Residual-Aware Sparsification and Pruning: Protecting Principal Subspaces During Sparsification or Pruning

Anonymous ACL submission

Abstract

Linear layers dominate LLM parameter count and memory traffic, making them a major computational bottleneck. Existing structured pruning and activation-sparsity methods reduce these costs by removing or skipping channels according to saliency or activation-based criteria. However, such methods can discard dominant singular directions of the original projection, making performance sensitive to the scoring rule or sparsification criterion. We propose **Residual-Aware Sparsification and Pruning** (RASP), a training-free drop-in wrapper for pruning and sparsity methods. For each targeted weight W, RASP decomposes $W = L_r + R_r$, where L_r captures the top-r singular directions and R_r is the residual. RASP then confines pruning or sparsification to R_r, preserving the dominant projection directions while bounding approximation error by the residual norm, rather than the full weight norm. At 50% sparsity, RASP consistently improves generation scores – by up to +7.33 points over GRIFFIN, +7.93 over WANDA, +2.10 over TEAL. Interestingly, RASP surpasses dense inference on several Gemma 2 9B IT generation benchmarks. Finally, RASP is able to achieve 40%-46% speed-ups compared to a dense model. Our code is publicly available **here**.

1 Introduction

Structured pruning and activation sparsity promise cheaper large language model (LLM) inference by removing or skipping feed-forward network (FFN) channels. However, channel-level sparsification is a coordinate-wise operation, while the transformations learned by MLP projections are organized in global directions. A channel can receive a low magnitude, weight–activation, or prompt-level top-k score while still participating in a dominant singular direction of the projection. Removing such channels can damage the principal subspace of the layer, introducing approximation error that compounds across transformer blocks and decoding steps. Figure 1 illustrates this failure mode: direct masks remove substantial principal-subspace energy, and the resulting performance degradation becomes increasingly severe at high sparsity.

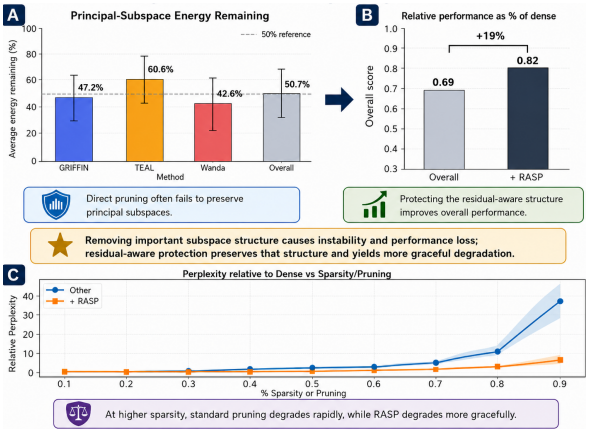


Figure 1: Direct pruning and activation-sparsity masks remove energy from the principal subspace, measured as $\|L_r(I - M)\|_F^2 / \|L_r\|_F^2$. Preserving residual-aware structure with RASP improves relative performance. As sparsity increases, direct pruning yields exponentially increasing relative perplexity, whereas RASP degrades more linearly.

Prior work has addressed LLM inference cost through post-training weight pruning and activation sparsity. SparseGPT showed that massive GPT-family models can be pruned in one shot using layer-wise second-order reconstruction, achieving high unstructured sparsity without retraining (Frantar and Alistarh, 2023). WANDA later showed that a simple weight–activation score can prune pretrained LLMs without retraining or weight updates, avoiding the expensive reconstruction step (Sun et al., 2023). More recent inference-time sparsification methods reduce FFN computation through structured channel selection or activation masking: GRIFFIN uses prompt-conditioned activation statistics to select sequence-level FFN experts during generation (Dong et al., 2024), while TEAL applies

training-free magnitude-based activation sparsity throughout the model (Liu et al., 2024b).

We propose **Residual-Aware Sparsification and Pruning** (RASP), a training-free wrapper for pruning and activation-sparsity methods. Rather than introducing a new saliency score or threshold, RASP changes the object being pruned. For each targeted projection W, RASP computes an SVD decomposition $W = L_r + R_r$, where L_r contains the top-r singular directions and R_r is the residual. Given any diagonal keep-mask M selected by an existing pruning or activation-sparsity rule, RASP replaces the usual direct approximation WM with $\widetilde{W}_{r,M} = L_r + R_r M$. Thus, the principal component is always evaluated densely, while sparsification is confined to the residual branch. This makes RASP complementary to GRIFFIN, WANDA, TEAL, and other channel-selection rules: the same mask can be used, and its destructive effect is restricted to the lower-energy residual correction.

This simple change yields a direct approximation guarantee. For the linear surrogate, $W - \widetilde{W}_{r,M} = R_r(I - M)$, and therefore

$$\|W - \widetilde{W}_{r,M}\|_2 \leq \|R_r\|_2 = \sigma_{r+1}(W).$$

The rank r is an explicit error-control knob: increasing r shrinks the component exposed to pruning. Empirically, this structural protection consistently improves training-free sparsification. Across Llama and Gemma models at various sparsities and ranks, RASP *generally improves* generation, reasoning, and classification over other methods. Interestingly, at 50% sparsity, we even see RASP *surpass the fully dense models* on generation tasks. We also implement a sparse residual-only GEMV (RGEMV) kernel, combining a dense low-rank path with sparse residual computation during autoregressive decoding. In summary, this paper makes the following contributions:

- We identify a structural failure mode of direct sparsification: channel-saliency masks can remove energy from dominant singular subspaces.
- We propose RASP, a training-free wrapper that preserves a dense low-rank principal component, sparsifies only to the residual, and supports sparse decoding with our RGEMV kernel.
- We prove that the linear RASP error is bounded by the residual singular tail, $\sigma_{r+1}(W)$, yielding a rank-controlled sparsity trade-off.
- We show that RASP improves GRIFFIN, WANDA, and TEAL across generation, classification, and reasoning benchmarks.

2 Background and Preliminaries

We use a column-vector convention. For an input activation $x \in \mathbb{R}^{d_{\text{in}}}$ and projection $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the dense linear output is $y = Wx$. During autoregressive decoding, applying such projections across all transformer MLP blocks incurs substantial memory traffic. Structured pruning and activation sparsity reduce this cost by constructing a binary channel keep-mask $M = \text{diag}(m)$, with $m_j \in \{0, 1\}$, so that masked channels can be skipped.

For a column mask, direct sparsification uses

$$W_{\text{direct}} = WM, \qquad W - W_{\text{direct}} = W(I - M).$$

This formulation is efficient, but it treats coordinate directions as the unit of pruning. Let

$$W = L_r + R_r$$

be a rank-r truncated-SVD decomposition, where L_r is the principal low-rank component and R_r is the residual. Then direct masking gives

$$W(I - M) = L_r(I - M) + R_r(I - M).$$

Thus, a direct mask can remove energy from the principal component L_r, not only from the residual correction. RASP is designed to avoid this failure by preserving L_r and applying masks only to R_r.

3 Methodology

We now describe **Residual-Aware Sparsification and Pruning** (RASP). As illustrated in Figure 2, RASP is a wrapper around an existing pruning or activation-sparsity rule. RASP changes where that mask is applied. Instead of masking the full projection, RASP preserves a dense principal component and applies the mask only to the residual branch.

3.1 Problem Statement

Problem. Consider a linear projection $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and an input activation $x \in \mathbb{R}^{d_{\text{in}}}$. A channel-selection method produces a diagonal keep-mask $M = \text{diag}(m)$, where $m_j \in \{0, 1\}$. Direct sparsification applies this mask to the original projection, $y_{\text{direct}} = WMx$, which gives approximation error

$$Wx - y_{\text{direct}} = W(I - M)x.$$

As discussed in Section 2, this error can include components from the principal subspace of W. Our

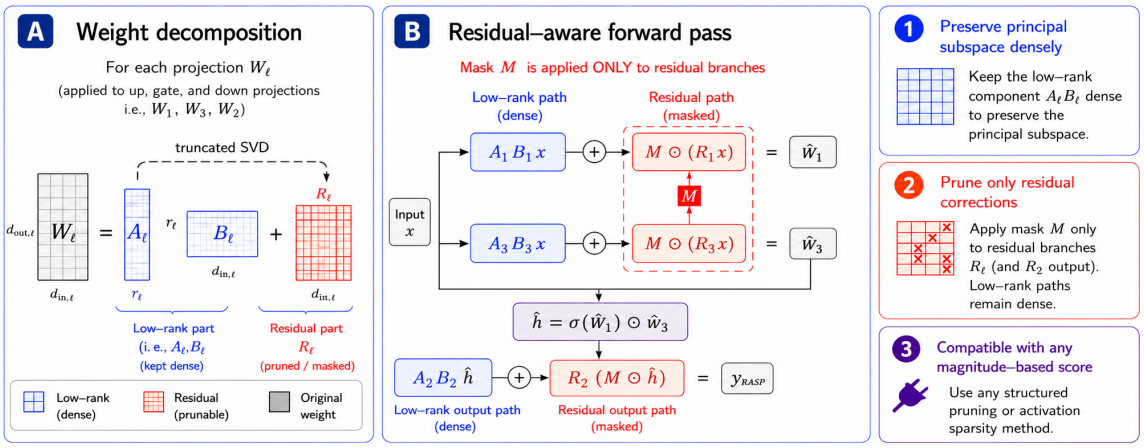


Figure 2: Overview of the RASP structure. Each projection matrix is decomposed into a dense low-rank principal component and a residual component. Existing pruning or activation-sparsity masks are then applied only to the residual component, preserving the principal subspace densely.

goal is to use the same mask M while preventing it from directly removing the dominant singular directions.

3.2 Residual-Aware Masking

For each targeted projection, RASP computes a rank-r truncated-SVD decomposition

$$W = L_r + R_r, \quad L_r = U_r \Sigma_r V_r^\top, \quad R_r = W - L_r.$$

The low-rank component L_r contains the top-r singular directions and is kept dense. The residual component R_r is exposed to sparsification. Given the same keep-mask M that would be used by the underlying pruning rule, RASP computes

$$y_{\text{RASP}} = (L_r + R_r M)x.$$

Equivalently, the residual-aware surrogate weight is $\widetilde{W}_{r,M} = L_r + R_r M$. Thus, $M = I$ recovers the dense projection, while $M = 0$ reduces the layer to the protected rank-r backbone L_r. For intermediate sparsity levels, the principal component remains dense and only the residual correction is pruned.

When useful for inference, the dense principal component can be parameterized as

$$L_r = A_r B_r, \quad A_r = U_r \Sigma_r^{1/2}, \quad B_r = \Sigma_r^{1/2} V_r^\top.$$

3.3 Properties of the RASP Operator

We now state the key properties of the residual-aware operator. These results explain why applying an existing mask to R_r, rather than directly to W, changes the error structure and gives a budget-aware justification for residual-aware pruning.

Lemma 1 (Residual-only approximation error)**.** *Let $W = L_r + R_r$, where L_r is the rank-r SVD approximation of W and $R_r = W - L_r$. For any diagonal keep-mask M, define*

$$\widetilde{W}_{r,M} = L_r + R_r M.$$

Then the pruning error is confined to the residual branch:

$$W - \widetilde{W}_{r,M} = R_r(I - M).$$

Consequently,

$$\|W - \widetilde{W}_{r,M}\|_2 \le \sigma_{r+1}(W).$$

Proof. See Appendix A.1 □

> Increasing r directly shrinks the part of the projection that can be damaged by pruning.

Lemma 2 (Rank–sparsity interpolation)**.** *Let M_ρ denote a mask with residual sparsity ρ, and define*

$$\widetilde{W}_{r,\rho} = L_r + R_r M_\rho.$$

Then RASP *interpolates between direct pruning, dense inference, and low-rank approximation:*

$$r = 0 \Rightarrow \widetilde{W}_{0,\rho} = W M_\rho, \qquad \rho = 0 \Rightarrow \widetilde{W}_{r,0} = W,$$
$$\rho = 1 \Rightarrow \widetilde{W}_{r,1} = L_r.$$

Proof. See Appendix A.6. □

> RASP acts as a continuum between the underlying pruning method and a protected low-rank approximation.

Theorem 1 (When RASP beats equal-budget direct pruning)**.** *Let $W \in \mathbb{R}^{m\times n}$, and let $W = L_r + R_r$, where $L_r = A_r B_r$ is the rank-r SVD approximation of W. Let $P(R_r, q)$ be a residual column-pruning operator that keeps q columns of R_r, and define the* RASP *approximation $\widetilde{W}_{r,q} = A_r B_r + P(R_r, q)$. Ignoring support-index metadata, $\widetilde{W}_{r,q}$ has parameter count $r(m+n) + mq$. Let $P(W, k)$ be a direct column-pruning operator on W with the same parameter budget, so that*

$$mk = r(m+n) + mq, \quad k = q + \frac{r(m+n)}{m}.$$

Let $c_{(1)} \geq \cdots \geq c_{(n)}$ denote the sorted column norms of W. If

$$\|R_r - P(R_r, q)\|_F^2 < \sum_{j>K} c_{(j)}^2,$$

then

$$\|W - \widetilde{W}_{r,q}\|_F^2 < \|W - P(W, k)\|_F^2$$

for every direct top-k column-pruning rule $P(W, k)$.

Proof. See Appendix A.5. □

> When the residual pruning error is smaller than the unavoidable dropped energy of direct pruning, residual-aware pruning is strictly better at the same parameter count.

This is the budget-matched justification for RASP. This theorem also allow us to calculate *equivalent sparsity*, i.e. for a given RASP q, r, what is the equivalent k?

3.4 RASP for Gated MLPs

Modern LLMs typically use gated MLPs such as SwiGLU. Let $x \in \mathbb{R}^{d_{\text{model}}}$, and let

$$W_1, W_3 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}, \qquad W_2 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$$

denote the up-, gate-, and down-projection matrices. The dense block computes

$$h = \sigma(W_1 x) \odot (W_3 x), \qquad y = W_2 h.$$

RASP decomposes each projection,

$$W_i = L_{i,r} + R_{i,r}, \qquad i \in \{1, 2, 3\}.$$

An existing sparsification rule then produces FFN-channel scores s. For a target sparsity ρ, we keep $k = (1-\rho)d_{\text{ff}}$ channels and construct $M = \text{diag}(m)$, where

$$m_j = \begin{cases} 1, & \text{if } s_j \in \text{TopK}(s, k), \\ 0, & \text{otherwise.} \end{cases}$$

Since M acts on the FFN channel dimension, it left-multiplies the residual branches of the up and gate projections and right-multiplies the residual branch of the down projection:

$$\widehat{h} = \sigma\big((L_{1,r} + MR_{1,r})x\big) \odot \big((L_{3,r} + MR_{3,r})x\big),$$

$$y_{\text{RASP}} = L_{2,r}h + R_{2,r}M\widehat{h}.$$

The low-rank paths remain dense, while the selected mask affects only the residual corrections. This is the forward-pass structure shown in Figure 2.

3.5 RASP Theoretical GFLOPs Reduction

We estimate the theoretical decoding-time GFLOPs reduction obtained by wrapping GRIFFIN and TEAL with RASP. We count the dense matrix-vector cost as $2mn$ FLOPs. For residual sparsity ρ and rank-r, the RASP-wrapped projection cost is therefore approximated as

$$\begin{aligned} \text{FLOPs}_{\text{RASP}}(W) = 2(1-\rho)mn \\ + 2r(m+n). \end{aligned} \tag{1}$$

3.6 Hardware-efficient Customized Kernels

We implement a Triton-based (Tillet et al., 2019) sparse Residual-GEMV (RGEMV) kernel for the equivalent RASP forward form $y_{\text{RASP}} = WMx + A_r B_r (I - M)x$. The first term, WMx, reuses the TEAL-style sparse activation path: the kernel thresholds the input activation, identifies the retained coordinates, selectively loads the corresponding column-major weight columns, and accumulates the sparse matrix-vector product during autoregressive decoding. The second term is implemented as a lightweight low-rank correction: the dropped coordinates $(I - M)x$ are first compressed by B_r into a rank-r vector, which is then expanded by A_r and fused into the sparse GEMV epilogue. This implementation avoids materializing the dense residual matrix $R_r = W - A_r B_r$, preserves the optimized sparse weight-loading behavior of the TEAL kernel, and adds only a small low-rank correction for channels removed by the activation mask.

4 Related Work

One-shot and structured pruning. SparseGPT showed that large generative models can be pruned in a one-shot setting using approximate second-order reconstruction (Frantar and Alistarh, 2023). WANDA showed that a simpler weight–activation saliency score can outperform magnitude pruning without expensive second-order updates (Sun et al., 2023). Structured pruning methods such as LLM-Pruner, FLAP, and SliceGPT remove larger model structures for more hardware-friendly compression (Ma et al., 2023; An et al., 2024; Ashkboos et al., 2024). These methods differ in saliency criteria and pruning granularity, but apply masks directly to the original weight representation. RASP instead changes where a selected mask is applied by preserving a low-rank principal component and pruning only the residual component.

Activation sparsity and dynamic channel selection. Activation-sparsity and contextual-sparsity methods exploit input-dependent structure during inference. DejaVu(Liu et al., 2023) predicts contextual sparsity in attention heads and MLP parameters, while PowerInfer (Song et al., 2024) exploits neuron-activation locality for sparse LLM serving . GRIFFIN uses prompt-phase feed-forward statistics to identify important channels and reuse them during generation (Dong et al., 2024). TEAL applies magnitude-threshold activation sparsity across LLM layers, calibrating layerwise sparsity levels and using sparse kernels to obtain decoding speedups (Liu et al., 2024b). CATS (Lee et al., 2024) further studies context-aware activation thresholding with sparse kernels . WINA (Chen et al., 2026) shows that activation magnitude alone can miss channel importance, since the output perturbation from zeroing a channel scales with the product of its activation magnitude and its downstream weight-column L_2 norm. In contrast, RASP reduces the damage from imperfect channel selections by protecting the dominant low-rank component and applying the mask only to the residual component.

Low-rank decomposition and residual adaptation. PiSSA (Meng et al., 2024) uses principal singular values and vectors to initialize low-rank adapters for parameter-efficient fine-tuning. SVD-based compression methods such as ASVD (Yuan et al., 2023) and SVD-LLM (Wang et al., 2025) approximate LLM weights with low-rank factors to reduce model size or computation. LoSparse (Li et al., 2023) combines low-rank and sparse components for weight compression. RASP addresses a different problem: it is a *training-free inference-time* wrapper that uses the low-rank component as a protected, unmasked path and restricts structured pruning to the residual component.

5 Experimental Results

In this section, we empirically evaluate the effectiveness of RASP across various large language models and benchmark datasets. To ensure a comprehensive assessment, we first test our approach on generation-style benchmarks, followed by an evaluation on classification objectives. Finally, we investigate its performance on complex reasoning tasks to demonstrate the versatility of our method.

5.1 Experimental setup

Backbone Models. We evaluate RASP on several decoder-only language models, including Llama-family (Grattafiori et al., 2024)(across 3B, 8B and 70B), Gemma-family (Team et al., 2024) (2B and 9B), and DeepSeek-distilled models (Liu et al., 2024a). For the Llama and Gemma models, we used instruction-tuned variants for our experiments.

Tasks. We evaluate on generation, reasoning, classification, code, and reading comprehension benchmarks. Representative tasks include GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2024), CoQA (Reddy et al., 2019), HumanEval (Chen et al., 2021), Qasper (Dasigi et al., 2021), ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021),and MMLU (Hendrycks et al., 2020).

Method	Weight/channel pruning	Activation sparsity	Structured mask	Calibration data
WANDA	✓	✗	✗	✓
GRIFFIN	✓	✗	✓	✗
TEAL	✗	✓	✓	✓

Table 1: Taxonomy of representative post-training sparsification methods used as base methods in our experiments.

Baselines. We compare dense inference against the underlying pruning or sparsification methods and their RASP-wrapped variants that are summarized in Table 1. GRIFFIN (Dong et al., 2024) treats standard MLP layers as a pseudo-Mixture of Experts (MoE). It tracks activation patterns during

the prompt phase to see which intermediate neurons continuously fire. It then locks down that specific subset of channels to act as the fixed "experts" for the subsequent generation tokens. When combined with GRIFFIN, RASP preserves the dominant FFN subspace while using prompt-level statistics to select residual experts. When combined with WANDA (Sun et al., 2023), it restricts weight-activation pruning to the residual correction. When combined with TEAL (Liu et al., 2024b), it turns magnitude-based activation sparsity into residual-only activation sparsity.

Implementations We use LM Evaluation Harness for all experiments (Gao et al., 2024). We report task accuracy or task-specific score, average benchmark score, *equivalent sparsity* (defined in section 3.3), perplexity where applicable, and wall-clock decoding latency. Full implementation and experimental details are provided in the Section B.1 of the Appendix.

5.2 Main results

Method / Model	Llama 3 (Instruct)			Gemma 2 (IT)	
	3.2 3B	3 8B	3 70B	2B	9B
Dense	43.95	44.90	53.84	34.69	44.86
GRIFFIN	23.70	22.89	38.85	15.94	31.60
+ RASP	**31.03**	**27.15**	**44.79**	**20.02**	**38.69**
WANDA	23.95	23.92	38.81	15.55	32.80
+ RASP	**31.88**	**29.65**	**45.23**	**20.04**	**38.76**
TEAL	42.01	39.90	52.06	31.79	44.41
+ RASP	**42.54**	**42.00**	**52.44**	**32.50**	**44.47**

Table 2: Results on six generation-style benchmarks: GSM8K, MATH500, CoQA, HumanEval, Qasper, and XSum. Reported values are averages over the six tasks; full per-task results are in Appendix B. Dense rows are not sparse. All other rows use equivalent sparsity $= 0.5$, and all RASP variants use rank $r = 256$.

Generation Tasks Table 2 presents the empirical results of our proposed RASP framework evaluated across six downstream generation tasks at 50% sparsity. We benchmark RASP against three prominent baseline methods—GRIFFIN, WANDA, and TEAL—using various scales of the Llama and Gemma 2 model families or highly aggressive pruning baselines like GRIFFIN and WANDA, which suffer severe performance degradation at 50% sparsity, RASP serves as a powerful recovery mechanism. For instance, on Llama 3.2 3B Instruct, RASP boosts GRIFFIN's performance from 23.70 to 31.03 and WANDA's performance from 23.95 to 31.88. Similar prominent gains of roughly 5% to 7% are observed on the Llama 3 8B Instruct and Gemma 2 2B IT variants. Overall, integrating RASP consistently yields substantial performance gains across all base sparsification methods and model architectures, effectively bridging the gap between sparse models and their dense counterparts. We observe that, on Gemma-2 9B IT, TEAL+RASP exceeds dense inference on four generation-style benchmarks: MATH500, CoQA, Qasper, and XSum, which may reflect residual-aware sparsification suppressing weak or task-irrelevant residual features while preserving the principal subspace.

Classification Tasks We next evaluate the effectiveness of RASP on downstream classification objectives, reporting both the average accuracy across six 0-shot tasks and performance on the 5-shot MMLU benchmark. The results across the Llama model family are summarized in Table 3. RASP generally improves classification performance, with the most consistent gains on MMLU. In Table 3, RASP improves 5-shot MMLU for every base method and model size, while the six-task 0-shot average improves in most settings. This suggests RASP is especially useful for recovering performance on knowledge-heavy classification benchmarks under 50% sparsity.

Model	Method	Avg Acc (0-shot)	MMLU (5-shot)
Llama 3.2 3B Instruct	Dense	65.95	60.76
	GRIFFIN	62.64	45.07
	+ RASP	**63.61**	**52.30**
	WANDA	62.88	46.21
	+ RASP	**64.04**	**53.17**
	TEAL	63.74	55.31
	+ RASP	**64.18**	**56.94**
Llama 3 8B Instruct	Dense	71.24	66.77
	GRIFFIN	66.89	51.20
	+ RASP	**67.89**	**57.58**
	WANDA	67.20	52.15
	+ RASP	**67.90**	**58.65**
	TEAL	**69.22**	61.41
	+ RASP	68.92	**62.85**
Llama 3 70B Instruct	Dense	77.48	80.22
	GRIFFIN	75.34	72.38
	+ RASP	**76.35**	**76.65**
	WANDA	75.63	72.60
	+ RASP	**76.04**	**77.05**
	TEAL	75.97	77.99
	+ RASP	**76.50**	**78.69**

Table 3: Classification results at sparsity $= 0.5$. Avg. is the mean accuracy over six zero-shot tasks: HellaSwag (HS), PIQA, BoolQ, ARC-Easy (ARC-e), ARC-Challenge (ARC-c), and WinoGrande (WinoG). MMLU is evaluated 5-shot and reported separately. Dense rows are not sparse, and all RASP variants use rank $r = 256$. Higher is better for all metrics.

Across both evaluation metrics, the empirical

data highlight three core strengths of the RASP framework. First, it acts as a powerful recovery mechanism for aggressive pruners like GRIFFIN and WANDA, dramatically reversing their performance collapse on MMLU by up to $+7.23\%$ absolute accuracy. Second, RASP exhibits excellent synergy with stronger baselines; when paired with TEAL on Llama 3 8B Instruct, it pushes the MMLU score to 62.85%. Finally, these gains across varying model scales and baseline pruners demonstrate that RASP is fundamentally robust and orthogonal to the underlying pruning mechanism, acting as a beneficial low-rank regularizer rather than just a patch for lost information.

Reasoning Tasks We further evaluate RASP on DeepSeek-R1-distilled Qwen 1.5B and 7B models, using TEAL as the base sparsification method. As shown in Table 4, RASP improves the average reasoning score over TEAL on both model sizes. On DeepSeek-R1-Distill-Qwen 1.5B, RASP raises the average score from 35.09 to 36.87, with gains on GSM8K and MATH500, although AIME24 decreases from 10.00 to 6.67. On the 7B model, the improvementis more consistent. RASP increases the average score from 57.20 to 60.33 and improves all three reasoning benchmarks, including a substantial gain on AIME24 from 36.67 to 43.33. This suggests that residual-aware sparsification improves reasoning performance on average, while the AIME24 result indicates that smaller models may be more sensitive to residual masking on harder reasoning benchmarks.

Model	Method	GSM8K	MATH	AIME	Avg.
DS-R1 Distill Qwen 1.5B	Dense	54.59	55.40	26.67	45.55
	TEAL	47.08	48.20	**10.00**	35.09
	+ RASP	**50.95**	**53.00**	6.67	**36.87**
DS-R1 Distill Qwen 7B	Dense	76.04	64.60	56.67	65.77
	TEAL	74.53	60.40	36.67	57.20
	+ RASP	**76.27**	**61.40**	**43.33**	**60.33**

Table 4: Reasoning results on DeepSeek-R1-Distill-Qwen 1.5B and 7B. Dense rows are not sparse; TEAL and TEAL+RASP rows use sparsity $= 0.5$. MATH denotes MATH500, and AIME denotes AIME24. All RASP variants use rank $r = 256$.

Inference Speed Up As described in section 3.5, we calculate the GFLOPs reduction compared to a fully-dense model. For GRIFFIN+RASP, we apply this accounting only to the MLP projections, since GRIFFIN selects FFN experts during generation. For TEAL+RASP, we apply it to all transformer-block linear projections, since TEAL targets activation sparsity across the model's linear layers. As shown in figure 3, RASP reduces the overall GFLOPS by up to 60.4% on Llama-3 8B, 59.6% on Gemma-2 9B and 60.1% on DeepSeek-R1 7B.

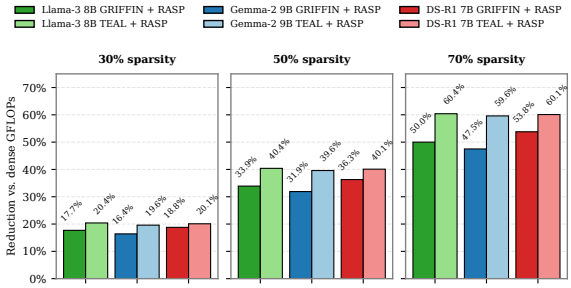


Figure 3: GFLOPs reduction from RASP-wrapped at 30%, 50%, and 70% sparsity. Each panel reports the reduction relative to dense transformer-block linear GFLOPs.

As described in section 3.6, we also created a RGEMV Triton kernel. We have plotted the time to decode in Figure 4. We benchmark a Llama-3 8B MLP projection with $X \in \mathbb{R}^{B\times T\times 4096}$, $W \in \mathbb{R}^{4096\times 14336}$, and sequence of length $T = 128$. TEAL+RASP closely tracks TEAL across sparsity levels while preserving the residual-aware low-rank path. Compared to dense and at a batch size $B = 1$, our RGEMV kernel achieves a 46% speedup at 65% equivalent sparsity and 40% speed up at at 50% equivalent sparsity.

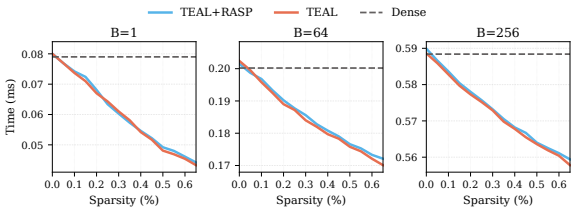


Figure 4: H100 latency for TEAL+RASP and TEAL across equivalent sparsity levels and batch sizes $B \in \{1, 64, 256\}$ on a Llama 3 8B MLP projection.

6 Ablations and Analysis

In this section, we conduct a series of ablation experiments to dissect the contributions of our proposed components and analyze the sensitivity of RASP to different hyperparameter configurations and selection strategies.

6.1 Does rank control error?

In Section B.5, we test whether increasing r reduces observed dense-vs-RASP block error in the same order predicted by the singular-value tail.

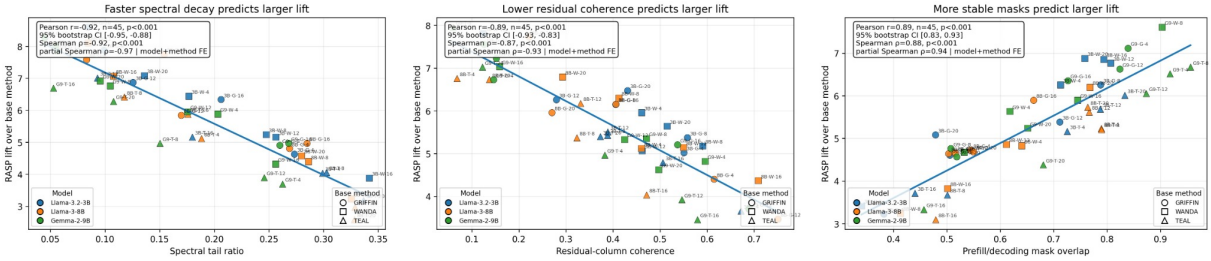


Figure 5: Structural diagnostics of RASP lift. Each point is a model–method–layer measurement at fixed sparsity and task aggregation. RASP lift is larger when spectral tail mass is smaller, residual-column coherence is lower, and prefill/decoding masks are more stable. All three trends have the predicted sign and remain significant under Spearman correlation and model/base-method fixed-effect controls.

What structural conditions favor RASP? We next test whether the empirical gains from RASP follow the structural conditions suggested by the theory (see experimental setup in Section B.2). Figure 5 reports model–method–layer measurements at fixed sparsity and task aggregation. RASP lift is largest when the singular spectrum has a smaller tail, when residual columns are less coherent, and when masks selected during prefill remain stable during decoding. The observed correlations are strong and have the predicted signs: spectral tail ratio is negatively correlated with RASP lift ($r = -0.92$), residual-column coherence is also negatively correlated with lift ($r = -0.89$), and prefill/decoding mask overlap is positively correlated with lift ($r = 0.89$), with all relationships significant at $p < 0.001$. These trends persist under rank-based correlation and after controlling for model and base-method fixed effects, indicating that RASP gains are not merely artifacts of a particular backbone or sparsification rule.

> RASP is most beneficial when the protected low-rank component captures the dominant projection structure and the residual branch is both diffuse and mask-stable.

6.2 When does prefill scoring fail?

We study the effect of prefill sparsification on generation tasks for TEAL and RASP + TEAL with Llama 3.2 3B Instruct and present the results in Figure 6. Our comparison includes the performance relative to the dense model when we do not sparsify the prefill (0% prefill sequence) and when we apply sparsity = 0.5 to half of the prefill sequence (50%) and the full prefill sequence (100%). The results show that RASP is less sensitive to prefill sparsification than TEAL. As we sparsify more prefill tokens, TEAL has a sharper degradation than RASP, dropping 13 points (relative to dense) when we prune the full prefill sequence. In contrast, RASP drops by just 8% relative to dense and improves TEAL by 7%.

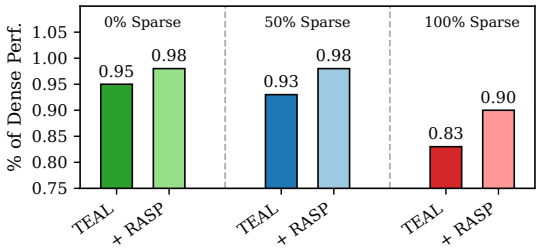


Figure 6: Relative performance on generation tasks at various % prefill sparsification for TEAL and TEAL + RASP.

> RASP is less sensitive to prefill sparsification because unstable masks can only remove residual corrections.

7 Conclusion

We proposed RASP, a residual-aware structured pruning framework that preserves the principal low-rank component of MLP weights and applies pruning only to the residual branch. RASP reframes pruning as residual correction selection rather than direct weight removal. This distinction matters because it separates two roles of an MLP projection: a dense principal subspace that is preserved, and a residual subspace that can be sparsified. The theory predicts that RASP should be most effective when the singular spectrum decays rapidly, when residual columns have low coherence, and when prefill-selected masks remain stable during decoding. Across pruning rules and model families, RASP provides a simple drop-in mechanism for improving accuracy–sparsity trade-offs while remaining compatible with sparse inference kernels.

Limitations

Our theoretical guarantees are derived for the linear RASP surrogate and local residual-output perturbations. Full gated MLPs introduce nonlinear interactions through SiLU and multiplicative gating, which are not fully captured by this analysis. Practical speedups also depend on efficient kernels and consistent realized sparsity. Therefore, nominal sparsity alone does not guarantee wall-clock acceleration.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(10):10865–10873.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Sihan Chen, Dan Zhao, Jongwoo Ko, Colby Banbury, Huiping Zhuang, Luming Liang, Pashmina Cameron, and Tianyi Chen. 2026. WINA: Weight informed neuron activation for accelerating large language model inference. In *The Fourteenth International Conference on Learning Representations*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610.

Harry Dong, Beidi Chen, and Yuejie Chi. 2024. Prompt-prompted adaptive structured pruning for efficient llm generation. *arXiv preprint arXiv:2404.01365*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Donghyun Lee, Jaeyong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. 2024. CATS: Context-aware thresholding for sparsity in large language models. In *First Conference on Language Modeling*.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 20336–20350. PMLR.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Kai Liu, Bowen Xu, Shaoyu Wu, Xin Chen, Hao Zhou, Yongliang Tao, and lulu hu. 2025. La roSA: Enhancing LLM efficiency via layerwise rotated sparse activation. In *Forty-second International Conference on Machine Learning*.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. 2023. Deja vu: Contextual sparsity for efficient

LLMs at inference time. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22137–22176. PMLR.

Zichang Liu and 1 others. 2024b. Training-free activation sparsity in large language models. *arXiv preprint arXiv:2408.14690*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Fanxu Meng and 1 others. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2024. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 590–606. Association for Computing Machinery.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, page 10–19, New York, NY, USA. Association for Computing Machinery.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. 2025. SVD-LLM v2: Optimizing singular value truncation for large language model compression. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4287–4296, Albuquerque, New Mexico. Association for Computational Linguistics.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800.

Appendix Overview

This appendix provides additional theoretical and experimental details that supplement the main text. Appendix A gives the full theoretical analysis of RASP, including the residual-only approximation and expected-error bounds, budget-matched comparisons with direct column pruning, operator-norm and column-spread conditions, rank–sparsity interpolation, and counterexamples that clarify the limits of the theory. Appendix B reports expanded experimental results, including full benchmark tables, prefill-sparsification experiments, LaRoSA experiments, and additional rank and sparsity analyses.

A Theoretical Analysis

We analyze RASP through the lens of low-rank approximation and structured column pruning. The goal is to formalize when preserving a dense low-rank component and pruning only residual structure is preferable to directly pruning the full weight matrix. Throughout this section, let

$$W \in \mathbb{R}^{m\times n}$$

denote a dense projection matrix. Let its singular value decomposition be

$$W = U\Sigma V^\top,$$

with singular values

$$\sigma_1(W) \geq \sigma_2(W) \geq \cdots \geq 0.$$

For a target rank r, define the truncated-SVD approximation

$$L_r = U_r\Sigma_r V_r^\top,$$

and the residual

$$R_r = W - L_r.$$

We let:

$$A_r = U_r\Sigma_r^{1/2}, \qquad B_r = \Sigma_r^{1/2}V_r^\top,$$

so that

$$A_rB_r = U_r\Sigma_r V_r^\top = L_r.$$

Here

$$A_r \in \mathbb{R}^{m\times r}, \qquad B_r \in \mathbb{R}^{r\times n}.$$

Thus the low-rank representation has a nominal parameter count

$$\mathrm{params}(A_r) + \mathrm{params}(B_r) = r(m+n).$$

Let $M = \mathrm{diag}(m_1, \ldots, m_n)$ be a diagonal keep-mask with $m_j \in \{0,1\}$. For a kept-column set $S \subseteq [n]$, we also write M_S for the corresponding diagonal mask. A top-k column pruning operator keeps k columns of W and zeroes the rest:

$$P(W,k) = WM_S, \qquad |S| = k.$$

The parameter count of this pruned representation, ignoring support-index metadata, is

$$\mathrm{params}(P(W,k)) = mk.$$

A.1 Residual-only approximation bound

RASP decomposes the weight matrix into a dense low-rank component and a residual component, then applies pruning only to the residual. The corresponding linear surrogate is

$$\widetilde{W}_{r,M} = L_r + R_rM.$$

This differs from direct pruning WM because the low-rank component L_r is never masked.

Theorem 2 (Residual-only approximation error)**.** *Let $W = L_r + R_r$, where L_r is the rank-r truncated-SVD approximation of W and $R_r = W - L_r$. Let M be any diagonal keep-mask. Define*

$$\widetilde{W}_{r,M} = L_r + R_rM.$$

Then

$$W - \widetilde{W}_{r,M} = R_r(I-M).$$

Consequently,

$$\begin{aligned}\|W - \widetilde{W}_{r,M}\|_2 &= \|R_r(I-M)\|_2 \\ &\leq \|R_r\|_2 = \sigma_{r+1}(W).\end{aligned}$$

and

$$\begin{aligned}\|W - \widetilde{W}_{r,M}\|_F &= \|R_r(I-M)\|_F \leq \|R_r\|_F \\ &= \left(\sum_{i>r}\sigma_i(W)^2\right)^{1/2}.\end{aligned}$$

Moreover, for any input $x \in \mathbb{R}^n$,

$$\|Wx - \widetilde{W}_{r,M}x\|_2 \leq \sigma_{r+1}(W)\|x\|_2.$$

Proof. By direct expansion,

$$\begin{aligned} W - \widetilde{W}_{r,M} &= (L_r + R_r) - (L_r + R_r M) \\ &= R_r - R_r M = R_r(I - M). \end{aligned}$$

This proves the exact residual-only error identity.

Since M is a diagonal projector, $I - M$ is also a diagonal projector and

$$\|I - M\|_2 \le 1.$$

Therefore,

$$\begin{aligned} \|W - \widetilde{W}_{r,M}\|_2 &= \|R_r(I - M)\|_2 \\ &\le \|R_r\|_2 \|I - M\|_2 \le \|R_r\|_2. \end{aligned}$$

Similarly,

$$\begin{aligned} \|W - \widetilde{W}_{r,M}\|_F &= \|R_r(I - M)\|_F \\ &\le \|R_r\|_F \|I - M\|_2 \le \|R_r\|_F. \end{aligned}$$

Because L_r is the rank-r truncated-SVD approximation, the Eckart–Young–Mirsky theorem gives

$$\|R_r\|_2 = \sigma_{r+1}(W), \qquad \|R_r\|_F^2 = \sum_{i>r} \sigma_i(W)^2.$$

Finally, for any x,

$$\begin{aligned} \|Wx - \widetilde{W}_{r,M}x\|_2 &= \|R_r(I - M)x\|_2 \\ &\le \|R_r(I - M)\|_2 \|x\|_2 \le \sigma_{r+1}(W)\|x\|_2. \end{aligned}$$

□

Remark 1. *The theorem shows that* RASP *exposes only the residual R_r to pruning. Thus the pruning error is controlled by the singular-value tail of W, not by the norm of the full matrix. Increasing r monotonically shrinks the residual tail and therefore tightens the worst-case approximation bound.*

Corollary 1 (Expected residual-only error)**.** *Let x be a random vector with second-moment matrix*

$$\Sigma_x = \mathbb{E}[xx^\top].$$

Then

$$\begin{aligned} &\mathbb{E}\|Wx - \widetilde{W}_{r,M}x\|_2^2 = \\ &\operatorname{tr}\Big((I - M)R_r^\top R_r(I - M) \qquad \Sigma_x\Big). \end{aligned}$$

Moreover,

$$\mathbb{E}\|Wx - \widetilde{W}_{r,M}x\|_2^2 \le \sigma_{r+1}(W)^2 \operatorname{tr}(\Sigma_x).$$

Proof. Let

$$E = R_r(I - M).$$

Then

$$Wx - \widetilde{W}_{r,M}x = Ex.$$

Therefore,

$$\mathbb{E}\|Ex\|_2^2 = \mathbb{E}[x^\top E^\top Ex] = \operatorname{tr}(E^\top E \Sigma_x).$$

Substituting $E = R_r(I - M)$ gives

$$E^\top E = (I - M)R_r^\top R_r(I - M),$$

and hence

$$\begin{aligned} &\mathbb{E}\|Wx - \widetilde{W}_{r,M}x\|_2^2 = \\ &\operatorname{tr}\Big((I - M)R_r^\top R_r(I - M) \qquad \Sigma_x\Big). \end{aligned}$$

For the upper bound,

$$\operatorname{tr}(E^\top E \Sigma_x) \le \|E^\top E\|_2 \operatorname{tr}(\Sigma_x) = \|E\|_2^2 \operatorname{tr}(\Sigma_x).$$

Using

$$\|E\|_2 = \|R_r(I - M)\|_2 \le \sigma_{r+1}(W)$$

finishes the proof. □

A.2 Equal-budget low-rank versus top-k column pruning

We next compare low-rank approximation and top-k column pruning under equal nominal parameter count. This comparison is not meant to claim that low-rank always beats pruning. Rather, it identifies an exact and measurable condition under which the low-rank approximation is strictly better.

Lemma 3 (Budget equality)**.** *Let $A_r \in \mathbb{R}^{m\times r}$ and $B_r \in \mathbb{R}^{r\times n}$. Then*

$$\operatorname{params}(A_r) + \operatorname{params}(B_r) = r(m + n).$$

If $P(W, k)$ is a pruning function that keeps exactly k columns of $W \in \mathbb{R}^{m\times n}$, then, ignoring support-index metadata,

$$\operatorname{params}(P(W, k)) = mk.$$

Therefore,

$$\operatorname{params}(P(W, k)) = \operatorname{params}(A_r) + \operatorname{params}(B_r)$$

if and only if

$$mk = r(m + n).$$

Equivalently,

$$k = \frac{r(m + n)}{m}$$

must be an integer in $\{0, \dots, n\}$.

Proof. The factor A_r contains mr entries and B_r contains rn entries. Therefore the low-rank representation stores

$$mr + rn = r(m + n)$$

floating-point values. A pruned matrix keeping k full columns stores

$$mk$$

floating-point values. Equating these counts gives

$$mk = r(m + n),$$

and solving for k gives

$$k = \frac{r(m + n)}{m}.$$

The value of k must be an integer between 0 and n for exact budget matching to be feasible. □

Theorem 3 (When low-rank beats equal-budget top-k column pruning)**.** *Let $W \in \mathbb{R}^{m\times n}$ and let $L_r = A_r B_r$ be its rank-r SVD approximation. Let $P(W, k)$ be any top-k column pruning operator that keeps k columns and zeroes the rest. Suppose the parameter budgets match:*

$$mk = r(m + n).$$

Let

$$c_j = \|w_j\|_2$$

denote the norm of column j of W, and let

$$c_{(1)} \geq c_{(2)} \geq \cdots \geq c_{(n)}$$

denote the sorted column norms. If

$$\sum_{i>r} \sigma_i(W)^2 < \sum_{j>k} c_{(j)}^2,$$

then

$$\|W - A_r B_r\|_F < \|W - P(W, k)\|_F$$

for every score-based top-k column pruning rule P.

Proof. Since $L_r = A_r B_r$ is the rank-r SVD approximation of W, the Eckart–Young–Mirsky theorem gives

$$\|W - A_r B_r\|_F^2 = \|W - L_r\|_F^2 = \sum_{i>r} \sigma_i(W)^2.$$

Now consider any kept-column set $S \subseteq [n]$ with $|S| = k$. Let M_S be the corresponding diagonal mask, so that

$$P_S(W) = W M_S.$$

Then

$$W - P_S(W) = W - W M_S = W(I - M_S).$$

The matrix $W(I - M_S)$ contains exactly the dropped columns of W and zeros in the kept-column positions. Therefore, by additivity of the Frobenius norm over columns,

$$\|W - P_S(W)\|_F^2 = \sum_{j\notin S} \|w_j\|_2^2.$$

Among all subsets S of size k, this dropped-column energy is minimized by keeping the k columns with largest column norms. Hence the best possible k-column pruning error is

$$\min_{|S|=k} \|W - P_S(W)\|_F^2 = \sum_{j>k} c_{(j)}^2.$$

Therefore, for any score-based top-k pruning rule P,

$$\|W - P(W, k)\|_F^2 \geq \sum_{j>k} c_{(j)}^2.$$

By assumption,

$$\sum_{i>r} \sigma_i(W)^2 < \sum_{j>k} c_{(j)}^2.$$

Combining the inequalities gives

$$\|W - A_r B_r\|_F^2 < \|W - P(W, k)\|_F^2.$$

Taking square roots proves

$$\|W - A_r B_r\|_F < \|W - P(W, k)\|_F.$$

□

Remark 2. *The theorem is exact and sharp. It does not say that low-rank approximation always beats pruning. It says that low-rank beats every top-k column pruning rule whenever the singular-value tail beyond rank r is smaller than the unavoidable energy of the dropped columns at budget-matched k. This condition is directly measurable for each LLM weight matrix.*

A.3 Operator-norm comparison

The Frobenius result compares total matrix energy. A complementary spectral-norm result compares worst-case input perturbation.

Theorem 4 (Operator-norm comparison)**.** *Let $W \in \mathbb{R}^{m\times n}$ and let L_r be its rank-r truncated-SVD approximation. Let $0 \le k < n$, and let $P(W,k)$ be any top-k column pruning operator. Let $c_{(k+1)}$ denote the $(k+1)$-st largest column norm of W. If*

$$\sigma_{r+1}(W) < c_{(k+1)},$$

then

$$\|W - L_r\|_2 < \|W - P(W,k)\|_2.$$

Proof. By the Eckart–Young–Mirsky theorem,

$$\|W - L_r\|_2 = \sigma_{r+1}(W).$$

Now consider any kept-column set S with $|S| = k$. The pruned error is

$$W - P_S(W) = W(I - M_S).$$

For any dropped column $j \notin S$,

$$(W - P_S(W))e_j = w_j,$$

where e_j is the j-th standard basis vector. Since $\|e_j\|_2 = 1$,

$$\|W - P_S(W)\|_2 \ge \|(W - P_S(W))e_j\|_2 = \|w_j\|_2.$$

Thus,

$$\|W - P_S(W)\|_2 \ge \max_{j\notin S} \|w_j\|_2.$$

Any set S of size k must drop at least one column whose norm is at least $c_{(k+1)}$. Therefore,

$$\|W - P_S(W)\|_2 \ge c_{(k+1)}.$$

The same lower bound applies to every score-based top-k pruning rule. Therefore, if

$$\sigma_{r+1}(W) < c_{(k+1)},$$

then

$$\begin{aligned}\|W - L_r\|_2 = \sigma_{r+1}(W) &< c_{(k+1)} \\ &\le \|W - P(W,k)\|_2.\end{aligned}$$

□

Corollary 2 (Worst-case input perturbation)**.** *Under the assumptions of Theorem 4, the rank-r truncated-SVD approximation has smaller worst-case unit-input error than any top-k column-pruned approximation:*

$$\sup_{\|x\|_2=1} \|(W - L_r)x\|_2 < \sup_{\|x\|_2=1} \|(W - P(W,k))x\|_2.$$

Proof. For any matrix E,

$$\|E\|_2 = \sup_{\|x\|_2=1} \|Ex\|_2.$$

Apply this identity to $E = W - L_r$ and $E = W - P(W,k)$, then use Theorem 4. □

A.4 A column-spread sufficient condition

The comparison theorem above is exact but depends on the sorted column-norm tail. We now give a simpler sufficient condition that is easier to interpret. It captures a regime in which the matrix is not column-sparse: its energy is spread across many columns. This is the regime in which coordinate pruning is structurally disadvantaged relative to low-rank approximation.

Assumption 1 (Column-energy spread)**.** *There exists $\alpha \in (0,1]$ such that every column of W satisfies*

$$\|w_j\|_2^2 \ge \alpha \frac{\|W\|_F^2}{n}, \qquad j = 1, \ldots, n.$$

Theorem 5 (Spectral decay plus column spread implies low-rank advantage)**.** *Assume $W \in \mathbb{R}^{m\times n}$ satisfies Assumption 1. Let $L_r = A_r B_r$ be the rank-r truncated-SVD approximation of W. Let $P(W,k)$ be any top-k column pruning operator. Suppose the budgets match:*

$$mk = r(m+n).$$

If

$$\sum_{i>r} \sigma_i(W)^2 < \alpha \left(1 - \frac{k}{n}\right) \|W\|_F^2,$$

then

$$\|W - A_r B_r\|_F < \|W - P(W,k)\|_F$$

for every score-based top-k column pruning rule P.

Proof. By Assumption 1, every dropped column contributes at least

$$\alpha \frac{\|W\|_F^2}{n}$$

to the squared Frobenius error. Since any top-k column pruning rule drops $n-k$ columns,

$$\|W - P(W,k)\|_F^2 \geq (n-k)\alpha \frac{\|W\|_F^2}{n} = \alpha\left(1 - \frac{k}{n}\right)\|W\|_F^2.$$

The SVD approximation satisfies

$$\|W - A_r B_r\|_F^2 = \sum_{i>r} \sigma_i(W)^2.$$

By assumption,

$$\sum_{i>r} \sigma_i(W)^2 < \alpha\left(1 - \frac{k}{n}\right)\|W\|_F^2.$$

Therefore,

$$\|W - A_r B_r\|_F^2 < \|W - P(W,k)\|_F^2,$$

and taking square roots gives the desired result. □

Remark 3. *This theorem gives an interpretable sufficient condition for LLM weight matrices: low-rank approximation is favored when the singular spectrum decays rapidly while column energy remains diffuse. Equivalently, low-rank wins when the matrix is close to a low-dimensional shared subspace but not close to being column-sparse.*

A.5 RASP can beat equal-budget direct pruning

Theorem 6 (When RASP beats equal-budget direct pruning)**.** *Let $W \in \mathbb{R}^{m\times n}$, and let $W = L_r + R_r$, where $L_r = A_r B_r$ is the rank-r truncated-SVD approximation of W. Let $P(R_r, q) = R_r M_S$ be a residual column-pruning operator that keeps q columns of R_r. The corresponding* RASP *approximation is*

$$\widetilde{W}_{r,S} = A_r B_r + P(R_r, q).$$

Ignoring support-index metadata, its parameter count is

$$r(m+n) + mq.$$

Let $P(W,K)$ be a direct column-pruning operator on W with the same parameter budget, so that

$$mK = r(m+n) + mq.$$

Equivalently,

$$K = q + \frac{r(m+n)}{m}.$$

Let $c_{(1)} \geq \cdots \geq c_{(n)}$ denote the sorted column norms of W. If

$$\|R_r - P(R_r, q)\|_F^2 < \sum_{j>K} c_{(j)}^2,$$

then

$$\|W - \widetilde{W}_{r,S}\|_F^2 < \|W - P(W,K)\|_F^2$$

for every score-based direct top-K column-pruning rule $P(W,K)$.

Proof. By construction,

$$\begin{aligned} W - \widetilde{W}_{r,S} &= W - (A_r B_r + P(R_r, q)) \\ &= (L_r + R_r) - (L_r + P(R_r, q)) \\ &= R_r - P(R_r, q). \end{aligned}$$

Therefore,

$$\|W - \widetilde{W}_{r,S}\|_F^2 = \|R_r - P(R_r, q)\|_F^2.$$

For any direct top-K column-pruning rule $P(W,K)$, the error is the energy of the dropped columns of W. The minimum possible direct top-K pruning error is obtained by keeping the K columns of W with largest column norms. Hence every direct top-K rule satisfies

$$\|W - P(W,K)\|_F^2 \geq \sum_{j>K} c_{(j)}^2.$$

By assumption,

$$\|R_r - P(R_r, q)\|_F^2 < \sum_{j>K} c_{(j)}^2.$$

Combining the inequalities gives

$$\|W - \widetilde{W}_{r,S}\|_F^2 < \|W - P(W,K)\|_F^2.$$

□

A.6 Rank–sparsity interpolation

RASP forms a continuum between direct pruning and low-rank approximation. Let M_ρ denote a mask corresponding to sparsity ratio ρ, where $\rho = 0$ keeps all residual channels and $\rho = 1$ removes all residual channels.

Lemma 4 (Rank–sparsity interpolation)**.** *Let*

$$\widetilde{W}_{r,\rho} = L_r + R_r M_\rho.$$

Then:

$$r = 0 \implies \widetilde{W}_{0,\rho} = W M_\rho,$$

$$\rho = 0 \implies \widetilde{W}_{r,0} = W,$$

and

$$\rho = 1 \implies \widetilde{W}_{r,1} = L_r.$$

Proof. If $r = 0$, then $L_0 = 0$ and $R_0 = W$. Therefore,

$$\widetilde{W}_{0,\rho} = L_0 + R_0 M_\rho = W M_\rho.$$

Thus, at rank zero, RASP reduces to the underlying pruning method.

If $\rho = 0$, then no residual channels are pruned and $M_\rho = I$. Hence

$$\widetilde{W}_{r,0} = L_r + R_r I = L_r + R_r = W.$$

Thus, at zero sparsity, RASP recovers the dense matrix exactly.

If $\rho = 1$, then all residual channels are removed and $M_\rho = 0$. Hence

$$\widetilde{W}_{r,1} = L_r + R_r 0 = L_r.$$

Thus, at full residual sparsity, RASP recovers the rank-r low-rank approximation. □

Remark 4. *The interpolation lemma is exact for the abstract linear surrogate. In a nonlinear gated MLP implementation, the $\rho = 1$ endpoint should be interpreted with care if the low-rank output path is computed from an unmasked hidden state. Nevertheless, the linear surrogate captures the core structural principle: rank controls the dense protected subspace, while sparsity controls how much of the residual correction is retained.*

A.7 Counterexamples and limits

The previous theorems provide sufficient conditions under which low-rank approximation beats top-k pruning. They do not imply that low-rank always wins. We record two counterexamples to clarify the limits of the theory.

Proposition 1 (Column-sparse matrices can favor pruning)**.** *There exist matrices W and budget-matched pairs (r, k) for which top-k column pruning has zero error while every rank-r approximation has positive error.*

Proof. For a concrete budget-matched construction, take $m = 4$, $n = 8$, and $r = 2$. Then

$$k = \frac{r(m+n)}{m} = \frac{2(4+8)}{4} = 6.$$

Choose $W \in \mathbb{R}^{4\times 8}$ with exactly six nonzero columns and rank 3. A top-k pruning rule that keeps precisely those six nonzero columns satisfies

$$P(W, k) = W,$$

and therefore

$$\|W - P(W, k)\| = 0.$$

However, since $\operatorname{rank}(W) = 3 > 2 = r$, no rank-$r$ matrix can equal W. In particular, the rank-r truncated-SVD approximation L_r has positive error:

$$\|W - L_r\| > 0.$$

Thus,

$$\|W - L_r\| > \|W - P(W, k)\|.$$

□

Proposition 2 (Orthogonal columns can favor pruning)**.** *Suppose $n \le m$ and the columns of W are pairwise orthogonal with norms*

$$c_{(1)} \ge c_{(2)} \ge \cdots \ge c_{(n)}.$$

Then the singular values of W are exactly

$$c_{(1)}, c_{(2)}, \ldots, c_{(n)}$$

up to ordering. Consequently,

$$\|W - L_r\|_F^2 = \sum_{i>r} c_{(i)}^2,$$

while the best k-column pruning error is

$$\min_{|S|=k} \|W - P_S(W)\|_F^2 = \sum_{i>k} c_{(i)}^2.$$

If $k \ge r$, then

$$\min_{|S|=k} \|W - P_S(W)\|_F \le \|W - L_r\|_F.$$

Thus, in the orthogonal-column regime, column pruning can be at least as good as rank-r low-rank approximation when $k \ge r$.

Proof. If the columns of W are orthogonal, then

$$W^\top W$$

is diagonal with entries

$$\|w_1\|_2^2, \ldots, \|w_n\|_2^2.$$

Therefore, the singular values of W are exactly the column norms, up to ordering:

$$\sigma_i(W) = c_{(i)}.$$

The rank-r truncated-SVD Frobenius error is therefore

$$\|W - L_r\|_F^2 = \sum_{i>r} \sigma_i(W)^2 = \sum_{i>r} c_{(i)}^2.$$

The best k-column pruning rule keeps the k largest-norm columns, so its squared error is

$$\sum_{i>k} c_{(i)}^2.$$

If $k \geq r$, then the tail beginning after k is no larger than the tail beginning after r:

$$\sum_{i>k} c_{(i)}^2 \leq \sum_{i>r} c_{(i)}^2.$$

Taking square roots gives the result. □

Remark 5. *These counterexamples clarify the scope of our claims. Low-rank approximation is not universally superior to pruning. It is superior under spectral decay and non-column-sparse energy distribution. Conversely, pruning is favored when the matrix is already close to column-sparse or when its columns are nearly orthogonal and a sufficient number of high-energy columns can be retained.*

A.8 Implications for RASP

The theoretical picture is therefore as follows. Direct top-k pruning assumes that the matrix can be well approximated by retaining a subset of coordinate directions. Low-rank approximation assumes that the matrix is well approximated by a low-dimensional shared subspace. RASP combines these two views:

$$W = L_r + R_r, \qquad \widetilde{W}_{r,M} = L_r + R_r M.$$

The low-rank component protects the principal subspace, while pruning is applied only to the residual correction. This gives a structural safeguard against destructive pruning of high-energy singular directions. The residual-only bound shows that the resulting error is controlled by the singular-value tail, and the equal-budget comparison identifies when low-rank structure is fundamentally more parameter-efficient than coordinate pruning.

B Experimental Results

B.1 Implementation and Experimental Setup

Since GRIFFIN (Dong et al., 2024) and WANDA (Sun et al., 2023) sparsify MLP layers, we directly apply RASP to the corresponding FFN projections. For TEAL (Liu et al., 2024b), RASP is applied only to the MLP layers, where activation sparsification is restricted to the residual correction component, while attention projections remain sparsified following the standard TEAL implementation. For TEAL, we use the same calibration scripts, calibration datasets, and prompt tokens provided by the official repository. Across all RASP variants, no additional retraining or finetuning of RASP parameters is performed.

Unless otherwise specified, all generation experiments use greedy decoding. For DeepSeek reasoning tasks, we follow the recommended decoding

Llama 3.2 3B Instruct							
Metric	Dense	G	G+RASP	W	W+RASP	T	T+RASP
GSM8K	74.15	12.36	31.99	11.14	35.56	68.99	69.37
MATH500	29.40	5.00	13.40	5.00	11.00	26.80	27.60
CoQA	79.73	78.69	78.94	78.83	78.94	78.42	78.57
HumanEval	50.00	17.68	32.32	21.34	36.59	48.17	49.39
Qasper	12.86	11.70	12.45	10.57	12.08	12.75	13.05
XSum	17.56	16.76	17.08	16.81	17.10	16.91	17.25
Avg.	43.95	23.70	**31.03**	23.95	**31.88**	42.01	**42.54**
Llama 3 8B Instruct							
Metric	Dense	G	G+RASP	W	W+RASP	T	T+RASP
GSM8K	74.91	11.60	20.55	10.54	25.55	68.16	70.81
MATH500	25.40	4.00	8.60	5.20	7.80	14.00	19.60
CoQA	77.61	74.36	75.74	76.47	77.75	75.63	79.11
HumanEval	59.76	20.12	27.44	23.78	35.98	51.83	51.83
Qasper	12.38	10.68	12.09	10.89	12.29	12.68	13.02
XSum	19.33	16.55	18.46	16.65	18.50	17.07	17.60
Avg.	44.90	22.89	**27.15**	23.92	**29.65**	39.90	**42.00**
Llama 3 70B Instruct							
Metric	Dense	G	G+RASP	W	W+RASP	T	T+RASP
GSM8K	87.12	59.97	74.75	61.56	74.98	86.20	86.73
MATH500	41.40	22.60	31.60	20.20	32.00	36.80	37.20
CoQA	80.66	80.88	80.94	80.78	80.87	80.91	81.04
HumanEval	80.49	37.80	48.78	38.41	50.61	75.61	76.83
Qasper	13.85	13.00	13.36	13.08	13.44	13.47	13.54
XSum	19.51	18.83	19.30	18.83	19.46	19.37	19.31
Avg.	53.84	38.85	**44.79**	38.81	**45.23**	52.06	**52.44**
Gemma 2 2B IT							
Metric	Dense	G	G+RASP	W	W+RASP	T	T+RASP
GSM8K	58.83	2.73	10.46	3.03	10.99	49.43	52.46
MATH500	18.20	2.40	2.60	1.00	4.20	14.00	14.00
CoQA	78.51	64.53	75.02	63.54	74.97	78.30	78.76
HumanEval	23.78	3.05	6.10	3.05	6.10	20.73	21.34
Qasper	11.01	8.92	8.27	8.74	7.97	10.52	10.80
XSum	17.80	14.00	17.66	13.92	16.00	17.75	17.66
Avg.	34.69	15.94	**20.02**	15.55	**20.04**	31.79	**32.50**
Gemma 2 9B IT							
Metric	Dense	G	G+RASP	W	W+RASP	T	T+RASP
GSM8K	79.53	43.97	65.81	47.01	66.34	79.15	79.00
MATH500	25.20	13.40	20.80	15.40	21.40	25.40	27.20
CoQA	79.17	80.14	80.88	80.05	80.86	79.29	79.61
HumanEval	56.10	24.39	36.59	26.83	35.98	53.66	51.22
Qasper	9.83	11.48	9.61	11.17	9.47	9.82	10.31
XSum	19.33	16.24	18.46	16.36	18.50	19.11	19.49
Avg.	44.86	31.60	**38.69**	32.80	**38.76**	44.41	**44.47**

Table 5: Generation and reasoning results on GSM8K, MATH500, CoQA, HumanEval, Qasper, and XSum. G, W, T denote GRIFFIN, WANDA, and TEAL, respectively. Dense columns are not sparse; all other columns use sparsity $= 0.5$. All RASP variants use rank $r = 256$. Higher is better for all metrics. A dash indicates an unavailable result.

Llama 3.2 3B Instruct							
Metric	Dense	G	G+RASP	W	W+RASP	T	T+RASP
HS	53.27	50.91	51.28	50.30	51.28	50.50	50.35
PiQA	76.01	74.05	74.76	75.30	75.14	75.03	75.14
BoolQ	78.93	75.05	76.21	75.11	76.12	76.94	78.26
ARC-e	75.34	71.76	71.72	72.94	73.11	73.19	73.44
ARC-c	43.34	39.08	41.30	39.51	40.96	40.53	42.15
WinoG	68.82	64.96	66.38	64.09	67.64	66.22	65.75
MMLU	60.76	45.07	52.30	46.21	53.17	55.31	56.94
Avg.	65.21	60.13	**61.99**	60.49	**62.49**	62.53	**63.15**
Llama 3 8B Instruct							
Metric	Dense	G	G+RASP	W	W+RASP	T	T+RASP
HS	58.01	55.84	55.84	55.31	55.91	55.11	55.19
PiQA	77.69	74.32	76.01	74.27	76.55	77.42	77.58
BoolQ	84.40	80.40	80.64	81.44	80.52	83.52	83.03
ARC-e	82.11	76.47	77.82	76.22	77.57	79.25	79.17
ARC-c	53.58	46.50	45.73	46.42	46.50	49.32	49.23
WinoG	71.67	67.80	71.27	69.53	70.32	70.72	69.30
MMLU	66.77	51.20	57.58	52.15	58.65	61.41	62.85
Avg.	70.60	64.65	**66.41**	65.05	**66.57**	**68.11**	68.05
Llama 3 70B Instruct							
Metric	Dense	G	G+RASP	W	W+RASP	T	T+RASP
HS	64.46	64.11	63.72	64.17	64.03	63.04	63.51
PiQA	82.10	80.90	80.69	80.63	80.79	80.36	81.07
BoolQ	87.83	85.35	87.16	85.96	86.61	87.71	88.26
ARC-e	86.45	84.55	87.72	84.81	85.10	85.73	85.40
ARC-c	65.19	60.75	61.43	62.12	63.23	62.80	63.23
WinoG	78.85	76.40	77.35	76.09	76.48	76.16	77.51
MMLU	80.22	72.38	76.65	72.60	77.05	77.99	78.69
Avg.	77.87	74.92	**76.39**	75.20	**76.18**	76.26	**76.81**

Table 6: Classification results on HellaSwag (HS), PIQA, BoolQ, ARC-Easy (ARC-e), ARC-Challenge (ARC-c), WinoGrande (WinoG), and MMLU. G, W, and T denote GRIFFIN, WANDA, and TEAL, respectively. MMLU is evaluated 5-shot; the remaining tasks are evaluated 0-shot. Dense columns are not sparse; all other columns use sparsity $= 0.5$. All RASP variants use rank $r = 256$. Higher is better for all metrics.

configuration with temperature 0.6 and top-p=0.9. For classification benchmarks, GRIFFIN, WANDA, and their corresponding RASP variants apply sparsification across the full input sequence. For TEAL and RASP+TEAL, sparsification is applied only to the second half of the sequence to align with the evaluation protocol used in the original TEAL implementation.

No pruning is applied during the prefill stage for any method on the generation benchmarks, and sparsification is enabled only during autoregressive decoding. For TEAL, we use the official calibration scripts and datasets from the original repository and adapt TEAL+RASP to support calibration within the residual decomposition framework.

Reasoning evaluations generate up to 4K tokens for GSM8K and MATH, and up to 32K tokens for AIME24. All experiments are conducted in `bfloat16` precision on NVIDIA GPUs, using H200 GPUs for 70B-scale models and H100 GPUs for all smaller models.

B.2 Structural Diagnostic Experiment

The structural-diagnostic experiment in Figure 5 is a post hoc layerwise analysis designed to test when RASP provides the largest improvement over its corresponding base sparsification method. Unlike the main benchmark results, which evaluate the full RASP-wrapped model, this analysis isolates individual model–method–layer settings. We evaluate three backbone models, three base methods, and five representative MLP layers per model, giving 45 measurements in total. For each measurement, we compare the task-aggregate score of the RASP-wrapped variant against the corresponding base method under the same sparsity, rank, and evaluation protocol, and define

$$\mathrm{Lift}_{m,b,\ell} = \mathrm{Score}(\mathrm{RASP}(m, b, \ell)) - \mathrm{Score}(b; m, \ell),$$

where m denotes the model, b the base sparsification method, and ℓ the target layer.

For each target layer, we compute three structural diagnostics. First, after forming the rank-r truncated-SVD decomposition $W_\ell = L_{\ell,r} + R_{\ell,r}$, we measure spectral decay by the normalized residual tail

$$\mathrm{TailRatio}(W_\ell) = \frac{\|R_{\ell,r}\|_F^2}{\|W_\ell\|_F^2} = \frac{\sum_{i>r} \sigma_i(W_\ell)^2}{\sum_i \sigma_i(W_\ell)^2}.$$

Second, we measure residual-column coherence by normalizing the columns of $R_{\ell,r}$ and averaging the absolute off-diagonal entries of the resulting Gram matrix. This captures whether the residual branch is diffuse across coordinates or dominated by highly aligned residual columns. Third, we measure mask stability by applying the base method's channel-scoring rule to prefill tokens and decoding tokens separately, then computing the normalized overlap between the selected keep sets. When a layer contains multiple MLP projections, the diagnostic is computed per projection and then averaged at the layer level.

We report Pearson and Spearman correlations between each diagnostic and RASP lift across the 45 model–method–layer measurements. Confidence intervals are obtained by bootstrap resampling the measurements. To check that the trends are not driven only by a particular backbone or sparsification rule, we also compute partial Spearman correlations after removing model and base-method fixed effects. Specifically, we rank-transform both the diagnostic and lift, regress each on model and

Llama 3.2 3B Instruct							
Metric	Dense	T 0%	T+R 0%	T 50%	T+R 50%	T 100%	T+R 100%
GSM8K	74.15	68.99	69.37	64.59	69.14	58.07	63.31
MATH500	29.40	26.80	27.60	27.60	29.40	21.20	25.60
CoQA	79.73	78.42	78.57	77.54	78.59	75.89	76.88
HumanEval	50.00	48.17	49.39	46.34	50.61	35.37	43.29
Qasper	12.86	12.75	13.05	13.84	13.65	13.25	13.37
Avg.	49.23	47.03	**47.60**	45.98	**48.28**	40.76	**44.49**
Δ	–	-2.20	-1.63	-3.25	-0.95	-8.47	-4.74
Gemma 2 2B IT							
Metric	Dense	T 0%	T+R 0%	T 50%	T+R 50%	T 100%	T+R 100%
GSM8K	58.83	49.43	52.46	47.54	52.92	46.78	52.62
MATH500	18.20	14.00	14.00	12.80	16.60	13.80	15.80
CoQA	78.51	78.30	78.76	76.64	78.52	74.93	77.39
HumanEval	23.78	20.73	21.34	22.56	21.34	19.51	24.39
Qasper	11.01	10.52	10.80	11.26	11.39	10.98	11.15
Avg.	38.07	34.60	**35.47**	34.16	**36.15**	33.20	**36.27**
Δ	–	-3.47	-2.59	-3.91	-1.91	-4.87	-1.80
Gemma 2 9B IT							
Metric	Dense	T 0%	T+R 0%	T 50%	T+R 50%	T 100%	T+R 100%
GSM8K	79.53	79.15	79.00	78.70	80.67	78.47	80.67
MATH500	25.20	25.40	27.20	27.00	25.20	25.80	25.20
CoQA	79.17	79.29	79.61	78.49	78.68	77.95	78.08
HumanEval	56.10	53.66	51.22	57.93	56.71	53.66	58.54
Qasper	9.83	9.82	10.31	10.08	9.90	10.04	9.82
Avg.	49.97	49.46	**49.47**	**50.44**	50.23	49.18	**50.46**
Δ	–	-0.50	-0.50	0.47	0.27	-0.78	0.50

Table 7: Effect of prefill sparsification on generation and reasoning tasks. T denotes TEAL, and $T + R$ denotes TEAL+RASP. The prefill percentage indicates the fraction of prefill tokens sparsified: 0% uses no prefill sparsification, 50% sparsifies the final half of the prefill sequence, and 100% sparsifies the full prefill sequence. Δ is the change in Avg. relative to the dense model. All non-dense columns use sparsity $= 0.5$, and all RASP variants use rank $r = 256$.

method indicators, and correlate the resulting residuals. This analysis is intended to explain variation in RASP gains rather than introduce a new evaluation protocol or additional benchmark setting.

B.3 Expanded Results

We present the full benchmark results for GSM8K, MATH500, CoQA, HumanEval, Qasper, and XSum in Table 5 Hellaswag, PiQA, BoolQ, ARC (Easy and Challenge), Winogrande, and MMLU in Table 6. Discussions on these results are presented in Section 5.2. Similarly, Table 7 shows the detailed of our ablation on the effects of pruning the prefill sequence for TEAL and RASP + TEAL.

B.4 Effects of Prefill Sparsification on Generation

We provide extensive results for prefill sparsification on TEAL and its corresponding RASP variant in Table 7. Since TEAL primarily targets decoding-time sparsification, these experiments evaluate the extent to which both methods remain robust when sparsification is additionally applied during the prefill stage. Across nearly all settings, TEAL+RASP reduces the performance degradation relative to the dense model and, in several cases, substantially recovers the lost accuracy introduced by prefill pruning.

The effect is most pronounced for smaller models. On Llama 3.2 3B Instruct, applying TEAL to the full prefill sequence (100%) drops performance relative to dense by 8.47 points. In contrast, TEAL+RASP recovers a significant portion of this degradation, with a lower decrease in performance by 4.74. Similar behaviour is observed on Gemma 2 2B IT, where RASP reduces the gap to the dense model. On the larger Gemma 2 9B IT model, the impact of prefill sparsification is substantially smaller overall, suggesting that the larger model may possess more stable activation under sparsification.

Overall, these results indicate that RASP consistently improves the stability of activation sparsification during prefill processing, especially in smaller models and under aggressive sparsification regimes.

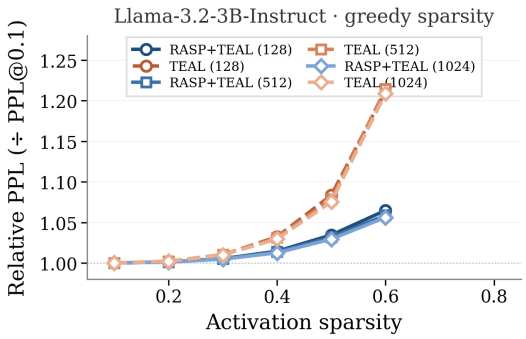


Figure 7: WikiText perplexity relative to sparsity 0.1 for Llama-3.2-3B-Instruct under greedy activation sparsity. Solid lines: RASP+TEAL; dashed: TEAL. Marker shape denotes evaluation generation length $\in \{128, 512, 1024\}$.

Is RASP less sensitive to higher sparsity? To understand the individual contributions of our proposed components, we conduct an ablation study evaluating the performance of the Llama 3.2 3B Instruct model under varying degrees of greedy activation sparsity. Specifically, we compare our joint approach (TEAL+RASP) against the baseline TEAL method across multiple evaluation generation lengths (128, 512, and 1024 tokens). As shown in the Figure 7, the results demonstrate that while both methods perform comparably at lower sparsity levels (≤ 0.3), TEAL+RASP (solid lines) exhibits significantly greater robustness as sparsity in-

creases. At a high activation sparsity of 0.6, TEAL alone (dashed lines) suffers from a sharp perplexity explosion while TEAL+RASP effectively stabilizes the model. Furthermore, this performance gap remains invariant across different evaluation generation lengths, indicating that the structural advantages of TEAL+RASP are highly consistent and successfully decouple aggressive activation pruning from catastrophic degradation in generation quality.

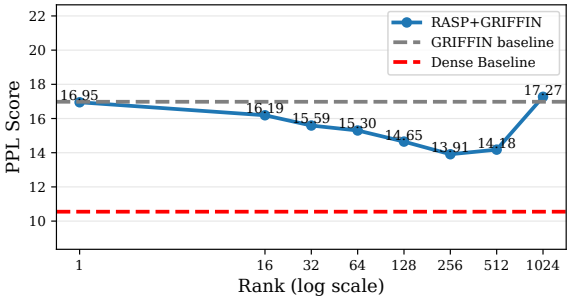
B.5 Does rank control error (continued)?

Experimental setup. Figure 8 reports WikiText perplexity (PPL) for RASP applied to a Llama-3-3B model. We use a total sequence length of 1024 tokens, keeping the first 128 tokens dense and applying pruning/sparsification only to the remaining tokens. All methods are evaluated at an equivalent sparsity of 50%. At rank $r = 1$, the low-rank matrix L_r contains only one principal component, so RASP is expected to closely match the pruning method it wraps. As r increases, L_r captures more structure and the residual becomes less sparse. At $r = 1024$, the residual has 0% sparsity, so the observed performance is effectively determined by L_r alone.

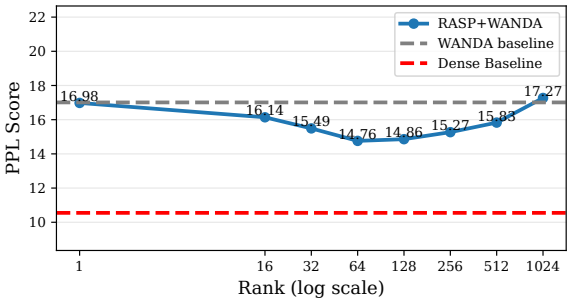
Rank scaling behavior. Across Figures 8a, 8b, and 8c, RASP follows a consistent rank-scaling pattern. At $r = 1$, the PPL is close to the corresponding pruning baseline, confirming that a rank-one L_r has limited capacity and contributes little beyond the wrapped method. As rank increases, PPL improves, indicating that the low-rank component captures useful structure that complements the sparse residual. However, the best performance occurs at intermediate ranks rather than at the maximum rank.

Best operating regime. The strongest results in Figure 8 occur at moderate ranks. In Figure 8a, RASP+GRIFFIN improves from 19.95 at $r = 1$ to 16.91 at $r = 256$. In Figure 8b, RASP+WANDA reaches its best PPL of 17.76 earlier, at $r = 64$. In Figure 8c, RASP+TEAL achieves the lowest absolute PPL, improving from 12.72 at $r = 1$ to 11.27 at $r = 256$. The main distinction is that WANDA peaks at a smaller rank, while GRIFFIN and TEAL continue improving until $r = 256$.

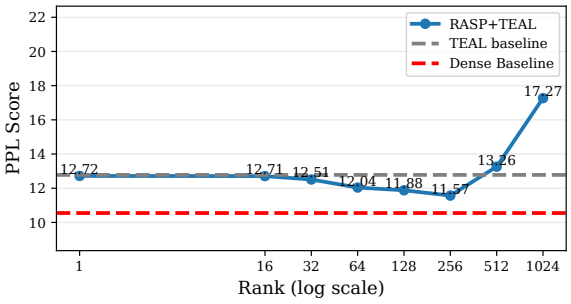
High-rank degradation. At high ranks, all three subfigures show degradation relative to the best intermediate-rank setting. In particular, at $r = 1024$, the residual is no longer sparsified, so the model is effectively relying on L_r alone. The worse PPL at this rank shows that the low-rank component by itself is insufficient and that the sparse residual remains important. Overall, Figure 8 suggests that RASP is most effective when L_r is large enough to capture useful principal structure, but not so large that the residual contribution is eliminated.



(a) GRIFFIN



(b) WANDA



(c) TEAL

Figure 8: PPL scaling with rank for RASP combined with GRIFFIN, WANDA, and TEAL. Each plot compares the RASP-enhanced method against the corresponding pruning baseline and dense baseline.

B.6 LaRoSA

We implement a RASP variant of LaRoSA (Liu et al., 2025) and evaluate it on classification benchmarks, observing improvements over the baseline LaRoSA configuration (Table 8). During implementation, we observed that sparsity is enforced in

the rotated activation space,

$$\hat{x} = \mathcal{S}(Qx),$$

while the reconstructed activation used in subsequent projections becomes

$$x' = Q^\top \hat{x}.$$

Although $\hat{x}$ is sparse by construction, the inverse rotation $Q^\top$ generally mixes coordinates and can therefore produce a substantially denser activation x'. Consequently,

$$\mathrm{nnz}(x') \neq \mathrm{nnz}(\hat{x}),$$

where $\mathrm{nnz}(\cdot)$ denotes the number of non-zero elements. Sparsity in the rotated basis therefore does not necessarily translate to equivalent sparsity in the original activation space. We additionally observe that sparsity statistics in the LaRoSA implementation are computed on the rotated activation $\hat{x}$, rather than on the reconstructed activation x' participating in downstream matrix multiplications.

We nevertheless include a comparison with a RASP variant to demonstrate the compatibility and versatility of our approach when combined with existing activation sparsification methods.

Model	Method	HS	PiQA	BoolQ	ARC-e	ARC-c	WinoG	MMLU	Avg.
Llama 3 3B Instruct	Dense	53.27	76.01	78.93	75.34	43.34	68.82	60.76	65.21
	LaRoSA	50.95	75.19	78.01	72.56	41.98	66.22	56.00	62.99
	+ RASP	51.42	75.63	77.31	72.98	41.13	66.77	56.34	**63.08**
Llama 3 8B Instruct	Dense	58.01	77.69	84.40	82.11	53.58	71.67	66.77	70.60
	LaRoSA	55.96	76.99	82.57	79.12	48.63	69.46	61.01	67.68
	+ RASP	55.71	77.20	82.11	79.00	50.51	69.46	62.20	**68.03**

Table 8: Classification results for dense baselines, LaRoSA, and its RASP-wrapped variant at argument sparsity $= 0.5$ (effective sparsity is much higher for LaRoSA). All RASP variants use rank $r = 256$. We include these results to illustrate that RASP can wrap additional sparsification methods beyond the main baselines.