

# Activation- and Influence-Aware Ranks (AIR): Function-Preserving SVD Compression for LLMs

Nico Harder<sup>1</sup> Daniel Becking<sup>1</sup> Karsten Mueller<sup>1</sup> Wojciech Samek<sup>1</sup>

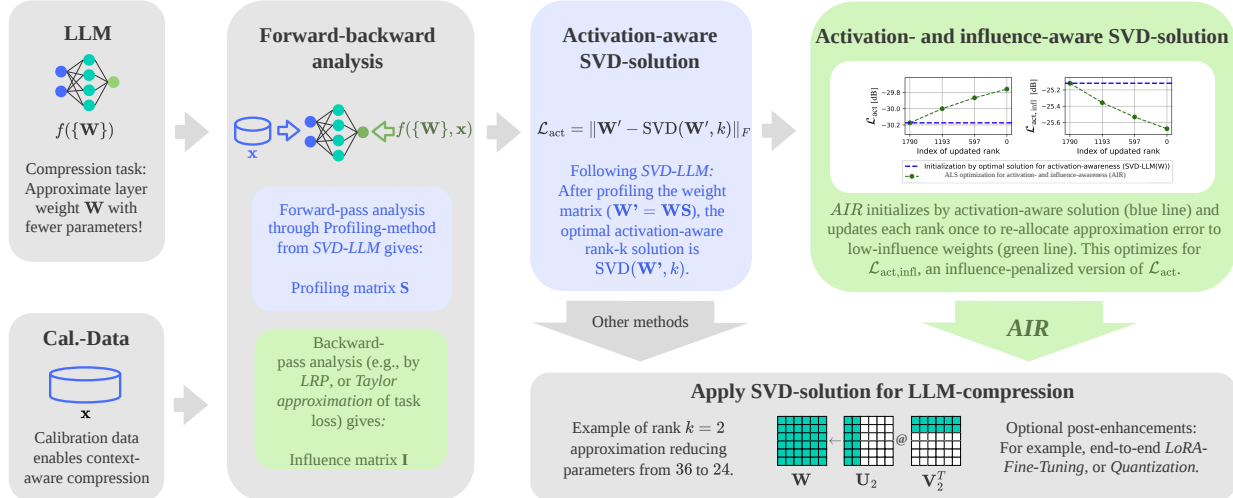


Figure 1. Schematic overview of activation- and influence-aware SVD-based LLM compression through AIR.

## Abstract

We present Activation- and Influence-Aware Ranks (AIR), an SVD-based LLM compression framework that guides each weight matrix’s low-rank approximation with a backward-signal influence metric. Starting from the activation-aware optimum of SVD-LLM( $W$ ), AIR runs a single closed-form alternating least squares (ALS) sweep that integrates influence element-wise under a monotone-descent guarantee. AIR is layer-local and composes orthogonally with end-to-end methods: alone it exceeds ACIP, and AIR+LoRA outperforms it further. AIR improves perplexity over SVD-LLM( $W$ ) by  $>18\%$  at  $\leq 60\%$  parameter retention, matches its quality with  $\sim 90\%$  less calibration data, and turns parameter savings into FLOP, peak-memory, and per-token latency gains.

<sup>1</sup>Fraunhofer HHI, Berlin, Germany. Correspondence to: Nico Harder <nico.harder@hhi.fraunhofer.de>.

Accepted at the ICML 2026 Workshop on Resource-Adaptive Foundation Model Inference (AdaptFM), Seoul, South Korea. Copyright 2026 by the author(s).

## 1. Introduction

Large language models (LLMs) gain performance through scale (Kaplan et al., 2020), motivating post-training compression for efficient deployment under varying resource budgets. Singular value decomposition (SVD) is well-suited: transformer weights develop implicit low-rank structure during training (Balzano et al., 2025), the Eckart–Young theorem (Eckart & Young, 1936) gives a Frobenius-optimal rank- $k$  approximation, and the rank-hierarchy of singular values provides a built-in importance ordering at the finest granularity possible without calibration data. Vanilla SVD, however, ignores the functional role of each weight and degrades downstream performance severely. Function-preserving SVD methods address this with calibration data: *forward-activation* approaches (ASVD (Yuan et al., 2024), SVD-LLM( $W$ ) (Wang et al., 2025b)) are cheap and layer-local but agnostic to whether the preserved activations influence the final prediction; *backward-signal* approaches expose this functional role yet have either underperformed activation-only baselines (FWSVD (Hsu et al., 2022)) or required full end-to-end optimization (ACIP (Genzel et al., 2025)). The gap is not *whether* a backward signal helps, but *how* it is integrated.

**Table 1.** Comparison of SVD-based LLM compression methods across features and requirements.

METHOD	FEATURES		ENHANCEMENTS			COSTS	
	ACTIVATION-AWARENESS	INFLUENCE-AWARENESS	DYN. RANK ALLOCATION	END-TO-END OPTIMIZATION	QUANTIZATION	COMPUTE-COST	DATA-COST
Vanilla SVD	×	×	Optional	Optional	Optional	Very Low	Zero
ACIP (Genzel et al.)	✓	End-to-end	Inherent	Inherent	Optional	High	High
ASVD, ASVD+ (Yuan et al.)	✓	×	Inherent	Optional	Optional	High	Medium
FWSVD (Hsu et al.)	×	Local proxy	Optional	Optional	Optional	Medium	Medium
SVD-LLM(W) (Wang et al.)	✓	×	Optional	Optional	Optional	Low	Medium
SVD-LLM (Wang et al.)	✓	End-to-end	Optional	Inherent	Optional	Low	Medium
AIR (Ours)	✓	Local proxy	Optional	Optional	Optional	Low	Very low

*AIR* (Activation- and Influence-Aware Ranks) closes this gap (Figure 1). A forward-backward analysis yields the profiling matrix  $\mathbf{S}$  from activations (Wang et al., 2025b) and an element-wise influence matrix  $\mathbf{I}$  from a backward signal. *AIR* jointly optimizes a hybrid activation- and influence-aware objective: starting from SVD( $\mathbf{W}\mathbf{S}, k$ ), per-rank ALS updates redistribute approximation error away from high-influence weights, sidestepping the intractability of element-wise weighted low-rank approximation (Srebro & Jaakkola, 2003) via closed-form per-rank updates with monotone descent. **Contributions.** (i) On LLaMA-7B, *AIR* achieves 18%/33%/45% lower WikiText-2 perplexity than SVD-LLM(W) at 60%/40%/20% retention, matches it with  $\sim 90\%$  less calibration data, and generalizes across LLaMA/Mistral/Vicuna families. (ii) A closed-form ALS mechanism for element-wise backward-signal integration that dominates the choice of signal (LRP- $\epsilon$ , Weight $\times$ Gradient, Fisher all equivalent) and is complementary to end-to-end optimization (*AIR*+LoRA outperforms ACIP at every rate). (iii) System-level efficiency: 60% retention turns into 64% peak memory and 53% per-token latency on a 40 GB A100.

## 2. Related Work

We position *AIR* against existing SVD-based compression methods (Table 1). Under *Enhancements*, “Inherent” marks methods that bundle the enhancement as part of their core algorithm and “Optional” those that treat it as a complementary add-on. ASVD (Yuan et al., 2024) and SVD-LLM (Wang et al., 2025b) rely on *forward activations*: ASVD scales weights by per-channel activation statistics, while SVD-LLM applies a whitening transformation that yields the optimal solution to an activation-aware objective. Methods using *backward signals* include FWSVD (Hsu et al., 2022), which guides compression by row-wise Fisher information, and ACIP (Genzel et al., 2025), which uses  $\ell_1$ -regularized gradient descent to learn rank masks and LoRA adapters end-to-end. These differ in how they exploit the signal (Table 1’s Influence-Awareness column): as a *layer-local proxy* guiding a closed-form solution (FWSVD), or

through *end-to-end* optimization (ACIP). The two strategies are not mutually exclusive: SVD-LLM and ACIP inherently bundle a subsequent end-to-end stage, while *AIR*, FWSVD, and SVD-LLM(W) leave it as an optional, complementary post-step. SVD-LLM(W) (the whitening-only variant) already achieves competitive results and motivates our primary baseline. FWSVD is the closest prior approach to *AIR*: it combines a backward signal with SVD as a local proxy, yet underperforms activation-aware baselines because (i) its objective uses no forward-pass signal and (ii) it aggregates influence row-wise, since the element-wise case admits no closed-form SVD solution (Srebro & Jaakkola, 2003). *AIR*’s ALS updates sidestep this and preserve element-wise granularity. Appendix A.1 organizes the broader family of backward-signal influence metrics (Weight $\times$ Gradient (Molchanov et al., 2017; Shrikumar et al., 2017), empirical Fisher (Hsu et al., 2022), OBD/OBS (LeCun et al., 1989; Hassibi & Stork, 1992), LRP (Bach et al., 2015; Achibat et al., 2024; Becking et al., 2022; Yeom et al., 2021; Hatefi et al., 2024)) via the underlying Taylor expansion.

## 3. AIR

This section describes the design of *AIR* (Figure 1); *AIR* operates on individual layers and we omit layer-indexing. SVD-based compression decomposes a weight matrix  $\mathbf{W} \in \mathbb{R}^{m \times n}$  into a rank- $k$  approximation  $\mathbf{W}_k = \mathbf{U}_k \Sigma_k \mathbf{V}_k^\top$  ( $k \leq \min(m, n)$ ), Frobenius-optimal by Eckart–Young (Eckart & Young, 1936).

### 3.1. Forward-backward analysis

**Activations in the forward pass** Following SVD-LLM (Wang et al., 2025b), we collect hidden states  $\mathbf{X}$  over the calibration data and apply a *Profiling* preprocessing to the weight matrix via

$$\mathbf{W}' = \mathbf{W}\mathbf{S}, \quad \text{where } \mathbf{S} = \text{cholesky}\left(\sum_{\mathcal{D}_{\text{cal}}} \mathbf{X}\mathbf{X}^\top\right). \quad (1)$$

Throughout this work,  $'$  indicates a variable in the latent space obtained through right-multiplication by the profiling matrix  $\mathbf{S}$ , which we interpret as the product of the forward-pass analysis.

**Influence from the backward pass.** A backward signal exposes the functional role of each weight by reading its contribution from the model’s output side. Since the task loss  $\mathcal{L}$  has no tractable closed form in the compressed weights, the standard approach is a Taylor expansion of  $\mathcal{L}$  around  $\mathbf{W}$ :

$$\mathcal{L}(\hat{\mathbf{W}}_k) \approx \underbrace{\mathcal{L}(\mathbf{W})}_{0\text{th}} + \underbrace{\nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W})^\top (\hat{\mathbf{W}}_k - \mathbf{W})}_{1\text{st (slope)}} + \underbrace{\frac{1}{2} (\hat{\mathbf{W}}_k - \mathbf{W})^\top \mathbf{H}_{\mathbf{W}} (\hat{\mathbf{W}}_k - \mathbf{W})}_{2\text{nd (curvature)}} + \dots, \quad (2)$$

with the 0th-order term constant in compression. Weight $\times$ Gradient (Molchanov et al., 2017; Shrikumar et al., 2017) is the first-order term; FWSVD’s diagonal empirical Fisher (Hsu et al., 2022) approximates the curvature. LRP (Bach et al., 2015) arrives at the same first-order quantity from another foundation, interpreting the model’s prediction as relevance to redistribute it within each layer under gradient-free propagation rules (Appendix A.1). *AIR* accepts any element-wise score  $\mathbf{I} \in \mathbb{R}^{m \times n}$  from this family. Empirically, the integration mechanism dominates the choice: principled signals yield identical perplexity through our update rule, while uninformative baselines fall behind (Table 7). We default to AttnLRP (Achtibat et al., 2024) with  $\epsilon = 10^{-6}$  (Hatefi et al., 2024) for its gradient-free rules and noise suppression of near-zero pre-activations: relevance is initialized at  $\text{rel} = f(\mathbf{x})$  and propagated to per-layer  $\mathbf{R} \in \mathbb{R}^{m \times n}$ . *AIR* sets  $\tilde{\mathbf{I}} = \mathbf{R}$ , accumulates  $\mathbf{I} = \sum_{d \in \mathcal{D}_{\text{cal}}} |\tilde{\mathbf{I}}^{(d)}|$ , and normalizes to unit mean per layer.

### 3.2. AIR - Problem

SVD-LLM’s *weight-profiling* (Eq. 1) simplifies the activation-aware objective from  $\|\mathbf{W}\mathbf{X} - (\mathbf{U}_k \Sigma_k \mathbf{V}_k^\top) \mathbf{X}\|_F^2$  to

$$\mathcal{L}_{\text{act}} = \|\mathbf{W}' - \mathbf{U}'_k \Sigma'_k \mathbf{V}'_k{}^\top\|_F^2, \quad (3)$$

minimized optimally by SVD-LLM(W) (Wang et al., 2025b) (vanilla-SVD on the *profiled* weight  $\mathbf{W}'$ , by Eckart-Young (Eckart & Young, 1936)). *AIR* expands  $\mathcal{L}_{\text{act}}$  with an element-wise influence penalty: each squared residual is scaled by  $(1 + \delta \cdot i_{ij})$ , multiplicatively integrating influence scores  $\mathbf{I}$  into the loss. The additive all-ones anchor ensures activation-aware approximation error is not discounted for low-influence weights (revisited empirically in Section 4.2, Figure 3). With  $\delta$  controlling the influence impact, *AIR*’s central hybrid activation- and influence-aware objective reads

$$\mathcal{L}_{\text{act,infl}} = \|\sqrt{\mathbf{1} + \delta \cdot \mathbf{I}} \odot (\mathbf{W}' - \mathbf{U}'_k \Sigma'_k \mathbf{V}'_k{}^\top)\|_F^2, \quad (4)$$

where  $\mathbf{I} \in \mathbb{R}^{m \times n}$  is the element-wise influence matrix from the backward-pass analysis (*not* the identity matrix),  $\mathbf{1} \in \mathbb{R}^{m \times n}$  is the all-ones matrix,  $\odot$  is the Hadamard product, and the square-root is element-wise; at  $\delta = 0$ ,  $\mathcal{L}_{\text{act,infl}}$  collapses to  $\mathcal{L}_{\text{act}}$ . The influence-weighted form admits no closed-form SVD solution in general (Srebro & Jaakkola, 2003); we instead solve  $\arg \min_{\mathbf{U}'_k, \Sigma'_k, \mathbf{V}'_k} \mathcal{L}_{\text{act,infl}}$  via the rank-wise closed-form ALS updates derived in Section 3.3 (full element-wise penalty derivation in Appendix A.2).

### 3.3. AIR - Solution

As initialization, *AIR* adopts SVD-LLM(W)’s activation-aware optimum  $\mathbf{U}'_k, \Sigma'_k, \mathbf{V}'_k \leftarrow \text{SVD}(\mathbf{W}', k)$  (Section 3.2), then runs an Alternating Least Squares (ALS) sweep over the ranks: a single closed-form update to each rank component  $(\sigma'_r, \mathbf{u}'_r, \mathbf{v}'_r)$ , iterating  $r = k - 1$  down to  $r = 0$ . This backward direction protects the leading components (small  $r$ ) from excessive distortion: the first few ALS iterations impose the largest updates and shrink as the residual error converges (Table 8). For each rank  $r$ , we first update  $\mathbf{v}'_r$  given  $\mathbf{U}'_k, \Sigma'_k$ , and the residual  $\mathbf{E}_r = \mathbf{W}' - \mathbf{W}'_k + \sigma'_r \mathbf{u}'_r \mathbf{v}'_r{}^\top$  (i.e.,  $\mathbf{W}'$  minus the current rank- $k$  approximation, with rank  $r$ ’s contribution added back), through

$$\mathbf{v}'_r = \left( \frac{\mathbf{u}'_r{}^\top ((\mathbf{1} + \delta \cdot \mathbf{I}) \odot \mathbf{E}_r)}{\sigma'_r \cdot (\mathbf{u}'_r{}^\top (\mathbf{1} + \delta \cdot \mathbf{I}))} \right)^\top, \quad (5)$$

where the division is element-wise and  $\mathbf{u}'_r{}^2$  denotes element-wise squaring. Then, given  $\Sigma'_k$  and the updated  $\mathbf{V}'_k$ , we obtain

$$\tilde{\mathbf{u}}'_r = \frac{((\mathbf{1} + \delta \cdot \mathbf{I}) \odot \mathbf{E}_r) \mathbf{v}'_r}{(\mathbf{1} + \delta \cdot \mathbf{I})(\mathbf{v}'_r{}^2)}, \quad (6)$$

again element-wise, and extract  $\sigma'_r = \|\tilde{\mathbf{u}}'_r\|_2$ ,  $\mathbf{u}'_r = \tilde{\mathbf{u}}'_r / \sigma'_r$ . Canceling  $\mathbf{v}'_r$  between numerator and denominator is not possible: these matrix-vector products mix different components of  $\mathbf{v}'_r$  across summations.

From this initialization (already the global optimum of  $\mathcal{L}_{\text{act}}$  by SVD-LLM(W)), every ALS step provably refines the perturbed objective:

**Proposition 3.1** (Monotone descent of the ALS sweep). *Starting from SVD(W', k) and applying Eqs. 5–6 for  $r = k - 1, \dots, 0$ ,  $\mathcal{L}_{\text{act,infl}}$  is non-increasing at every step (proof in Appendix A.2).*

**Applying the AIR solution to LLM compression.** We project the decomposition back into the native space and absorb the singular values:  $\mathbf{U}_k \leftarrow \mathbf{U}'_k \sqrt{\Sigma'_k}$ ,  $\mathbf{V}_k^\top \leftarrow \sqrt{\Sigma'_k} (\mathbf{V}'_k{}^\top \mathbf{S}^{-1})$ . Compression is realized through  $\mathbf{W} \leftarrow \mathbf{U}_k \mathbf{V}_k^\top$ ; at inference,  $\mathbf{y} = \mathbf{U}_k (\mathbf{V}_k^\top \mathbf{x})$  reduces MACs from  $mn$  to  $k(m + n)$  (full FLOP analysis in Appendix A.4; pseudocode in Algorithm 1).

**Table 2.** SVD-based compression of LLaMA 7B without enhancements: WikiText-2 / C4 perplexity ( $\downarrow$ ) and average reasoning accuracy ( $\uparrow$ , mean over OpenBookQA, ARC-E, WinoGrande, HellaSwag, PIQA, MathQA). Full table including JS-divergence and per-task reasoning in Appendix A.5 (Table 12).

PARAM.	METHOD	PERPLEXITY $\downarrow$		REAS. $\uparrow$
		WikiText2	C4	Avg.
100%	Base Model	5.68	7.34	57%
80%	Vanilla SVD	19438	16115	32.8%
	FWSVD	22026	32048	31.3%
	ASVD	116	105	39.5%
	SVD-LLM(W)	7.87	16.65	48.8%
	<b>AIR</b>	<b>7.51</b> ( $\downarrow$ 4.6%)	<b>14.24</b> ( $\downarrow$ 14.5%)	<b>49.9%</b> ( $\uparrow$ 2.3%)
60%	Vanilla SVD	52839	46630	32.1%
	FWSVD	81838	111860	31.1%
	ASVD	4915	8103	31.3%
	SVD-LLM(W)	13.81	56.33	40.0%
	<b>AIR</b>	<b>11.27</b> ( $\downarrow$ 18.4%)	<b>35.81</b> ( $\downarrow$ 36.4%)	<b>41.6%</b> ( $\uparrow$ 4.0%)
40%	Vanilla SVD	105082	105082	31.8%
	FWSVD	134928	126754	31.5%
	ASVD	143631	134928	31.4%
	SVD-LLM(W)	63.83	345	33.3%
	<b>AIR</b>	<b>42.52</b> ( $\downarrow$ 33.4%)	<b>277</b> ( $\downarrow$ 19.7%)	<b>33.6%</b> ( $\uparrow$ 0.9%)
20%	Vanilla SVD	196320	208981	<b>32.4%</b> ( $\uparrow$ 2.2%)
	FWSVD	268337	208981	31.2%
	ASVD	24959	23447	31.2%
	SVD-LLM(W)	854	8626	31.3%
	<b>AIR</b>	<b>472</b> ( $\downarrow$ 44.7%)	<b>2550</b> ( $\downarrow$ 70.4%)	31.7%

## 4. Experiments

This section evaluates the performance of *AIR* against and in combination with other compression methods. Some SVD-based methods inherently require enhancements such as dynamic rank allocation or gradient-based fine-tuning (Sec. 2). To facilitate fair comparisons, each experiment groups methods with similar requirements: Table 2 considers methods without enhancements; Table 5 filters for methods using post-compression end-to-end fine-tuning through LoRA. Each method receives identical calibration data and is evaluated with the same protocol. **Default experimental configuration.** Unless otherwise specified, our experiments employ: model `jeffwan/llama-7b-hf` at 16-bit half-precision, evaluation on WIKITEXT-2, calibration with 256 WIKITEXT-2 samples of 2048 tokens each, AttnLRP with the Epsilon rule at  $\epsilon = 10^{-6}$ , influence weighting strength  $\delta = 2.0$ , parameter rate 60%, token generation at batch size 64 with sequence length 512. Best results in bold, relative gains over second-best in green.

### 4.1. Performance-efficiency trade-off

**Methods without enhancements.** Our primary efficiency metric is the *parameter count rate*  $k(m+n)/(m \cdot n)$  for a rank- $k$  approximation of  $\mathbf{W} \in \mathbb{R}^{m \times n}$ , applied uniformly across layers (a reliable proxy for FLOP, mem-

**Table 3.** Perplexity on different models, evaluated at 60% parameter rate.

	Base Model	ASVD	SVD-LLM(W)	AIR
Mistral-7B	5.24	18261	87.2	<b>60</b> ( $\downarrow$ 31.2%)
Vicuna-7B	6.78	3827	18.58	<b>16.65</b> ( $\downarrow$ 10.36%)
TinyLLaMA (1.1B)	7.99	9182	39.33	<b>29.68</b> ( $\downarrow$ 24.53%)
LLaMA 2-7B	5.49	3596	16.65	<b>14.47</b> ( $\downarrow$ 13.09%)
LLaMA 3-8B	6.13	19438	230	<b>67.95</b> ( $\downarrow$ 70.46%)
LLaMA 30B	4.11	16.91	7.74	<b>7.16</b> ( $\downarrow$ 7.49%)

**Table 4.** *AIR* vs. structured pruning, LLaMA 7B (13.5GB), perplexity ( $\downarrow$ ).

MEMORY	10GB	9GB	8GB	7GB
BlockPruner	9.88*	12.21*	18.94*	21.68*
SliceGPT	9.49	12.18	17.18	26.12
<b>AIR</b>	<b>8.24</b> ( $\downarrow$ 13.2%)	<b>9.64</b> ( $\downarrow$ 20.9%)	<b>12.37</b> ( $\downarrow$ 28.0%)	<b>18.29</b> ( $\downarrow$ 30.0%)

\* Value taken from (Wang et al., 2025b).

ory, and runtime gains; Table 9). Table 2 compares *AIR* and SVD-based baselines (Wang et al., 2025b; Yuan et al., 2024) on WikiText-2 (Merity et al., 2016)/C4 (Raffel et al., 2020) perplexity and average common-sense reasoning accuracy (Clark et al., 2018; Mihaylov et al., 2018; Zellers et al., 2019; Bisk et al., 2020; Sakaguchi et al., 2021; Amini et al., 2019) (full per-task and JS-divergence (Khanal & Capone, 2024) numbers, including 80%/20% retention rows, in Appendix Table 12). *AIR*’s closed-form ALS adds only  $\sim$ 12 minutes of A100 compute on LLaMA-7B and remains complementary to LoRA (Table 5). *AIR* consistently outperforms SVD-LLM(W), with the gap widening at higher compression, evidence that simultaneous activation- and influence-awareness preserves parameters that purely activation-aware methods cannot distinguish. FWSVD performs worse than vanilla SVD, reflecting the two issues *AIR* addresses: weak (row-wise) signal integration combined with no activation-awareness. **Generalization across models.** Table 3 extends evaluation to Mistral 7B (Jiang et al., 2023), Vicuna-7B (Chiang et al., 2023), TinyLlama (1.1B) (Zhang et al., 2024), LLaMA 2-7B (Touvron et al., 2023b), LLaMA 3-8B (AI@Meta, 2024), and LLaMA 30B (Touvron et al., 2023a): *AIR* achieves the lowest perplexity at 60% rate on every model. **Structured-pruning comparison.** Table 4 compares *AIR* against SliceGPT (Ashkboos et al., 2024) and BlockPruner (Zhong et al., 2025) at matched memory footprints; *AIR*’s advantage grows under tighter memory budgets.

**Methods with enhancements. End-to-end fine-tuning through LoRA.** Table 5 evaluates LoRA fine-tuning (Hu et al., 2021) after compression. *AIR*+LoRA and SVD-LLM(W)+LoRA use Alpaca (Dubois et al., 2024) with 50k samples (sequential U/V optimization following (Wang et al., 2025b)); ACIP (Genzel et al., 2025) bundles LoRA inherently with 1k fine-tuning steps (Merantix Momentum,

**Table 5.** LoRA fine-tuning enhancement, perplexity ( $\downarrow$ ). Further end-to-end ablations (full retraining on the calibration set, with and without LoRA) are reported in Appendix A.8.

PARAM. RATE	80%	60%	40%	20%
ACIP (LoRA inherent)	8.64	12.57	25	3378
SVD-LLM(W) + LoRA	7.31	9.41	16.82	114
AIR + LoRA	<b>7.24</b> ( $\downarrow$ 1%)	<b>9.09</b> ( $\downarrow$ 3%)	<b>14.49</b> ( $\downarrow$ 14%)	<b>46.86</b> ( $\downarrow$ 59%)

**Table 6.** AIR at 60% parameter rate stacked with RTN/GPTQ quantization across 8/4/2-bit widths (LLaMA 7B, base 13.5 GB); file-size rate ( $\downarrow$ ) and WikiText-2 perplexity ( $\downarrow$ ).

BIT WIDTH	8-bit		4-bit		2-bit	
	Mem.	PPL	Mem.	PPL	Mem.	PPL
AIR (60%) + RTN	35.49%	11.44	20.63%	13.17	13.19%	81838
AIR (60%) + GPTQ	35.47%	11.27	20.61%	11.99	13.17%	17154

SIGNAL ( $\mathbf{I}$ )	PSEUDO		BACKWARD, PRINCIPLED		
	ONES	W	W $\times$ GRAD	FISHER	LRP- $\epsilon$
AIR (60%)	13.80	14.02	11.27	11.27	<b>11.27</b>

**Table 7.** Perplexity at different signals (magnitudes only) populating influence matrix  $\mathbf{I}$ .  $\mathbf{W}$  denotes weights.

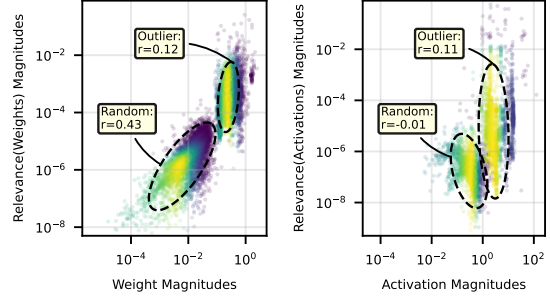
	DIRECTION		SWEEP COUNT		
	BACKW.	FORW.	1	2	10
AIR (60%)	<b>11.27</b>	11.81	<b>11.27</b>	11.44	11.44

**Table 8.** ALS ablations: perplexity across sweep directions and counts.

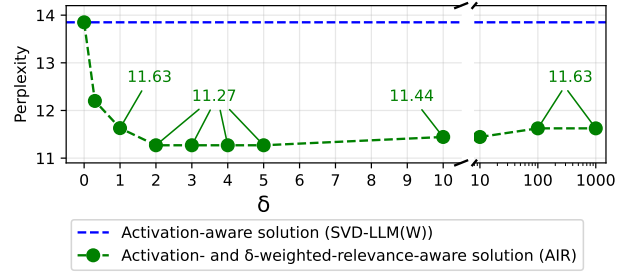
2024). Although AIR and end-to-end fine-tuning both exploit a backward signal, they remain complementary rather than redundant: AIR alone is already competitive with end-to-end ACIP, and AIR+LoRA outperforms every baseline at every rate (gap up to 59% at 20% retention). Appendix A.8 extends this to full retraining of the compressed model on the calibration set; AIR, LoRA, and full retraining each contribute independent gains. **Quantization.** Table 6 combines AIR with RTN and GPTQ (Frantar et al., 2023). At 60% rate with 4-bit GPTQ, AIR retains 11.99 perplexity (vs. 11.27 unquantized) at 20.6% memory, indicating the low-rank structures AIR learns are amenable to lower-precision representation.

## 4.2. ALS dynamics and influence-weighting

The integration mechanism, not the signal choice, drives AIR’s gains: ablating the influence source (Table 7), all three principled backward signals (LRP- $\epsilon$ , Weight  $\times$  Gradient, diagonal Fisher) yield identical perplexity, while uninformative baselines (ones, weight magnitude) fall behind. The



**Figure 2.** Weight (left) and activation (right) magnitudes vs. attributed relevance across all layers of LLaMA 7B; *Random* samples and *Outliers*.

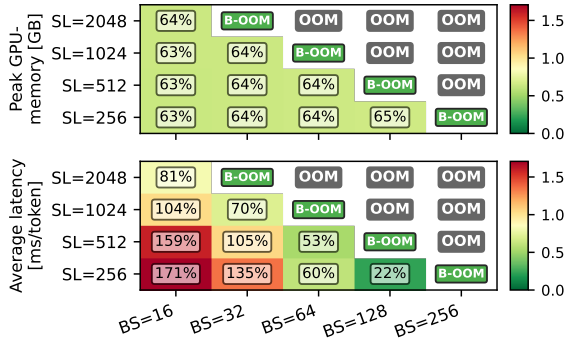


**Figure 3.** Perplexity vs. influence-weighting strength  $\delta$ , yielding the optimal working point.

inset of Figure 1 visualizes the ALS dynamics in a single layer (1228 ranks at 60% parameter rate):  $\mathcal{L}_{\text{act, infl}}$  decreases monotonically from the activation-only initialization to the AIR solution while  $\mathcal{L}_{\text{act}}$  correspondingly grows, evidence that ALS re-allocates reconstruction error from high- to low-influence positions. The sweep completes in 1.5 s per layer on an A100 ( $\sim$ 12 min model-wide, including profiling and influence accumulation); a single backward sweep ( $r = k - 1 \rightarrow 0$ ) is optimal because the larger early updates then land on subordinate rather than dominant ranks, and additional sweeps continue to lower  $\mathcal{L}_{\text{act, infl}}$  but slightly degrade perplexity (Table 8), consistent with mild overfitting to the calibration-set proxy. **Influence-weighting strength.** Figure 3 shows perplexity vs.  $\delta$  in Eq. 4:  $\delta = 0$  recovers SVD-LLM(W), the optimum sits at  $\delta = 2.0$  (our default), and perplexity rises mildly at large  $\delta$  where the objective effectively degenerates to a purely influence-weighted reconstruction  $(\mathbf{1} + \delta\mathbf{I}) \rightarrow \delta\mathbf{I}$ , empirically confirming the role of the all-ones anchor. **Influence vs. native-space magnitudes.** Figure 2 shows correlations between weight (left) and activation (right) magnitudes and their attributed relevance for two subsets per layer of LLaMA 7B: (1) *Random*, 1 ppm of total parameters, and (2) *Outliers*, the 1 ppm highest-magnitude weights/activations. Random-sample weights/activations show only modest correlation; the relationship essentially vanishes for outlier weights and for activations in either subset. Considering the high impact of outliers in LLMs (Xiao et al., 2023), this disconnect indicates that neither weight

TYPE	METRIC	COMPR. / BASE			
STATIC	PARAM.	80%	60%	40%	20%
	FLOPS	80%	61%	41%	21%
	MEM.	81%	62%	42%	23%
RUNTIME	PEAK GPU	75%	64%	52%	42%
	LATENCY	53%	53%	52%	53%

**Table 9.** Parameter rate is a reliable proxy for holistic SVD-based efficiency.

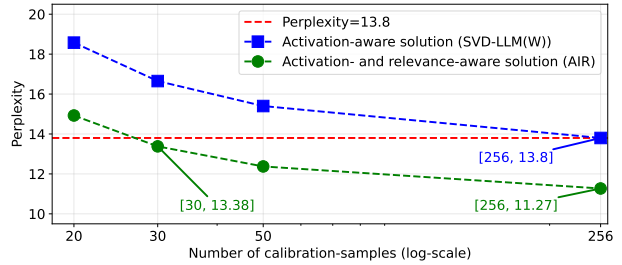


**Figure 4.** Runtime efficiency: Peak GPU-memory and per-token latency rates across batch sizes and sequence lengths; LLaMA 7B at 60% parameter rate. “OOM” marks cells exceeding the 40 GB A100 budget; “B-OOM” marks cells where only the base model OOMs while *AIR*(60%) still fits. Absolute values in Appendix Fig. 7.

nor activation magnitude reliably proxies influence, creating headroom for compression that preserves high-relevance, low-magnitude parameters that magnitude-based methods would discard (spatial view in Appendix A.10).

### 4.3. Efficiency

**Token-generation.** Realizing the efficiency gains promised by SVD-based compression requires a forward-pass implementation that matches the parameter reduction at every level. Static gains do not arise automatically: memory reductions track the parameter count only when truncated ranks are physically pruned rather than zero-masked, and FLOP reductions require sequential propagation  $\mathbf{y} = \mathbf{U}_k(\mathbf{V}_k^\top \mathbf{x})$ , avoiding the preliminary full-weight reconstruction  $\mathbf{W} = \mathbf{U}_k \mathbf{V}_k^\top$ . Runtime gains during autoregressive decoding demand additional care: KV-cache low-rank caching, pre-allocated buffers, and RoPE pre-application. *AIR* integrates all of these into its inference path; at 60% parameter rate on LLaMA-7B, this yields a near-uniform peak GPU-memory ratio of  $\sim 64\%$  across the batch-size  $\times$  sequence-length grid (Figure 4). Per-token latency shows two regimes: under high bandwidth pressure (large BS and/or long SL, near the OOM frontier) *AIR* reduces latency the most, down to 22% at BS=128, SL=256; at small batch sizes and short sequences the two-matmul overhead can exceed the base model’s latency (up to 171% at BS=16, SL=256). At our



**Figure 5.** Data-efficiency: WikiText-2 perplexity vs. number of calibration samples for *AIR* and SVD-LLM(W).

default operating point (BS=64, SL=512), *AIR* reaches 53% of base latency, matching the near-constant latency rate at this fixed reference (Table 9). The additional payoff of more aggressive compression is the matched peak-GPU reduction, which shifts the OOM frontier outward and unlocks the near-OOM regime where the largest latency gains arise; e.g., *AIR*(60%) fits BS=64, SL=1024, where the base model OOMs on a 40 GB A100. Prior SVD-based methods often achieve the static numbers but overclaim runtime gains; Appendix A.4 details our optimizations and audits these claims, including a corrected accounting that turns a reported  $2.65\times$  SVD-LLM speedup into a 27% slowdown once early-EOS termination is accounted for. **Data efficiency.** *AIR* matches SVD-LLM(W)’s 256-sample quality with  $\sim 90\%$  less calibration data, and the gap persists as more samples are added (Figure 5); if gains came purely from better activation estimation, more samples would close the gap.

## 5. Conclusion

*AIR* closes the gap between activation-aware SVD compression and end-to-end retraining in a single closed-form, layer-local sweep. On LLaMA-7B it lowers perplexity over SVD-LLM(W) at all parameter rates (widening under aggressive compression), matches it with  $\sim 90\%$  less calibration data, and converts parameter savings into runtime gains ( $\sim 64\%$  peak memory, 53% latency at 60% retention on a 40 GB A100); gains carry over to Mistral, Vicuna, LLaMA-2/3/30B and to structured-pruning baselines. Two takeaways. (1) *Layer-local and end-to-end optimization are complementary*: LoRA, full retraining (Appendix A.8), and GPTQ quantization each compose without erasing *AIR*’s advantage. (2) *The integration mechanism dominates the choice of signal*: principled backward signals yield equivalent perplexity through our update rule. **Limitations.** Scaling to 70B+ and encoder-decoder/MoE architectures remains open; *AIR*’s uniform per-layer rate leaves dynamic rank allocation, token-level adaptation, and budget-conditional inference as natural extensions. **Code:** <https://github.com/NicodeHarder/air>.

## References

- Achtibat, R., Hatefi, S. M. V., Dreyer, M., Jain, A., Wiegand, T., Lapuschkin, S., and Samek, W. AttnLRP: Attention-aware layer-wise relevance propagation for transformers. 235:135–168, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/achtibat24a.html>.
- AI@Meta. Llama 3 model card. 2024. URL [https://github.com/meta-llama/llama3/blob/main/MODEL\\_CARD.md](https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md).
- Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.
- Ashkboos, S., Croci, M. L., Gennari do Nascimento, M., Hoefler, T., and Hensman, J. SliceGPT: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.
- Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS one*, 10(7):e0130140, 2015.
- Balzano, L., Ding, T., Haeffele, B. D., Kwon, S. M., Qu, Q., Wang, P., Wang, Z., and Yaras, C. An overview of low-rank structures in the training and adaptation of large models, 2025. URL <https://arxiv.org/abs/2503.19859>.
- Becking, D., Dreyer, M., Samek, W., Müller, K., and Lapuschkin, S. *ECQx: Explainability-Driven Quantization for Low-Bit and Sparse DNNs*, pp. 271–296. Springer International Publishing, 2022. ISBN 9783031040832. doi: 10.1007/978-3-031-04083-2\_14. URL [http://dx.doi.org/10.1007/978-3-031-04083-2\\_14](http://dx.doi.org/10.1007/978-3-031-04083-2_14).
- Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.
- Chekalina, V., Moskovskiy, D., Cherniuk, D., Kurkin, M., Kuznetsov, A., and Frolov, E. Generalized fisher-weighted svd: Scalable kronecker-factored fisher approximation for compressing large language models, 2025. URL <https://arxiv.org/abs/2505.17974>.
- Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.
- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- Dubois, Y., Li, X., Taori, R., Zhang, T., Gulrajani, I., Ba, J., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpaca-farm: A simulation framework for methods that learn from human feedback, 2024. URL <https://arxiv.org/abs/2305.14387>.
- Eckart, C. and Young, G. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
- Frantar, E., Kachuee, M., and Alistarh, D. GPTQ: Accurate post-training quantization for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2023.
- Genzel, M., Putzky, P., Zhao, P., Schulze, S., Mollenhauer, M., Seidel, R., Dietzel, S., and Wollmann, T. Choose your model size: Any compression of large language models without re-computation, 2025. URL <https://arxiv.org/abs/2502.01717>.
- Hassibi, B. and Stork, D. G. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 5, pp. 164–171, 1992.
- Hatefi, S. M. V., Dreyer, M., Achtibat, R., Wiegand, T., Samek, W., and Lapuschkin, S. Pruning by explaining revisited: Optimizing attribution methods to prune CNNs and transformers, 2024. URL <https://arxiv.org/abs/2408.12568>.
- Hsu, Y.-C., Hua, T., Chang, S., Lou, Q., Shen, Y., and Jin, H. Language model compression with weighted low-rank factorization, 2022. URL <https://arxiv.org/abs/2207.00112>.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021.
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023. URL <https://arxiv.org/abs/2310.06825>.

- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL <https://arxiv.org/abs/2001.08361>.
- Khanal, B. and Capone, J. M. Evaluating the impact of compression techniques on task-specific performance of large language models, 2024. URL <https://arxiv.org/abs/2409.11233>.
- Kunstner, F., Hennig, P., and Balles, L. Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 2, pp. 598–605, 1989.
- Martens, J. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.
- Merantix Momentum. `acip_llama1_7b`. [https://huggingface.co/MerantixMomentum/acip\\_llama1\\_7b](https://huggingface.co/MerantixMomentum/acip_llama1_7b), 2024. Model repository for the Any Compression (ACIP) LLaMA-7B weights.
- Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.
- Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.
- Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations (ICLR)*, 2017.
- Montavon, G., Binder, A., Lapuschkin, S., Samek, W., and Müller, K.-R. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pp. 193–209, 2019.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.
- Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. In *Communications of the ACM*, 2021.
- Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *International Conference on Machine Learning (ICML)*, pp. 3145–3153. PMLR, 2017.
- Srebro, N. and Jaakkola, T. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pp. 720–727, 2003.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.
- Wang, X., Alam, S., Wan, Z., Shen, H., and Zhang, M. Svd-llm v2: Optimizing singular value truncation for large language model compression, 2025a. URL <https://arxiv.org/abs/2503.12340>.
- Wang, X., Zheng, Y., Wan, Z., and Zhang, M. Svd-llm: Truncation-aware singular value decomposition for large language model compression. 2025b. URL <https://arxiv.org/abs/2403.07378>.
- Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.
- Yeom, S.-K., Seegerer, P., Lapuschkin, S., Binder, A., Wiedemann, S., Müller, K.-R., and Samek, W. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115:107899, July 2021. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.107899. URL <http://dx.doi.org/10.1016/j.patcog.2021.107899>.
- Yuan, Z., Shang, Y., Song, Y., Wu, Q., Yan, Y., and Sun, G. Asvd: Activation-aware singular value decomposition for compressing large language models, 2024. URL <https://arxiv.org/abs/2312.05821>.
- Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.
- Zhang, P., Zeng, G., Wang, T., and Lu, W. Tinyllama: An open-source small language model, 2024. URL <https://arxiv.org/abs/2401.02385>.

Zhong, L., Wan, F., Chen, R., Quan, X., and Li, L. Block-pruner: Fine-grained pruning for large language mod-

els, 2025. URL <https://arxiv.org/abs/2406.10594>.

## A. Appendix

**Organization.** Appendix A.1 develops the functional-awareness framework in full, including the Taylor-expansion hierarchy that organizes gradient-, curvature-, and Fisher-based influence metrics, and the mathematical equivalence between LRP-0 and Weight $\times$ Gradient that supports the signal-agnostic claim of Sec. 3. Appendix A.2 derives the closed-form ALS updates and proves Propositions A.1–A.2. Appendix A.3 provides the extended ablations and relevance-property analyses that the condensed Sec. 4.2 summarizes (influence-signal ablation, ALS sweep direction/count, scatter and spatial heatmaps). Appendix A.4 presents the systems-level analysis of SVD-based inference, including an audit of throughput and KV-cache claims in prior work and the forward-pass optimizations that realize end-to-end gains. Appendix A.8 reports a further end-to-end ablation that stacks *AIR* with full retraining on the calibration set, in addition to and in combination with LoRA fine-tuning, providing the most demanding test of the additivity reading discussed in Sec. 4. Appendix A.9 provides the qualitative compression-failure analysis referenced in the main text, and Appendix A.10 visualizes the spatial structure of influence within transformer weight matrices.

### A.1. Functional Awareness in Compression

Recall our general SVD-based compression through  $\hat{\mathbf{W}}_k = \mathbf{U}_k \Sigma_k \mathbf{V}_k^\top$ , the rank- $k$  approximation of  $\mathbf{W}$ . Vanilla SVD minimizes weight reconstruction error  $\|\mathbf{W} - \hat{\mathbf{W}}_k\|_F$ , treating all weight elements as equally important regardless of their functional role in the network. Achieving *functional awareness*, ensuring that compression preserves the model’s input–output behavior, requires incorporating calibration data  $\mathcal{D}_{\text{cal}}$  to distinguish functionally critical from redundant parameters. Existing approaches can be categorized by whether they derive this awareness from the *forward pass* (activation-based, local) or the *backward pass* (loss- or output-based, global).

Forward-pass methods operate layer-locally: they are computationally cheap and can yield provably optimal solutions to their respective local objectives, but remain unaware of whether the preserved activations actually contribute to the final prediction. Backward-pass methods address this limitation by considering the global prediction loss or model output, but since directly minimizing the global objective amounts to costly retraining, they rely on *influence metrics* as tractable proxies. *AIR* combines both: activation-awareness from the forward pass with influence-awareness from the backward pass.

Throughout,  $\mathbf{W} \in \mathbb{R}^{m \times n}$  denotes a layer weight matrix,  $x_i$  denotes the hidden state (activation) of input neuron  $i$ , pre-activations are  $z_{ij} = x_i w_{ij}$  with  $z_j = \sum_i z_{ij}$ , and  $f(\mathbf{x})$  denotes the model output. We omit layer indexing for notational simplicity.

#### FORWARD-PASS METHODS: LOCAL ACTIVATION-AWARENESS

For brevity in this overview, we use the equivalent scalar Frobenius form for  $\mathcal{L}_{\text{act}}$ ; see Section 3 (Eq. 3) for the element-wise matrix form used throughout the *AIR* derivation, with the two related by  $\sum_{ij} [\mathcal{L}_{\text{act}}]_{ij} = \|\cdot\|_F^2$ .

Forward-pass methods aim to preserve the integrity of the local layer output  $\mathbf{Y} = \mathbf{W}\mathbf{X}$  under compression, minimizing  $\mathcal{L}_{\text{act}} = \|\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}_k \mathbf{X}\|_F^2$ .

**ASVD (Yuan et al., 2024).** ASVD introduces a diagonal scaling matrix  $\mathbf{S}_0 = \text{diag}(s_1, \dots, s_n)$ , where each  $s_j$  reflects the significance of input channel  $j$ , typically set as  $s_j = (|x_j|)^\alpha$  with  $|x_j|$  denoting the mean absolute activation of channel  $j$  and  $\alpha$  as a tunable exponent. ASVD decomposes the scaled weight  $\mathbf{W}\mathbf{S}_0$  via SVD, then absorbs  $\mathbf{S}_0^{-1}$  into the reconstruction:  $\mathbf{Y} \approx (\mathbf{U}_k \Sigma_k \mathbf{V}_k^\top) \mathbf{S}_0^{-1} \mathbf{X}$ . This heuristically reweights the SVD to account for activation magnitudes across channels, but does not provably minimize  $\mathcal{L}_{\text{act}}$ ; the scaling is approximate and the choice of  $\alpha$  requires tuning.

**SVD-LLM (Wang et al., 2025b).** SVD-LLM applies a whitening transformation  $\mathbf{W}' = \mathbf{W}\mathbf{S}$  with  $\mathbf{S} = \text{chol}(\sum_{\mathcal{D}_{\text{cal}}} \mathbf{X}\mathbf{X}^\top)$ , absorbing the full second-order activation structure into the weight. Since SVD of  $\mathbf{W}'$  yields the optimal rank- $k$  approximation in Frobenius norm (Eckart–Young), and  $\mathcal{L}_{\text{act}} = \|\mathbf{W}' - \hat{\mathbf{W}}'_k\|_F^2$  after profiling, this provides the *provably optimal* activation-aware solution, a key advantage over ASVD’s heuristic scaling.

Both methods are efficient (requiring only forward passes over  $\mathcal{D}_{\text{cal}}$ ) but share a key limitation: they optimize for local layer-output fidelity without awareness of whether the preserved activations actually contribute to the model’s final prediction.

## BACKWARD-PASS METHODS: GLOBAL PREDICTION-AWARENESS VIA INFLUENCE METRICS

To incorporate global prediction-awareness, backward-pass methods consider the task loss of the compressed model. Denoting the full set of model parameters by  $\theta$ , the task loss reads:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_{(\mathbf{x}, y) \in \mathcal{D}_{\text{cal}}} \ell(y, f(\theta, \mathbf{x})), \quad (7)$$

where  $\ell$  denotes the per-sample loss (e.g., cross-entropy). The ideal compression minimizes  $\mathcal{L}(\hat{\theta}_k)$ , the task loss evaluated at the compressed parameters  $\hat{\theta}_k$ . Directly minimizing this objective amounts to training, which can be prohibitively expensive for LLMs. Instead, backward-pass methods propagate information from the model output to yield *influence scores* for weight components. The compression task then decomposes into two steps: (1) defining an effective influence score  $i_{ij}$  for each weight  $w_{ij}$ , which quantifies the perturbation effect of modifying it during compression; and (2) integrating this influence to guide compression, e.g., by finding compressed weights  $\hat{w}_{ij}$  that minimize the influence-weighted surrogate  $\sum_{i,j} i_{ij} (w_{ij} - \hat{w}_{ij})^2$ .

**Defining Influence via Taylor Expansion.** The influence metrics discussed below can be understood as successive approximations to the task loss at the compressed weights  $\mathcal{L}(\hat{\theta}_k)$ : the derivatives are computed at the original parameters  $\theta$ , while the expansion is evaluated at  $\hat{\theta}_k$ , so one simultaneously differentiates the task loss *and* approximates the compression objective. The Taylor expansion is used here not to directly optimize  $\hat{\theta}_k$  (which would require inverting the Hessian or iterative descent), but to *probe the loss landscape* around  $\theta$ : the resulting per-element scores characterize how sensitive  $\mathcal{L}$  is to each weight, and these scores then guide a tractable surrogate optimization (see ‘‘Integrating Influence’’ below). The expansion is simplified by considering one layer at a time (ignoring cross-layer dependencies), so that compression of  $\mathbf{W}$  to  $\hat{\mathbf{W}}_k$  is the only perturbation. The Taylor expansion of the loss around  $\mathbf{W}$ , evaluated at  $\hat{\mathbf{W}}_k$ , then gives:

$$\begin{aligned} \mathcal{L}(\hat{\mathbf{W}}_k) \approx & \underbrace{\mathcal{L}(\mathbf{W})}_{\text{0th order}} + \underbrace{\nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W})^\top (\hat{\mathbf{W}}_k - \mathbf{W})}_{\text{1st order}} \\ & + \underbrace{\frac{1}{2} (\hat{\mathbf{W}}_k - \mathbf{W})^\top \mathbf{H}_{\mathbf{W}} (\hat{\mathbf{W}}_k - \mathbf{W})}_{\text{2nd order}} + \dots, \end{aligned} \quad (8)$$

where  $\mathbf{H}_{\mathbf{W}} = \nabla_{\mathbf{W}}^2 \mathcal{L}(\mathbf{W}) \in \mathbb{R}^{p \times p}$  is the Hessian restricted to the  $p = mn$  parameters of this layer. Since the 0th-order term  $\mathcal{L}(\mathbf{W})$  is a constant independent of the compression, it is irrelevant for minimization, and influence metrics are derived from the remaining terms.

**First-Order Influence (gradXweight).** Retaining only the first-order term and restricting to a single-element perturbation ( $\hat{w}_{ij} = 0$ , all others unchanged, so  $\hat{w}_{ij} - w_{ij} = -w_{ij}$ ) gives the estimated loss change  $-w_{ij} \cdot \partial \mathcal{L} / \partial w_{ij}$ . This first-order term captures the *slope* of the loss landscape: it measures how steeply  $\mathcal{L}$  changes when  $w_{ij}$  is perturbed. The signed influence reads:

$$i_{ij}^{\text{gradXweight}} = -w_{ij} \cdot \frac{\partial \mathcal{L}}{\partial w_{ij}} = -z_{ij} \cdot \frac{\partial \mathcal{L}}{\partial z_j}, \quad (9)$$

since  $\partial \mathcal{L} / \partial w_{ij} = (\partial \mathcal{L} / \partial z_j) \cdot x_i$ . This is identical to ‘‘Weight  $\times$  Gradient’’ (Molchanov et al., 2017; Shrikumar et al., 2017); the two names reflect different motivations but yield the same formula. The first-order approach ignores curvature and all higher-order terms, yet this approximation has proved effective empirically and receives further legitimacy through its mathematical equivalence to LRP-0-based influence (see below), an independent framework that arrives at the same quantity. First-order influence also remains the only computationally feasible option among the Taylor-derived metrics; although the gradient is computed at the *original* weights  $\mathbf{W}$ , it yields scores that guide compression toward  $\hat{\mathbf{W}}_k$ . The reformulation in terms of pre-activations  $z_{ij}$  facilitates comparison with LRP below.

**Second-Order Influence (Hessian and Approximations).** Retaining only the second-order term captures the *curvature* of the loss landscape: it measures how the slope itself changes around  $w_{ij}$ , so that even a weight with small gradient can be important if the loss curves sharply in its vicinity. Again restricting to a single element:

$$i_{ij}^{\text{Hessian}} = \frac{1}{2} w_{ij}^2 \cdot \frac{\partial^2 \mathcal{L}}{\partial w_{ij}^2}, \quad (10)$$

where  $\partial^2 \mathcal{L} / \partial w_{ij}^2$  is a diagonal element of  $\mathbf{H}_{\mathbf{W}}$ . Note that  $(-w_{ij})^2 = w_{ij}^2$ , so this term is always non-negative (assuming positive semi-definite curvature) and requires no absolute value. Computing  $\mathbf{H}_{\mathbf{W}}$  requires  $O(p^2)$  storage, infeasible for LLMs, motivating two lines of approximation:

**(a) Direct structural approximation.** Optimal Brain Damage (LeCun et al., 1989) retains only the diagonal of  $\mathbf{H}_{\mathbf{W}}$ ; Optimal Brain Surgeon (Hassibi & Stork, 1992) additionally considers off-diagonal elements via inverse-Hessian row computations.

**(b) Fisher information as Hessian proxy.** FWSVD (Hsu et al., 2022) replaces  $\mathbf{H}_{\mathbf{W}}$  with the empirical Fisher  $\hat{\mathbf{F}} = \frac{1}{|\mathcal{D}|} \sum_d (\nabla_w \mathcal{L}_d) (\nabla_w \mathcal{L}_d)^\top$ , reducing the second derivative to a sum of squared first-order gradients. This substitution captures second-order curvature only under restrictive conditions on the data and parameters that are essentially never met in practice (Martens, 2020; Kunstner et al., 2019). FWSVD further simplifies  $\mathbf{F}$  to a diagonal, yielding per-element influence  $i_{ij}^{\text{Fisher}} = \frac{1}{|\mathcal{D}|} \sum_d (\partial \mathcal{L}_d / \partial w_{ij})^2$ . GFWSVD (Chekalina et al., 2025) recovers some correlation structure via Kronecker factorization.

**Note on terminology: “Hessian” in the LLM compression literature.** The term “Hessian” appears in two distinct senses in this literature, and we adopt only one of them throughout this section. SparseGPT, Wanda, OBC, SlimGPT, and SVD-LLM rely on the *layer-wise reconstruction Hessian*  $\mathbf{X}^\top \mathbf{X}$ , i.e. the Hessian of the local objective  $\|\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}\mathbf{X}\|_F^2$ ; in our taxonomy this is an activation-driven local quantity, and ASVD and SVD-LLM accordingly describe the same object as “activation-aware” rather than as a Hessian (forward-pass column of Table 10). The Taylor-derived “Hessian” of this section is the *loss Hessian*  $\partial^2 \mathcal{L} / \partial \mathbf{w}^2$  targeted by OBD, a distinct curvature object. To our knowledge, the diagonal of the loss Hessian has not been used as an element-wise saliency for LLM compression: diagonal Fisher provides a cheaper proxy, and exact diagonal estimation via Hutchinson / Hessian-vector-product estimators (cf. AdaHessian, Sophia) is substantially more expensive at LLM scale.

**Unified view and approximation chain.** First-order Taylor, Hessian, and Fisher all characterize the *same* global loss  $\mathcal{L}$  (Eq. 7) at increasing derivative orders (slope, then curvature) with the Fisher being the cheapest second-order proxy (approximating curvature via squared gradients). Influence-based compression in general is built on a chain of approximations: (i) considering only one layer at a time, ignoring cross-layer dependencies; (ii) retaining only a single term of an infinite Taylor series; and (iii) for second-order methods, restricting the Hessian to its diagonal or a low-rank surrogate. These are simplifications, yet in practice they provide effective guidance for compression. The first-order Taylor influence also finds independent validation through Layer-wise Relevance Propagation: as we show below, it is mathematically equivalent to LRP-0. Despite this equivalence, LRP is derived from a distinct foundation, not based on loss gradients but on relevance initialized at the model output and propagated backward under layer-wise conservation. This offers a gradient-free perspective on influence, which has also been shown empirically effective for pruning (Yeom et al., 2021; Hatefi et al., 2024).

**LRP-Based Relevance.** LRP (Bach et al., 2015) assigns influence through a different mechanism: rather than expanding  $\mathcal{L}$ , it decomposes a scalar model output  $f_c(\mathbf{x})$ , the logit of the predicted token, via relevance conservation. For the  $\epsilon$ -rule at weight  $w_{ij}$ , with pre-activations  $z_{ij} = x_i w_{ij}$  and  $z_j = \sum_i z_{ij}$ :

$$r_{i \leftarrow j}^{\text{LRP-}\epsilon} = \frac{z_{ij}}{z_j + \epsilon \cdot \text{sign}(z_j)} \cdot R_j, \quad (11)$$

where  $R_j$  is the relevance received from the layer above. This shares the forward factor  $z_{ij}$  with first-order Taylor but differs in the backward signal: Taylor propagates  $\partial \mathcal{L} / \partial z_j$  (loss gradient), while LRP propagates  $R_j / z_j$  (relevance ratio).

For deep ReLU networks with the basic LRP-0 rule ( $\epsilon = 0$ ), piecewise linearity causes the redistribution ratios  $z_{ij} / z_j$  to coincide with the local linear coefficients (Shrikumar et al., 2017; Montavon et al., 2019). Chaining from the output to any weight  $w_{ij}$  yields:

$$r_{i \leftarrow j}^{\text{LRP-0}} = \frac{z_{ij}}{z_j} \cdot R_j = z_{ij} \cdot \frac{\partial f_c}{\partial z_j} = w_{ij} \cdot \frac{\partial f_c}{\partial w_{ij}}, \quad (12)$$

where the magnitude is the same as from Weight  $\times$  Gradient when applied to some logit  $f_c$  (instead of  $\mathcal{L}$ ). This identity holds for any scalar function propagated from the output; initializing LRP from  $\mathcal{L}$  instead of  $f_c$  would recover exactly the first-order Taylor term (Eq. 9), up to sign. The practical difference,  $f_c$  vs.  $\mathcal{L}$ , is thus one of initialization choice: since

**Table 10.** Overview of functional-awareness strategies for SVD-based compression. “Scope” indicates local (layer) vs. global (model) awareness. For backward-pass influence metrics, “Order” refers to the derivative order of  $\mathcal{L}$ . The diagonal of the loss Hessian (OBD row) is computable via Hutchinson / Hessian-vector-product estimators (cf. AdaHessian, Sophia) but is substantially more expensive than diagonal Fisher and has not, to our knowledge, been used as an element-wise saliency for LLM compression. <sup>†</sup>LRP- $\epsilon$  inherits 1st-order Taylor character via the LRP-0 equivalence (Eq. 12), with the  $\epsilon$ -stabilizer adding noise suppression.

Method	Type	Scope	Order	Data	What it captures	PPL at 60% (as I in AIR)
Vanilla SVD	–	Local	–	No	Weight reconstruction error	–
ASVD	Forward	Local	–	Yes	Heuristic activation scaling	–
SVD-LLM	Forward	Local	–	Yes	Optimal activation-aware reconstruction	–
Ones	–	Global	0th	No	Dummy attribution	13.80
Weight magnitude	–	Global	0th	No	Parameter size	14.02
Gradient	Backward	Global	1st	Yes	Loss sensitivity	–
Taylor 1st order (Weight×Gradient)	Backward	Global	1st	Yes	Estimated loss change from removal	11.27
Diagonal Fisher	Backward	Global	Pseudo-2nd	Yes	Diagonal-Hessian curvature proxy	11.27
Hessian (full)	Backward	Global	2nd	Yes	Loss curvature	infeasible
Hessian (OBD, diagonal)	Backward	Global	2nd	Yes	Loss curvature, diagonal only	not evaluated
LRP- $\epsilon$ (AIR default)	Backward	Global	1st <sup>†</sup>	Yes	Functional contribution to $f(\mathbf{x})$	<b>11.27</b>

cross-entropy mixes all logits through the softmax, the two signals differ in general, but align closely when the predicted token dominates the output distribution.

The  $\epsilon$ -rule introduces a separate deviation: the stabilizer prevents the cancellation of  $z_j$  in Eq. (12), producing influence estimates distinct from any gradient-based quantity. For small  $\epsilon$  (e.g.,  $\epsilon = 10^{-6}$  as used in this work), this deviation is negligible whenever  $|z_j| \gg \epsilon$ , and the two criteria yield near-identical influence landscapes.

**Integrating Influence into Compression.** Regardless of how influence is defined, all methods discussed above yield per-element scores that must be integrated into the compression objective. Following established practice (Yeom et al., 2021; Hatefi et al., 2024), influence is represented by its *magnitude only*: the sign of the Taylor term or LRP relevance indicates the *direction* of effect (beneficial vs. detrimental perturbation), but for guiding compression we care only about the *size* of the functional impact, since reconstruction errors can have either sign. AIR does not use influence scores merely to rank or select weights but instead *scales* the approximation error element-wise, mathematically representing the full first-order Taylor product (not just one of its factors). Concretely, influence magnitudes are accumulated over calibration data and collected in the influence matrix  $\mathbf{I}_{\mathbf{W}} \in \mathbb{R}^{m \times n}$ , then normalized to unit mean per layer. This normalization ensures that a weight with average influence receives no modification (multiplicative factor of one), while functionally critical weights are penalized more heavily for reconstruction error. The normalized influence enters the hybrid objective (Eq. 4) by reweighting each squared residual element-wise with  $(1 + \delta \cdot i_{ij})$ , where  $\delta$  controls the influence weighting strength. A detailed derivation of the resulting ALS update rules is provided in Subsection A.2.

#### A.1.1. SUMMARY OF FUNCTIONAL AWARENESS IN SVD-BASED COMPRESSION

Table 10 provides a comparative overview. Forward-pass methods offer provably optimal local solutions (SVD-LLM) but lack prediction-awareness. Loss-based backward-pass methods (Taylor, Hessian, Fisher) form a hierarchy of increasingly sophisticated, and increasingly approximated, characterizations of the global task loss. LRP-based relevance is derived from a distinct foundation: it decomposes the model’s prediction rather than the loss, avoids the Fisher approximation cascade, and provides element-wise influence with built-in noise suppression via the  $\epsilon$ -rule (which becomes more pronounced at larger  $\epsilon$ ). AIR exploits this complementarity by combining activation-awareness (forward-pass,  $\mathbf{X}$ ) with task-awareness (backward-pass influence signal,  $\mathbf{I}_{\mathbf{W}}$ , defaulting to LRP- $\epsilon$ ) in its hybrid objective (Eq. 4). The condensed ablation in Sec. 4 (Table 7) shows that the AIR mechanism is robust to the specific backward signal used to fill  $\mathbf{I}_{\mathbf{W}}$ .

## A.2. ALS Optimization

We provide the complete derivation of the closed-form updates for our influence-weighted ALS optimization. Algorithm 1 summarizes the resulting procedure as pseudocode.

---

### Algorithm 1 AIR

---

**Require:** Weight matrix  $\mathbf{W}$ , profiling matrix  $\mathbf{S}$ , influence matrix  $\mathbf{I}$ , influence weighting strength  $\delta$ , target rank  $k$

**Ensure:** Activation- and influence-aware compression  $\mathbf{U}_k \mathbf{V}_k^\top$

- 1: Apply profiling:  $\mathbf{W}' \leftarrow \mathbf{W}\mathbf{S}$
  - 2: Initialize via SVD:  $\mathbf{U}'_k, \mathbf{\Sigma}'_k, \mathbf{V}'_k \leftarrow \text{SVD}(\mathbf{W}', k)$
  - 3: Compute  $\mathbf{W}'_k \leftarrow \mathbf{U}'_k \mathbf{\Sigma}'_k \mathbf{V}'_k{}^\top$
  - 4: **for**  $r = k - 1$  **down to** 0 **do**
  - 5:   Compute residual:  $\mathbf{E}_r \leftarrow \mathbf{W}' - \mathbf{W}'_k + \sigma'_r \mathbf{u}'_r \mathbf{v}'_r{}^\top$
  - 6:   Update  $\mathbf{v}'_r$  via Eq. (5)
  - 7:   Update  $\mathbf{u}'_r, \sigma'_r$  via Eq. (6)
  - 8:   Update  $\mathbf{W}'_k$  incrementally
  - 9: **end for**
  - 10: Reverse profiling:  $\mathbf{V}'_k{}^\top \leftarrow \mathbf{V}'_k{}^\top \mathbf{S}^{-1}$
  - 11: Absorb singular values:  $\mathbf{U}_k \leftarrow \mathbf{U}'_k \sqrt{\mathbf{\Sigma}'_k}, \mathbf{V}_k{}^\top \leftarrow \sqrt{\mathbf{\Sigma}'_k} \mathbf{V}'_k{}^\top$
- 

**Problem formulation** Expanding Eq. (4) element-wise, we aim to minimize:

$$\mathcal{L}(\mathbf{U}'_k, \mathbf{\Sigma}'_k, \mathbf{V}'_k) = \sum_{i,j} (1 + \delta \cdot i_{ij}) (\mathbf{W}'_{ij} - [\mathbf{U}'_k \mathbf{\Sigma}'_k \mathbf{V}'_k{}^\top]_{ij})^2 \quad (13)$$

where  $\delta \geq 0$  controls the influence weighting strength: when  $\delta = 0$ , we recover standard (unweighted) low-rank approximation; when  $\delta > 0$ , elements with higher influence  $i_{ij}$  receive greater weight in the reconstruction loss.

The reconstruction can be written as a sum of rank-1 components:

$$\mathbf{W}'_k = \mathbf{U}'_k \mathbf{\Sigma}'_k \mathbf{V}'_k{}^\top = \sum_{r=0}^{k-1} \sigma'_r \mathbf{u}'_r \mathbf{v}'_r{}^\top \quad (14)$$

where  $\sigma'_r$  denotes the  $r$ -th singular value, and  $\mathbf{u}'_r, \mathbf{v}'_r$  are the corresponding left and right singular vectors.

**Coordinate descent strategy** ALS optimizes one rank- $r$  component ( $\sigma'_r, \mathbf{u}'_r, \mathbf{v}'_r$ ) at a time while holding all others fixed. We define the residual when excluding component  $r$ :

$$\mathbf{E}^{(r)} = \mathbf{W}' - \sum_{j \neq r} \sigma'_j \mathbf{u}'_j \mathbf{v}'_j{}^\top = \mathbf{W}' - \mathbf{W}'_k + \sigma'_r \mathbf{u}'_r \mathbf{v}'_r{}^\top \quad (15)$$

The optimization for component  $r$  becomes:

$$\min_{\sigma'_r, \mathbf{u}'_r, \mathbf{v}'_r} \sum_{i,j} (1 + \delta \cdot i_{ij}) (\mathbf{E}_{ij}^{(r)} - \sigma'_r u'_{r,i} v'_{r,j})^2 \quad (16)$$

**Derivation of  $\mathbf{v}'_r$  Update** Fixing  $\mathbf{u}'_r$  and  $\sigma'_r$ , we minimize with respect to  $\mathbf{v}'_r$ . The objective for element  $v'_{r,j}$  is:

$$\mathcal{L}_{v'_{r,j}} = \sum_i (1 + \delta \cdot i_{ij}) (\mathbf{E}_{ij}^{(r)} - \sigma'_r u'_{r,i} v'_{r,j})^2 \quad (17)$$

Taking the derivative and setting to zero:

$$\frac{\partial \mathcal{L}_{v'_{r,j}}}{\partial v'_{r,j}} = -2 \sum_i (1 + \delta \cdot i_{ij}) (\mathbf{E}_{ij}^{(r)} - \sigma'_r u'_{r,i} v'_{r,j}) \sigma'_r u'_{r,i} = 0 \quad (18)$$

Solving for  $v'_{r,j}$ :

$$\sigma'_r \sum_i (1 + \delta i_{ij}) \mathbf{E}_{ij}^{(r)} u'_{r,i} = \sigma'^2_r v'_{r,j} \sum_i (1 + \delta i_{ij}) (u'_{r,i})^2 \quad (19)$$

$$v'_{r,j} = \frac{\sum_i (1 + \delta i_{ij}) \mathbf{E}_{ij}^{(r)} u'_{r,i}}{\sigma'_r \sum_i (1 + \delta i_{ij}) (u'_{r,i})^2} \quad (20)$$

In matrix notation, this yields the closed-form update:

$$\mathbf{v}'_r = \left( \frac{\mathbf{u}'_r{}^\top ((\mathbf{1} + \delta \cdot \mathbf{I}) \odot \mathbf{E}^{(r)})}{\sigma'_r \cdot (\mathbf{u}'_r{}^2)^\top (\mathbf{1} + \delta \cdot \mathbf{I})} \right)^\top \quad (21)$$

where the division is element-wise and  $\mathbf{u}'_r{}^2$  denotes element-wise squaring. The transpose ensures  $\mathbf{v}'_r$  is a column vector: the numerator computes a row vector where the  $j$ -th element is the influence-weighted inner product of  $\mathbf{u}'_r$  with the  $j$ -th column of  $\mathbf{E}^{(r)}$ , and the denominator is a row vector of normalization factors for each column. This corresponds to Equation (5) in the main text.

**Derivation of  $\mathbf{u}'_r$  Update** Fixing  $\mathbf{v}'_r$ , we minimize with respect to  $\mathbf{u}'_r$  and  $\sigma'_r$  jointly. The objective for the  $i$ -th row is:

$$\mathcal{L}_i = \sum_j (1 + \delta \cdot i_{ij}) (\mathbf{E}_{ij}^{(r)} - \sigma'_r u'_{r,i} v'_{r,j})^2 \quad (22)$$

Let  $\tilde{u}'_{r,i} = \sigma'_r u'_{r,i}$  denote the unnormalized left singular vector element. Taking the derivative with respect to  $\tilde{u}'_{r,i}$  and setting to zero:

$$\frac{\partial \mathcal{L}_i}{\partial \tilde{u}'_{r,i}} = -2 \sum_j (1 + \delta \cdot i_{ij}) (\mathbf{E}_{ij}^{(r)} - \tilde{u}'_{r,i} v'_{r,j}) v'_{r,j} = 0 \quad (23)$$

Solving for  $\tilde{u}'_{r,i}$ :

$$\sum_j (1 + \delta \cdot i_{ij}) \mathbf{E}_{ij}^{(r)} v'_{r,j} = \sum_j (1 + \delta \cdot i_{ij}) \tilde{u}'_{r,i} (v'_{r,j})^2 \quad (24)$$

$$\tilde{u}'_{r,i} = \frac{\sum_j (1 + \delta \cdot i_{ij}) \mathbf{E}_{ij}^{(r)} v'_{r,j}}{\sum_j (1 + \delta \cdot i_{ij}) (v'_{r,j})^2} \quad (25)$$

In matrix notation:

$$\tilde{\mathbf{u}}'_r = \frac{((\mathbf{1} + \delta \cdot \mathbf{I}) \odot \mathbf{E}^{(r)}) \mathbf{v}'_r}{(\mathbf{1} + \delta \cdot \mathbf{I}) (\mathbf{v}'_r{}^2)} \quad (26)$$

where the division is element-wise and  $\mathbf{v}'_r{}^2$  denotes element-wise squaring. Note that simplifying by canceling  $\mathbf{v}'_r$  between numerator and denominator is not possible: since these are matrix-vector products, each resulting element  $i$  involves distinct summations  $\sum_j (\cdot) v'_{r,j}$  and  $\sum_j (\cdot) (v'_{r,j})^2$ , mixing different components of  $\mathbf{v}'_r$ . This corresponds to Equation (6) in the main text.

Finally, we extract the singular value and normalized left singular vector:

$$\sigma'_r = \|\tilde{\mathbf{u}}'_r\|_2, \quad \mathbf{u}'_r = \frac{\tilde{\mathbf{u}}'_r}{\sigma'_r} \quad (27)$$

This normalization ensures  $\|\mathbf{u}'_r\|_2 = 1$ , maintaining the standard SVD convention.

**Convergence Properties and Justification for a Single ALS Sweep** We begin with two formal statements that characterize the convergence behavior of the AIR solution. The first establishes the optimality of the activation-aware initialization in the special case  $\delta = 0$ ; the second establishes monotone descent of the influence-weighted objective along the ALS sweep.

**Proposition A.1** (Optimality of the activation-aware initialization). *At  $\delta = 0$ , the influence-weighted objective  $\mathcal{L}_{act,infl}$  (Eq. 4) reduces to the activation-aware objective  $\mathcal{L}_{act}$  (Eq. 3). The initialization  $(\mathbf{U}'_k, \mathbf{\Sigma}'_k, \mathbf{V}'_k) \leftarrow \text{SVD}(\mathbf{W}', k)$  is the global minimizer of  $\mathcal{L}_{act}$  over all rank- $k$  factorizations.*

*Proof.* At  $\delta = 0$ ,  $(\mathbf{1} + \delta \cdot \mathbf{I}) = \mathbf{1}$ , so  $\mathcal{L}_{act,infl} = \mathcal{L}_{act} \odot \mathbf{1} = \mathcal{L}_{act}$ . The scalar objective minimized is  $\sum_{ij} [\mathcal{L}_{act}]_{ij} = \|\mathbf{W}' - \mathbf{U}'_k \mathbf{\Sigma}'_k \mathbf{V}'_k{}^\top\|_F^2$ . The Eckart–Young theorem (Eckart & Young, 1936) guarantees that the truncated SVD provides the global minimum of this Frobenius-norm reconstruction error over all rank- $k$  matrices. Since the profiled weight  $\mathbf{W}'$  is fixed and the optimization is over its rank- $k$  factorizations,  $\text{SVD}(\mathbf{W}', k)$  is the global minimizer.  $\square$

For  $\delta > 0$ , the joint optimization over  $(\mathbf{U}'_k, \mathbf{\Sigma}'_k, \mathbf{V}'_k)$  is non-convex due to the product structure  $\mathbf{U}'_k \mathbf{\Sigma}'_k \mathbf{V}'_k{}^\top$ . Standard global-optimality arguments do not apply, but a coordinate-wise descent guarantee does:

**Proposition A.2** (Monotone descent of the ALS sweep). *Let  $(\mathbf{U}'_k{}^{(0)}, \mathbf{\Sigma}'_k{}^{(0)}, \mathbf{V}'_k{}^{(0)}) = \text{SVD}(\mathbf{W}', k)$ . Define the sequence of iterates obtained by sweeping  $r = k - 1, k - 2, \dots, 0$  and applying Eq. 5 followed by Eq. 6 at each step. Then the influence-weighted objective  $\mathcal{L}_{act,infl}$  is non-increasing along this sequence: for every step  $r$ ,*

$$\mathcal{L}_{act,infl}((\mathbf{U}'_k, \mathbf{\Sigma}'_k, \mathbf{V}'_k)_{\text{after step } r}) \leq \mathcal{L}_{act,infl}((\mathbf{U}'_k, \mathbf{\Sigma}'_k, \mathbf{V}'_k)_{\text{before step } r}),$$

*with equality if and only if the corresponding component is already at the coordinate-wise optimum.*

*Proof.* At step  $r$ , the update for  $\mathbf{v}'_r$  holds  $\mathbf{U}'_k, \mathbf{\Sigma}'_k$ , and all  $\mathbf{v}'_j$  with  $j \neq r$  fixed. Under this restriction,  $\mathcal{L}_{act,infl}$  is a strictly convex quadratic in  $\mathbf{v}'_r$ : the Hessian is the diagonal matrix with  $j$ -th entry  $\sigma_r'^2 \sum_i (1 + \delta \cdot i_{ij})(u'_{r,i})^2 > 0$  (positive whenever  $\sigma_r' \neq 0$  and the influence weights are non-negative, both of which hold by construction). Setting the gradient to zero yields the unique minimizer given by Eq. 5; the derivation is carried out element-wise above. Hence the new iterate has objective value no greater than the previous one. The same argument applies to the joint update of  $(\mathbf{u}'_r, \sigma_r')$  via Eq. 6: with all other components fixed, the objective is a strictly convex quadratic in the unnormalized vector  $\tilde{\mathbf{u}}'_r = \sigma_r' \mathbf{u}'_r$ , and Eq. 6 provides its unique minimizer. The subsequent normalization  $\sigma_r' = \|\tilde{\mathbf{u}}'_r\|_2$ ,  $\mathbf{u}'_r = \tilde{\mathbf{u}}'_r / \sigma_r'$  is a re-parameterization of the same point in the rank-1 component  $\sigma_r' \mathbf{u}'_r \mathbf{v}'_r{}^\top$  and leaves the objective value unchanged. Composing the two updates per step over the  $k$  steps of the sweep yields the stated monotone descent.  $\square$

**Remark A.3** (Scope of the guarantee). Proposition A.2 establishes monotone descent of the objective along the sweep, not convergence of the iterate sequence to a global minimum: the underlying problem is non-convex in the joint factorization  $\mathbf{U}'_k \mathbf{\Sigma}'_k \mathbf{V}'_k{}^\top$ . This is the standard guarantee for block-coordinate descent on non-convex objectives and matches the convergence statements available for related closed-form one-pass methods such as GPTQ (Frantar et al., 2023). Combined with Proposition A.1, however, the result has practical force: AIR starts at the global minimum of the unperturbed objective  $\mathcal{L}_{act}$ , and every subsequent step provably does not worsen the perturbed objective  $\mathcal{L}_{act,infl}$ .

As shown in Table 8 (Appendix A.3), the backward iteration order (from  $r = k - 1$  to  $r = 0$ ) matters: it protects the dominant singular components (corresponding to larger singular values, i.e., smaller  $r$  indices) from distortion, as the largest perturbations occur in the first iterations of the ALS when individual rank updates can be largest.

A natural question is whether multiple ALS sweeps would further improve compression quality. Our experiments show that while  $\mathcal{L}_{act,infl}$  can continue to decrease marginally in subsequent sweeps, downstream perplexity does not improve. We identify three reasons why a single sweep suffices:

(1) *Strong initialization from the unperturbed optimum.* By Proposition A.1, AIR initializes from the global minimum of the activation-aware objective  $\mathcal{L}_{act}$  (Eq. 3), and by Proposition A.2 every subsequent ALS step does not increase the perturbed objective  $\mathcal{L}_{act,infl}$  (Eq. 4). Since  $\mathcal{L}_{act,infl}$  is a bounded multiplicative perturbation of  $\mathcal{L}_{act}$ , with  $\mathbf{I}$  normalized to unit mean, a single coordinate descent sweep from this optimum captures the dominant correction. Subsequent sweeps would only adjust for second-order interactions between rank updates, which are small because the rank components are approximately orthogonal at initialization (being exact singular vectors of  $\mathbf{W}'$ ). Empirically, these small additional adjustments slightly degrade downstream perplexity (Table 8) even as  $\mathcal{L}_{act,infl}$  continues to decrease, consistent with mild overfitting to the local

proxy on the calibration set:  $\mathcal{L}_{\text{act,infl}}$  is computed from a finite calibration sample and serves as a surrogate for the end-task loss, so further refinement of the proxy beyond its dominant correction need not transfer to held-out perplexity.

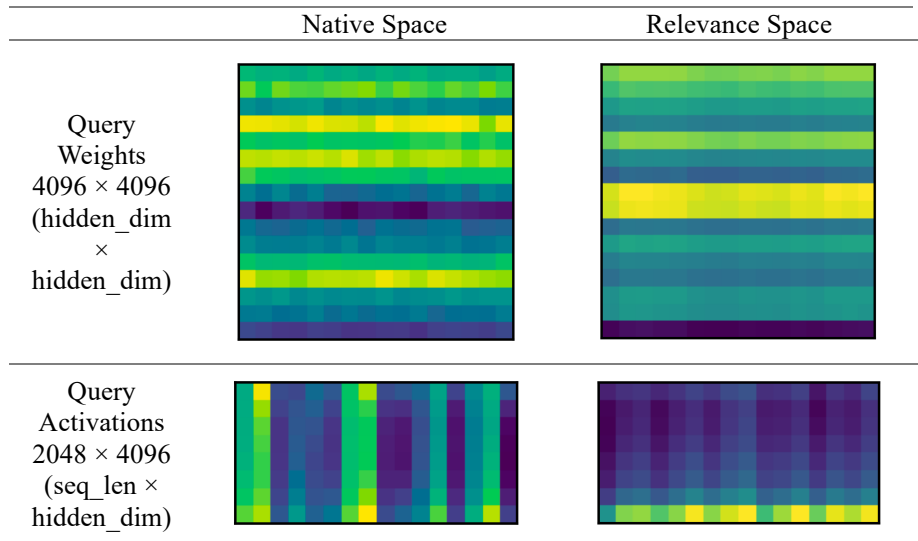
(2) *Structural analogy to GPTQ*. Our single-sweep strategy parallels GPTQ (Frantar et al., 2023), which processes weight columns in a single left-to-right pass: at each column, it applies a closed-form optimal update (derived from inverse Hessian information), propagates the residual to remaining columns, and never revisits. Despite this single-pass design, and without formal convergence guarantees beyond empirical validation, GPTQ achieves near-optimal quantization up to 175B parameters. The parallel is direct: both methods decompose a matrix optimization into sequential subproblems over individual components (columns for GPTQ, ranks for AIR), solve each exactly via closed-form expressions, and propagate the residual in a single sweep. Our setting is arguably more favorable, as AIR starts from a provably optimal initialization rather than from the original pretrained weights.

(3) *Empirical confirmation*. We verified that additional ALS sweeps do not improve downstream perplexity, suggesting that the single sweep already captures all compression-relevant information from the influence signal.

### A.3. ALS dynamics: extended ablations and relevance properties

This appendix contains the relevance-property analysis that the main-body Sec. 4.2 summarizes (the ALS ablation tables, Tables 7 and 8, are reported in the main body).

**Spatial relevance patterns.** Figure 6 provides a spatial view of the same observation made in main-text Figure 2 (the disconnect between native-space magnitudes and attributed relevance), at the matrix level. The corresponding broader maps appear in Appendix A.10: regions of high relevance often do not coincide with regions of high activation or weight magnitude.



**Figure 6.** Spatial heatmaps in native vs. relevance space, shown for the attention query weight and activation matrices in layer block 10 of LLaMA 7B. Elements are accumulated over  $256 \times 256$  blocks.

### A.4. Efficiency for token-generation through implementation

SVD-based compression achieves proportional reductions in parameter count, FLOPs, and memory footprint (main-text Table 9). However, for autoregressive token generation, the two deployment-critical metrics, *peak GPU memory* and *per-token latency*, do not immediately benefit in proportion. Instead, they require specific forward-pass optimizations and further depend on the operating regime (batch size, sequence length). This section analyzes these challenges, examines efficiency claims in prior work, and presents the optimizations that make SVD-based compression effective for token generation.

A.4.1. WHY STANDARD SVD COMPRESSION DOES NOT REDUCE PEAK MEMORY OR LATENCY

**KV cache: no memory reduction without optimization.** During autoregressive generation, the KV cache, not the weight matrices, dominates GPU memory for long sequences. Standard SVD-compressed models compute the two-stage projection  $\mathbf{U}_k(\mathbf{V}_k^\top \mathbf{x})$  and expand the result back to full dimension before entering the KV cache, so the cache stores keys and values at the *original*  $n_{kv} \cdot d_h$  dimension per token. The weight compression thus yields zero cache reduction, and peak GPU memory remains dominated by the uncompressed KV cache.

**Two-matmul overhead: no latency reduction.** Replacing a single matrix multiplication  $\mathbf{y} = \mathbf{W}\mathbf{x}$  with  $\mathbf{y} = \mathbf{U}_k(\mathbf{V}_k^\top \mathbf{x})$  reduces the MAC count from  $mn$  to  $k(m+n)$ , a factor  $\frac{k(m+n)}{mn}$  that mirrors the parameter ratio. However, on modern GPUs, autoregressive decoding processes inputs of shape  $(b, 1, d)$ , placing inference in a *memory-bandwidth-bound* regime where time is dominated by reading weights from HBM, not arithmetic. In this regime, two smaller weight reads are not necessarily faster than one contiguous read due to kernel launch overhead, reduced cache locality from intermediate materialization, and lower SM occupancy when individual GEMMs are too small to saturate the GPU. The theoretical FLOP reduction therefore does not translate proportionally, or at all, into wall-clock speedup.

This is consistent across the literature, though rarely stated explicitly. ACIP (Genzel et al., 2025) is a notable exception: LLaMA-7B compressed to 40% achieves only 1595 tokens/s vs. 2448 tokens/s uncompressed, a 35% *slowdown* despite 58% fewer FLOPs. None of the SVD-based methods surveyed (ASVD (Yuan et al., 2024), SVD-LLM (Wang et al., 2025b), SVD-LLM V2 (Wang et al., 2025a)) provide fused CUDA kernels to bridge this gap.

A.4.2. SCRUTINY OF EFFICIENCY CLAIMS IN PRIOR WORK

**Inflated throughput metrics.** SVD-LLM’s `eff_eval()` computes throughput as  $b \cdot l_{\max}/t_{\text{wall}}$ , counting the maximum generation length regardless of early EOS termination. Compressed models produce EOS earlier due to degraded quality, so the metric counts *phantom tokens* never generated, systematically inflating speedups for more compressed models. On TinyLlama 1.1B at 60% parameter rate, this metric reports a  $2.65\times$  speedup (322 vs. 855 tok/s). However, the base model generates 726/1024 tokens on average while the compressed model generates only 200/1024. Corrected throughput: 229 tok/s (base) vs. 167 tok/s (compressed); the compressed model is 27% *slower*, consistent with the memory-bandwidth-bound analysis above.

**Unsubstantiated KV cache compression.** Both ASVD (Yuan et al., 2024) and SVD-LLM (Wang et al., 2025b) discuss KV cache compression in their papers, but their implementations differ from the claims. ASVD’s `compress_kv_cache` flag restricts rank allocation search to K and V projections, but the `SVDLinear` module still computes  $\mathbf{U}_k(\mathbf{V}_k^\top \mathbf{x})$  at full output dimension with no modified attention module, yielding no cache reduction. SVD-LLM provides a dedicated module (`svd_llama_kvcache.py`) caching low-rank intermediates for *both* keys and values, but their strategy re-expands both to full dimension before attention at every step, trading cache memory for per-step computation overhead (acknowledged in their limitations section). The published code contains errors (incorrect RoPE function signatures) and multiple `TODO` annotations, indicating a non-functional prototype.

A.4.3. OUR PROPOSED OPTIMIZATIONS

**Common practices.** All SVD-based methods propagate hidden states sequentially through the decomposed factors rather than reconstructing the full weight matrix, i.e.  $\mathbf{y} = \mathbf{U}_k(\Sigma_k(\mathbf{V}_k^\top \mathbf{x}))$  instead of the naive  $\mathbf{y} = (\mathbf{U}_k \Sigma_k \mathbf{V}_k^\top) \mathbf{x}$ . This avoids materializing the full  $m \times n$  matrix and reduces the MAC count from  $mn$  to  $k(m+n)$ , a factor matching the parameter rate. Additionally, the diagonal  $\Sigma_k$  is absorbed via  $\mathbf{A} = \mathbf{U}_k \sqrt{\Sigma_k}$ ,  $\mathbf{B} = \sqrt{\Sigma_k} \mathbf{V}_k^\top$ , eliminating the middle matmul and reducing the forward pass to two matrix multiplications:  $\mathbf{y} = \mathbf{A}(\mathbf{B}\mathbf{x})$ . The balanced absorption distributes magnitudes evenly, which also benefits downstream quantization (Yuan et al., 2024; Wang et al., 2025b).

**Beyond common practices: KV cache and buffer optimizations.** We introduce three optimizations that target the KV cache bottleneck. All three route caching through an internal path that bypasses the standard HuggingFace KV cache, with a shared prerequisite structure.

(O1) *RoPE pre-application (prerequisite).* Rotary Position Embeddings are applied to keys immediately during cache writes, storing  $\mathbf{k}_t = \mathbf{R}(t) \mathbf{U}_k^{(k)}(\mathbf{V}_k^{(k)T} \mathbf{x}_t)$ . Without this, the standard code path attempts to re-apply RoPE to the full accumulated cache with position IDs corresponding only to the current token, producing incorrect key states. O1 is a prerequisite for O2

**Table 11.** Ablation of forward-pass optimizations on peak GPU memory and per-token latency. Each O1-prefixed row adds optimizations on top of the internal-cache path; the first row is the default HuggingFace cache path. The base model at BS=64, SL=512 (29.34 GB peak memory, 1540  $\mu$ s/token) serves as reference (100%).

PARAM. RATE	PEAK GPU MEM. RATE		LATENCY RATE	
	100%	60%	100%	60%
Compressed, w/o opt.	100.3%	84.0%	125.5%	117.3%
O1	100.4%	84.0%	124.5%	118.9%
O1+O3	100.4%	84.0%	50.1%	50.5%
O1+O2	86.3%	64.3%	91.2%	76.6%
O1+O2+O3	86.4%	64.2%	46.6%	44.2%

and O3.

(O2) *Fused low-rank value caching.* Values undergo no position-dependent transformation, so the intermediate  $\mathbf{v}_t = \mathbf{V}_k^{(v)T} \mathbf{x}_t \in \mathbb{R}^{r_v}$  (with  $r_v \ll n_{kv} \cdot d_h$ ) can be cached directly. The expansion through  $\mathbf{U}_k^{(v)}$  is deferred and fused into the attention computation:

$$\text{Attn}(\mathbf{A}, \mathbf{v}_{1:s}) = \underbrace{\left( \sum_{t=1}^s a_t \cdot \mathbf{v}_t \right)}_{\in \mathbb{R}^{r_v}} \cdot \mathbf{U}_k^{(v)T}, \tag{28}$$

where  $\mathbf{A} \in \mathbb{R}^{n_h \times 1 \times s}$  are the attention weights. The weighted sum operates in  $r_v$ -dimensional space; only the final result is projected to full dimension. This reduces the per-token value cache from  $n_{kv} \cdot d_h$  to  $r_v$  elements.

Keys cannot benefit from analogous caching: RoPE is a position-dependent rotation  $\mathbf{R}(n)$  in the full  $d_h$ -dimensional space, so low-rank key caching would require re-expanding and re-applying RoPE to all  $s$  cached keys at every step ( $\mathcal{O}(s \cdot r_k \cdot d_h)$  per layer). Absorbing RoPE into the projection is equally infeasible, as  $\mathbf{R}(n) \cdot \mathbf{U}_k^{(k)}$  produces a position-specific ( $d_h \times r_k$ )-matrix requiring more storage than full-dimension keys.

	STANDARD	OPTIMIZED
Keys	$n_{kv} \cdot d_h$	$n_{kv} \cdot d_h$
Values	$n_{kv} \cdot d_h$	$r_v$
Total	$2 \cdot n_{kv} \cdot d_h$	$n_{kv} \cdot d_h + r_v$

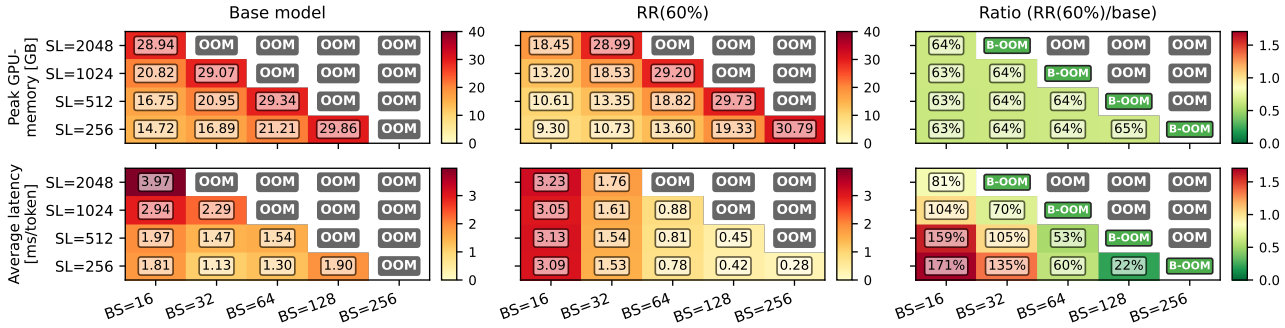
(O3) *Pre-allocated cache buffers.* Cache tensors are allocated once at the maximum sequence length and filled via in-place writes, avoiding per-token dynamic memory allocations. The optimized path also precomputes  $\mathbf{U}_k^{(v)}$  reshaped per attention head, avoiding repeated reshape and repeat operations at each decode step.

#### A.4.4. ABLATION OF FORWARD-PASS OPTIMIZATIONS

Table 11 reports the impact of each optimization on peak GPU memory and per-token latency during autoregressive generation (batch size 64, sequence length 512, LLaMA 7B at 100% and 60% parameter rate on a single A100 GPU), as rates relative to the base model. The first row is the compressed model on the default HuggingFace cache path; rows below add O1 (RoPE pre-application) as prerequisite to the alternative internal-cache path.

O1 alone (switching from the default HuggingFace cache path to the internal cache path with RoPE pre-application) does not by itself improve latency, and at 60% parameter rate slightly worsens it (latency rate 117.3%  $\rightarrow$  118.9%), because the internal caching path lacks the optimized memory access patterns of HuggingFace’s DynamicCache while providing no memory benefit at full dimension. O2 provides the dominant memory reduction (from 84.0% to 64.3% at 60% param rate, comparing O1 to O1+O2), confirming value cache compression as the primary driver. O3 provides the dominant latency improvement (from 118.9% to 50.5% at 60% param rate, comparing O1 to O1+O3), confirming that eliminating per-token dynamic allocations is the main driver of latency. The full combination O1+O2+O3 achieves the best overall trade-off, demonstrating that memory and latency optimizations are complementary.

**Implications.** The absence of inference speedup under standard implementations does not diminish SVD-based compression: reductions in parameter count, memory footprint, and peak GPU allocation (Table 9) matter for memory-constrained



**Figure 7.** Extended runtime efficiency analysis (full version of Figure 4 in the main body): peak GPU-memory in GB (top row) and average per-token latency in ms/token (bottom row), reported in absolute units for the uncompressed base model (left column) and *AIR*(60%) (middle column), alongside the cell-wise ratio *AIR*(60%)/base (right column, reproduced from the main body). “OOM” marks cells that exceed the 40 GB A100 memory budget; “B-OOM” marks cells where only the base model runs out of memory while *AIR*(60%) still fits.

deployment, larger batch sizes, or combination with quantization. However, translating FLOP reductions into latency improvements requires systems-level co-design, as shown by our O1–O3 optimizations. In contrast to the unsubstantiated KV cache claims of prior work (§A.4.2), our approach (i) caches values in low-rank form and *avoids re-expansion* through fused attention, (ii) maintains keys at full dimension with RoPE correctly applied, and (iii) implements optimizations (weight precomputation, pre-allocated buffers) that make the strategy deployment-ready. These considerations apply to all SVD-based methods and are complementary to approximation quality, the focus of *AIR*.

A.4.5. END-TO-END RESULTS: PEAK GPU MEMORY AND PER-TOKEN LATENCY

We now report end-to-end peak GPU memory and per-token latency for the full O1+O2+O3 *AIR* pipeline against the uncompressed base model.

Figure 7 (the full version of the main-body Figure 4, with absolute values for both the base model and *AIR*(60%)) presents peak GPU memory and average per-token latency for a scenario where a user prompts LLaMA 7B (compressed to 60% parameter rate) with a query of 16 tokens and receives a response between 256 and 2048 tokens. The axes *Tokens* and batch size *BS* describe the data structure of the final response; since this structure builds up token by token during autoregressive generation, it also determines the model’s cumulative input context at each decoding step and thus the peak memory and average per-token latency reported in the figure.

Peak GPU memory determines *feasibility*: given a fixed GPU budget, it dictates which batch-size and sequence-length combinations can be served at all. Both the base model and *AIR* scale their memory usage with batch size and sequence length at a similar rate, but *AIR* operates at a lower level: the cell-wise ratio remains ~64% throughout. This uniform offset means the base model hits the OOM boundary one diagonal earlier in the batch-size × sequence-length grid, while *AIR* can progress to the next larger configuration. For instance, at BS=64 with 2048 tokens, or BS=128 with 512 tokens, the base model runs out of memory while *AIR*(60%) completes.

The more deployment-relevant metric, however, is *latency* (or, equivalently, throughput). In a real deployment, the reasoning is sequential: (1) determine the available GPU memory budget (here, 40 GB on an A100), (2) estimate the expected response sequence length required to serve each user, and (3) given these constraints, choose the batch size that minimizes per-token latency, equivalently, maximizes throughput, serving as many users as possible at the required response lengths. Peak GPU memory thus acts as a precondition that determines the feasible region in the batch-size × sequence-length grid; latency then determines the optimal operating point within that region.

The cell-wise latency ratios (third column of Figure 7) show two regimes. At large batch sizes and short sequences, *AIR* achieves speedups (down to 22% at BS=128, SL=256), as the reduced memory footprint alleviates bandwidth pressure and allows higher GPU utilization. At small batch sizes and short sequences, the two-matmul overhead of SVD-based compression dominates, and latency exceeds the base model’s (up to 171% at BS=16, SL=256).

A more deployment-relevant perspective emerges when we follow the practical strategy outlined above. Consider 256-token responses on a 40 GB A100: for each model, we select the batch size that yields the lowest latency among those that

fit in memory. The base model achieves its optimum at BS=64 with 0.81 ms/tok; its latency *increases* again at BS=128 (1.23 ms/tok) due to memory pressure, whereas *AIR*'s latency continues to decrease monotonically with batch size. *AIR* fits BS=256 and reaches 0.29 ms/tok, yielding a latency rate of  $0.29/0.81 \approx 35\%$ . This represents a different comparison from the cell-wise best of 22% reported above: not a matched-configuration ratio, but the ratio between each model's respective best operating point given the same hardware constraints.

### A.5. Full perplexity, JS-divergence, and reasoning comparison

Table 12 expands the condensed main-text Table 2 to all four parameter rates (80%, 60%, 40%, 20%), reports JS-divergence (Khanal & Capone, 2024) alongside perplexity, and breaks down reasoning accuracy across all six tasks (OpenBookQA, ARC-E, WinoGrande, HellaSwag, PIQA, MathQA).

**Table 12.** SVD-based compression of LLaMA 7B without enhancements: perplexity ( $\downarrow$ ), JS-divergence vs. base ( $\uparrow$ , 0% = identical, negative = divergence), and reasoning tasks ( $\uparrow$ ) at varying parameter rates. Full version of main-text Table 2.

PARAM.	METHOD	PERPLEXITY $\downarrow$		JS-DIVERGENCE $\uparrow$		REASONING TASKS $\uparrow$						
RATE		WikiText2 $\downarrow$	C4 $\downarrow$	WikiText2 $\uparrow$	C4 $\uparrow$	OpenB. $\uparrow$	ArcE. $\uparrow$	WinoG. $\uparrow$	HellaS. $\uparrow$	PIQA $\uparrow$	MathQA $\uparrow$	Average $\uparrow$
100%	Base Model	5.68	7.34	0%	0%	34%	75%	70%	57%	79%	27%	57%
80%	Vanilla SVD	19438	16115	-95%	-94%	16.2%	27.4%	51.2%	26.3%	54.1%	21.3%	32.8%
	FWSVD	22026	32048	-96%	-96%	15.4%	26.7%	49.3%	25.5%	53.3%	17.8%	31.3%
	ASVD	116	105	-66%	-62%	21.6%	41.3%	57.2%	32.6%	63.1%	20.9%	39.5%
	SVD-LLM(W)	7.87	16.65	-11%	-23%	26.6%	64%	65.7%	43.3%	69.9%	23.2%	48.8%
	AIR	<b>7.51</b> ( $\downarrow 4.6\%$ )	<b>14.24</b> ( $\downarrow 14.5\%$ )	<b>-8%</b> ( $\uparrow 27.3\%$ )	<b>-19%</b> ( $\uparrow 17.4\%$ )	26.6%	66.3%	66.1%	45.3%	71.1%	24.1%	<b>49.9%</b> ( $\uparrow 2.3\%$ )
60%	Vanilla SVD	52839	46630	-98%	-97%	15.4%	26%	52.7%	25.6%	52.6%	20.5%	32.1%
	FWSVD	81838	111860	-98%	-98%	16.6%	24.8%	49.2%	25.8%	52.8%	17.3%	31.1%
	ASVD	4915	8103	-92%	-93%	13.4%	26.6%	49.4%	26.2%	53.6%	18.6%	31.3%
	SVD-LLM(W)	13.81	56.33	-25%	-48%	20.6%	47.3%	55.4%	32.5%	61.2%	22.7%	40.0%
	AIR	<b>11.27</b> ( $\downarrow 18.4\%$ )	<b>35.81</b> ( $\downarrow 36.4\%$ )	<b>-20%</b> ( $\uparrow 20.0\%$ )	<b>-40%</b> ( $\uparrow 16.7\%$ )	19.2%	51.0%	59%	34.7%	63.7%	22.2%	<b>41.6%</b> ( $\uparrow 4.0\%$ )
40%	Vanilla SVD	105082	105082	-98%	-97%	16.2%	26.4%	50.6%	25.5%	51.8%	20.3%	31.8%
	FWSVD	134928	126754	-98%	-98%	16.4%	26.7%	49.2%	25.6%	52.9%	18.2%	31.5%
	ASVD	143631	134928	-98%	-98%	14.6%	25.4%	49.2%	25.6%	52.2%	21.1%	31.4%
	SVD-LLM(W)	63.83	345	-54%	-72%	14.8%	30.1%	50.8%	27.3%	55%	21.5%	33.3%
	AIR	<b>42.52</b> ( $\downarrow 33.4\%$ )	<b>277</b> ( $\downarrow 19.7\%$ )	<b>-48%</b> ( $\uparrow 11.1\%$ )	<b>-69%</b> ( $\uparrow 4.2\%$ )	13.4%	32.2%	51.5%	27.5%	55.4%	21.7%	<b>33.6%</b> ( $\uparrow 0.9\%$ )
20%	Vanilla SVD	196320	208981	-98%	-98%	16%	26.6%	50.2%	25.6%	54.1%	22%	<b>32.4%</b> ( $\uparrow 2.2\%$ )
	FWSVD	268337	208981	-98%	-98%	15.2%	25%	49.2%	25.5%	53.4%	19.1%	31.2%
	ASVD	24959	23447	-96%	-95%	11.4%	25.3%	51.6%	25.7%	52.9%	20.1%	31.2%
	SVD-LLM(W)	854	8626	-77%	-84%	15.2%	25.8%	48.7%	26%	52%	20.2%	31.3%
	AIR	<b>472</b> ( $\downarrow 44.7\%$ )	<b>2550</b> ( $\downarrow 70.4\%$ )	<b>-68%</b> ( $\uparrow 11.7\%$ )	<b>-83%</b> ( $\uparrow 1.2\%$ )	15.2%	25.8%	49.7%	26.3%	52.9%	20.3%	31.7%

### A.6. C4 as calibration data

To verify that *AIR*'s gains generalize beyond WIKITEXT-2 as the calibration source, we re-run the 60% parameter-rate setting with C4 as the calibration set. *AIR* reaches 19.47 perplexity vs. 34.17 for SVD-LLM(W), a 43% improvement, confirming that the gains are not specific to the calibration distribution. Other parameter rates were not re-run for this calibration variant.

### A.7. Reasoning after LoRA Fine-Tuning

The main-text comparison of LoRA fine-tuning (Sec. 4, Table 5) reports perplexity. To verify that perplexity gains translate into downstream task performance, Table 13 reports average zero-shot accuracy across the same reasoning benchmarks used in Table 2 (OpenBookQA, ARC-Easy, WinoGrande, HellaSwag, PIQA, MathQA). The accuracy ranking mirrors the perplexity ranking: *AIR* + LoRA matches or outperforms SVD-LLM(W) + LoRA at every compression rate, with the gap widening at higher compression ( $\uparrow 5.3\%$  at 20% parameter rate). The advantage is largest where capacity is most constrained, the regime that matters most for deployment.

### A.8. Full retraining on the calibration set: a complementary end-to-end ablation

The main-text Table 5 compares *AIR* and the prior state of the art under the most common end-to-end enhancement, LoRA fine-tuning on external data. This appendix probes a second end-to-end enhancement, *full retraining of the compressed model on the calibration set itself*, and reports the cross-product with LoRA fine-tuning. The motivation is twofold. First, full retraining is rarely considered as a deployable post-compression step; we argue below why it is, in fact, deployable. Second,

**Table 13.** Enhanced SVD-compression through LoRA fine-tuning for LLaMA 7B, measured by average zero-shot accuracy ( $\uparrow$ ). Baseline average accuracy: 57%.

PARAM. RATE	80%	60%	40%	20%
SVD-LLM(W) + LoRA	<b>53.5</b> ( $\uparrow 0.4\%$ )	47.6	40.5	32.3
AIR + LoRA	53.3	<b>48.0</b> ( $\uparrow 0.8\%$ )	<b>40.8</b> ( $\uparrow 0.7\%$ )	<b>34.0</b> ( $\uparrow 5.3\%$ )

**Table 14.** Enhanced SVD-compression through retraining on given calibration data and/or fine-tuning on other data, measured by perplexity ( $\downarrow$ ).

	Default	LoRA (Alpaca)	Retrain (WikiText-2)	Retrain(WikiText-2) + LoRA(Alpaca)
SVD-LLM(W)(40%)	63.8	16.82	13.59	13.22
AIR (40%)	<b>42.5</b> ( $\downarrow 33.4\%$ )	<b>14.5</b> ( $\downarrow 13.8\%$ )	<b>12.77</b> ( $\downarrow 6.0\%$ )	<b>12.25</b> ( $\downarrow 7.3\%$ )
SVD-LLM(W)(20%)	854	114	98.9	94.76
AIR (20%)	<b>472</b> ( $\downarrow 44.7\%$ )	<b>46.9</b> ( $\downarrow 58.9\%$ )	<b>47.4</b> ( $\downarrow 52.1\%$ )	<b>38.35</b> ( $\downarrow 59.5\%$ )

and more importantly for the central claim of the paper (the additivity reading of the design space), full retraining is the most exhaustive end-to-end use of the backward signal available to us, so the question of whether AIR’s local backward-signal proxy still contributes once full retraining has been applied is a more demanding test than the LoRA comparison alone can provide.

**Why full retraining qualifies as a deployable enhancement.** A natural question is whether, given that AIR already leverages the backward pass (and thus model-output information), one should simply perform *full post-training* on the calibration data directly, foregoing the influence-metric approximation altogether. Existing work rarely explores this option, presumably assuming it to be prohibitively expensive for LLMs and potentially harmful due to catastrophic forgetting. We argue, however, that both concerns are overstated in the SVD-compression setting: a 7B-parameter model converges within 5–10 epochs on 256 calibration samples (WikiText-2) in about 12 minutes on an A100 GPU, and catastrophic forgetting is less of a concern after SVD truncation, which itself already disrupts the original weight structure.

We also reconsider a common categorization of post-compression methods by the binary criterion of *whether a parameter update is applied or not*. A more useful distinction is *how much additional data and compute are required, and whether the contribution is complementary to other enhancements*. Under this lens, full retraining on the same small calibration set uses no more data than the compression step itself.

**Compositionality with AIR and LoRA.** Table 14 reports the cross-product of AIR (vs. SVD-LLM) with the four enhancement combinations {Default, LoRA on Alpaca, Retrain on WikiText-2, Retrain + LoRA} at 40% and 20% parameter rates. At the rates we have measured, each of the three mechanisms (the local influence proxy through AIR, full retraining on the calibration set, LoRA fine-tuning on Alpaca) appears to provide an independent gain on top of the others rather than substituting for them, and the three-way combination is the lowest-perplexity column for both AIR and SVD-LLM. We treat this as a working hypothesis to be sharpened by filling the remaining cells (80% and 60%) and extending the comparison to additional model families; if it holds, it suggests that the two appearances of the backward signal in compression, as a per-element local influence score and as a parameter-update direction, are complementary contributions rather than redundant ones.

**A.9. Sentence-completion after LoRA fine-tuning: A qualitative analysis of AIR+LoRA vs. SVD-LLM(W)+LoRA**

Since LLaMA 7B is not fine-tuned as a chatbot, we decided to test for sentence completion rather than replying as an assistant to a user’s question. Note that sentence completion is inherently open-ended; the model predicts plausible continuations rather than answering specific questions. As such, we consider topic shifts to be minor issues rather than indicators of model degradation, since the training data (books, articles, etc.) often contains such transitions that might be coherent in their original context. The outputs reported below are produced by the LoRA-fine-tuned variants of both methods (SVD-LLM(W)+LoRA and AIR+LoRA, matching the setting of Table 5); for brevity we refer to them simply as SVD-LLM(W) and AIR throughout the discussion.

**Activation- and Influence-Aware Ranks (AIR)**

Both prompts show a consistent pattern. At 80% parameter rate, both SVD-LLM(W) and AIR produce fluent, coherent English text with correct primary facts (e.g., Paris as capital), though SVD-LLM(W) shows factual errors (Eiffel Tower height, tallest structure claim) and a spelling mistake. At 60% parameter rate, performance remains comparable with both methods producing interpretable output, though SVD-LLM(W) shows factual inaccuracies (department count) and awkward phrasing. At 40% parameter rate, both methods show degradation: SVD-LLM(W) exhibits terminology errors (“Romantic” instead of Romance) and factual underestimations, while AIR produces multiple factual errors (most widely spoken language, UN status) alongside typographical issues (“Romanoic”). Both methods still generate coherent English sentences at this compression level.

At 20% parameter rate, the methods diverge. SVD-LLM(W) produces empty responses with no tokens generated. AIR continues to generate grammatically structured English sentences. AIR’s content quality degrades with fabricated information (non-existent International Assistance Program), neologisms (“differentities”), and nonsensical continuations, but the model retains its language generation capability and syntactic structure. AIR thus avoids catastrophic failure under high compression, though the quality of its outputs at this level is substantially degraded. At 20% parameters, AIR still correctly identifies “Paris” as the capital of France, while SVD-LLM(W) produces no output.

Combined comparison: SVD-LLM(W)+LoRA vs. AIR+LoRA			
Color legend: <span style="background-color: #f0f0f0;">Factual error</span> <span style="background-color: #fff2cc;">Grammar/spelling issues</span> <span style="background-color: #f4cccc;">Complete collapse</span>			
PARAM. RATE	SVD-LLM(W)+LoRA	AIR+LoRA	COMMENT
<i>Input-Sample: “The capital of France is”</i>			
80%	Paris. The Eiffel Tower is a famous landmark in Paris, and it was built by <span style="background-color: #fff2cc;">Gustave Eiffel</span> in 1889. It stands at 324 meters tall and is <span style="background-color: #fff2cc;">the tallest structure in Europe</span> .	Paris. The capital city of the United States is Washington, D.C. The capitol building in Washington is called the Capitol Building.	Both correct on Paris. SVD-LLM: spelling error (Eifel→Eiffel); Eiffel Tower not tallest structure in Europe. AIR: topic shift to U.S. capitals; content itself is accurate.
60%	Paris. The French government is a democratic republic, with the President as head of state and the Prime Minister as the head minister. France has <span style="background-color: #fff2cc;">56 departments</span> .	Paris. The French Revolution was a political upheaval that occurred in 1789 and lasted for several years...	Both correct on Paris. SVD-LLM: awkward phrasing (“head minister” instead of head of government), wrong department count (actually 101 including overseas). AIR: topic shift to French Revolution; content is historically accurate.
40%	Paris. The French language is a <span style="background-color: #fff2cc;">Romantic</span> language, and it has been used in many countries... with an estimated <span style="background-color: #fff2cc;">10 million speakers</span> in the United...	Paris. The French language is <span style="background-color: #fff2cc;">the most widely spoken language in Europe</span> , with an estimated 120 million speakers worldwide. It is also the <span style="background-color: #fff2cc;">official language of... the UN</span> . French is a <span style="background-color: #fff2cc;">Romanoic</span> language.	Both identify Paris. SVD-LLM: “Romantic” should be Romance; “10 million speakers” is implausible (too high if intended as US speakers, ~1–2M; far too low if global, ~280M). AIR: “Romanoic” typo; false claim (English/Russian more widely spoken in Europe); 120M too low; French is one of 6 UN languages but not “the” official language.
20%	<span style="background-color: #f4cccc;">(empty, no tokens generated)</span>	Paris, which is the city of Paris. <span style="background-color: #fff2cc;">In 2015, the International Assistance Program (International Assistive System) was launched by the United States and Canada to assist the rescue of injured soldiers in the Middle East.</span>	SVD-LLM produces nothing. AIR identifies Paris correctly but with tautological phrasing and completely fabricated content about a non-existent program.
<i>Input-Sample: “The universe is”</i>			
80%	a vast and complex system, and understanding it requires the development of sophisticated mathematical models. The goal of this course is to introduce students to the basic concepts of mathematics that are used in modeling physical phenomena. We will learn about...	a vast and complex system, and the study of it requires a great deal of knowledge. The more we know about our surroundings, the better we can understand them and make use of them. This is why science is so important. It...	Both coherent. SVD-LLM: shifts to academic course description. AIR: smooth philosophical continuation.
60%	a complex system of interconnected processes, and the study of these processes is known as physics. Physics is often divided into several subfields: particle physics, astrophysics, cosmology, <span style="background-color: #fff2cc;">electromagnetic physics (electronics)</span> ...	a complex system of interconnected processes, and the study of these processes is known as physics. The field of physics encompasses many subfields that focus on different aspects of physical phenomena. Some of the most well-known subdis...	Very similar responses. SVD-LLM: “electromagnetic physics (electronics)” is non-standard terminology (should be electromagnetism; electronics is engineering). AIR: coherent, standard phrasing.

## Activation- and Influence-Aware Ranks (AIR)

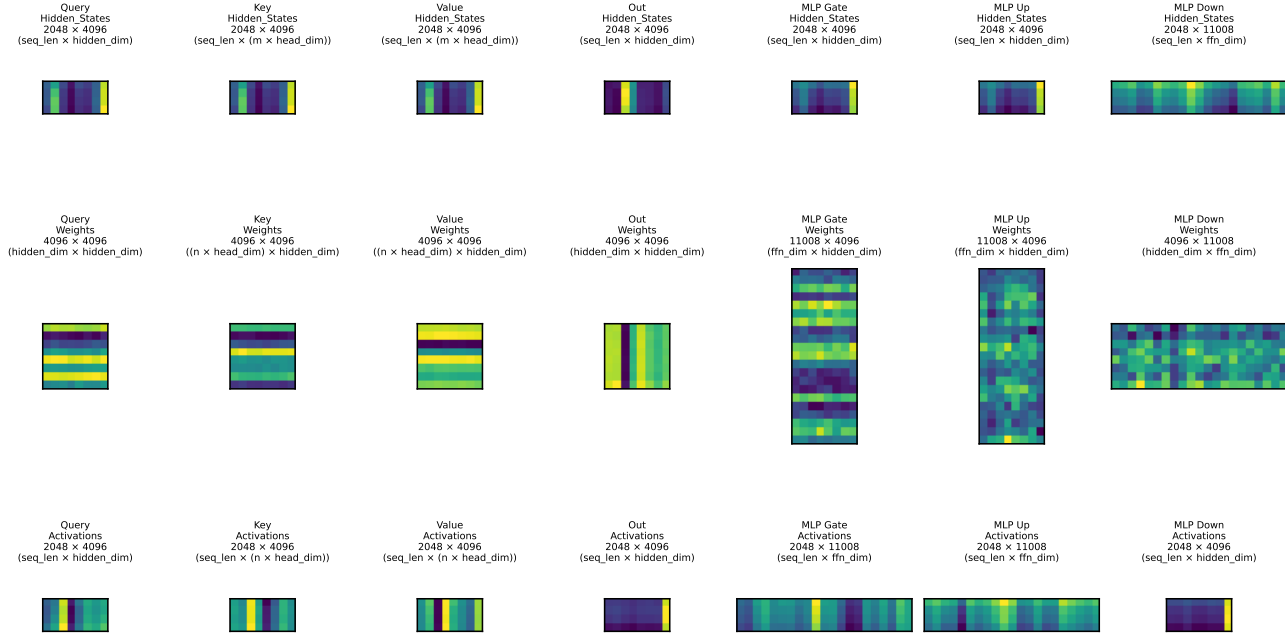
Combined comparison: SVD-LLM(W)+LoRA vs. AIR+LoRA			
Color legend: Factual error Grammar/spelling issues Complete collapse			
PARAM. RATE	SVD-LLM(W)+LoRA	AIR+LoRA	COMMENT
40%	so vast and complex. But there are some things that we know about, like gravity, light, and time. These basic laws of physics play a central role in our understanding of the universe. The fundamental laws: 1. Newton...	a vast and complex system of interconnected stars, planets, galaxies, and other objects. The Sun is the most visible star in our solar system, but it is only one among billions of stars that exist in the galaxy. The...	Both coherent and on-topic. SVD-LLM: minor grammatical issue ("so vast" lacks explicit subject); otherwise reasonable. AIR: factually accurate and well-structured.
20%	(empty, no tokens generated)	a complex object, and it can be viewed as an array of different objects. Here are some examples of the various ways to see the differentities of objects: a, a, b, c, d, or d. These are all...	SVD-LLM: empty output. AIR: neologism "differentities" (made-up word); nonsensical enumeration; but maintains grammatical structure and English-like output.

### A.10. Spatial patterns in weight and relevance distributions

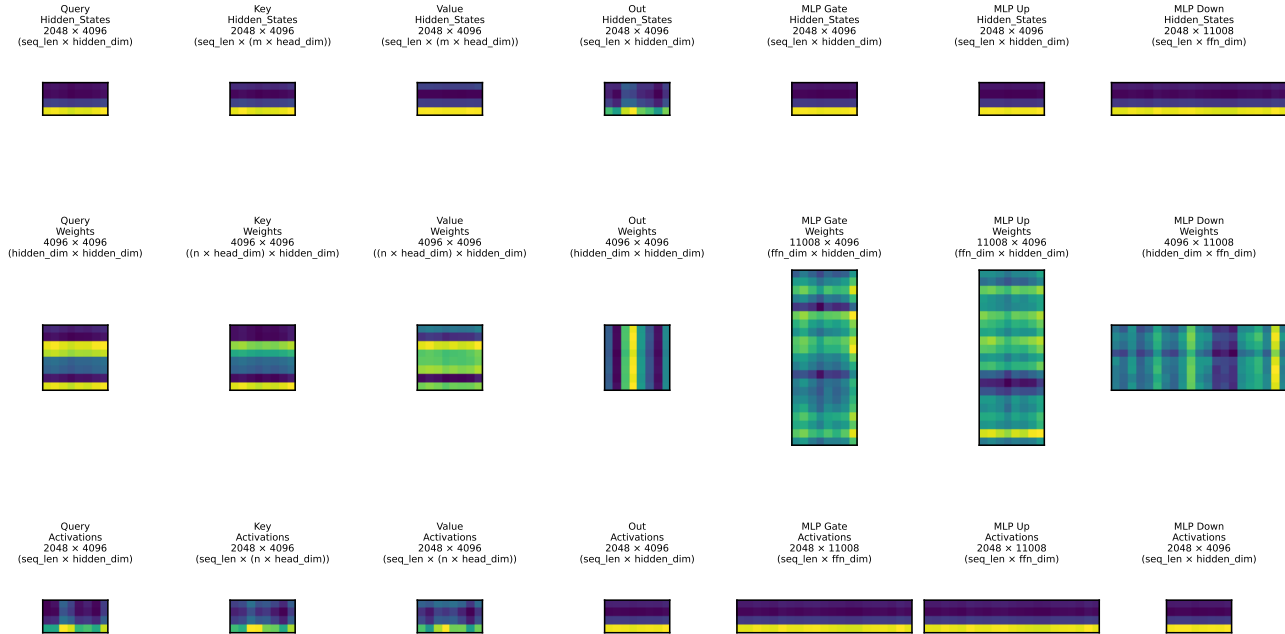
Figure 8 shows spatial patterns in the distribution of functional influence for the LRP- $\epsilon$  default instantiation of **I** (the figure is computed from this specific signal). While weight magnitudes exhibit relatively uniform distributions across matrix dimensions, influence magnitudes show concentrated regions of high influence with structural patterns. This spatial concentration is most evident in MLP layers, where influence forms distinct bands and clusters rather than being uniformly distributed. These patterns indicate that functional influence is not randomly distributed but follows systematic spatial organization, providing a visual explanation for why AIR’s influence-weighted penalty redistributes approximation error to less important regions of the weight matrix.

## Activation- and Influence-Aware Ranks (AIR)

### Native Space



### Relevance Space



**Figure 8.** Spatial heatmaps in layer block 10 of LLaMA 7B shown for weight magnitudes (upper figure) and relevance magnitudes (lower figure). Each sub-figure shows hidden states, weights, and activations (rows) across attention layers (Q,K,V,O) and MLP layers (Gate,Up,Down) (columns). In a simplified manner, activations  $A$  emerge based on hidden-states  $X$  and weights  $W$  through  $A = X \cdot W^T$ . Due to the large matrix dimensions (up to  $11008 \times 4096$ ), we aggregate values over  $512 \times 512$  blocks, with yellow indicating high accumulations and purple indicating low accumulations.