Activation Residual Hessian Quantization for Low-Bit Large Language Model Inference

Yifeng Wang[1], Zhun Sun[1,2], Hakaze Cho[1,2], Linlin Wang[3,4], Keisuke Sakaguchi[1]
[1]Tohoku University, [2]RIKEN, [3]East China Normal University, [4]City University of Hong Kong
wang.yifeng.q6@dc.tohoku.ac.jp, {sun.zhun.a5, keisuke.sakaguchi}@tohoku.ac.jp

Abstract

Low-bit activation-weight quantization is crucial for efficient large language model inference but remains fragile because activation quantization errors propagate across Transformer layers. We propose *Activation Residual Hessian Quantization* (ARHQ), a post-training residual-aware low-rank correction framework that directly targets this propagation. ARHQ measures the residuals produced by the deployed activation quantizer, builds an input-side residual Hessian, and identifies the weight directions that most amplify these residuals. This yields a closed-form truncated-SVD solution: residual-sensitive components are placed in a high-precision low-rank branch, while the remaining weights stay in the low-bit path. ARHQ requires no backpropagation, supports LoRA-style deployment, and complements activation smoothing. Across LLaMA-family models and Qwen3-4B-Thinking, ARHQ improves W4A4 reconstruction, reduces WikiText2 perplexity over matched-rank SVDQuant, preserves downstream reasoning and coding accuracy, and achieves up to 3.21× prefill speedup over BF16 (3.11× over AWQ W4A16) with over 60% memory saving.

1 Introduction

Large language model (LLM) inference is increasingly limited by memory capacity, memory bandwidth, and serving latency (Sheng et al., 2023; Kwon et al., 2023; Lin et al., 2024). Low-bit post-training quantization is therefore central to efficient deployment (Dettmers et al., 2022; Frantar et al., 2022; Xiao et al., 2023). While weight-only Post-training quantization (PTQ) has achieved strong results at 3–4 bits (Frantar et al., 2022; Lin et al., 2024; Dettmers et al., 2023b), aggressive activation-weight quantization remains substantially more fragile (Xiao et al., 2023; Shao et al., 2024; Zhao et al., 2024; Gai and Li, 2024).

Beyond activation outliers, a central challenge in activation-weight quantization is that activations are quantized online at every layer. The deployed quantizer therefore introduces input-dependent residual errors into the hidden states. These residuals are then propagated through subsequent linear transformations, where some directions can be amplified much more strongly than others. As a result, downstream degradation is not determined only by the size of the activation error itself, but also by how strongly the layer weights amplify the residual directions left by the quantizer. We refer to this failure mode as *activation residual amplification*. Such residual errors can propagate and accumulate across Transformer blocks (Dettmers et al., 2022; Xiao et al., 2023; Arai and Ichikawa, 2025), especially under aggressive activation-weight quantization, where prior studies report substantial accuracy risks on language modeling and reasoning evaluations (Li et al., 2024b; Zhao et al., 2024; Liu et al., 2025a). Existing PTQ methods address this issue only indirectly: weight-sensitive methods estimate weight error, activation-aware methods protect salient channels, and smoothing or rotation methods reduce activation outliers (Frantar et al., 2022; Lin et al., 2024; Xiao et al., 2023; Ashkboos et al., 2024; Yuan et al., 2023; Li et al., 2024a). Low-rank compensation methods such as SVDQuant (Li et al., 2024a) and ResQ (Saxena et al., 2025) share a similar deployment structure, a low-bit dense branch plus a high-precision low-rank branch, but select the high-precision subspace from weight energy, activation energy, or magnitude outliers. In contrast, ARHQ targets the residual geometry induced by the deployed quantizer: the error structure that remains after the chosen quantizer, scaling rule, and calibration distribution have already been applied.

We propose *Activation Residual Hessian Quantization* (ARHQ), a post-training residual-aware low-rank correction method. ARHQ measures the activation quantization residuals produced by the de-

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 2613–2627
November 6-10, 2026 ©2026 Association for Computational Linguistics

ployed quantizer on calibration data and constructs a layer-local residual Hessian that captures how strongly different weight directions propagate these residuals. The resulting objective admits a closed-form truncated-SVD solution in a residual-scaled weight space. ARHQ assigns the most residual-sensitive directions to a high-precision low-rank branch while keeping the remaining weights in the low-bit path. This design requires no backpropagation, supports LoRA-style (Hu et al., 2022) deployment, and adapts to the actual quantizer, scaling rule, and calibration distribution used in the deployed low-bit pipeline.

Across LLaMA-family models and Qwen3-4B-Thinking, ARHQ improves W4A4 layer reconstruction over matched-rank SVDQuant, reduces perplexity, and better preserves zero-shot, reasoning, and coding performance, while remaining substantially faster than BF16 and W4A16 inference in compute-bound prefill. Our contributions are as follows:

- We identify activation residual amplification as a distinct failure mode of low-bit activation-weight quantization, separating it from weight error, activation magnitude, and outlier preservation.
- We propose ARHQ, a residual-aware low-rank weight-splitting method that uses the measured activation quantization residual Hessian to select high-precision correction directions.
- We show that ARHQ is complementary to activation smoothing, improves reconstruction quality and downstream accuracy over matched-rank SVDQuant, and offers practical W4A4 serving trade-offs.

2 Related Work

Post-training quantization for LLMs Weight-only PTQ methods have made substantial progress by identifying sensitive weights or directions that should be protected under low-bit compression. GPTQ uses approximate second-order information to quantize Transformer layers without full retraining (Frantar et al., 2022), while AWQ protects salient weights identified through activation statistics (Lin et al., 2024). GPTAQ further accounts for inputs perturbed by preceding quantized layers (Li et al., 2025). Sparse and outlier-aware variants further show that a small subset of weight components can dominate quantization quality: SpQR stores outlier weights in sparse high precision (Dettmers et al., 2023b), and SqueezeLLM combines dense low-bit quantization with sparse outlier representations (Kim et al., 2023). These methods motivate the view that low-bit LLM quantization is often governed by a small set of sensitive components.

Activation difficulty, smoothing, and rotations Activation quantization is more challenging because activations are input-dependent, quantized online, and often contain large outlier channels. LLM.int8() identifies emergent activation outliers and preserves them through mixed-precision matrix multiplication (Dettmers et al., 2022). Outlier Suppression mitigates activation outliers with asymmetric quantization and range control (Wei et al., 2022), while SmoothQuant applies an equivalent scaling transformation that migrates quantization difficulty from activations to weights (Xiao et al., 2023). Another line of work changes the quantization basis: QuIP uses incoherence processing (Chee et al., 2023), QuaRot applies fixed orthogonal rotations such as Hadamard transforms (Ashkboos et al., 2024), and SpinQuant learns rotation matrices for improved quantization quality (Liu et al., 2025b). OSTQuant learns orthogonal and scaling transformations (Hu et al., 2025), and TurboQuant pairs random rotations with near-optimal scalar quantizers (Zandieh et al., 2026). These methods reduce or redistribute activation-side difficulty before quantization.

Low-rank compression and compensation Low-rank structure has been widely used for model compression and quantization compensation. ASVD incorporates activation information into SVD-based compression (Yuan et al., 2023), and SVD-LLM studies truncation-aware SVD and whitening strategies for compressing LLMs (Wang et al., 2025). QLoRA shows that low-rank adapters can be combined with quantized backbones for efficient finetuning (Dettmers et al., 2023a). More closely related to our deployment form, SVDQuant uses low-rank components to absorb outliers for 4-bit diffusion model quantization (Li et al., 2024a), and ResQ keeps the highest-variance activation subspace in higher precision (Saxena et al., 2025). ARHQ differs from both in how the high-precision subspace is selected (Section 3.3).

3 Method

ARHQ is a post-training method for constructing a mixed-precision linear layer under aggressive

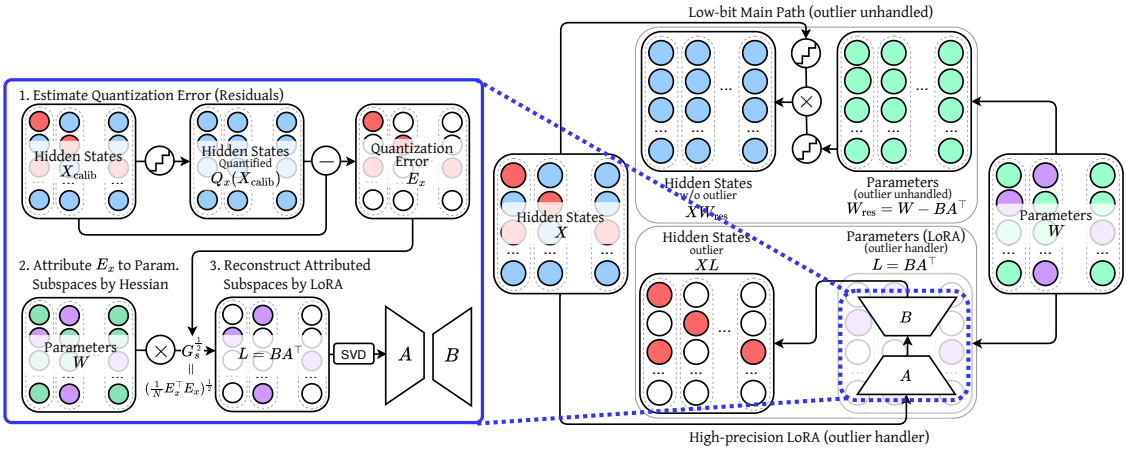


Figure 1: ARHQ decomposes each linear weight into a low-bit residual branch and a small BF16 low-rank branch. The low-rank branch is selected by measuring the activation quantization residuals produced by the deployed quantizer and performing a truncated SVD in the residual-Hessian metric. At inference time, the dense path remains quantized (*i.e.,* W4A4), while the BF16 low-rank branch corrects residual-sensitive directions.

activation-weight quantization (*e.g.,* W4A4 with 4-bit weights and 4-bit activations). For each Transformer linear layer, the method takes as input the full-precision weight matrix, a set of calibration activations, the deployed activation quantizer, and a target low-rank budget. It outputs a dense low-bit residual branch and a small high-precision low-rank branch.

The method consists of four steps as shown in Figure 1. First, we run the deployed activation quantizer on calibration activations and measure the residuals it actually produces. Second, we use these residuals to define a layer-local residual Hessian, which identifies the input directions where activation quantization errors have energy. Third, we solve a residual-weighted low-rank approximation problem to select the weight directions that most strongly amplify these residuals. Finally, we deploy the result as a quantized dense branch plus a BF16 LoRA-style correction branch.

3.1 From Residual Amplification to the ARHQ Objective

This subsection defines the optimization problem solved by ARHQ. The goal is to decide which part of the weight matrix should be moved out of the dense low-bit branch and represented by a high-precision low-rank branch. Intuitively, ARHQ selects the directions that most strongly amplify the activation residuals left by the deployed activation quantizer instead of the highest-energy activation directions. Mathematically, consider a linear layer with full-precision output:

$$Y = XW^\top, \tag{1}$$

where $X \in \mathbb{R}^{N \times D_{\text{in}}}$ and $W \in \mathbb{R}^{D_{\text{out}} \times D_{\text{in}}}$. Let $Q_x(\cdot)$ and $Q_w(\cdot)$ denote the activation and weight quantizers used by the deployed low-bit pipeline. On calibration activations, we measure the activation quantization residual as:

$$E_x = Q_x(X) - X. \tag{2}$$

The weight quantization residual is similarly defined as:

$$E_w = Q_w(W) - W. \tag{3}$$

These residuals determine the error of a quantized linear layer:

$$\begin{aligned} \hat{Y} - Y &= Q_x(X)Q_w(W)^\top - XW^\top \\ &= E_x W^\top + X E_w^\top + E_x E_w^\top. \end{aligned} \tag{4}$$

The first term, $E_x W^\top$, is the activation-residual propagation term. It captures how the activation errors introduced online by $Q_x(\cdot)$ are propagated through the full layer weight. ARHQ focuses on this term because it is directly induced by activation quantization, and because it reveals which weight directions are most exposed to the measured activation residuals. To reduce this amplification, ARHQ splits the weight matrix into two branches:

$$W = W_{\text{res}} + L, \qquad \text{rank}(L) \le r. \tag{5}$$

The dense residual branch W_{res} remains in the low-bit path, while the low-rank branch L is kept in

Algorithm 1 Activation Residual Hessian Quantization in the Smoothed Space

Require: Original weight matrix $W \in \mathbb{R}^{D_{out} \times D_{in}}$, Calibration activations $X_{\text{calib}} \in \mathbb{R}^{N_{\text{calib}} \times D_{in}}$, Positive diagonal smoothing matrix $S = \text{diag}(s)$, Activation quantizer $Q_x(\cdot)$, Target rank r, Regularization floor ϵ

Ensure: Main branch residual weight $W_{\text{res},s}$, Down-projection matrix A_s, Up-projection matrix B

Stage 1: (Optional) Smoothing Preprocessing

$X_s \leftarrow X_{\text{calib}} S^{-1}$ ▷ Smooth calibration activations

$W_s \leftarrow WS$ ▷ Smooth weight matrix

Stage 2: Residual Covariance Computation

$X_q \leftarrow Q_x(X_s)$ ▷ Fake-quantize in the smoothed space

$E_s \leftarrow X_q - X_s$ ▷ Compute smoothed activation residuals

$G_s \leftarrow \frac{1}{N_{\text{calib}}} E_s^\top E_s$ ▷ Construct residual covariance (Hessian)

Stage 3: Numerical Regularization

Compute eigendecomposition $G_s = U \text{diag}(\lambda) U^\top$

$G_{s,\epsilon} \leftarrow U \text{diag}(\max(\lambda_i, \epsilon)) U^\top$ ▷ Apply eigenvalue floor

Construct $G_{s,\epsilon}^{1/2}$ and $G_{s,\epsilon}^{-1/2}$

Stage 4: Scaled Truncated SVD

$M_s \leftarrow W_s G_{s,\epsilon}^{1/2}$ ▷ Scale weights by residual metric

$U_r, \Sigma_r, V_r \leftarrow \text{TruncatedSVD}(M_s, r)$ ▷ Extract top-r components

Stage 5: Factorization for Deployment

$B \leftarrow U_r \Sigma_r$ ▷ Construct up-projection factor

$A_s \leftarrow G_{s,\epsilon}^{-1/2} V_r$ ▷ Construct down-projection factor

$L_s \leftarrow B A_s^\top$ ▷ Form high-precision low-rank branch

$W_{\text{res},s} \leftarrow W_s - L_s$ ▷ Compute residual weight for main branch

return $W_{\text{res},s}, A_s, B$

high precision. At inference time, the layer is implemented as:

$$\hat{Y} = Q_x(X) Q_w(W_{\text{res}})^\top + X L^\top. \tag{6}$$

The role of L is therefore to remove from the low-bit branch those weight directions that would otherwise amplify the activation quantization residuals. Since $W_{\text{res}} = W - L$, the residual amplification left in the dense low-bit branch is:

$$\|E_x (W - L)^\top\|_F^2. \tag{7}$$

ARHQ selects the rank-r branch by minimizing this quantity:

$$\min_{\text{rank}(L) \le r} \|E_x (W - L)^\top\|_F^2. \tag{8}$$

This objective formalizes the central principle of ARHQ: the high-precision branch should capture the weight directions that are most sensitive to the activation residuals produced by the actual deployed quantizer.

We now rewrite the objective in a residual-induced metric. Define the activation residual Hessian as $G_x = \frac{1}{N} E_x^\top E_x$. For a residual weight matrix R, the residual-amplification energy is

$$\begin{aligned} \mathcal{A}(R; E_x) &= \frac{1}{2N} \|E_x R^\top\|_F^2 \\ &= \frac{1}{2} \text{Tr}(R G_x R^\top). \end{aligned} \tag{9}$$

Since $\nabla_R \mathcal{A}(R; E_x) = R G_x$, the matrix G_x is the input-side Hessian of this layer-local quadratic objective. It is not a Hessian of the language-modeling loss. Instead, it measures where the activation quantizer leaves residual energy and how this energy can be amplified by the remaining low-bit weights. Using G_x, the objective in Equation (8) becomes

$$\|E_x (W - L)^\top\|_F^2 = N \|(W - L) G_x^{1/2}\|_F^2. \tag{10}$$

Thus, selecting the low-rank correction becomes a weighted low-rank approximation problem, where the weighting is determined by the measured activation quantization residuals.

ARHQ does not optimize the full quantization reconstruction loss. After the split, the term $Q_w(W - L)$ makes the exact objective quantizer-dependent, non-smooth, and coupled to L. Instead,

ARHQ isolates the activation-residual amplification term, which is induced by online activation quantization and admits a closed-form post-training solution. The remaining weight error is handled by quantizing $W_{\text{res}} = W - L$.

We now solve the residual-weighted objective in Equation (10) and convert the solution into a deployable low-rank form. The residual Hessian defines the metric in which the weight matrix should be approximated. Therefore, instead of applying SVD directly to W, ARHQ first forms

$$M = WG_x^{1/2}. \tag{11}$$

Let $M_r = U_r\Sigma_r V_r^\top$ be the rank-r truncated SVD of M. The corresponding low-rank branch in the original weight space is

$$L^\star = M_r G_x^{-1/2}, \tag{12}$$

which follows from the change of variables $\widetilde{L} = LG_x^{1/2}$. We provide a more formal proposition in Appendix A.

For deployment, we store $L^\star$ in a LoRA-style factorized form:

$$L^\star = BA^\top, \;\; B = U_r\Sigma_r, \;\; A = G_x^{-1/2}V_r. \tag{13}$$

The dense low-bit branch is then

$$W_{\text{res}} = W - L^\star, \tag{14}$$

and the high-precision side branch computes $(XA)B^\top$.

3.2 ARHQ in the Smoothed Space

ARHQ is naturally compatible with activation smoothing. Smoothing first changes the coordinate system in which activation quantization is performed, and ARHQ then measures the residuals that remain in that system.

Let $S = \text{diag}(s)$ be a positive diagonal smoothing matrix. We use the standard equivalent transformation

$$X_s = XS^{-1}, \;\; W_s = WS, \tag{15}$$

which preserves the full-precision output:

$$X_s W_s^\top = XW^\top. \tag{16}$$

ARHQ is then applied in the smoothed space. Given calibration activations, we compute

$$E_s = Q_x(X_{\text{calib}}S^{-1}) - X_{\text{calib}}S^{-1}, \tag{17}$$

$$G_s = \frac{1}{N_{\text{calib}}} E_s^\top E_s. \tag{18}$$

After applying an eigenvalue floor to obtain $G_{s,\epsilon}$, we form

$$M_s = W_s G_{s,\epsilon}^{1/2}. \tag{19}$$

Let $M_{s,r} = U_r\Sigma_r V_r^\top$ be its rank-r truncated SVD. The deployed factors and residual branch are

$$B = U_r\Sigma_r, \quad A_s = G_{s,\epsilon}^{-1/2}V_r, \tag{20}$$

$$W_{\text{res},s} = W_s - BA_s^\top. \tag{21}$$

Smoothing and ARHQ play different roles: smoothing reduces or redistributes activation quantization difficulty, while ARHQ corrects the residual-sensitive directions that remain after smoothing.

3.3 Relation to SVDQuant

ARHQ shares the dense-plus-low-rank structure of SVDQuant (Li et al., 2024a); the two differ only in how the rank-r branch is selected. Both solve $\min_{\text{rank}(L)\le r}\|(W-L)C^{1/2}\|_F^2$: SVDQuant uses $C = I$ (the weight spectrum), an activation-Hessian variant (AHQ) uses $C = X^\top X/N$, and ARHQ uses $C = G_x$, which, unlike weight or activation energy, reflects the residual actually left by the deployed quantizer. At a matched rank, the three have the same inference cost, so SVDQuant is our most direct controlled baseline.

3.4 Inference Path and Computational Cost

The previous subsections describe how ARHQ constructs the two branches. This subsection explains how they are used during inference and what overhead they introduce.

At inference time, the smoothed activation XS^{-1} is available at the linear-layer boundary before online activation quantization. The deployed ARHQ layer evaluates two parallel branches:

$$\hat{Y} = Q_x(XS^{-1})Q_w(W_{\text{res},s})^\top + (XS^{-1}A_s)B^\top. \tag{22}$$

The first term is the dense low-bit branch. Both the smoothed activation and the residual weight are quantized, so this branch preserves the standard W4A4 matrix multiplication path. The second term is the high-precision low-rank branch. It uses the BF16 activation tensor available before online activation quantization and applies a rank-r correction selected by the residual Hessian.

Therefore, ARHQ should be interpreted as a W4A4 dense path augmented with a small BF16 low-rank correction branch, rather than as a purely

Method	Attention					FFN				Overall Average
	q_{proj}	k_{proj}	v_{proj}	o_{proj}	Avg.	gate	up	down	Avg.	
SVDQuant	24.97	24.51	18.98	18.76	21.80	25.58	19.09	19.86	21.51	21.68
AHQ	25.34	25.25	19.20	18.51	22.08	25.51	19.06	20.06	21.54	21.85
ARHQ	26.80	26.92	20.09	19.00	23.20	26.95	20.16	20.29	22.47	22.89
SVDQuant + *Smooth*	26.07	25.92	19.92	19.08	22.75	26.78	20.25	20.14	22.39	22.60
AHQ + *Smooth*	26.00	25.98	19.88	18.76	22.66	26.35	19.99	20.11	22.15	22.44
ARHQ + *Smooth*	**27.06**	**27.24**	**20.41**	**19.28**	**23.50**	**27.28**	**20.49**	**20.43**	**22.73**	**23.17**

Table 1: Layer-wise reconstruction SQNR (dB) on Qwen3-4B-Thinking under W4A4 quantization with rank $r = 64$. Results are averaged over 252 linear layers (36 blocks $\times$ 7 projections) and reported separately for attention and FFN projections. SVDQuant, AHQ, and ARHQ differ in the metric used to select the high-precision low-rank branch: weight geometry, activation Hessian, and activation residual Hessian, respectively. "+ Smooth" applies activation smoothing before decomposition.

integer-only W4A4 operator. The deployment role of each component is explicit:

$$\text{low-bit branch:} \quad Q_x(XS^{-1})Q_w(W_{\text{res},s})^\top, \tag{23}$$

$$\text{correction branch:} \quad (XS^{-1}A_s)B^\top. \tag{24}$$

For sequence length N, input dimension D_{in}, output dimension D_{out}, and rank r, the correction branch introduces an additional arithmetic cost of

$$O(Nr(D_{\text{in}} + D_{\text{out}})). \tag{25}$$

The additional parameter count is

$$r(D_{\text{in}} + D_{\text{out}}). \tag{26}$$

For square layers with $D_{\text{in}} = D_{\text{out}} = D$, the relative parameter overhead is approximately

$$\frac{2r}{D}. \tag{27}$$

Thus, the rank r directly controls the quality–efficiency trade-off. Larger ranks provide stronger residual correction, while smaller ranks preserve more of the throughput and memory benefits of the low-bit dense branch. Algorithm 1 summarizes the full ARHQ procedure in the smoothed space.

4 Experiments

4.1 Setup summary

We evaluate ARHQ in a bottom-up order: layer-wise reconstruction, perplexity and zero-shot accuracy on LLaMA-family models, reasoning and coding benchmarks on instruction/reasoning models, and finally inference efficiency.

For calibration, we sample 100K tokens from NVIDIA's Nemotron post-training dataset (Bercovich et al., 2025)[1]. The calibration tokens are used only to estimate smoothing scales, activation quantization residuals, and the residual Hessian G_x. All perplexity, zero-shot, reasoning, coding, and layer-wise SQNR evaluations are performed on held-out data. SVDQuant (our reproduction), AHQ, and ARHQ use the same calibration activations and smoothing configuration whenever applicable. The high-precision LoRA branch is evaluated at ranks $r \in \{32, 64, 128\}$ for the main model-level studies.

Unless otherwise specified, we use W4A4KV16 quantization: the residual-weight branch W_{res} and online activations are quantized to NVFP4, while the low-rank branch and the KV cache are kept in BF16. The same quantizer, scaling rule, and calibration activations are used to compute $E_x = Q_x(X) - X$. Since ARHQ only depends on the measured activation quantization residual, it is not tied to NVFP4; we also include a smaller set of INT4 experiments to verify portability (symmetric, group size 64 for weights and activations, no clipping). For reproducibility, accuracy is evaluated with simulated quantization in HuggingFace Transformers, and efficiency experiments are implemented with `vllm==0.20.1`, `torch==2.11.0`, and CUDA 12.9.

The evaluated models include LLaMA-family base and instruction models, including LLaMA-2-7B (Touvron et al., 2023), LLaMA-3-8B, LLaMA-3-70B, and LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024), as well as Qwen3-4B-Thinking (Yang et al., 2025). For language-model quality, we report perplexity on WikiText2 (Merity et al., 2016) and a

[1] https://huggingface.co/datasets/nvidia/Llama-Nemotron-Post-Training-Dataset

Method	ARC-c	ARC-e	BoolQ	HellaS.	Lam.	OBQA	PIQA	SIQA	WinoG.	Avg.	Wiki2 (↓)
*Baseline model with **16bit**-weights and **16bit**-activations*											
BF16	53.50	77.74	81.10	79.18	75.74	44.80	80.63	47.08	73.01	68.09	6.14
*Reference models with **4bit**-weights and **16bit**-activations*											
RTN-W4A16	48.98	73.23	72.75	75.90	63.85	43.20	78.40	43.81	73.16	63.70	8.13
SmoothQuant-W4A16	47.44	72.35	72.11	74.92	62.41	43.00	77.69	43.91	71.27	62.79	8.12
GPTQ-W4A16	49.74	72.52	71.28	68.34	46.69	43.60	78.78	46.47	71.82	61.03	7.43
OmniQuant-W4A16	50.09	74.54	79.51	76.92	70.31	43.80	79.54	44.52	71.74	65.66	7.19
AWQ-W4A16	52.22	76.68	80.31	77.51	74.81	44.20	79.60	46.26	71.67	67.03	7.36
QuaRot-W4A16	51.88	77.53	79.60	77.87	74.11	44.40	80.14	46.37	73.56	67.27	6.53
SpinQuant-W4A16	52.13	72.28	79.20	78.40	73.76	44.80	79.98	45.50	72.77	66.54	6.49
OSTQuant-W4A16	52.82	79.84	80.31	77.86	76.48	42.80	80.74	45.55	73.80	67.80	6.53
*Reference models with **4bit**-weights and **4bit**-activations*											
RTN	23.72	30.89	46.30	31.26	3.03	27.60	52.72	35.26	50.04	33.42	inf
SmoothQuant	23.29	28.28	48.93	29.19	1.57	28.60	54.46	33.37	49.64	33.04	inf
GPTQ	23.46	32.07	43.79	30.10	2.41	28.00	53.97	34.14	48.86	32.98	inf
QuaRot*	43.34	67.93	74.07	73.05	66.19	41.40	75.57	33.32	65.90	60.09	8.20
SpinQuant*	44.71	70.92	69.08	73.85	68.14	41.00	75.14	43.96	68.51	61.70	8.03
OSTQuant	48.81	73.48	79.82	75.97	72.62	42.40	78.18	45.75	69.22	65.14	7.24
*Proposed models with **4bit**-weights and **4bit**-activations*											
ARHQ *rank 32*	**50.43**	74.41	77.40	76.99	73.92	**44.20**	78.62	45.75	**73.09**	66.09	7.04
ARHQ *rank 64*	48.55	**75.51**	80.18	**77.08**	**75.39**	43.60	**79.49**	45.34	72.30	66.38	6.95
ARHQ *rank 128*	49.57	73.70	**81.41**	76.87	75.18	44.00	79.22	**46.32**	71.43	**66.41**	**6.89**

Table 2: Zero-shot accuracy and WikiText2 perplexity on LLaMA-3-8B. Baselines have no high-precision branch, so the comparison is system-level rather than equal-budget (see Table 4). Best reference results are underlined; best ARHQ results are in bold. *Re-run with corrected configurations under W4A4KV4.

Methods	MATH	MMLU	GSM	HEval
Evaluation using LLaMA-3.1-8B-Instruct				
BF16	48.00	43.43	80.52	58.54
SVDQuant *rank 32*	39.80	34.46	74.45	52.44
SVDQuant *rank 64*	38.80	36.36	76.95	53.05
ARHQ *rank 32*	36.80	36.22	75.74	51.83
ARHQ *rank 64*	**41.40**	**39.10**	**78.39**	**55.49**
Evaluation using Qwen3-4B-Thinking				
BF16	93.55	62.69	95.54	95.12
SVDQuant *rank 32*	91.70	61.09	91.62	92.84
SVDQuant *rank 64*	92.50	60.96	**94.62**	94.51
ARHQ *rank 32*	92.70	62.26	94.28	94.51
ARHQ *rank 64*	**92.65**	**62.20**	94.29	**94.62**

Table 3: Evaluation results on reasoning and coding benchmarks. The best performances of *rank 32* models are underlined, while the best performances of *rank 64* models are depicted in bold font.

Method	Rank	Avg.	Wiki2 (↓)
BF16	–	68.09	6.14
Homogeneous W4A4			
GPTAQ	–	62.40	7.38
W4A4 + high-precision low-rank branch			
ResQ†	1/8 (8-bit)	64.41‡	7.00
SVDQuant	32	65.53	7.06
SVDQuant	64	65.81	7.01
SVDQuant	128	66.15	6.93
ARHQ	32	66.09	7.04
ARHQ	64	66.38	6.95
ARHQ	128	66.41	6.89
ARHQ-w	32	**66.69**	6.96
ARHQ-w	64	65.87	6.91
ARHQ-w	128	66.50	**6.79**

Table 4: Comparison with GPTAQ and mixed-precision methods on LLaMA-3-8B (W4A4KV16; 9-task zero-shot average and WikiText2 perplexity). †KV-cache quantization disabled; 1/8 of the hidden dimension kept in 8-bit. ‡Eight tasks without LAMBADA; the eight-task averages of ARHQ are 65.11/65.26/65.31.

9-task zero-shot average over ARC-Challenge and ARC-Easy (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), LAMBADA (Paperno et al., 2016), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021). For reasoning and coding, we evaluate MATH500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), MMLU-Pro (Wang et al., 2024), and HumanEval (Chen et al., 2021), averaging four sampled generations for Qwen3-4B-Thinking. For layer-wise analysis, we report output SQNR on attention and FFN projections.

We compare with homogeneous baselines without a high-precision branch (RTN, SmoothQuant (Xiao et al., 2023), GPTQ (Frantar et al., 2022), OmniQuant (Shao et al., 2024), AWQ (Lin et al., 2024), GPTAQ (Li et al., 2025), QuaRot (Ashkboos et al., 2024), SpinQuant (Liu et al., 2025b), and OSTQuant (Hu et al., 2025); QuaRot and SpinQuant keep their original

4-bit KV cache), for which the comparison is system-level rather than equal-budget, and with mixed-precision baselines: SVDQuant (Li et al., 2024a), reproduced with our calibration data, smoothing, quantizer, and rank as the direct controlled baseline, and ResQ (Saxena et al., 2025) under W4A4KV16 with its default 1/8 fraction of 8-bit channels (Appendix C).

4.2 Layer-wise reconstruction and ablations

We first isolate the quantized linear layer, which directly tests the low-rank objective. Given a floating-point output Y and the output $\hat{Y}$ produced by the quantized residual branch plus the high-precision low-rank branch, we compute the signal-to-quantization-noise ratio (SQNR) as:

$$\mathrm{SQNR}(Y, \hat{Y}) = 10 \log_{10} \left(\frac{\|Y\|_F^2}{\|Y - \hat{Y}\|_F^2} \right). \quad (28)$$

A higher SQNR indicates better preservation of the floating-point linear output.

The SQNR study evaluates all major Transformer linear projections, including attention projections (q, k, v, o) and FFN projections $(gate, up, down)$ on Qwen3-4B-Thinking model. We compare ARHQ against SVDQuant and AHQ, both with and without smoothing. SVDQuant selects low-rank directions solely from the weight matrix, whereas AHQ selects it using an activation-Hessian-weighted objective.

This ablation isolates the effect of the low-rank selection criterion and its interaction with activation smoothing. As shown in Table 1, AHQ gives only marginal and non-uniform gains over SVDQuant, whereas ARHQ consistently achieves the highest SQNR across attention, FFN, and the overall average. In particular, ARHQ improves the overall SQNR over SVDQuant by 1.21 dB without smoothing and by 0.57 dB with smoothing. The best result is obtained by ARHQ+Smooth, suggesting that the measured activation quantization residual provides a more direct signal for low-rank correction and remains complementary to activation smoothing.

4.3 Perplexity and zero-shot evaluation on LLaMA models

We next evaluate whether the deployed ARHQ pipeline translates layer-wise reconstruction gains into standard language-model metrics. On LLaMA-family base models, we report WikiText2 perplexity and the 9-task zero-shot average, with rank ablations at $r = 32, 64, 128$ to characterize the quality–efficiency trade-off. ARHQ is evaluated as a W4A4 dense branch augmented with a BF16 low-rank correction branch.

As summarized in Table 2, ARHQ achieves strong end-to-end quality under aggressive W4A4 quantization. Compared with existing W4A4 quantization methods, ARHQ remains competitive or better across both zero-shot accuracy and perplexity. Notably, ARHQ also shows competitive performance against W4A16 quantization settings, despite using low-bit activations. Larger correction ranks further improve perplexity in most cases . Overall, these results demonstrate the effectiveness of ARHQ as a practical mixed-precision path for accurate low-bit inference.

Table 4 isolates the selection criterion: at every matched rank, ARHQ outperforms SVDQuant in both perplexity and zero-shot accuracy at the same inference cost, and even at rank 32 it exceeds ResQ in zero-shot accuracy (65.11 vs. 64.41 on ResQ's eight tasks) and GPTAQ in both metrics. Finally, ARHQ-w adds a calibration-time refinement that also accounts for the weight quantization error of W_{res} (Appendix D), with the same rank and runtime operations. It lowers perplexity at every rank (e.g., 7.04→6.96 at rank 32) and improves zero-shot accuracy at ranks 32 and 128, but not at rank 64.

4.4 Reasoning and coding benchmarks

We further evaluate instruction and reasoning models, where quantization errors can be amplified during long generations and multi-step reasoning. We compare against a reproduced SVDQuant low-rank split in the same inference pipeline. The evaluated models include LLaMA-3.1-8B-Instruct and Qwen3-4B-Thinking. The benchmark suite covers mathematical reasoning, broad professional knowledge, grade-school arithmetic, and code generation through MATH500, MMLU-Pro, GSM8K, and HumanEval.

As shown in Table 3, ARHQ is most beneficial when the low-rank branch has sufficient capacity. At small ranks, the residual Hessian may identify directions that improve reconstruction but do not uniformly dominate all downstream tasks. With a larger rank budget, ARHQ more consistently improves over the matched SVDQuant, especially on LLaMA-3.1-8B-Instruct. On Qwen3-4B-Thinking, the results are closer and task-dependent,

Model	Rank	SeqLen	Prefill Throughput		Prefill Ratio	Decode Throughput		Decode Ratio	Memory		Memory Saving
			BF16	ARHQ		BF16	ARHQ		BF16	ARHQ	
LLaMA3.1-8B	32	512	6041.5	19376.4	3.21×	195.9	336.0	1.72×	15.21	6.02	60.42%
		2048	6118.2	16912.3	2.76×	190.3	356.8	1.88×	15.96	6.77	57.57%
		8192	5419.3	12625.3	2.33×	173.6	305.8	1.76×	18.96	9.77	48.47%
	128	512	6041.5	15575.9	2.58×	195.9	332.1	1.69×	15.21	6.49	57.33%
		2048	6118.2	13813.7	2.26×	190.3	318.5	1.67×	15.96	7.24	54.64%
		8192	5419.3	11256.2	2.08×	173.6	272.7	1.57×	18.96	10.24	45.99%
Qwen3-4B	32	512	10972.3	21866.2	1.99×	316.5	470.0	1.49×	7.77	3.03	61.02%
		2048	10611.5	22062.4	2.08×	310.5	463.2	1.49×	8.62	3.88	54.97%
		8192	8288.1	14884.2	1.80×	263.1	351.8	1.34×	11.99	7.25	39.55%
	128	512	10972.3	20305.1	1.85×	316.5	402.7	1.27×	7.77	3.40	56.27%
		2048	10611.5	19308.8	1.82×	310.5	410.5	1.32×	8.62	4.25	50.68%
		8192	8288.1	13693.0	1.65×	263.1	329.7	1.25×	11.99	7.62	36.47%

Table 5: Prefill/decode throughput and memory usage are reported for LLaMA3.1-8B and Qwen3-4B under BF16 and ARHQ with LoRA ranks 32 and 128 across different sequence lengths. Throughput ratios denote ARHQ speedup over BF16, memory saving denotes relative GiB reduction, and all results are measured with batch size 4, BF16 LoRA, and vLLM compile plus CUDA graph on an NVIDIA GeForce RTX 5090 GPU.

but ARHQ remains near the BF16 reference while preserving the W4A4 serving benefits.

4.5 Inference efficiency and memory

Finally, we measure the serving cost of the quantized model with the high-precision LoRA correction branch. The deployed ARHQ path combines a low-bit weight branch with a LoRA-style high-precision correction branch, and we report prefill throughput, decode throughput, and peak GPU memory.

As reported in Table 5, ARHQ consistently improves serving efficiency over the BF16 baseline across both models, ranks, and sequence lengths. It delivers large prefill and decode speedups while substantially reducing peak GPU memory. Increasing the correction rank from 32 to 128 reduces the efficiency gain due to the larger high-precision branch, but ARHQ still maintains clear throughput and memory advantages over BF16. These results show that ARHQ defines practical operating points that balance reconstruction quality, model accuracy, and serving efficiency.

Beyond BF16, Table 7 in the appendix compares against AWQ W4A16 on LLaMA-3.1-8B. ARHQ is 2.2–3.1× faster in compute-bound prefill, but reaches only 40–83% of AWQ's throughput in memory-bound decoding at batch size 4, mainly due to the extra memory traffic of the low-rank branch; at batch size 24, ARHQ at rank 32 exceeds AWQ (107%). ARHQ thus targets prefill and larger-batch serving rather than latency-oriented low-batch decoding. All measurements use a single GPU without tensor-parallel communication.

5 Conclusion

We presented ARHQ, a post-training residual-aware low-rank correction method for low-bit activation-weight quantization. By measuring the residuals produced by the deployed activation quantizer and constructing an activation residual Hessian, ARHQ identifies weight directions that most amplify activation quantization errors and assigns them to a small high-precision branch. At a matched rank, it outperforms SVDQuant in reconstruction, perplexity, and downstream accuracy, with the largest efficiency gains in compute-bound prefill. These results suggest that aggressive activation-weight quantization should be treated not only as an outlier or weight-sensitivity problem, but also as a residual-propagation problem determined by the deployed quantizer.

Acknowledgments

This research was supported by AMED under Grant Number JP26wm0625405. We thank the anonymous reviewers and the area chair for their constructive feedback.

Limitations

While ARHQ mitigates activation residual propagation, it has several limitations.

Activation-residual-only objective The closed-form ARHQ objective optimizes only the activation-residual propagation term $E_x W^\top$ and does not model the weight-quantization term $X E_w^\top$ or the cross term $E_x E_w^\top$ of the deployed W4A4 error. In particular, it does not explicitly control whether the residual weight $W_{\text{res}} = W - L$ remains quantization-friendly. A joint objective would lose the closed-form solution because the weight quantizer is non-differentiable and depends on L. We quantify this gap with a calibration-time refinement, ARHQ-w (Appendix D), which incorporates the dense-branch weight quantization error and improves LLaMA-3-8B at rank 32 from 7.04 to 6.96 perplexity without changing the inference path. The original ARHQ objective should therefore be understood as isolating one important component of the deployed error rather than as a global optimizer of the full W4A4 error.

Sequential smoothing and splitting ARHQ applies activation smoothing before low-rank splitting. Since smoothing changes the residual Hessian, jointly optimizing smoothing scales and the low-rank branch may yield a better quality–efficiency trade-off, but is left for future work.

Uniform rank allocation We use the same rank r for all layers. However, layers have different sensitivity to activation residuals. Adaptive rank allocation based on layer-wise spectra or residual sensitivity could further improve efficiency.

Scope of efficiency gains ARHQ targets compute-bound regimes. Its W4A4 dense path benefits from low-precision tensor-core GEMMs during prefill and large-batch serving, but the BF16 low-rank branch is also executed during decoding. In low-batch, memory-bound decoding, weight-only W4A16 kernels such as AWQ are faster than ARHQ (Table 7). Our current implementation runs the correction branch as a separate LoRA-style kernel; a fused kernel could reduce launch overhead. All efficiency results are measured on a single GPU, so they do not capture the additional communication that the low-rank branch may incur under tensor-parallel serving.

References

Yamato Arai and Yuma Ichikawa. 2025. Quantization error propagation: Revisiting layer-wise post-training quantization. In *Advances in Neural Information Processing Systems*, volume 38, pages 151916–151951. Curran Associates, Inc.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, and 117 others. 2025. Llama-nemotron: Efficient reasoning models. *arXiv preprint arXiv:2505.00949*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. 2023. Quip: 2-bit quantization of large language models with guarantees. *Advances in neural information processing systems*, 36:4396–4429.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, pages 2924–2936.

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

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pages 30318–30332. Curran Associates, Inc.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023a. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023b. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

Vage Egiazarian, Roberto L Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2025. Bridging the gap between promise and performance for microscaling fp4 quantization. *arXiv preprint arXiv:2509.23202*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Yundong Gai and Ping Li. 2024. Atq: Activation transformation forweight-activation quantization of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 17187–17194.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 540 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Xing Hu, Yuan Cheng, Dawei Yang, Zukang Xu, Zhihang Yuan, Jiangyong Yu, Chen Xu, Zhe Jiang, and Sifan Zhou. 2025. Ostquant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. *arXiv preprint arXiv:2501.13987*.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. 2023. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*.

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

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. 2024a. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*.

Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024b. Evaluating quantized large language models. *arXiv preprint arXiv:2402.18158*.

Yuhang Li, Ruokai Yin, Donghyun Lee, Shiting Xiao, and Priyadarshini Panda. 2025. Gptaq: Efficient finetuning-free quantization for asymmetric calibration. In *Forty-second International Conference on Machine Learning*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100.

Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng Yu, Chun Yuan, and Lu Hou. 2025a. Quantization hurts reasoning? an empirical study on quantized reasoning models. *arXiv preprint arXiv:2504.04823*.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2025b. Spinquant: Llm quantization with learned rotations. *URL https://arxiv.org/abs/2405.16406*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. 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*, pages 2381–2391. Association for Computational Linguistics.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro

Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 1525–1534.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 4463–4473.

Utkarsh Saxena, Sayeh Sharify, Kaushik Roy, and Xin Wang. 2025. ResQ: Mixed-precision quantization of large language models with low-rank residuals. In *Forty-second International Conference on Machine Learning*.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Gao Peng, Yu Qiao, and Ping Luo. 2024. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, volume 2024, pages 45472–45496.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 10.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2025. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, volume 2025, pages 19299–19319.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. 2026. Turboquant: Online vector quantization with near-optimal distortion rate. In *The Fourteenth International Conference on Learning Representations*.

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

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

A Closed-form solution of ARHQ

Proposition 1 (ARHQ closed-form decomposition)**.** *Let $W \in \mathbb{R}^{D_{\text{out}} \times D_{\text{in}}}$ and assume the activation residual covariance $G_x \succ 0$. Define the scaled weight matrix $M = WG_x^{1/2}$. If M_r is the optimal rank-r approximation of M under the Frobenius norm, then the exact solution to the weighted objective*

$$\min_{\text{rank}(L)\le r} \|(W - L)G_x^{1/2}\|_F^2 \tag{29}$$

is strictly given by

$$L^\star = M_r G_x^{-1/2} \tag{30}$$

Proof. We introduce the change of variables $\widetilde{L} = LG_x^{1/2}$. Because $G_x \succ 0$, the matrix $G_x^{1/2}$ is invertible, which guarantees that right-multiplication preserves the matrix rank, i.e., $\text{rank}(\widetilde{L}) = \text{rank}(L)$. The weighted objective can thus be rewritten as a standard unweighted approximation problem:

$$\min_{\text{rank}(\widetilde{L})\le r} \|M - \widetilde{L}\|_F^2 \tag{31}$$

By the Eckart-Young-Mirsky theorem, the optimal solution for $\widetilde{L}$ is the truncated SVD of M, denoted as M_r. Mapping this solution back to the original parameter space yields $L^\star = M_r G_x^{-1/2}$, concluding the proof. □

B Numerical Regularization

In practice, G_x can be singular or ill-conditioned, e.g., when the calibration token count is small relative to the input dimension, when some channels have nearly zero quantization residual, or when smoothing reduces the effective residual rank. Directly applying $G_x^{-1/2}$ may then produce unstable low-rank factors. We therefore floor the eigenvalues: with $G_x = U\text{diag}(\lambda)U^\top$, we use

$$G_{x,\epsilon} = U\text{diag}(\max(\lambda_i, \epsilon))U^\top, \tag{32}$$

and solve $\min_{\text{rank}(L)\le r}\|(W - L)G_{x,\epsilon}^{1/2}\|_F^2$. The closed-form solution keeps the same form with G_x replaced by $G_{x,\epsilon}$, and the floor prevents $G_x^{-1/2}$ from over-amplifying directions with negligible measured residual energy.

C Baseline Reproduction Details

SVDQuant. We reproduce the SVDQuant low-rank split in our pipeline: after the same smoothing as ARHQ (all LLaMA experiments use smoothing; only Table 1 also reports the unsmoothed variant), the low-rank branch is the rank-r truncated SVD of the smoothed weight, and the residual is quantized with the same NVFP4 quantizer. It therefore differs from ARHQ only in the selection metric.

ResQ. We use the official implementation with its default setting, which keeps a 1/8 fraction of the hidden dimension in 8-bit precision, and disable KV-cache quantization (W4A4KV16) to match our setting. Its zero-shot average in Table 4 covers eight tasks, as LAMBADA was not evaluated.

QuaRot and SpinQuant. After correcting a configuration issue in our earlier SpinQuant reproduction, both methods are rerun under the same calibration data, quantization format, and evaluation protocol as ARHQ. SpinQuant's rotations are optimized for 100 steps with learning rate 1.5.

D ARHQ-w: Accounting for Weight Quantization Error

The closed-form ARHQ split ignores the error of quantizing the dense branch $W_{\text{res}} = W - L$. ARHQ-w is a calibration-time refinement that incorporates this error while keeping the rank, the W4A4 dense path, the BF16 correction branch, and all runtime operations unchanged, so it adds no inference cost. Specifically, it iteratively incorporates the dense-branch weight quantization error during calibration when constructing the low-rank branch. On LLaMA-3-8B, ARHQ-w reduces perplexity at all ranks (Table 4).

E Subspace Comparison with SVDQuant

To compare the subspaces selected by ARHQ and SVDQuant, we measure the fraction of activation-residual amplification energy removed by a rank-r branch L:

$$\rho(L) = 1 - \frac{\|E_x(W - L)^\top\|_F^2}{\|E_x W^\top\|_F^2}. \tag{33}$$

By construction (Appendix A), ARHQ maximizes ρ among rank-r branches, so this analysis quantifies how far the weight-spectrum subspace of SVDQuant is from the residual-optimal one. We compute ρ for the ARHQ and SVDQuant factors of LLaMA-3-8B (NVFP4, with smoothing), evaluating both methods under the same G_x; ρ is therefore measured on calibration statistics rather than held-out data. As shown in Table 6, ARHQ

captures more residual-amplification energy than SVDQuant for every projection type and rank, and it is higher in all 224 linear layers at each of the three ranks. The gap is largest for `o_proj` (8.5 points at $r = 128$) and grows with the rank (from 2.0 points at $r = 32$ to 3.8 points at $r = 128$ on average). This confirms empirically that the weight-spectrum subspace selected by SVDQuant is not aligned with the directions that propagate the deployed quantizer's residual.

	$r = 32$		$r = 64$		$r = 128$	
	SVDQ.	ARHQ	SVDQ.	ARHQ	SVDQ.	ARHQ
q	21.44	**22.92**	29.53	**31.25**	39.62	**41.77**
k	26.77	**29.58**	38.75	**41.26**	53.88	**56.28**
v	10.17	**12.02**	18.17	**20.97**	31.51	**35.27**
o	9.81	**13.96**	15.67	**21.75**	24.34	**32.82**
gate	7.70	**8.36**	10.77	**12.02**	15.51	**17.97**
up	3.94	**4.86**	6.34	**8.02**	10.43	**13.54**
down	7.46	**9.33**	10.48	**13.19**	15.48	**19.46**
Avg.	12.47	**14.43**	18.53	**21.21**	27.25	**31.01**

Table 6: Residual-amplification energy ratio ρ (%) captured by the rank-r branch of SVDQuant and ARHQ on LLaMA-3-8B, averaged over the 32 layers for each projection type. SVDQ.: SVDQuant.

F Additional Experimental Results

Tables 8 and 9 report additional LLaMA-family results beyond the main LLaMA-3-8B comparison. These experiments follow the same evaluation protocol as the main results: perplexity is measured on WikiText2, and zero-shot accuracy is averaged over the same nine tasks. The purpose of these tables is to test whether the gains of ARHQ are specific to one model scale or remain visible across different LLaMA-family models.

Across these additional settings, ARHQ remains competitive under W4A4 quantization and generally improves as the rank increases. This trend is consistent with the interpretation that the high-precision branch corrects a low-dimensional subset of residual-sensitive directions: a larger rank increases the correction capacity, while still keeping the branch substantially smaller than the original dense matrix.

G Comparison with W4A16 Inference

Table 7 compares the throughput of ARHQ with AWQ W4A16 on LLaMA-3.1-8B, as discussed in Section 4.5.

BS	SeqLen	AWQ	ARHQ / AWQ		
		tok/s	r=32	r=64	r=128
Prefill					
4	512	6232	3.11×	2.89×	2.50×
4	2048	5765	2.93×	2.78×	2.40×
4	8192	5069	2.49×	2.43×	2.22×
Decode					
4	512	518.8	0.65×	0.50×	0.40×
4	2048	446.1	0.80×	0.77×	0.71×
4	8192	367.7	0.83×	0.79×	0.74×
24	4096	1155.4	1.07×	–	–

Table 7: Throughput of ARHQ (W4A4 + BF16 low-rank branch) relative to AWQ W4A16 on LLaMA-3.1-8B, on the same GPU and serving stack as Table 5. BS: batch size. ARHQ is faster in compute-bound prefill and large-batch decoding, and slower in low-batch, memory-bound decoding.

H Quantizer-format comparison

Hardware formats such as NVFP4 target efficient 4-bit floating-point training and inference on recent accelerators. Emerging analyses of microscaling FP4 formats study the gap between theoretical promise and empirical quantization behavior, including cases where conventional outlier mitigation may not fully transfer to FP4 settings (Egiazarian et al., 2025). These observations motivate quantizer-adaptive methods. ARHQ is designed for this setting because its metric $G_x = E_x^\top E_x/N$ is estimated after applying the actual activation quantizer, codebook, scaling rule, and calibration distribution.

Table 10 compares ARHQ under different 4-bit activation-weight formats. The same residual-aware construction applies because ARHQ only requires the measured activation residual

$$E_x = Q_x(X) - X. \tag{34}$$

Thus, the method is not tied to a particular quantizer family. Changing the quantizer changes the residual covariance G_x, and ARHQ adapts by recomputing the low-rank split under the new residual metric.

The comparison supports the view that ARHQ should be understood as a quantizer-adaptive splitting criterion rather than an NVFP4-specific heuristic. When the activation codebook, scaling rule, or clipping behavior changes, the residual covariance changes accordingly, and the low-rank branch is selected from the directions that are important under that deployed quantizer.

Method	ARC-c	ARC-e	BoolQ	HellaS.	Lam.	OBQA	PIQA	SIQA	WinoG.	Avg.	Wiki2 (↓)
*Baseline model with **16bit**-weights and **16bit**-activations*											
BF16	46.42	74.33	77.71	75.94	73.69	44.20	79.16	45.91	69.53	65.21	5.47
*Reference models with **4bit**-weights and **16bit**-activations*											
RTN-W4A16	42.15	67.59	73.06	72.34	67.18	41.80	76.50	44.11	66.69	61.27	7.02
SmoothQuant-W4A16	39.59	65.19	69.82	68.84	62.27	40.20	75.95	44.17	63.85	58.88	8.03
GPTQ-W4A16	42.49	69.53	61.31	73.83	67.61	42.40	77.64	44.52	68.43	60.86	9.84
OmniQuant-W4A16	42.49	71.00	74.34	73.85	70.70	44.20	78.40	44.93	68.82	63.19	5.74
AWQ-W4A16	44.11	70.75	78.07	74.98	70.68	43.80	78.13	45.14	69.38	63.89	5.83
QuaRot-W4A16	43.94	73.15	76.97	74.87	73.06	44.00	78.24	45.09	69.38	64.30	5.62
SpinQuant-W4A16	43.34	72.69	73.36	75.10	73.80	43.00	77.86	45.60	67.56	63.59	5.58
OSTQuant-W4A16	44.54	73.31	75.57	75.04	73.67	44.20	78.89	45.50	68.59	64.37	5.64
*Reference models with **4bit**-weights and **4bit**-activations*											
RTN	25.34	28.03	50.52	27.71	1.01	26.20	50.82	33.93	48.38	32.44	inf
SmoothQuant	28.33	26.39	49.39	27.28	1.18	23.40	48.80	33.62	50.75	32.13	inf
GPTQ	24.40	28.70	51.62	28.66	1.36	24.60	51.14	34.49	49.49	32.72	inf
QuaRot*	40.53	68.86	72.66	71.15	68.12	36.40	76.71	32.86	65.27	59.17	6.37
SpinQuant*	39.85	65.36	71.99	70.26	66.06	38.80	75.73	42.37	65.19	59.51	6.56
OSTQuant	44.03	71.93	75.41	74.94	73.22	43.20	78.51	45.85	68.03	63.90	5.60
*Proposed models with **4bit**-weights and **4bit**-activations*											
ARHQ rank 32	45.05	72.77	**76.36**	74.45	73.49	42.60	78.24	45.04	68.51	64.06	5.75
ARHQ rank 64	**45.65**	**73.65**	74.71	74.36	**74.17**	43.40	78.40	44.73	**68.75**	64.20	5.74
ARHQ rank 128	44.45	73.11	76.18	**74.53**	74.00	**43.60**	**78.51**	**45.60**	68.51	**64.28**	**5.71**

Table 8: Zero-shot accuracy and WikiText2 perplexity on LLaMA-2-7B (BF16 baseline). Best reference results are underlined; best proposed results are in bold. *Re-run with corrected configurations under W4A4KV4 (4-bit KV cache), following the original setups.

Method	ARC-c	ARC-e	BoolQ	HellaS.	Lam.	OBQA	PIQA	SIQA	WinoG.	Avg.	Wiki2 (↓)
*Baseline model with **16bit**-weights and **16bit**-activations*											
BF16	64.42	85.98	85.14	84.95	79.47	48.46	84.39	50.82	80.66	73.81	2.86
*Reference models with **4bit**-weights and **16bit**-activations*											
RTN-W4A16	26.28	25.55	37.83	26.36	0.00	29.00	50.98	34.70	49.64	31.15	inf
SmoothQuant-W4A16	51.88	77.53	80.09	80.47	73.16	46.60	80.58	45.29	75.85	67.94	6.70
GPTQ-W4A16	25.77	25.29	37.83	26.36	0.12	28.40	51.74	34.90	52.64	31.45	inf
OmniQuant-W4A16	48.29	75.42	77.92	77.80	75.59	45.20	80.41	46.62	70.17	66.38	5.02
AWQ-W4A16	52.26	78.95	83.24	81.52	73.05	47.67	81.25	44.43	77.98	68.93	5.92
*Reference models with **4bit**-weights and **4bit**-activations*											
RTN	27.47	25.88	37.83	26.26	0.00	27.20	51.63	35.26	49.33	31.21	inf
SmoothQuant	25.60	34.47	50.46	32.48	1.98	30.00	54.24	33.83	48.93	34.67	inf
GPTQ	25.77	26.09	43.64	26.42	0.00	27.40	52.01	32.55	49.33	31.47	inf
QuaRot	50.60	73.65	77.46	77.83	71.96	43.20	78.13	45.29	71.90	65.56	6.35
SpinQuant	53.84	77.69	80.24	78.19	73.06	45.00	78.67	43.24	73.01	66.99	6.10
OSTQuant	61.84	84.56	84.14	82.47	77.08	46.07	83.38	50.23	80.13	72.21	3.97
*Proposed models with **4bit**-weights and **4bit**-activations*											
ARHQ *rank 32*	61.93	84.13	82.12	82.14	77.43	47.20	83.43	50.32	80.21	72.10	3.74
ARHQ *rank 64*	62.05	**84.67**	**84.31**	82.79	78.30	46.32	83.52	50.26	80.03	72.47	3.67
ARHQ *rank 128*	**62.11**	83.92	84.28	**83.04**	**78.82**	**47.64**	**83.98**	**50.43**	**80.33**	**72.73**	**3.60**

Table 9: Zero-shot accuracy and WikiText2 perplexity on LLaMA-3-70B (BF16 baseline). Best reference results are underlined; best proposed results are in bold.

Method	ARC-c	ARC-e	BoolQ	HellaS.	Lam.	OBQA	PIQA	SIQA	WinoG.	Avg.	Wiki2 (↓)
Baseline model with 16bit-weights and 16bit-activations											
BF16	53.50	77.74	81.10	79.18	75.74	44.80	80.63	47.08	73.01	68.09	6.14
Proposed models with 4bit-weights and 4bit-activations in NVFP4 format											
ARHQ *rank 32*	50.43	74.41	77.40	76.99	73.92	44.20	78.62	45.75	73.09	66.09	7.04
ARHQ *rank 128*	49.57	73.70	81.41	76.87	75.18	44.00	79.22	46.32	71.43	66.41	6.89
Proposed models with 4bit-weights and 4bit-activations in INT4 format											
ARHQ *rank 32*	47.18	74.41	73.58	76.21	72.95	42.80	78.78	44.83	70.80	64.62	7.29
ARHQ *rank 128*	47.87	76.05	79.05	76.81	71.80	41.80	78.24	45.75	71.90	65.47	7.06

Table 10: Comparison of zero-shot accuracy and WikiText2 perplexity on LLaMA-3-8B using different low bit approaches.