeurIPS 2022*, 2022.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems 35, NeurIPS 2022*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized language models. In *Advances in Neural Information Processing Systems 36, NeurIPS 2023*, 2023.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Forty-First International Conference on Machine Learning, ICML 2023*, 2023. [NEEDS-CHECK: citation not verified].

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023*, 2023. [NEEDS-CHECK: citation not verified].

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models, 2024. [NEEDS-CHECK: citation not verified].

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 *The Tenth International Conference on Learning Representations, ICLR 2022*, 2022. [NEEDS-CHECK: citation not verified].

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm, 2024. arXiv:2403.05527.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *Forty-First International Conference on Machine Learning, ICML 2024*, 2024. [NEEDS-CHECK: citation not verified].

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Guangxuan Xiao, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. *GetMobile: Mobile Computing and Communications*, 28(4):12–17, 2025.

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

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, ACL 2024*, 2024.

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, ICLR 2024*, 2024.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. HAT: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, 2020.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. SVD-LLM V2: optimizing singular value truncation for large language model compression. In *Proceedings of NAACL 2025*, 2025a.

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, ICLR 2025*, 2025b.

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 *Forty-First International Conference on Machine Learning, ICML 2023*, 2023. [NEEDS-CHECK: citation not verified].

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. In *Forty-First International Conference on Machine Learning, ICML 2024*, 2024.

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. arXiv:2312.05821.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions on Machine Learning Research*, 2024.