RAMPQ: QUANTIZING WHAT RECOVERY CANNOT FIX FOR MIXED-PRECISION LLM QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-bit quantization reduces the memory footprint of large language models but often degrades model quality at aggressive precision. Mixed-precision quantization alleviates this problem by assigning different bit-widths based on module sensitivity or quantization error (Huang et al., 2024; Lee et al., 2025), while low-rank recovery adds a low-rank branch to recover the quantization error (Zhang et al., 2024; 2025; Saha et al., 2024). Pre-recovery quantization error, however, does not necessarily reflect the error that remains after recovery. We show that quantization-error recoverability varies substantially across modules and bit-widths, such that recovery can reorder where higher precision is most beneficial. This motivates allocating precision according to the error that survives recovery. We propose **RampQ**, a **R**ecovery-**A**ware **M**ixed-**P**recision **Q**uantization framework that performs allocation in the recovered representation space. RampQ constructs bit-specific $W_q + AB$ representations, evaluates each precision choice using module sensitivity and post-recovery residual error, and selects representations under their joint memory cost. A model-level refinement further searches complete-model assignments using selective end-to-end perplexity feedback. Our evaluation results show that RampQ improves the quality memory trade off over mixed precision and low rank recovery baselines. Compared with the best baseline under comparable memory settings, RampQ reduces perplexity on average by 8.1% on WikiText-2 and 5.7% on C4, and improves average zero-shot accuracy by 1.67 percentage points.

1 INTRODUCTION

Weight-only quantization is widely used to reduce the memory footprint of large language models (LLMs) (Frantar et al., 2022; Lin et al., 2024). At aggressive low-bit precision, however, the quantization error can significantly degrade model quality. Two complementary approaches have been studied to address this problem. Low-rank correction methods compensate for quantization error by representing the weight as $W_q + AB$ (Zhang et al., 2024; 2025; Saha et al., 2024). Mixed-precision quantization instead assigns different bit-widths to layers/modules based on their sensitivity or quantization error (Dong et al., 2019; 2020; Wang et al., 2019; Huang et al., 2024; Lee et al., 2025).

Recent work also combines mixed precision with low-rank reconstruction (Ou et al., 2024). A typical approach is to first identify where quantization is most harmful and then recover the resulting errors. This composition, however, overlooks an important mismatch. Conventional mixed-precision criteria evaluate the quantized backbone W_q, whereas the recovered weight is represented as $W_q + AB$. The error $W - W_q$ is therefore only an intermediate quantity, while what remains after recovery is the residual error $W - (W_q + AB)$. Accordingly, precision decisions made before recovery remain reliable only when recovery has similar effectiveness across modules and bit-widths.

However, our analysis shows that *quantization-error recoverability varies substantially across both modules and bit-widths.* This heterogeneity changes where higher precision is most beneficial. For example, a module that appears highly sensitive before recovery may require less precision if most of its quantization error is recoverable. In contrast, a less sensitive module may benefit more from higher precision when its error persists after recovery.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This observation leads to a key principle of recovery-aware quantization. Recovery does not merely reduce quantization error; it can *reorder the precision priorities* of different module–bit choices. Precision should therefore be allocated according to *the error that survives recovery*, rather than the error that recovery can remove.

We propose **RampQ**, a **R**ecovery-**A**ware **M**ixed-**P**recision **Q**uantization framework for LLMs, which performs mixed-precision allocation in the recovered representation space. For each module j and bit-width b, RampQ constructs a bit-specific recovered representation $W_{q,j}^{(b)} + A_j^{(b)} B_j^{(b)}$, and measures the error remaining after recovery. RampQ then combines the post-recovery residual with module sensitivity to estimate the impact of each precision choice. Bit-widths are selected under a hybrid memory budget that accounts for both the quantized backbone and its low-rank correction. In this way, RampQ treats each module–bit pair as a complete recovered candidate rather than evaluating the quantized backbone alone.

The module-local criterion provides an efficient basis for allocation but does not capture all interactions among simultaneously compressed modules. RampQ thus performs a model-level refinement that starts from the recovery-aware allocation and searches neighboring complete-model assignments under the same memory budget. The refinement evaluates selected assignments using end-to-end perplexity and progressively updates the allocation to identify the best configuration. In this refinement stage, a lightweight ranking surrogate prioritizes promising assignments for evaluation, reducing the number of expensive full-model evaluations. The resulting search produces the final bit allocation while accounting for model-level effects that are not represented by the module-wise proxy.

We evaluate RampQ on seven LLMs with average bit budgets from 2.5 to 3.5. The results show that RampQ improves perplexity and downstream accuracy over mixed-precision and low-rank recovery baselines under comparable effective memory budgets. Compared with the best baseline at each setting, RampQ reduces perplexity on average by 8.1% on WikiText-2 and 5.7% on C4, and improves average zero-shot accuracy by 1.67 percentage points.

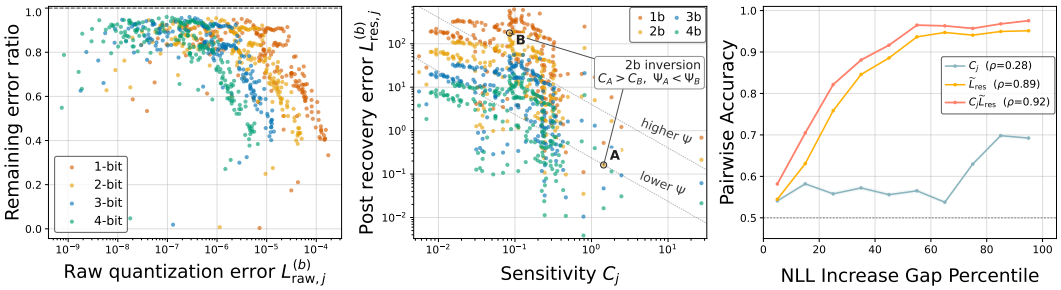
Our contributions are as follows.

- We identify *recovery-induced precision reprioritization* in low-bit LLM quantization. Quantization-error recoverability varies substantially across modules and bit-widths, such that low-rank recovery can change where higher precision is most beneficial.
- We propose **RampQ**, which performs mixed-precision allocation in the recovered representation space. RampQ constructs bit-specific $W_q + AB$ representations, evaluates each precision choice, and selects bit-widths under a memory budget that accounts for both the quantized backbone and its low-rank correction. RampQ further refines this allocation by searching complete-model assignments using end-to-end perplexity feedback.
- We evaluate RampQ across seven LLMs and multiple low-bit budgets. The results show that recovery-aware allocation consistently improves the quality–memory trade-off over mixed precision and low rank recovery baselines, and that simply adding low-rank recovery to a conventional mixed-precision allocation is insufficient.

2 Related Work

Post-training quantization. Post-training quantization (PTQ) reduces the memory footprint of LLMs. GPTQ (Frantar et al., 2022) uses approximate second-order information to reduce quantization error, while AWQ (Lin et al., 2024) protects salient weight channels identified by activation statistics. Other low-bit methods improve quantization using structured codebooks or transformations (Chee et al., 2023; Tseng et al., 2024). However, aggressive low-bit quantization can still produce large quantization errors, motivating methods that recover the lost model quality.

Low-rank error correction. Low-rank error correction represents quantized weights as $W_q + AB$, where the low-rank branch reduces the error introduced by quantization. LQER (Zhang et al., 2024) uses activation-induced scaling to improve low-rank reconstruction, while QERA (Zhang et al., 2025) develops an analytical framework for activation-aware quantization-error reconstruction. CALDERA (Saha et al., 2024) jointly optimizes a low-precision backbone and low-rank factors to improve reconstruction quality under aggressive low-bit quantization. These methods improve low-bit



(a) Recoverability variation (b) Recovery-aware priority (c) Priority alignment

Figure 1: Recovery-aware analysis on LLaMA-3.2-3B. (a) Variation in quantization-error recoverability. (b) Recovery-induced changes in module priority. (c) Pairwise ordering against measured NLL impact over 784 candidates (306,936 pairs); ρ denotes Spearman correlation and shading shows 95% bootstrap confidence intervals.

quantization quality for a chosen precision, but generally do not ask how recovery should change the choice of precision itself. They therefore do not address how bit-widths should be allocated when recoverability differs across precision choices.

Mixed-precision quantization. Mixed-precision quantization (MPQ) assigns different bit-widths across layers or modules based on their sensitivity to quantization. HAWQ and HAWQ-V2 (Dong et al., 2019; 2020) use Hessian-based criteria to estimate sensitivity and guide precision allocation. AMLQ (Ou et al., 2024) further combines mixed-precision quantization with low-rank error reconstruction, adapting bit-width and correction rank based on layer sensitivity. However, its allocation does not account for the error remaining after low-rank recovery. RampQ studies recovery not only as a mechanism for reducing quantization error, but as a transformation that changes the relative value of precision across candidate representations.

3 Recovery-Aware Analysis of Mixed-Precision Quantization

When the final weight is represented as $W_q + AB$, precision allocation should account for the error remaining after recovery rather than pre-recovery quantization error alone. To study the effect of recoverability separately from recovery capacity, we fix the recovery rank to $r = 32$ for all bit-widths in this section. This allows us to compare post-recovery error and precision priority under the same recovery capacity. In RampQ, however, we use bit-dependent recovery ranks selected based on the analysis in Appendix B.6.

We examine three aspects. First, we measure how quantization-error recoverability varies across layer-modules and bit-widths. Second, we examine how post-recovery error changes precision priority when combined with module sensitivity. Third, we compare the resulting recovery-aware priority with the measured NLL impact of different precision choices. Based on these analyses, RampQ uses post-recovery error together with module sensitivity for precision allocation.

3.1 Recovery is Heterogeneous across Precision Choices

Mixed-precision quantization typically evaluates each precision choice using module sensitivity or the quantization error measured before recovery (Dong et al., 2019; 2020; Huang et al., 2024; Lee et al., 2025). With low-rank recovery, however, a large quantization error does not necessarily rule out a low-bit choice if most of the error can be recovered. We therefore examine how recoverability varies across layer-modules and bit-widths.

For a layer-module j quantized to b bits, we define the quantization error, recovered weight, and remaining error as

$$E_j^{(b)} = W_j - W_{q,j}^{(b)}, \qquad \widetilde{W}_j^{(b,r)} = W_{q,j}^{(b)} + A_j^{(b,r)} B_j^{(b,r)}, \qquad R_j^{(b,r)} = W_j - \widetilde{W}_j^{(b,r)}. \tag{1}$$

Following the output-error approximation in Appendix A.2, we measure the activation-weighted errors before and after recovery as

$$L_{\mathrm{raw},j}^{(b)} = \left\| E_j^{(b)} D_j \right\|_F^2, \qquad L_{\mathrm{res},j}^{(b,r)} = \left\| R_j^{(b,r)} D_j \right\|_F^2, \qquad \gamma_j^{(b,r)} = \frac{L_{\mathrm{res},j}^{(b,r)}}{L_{\mathrm{raw},j}^{(b)} + \epsilon}. \tag{2}$$

Here, D_j is a diagonal activation weight estimated from calibration activations, and $\gamma_j^{(b,r)}$ represents the fraction of error remaining after recovery. We use $\epsilon = 10^{-12}$ throughout. A smaller $\gamma_j^{(b,r)}$ indicates higher recoverability.

Figure 1a shows that the remaining-error ratio varies widely across layer-modules and bit-widths. Even precision choices with similar raw quantization errors $L_{raw,j}^{(b)}$ can retain very different error fractions after the same-rank recovery, from below 0.5 to nearly 0.9. This variation indicates that raw quantization error does not directly determine the residual error of the recovered representation. In particular, a low-bit choice can remain competitive when a large fraction of its quantization error is recoverable. Thus, quantization error and recoverable error should be treated as distinct factors when evaluating precision choices.

3.2 Recovery Induces Precision Reprioritization

Recoverability describes how much quantization error is removed by low-rank recovery, but the effect of the remaining error also depends on module sensitivity. The same amount of remaining error can lead to different changes in model loss for modules with different sensitivities. We thus characterize each recovered precision choice using both module sensitivity and its post-recovery residual error.

We estimate module sensitivity using the empirical Fisher trace, computed as the calibration-averaged squared gradient norm (Zandonati et al., 2023)

$$C_j = \mathbb{E}_{s\sim\mathcal{D}_{\mathrm{cal}}} \left[\left\| \nabla_{W_j} \ell(s) \right\|_F^2 \right], \qquad \Psi_j^{(b,r)} = C_j L_{\mathrm{res},j}^{(b,r)}. \tag{3}$$

Here, C_j represents the sensitivity of module j, while $L_{\mathrm{res},j}^{(b,r)}$ measures the error remaining for the b-bit choice after recovery. Their product $\Psi_j^{(b,r)}$ increases when either the module is more sensitive or more error remains after recovery, and serves as the recovery-aware precision priority.

The effect of recovery on this priority can be expressed using the remaining-error ratio $\gamma_j^{(b,r)}$ defined in Eq. 2. Let the corresponding pre-recovery priority be

$$\Psi_{\mathrm{raw},j}^{(b)} = C_j L_{\mathrm{raw},j}^{(b)}, \qquad \Psi_j^{(b,r)} \approx \gamma_j^{(b,r)} \Psi_{\mathrm{raw},j}^{(b)}. \tag{4}$$

This relation exposes the key effect of recovery on mixed-precision allocation. If $\gamma_j^{(b,r)}$ were similar across module–bit choices, recovery would scale their priorities by similar amounts and largely preserve the pre-recovery ordering. However, as discussed in Section 3.1, recoverability varies substantially across both modules and bit-widths. Recovery therefore rescales different precision choices by different amounts and can change their relative ordering.

Figure 1b illustrates how the two terms affect the priority. Each point represents a module–bit choice, positioned by its module sensitivity C_j and post-recovery error $L_{\mathrm{res},j}^{(b)}$. The dashed lines indicate equal values of the recovery-aware priority $\Psi_j^{(b)} = C_j L_{\mathrm{res},j}^{(b)}$. A module with higher sensitivity does not necessarily receive higher priority when most of its quantization error is recoverable. For the two 2-bit layer-modules marked as A and B in the figure, A has higher sensitivity than B but leaves much less error after recovery. Accordingly, $C_A > C_B$ whereas $\Psi_A < \Psi_B$, reversing the ordering given by sensitivity alone. We next examine whether the recovery-aware ordering agrees with the measured NLL impact of each precision choice.

3.3 Residual-aware Priority Predicts Actual Model Impact

We next test whether the recovery-aware ordering induced by residual error and module sensitivity agrees with the measured NLL impact of individual module–bit choices. For RampQ, we normalize

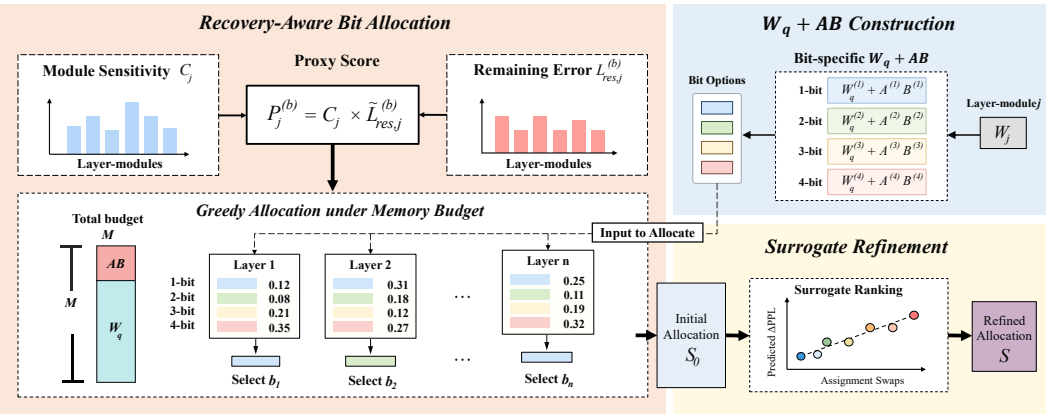


Figure 2: Overview of RampQ.

the post recovery error by a reference precision to reduce scale differences across modules.

$$P_j^{(b)} = C_j \frac{L_{\mathrm{res},j}^{(b)}}{L_{\mathrm{res},j}^{(b_{\mathrm{ref}})} + \epsilon}, \qquad b_{\mathrm{ref}} = 4. \tag{5}$$

The normalization compares each bit width with the 4 bit reference within the same module.

For each precision choice (j, b), we replace only module j in the full precision model with its bit specific $W_q + AB$ representation. We measure the NLL increase using an independent evaluation set of 32 sequences, each with a length of 2048. The measured increase represents the effect of the precision choice on the full model. We then compare pairs of precision choices and examine whether module sensitivity C_j, the normalized post recovery error, and $P_j^{(b)}$ assign higher priority to the choice with the larger NLL increase. The pairs are ordered by the difference between their measured NLL increases, and the difference is expressed as a percentile. For each percentile, we report the fraction of pairs correctly ordered by each signal.

Figure 1c compares the three signals with the measured NLL impact. Post recovery error agrees with the measured NLL ordering more often than module sensitivity alone. Using both post recovery error and module sensitivity provides the highest agreement among the three signals. This alignment supports the recovery-aware priority used by RampQ. Appendix A.4 further provides a first-order motivation for combining module sensitivity with post-recovery residual error.

4 RampQ: Recovery-Aware Mixed-Precision Quantization

Figure 2 shows an overview of RampQ. RampQ consists of three stages. First, for each layer module and bit width, RampQ constructs a $W_q + AB$ representation by optimizing the quantized backbone and low rank correction. Second, RampQ assigns bit widths under a hybrid memory budget using recovery aware priority. Third, RampQ refines the initial assignment using selective end-to-end perplexity evaluations and a lightweight ranking surrogate.

4.1 Bit Specific $W_q + AB$ Construction

RampQ allocates precision using the final $W_q + AB$ representation. For each layer module j and bit width $b \in \mathcal{B}$, we first construct

$$\widetilde{W}_j^{(b)} = W_{q,j}^{(b)} + A_j^{(b)} B_j^{(b)}. \tag{6}$$

We construct this representation by minimizing the activation weighted reconstruction error

$$\min_{\substack{W_{q,j}^{(b)} \in \mathcal{Q}_b \\ \mathrm{rank}(A_j^{(b)} B_j^{(b)}) \le r_b}} \left\| \left(W_j - W_{q,j}^{(b)} - A_j^{(b)} B_j^{(b)} \right) D_j \right\|_F^2, \tag{7}$$

where $\mathcal{Q}_b$ denotes the set of weights quantized to b bits.

We initialize the backbone using activation aware quantization. For the low rank correction, the limited rank makes it difficult to recover all remaining errors equally. We retain the activation weights of the top k input dimensions and scale the remaining weights by $\rho < 1$, producing $\widetilde{D}_j$. This weighting gives more emphasis to the top k dimensions during the low rank initialization. We then approximately optimize Eq. 7 by alternating backbone requantization and low rank refitting. Appendix B.1 describes the backbone quantization, low rank initialization, and alternating reconstruction procedure in detail. Each bit width uses a recovery rank r_b. We set $r_b = 128$ for 1 and 2 bits, $r_b = 32$ for 3 bits, and $r_b = 16$ for 4 bits. Appendix B.6 presents the residual spectrum analysis and rank selection results.

4.2 Recovery Aware Allocation under a Hybrid Budget

After constructing the bit-specific recovered representations, RampQ performs allocation over the resulting set of recovered candidates. Each module–bit choice is represented by its complete $W_{q,j}^{(b)} + A_j^{(b)} B_j^{(b)}$ representation, along with its recovery-aware priority $P_j^{(b)}$ and storage cost. For module j and bit-width b, its storage cost is

$$M_j^{(b)} = M\left(W_{q,j}^{(b)}\right) + M\left(A_j^{(b)}, B_j^{(b)}\right), \tag{8}$$

accounting for both quantized backbone and low-rank correction. We formulate the allocation problem as

$$\min_{\{b_j\}} \sum_j P_j^{(b_j)} \quad \text{s.t.} \quad \sum_j M_j^{(b_j)} \le \mathcal{M}, \qquad b_j \in \mathcal{B}, \tag{9}$$

where $\mathcal{M}$ is the total memory budget for the $W_q + AB$ representations of the quantized linear weights. We report the effective average bit-width as $8 \sum_j M_j^{(b_j)} / \sum_j |W_j|$. This value includes the storage of both W_q and AB and excludes shared nonquantized parameters.

We use a simple greedy procedure to obtain a feasible solution to Eq. 9. The procedure applies adjacent bit-width updates from multiple uniform bit initializations and prioritizes changes according to their proxy-cost trade-off. For a change from the current precision b to the next lower precision b^-, we compute the increase in proxy score and the memory saving as

$$\Delta P_j^{\downarrow} = P_j^{(b^-)} - P_j^{(b)}, \qquad \Delta M_j^{\downarrow} = M_j^{(b)} - M_j^{(b^-)}. \tag{10}$$

If the current assignment exceeds the memory budget, RampQ selects the downgrade with the smallest $\Delta P_j^{\downarrow} / \Delta M_j^{\downarrow}$. After the assignment satisfies the budget, the remaining memory is used for adjacent upgrades $b \to b^+$, selected by the largest proxy reduction per additional memory,

$$\frac{\Delta P_j^{\uparrow}}{\Delta M_j^{\uparrow}} = \frac{P_j^{(b)} - P_j^{(b^+)}}{M_j^{(b^+)} - M_j^{(b)}}. \tag{11}$$

We keep the feasible assignment with the lowest total proxy score. Appendix C.1 describes the initialization and greedy allocation procedure in detail.

We refer to the method up to this stage as **RampQ-G**. Because Eq. 9 uses a module-local proxy, the resulting assignment does not explicitly capture interactions among simultaneously compressed modules. Section 4.3 therefore refines this initial allocation using end-to-end perplexity feedback.

4.3 Model-Level Refinement for Cross-Module Effects

To account for cross-module interactions, we refine the initial RampQ-G assignment using end-to-end perplexity feedback. From this assignment, RampQ searches neighboring complete-model assignments under the same memory budget and evaluates selected assignments on the WikiText-2 validation set. Using the measured perplexity, we update the search and determine the final allocation.

Evaluating every neighboring assignment with full-model PPL is computationally expensive. We therefore use a lightweight ranking surrogate trained for pairwise ranking (Hu et al., 2021; Tolstikhin et al., 2021), using assignments whose PPL has already been measured, to prioritize promising

Table 1: WikiText-2 test perplexity across models and average-bit budgets. Lower is better.

Method	Bits	LLaMA2		LLaMA3		Qwen3		Mistral
		7B	13B	3.2-3B	3.1-8B	8B	14B	7B
FP16	16.0	5.48	4.90	7.81	6.24	9.73	8.64	5.32
CALDERA	2.4	6.71	5.79	10.66	8.84	11.78	12.91[1]	6.67
LQER	2.4	71.97	21.73	1.19e4	1.27e4	1.20e3	61.01	86.93
QERA	2.4	68.02	20.39	9.93e3	9.32e3	8.66e2	61.01	66.89
AMLQ[2]	2.5	7.68	6.34	–	–	–	–	–
AMQ	2.5	9.24	6.88	19.04	17.36	13.78	12.00	8.34
RampQ-G	2.5	6.49	5.68	11.26	8.76	12.50	10.21	6.34
RampQ	**2.5**	**6.30**	**5.54**	**10.53**	**8.35**	**10.76**	**9.56**	**6.16**
GPTQ	3.0	6.40	5.56	12.83	8.25	10.98	9.54	5.92
AWQ	3.0	6.26	5.37	10.58	8.34	11.38	9.90	6.02
AMQ	3.0	6.83	5.68	11.40	9.44	11.21	9.70	6.46
RampQ-G	3.0	5.90	5.23	9.30	7.45	10.64	9.28	5.71
RampQ	**3.0**	**5.84**	**5.17**	**9.10**	**7.32**	**9.93**	**8.85**	**5.66**
AMLQ[2]	3.3	5.82	5.13	–	–	–	–	–
LQER	3.4	6.89	5.63	12.05	9.31	12.16	9.93	6.17
QERA	3.4	6.63	5.62	12.05	9.27	12.13	9.92	6.16
AMQ	3.5	5.95	5.20	10.48	8.48	10.29	9.21	5.69
RampQ-G	3.5	5.73	5.09	8.69	6.97	10.21	8.97	5.53
RampQ	**3.5**	**5.65**	**5.04**	**8.55**	**6.88**	**9.79**	**8.73**	**5.51**

assignments for evaluation. It determines which of two assignments is expected to have lower perplexity.

At each generation, we generate neighboring assignments that satisfy the memory budget and rank them using the surrogate. We evaluate validation perplexity for the top-ranked assignments and a small exploration set. Only measured perplexity is used to update the beam and select the final assignment; the surrogate only reduces full-model evaluations. Appendix D details neighbor generation, budget control, beam updates, stopping criteria, surrogate training, and evaluation cost.

5 Experiments

5.1 Experimental Setup

We evaluate RampQ on LLaMA2-7B/13B (Touvron et al., 2023), LLaMA3.2-3B/3.1-8B (Grattafiori et al., 2024), Qwen3-8B/14B (Yang et al., 2025), and Mistral-7B (Jiang et al., 2023). We compare RampQ with AMQ (Lee et al., 2025) for mixed precision quantization and CALDERA (Saha et al., 2024), LQER (Zhang et al., 2024), and QERA (Zhang et al., 2025) for low rank recovery. GPTQ (Frantar et al., 2022) and AWQ (Lin et al., 2024) are also reported as fixed bit PTQ references where applicable. We compare the methods at matched or closest effective memory budgets, including the storage of low rank factors. For AMLQ (Ou et al., 2024), we report published WikiText-2 results because no official implementation was available. As these results use the original calibration setup and cover only a subset of models and budgets, we include them for reference but exclude them from aggregate comparisons. Following the original configurations, CALDERA uses rank 256 factors quantized to 4 bits, while LQER and QERA use rank 64 FP16 factors with MXINT backbones.

We use 64 SlimPajama-6B (Soboleva et al., 2023) calibration samples with a sequence length of 2048. We use lm evaluation harness (Gao et al., 2024) for all language model evaluations. We report perplexity (PPL) with a sequence length of 2048 on the WikiText-2 test split (Merity et al., 2016), C4 (Raffel et al., 2020), and PTB (Marcus et al., 1993). We also report zero shot accuracy on ARC e and ARC c (Clark et al., 2018), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), and BoolQ (Clark et al., 2019), together with 5 shot accuracy on MMLU (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021).

[1] Kronecker transform is used for Qwen3-14B because Hadamard is unavailable.

[2] AMLQ results are taken from the original paper (Ou et al., 2024).

Table 2: Performance under different average bit settings. C4 is lower-is-better, while Avg., MMLU, and GSM8K are higher-is-better.

		C4 Perplexity / Zero-shot Accuracy								Knowledge & Reasoning Accuracy			
		LLaMA2-7B		LLaMA3.1-8B		Qwen3-8B		Qwen3-14B		Mistral-7B		LLaMA2-13B	
Method	**Bits**	**C4**	**Avg.**	**C4**	**Avg.**	**C4**	**Avg.**	**C4**	**Avg.**	**MMLU**	**GSM8K**	**MMLU**	**GSM8K**
FP16	16	7.26	70.41	6.24	75.05	15.46	74.09	13.81	77.33	62.40	36.32	55.22	23.05
CALDERA	2.4	9.01	60.31	13.98	69.03	18.15	69.52	18.65	72.85	54.67	16.07	48.41	3.11
LQER	2.4	62.95	44.04	6.54e3	36.53	7.17e2	39.93	67.80	48.21	25.07	1.44	24.55	1.74
QERA	2.4	57.21	44.97	6.27e3	36.44	8.15e2	40.74	62.38	48.18	24.85	1.52	24.55	1.52
AMQ	2.5	12.37	60.56	26.32	60.30	20.65	67.93	18.30	61.43	45.41	3.49	43.80	5.69
RampQ-G	2.5	8.71	66.07	13.59	70.14	19.02	69.60	16.01	74.36	54.41	18.65	50.69	11.14
RampQ	**2.5**	**8.56**	**66.47**	**13.21**	**70.50**	**17.49**	**70.61**	**15.85**	**74.65**	**54.79**	**19.94**	**50.77**	**13.87**
GPTQ	3.0	8.59	67.26	12.91	68.53	17.03	71.06	15.04	74.56	56.40	24.64	45.67	7.05
AWQ	3.0	8.35	67.55	13.19	70.92	17.50	70.04	15.50	73.90	56.31	18.80	51.57	15.54
AMQ	3.0	9.03	65.59	14.68	69.39	17.06	71.53	15.32	74.61	53.57	16.00	49.62	13.50
RampQ-G	3.0	7.86	67.92	11.66	72.58	16.73	71.92	14.96	76.17	58.46	25.47	53.16	**17.13**
RampQ	**3.0**	**7.84**	**68.03**	**11.49**	**73.24**	**16.26**	**72.65**	**14.44**	**76.23**	**59.31**	**27.14**	**53.49**	17.06
LQER	3.4	9.37	66.65	14.73	67.65	18.86	67.77	15.63	73.70	55.68	22.06	49.54	14.10
QERA	3.4	8.99	66.68	14.70	67.73	18.84	67.72	15.62	73.45	55.56	22.29	49.64	13.65
AMQ	3.5	7.90	68.41	11.56	71.58	15.98	72.94	14.60	75.98	**61.12**	29.64	52.32	18.57
RampQ-G	3.5	7.62	68.43	10.87	**74.28**	16.03	73.08	14.25	76.38	60.70	29.72	54.71	18.80
RampQ	**3.5**	**7.58**	**68.96**	**10.74**	74.23	**15.84**	**73.28**	**14.19**	**76.68**	60.72	**30.40**	**54.86**	**19.64**

5.2 Main Results

Tables 1 and 2 summarize the main results. Across seven LLMs and average bit budgets from 2.5 to 3.5, RampQ improves both PPL and downstream accuracy over mixed-precision and low rank recovery baselines. Compared with the best baseline at each setting, RampQ reduces PPL by 8.1% on WikiText-2 and 5.7% on C4 on average, while improving zero-shot accuracy by 1.67 percentage points. The gains persist under aggressive memory budgets, where the fixed rank-based methods show large PPL increases. RampQ-G already improves the initial assignment using recovery aware priority, and the refinement stage further reduces PPL using full model feedback.

5.3 Effectiveness of Recovery Aware Allocation

Recovery aware allocation. Table 3 compares RampQ with AMQ after adding the same low rank recovery scheme. AMQ+AB uses the original AMQ predictor and search procedure to assign bit widths under the complete $W_q + AB$ memory budget. After allocation, we keep W_q fixed and fit the low rank correction using the same bit dependent ranks and 4 bit factors as RampQ. AMQ+AB therefore has comparable recovery capacity, but its precision assignment does not use the error measured after recovery. AMQ+AB does not consistently improve AMQ, especially at the 2.5 bit budget. RampQ-G instead uses post recovery error during precision allocation and achieves lower PPL across all settings in the table. RampQ further reduces PPL using full model feedback during the refinement stage. The comparison indicates that adding low rank recovery after allocation is not sufficient when the recoverability of different precision choices varies.

Table 3: C4 perplexity comparison under matched memory budgets. Lower is better.

Model	**Avg. Bits**	**AMQ**	**AMQ+AB**	**RampQ-G**	**RampQ**
LLaMA3.2-3B	2.5	21.37	25.19	16.69	**15.92**
	3.0	15.08	14.33	13.94	**13.68**
	3.5	13.58	13.63	12.92	**12.70**
LLaMA3.1-8B	2.5	26.32	36.90	13.59	**13.21**
	3.0	14.68	14.72	11.66	**11.49**
	3.5	11.56	11.34	10.87	**10.74**

Recovery aware proxy. Table 4 compares the signals used for greedy allocation. Module sensitivity C_j measures the importance of each module but does not distinguish different bit widths. Combining sensitivity with raw quantization error, $C_j L_{\text{raw}}$, adds precision dependent error information but does not reflect the error remaining after recovery. Using $\widetilde{L}_{\text{res}}$ directly measures the remaining error but

Table 4: C4 perplexity of greedy allocations using different proxy signals under matched hybrid memory budgets. Lower is better.

Bits	LLaMA2-13B				Qwen3-14B			
	$\mathbf{C_j}$	$\mathbf{C_j L_{raw}}$	$\widetilde{\mathbf{L}}_{\mathbf{res}}$	$\mathbf{C_j}\widetilde{\mathbf{L}}_{\mathbf{res}}$	$\mathbf{C_j}$	$\mathbf{C_j L_{raw}}$	$\widetilde{\mathbf{L}}_{\mathbf{res}}$	$\mathbf{C_j}\widetilde{\mathbf{L}}_{\mathbf{res}}$
2.5	243.70	*7.86*	8.20	**7.81**	699.15	218.45	*16.23*	**16.01**
3.0	7.67	7.49	*7.20*	**7.15**	279.00	70.84	*15.85*	**14.96**
3.5	7.75	7.26	*7.04*	**6.98**	49.71	27.16	*14.28*	**14.25**

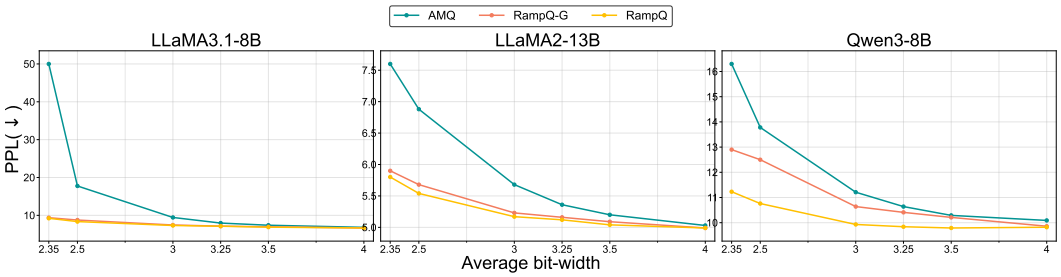


Figure 3: Perplexity comparison across average bit settings. Lower is better.

does not include module sensitivity. The combined score $C_j \widetilde{L}_{\text{res}}$ gives the lowest PPL for both models at all evaluated budgets. The results agree with the priority analysis in Section 3.3 and support using both module sensitivity and post recovery error for precision allocation.

5.4 Perplexity–Bit Trade-off

Figure 3 compares PPL across different average bit budgets on three representative models. RampQ achieves lower or comparable PPL across the evaluated budgets with a larger gain at tighter budgets where precision choices have a greater effect on the error remaining after recovery.

5.5 Inference Throughput and Memory

We also evaluate the inference cost of RampQ on a single NVIDIA RTX 3090 GPU. Figure 4 compares throughput and peak GPU memory. RampQ executes $W_q x$ using FLUTE quantized GEMM kernels (Guo et al., 2024) and computes the low rank update $A(Bx)$ using a separate optimized kernel path. Although the additional recovery branch makes RampQ slower than the weight only AMQ path, RampQ maintains higher throughput than CALDERA while using similar peak GPU memory to AMQ. Appendix E.2 reports the measurement settings and latency breakdown.

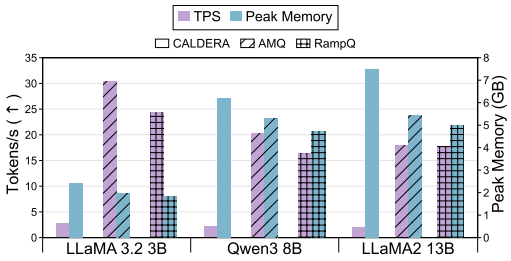


Figure 4: End to end generation throughput and peak GPU memory measured with batch size 1, a 64 token prompt, and 128 generated tokens.

6 Conclusions

We introduced RampQ, a recovery-aware mixed-precision quantization framework that allocates precision in the recovered representation space. Our analysis shows that heterogeneous recoverability across modules and bit-widths can reorder where higher precision is most beneficial. RampQ accounts for this effect by evaluating complete $W_q + AB$ representations using module sensitivity and post-recovery residual error under their joint memory cost. Across seven LLMs, RampQ reduces perplexity over the best baseline by 8.1% on WikiText-2 and 5.7% on C4 on average, while improving average zero-shot accuracy by 1.67 percentage points.

AI USE STATEMENT

We used generative AI tools for writing polish and literature retrieval and discovery. After completing author written drafts, we used a large language model to assist with proofreading and language editing, including grammar, spelling, punctuation, clarity, and consistency. The model was not used to generate new research claims, mathematical arguments, experimental results, or interpretations. We also used a large language model to assist with literature discovery by suggesting alternative search terms and identifying potentially relevant related work. All suggested references were manually verified against the original sources before inclusion.

We reviewed all AI assisted work and take responsibility for the final content of this paper.

REFERENCES

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. 2023. URL https://openreview.net/forum?id=xrk9g5vcXR.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18518–18529. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d77c703536718b95308130ff2e5cf9ee-Paper.pdf.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4 edition, 2013.

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, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru,

Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik

Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Han Guo, William Brandon, Radostin Cholakov, Jonathan Ragan-Kelley, Eric Xing, and Yoon Kim. Fast matrix multiplications for lookup table-quantized llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12419–12433, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Chi Hu, Chenglong Wang, Xiangnan Ma, Xia Meng, Yinqiao Li, Tong Xiao, Jingbo Zhu, and Changliang Li. RankNAS: Efficient neural architecture search by pairwise ranking. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2469–2480, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.191. URL https://aclanthology.org/2021.emnlp-main.191/.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Xianglong Liu, Luca Benini, Michele Magno, and Xiaojuan Qi. Slim-llm: Salience-driven mixed-precision quantization for large language models, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Sangjun Lee, Seung-taek Woo, Jun-gyu Jin, Changhun Lee, and Eunhyeok Park. AMQ: Enabling AutoML for mixed-precision weight-only quantization of large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35532–35550,

Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1799. URL https://aclanthology.org/2025.emnlp-main.1799/.

Shiyao Li, Xuefei Ning, Ke Hong, Tengxuan Liu, Luning Wang, Xiuhong Li, Kai Zhong, Guohao Dai, Huazhong Yang, and Yu Wang. LLM-MQ: Mixed-precision quantization for efficient llm deployment. In *The Efficient Natural Language and Speech Processing Workshop with NeurIPS*, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys*, 2024.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Lin Ou, Jinpeng Xia, Yuewei Zhang, Chuzhan Hao, and Hao Henry Wang. Adaptive quantization error reconstruction for LLMs with mixed precision. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=T5pGDydMkS.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=lkx3OpcqSZ.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP architecture for vision. In *Thirty-fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=EI2KOXKdnP.

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, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP$\#$: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=9BrydUVcoe.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ben Zandonati, Adrian Alan Pol, Maurizio Pierini, Olya Sirkin, and Tal Kopetz. FIT: A metric for model sensitivity. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PDG4-Y3aboN.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Cheng Zhang, Jianyi Cheng, George A. Constantinides, and Yiren Zhao. Lqer: low-rank quantization error reconstruction for llms. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Cheng Zhang, Jeffrey T. H. Wong, Can Xiao, George Anthony Constantinides, and Yiren Zhao. QERA: an analytical framework for quantization error reconstruction. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=LB5cKhgOTu.

Table 5: Notation used in Appendix A.

Notation	Definition
j	Layer-module index
b	Candidate bit-width
r	Low-rank correction rank
W_j	Full-precision weight of module j
$W_{q,j}^{(b)}$	b-bit quantized weight of module j
$E_j^{(b)} = W_j - W_{q,j}^{(b)}$	Quantization error before recovery
$A_j^{(b,r)},\ B_j^{(b,r)}$	Rank-r low-rank correction factors
$R_j^{(b,r)} = W_j - W_{q,j}^{(b)} - A_j^{(b,r)} B_j^{(b,r)}$	Remaining error after recovery
x	Input activation
$\Sigma_{x,j} = \mathbb{E}_x[xx^\top]$	Second-moment matrix of input activations
$D_j = \mathbf{diag}\left(\sqrt{\mathbb{E}[x_1^2]+\epsilon}, \ldots, \sqrt{\mathbb{E}[x_d^2]+\epsilon}\right)$	Diagonal activation weight
$L_{\mathbf{raw},j}^{(b)} = \|E_j^{(b)} D_j\|_F^2$	Activation-weighted quantization error
$L_{\mathbf{res},j}^{(b,r)} = \|R_j^{(b,r)} D_j\|_F^2$	Activation-weighted remaining error
$\gamma_j^{(b,r)} = \frac{L_{\mathbf{res},j}^{(b,r)}}{L_{\mathbf{raw},j}^{(b)} + \epsilon}$	Remaining-error ratio
$\widetilde{D}_j$	Saliency-reweighted activation weight
$\overline{C}_j$	Activation conditioned sensitivity in the first order analysis
C_j	Gradient based module sensitivity used by RampQ
$\widetilde{L}_{\mathbf{res},j}^{(b)} = \frac{L_{\mathbf{res},j}^{(b)}}{L_{\mathbf{res},j}^{(4)} + \epsilon}$	Reference-normalized remaining error
$P_j^{(b)} = C_j \widetilde{L}_{\mathbf{res},j}^{(b)}$	Recovery-aware allocation proxy

APPENDIX

- Section A. Theoretical motivation for the objectives used in RampQ.
- Section B. Implementation details and additional analysis of bit-specific $W_q + AB$ construction.
- Section C. Additional analysis of recovery-aware mixed-precision allocation.
- Section D. Implementation details and additional analysis of surrogate-guided refinement.
- Section E. Inference implementation and runtime measurements.
- Section F. Additional experimental results across models and evaluation tasks.

A THEORETICAL MOTIVATION

A.1 NOTATION

Table 5 summarizes the notation used in this appendix. We omit the module index j when no ambiguity arises.

A.2 Diagonal Activation Weighting for Output Error Approximation

We justify the activation weighted error measure used throughout RampQ. For a module j quantized to b bits, we define the quantization error as

$$E_j^{(b)} = W_j - W_{q,j}^{(b)}. \tag{12}$$

For an input activation x, this error produces the output error $E_j^{(b)}x$. Its expected squared magnitude can be written as

$$\mathbb{E}_x\left[\left\|E_j^{(b)}x\right\|_2^2\right] = \mathbb{E}_x\left[x^\top {E_j^{(b)}}^\top E_j^{(b)}x\right] \tag{13}$$

$$= \mathbb{E}_x\left[\mathrm{Tr}\left({E_j^{(b)}}^\top E_j^{(b)}xx^\top\right)\right] \tag{14}$$

$$= \mathrm{Tr}\left({E_j^{(b)}}^\top E_j^{(b)}\Sigma_{x,j}\right), \tag{15}$$

where

$$\Sigma_{x,j} = \mathbb{E}_x\left[xx^\top\right] \tag{16}$$

is the second moment matrix of the calibration activations. The expected output error therefore depends on both the magnitude of the quantization error and its interaction with the activation statistics.

Direct use of the full second moment matrix requires storing a dense $d \times d$ matrix for every module. It must also be incorporated repeatedly into the reconstruction objectives evaluated at different bit widths. We use a lightweight approximation that retains only the channel wise second moments by replacing $\Sigma_{x,j}$ with its diagonal. We define

$$D_j = \mathrm{diag}\left(\sqrt{\mathbb{E}[x_1^2]+\epsilon}, \ldots, \sqrt{\mathbb{E}[x_d^2]+\epsilon}\right), \qquad \Sigma_{x,j} \approx D_j^2, \tag{17}$$

where $\epsilon > 0$ is a small stabilization constant. Substituting this approximation into Eq. 15 gives

$$\mathbb{E}_x\left[\left\|E_j^{(b)}x\right\|_2^2\right] \approx \left\|E_j^{(b)}D_j\right\|_F^2. \tag{18}$$

After applying a rank r low rank correction, the remaining error is

$$R_j^{(b,r)} = W_j - W_{q,j}^{(b)} - A_j^{(b,r)}B_j^{(b,r)}. \tag{19}$$

Here, $A_j^{(b,r)}B_j^{(b,r)}$ is a rank r approximation of the quantization error obtained from the leading singular directions of the activation weighted residual. Appendix A.3 describes the corresponding weighted SVD formulation and the construction of the low rank factors.

Using the same activation weighting, we measure the errors before and after recovery as

$$L_{\mathrm{raw},j}^{(b)} = \left\|E_j^{(b)}D_j\right\|_F^2, \qquad L_{\mathrm{res},j}^{(b,r)} = \left\|R_j^{(b,r)}D_j\right\|_F^2. \tag{20}$$

These quantities give lightweight approximations to the expected squared output errors before and after low rank recovery under the same calibration statistics. The diagonal approximation avoids storing and manipulating the full activation second moment matrix while retaining channel dependent activation magnitude.

A.3 Weighted Low Rank Reconstruction

Low rank reconstruction has been used to recover structured quantization error by approximating the error left by a quantized backbone (Zhang et al., 2024; 2025; Saha et al., 2024). RampQ uses the same idea with the activation weighting introduced in Appendix A.2. For clarity, we omit the module and bit width indices and write the quantization error as

$$E = W - W_q. \tag{21}$$

RampQ also uses a reweighted version of the diagonal activation weight, denoted by $\widetilde{D}$, to give more emphasis to input dimensions with higher activation importance during low rank reconstruction. Appendix B.1 describes its practical construction.

For a rank r correction AB, the error remaining after recovery is $E - AB$. We obtain the correction by solving

$$\min_{\mathrm{rank}(AB)\le r} \left\| (E - AB)\widetilde{D} \right\|_F^2 . \tag{22}$$

Let

$$Z = E\widetilde{D}, \qquad Y = AB\widetilde{D}. \tag{23}$$

Since $\widetilde{D}$ is positive diagonal and invertible, right multiplication by $\widetilde{D}$ preserves rank. Equation 22 is therefore equivalent to

$$\min_{\mathrm{rank}(Y)\le r} \|Z - Y\|_F^2 . \tag{24}$$

By the Eckart–Young–Mirsky theorem (Golub & Van Loan, 2013), the optimal rank r approximation of Z under the Frobenius norm is given by its truncated singular value decomposition. Let

$$Z_r = U_r \Sigma_r V_r^\top \tag{25}$$

denote the leading rank r approximation of $Z = E\widetilde{D}$. The optimal solution in the weighted space is

$$Y^\star = Z_r = (E\widetilde{D})_r. \tag{26}$$

Mapping the solution back to the original weight space gives

$$(AB)^\star = (E\widetilde{D})_r \widetilde{D}^{-1}. \tag{27}$$

One valid factorization of the optimal correction is

$$A^\star = U_r \Sigma_r^{1/2}, \qquad B^\star = \Sigma_r^{1/2} V_r^\top \widetilde{D}^{-1}. \tag{28}$$

Any factorization whose product equals Eq. 27 represents the same rank r correction.

The finite correction capacity is allocated according to the dominant singular directions of the weighted quantization error rather than those of the unweighted error. The choice of $\widetilde{D}$ therefore changes which error directions are prioritized under a fixed recovery rank.

Equation 27 characterizes the optimal rank r correction before low precision factor quantization. RampQ stores the resulting factors at low precision, so factor quantization can move the realized correction away from this optimum. This difference motivates the alternating backbone and correction refinement and the checkpoint selection described in Appendix B.2.

A.4 First Order Motivation for Recovery Aware Allocation

We use a first order approximation to motivate combining module sensitivity with the error remaining after recovery.

For a calibration sample s, let

$$G_{j,s} = \nabla_{W_j} \ell(s) \tag{29}$$

denote the gradient of the loss with respect to module j. For a $W_q + AB$ representation at precision b, the remaining error is $R_j^{(b,r)}$ defined in Eq. 19. The represented weight is

$$W_j - R_j^{(b,r)}. \tag{30}$$

Its first order change in loss is approximated by

$$\delta \ell_{j,s}^{(b,r)} \approx - \left\langle G_{j,s}, R_j^{(b,r)} \right\rangle_F . \tag{31}$$

Using the diagonal activation weight D_j defined in Eq. 17, the inner product can be rewritten as

$$\left\langle G_{j,s}, R_j^{(b,r)} \right\rangle_F = \left\langle G_{j,s} D_j^{-1}, R_j^{(b,r)} D_j \right\rangle_F . \tag{32}$$

Applying the Cauchy–Schwarz inequality gives

$$\left|\delta\ell_{j,s}^{(b,r)}\right|^2 \lesssim \left\|G_{j,s}D_j^{-1}\right\|_F^2 \left\|R_j^{(b,r)}D_j\right\|_F^2. \tag{33}$$

The second term is the activation weighted error remaining after recovery

$$L_{\mathrm{res},j}^{(b,r)} = \left\|R_j^{(b,r)}D_j\right\|_F^2. \tag{34}$$

Averaging the first term over calibration samples gives

$$\overline{C}_j = \mathbb{E}_s\left[\left\|G_{j,s}D_j^{-1}\right\|_F^2\right]. \tag{35}$$

We then obtain

$$\mathbb{E}_s\left[\left|\delta\ell_{j,s}^{(b,r)}\right|^2\right] \lesssim \overline{C}_j L_{\mathrm{res},j}^{(b,r)}. \tag{36}$$

This approximation suggests that the effect of a precision choice depends on both module sensitivity and the error that remains after recovery. We use this relation only to motivate the multiplicative structure of the allocation score.

Directly using $\overline{C}_j$ can be numerically unstable because small activation scales in D_j amplify individual gradient directions. We instead use the calibration averaged gradient energy

$$C_j = \mathbb{E}_s\left[\|G_{j,s}\|_F^2\right] \tag{37}$$

as a stable estimate of module sensitivity. Gradient based sensitivity measures have also been used as practical approximations for model response in prior sensitivity and mixed precision quantization studies (Zandonati et al., 2023; Lee et al., 2025).

The scale of the remaining error can also differ substantially across modules. For allocation across the model, we compare each precision choice with the 4 bit representation of the same module

$$\widetilde{L}_{\mathrm{res},j}^{(b)} = \frac{L_{\mathrm{res},j}^{(b)}}{L_{\mathrm{res},j}^{(b_{\mathrm{ref}})} + \epsilon}, \qquad b_{\mathrm{ref}} = 4. \tag{38}$$

This reference normalization reduces module dependent scale differences when precision choices are compared across the model. It is a practical normalization rather than a consequence of the first order approximation.

The resulting allocation proxy is

$$P_j^{(b)} = C_j \widetilde{L}_{\mathrm{res},j}^{(b)}. \tag{39}$$

The first order approximation motivates the dependence on module sensitivity and remaining error rather than the exact numerical form of the proxy. The practical sensitivity estimate and reference normalization are used for stable allocation across modules. Section 3 and Table 4 evaluate the resulting proxy empirically.

B Details and Analysis of Bit-Specific $W_q + AB$ Construction

This appendix describes the implementation and analysis of the bit specific $W_q + AB$ construction introduced in Section 4.1. We first present the initialization and alternating reconstruction procedure. We then study activation weighting, activation prioritized recovery, calibration stability, rank selection, and reconstruction cost. All bit specific representations are constructed offline and reused across different target memory budgets. The construction settings are fixed across models and target memory budgets, and the following analyses examine their behavior under representative settings.

B.1 Bit-Specific $W_q + AB$ Initialization

For each layer module j and bit width $b \in \mathcal{B}$, RampQ initializes

$$\widetilde{W}_j^{(b,0)} = W_{q,j}^{(b,0)} + A_j^{(b,0)} B_j^{(b,0)}. \tag{40}$$

The quantized backbone is initialized with activation aware quantization. The low rank correction is initialized to give more recovery capacity to error directions associated with activation important input dimensions.

Activation aware backbone initialization. The initial b bit backbone is constructed using group wise Lloyd–Max nonuniform quantization with group size 128. To reduce scale imbalance across input columns, we introduce a trainable column scaling vector α_j and define

$$S_{\alpha_j} = \mathrm{diag}(\alpha_j), \qquad \alpha_{j,c} > 0. \tag{41}$$

Before quantization, the weight is scaled and then mapped back to the original weight space

$$W_{q,j}^{(b,0)} = Q_b\left(W_j S_{\alpha_j}^{-1}\right) S_{\alpha_j}. \tag{42}$$

We optimize the scaling vector using the activation weighted quantization error

$$\min_{\alpha_j > 0} \left\| \left(W_j - Q_b(W_j S_{\alpha_j}^{-1}) S_{\alpha_j}\right) D_j \right\|_F^2. \tag{43}$$

Lloyd–Max quantization contains discrete assignments, so the quantization parameters and scaling vector are updated alternately. Given α_j, the assignments and centroids are recomputed for the scaled weight. With the assignments fixed, α_j is updated to reduce Eq. 43. The quantizer is then recomputed using the updated scaling. This procedure is used only to obtain the initial backbone $W_{q,j}^{(b,0)}$. Column scaling is not reapplied during the subsequent backbone and correction reconstruction.

The resulting quantization error is

$$E_j^{(b,0)} = W_j - W_{q,j}^{(b,0)}. \tag{44}$$

Activation prioritized low rank initialization. The finite recovery capacity is then directed toward error directions associated with activation important input dimensions. Let $\mathcal{S}_{j,k}$ denote the top k input dimensions ranked by $d_{j,c}^2$. We construct a reweighted activation matrix as

$$\widetilde{d}_{j,c} = \begin{cases} d_{j,c}, & c \in \mathcal{S}_{j,k}, \\ \rho d_{j,c}, & c \notin \mathcal{S}_{j,k}, \end{cases} \qquad \widetilde{D}_j = \mathrm{diag}(\widetilde{d}_j). \tag{45}$$

The initial rank r_b correction is obtained from

$$\min_{\mathrm{rank}(AB) \le r_b} \left\| \left(E_j^{(b,0)} - AB\right) \widetilde{D}_j \right\|_F^2. \tag{46}$$

As derived in Appendix A.3, the objective assigns the limited recovery rank to the dominant directions of the reweighted quantization error.

We set $\rho = 0.25$ and $k = \lfloor 0.25 r_b \rfloor$. The resulting low rank factors are stored at 4 bits using the same group wise Lloyd–Max nonuniform quantization scheme with group size 128. The recovery ranks are $r_b = 128$ for 1 and 2 bit representations, $r_b = 32$ for 3 bit representations, and $r_b = 16$ for 4 bit representations.

The initialized backbone and low rank correction are used as the starting point of the alternating reconstruction in Appendix B.2. The following sections examine the effects of activation reweighting, calibration size, and recovery rank.

B.2 Alternating Reconstruction

Starting from Appendix B.1, RampQ alternates between backbone requantization and low rank refitting. Updating either component changes the error that must be represented by the other.

Let $A_{q,j}^{(b,t)}$ and $B_{q,j}^{(b,t)}$ denote the stored p bit low rank factors at iteration t, where $p = 4$ in all experiments. The factors use the same group wise Lloyd–Max nonuniform quantizer as the backbone with group size 128.

Given the correction from the previous iteration, the target for backbone requantization is

$$\widehat{W}_j^{(b,t)} = W_j - A_{q,j}^{(b,t-1)} B_{q,j}^{(b,t-1)}. \tag{47}$$

We requantize this target using the activation weighted objective

$$W_{q,j}^{(b,t)} = \mathrm{Quant}_b^{D_j}\left(\widehat{W}_j^{(b,t)}\right), \qquad t \ge 1, \tag{48}$$

where $\mathrm{Quant}_b^{D_j}$ denotes group wise Lloyd–Max quantization that minimizes

$$\left\| \left(\widehat{W}_j^{(b,t)} - W_q \right) D_j \right\|_F^2 \tag{49}$$

over b bit quantized weights W_q. The column scaling used for initialization is not applied in this stage.

The new quantization error is

$$E_j^{(b,t)} = W_j - W_{q,j}^{(b,t)}. \tag{50}$$

A rank r_b correction is then fitted by solving

$$\min_{\mathrm{rank}(AB) \le r_b} \left\| \left(E_j^{(b,t)} - AB \right) \widetilde{D}_j \right\|_F^2, \tag{51}$$

after which the resulting factors are quantized to p bits. The remaining error after factor quantization is

$$R_j^{(b,t)} = W_j - W_{q,j}^{(b,t)} - A_{q,j}^{(b,t)} B_{q,j}^{(b,t)}. \tag{52}$$

We repeat the backbone and correction updates for $t = 1, \ldots, T$.

The weighted low rank solution in Appendix A.3 gives the optimal rank r_b correction before factor quantization. Since the factors are stored at low precision, quantization can move the realized correction away from this optimum. The alternating procedure is therefore not guaranteed to improve monotonically.

We evaluate the reconstruction objective after each iteration

$$L_{\mathrm{rec},j}^{(b,t)} = \left\| R_j^{(b,t)} \widetilde{D}_j \right\|_F^2, \tag{53}$$

and select the iterate with the smallest value

$$t_j^{\star(b)} = \arg \min_{0 \le t \le T} L_{\mathrm{rec},j}^{(b,t)}. \tag{54}$$

The selected iterate determines the final quantized backbone and low rank factors for each bit specific $W_q + AB$ representation. We denote them by $W_{q,j}^{(b)}$, $A_{q,j}^{(b)}$, and $B_{q,j}^{(b)}$. The remaining error becomes

$$R_j^{(b)} = W_j - W_{q,j}^{(b)} - A_{q,j}^{(b)} B_{q,j}^{(b)}, \tag{55}$$

and the post recovery error is evaluated with the common activation weight D_j

$$L_{\mathrm{res},j}^{(b)} = \left\| R_j^{(b)} D_j \right\|_F^2. \tag{56}$$

The two activation matrices have different roles. $\widetilde{D}_j$ controls low rank fitting and iterate selection so that more recovery capacity is assigned to activation important error directions. D_j provides a common error measure for comparing representations across bit widths during mixed precision allocation. The storage cost $M_j^{(b)}$ includes both the selected quantized backbone and its low rank factors.

Algorithm 1 summarizes the complete construction procedure.

B.3 Effect of Activation Weighting

Table 6 examines activation weighting in both backbone quantization and low rank reconstruction. Applying D to either component improves 2 bit performance, while using it in both stages gives the lowest perplexity. The result supports the activation weighted error approximation in Appendix A.2 and its use during $W_q + AB$ construction.

Figure 5 examines the corresponding quantization error spectrum. Without activation weighting, the singular energy is distributed over more directions. Using the activation weight concentrates a larger fraction of the error in the leading singular components. A correction with limited rank can consequently represent more of the activation relevant quantization error under the same rank budget.

The table and spectral analysis indicate that activation weighting improves the quantized backbone and produces an error structure that is better suited to low rank recovery.

Algorithm 1 Bit-Specific $W_q + AB$ Construction with Alternating Reconstruction

Require: Full-precision weight $\mathbf{W}_j$, bit-width b, recovery rank r_b, activation weight $\mathbf{D}_j$, reweighted activation matrix $\widetilde{\mathbf{D}}_j$, factor precision p, refinement iterations T

Ensure: Bit-specific representation $\left(\mathbf{W}_{q,j}^{(b)}, \mathbf{A}_{q,j}^{(b)}, \mathbf{B}_{q,j}^{(b)}, L_{\mathrm{res},j}^{(b)}, M_j^{(b)}\right)$

Initialization

Initialize activation-aware backbone: $\mathbf{W}_{q,j}^{(b,0)} \leftarrow \mathrm{Quant}_b^{\boldsymbol{\alpha}_j, \mathbf{D}_j}(\mathbf{W}_j)$ ▷ $\boldsymbol{\alpha}_j$ is optimized during initialization

Compute quantization error: $\mathbf{E}_j^{(b,0)} \leftarrow \mathbf{W}_j - \mathbf{W}_{q,j}^{(b,0)}$

Fit activation-prioritized low-rank correction: $(\mathbf{A}_j^{(b,0)}, \mathbf{B}_j^{(b,0)}) \leftarrow \mathrm{LowRank}_{r_b}^{\widetilde{\mathbf{D}}_j}(\mathbf{E}_j^{(b,0)})$

Quantize correction factors: $\mathbf{A}_{q,j}^{(b,0)} \leftarrow Q_p(\mathbf{A}_j^{(b,0)}),\ \mathbf{B}_{q,j}^{(b,0)} \leftarrow Q_p(\mathbf{B}_j^{(b,0)})$

Compute remaining error: $\mathbf{R}_j^{(b,0)} \leftarrow \mathbf{W}_j - \mathbf{W}_{q,j}^{(b,0)} - \mathbf{A}_{q,j}^{(b,0)}\mathbf{B}_{q,j}^{(b,0)}$

Evaluate reconstruction objective: $L_{\mathrm{rec},j}^{(b,0)} \leftarrow \left\|\mathbf{R}_j^{(b,0)}\widetilde{\mathbf{D}}_j\right\|_F^2$

Alternating Reconstruction

for $t = 1, \ldots, T$ **do**

  Form backbone target: $\widehat{\mathbf{W}}_j^{(b,t)} \leftarrow \mathbf{W}_j - \mathbf{A}_{q,j}^{(b,t-1)}\mathbf{B}_{q,j}^{(b,t-1)}$

  Requantize backbone: $\mathbf{W}_{q,j}^{(b,t)} \leftarrow \mathrm{Quant}_b^{\mathbf{D}_j}\left(\widehat{\mathbf{W}}_j^{(b,t)}\right)$

  Update quantization error: $\mathbf{E}_j^{(b,t)} \leftarrow \mathbf{W}_j - \mathbf{W}_{q,j}^{(b,t)}$

  Refit low-rank correction: $(\mathbf{A}_j^{(b,t)}, \mathbf{B}_j^{(b,t)}) \leftarrow \mathrm{LowRank}_{r_b}^{\widetilde{\mathbf{D}}_j}\left(\mathbf{E}_j^{(b,t)}\right)$

  Requantize correction factors: $\mathbf{A}_{q,j}^{(b,t)} \leftarrow Q_p(\mathbf{A}_j^{(b,t)}),\ \mathbf{B}_{q,j}^{(b,t)} \leftarrow Q_p(\mathbf{B}_j^{(b,t)})$

  Compute remaining error: $\mathbf{R}_j^{(b,t)} \leftarrow \mathbf{W}_j - \mathbf{W}_{q,j}^{(b,t)} - \mathbf{A}_{q,j}^{(b,t)}\mathbf{B}_{q,j}^{(b,t)}$

  Evaluate reconstruction objective: $L_{\mathrm{rec},j}^{(b,t)} \leftarrow \left\|\mathbf{R}_j^{(b,t)}\widetilde{\mathbf{D}}_j\right\|_F^2$

end for

Iterate Selection

$t_j^{\star(b)} \leftarrow \arg\min_{0 \le t \le T} L_{\mathrm{rec},j}^{(b,t)}$

$\mathbf{W}_{q,j}^{(b)} \leftarrow \mathbf{W}_{q,j}^{(b,t_j^{\star(b)})}$

$\mathbf{A}_{q,j}^{(b)} \leftarrow \mathbf{A}_{q,j}^{(b,t_j^{\star(b)})},\ \mathbf{B}_{q,j}^{(b)} \leftarrow \mathbf{B}_{q,j}^{(b,t_j^{\star(b)})}$

Final Evaluation

Compute final remaining error: $\mathbf{R}_j^{(b)} \leftarrow \mathbf{W}_j - \mathbf{W}_{q,j}^{(b)} - \mathbf{A}_{q,j}^{(b)}\mathbf{B}_{q,j}^{(b)}$

Measure post-recovery error: $L_{\mathrm{res},j}^{(b)} \leftarrow \left\|\mathbf{R}_j^{(b)}\mathbf{D}_j\right\|_F^2$

Compute hybrid storage cost: $M_j^{(b)} \leftarrow M(\mathbf{W}_{q,j}^{(b)}) + M(\mathbf{A}_{q,j}^{(b)}) + M(\mathbf{B}_{q,j}^{(b)})$

return $\left(\mathbf{W}_{q,j}^{(b)}, \mathbf{A}_{q,j}^{(b)}, \mathbf{B}_{q,j}^{(b)}, L_{\mathrm{res},j}^{(b)}, M_j^{(b)}\right)$

Table 6: Effect of activation weighting on 2-bit $W_q + AB$ construction on LLaMA3.1-8B WikiText-2. Lower is better.

Backbone	**Low-rank reconstruction**	
	Correction	**PPL ↓**
Unweighted W_q	None	5.18×10^4
	Unweighted AB	9.33×10^2
	D-weighted AB	17.29
D-weighted W_q	None	44.77
	Unweighted AB	19.84
	D-weighted AB	**16.56**

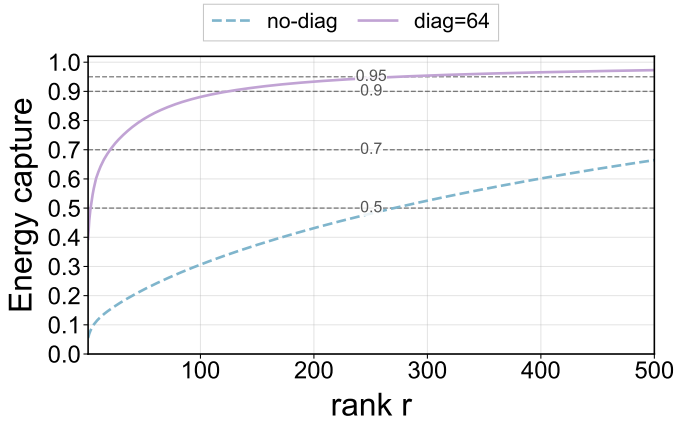
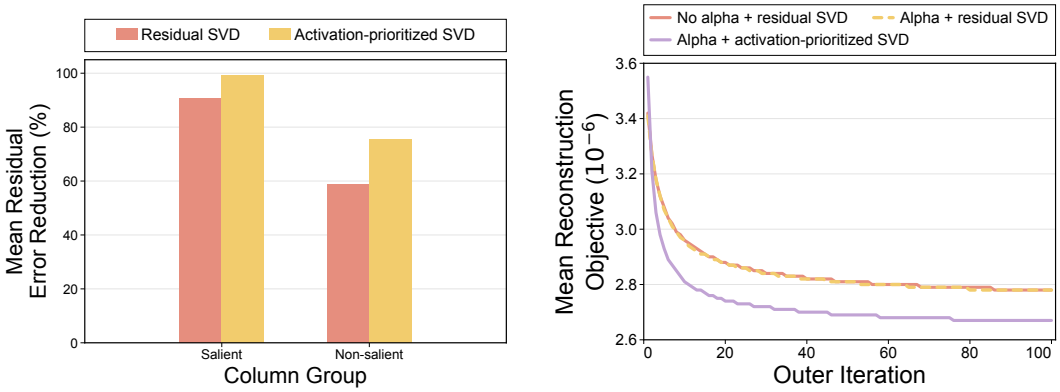


Figure 5: Singular energy capture of 2-bit quantization error on LLaMA3.2-3B using 64 calibration samples. Activation weighting concentrates a larger fraction of the error energy in the leading singular directions.



(a) Error reduction by activation group.

(b) Reconstruction objective during refinement.

Figure 6: Effect of trainable column scaling and top-k activation reweighting on 2-bit LLaMA2-7B $W_q + AB$ construction, averaged over 224 layer-module weights. (a) Error reduction for the top-k and remaining input dimensions. (b) Reconstruction objective during alternating refinement.

B.4 Effect of Column Scaling and Activation-Prioritized Recovery

Figure 6 studies the different roles of trainable column scaling and activation prioritized low rank reconstruction. Column scaling improves the initial low bit backbone. Activation reweighting determines how the available low rank capacity is distributed across quantization error directions.

Figure 6a compares error reduction for the top k activation important input dimensions and the remaining dimensions. Reweighting increases error reduction for the top k group while allocating less recovery capacity to the other inputs. The weighted objective therefore changes where the available rank is used rather than reducing all error directions uniformly.

Figure 6b reports the reconstruction objective during the subsequent alternating procedure. Trainable column scaling is used only for backbone initialization. The fixed matrix $\widetilde{D}_j$ continues to control low rank refitting throughout the remaining iterations. A better initial backbone is obtained from column scaling, while activation reweighting reduces the reconstruction objective by directing the correction toward activation important error directions.

These observations support the separate use of column scaling for backbone initialization and activation reweighting for low rank reconstruction.

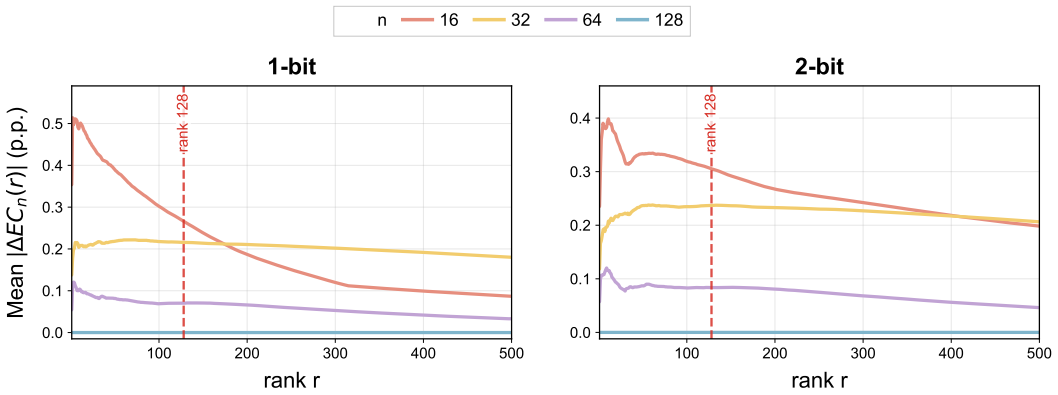


Figure 7: Calibration-size stability of activation-weighted quantization-error capture on LLaMA3.2-3B for 1-bit and 2-bit backbones.

B.5 Calibration Size and Activation-Reweighting Settings

The bit specific $W_q + AB$ construction depends on activation statistics estimated from calibration data and on the number of input dimensions emphasized during low rank reconstruction. We examine both settings.

Calibration size stability. Figure 7 evaluates the stability of the activation weighted quantization error structure on LLaMA3.2-3B with $n \in \{16, 32, 64, 128\}$ calibration samples. For each value of n, we construct the corresponding activation weight D_n and compare the singular energy capture of $(W - W_q)D_n$ with the result from 128 samples.

The difference decreases as more calibration samples are used. The estimate from 16 samples has the largest deviation, whereas the 32 and 64 sample results become increasingly close to the reference. At the recovery rank $r = 128$ used for the 1 and 2 bit representations, 64 samples already give a similar error capture profile to the 128 sample setting. We therefore use 64 calibration samples in all experiments.

Table 7: Effect of the number of activation-important dimensions on Qwen3-8B C4 perplexity. Lower is better.

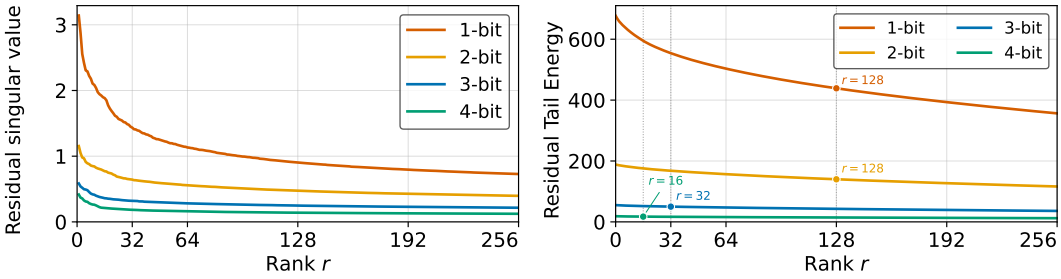
Bits	**Number of emphasized dimensions k**					
	4	**8**	**16**	**32**	**64**	**128**
1	9.88e3	9.02e3	4.32e3	**2.12e3**	3.59e3	4.02e3
2	21.23	21.15	21.50	**21.01**	21.35	21.66
3	16.70	**16.66**	16.79	16.88	–	–
4	**15.70**	15.73	15.73	–	–	–

Number of emphasized input dimensions. We also vary the number of activation important input dimensions that retain their original values in the reweighted matrix $\widetilde{D}_j$. Under Eq. 45, the top k dimensions ranked by $d_{j,c}^2$ retain the original activation weights. The other dimensions are scaled by ρ. The value of k controls how strongly the available recovery capacity is concentrated on activation important error directions.

Table 7 reports C4 perplexity on Qwen3-8B while varying k. We use the rank proportional rule

$$k = \lfloor 0.25 r_b \rfloor , \tag{57}$$

which gives $k = 32$, 32, 8, and 4 for the 1, 2, 3, and 4 bit representations, respectively. The results show that this setting performs well across the tested bit widths.



(a) Quantization-error singular-value spectrum

(b) Quantization-error tail energy

Figure 8: Spectral analysis of activation-weighted quantization errors across backbone bit-widths. (a) Singular-value spectra of $(W - W_q^{(b)})D$. (b) Tail energy $\sum_{i>r} \sigma_{b,i}^2$ after rank-r truncation. Markers indicate the selected ranks of 128, 128, 32, and 16 for 1-, 2-, 3-, and 4-bit representations, respectively.

The choice of k has a larger effect at 1 and 2 bits because the quantized backbone leaves more error for the low rank branch. Its effect becomes smaller as backbone precision increases. We use the same rank proportional rule for all bit specific representations.

B.6 Recovery Rank and Rank–Precision Trade-off

Recovery rank determines both correction capacity and the storage required by the low rank branch. Using a large rank at every bit width can add unnecessary cost to higher precision representations. A rank that is too small can leave substantial error under aggressive quantization. We use bit dependent recovery ranks to reflect the different quantization error structures across bit widths. We examine this design through the quantization error spectrum, a direct rank sweep, and a comparison under matched correction storage.

Quantization error structure across bit widths. Figure 8a shows the singular value spectrum of the activation weighted quantization error $(W - W_q^{(b)})D$. Lower bit backbones have larger singular values, which indicates more error to be represented by the low rank branch.

Figure 8b measures the error energy outside the leading rank r subspace

$$\mathcal{E}_{\text{tail}}^{(b)}(r) = \sum_{i>r} \sigma_{b,i}^2, \tag{58}$$

where $\sigma_{b,i}$ is the ith singular value at bit width b. For the same rank, the 1 and 2 bit representations retain more tail energy than the 3 and 4 bit representations. Aggressive quantization therefore requires a wider correction subspace, while higher precision backbones can use smaller ranks.

Direct rank sweep. Table 8 reports WikiText-2 perplexity for different correction ranks on Mistral-7B and LLaMA3.1-8B with factor precision fixed at 4 bits. Every setting uses the complete alternating reconstruction procedure in Appendix B.2, so the comparison isolates the effect of recovery rank under the same construction method.

Across both models, the 1 and 2 bit representations benefit substantially from a larger recovery rank. Reducing the rank from 128 to 64 increases perplexity considerably at both bit widths. This behavior is consistent with the higher tail energy in Figure 8 and supports using rank 128 for the 1 and 2 bit representations.

The gain from additional rank becomes smaller at higher backbone precision. At 3 bits, reducing the rank from 64 to 32 increases perplexity by only 0.08 on Mistral-7B and 0.03 on LLaMA3.1-8B, while rank 16 causes a larger degradation on both models. At 4 bits, the difference between ranks 64 and 16 is only 0.03 on Mistral-7B and 0.04 on LLaMA3.1-8B. We therefore use rank 32 at 3 bits and rank 16 at 4 bits.

Table 8: Effect of low-rank correction rank across backbone bit-widths on Mistral-7B and LLaMA3.1-8B with 4-bit low-rank factors. WikiText-2 perplexity is reported; lower is better. Bold values correspond to the selected rank for each bit-width.

Model	W_q **bits**	**Low-rank correction rank**				**Selected**
		128	**64**	**32**	**16**	
Mistral-7B	1-bit	**582.95**	1162.28	–	–	**128**
	2-bit	**7.43**	9.63	–	–	**128**
	3-bit	–	5.61	**5.69**	6.12	**32**
	4-bit	–	5.41	5.42	**5.44**	**16**
LLaMA3.1-8B	1-bit	**3999.93**	6536.16	–	–	**128**
	2-bit	**11.21**	15.18	–	–	**128**
	3-bit	–	7.43	**7.46**	7.78	**32**
	4-bit	–	6.62	6.65	**6.66**	**16**

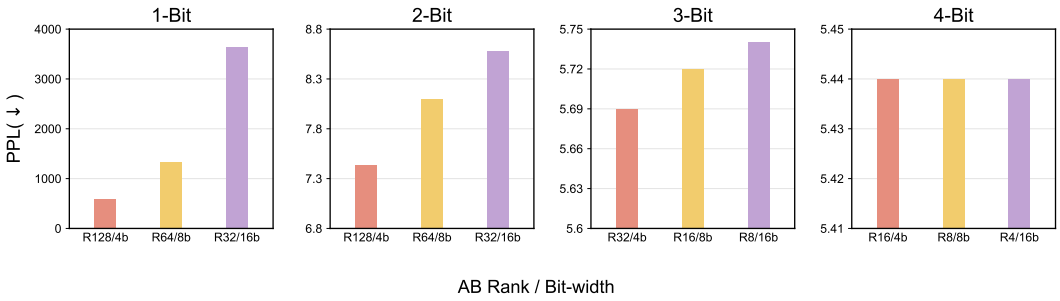


Figure 9: Storage-matched rank–precision trade-off on Mistral-7B WikiText-2, where R denotes the correction rank and b the backbone bit-width. Lower is better.

The resulting rank schedule is

$$r_b = \begin{cases} 128, & b \in \{1, 2\}, \\ 32, & b = 3, \\ 16, & b = 4. \end{cases} \tag{59}$$

We use this schedule across all models without model specific tuning. The schedule is fixed before mixed precision allocation, and the storage of its low rank factors is included in the hybrid cost $M_j^{(b)}$.

Rank precision trade off under matched correction storage. Figure 9 compares combinations of correction rank and factor precision under matched factor storage. Each configuration uses the same alternating reconstruction procedure in Appendix B.2 before perplexity is measured.

For 1 and 2 bit backbones, using a larger rank with 4 bit factors consistently gives lower perplexity than using a smaller rank with higher factor precision. Under aggressive quantization, a wider correction subspace is more useful than representing a narrower subspace with greater numerical precision.

At 3 and 4 bits, the difference between these choices becomes smaller because the backbone leaves less error and additional recovery rank provides limited improvement. The direct rank sweep and storage matched comparison support the use of 4 bit factors with ranks 128, 128, 32, and 16 for 1, 2, 3, and 4 bit representations, respectively.

Table 9: Convergence and quality–cost trade-off of 2-bit recovered candidate construction on LLaMA3.1-8B WikiText-2. Time is measured on a single NVIDIA RTX 3090 GPU.

Method	Reconstruction configuration				Performance	
	Iter.	Rank	AB bits	Avg. bits	PPL ↓	Time ↓
RampQ	Base	128	4	2.2	20.11	0.18 h
	3	128	4	2.2	15.41	0.32 h
	10	128	4	2.2	13.50	0.55 h
	50	128	4	2.2	12.19	1.75 h
	100	**128**	**4**	**2.2**	**11.21**	**3.20 h**
	200	128	4	2.2	11.19	6.05 h
CALDERA	50	256	4	2.4	8.84	39.9 h

B.7 Convergence and Construction Cost

Table 9 reports the convergence and computational cost of the alternating procedure in Appendix B.2. We evaluate 2 bit $W_q + AB$ construction on LLaMA3.1-8B with recovery rank 128 and 4 bit low rank factors while varying the number of refinement iterations.

Most of the improvement occurs during the early iterations. Relative to initialization, 10 iterations reduce WikiText-2 perplexity by 6.61. Extending the procedure to 100 iterations gives an additional reduction of 2.29. Increasing the budget from 100 to 200 iterations reduces perplexity by only 0.02, while wall clock time increases from 3.20 to 6.05 GPU hours. The improvement therefore largely saturates by $T = 100$, which we fix for all experiments. The resulting $W_q + AB$ representations are built offline and reused for different target memory budgets.

Table 9 also includes CALDERA (Saha et al., 2024) as a reference. CALDERA gives lower perplexity in this isolated 2 bit setting but requires 39.9 GPU hours. RampQ uses a lighter reconstruction procedure because representations must be prepared across multiple bit widths and layer modules before mixed precision allocation.

B.8 Disentangling Representation Construction and Recovery-Aware Allocation

Table 10: Controlled comparison of representation construction and allocation signals on LLaMA2-13B C4. Initialization uses the $W_q + AB$ representation at $t = 0$. All settings share the same memory budget and greedy allocation procedure. Lower is better.

Representation construction	Allocation signal	Average-bit budget		
		2.5	3.0	3.5
Initialization ($t = 0$)	Pre-recovery	8.33	8.10	8.02
	Recovery-aware	8.05	7.96	7.54
Alternating refinement	Pre-recovery	7.86	7.49	7.26
	Recovery-aware	7.81	7.15	6.98
Alternating refinement	Recovery-aware + surrogate	**7.56**	**7.12**	**6.87**

We separate the effects of representation construction and recovery aware allocation using a controlled 2×2 comparison. The initialization setting uses the $W_q + AB$ representation at $t = 0$, while alternating refinement selects the best iterate from $t = 0, \ldots, T$ with $T = 100$. Both settings use the same activation aware initialization, recovery ranks, and 4 bit low rank factors.

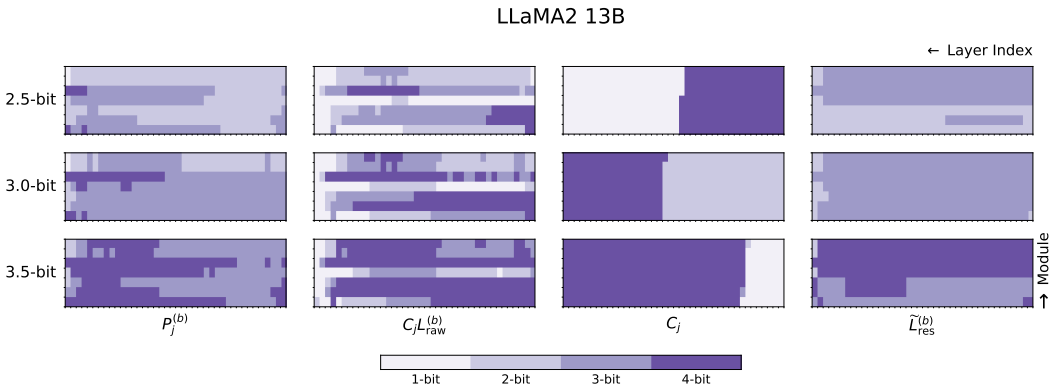


Figure 10: Layerwise bit allocation patterns on LLaMA2-13B. From left to right, recovery-aware, sensitivity-weighted raw-error, sensitivity-only, and remaining-error-only allocations are shown.

For each representation setting, we compare allocation using $C_j L_{\text{raw},j}^{(b)}$ with the recovery aware score $C_j \widetilde{L}_{\text{res},j}^{(b)}$. All other allocation settings are fixed, and surrogate refinement is disabled.

Table 10 reports C4 perplexity on LLaMA2-13B. Recovery aware allocation lowers perplexity at all budgets for both representation settings. Alternating refinement also improves both allocation variants, showing that representation quality and allocation signal contribute separately. Applying surrogate refinement gives the lowest perplexity at all three budgets.

C Details and Analysis of Recovery-Aware Allocation

C.1 Recovery-Aware Greedy Allocation

For each layer module j and bit width b, Appendix B.1 constructs a bit specific $W_q + AB$ representation and records its post recovery error $L_{\text{res},j}^{(b)}$. $M_j^{(b)}$ accounts for the packed quantized values, one FP8 scale per group of 128 values, and the amortized FP16 shared-codebook storage for both W_q and the 4-bit low-rank factors. Backend-specific alignment and runtime workspace are excluded. Following Section 4.2, the hybrid memory cost $M_j^{(b)}$ includes both the quantized backbone and its low rank factors, as defined in Eq. 8. RampQ selects one bit width for each module using the recovery aware proxy $P_j^{(b)}$ under this memory budget.

Recovery aware proxy. The allocation score combines module sensitivity with the error remaining after recovery. To reduce scale differences across modules, we normalize the post recovery error by a reference precision

$$P_j^{(b)} = C_j \frac{L_{\text{res},j}^{(b)}}{L_{\text{res},j}^{(b_{\text{ref}})} + \epsilon}, \qquad b_{\text{ref}} = 4. \tag{60}$$

The normalization expresses each bit width relative to the 4 bit representation of the same module. A larger score is assigned when a sensitive module retains more error after recovery.

C.2 Effect of Recovery-Aware Allocation Signals

Section 5.3 quantitatively compares recovery aware allocation with sensitivity weighted raw error, sensitivity only, and remaining error only allocation. We examine the corresponding layerwise assignments to identify how each signal changes the precision pattern.

Figures 10 and 11 produce different precision patterns for the four allocation signals across both model families. Including post recovery error changes which modules retain high precision compared with assignments based only on sensitivity or raw quantization error.

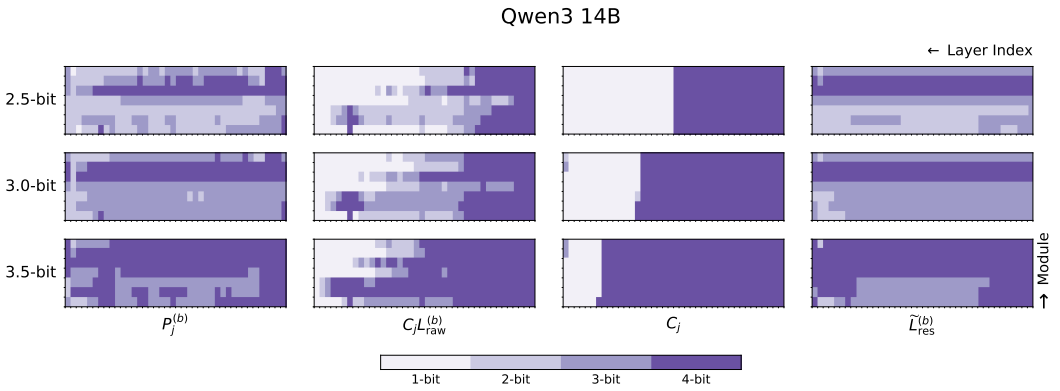


Figure 11: Layerwise bit allocation patterns on Qwen3-14B. From left to right, recovery-aware, sensitivity-weighted raw-error, sensitivity-only, and remaining-error-only allocations are shown.

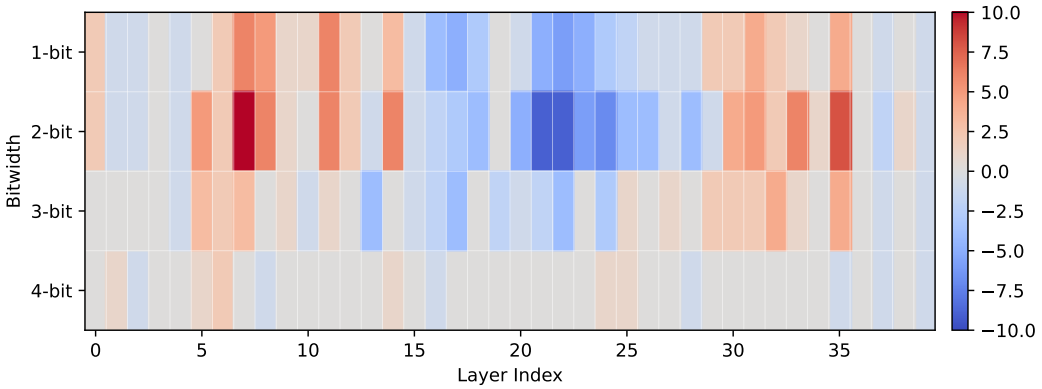


Figure 12: Change in module priority rank from sensitivity-only to recovery-aware allocation on Qwen3-14B. We report $\Delta r = r_{\text{recovery}} - r_{\text{sensitivity}}$; negative values indicate increased priority under recovery-aware allocation, and positive values indicate decreased priority.

Table 11: Effect of including 1-bit candidates in the mixed-precision search space on LLaMA3.1-8B. PTB perplexity is reported across average-bit budgets; lower is better.

Candidate search space	**Average-bit budget**				
	2.35	**2.50**	**3.00**	**3.25**	**3.50**
With 1-bit candidates	**13.80**	**12.80**	**11.26**	10.86	10.65
Without 1-bit candidates	13.90	12.98	11.30	10.86	10.65

Figure 12 shows the corresponding change in module priority. Modules whose low bit error is well recovered tend to move toward lower priority. Those that retain more error after recovery can instead move upward in the ranking. This behavior follows the recovery induced reprioritization analyzed in Section 3.2.

C.3 Role of 1-Bit Precision under Tight Budgets

Many previous mixed precision quantization methods restrict the available precisions to 2 bits or above because 1 bit weights can cause a large accuracy drop (Li et al., 2023; Huang et al., 2024; Lee et al., 2025; Ou et al., 2024). RampQ also considers 1 bit representations after low rank recovery. The bit specific $W_q + AB$ construction in Section 4.1 improves these aggressively quantized representations through alternating reconstruction. A 1 bit choice can therefore remain available when its post recovery error is sufficiently small.

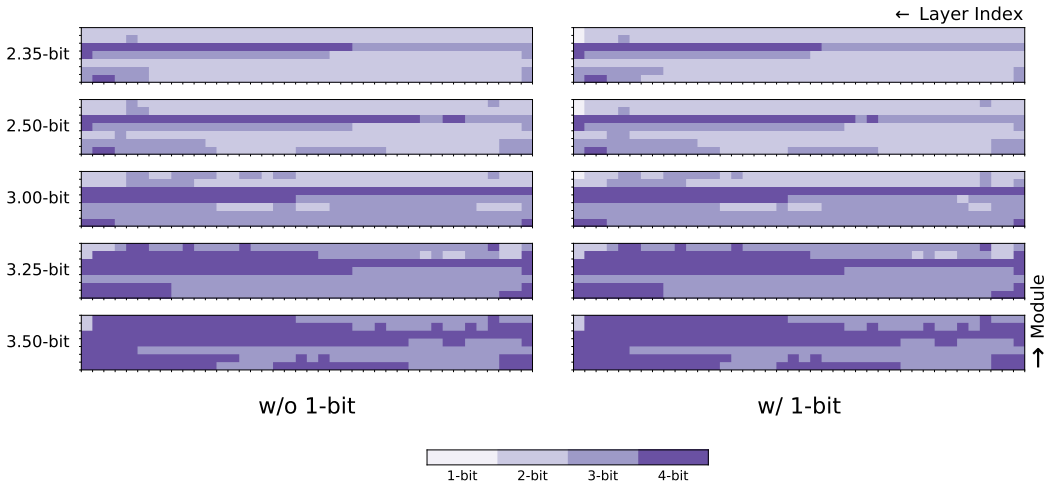


Figure 13: Layerwise bit allocation patterns with and without 1-bit precision on LLaMA3.1-8B.

Table 12: Module-wise distribution of assigned bit-widths on LLaMA3.1-8B after surrogate-guided search. Values denote the percentage of modules assigned to each bit-width under average-bit budgets of 2.5, 3.0, and 3.5.

Module	**2.5 average bits**				**3.0 average bits**				**3.5 average bits**			
	1-bit	**2-bit**	**3-bit**	**4-bit**	**1-bit**	**2-bit**	**3-bit**	**4-bit**	**1-bit**	**2-bit**	**3-bit**	**4-bit**
Q	28.1	56.2	15.6	0.0	6.2	65.6	25.0	3.1	0.0	28.1	50.0	21.9
K	25.0	50.0	25.0	0.0	9.4	71.9	15.6	3.1	3.1	6.2	31.2	59.4
V	0.0	12.5	43.8	43.8	0.0	0.0	28.1	71.9	0.0	0.0	9.4	90.6
O	0.0	31.2	46.9	21.9	0.0	12.5	34.4	53.1	0.0	0.0	15.6	84.4
Up	0.0	53.1	46.9	0.0	0.0	0.0	93.8	6.2	0.0	0.0	25.0	75.0
Down	0.0	31.2	65.6	3.1	0.0	0.0	84.4	15.6	0.0	0.0	28.1	71.9
Gate	0.0	87.5	12.5	0.0	0.0	15.6	84.4	0.0	0.0	0.0	87.5	12.5

Table 11 compares RampQ with and without 1 bit precision on LLaMA3.1-8B. Including 1 bit choices improves perplexity under the tightest memory budgets. The difference becomes small as the average bit budget increases. Its main benefit is to provide an additional low cost option when memory is highly constrained.

Figure 13 shows how this option is used in the final assignment. RampQ assigns 1 bit precision to only a subset of modules whose post recovery error remains small. The saved memory can then be reassigned to modules that benefit more from higher precision. The 1 bit representation therefore serves as a selective precision choice within recovery aware allocation rather than a uniform setting for the entire model.

C.4 Final Allocation Patterns

We report the assignments obtained after surrogate guided refinement. The following analysis summarizes the module wise bit distributions and the division of memory between the quantized backbone and low rank correction.

C.4.1 Module-Wise Bit Allocation across Budgets

Table 12 summarizes the final bit distributions on LLaMA3.1-8B across different average bit budgets. Under the tightest budget, 1 bit assignments appear mainly in the query and key projections, while

the value and output projections use higher precision. As more memory becomes available, these low bit assignments shift toward 3 and 4 bits.

The MLP projections follow a different distribution. Up and Down increasingly use 4 bits as the budget grows, whereas Gate moves mainly from 2 to 3 bits. The allocation therefore changes across projection types rather than following a uniform precision schedule over the model.

C.4.2 Budget Composition between W_q and AB

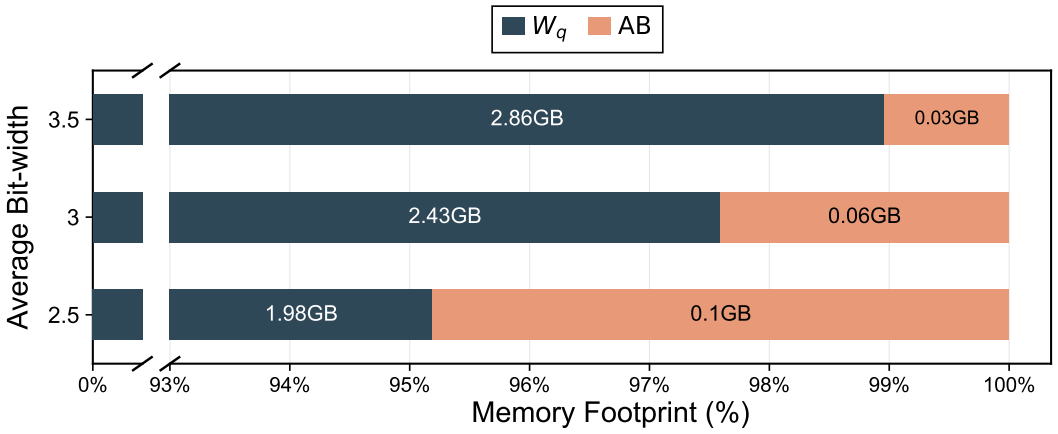


Figure 14: Memory composition of W_q and AB on LLaMA3.1-8B across average-bit budgets after surrogate-guided refinement.

Figure 14 reports how the final memory is divided between W_q and AB. The quantized backbone accounts for most of the footprint at every budget, while the low rank branch adds a smaller amount of storage. At average bit budgets of 2.5, 3.0, and 3.5, the AB branch occupies approximately 0.10, 0.06, and 0.03 GB, compared with 1.98, 2.43, and 2.86 GB for W_q, respectively. The memory used by AB decreases as the budget increases because the selected higher precision representations use smaller recovery ranks.

D Details and Analysis of Surrogate-Guided Refinement

The recovery aware proxy provides an efficient initial assignment but does not directly model interactions among precision choices from different modules. RampQ refines this assignment using full model perplexity feedback from Monte Carlo search. A lightweight surrogate is used to reduce the number of expensive perplexity evaluations.

This appendix describes the implementation and analysis of the surrogate guided refinement introduced in Section 4.3. We present the search configuration, surrogate architecture, online pairwise training, evaluation cost, and ranking behavior during refinement. We also compare surrogate guidance with exact Monte Carlo search and study the effects of online adaptation and pairwise ranking.

D.1 Search and Surrogate Implementation

Search configuration. RampQ starts from the recovery aware greedy assignment with a beam size of 10. Each beam assignment generates 20 neighboring assignments. One module whose precision can be increased and one whose precision can be decreased are sampled uniformly, and both precisions are changed by one bit. Each proposal is projected onto the hybrid memory band associated with the target budget. The beam produces up to 200 proposals per generation. We retain the 80 assignments with the smallest recovery aware proxy values, where

$$P_j^{(b)} = C_j \widetilde{L}_{\mathrm{res},j}^{(b)} \tag{61}$$

is defined in Eq. 60.

The surrogate orders these 80 assignments. True WikiText-2 validation perplexity is measured for the top 10 assignments and 8 additional assignments sampled uniformly for exploration. Among the evaluated assignments, the 10 with the lowest measured perplexity form the next beam. Refinement terminates when the best perplexity improves by less than 10^{-3} in both absolute and relative terms for 12 consecutive generations, or when the best assignment remains unchanged for 12 generations. We do not impose an additional fixed generation limit. A search lasting G generations therefore uses $18G$ true perplexity evaluations.

Surrogate initialization and architecture. The surrogate is initialized with 800 assignments whose validation perplexity has been measured. These assignments are obtained by evaluating all 80 proxy filtered assignments over 10 warm up generations. For the 2.5 and 3.0 bit searches, warm up targets are sampled uniformly from $[2.5, 3.0]$. The 3.5 bit search uses targets from $[3.0, 3.5]$. The 3.0 bit search reuses the initialization obtained from the 2.5 bit range. Each independently initialized range requires 800 true perplexity evaluations in addition to the $18G$ evaluations used during refinement at a target budget.

An assignment $\mathbf{b} = (b_1, \ldots, b_J)$ specifies one complete MPQ configuration, where b_j is the bit width of layer module j. Each module is represented using its assigned bit width, recovery aware proxy value, and module size

$$\mathbf{h}_j^{(0)} = \phi_{\text{bit}}(b_j) \oplus \phi_{\text{num}}\left(\log P_j^{(b_j)}, \log |W_j|\right) + \text{pos}(j), \tag{62}$$

where $|W_j|$ is the number of parameters in module j. The ranking model contains two MLP Mixer blocks (Tolstikhin et al., 2021). We use a model width of 128, a 32 dimensional bit embedding, token mixing and channel mixing widths of 128 and 256, GELU activations, and dropout of 0.1. Mean pooling followed by a linear head produces the ranking score $s(\mathbf{b})$.

Online pairwise training. Training pairs are formed from assignments evaluated in the same generation. For a pair (A, B), measured perplexities define

$$q_A = \frac{\exp(-\,\text{PPL}(A)/\tau)}{\exp(-\,\text{PPL}(A)/\tau) + \exp(-\,\text{PPL}(B)/\tau)}, \qquad q_B = 1 - q_A, \tag{63}$$

while the surrogate predicts

$$p_A = \sigma\left(\frac{s(B) - s(A)}{\tau}\right), \qquad p_B = 1 - p_A. \tag{64}$$

The training objective minimizes the KL divergence between measured and predicted preferences. After every generation, newly evaluated assignments are added to the training buffer and used to update the surrogate.

We set $\tau = 1$ in Eqs. 63 and 64. The initial model is trained for 50 epochs using 2,000 sampled pairs from each warm up generation. We use AdamW with a learning rate of 3×10^{-4}, weight decay of 0.01, and batch size 256. Each online update runs for three epochs using 1,200 sampled pairs with the same optimizer settings. All experiments use random seed 42. Search feedback is computed on the full WikiText-2 validation split, which is concatenated and evaluated using nonoverlapping windows of up to 2048 tokens.

Algorithm 2 summarizes the training procedure. Appendix D.3 compares the online pairwise model with fixed and regression based alternatives.

D.2 Evaluation of Surrogate-Guided Refinement

D.2.1 Comparison with Exact Monte Carlo Search

We compare surrogate guided refinement with exact Monte Carlo search to measure the effect of reducing true perplexity evaluations on the search trajectory and final model quality. Both methods start from the same RampQ-G assignment. They use the same neighbor generation procedure and the same $P_j^{(b)}$ based proxy filtering, retaining 80 assignments in each generation. Exact Monte Carlo search evaluates all 80 assignments with true perplexity. RampQ instead evaluates the 10 assignments ranked highest by the surrogate together with 8 uniformly sampled assignments.

Algorithm 2 Online Pairwise Training of the Surrogate

Require: Ranking buffer $\mathcal{D}_{\text{rank}}$ containing measured assignments, perplexities, and generation indices; surrogate f_θ; learning rate η; training iterations T_{surr}
Ensure: Updated surrogate f_θ

for $t = 1, \ldots, T_{\text{surr}}$ **do**
  Sample same-generation pair: $(A, B) \sim \mathcal{D}_{\text{rank}}$, with A and B drawn from the same generation
  Compute measured preference using Eq. 63
  Encode A and B using Eq. 62
  Predict ranking scores: $s_A \leftarrow f_\theta(A),\ s_B \leftarrow f_\theta(B)$
  Compute predicted preference using Eq. 64
  Compute pairwise loss: $\mathcal{L}_{\text{pair}} \leftarrow D_{\text{KL}}\left((q_A, q_B) \,\|\, (p_A, p_B)\right)$
  Update surrogate: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{pair}}$
end for
return f_θ

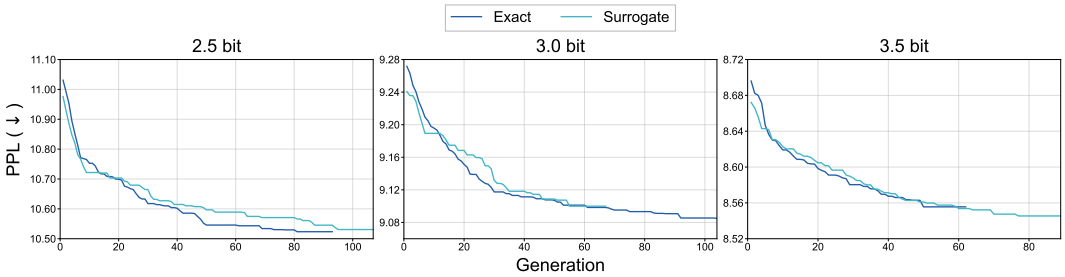


Figure 15: Exact and surrogate-guided Monte Carlo search on LLaMA3.2-3B WikiText-2 across average-bit budgets.

Table 13: Comparison of exact and surrogate-guided Monte Carlo search on LLaMA3.2-3B WikiText-2, measured using NVIDIA RTX 3090 GPU. Search time covers the complete assignment-refinement stage.

Bits	Exact MC		Surrogate MC	
	PPL(↓)	Time(↓)	PPL(↓)	Time(↓)
2.5	10.52	71.79 h	10.53	18.12 h
3.0	9.09	82.94 h	9.10	11.67 h
3.5	8.56	49.51 h	8.55	15.12 h

Figure 15 compares the refinement trajectories at the three memory budgets. Surrogate guided search follows the exact search closely and reaches a similar perplexity throughout refinement.

Table 13 reports similar final perplexity for the two search methods. Across the three budgets, the average absolute perplexity difference is approximately 0.01. Average search time decreases from 68.08 to 14.97 GPU hours, which corresponds to a 78.0% reduction. Surrogate guidance reduces the search cost while retaining the final quality reached by exact Monte Carlo search.

D.2.2 Surrogate Ranking Quality

We measure ranking quality using the overlap between the surrogate top 10 and the 10 assignments with the lowest true perplexity.

Figure 16 reports the best measured perplexity and top 10 overlap over the search generations. Perplexity decreases on both models as refinement proceeds. At the same time, the overlap generally increases, showing that the surrogate ranking becomes more consistent with the ranking from true perplexity.

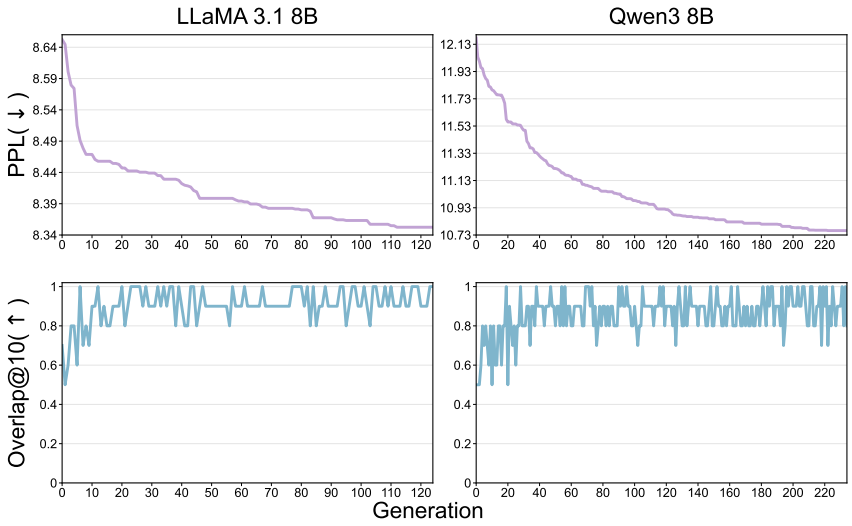


Figure 16: Best measured perplexity and surrogate top-10 overlap during refinement on LLaMA3.1-8B and Qwen3-8B. Lower perplexity and higher overlap are better.

Table 14: Search cost breakdown on LLaMA3.2-3B, measured using NVIDIA RTX 3090 GPU. Runtime and relative share are reported for each search component.

Search component	**Time (s)**	**Share (%)**
Neighbor assignment generation	0.13	0.02
Proxy filtering	0.01	0.00
Surrogate ranking	0.70	0.08
Online surrogate update	3.15	0.36
True PPL evaluation	**864.95**	**99.54**
Total	**868.94**	**100.00**

Table 15: End-to-end offline compression cost of RampQ on LLaMA3.1-8B, measured using NVIDIA RTX 4090 GPU. The shared preparation is performed once and reused across target memory budgets.

Stage	**2.5-bit**	**3.0-bit**	**3.5-bit**
Shared preparation		15.62 h	
Greedy allocation	0.17 s	0.20 s	0.22 s
MC refinement	30.31 h	27.97 h	23.55 h
Final evaluation	47 s	50 s	49 s
Total	**45.94 h**	**43.60 h**	**39.18 h**

D.2.3 Search and Model Preparation Cost

Table 14 reports the wall clock cost per generation for surrogate guided refinement on LLaMA3.2-3B. True perplexity evaluation accounts for 99.54% of the runtime. Assignment generation, proxy filtering, and surrogate ranking together take less than one second, while the online update takes 3.15 seconds. The search cost is thus dominated by full model perplexity evaluation.

Table 15 reports the total offline cost for LLaMA3.1-8B. Shared model preparation takes 15.62 hours and is performed once. The resulting representations are reused across memory budgets. Greedy allocation takes at most 0.22 seconds per budget, while the remaining cost comes mainly from

Table 16: Perplexity of intermediate RampQ assignments on LLaMA3.2-3B. The final two columns report the converged perplexity and the generation at convergence. Lower perplexity is better.

Average bits	Intermediate refinement			Convergence	
	Gen. 10	Gen. 20	Gen. 40	Final PPL	Conv. gen.
2.5	10.72	10.70	10.61	**10.53**	107
3.0	9.19	9.17	9.12	**9.10**	67
3.5	8.62	8.60	8.57	**8.55**	89

perplexity guided refinement. The total costs are 45.94, 43.60, and 39.18 hours at 2.5, 3.0, and 3.5 average bits, respectively. The reported totals correspond to the full RampQ setting, where refinement is run until convergence.

This cost is incurred only during model construction and does not affect inference. The refinement step is also optional. RampQ-G uses the same recovery aware allocation without perplexity guided refinement. When refinement is used, surrogate guidance reduces the average search time by 78.0% compared with exact Monte Carlo search, with an average absolute perplexity difference of approximately 0.01 (Table 13).

The refinement budget can also be controlled by stopping the search at a fixed generation. Table 16 reports intermediate results on LLaMA3.2-3B. At 40 generations, perplexity is within 0.8% of the converged result across all three memory budgets, while full convergence takes 67–107 generations. The search times reported in Table 13 correspond to refinement until convergence. Stopping earlier provides a lighter setting when the available offline search budget is limited.

D.2.4 Search Stability Across Random Seeds

Surrogate guided refinement includes random neighbor sampling and exploration, so the final assignment can vary across runs. We repeat the full refinement procedure on LLaMA3.2-3B using three random seeds. The bit specific representations and initial RampQ-G assignment are fixed, while only the random seed used during refinement is changed.

Table 17: Search stability across three random seeds on LLaMA3.2-3B WikiText-2. Lower is better.

Average bits	WikiText-2 perplexity			PPL reduction vs. RampQ-G
	RampQ-G	RampQ mean $\pm$ std.	Min–Max	
2.5	11.26	$\mathbf{10.5289 \pm 0.0029}$	10.5257–10.5312	6.49%
3.0	9.30	$\mathbf{9.1013 \pm 0.0013}$	9.0999–9.1025	2.14%
3.5	8.69	$\mathbf{8.5529 \pm 0.0015}$	8.5512–8.5542	1.58%

Table 17 reports the final perplexity across the three runs. The standard deviation is below 0.003 at all memory budgets. Mean perplexity is lower than RampQ-G by 0.7311, 0.1987, and 0.1371 at 2.5, 3.0, and 3.5 average bits, respectively. The variation across seeds is much smaller than the gain from refinement, showing consistent improvement across the evaluated runs.

D.3 Ablation of Surrogate Modeling Choices

We compare the online pairwise surrogate used by RampQ with (i) a fixed pairwise model without online updates and (ii) direct linear regression of measured perplexity. All variants operate on the same Monte Carlo assignment pool.

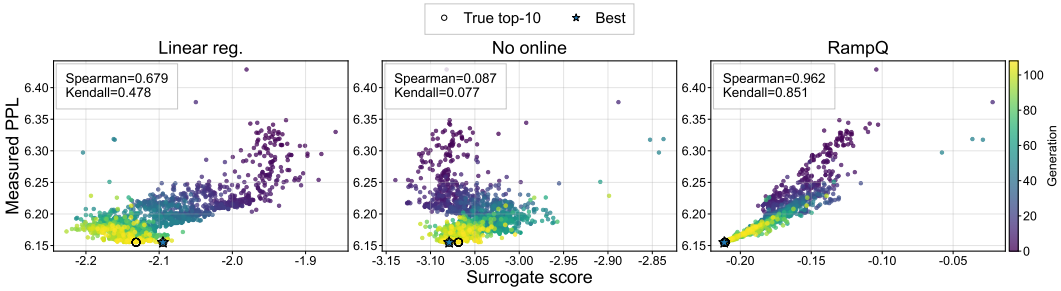


Figure 17: Surrogate score versus measured perplexity on Mistral-7B during Monte Carlo refinement. Circles indicate true top-10 assignments and the star the best measured assignment.

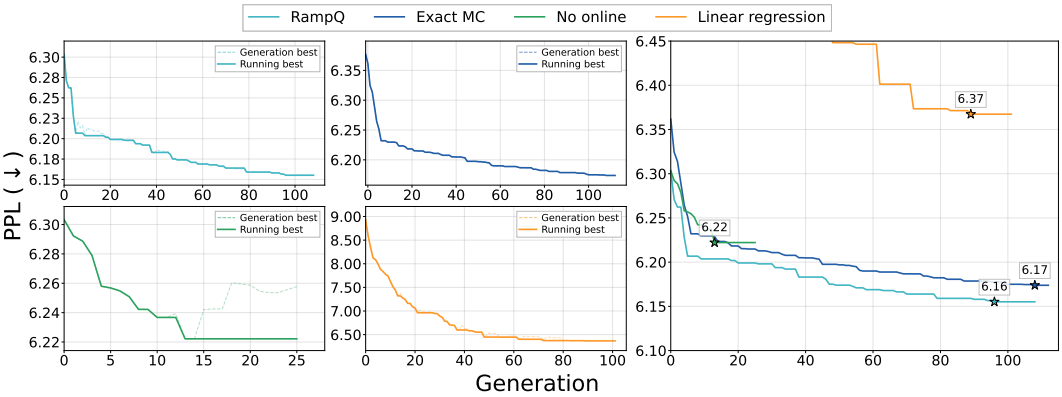


Figure 18: Refinement trajectories of exact Monte Carlo search and surrogate variants on Mistral-7B.

Table 18: Ablation of surrogate-search variants on Mistral-7B. WikiText-2 perplexity and total search time are reported. Lower values are better for both metrics.

Search variant	**WikiText-2 PPL ↓**	**Search time ↓**
Exact Monte Carlo evaluation	6.17	229.2 h
Surrogate without online update	6.25	8.33 h
Linear-regression surrogate	6.38	33.7 h
RampQ surrogate search	**6.16**	**36.3 h**

D.3.1 Ranking Alignment

Figure 17 compares each surrogate score with the ranking obtained from measured perplexity. Linear regression reaches Spearman and Kendall correlations of 0.679 and 0.478. The fixed pairwise model obtains 0.087 and 0.077. The online pairwise model reaches 0.962 and 0.851, respectively, and places the true top assignments more consistently in the favorable score region.

D.3.2 Refinement Quality and Cost

Figure 18 compares the resulting refinement trajectories. The online pairwise surrogate follows exact Monte Carlo search closely. Removing online updates or replacing pairwise ranking with perplexity regression leads to higher final perplexity.

Table 18 summarizes the final perplexity and search cost. Exact Monte Carlo reaches a perplexity of 6.17 in 229.2 GPU hours. RampQ obtains 6.16 in 36.3 GPU hours. The fixed pairwise model reaches 6.25 in 8.33 GPU hours, while linear regression gives 6.38 in 33.7 GPU hours.

Table 19: End-to-end generation throughput and latency breakdown. The AB branch latency is the sum of Bx and $A(Bx) + \mathrm{add}$. Small discrepancies are due to rounding.

		End-to-end generation		Latency breakdown (ms/token)				
Model	Avg. bits	Latency (ms/token)	TPS	$W_q x$	Bx	$A(Bx) + \mathrm{add}$	AB branch	Other ops.
LLaMA-3.2-3B	2.5	38.39	26.05	16.07	10.69	5.39	16.09	6.23
LLaMA-3.2-3B	4.0	38.58	25.92	16.15	10.74	5.42	16.17	6.26
Qwen3-8B	2.5	58.65	17.05	27.37	17.33	6.53	23.85	7.43
Qwen3-8B	4.0	56.82	17.60	26.52	16.79	6.33	23.11	7.19

The comparison supports online pairwise ranking for reducing perplexity evaluations while preserving refinement quality.

E Inference Implementation and Runtime Measurements

E.1 Throughput Measurement

We measure end to end generation throughput with batch size 1, a 64 token input sequence, and 128 newly generated tokens. Generation uses greedy decoding with `do_sample=False` and `num_beams=1`, with the KV cache enabled. Before each benchmark, we warm up the GPU with 100 matrix multiplications of size 4096×4096. We measure the complete `model.generate` call with CUDA synchronization immediately before and after execution.

Each measurement is repeated five times. We report the median throughput as

$$\mathrm{TPS} = \frac{128}{\mathrm{median}_i \, T_i}, \tag{65}$$

where T_i is the wall clock generation time of the ith run. Latency per generated token is obtained as $1000/\mathrm{TPS}$ milliseconds. The measured time includes both prompt prefill and autoregressive decoding, with prefill amortized over the 128 generated tokens. Peak GPU memory is recorded from the maximum allocated CUDA memory after resetting the peak memory statistics immediately before each benchmark.

E.2 Latency Breakdown

Table 19 separates end to end generation latency into the quantized backbone computation $W_q x$, the low rank recovery branch, and the remaining model operations. The quantized backbone is executed with FLUTE (Guo et al., 2024) based kernels. The recovery branch is computed separately as Bx, followed by $A(Bx)$ and residual addition. We use optimized GPU kernels for the low rank branch, while keeping it separate from the quantized backbone computation.

The AB branch accounts for approximately 42% of the total latency on LLaMA-3.2-3B and 41% on Qwen3-8B. Within each model, total latency changes little between the 2.5 and 4.0 bit configurations. Reducing backbone precision therefore does not produce a proportional end to end speedup in the current implementation. Despite the additional recovery computation, RampQ maintains higher throughput than CALDERA (Saha et al., 2024) under the evaluated settings.

The current implementation executes the low rank recovery branch separately from the quantized backbone. Kernel fusion or dedicated support for the recovery branch could further reduce this overhead, which we leave for future work.

F Additional Results

Table 20: Zero-shot results on LLaMA2-7B under different average-bit budgets.

Method	Bits	ARC-E	ARC-C	PIQA	HellaS	WinoG	BoolQ	AVG
		Acc ↑	Acc ↑	Acc ↑	Acc-norm ↑	Acc ↑	Acc ↑	Acc ↑
FP16	-	74.58	46.25	79.11	85.99	68.82	77.71	70.41
CALDERA	2.4	50.80	36.43	74.92	66.58	64.09	69.05	60.31
LQER	2.4	36.28	26.54	59.85	36.86	52.57	56.12	44.04
QERA	2.4	36.11	27.39	60.61	37.38	52.41	55.90	44.97
AMQ	2.5	58.88	36.86	73.50	65.01	62.75	66.39	60.56
RampQ-G	2.5	**69.70**	**41.13**	77.09	70.70	67.72	70.06	66.07
RampQ	**2.5**	68.81	39.93	**77.15**	**71.19**	**67.96**	**73.76**	**66.47**
GPTQ	3.0	70.11	43.17	77.28	72.09	67.32	73.58	67.26
AWQ	3.0	68.98	43.00	77.20	73.22	66.85	76.06	67.55
AMQ	3.0	68.22	41.72	76.55	71.27	67.32	68.44	65.59
RampQ-G	3.0	71.25	42.14	**77.64**	73.50	67.72	75.27	67.92
RampQ	**3.0**	**71.76**	**42.24**	77.20	**73.60**	**67.96**	**75.41**	**68.03**
LQER	3.4	68.94	42.49	77.31	72.52	68.11	70.52	66.65
QERA	3.4	68.86	41.81	77.48	72.30	67.48	72.14	66.68
AMQ	3.5	71.55	**44.20**	77.86	73.92	**69.06**	73.88	68.41
RampQ-G	3.5	**73.23**	43.86	77.29	73.86	68.67	73.64	68.43
RampQ	**3.5**	72.81	43.26	**78.45**	**74.12**	68.27	**76.82**	**68.96**

Table 21: Zero-shot results on LLaMA3.1-8B under different average-bit budgets.

Method	Bits	ARC-E	ARC-C	PIQA	HellaS	WinoG	BoolQ	AVG
		Acc ↑	Acc ↑	Acc ↑	Acc-norm ↑	Acc ↑	Acc ↑	Acc ↑
FP16	-	81.10	53.50	81.12	78.86	73.72	82.02	75.05
CALDERA	2.4	71.17	43.69	77.42	**72.07**	70.24	79.60	69.03
LQER	2.4	28.07	23.89	51.20	26.49	50.04	39.51	36.53
QERA	2.4	26.35	25.34	51.52	26.51	48.78	40.12	36.44
AMQ	2.5	59.47	36.26	72.52	60.25	64.56	68.75	60.30
RampQ-G	2.5	74.45	46.76	77.64	71.02	**70.88**	**80.12**	70.14
RampQ	**2.5**	**74.96**	**47.61**	**77.80**	71.95	70.80	79.91	**70.50**
GPTQ	3.0	68.01	43.00	76.71	74.05	72.61	76.82	68.53
AWQ	3.0	72.30	47.70	78.67	74.28	71.19	81.35	70.92
AMQ	3.0	71.84	44.88	77.69	70.80	71.51	79.63	69.39
RampQ-G	3.0	76.89	49.83	79.38	75.46	73.40	80.49	72.58
RampQ	**3.0**	**76.94**	**50.60**	**79.98**	**75.89**	**74.03**	**82.02**	**73.24**
LQER	3.4	71.13	41.89	76.33	71.56	70.24	74.77	67.65
QERA	3.4	71.46	43.17	75.90	71.34	69.46	75.05	67.73
AMQ	3.5	74.71	47.27	79.27	75.99	72.14	80.09	71.58
RampQ-G	3.5	**79.21**	**52.90**	80.09	77.10	**74.27**	82.11	**74.28**
RampQ	**3.5**	79.17	52.05	**80.25**	**77.32**	73.88	**82.75**	74.23

Table 22: Zero-shot results on Qwen3-8B under different average-bit budgets.

Method	Bits	ARC-E	ARC-C	PIQA	HellaS	WinoG	BoolQ	AVG
		Acc ↑	Acc ↑	Acc ↑	Acc-norm ↑	Acc ↑	Acc ↑	Acc ↑
FP16	-	80.93	56.57	77.75	74.94	67.80	86.57	74.09
CALDERA	2.4	75.46	48.38	74.54	66.73	66.61	**85.41**	69.52
LQER	2.4	33.71	25.34	54.08	31.30	49.17	45.96	39.93
QERA	2.4	31.36	25.85	54.57	31.45	51.78	49.45	40.74
AMQ	2.5	72.43	46.33	73.72	65.04	65.43	84.65	67.93
RampQ-G	2.5	76.43	49.23	74.43	66.36	67.48	83.64	69.60
RampQ	**2.5**	**77.74**	**49.66**	**76.22**	**67.10**	**67.80**	85.14	**70.61**
GPTQ	3.0	76.77	51.88	75.73	70.88	65.82	85.29	71.06
AWQ	3.0	75.84	49.57	74.97	70.19	65.11	84.56	70.04
AMQ	3.0	77.23	51.71	76.44	**71.76**	66.30	85.72	71.53
RampQ-G	3.0	78.62	50.68	76.55	70.70	**68.82**	86.15	71.92
RampQ	**3.0**	**79.55**	**52.90**	**77.15**	70.70	**68.82**	**86.76**	**72.65**
LQER	3.4	71.21	46.50	74.27	68.21	63.85	82.57	67.77
QERA	3.4	71.34	46.76	74.65	68.18	63.54	81.83	67.72
AMQ	3.5	78.75	**54.44**	76.44	**73.67**	68.11	86.24	72.94
RampQ-G	3.5	**79.88**	53.16	**77.31**	72.06	68.75	**87.31**	73.08
RampQ	**3.5**	79.42	53.50	77.26	72.53	**70.01**	86.97	**73.28**

Table 23: Zero-shot results on Qwen3-14B under different average-bit budgets.

Method	Bits	ARC-E	ARC-C	PIQA	HellaS	WinoG	BoolQ	AVG
		Acc ↑	Acc ↑	Acc ↑	Acc-norm ↑	Acc ↑	Acc ↑	Acc ↑
FP16	-	82.83	60.24	79.82	78.83	72.93	89.33	77.33
CALDERA	2.4	76.39	54.44	76.01	70.45	**72.06**	87.77	72.85
LQER	2.4	42.80	26.79	64.22	41.69	61.48	52.25	48.21
QERA	2.4	41.58	27.82	65.38	41.07	59.74	53.51	48.18
AMQ	2.5	61.78	41.30	70.35	61.87	59.43	73.82	61.43
RampQ-G	2.5	81.02	55.97	**77.31**	72.70	71.01	**88.13**	74.36
RampQ	**2.5**	**81.06**	**57.51**	77.04	**73.09**	71.03	**88.13**	**74.65**
GPTQ	3.0	77.90	55.72	**78.78**	76.07	**71.82**	87.06	74.56
AWQ	3.0	78.11	54.10	79.22	74.90	70.24	86.85	73.90
AMQ	3.0	79.84	57.17	78.24	75.50	69.46	87.46	74.61
RampQ-G	3.0	**82.83**	**59.64**	78.40	76.27	71.51	**88.38**	76.17
RampQ	**3.0**	82.32	**59.64**	78.40	**77.04**	71.64	88.33	**76.23**
LQER	3.4	78.87	54.95	78.67	74.13	68.19	87.40	73.70
QERA	3.4	78.32	54.86	78.18	74.18	68.11	87.06	73.45
AMQ	3.5	80.43	58.53	**80.30**	**77.42**	71.51	87.68	75.98
RampQ-G	3.5	80.77	59.90	79.16	77.39	72.38	**88.69**	76.38
RampQ	**3.5**	**81.86**	**60.49**	79.43	77.15	**72.45**	**88.69**	**76.68**