HiCAP-Q: Hierarchical Capability-Aware Mixed-Precision Quantization for Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Large language model quantization effectively reduces inference costs and memory usage. However, existing methods typically minimize weight or activation reconstruction errors during calibration. These objectives are only weakly aligned with preserving downstream capabilities. As a result, a small reconstruction error does not necessarily imply a small degradation in domain-specific abilities such as mathematical reasoning, code generation, or long-context understanding. We propose **HiCAP-Q**, a capability-aware mixed-precision post-training quantization framework. It preserves model-specific capabilities under a user-specified bit budget. Our key insight is that different capabilities rely on different model parameters. Some layers and linear modules are critical for mathematics, coding, or long-context reasoning. Others can tolerate more aggressive quantization. **HiCAP-Q** exploits this structure by reallocating precision from capability-insensitive parameter groups to capability-critical ones. **HiCAP-Q** consists of three stages. First, it constructs capability-specific probing inputs at low cost and without human annotation. It uses the target model and, optionally, a judge model. An **input score** selects prompts based on question quality, task relevance, and response confidence. Second, the framework estimates capability sensitivity at the Transformer-layer and linear-module levels. A capability loss score measures the degradation caused by quantizing each parameter group. Third, the framework ranks parameter groups by sensitivity and assigns mixed precision under a target bit budget. Experiments on MMLU-Pro, AIME-2026, and GPQA show consistent gains over reconstruction-based baselines. With only 10% more average bits than uniform low-bit quantization, **HiCAP-Q** substantially narrows the gap to the full-precision model. It also preserves strong general-task performance.

1 Introduction

As large language models (LLMs) continue to scale, inference requires increasing memory, bandwidth, and computation.Hendrycks et al. (2021a)Hendrycks et al. (2021a) These demands create major deployment challenges. They are especially severe on edge devices and consumer GPUs.Lin et al. (2024) They also limit multi-tenant serving. In many settings, storing and transferring model weights has become a key bottleneck.Zirui Liu et al. (2023) Post-training quantization (PTQ) reduces these costs by lowering the precision of model weights and activations.Dettmers et al. (2022) It requires little or no additional training.

Most PTQ methods optimize weight reconstruction, activation reconstruction, output deviation, or perplexity (PPL). These objectives measure local fidelity or average language-modeling quality.Dettmers et al. (2023) However, they do not directly measure downstream capabilities. Models with similar reconstruction errors or PPL can therefore show very different performance in mathematics, coding, science, or long-context understanding.

This mismatch arises because parameter groups contribute unequally to different capabilities. Some layers or linear modules may be particularly important for mathematical reasoning but have limited influence on code generation, while other modules may show the opposite pattern.Saxena et al. (2025) Consequently, ranking parameters with a single global reconstruction objective is insuffi-

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.

cient for capability-oriented precision allocation.Egiazarian et al. (2024) Under a fixed bit budget, the key question is not simply how to minimize overall reconstruction error, but how to identify the parameter groups that are critical to the capabilities of interest and allocate precision to them preferentially.

We introduce HICAP-Q, a capability-aware PTQ framework. It treats quantization as capability preservation under a deployment bit budget. The framework first constructs probing sets for the target capabilities. It then estimates capability-specific sensitivity at the Transformer-layer and linear-module levels. Finally, it assigns mixed-precision configurations under a user-specified bit budget. Users can select their target capabilities and deployment constraints. No model retraining is required. The estimated sensitivities also reveal which model components are vulnerable to quantization.

Our contributions are as follows:

- **Capability-aware PTQ formulation.** We formulate PTQ as capability preservation under a bit budget. This formulation goes beyond conventional reconstruction objectives.
- **Capability-specific sensitivity estimation.** We estimate the sensitivity of Transformer layers and linear modules. The estimator uses automatically constructed probing sets for each target capability.
- **Budget-constrained mixed-precision allocation.** We allocate higher precision to capability-critical components. The allocation satisfies a user-specified average bit-width budget.
- **Empirical validation.** We evaluate HICAP-Q on mathematics, science, general knowledge, and long-context benchmarks. It better preserves target capabilities than reconstruction-based PTQ baselines at similar average bit-widths.

2 MOTIVATION

2.1 FROM PERPLEXITY TO DOWNSTREAM CAPABILITY PRESERVATION

Strategy	PPL ↓	MMLU acc ↑	GSM8K acc ↑
fp16	8.989	72.696	41.698
first25	9.329 (+0.340)	71.379 (−1.317)	27.597 (−14.101)
last25	9.160 (+0.171)	71.265 (−1.431)	31.918 (−9.780)
random25_seed42	9.226 (+0.237)	71.856 (−0.840)	31.615 (−10.083)

Table 1: PPL and downstream capability are not consistently aligned across layer-allocation strategies on Qwen3-8B. Values in parentheses denote changes relative to FP16.

Post-training quantization (PTQ) typically minimizes weight/activation reconstruction error or calibration perplexity (PPL)Xiao et al. (2024). While these objectives control quantization distortion, they do not directly measure whether downstream capabilities are preserved. Quantization errors can be input- and capability-dependent: a small average perturbation may still disrupt the computation required for a specific capability, such as reasoning or factual knowledge.

We demonstrate this issue on **Qwen3-8B**Yang et al. (2025)using different layer-allocation strategies under the same quantization setting. Table 1 shows that PPL and downstream accuracy are not consistently aligned. For example, `last25` achieves a better PPL than `random25_seed42` (9.160 vs. 9.226), but yields lower MMLU accuracy (71.265 vs. 71.856). Moreover, `first25` increases PPL by only 0.340 relative to FP16, while reducing MMLU by 1.317 points and GSM8K by 14.101 points. These results suggest that PPL alone is insufficient for preserving downstream capabilities.

2.2 CAPABILITY-CONDITIONED PARAMETER SENSITIVITY

This observation motivates a capability-conditioned view of quantization. For a target capability c and parameter group g (e.g., a layer or linear module), we define its capability sensitivity as the

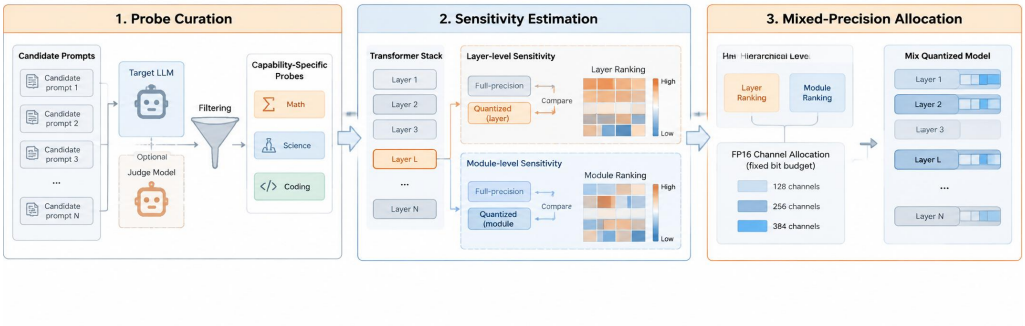


Figure 1: **Overview of HiCAP-Q.** The framework comprises three stages: **(1) Capability-aware probe curation (2) Multi-granularity sensitivity estimation** and **(3) Hierarchical mixed-precision allocation**

behavioral change caused by quantizing g on capability-relevant inputs, while keeping other groups unchanged.

This sensitivity does not assume that individual modules explicitly store a particular capability. Instead, it measures their functional importance under relevant inputs. Since module importance may vary across capabilities, a single global ranking based only on input-agnostic reconstruction error may be insufficient. Therefore, we consider both **capability-relevant probe data** and **module-wise sensitivity** for precision allocation.

2.3 Capability-Aware Mixed-Precision Allocation

Based on this principle, we propose **HiCAP-Q**, a capability-aware mixed-precision quantization framework. HiCAP-Q constructs label-free, capability-specific probe inputs, estimates layer- and module-wise sensitivity through quantization-induced behavioral loss, and allocates precision accordingly. Rather than replacing reconstruction-based objectives, HiCAP-Q complements them with capability-relevant signals to reduce downstream capability loss.

3 Method

3.1 Overview

As illustrated in Figure 1, HiCAP-Q is a capability-aware post-training mixed-precision quantization framework. Given a full-precision language model, a set of target capabilities, and an average-bit budget, our goal is to preserve the model components that are most important to the target capabilities while aggressively quantizing less sensitive components.

We consider the capability space

$$\mathcal{C} = \{\texttt{math}, \texttt{science}, \texttt{history}, \texttt{geography}, \texttt{literature}, \texttt{technology}, \texttt{coding}\},$$

HiCAP-Q consists of three stages. First, we build a compact, capability-specific probe set from an unlabeled prompt pool. The target model generates reference responses. It or an optional judge model then filters low-quality, ambiguous, and irrelevant samples.

Second, we estimate quantization sensitivity at the layer and module levels. Layer-level probing quantizes all target linear modules in one Transformer layer to capture component interactions. Module-level probing quantizes each linear module separately for finer localization. We compare each quantized model with the full-precision model using multiple behavioral metrics. This produces separate layer- and module-level sensitivity rankings.

Third, we combine both rankings to guide mixed-precision allocation. Sensitive modules and modules in sensitive layers receive more FP16 channels. All remaining channels use a fixed low-bit format. The allocation satisfies a parameter-weighted average-bit budget and prioritizes capability-critical components.

3.2 Parameter-Group Definitions

Let f_θ denote the full-precision model with parameters θ. We distinguish two granularities of parameter groups:$\mathcal{T} = \{t_1, \ldots, t_N\}, \qquad \mathcal{M} = \{m_1, \ldots, m_M\}$, where $\mathcal{T}$ denotes the set of Transformer layers and $\mathcal{M}$ denotes the set of quantizable linear modules. The latter includes attention projections and MLP projections. For each linear module $m \in \mathcal{M}$, we use $\pi(m) \in \mathcal{T}$ to denote its parent Transformer layer.

For a quantized model $f_{\theta'}$, we define the capability loss $\ell_c(x; \theta, \theta')$ as a behavioral discrepancy between the full-precision model f_θ and the quantized model $f_{\theta'}$ on input x with respect to capability c. The specific discrepancy metrics used to instantiate this abstract loss are defined in Appendix .1.

3.3 Capability-Aware Probe Curation

We begin with a small candidate prompt pool $\mathcal{X}$. The prompts may be collected from unlabeled corpora, generated by a language model, or sampled from the training splits of public datasets without using their ground-truth answers.

Target-model inference. For each candidate prompt $x \in \mathcal{X}$, the full-precision target model generates a response $A_x \sim f_\theta(\cdot \mid x)$ and reports a self-confidence score $q_{\text{self}}(x, A_x)$. The confidence can be obtained through verbalized self-evaluation or derived from sequence-level token probabilities.

Capability and quality screening. A screening model J then evaluates the tuple $(x, A_x, q_{\text{self}})$. The screening model can be either the target model itself or a lightweight external model. It predicts a capability label $\widehat{c}_x \in \mathcal{C}$ and produces scores for:

- **capability relevance**: whether x primarily evaluates the predicted capability $\widehat{c}_x$;
- **response quality**: whether A_x is coherent, complete, and likely to be correct;
- **response stability**: whether the model is expected to produce semantically consistent answers under repeated inference or small prompt variations;
- **model confidence**: whether the target model is sufficiently confident in its generated answer.

The screening model may additionally reject malformed, ambiguous, or multi-domain prompts.Chen et al. (2025) We retain only examples that pass the capability-relevance and quality thresholds and select the highest ranked examples for each category.Kim et al. (2024)

3.4 Multi-Metric Capability Sensitivity Estimation

We estimate capability sensitivity at both linear-module and Transformer-layer granularities. Let

$$\mathcal{U}_{\text{lin}} = \mathcal{M}, \qquad \mathcal{U}_{\text{layer}} = \mathcal{T},$$

where $\mathcal{M}$ and $\mathcal{T}$ denote the sets of target linear modules and Transformer layers, respectively. For each group $u \in \mathcal{U}_r$, $r \in \{\text{lin}, \text{layer}\}$, we quantize only u under a fixed probing configuration q_0, while keeping all other parameters in full precision. For layer-level probing, all target linear modules within the selected layer are quantized jointly, thereby capturing interactions between attention and MLP projections. We denote the resulting model by $f_{\theta^{(u,q_0)}}$.

Reference-aligned inference. Free-running generation may amplify early token perturbations and cause divergent decoding trajectories.Hu et al. (2025) To isolate the direct effect of quantization, we apply teacher forcing and evaluate both the full-precision and partially quantized models on the same reference response A_x. Let $Z_\theta(x, A_x)$ and $Z_{\theta^{(u,q_0)}}(x, A_x)$ denote their token-aligned output logits.

Multi-metric sensitivity. We measure behavioral discrepancy using

$$\mathcal{E} = \{\mathrm{MAE}, \mathrm{MAX}, \mathrm{RMSE}, \mathrm{COS}, \mathrm{CE}\},$$

whose definitions are provided in Appendix .1. For capability c, group u, and metric $e \in \mathcal{E}$, the raw sensitivity is

$$S^{\mathrm{raw}}_{r,e}(u, c) = \frac{1}{|X_c|} \sum_{(x,A_x)\in X_c} e(Z_\theta(x, A_x), Z_{\theta^{(u,q_0)}}(x, A_x)) \,.$$

A larger value indicates that quantizing u causes greater behavioral deviation for capability c.Lee et al. (2024) These metric-specific scores instantiate the capability loss in Section 3.2. They are normalized and fused to obtain the final sensitivities S_{lin} and S_{layer} used for FP16-channel allocation.

Metric-Debiased Sensitivity Fusion To prevent discrepancy metrics with larger scales or variances from dominating sensitivity estimation, we independently transform each metric into percentile ranks within each capability and granularity before weighted aggregation. We further fuse the resulting scores across target capabilities according to their preservation priorities, yielding unified module- and layer-level sensitivities for subsequent ranking and FP16-channel allocation; full definitions are provided in Appendix .2.

3.5 Hierarchical Sensitivity-Guided Mixed-Precision Allocation

We use the estimated sensitivities to construct a mixed-precision configuration. Linear modules and Transformer layers have different numbers of groups and different score distributions. Their raw scores are therefore not directly comparable. We rank the groups separately at each granularity. We then combine the two rankings into hierarchical sensitivity levels.

Hierarchical sensitivity levels. Let $\rho_{\mathrm{lin}}(m)$ and $\rho_{\mathrm{layer}}(t)$ denote the percentile sensitivity ranks of module m and Transformer layer t, respectively. Both lie in $(0, 1]$, and a smaller value indicates higher sensitivity. Let $\pi(m)$ denote the parent layer of module m.

Given module- and layer-level threshold sets

$$\mathcal{K} = \{\kappa_1, \ldots, \kappa_J\}, \qquad \mathcal{R} = \{r_1, \ldots, r_H\},$$

we compute

$$v_{\mathrm{lin}}(m) = \sum_{\kappa\in\mathcal{K}} \mathbb{I}[\rho_{\mathrm{lin}}(m) \le \kappa] \,,$$

and

$$v_{\mathrm{layer}}(m) = \sum_{r\in\mathcal{R}} \mathbb{I}[\rho_{\mathrm{layer}}(\pi(m)) \le r] \,.$$

The hierarchical sensitivity level is

$$h(m) = v_{\mathrm{lin}}(m) + v_{\mathrm{layer}}(m), \qquad h(m) \in \{0, \ldots, J + H\}.$$

This yields at most $L = J + H + 1$ levels. A larger $h(m)$ indicates higher sensitivity at the module level, the parent-layer level, or both. Percentile ranking makes scores from the two granularities comparable.Jeon et al. (2025)

Level-guided FP16-channel allocation. Each module uses a hybrid-precision representation. Some channels remain in FP16, while the others are quantized to NVFP4. Let C_m be the number of channels in module m, and let k_m be the number retained in FP16.

We assign an FP16-channel ratio to each sensitivity level:

$$0 \le \alpha^{(0)} \le \alpha^{(1)} \le \cdots \le \alpha^{(L-1)} \le 1.$$

The FP16-channel count of module m is

$$k_m = \mathrm{round}\left(\alpha^{(h(m))} C_m\right).$$

Thus, more sensitive modules receive a larger FP16-channel ratio. The remaining $C_m - k_m$ channels are quantized to NVFP4.

Budget-constrained channel allocation. Let b_{low} denote the effective bit width of NVFP4. Assuming equal-sized channels within each module, the effective bit width of module m is

$$\widetilde{b}_m(k_m) = b_{\text{low}} + (16 - b_{\text{low}})\frac{k_m}{C_m}.$$

We define the parameter-weighted average bit width as

$$\text{Budget}(\mathbf{k}) = \frac{\sum_{m\in\mathcal{M}} |\theta_m|\, \widetilde{b}_m(k_m)}{\sum_{m\in\mathcal{M}} |\theta_m|}.$$

Given a target budget B, we choose a monotone FP16-channel schedule such that

$$\text{Budget}(\mathbf{k}) \le B.$$

Because channel counts are discrete, the remaining budget may not support an entire sensitivity level. We therefore rank modules within that level using

$$R(m) = \frac{\widehat{S}_{\text{joint}}(m)}{\Delta\,\text{Cost}_m},$$

where

$$\widehat{S}_{\text{joint}}(m) = \eta\,\overline{S}_{\text{lin}}(m) + (1-\eta)\,\overline{S}_{\text{layer}}(\pi(m)), \qquad 0 \le \eta \le 1,$$

and

$$\Delta\,\text{Cost}_m = \frac{|\theta_m|}{C_m}\,(16 - b_{\text{low}})$$

is the cost of retaining one additional channel in FP16. We allocate additional FP16 channels in descending order of $R(m)$ until the budget is reached. This refinement applies only within the same sensitivity level, while $h(m)$ remains the primary allocation criterion.

4 Experiment

4.1 Experimental Setup

Experimental setting. We evaluate HiCAP-Q under weight-only post-training quantization, with 4-bit weights as the default configuration. The probe data are constructed following the procedure described in Section 3.3. For the Mix strategy, we set the linear-level and layer-level percentile thresholds to

$$\mathcal{K} = \{0.15, 0.30\}, \qquad \mathcal{R} = \{0.10, 0.30\}.$$

These thresholds are used to determine the hierarchical sensitivity level of each module, which subsequently guides the mixed-precision allocation.

Probe data construction. The probe set is used only for sensitivity estimation and does not involve ground-truth answers or evaluation labels. We sample candidate prompts from the official training split of each benchmark when available, and construct the final probe set using the capability-aware screening procedure in Section 3.3. For AIME 2026Zhang & Team (2026), we instead use math-related data from GSM8KCobbe et al. (2021a), MMLUHendrycks et al. (2021b), and MMLU-ProWang et al. (2024) due to its limited training split. For safety sensitivity estimation, we use AdvBench prompts as probes for HarmBenchMazeika et al. (2024), without using AdvBenchChen et al. (2022) for evaluation. All test sets remain strictly held out.

Baselines. We compare HiCAP-Q with representative state-of-the-art weight-only PTQ methods: ResQSaxena et al. (2025), AWQLin et al. (2024), GPTQFrantar et al. (2023), SliM-LLMHuang et al. (2025), and QuIPChee et al. (2024)#. All baselines are evaluated with their standard 4-bit configurations, plus randomly assigned fp16 channels, yielding an average budget of 4.4 bits. We additionally report the full-precision FP16 model as an upper-bound reference.

Table 2: Quantization results on GSM8K, TriviaQA and MMLU-Pro (fp16 baseline in green, our Mix in red).

Model	Strategy	Eff. bits	GSM8K	TriviaQA	MMLU-Pro
Qwen-8b	fp16	-	0.417	0.445	0.385
	ResQ	4.4	0.361	0.394	0.310
	AWQ	4.4	0.350	0.368	0.289
	GPTQ	4.4	0.349	0.373	0.344
	SliM-LLM	4.4	0.362	0.383	0.259
	QuIP#	4.4	0.339	0.351	0.333
	Mix	4.4	0.386	0.431	0.376
Qwen3.5-A3B	fp16	-	0.412	0.629	0.537
	ResQ	4.4	0.257	0.559	0.367
	AWQ	4.4	0.305	0.476	0.298
	GPTQ	4.4	0.343	0.597	0.356
	SliM-LLM	4.4	0.269	0.395	0.430
	QuIP#	4.4	0.320	0.522	0.463
	Mix	4.4	0.383	0.590	0.537
Llama3.1-8b	fp16	-	0.701	0.679	0.457
	ResQ	4.4	0.640	0.608	0.361
	AWQ	4.4	0.607	0.450	0.414
	GPTQ	4.4	0.617	0.391	0.386
	SliM-LLM	4.4	0.666	0.508	0.265
	QuIP#	4.4	0.541	0.588	0.303
	Mix	4.4	0.648	0.533	0.421
Spark-X2.5-4B	fp16	-	0.632	0.552	0.385
	ResQ	4.4	0.546	0.537	0.301
	AWQ	4.4	0.577	0.485	0.279
	GPTQ	4.4	0.448	0.425	0.292
	SliM-LLM	4.4	0.393	0.338	0.337
	QuIP#	4.4	0.504	0.516	0.184
	Mix	4.4	0.569	0.511	0.358
gemma-4.4-12B	fp16	-	0.752	0.521	0.497
	ResQ	4.4	0.665	0.417	0.387
	AWQ	4.4	0.622	0.358	0.401
	GPTQ	4.4	0.624	0.445	0.350
	SliM-LLM	4.4	0.582	0.333	0.441
	QuIP#	4.4	0.677	0.380	0.377
	Mix	4.4	0.695	0.346	0.462

Models and evaluation benchmarks. We evaluate five representative LLMs: Qwen3-8BYang et al. (2025), Qwen3.5-A3BQwen Team (2026), Llama-3.1-8BGrattafiori et al. (2024), Spark-X2.5-4BTeam (2026b), and Gemma-4-12BTeam (2026a). The evaluation suite covers general knowledge, mathematical reasoning, scientific reasoning, code generation, and model safety. Specifically, we report results on GSM8K, TriviaQAJoshi et al. (2017), MMLU-ProWang et al. (2024), GPQARein et al. (2023), AIME 2026Zhang & Team (2026), SciCodeTian et al. (2024), and HarmBenchMazeika et al. (2024). Higher scores are better for all benchmarks except HarmBench, where we report attack success rate (ASR), and lower values indicate better safety robustness.

4.2 Main Results

Tables 2 and 3 compare HICAP-Q with representative 4-bit PTQ baselines across multiple capabilities. Although our Mix configuration uses only 4.4 effective bits on average, it consistently preserves more full-precision performance than uniform 4-bit quantization methods.

On general knowledge and reasoning benchmarks, including GSM8K, TriviaQA, and MMLU-Pro, Mix achieves strong performance across different model families and substantially narrows the gap to the FP16 reference. It also performs well on specialized capabilities such as mathematical reasoning, scientific reasoning, and code generation, with clear gains on benchmarks such as GPQA, AIME 2026, and SciCode.

For HarmBench, where lower attack success rate is better, Mix maintains competitive safety robustness while preserving stronger utility on other tasks. Overall, these results show that quantization sensitivity is highly non-uniform across layers and modules, and that capability-aware mixed-precision allocation provides a better accuracy–compression trade-off than existing PTQ baselines.

Table 3: Quantization results on GPOA, AIME-2026, SciCode and HarmBench (attack success rate, %). Green rows denote the fp16 baselines; red rows denote our Mix strategy.

Model	Strategy	Eff. bits	GPQA	AIME-2026	SciCode	HarmBench↓
Qwen-8b	fp16	-	0.369	0.533	7	52.0
	ResQ	4.4	0.289	0.333	3	41.0
	AWQ	4.4	0.328	0.267	2	45.0
	GPTQ	4.4	0.237	0.133	1	46.0
	SliM-LLM	4.4	0.257	0.333	5	44.5
	QuIP#	4.4	0.315	0.267	4	40.0
	Mix	4.4	0.317	0.467	5	39.0
Qwen3.5-A3B	fp16	-	0.557	0.667	14	48.0
	ResQ	4.4	0.413	0.467	9	29.5
	AWQ	4.4	0.478	0.333	8	31.0
	GPTQ	4.4	0.367	0.267	7	40.5
	SliM-LLM	4.4	0.508	0.267	7	45.0
	QuIP#	4.4	0.422	0.467	6	34.0
	Mix	4.4	0.493	0.533	10	45.0
Llama3.1-8b	fp16	-	0.307	0.133	9	58.0
	ResQ	4.4	0.265	0.067	4	45.0
	AWQ	4.4	0.252	0.067	3	36.0
	GPTQ	4.4	0.250	0.000	5	39.5
	SliM-LLM	4.4	0.206	0.067	6	45.0
	QuIP#	4.4	0.159	0.000	5	31.0
	Mix	4.4	0.282	0.067	7	38.0
Spark-X2.5-4B	fp16	-	0.620	0.333	6	42.0
	ResQ	4.4	0.501	0.200	3	28.5
	AWQ	4.4	0.481	0.133	3	36.0
	GPTQ	4.4	0.525	0.133	2	43.5
	SliM-LLM	4.4	0.545	0.200	2	32.0
	QuIP#	4.4	0.393	0.267	4	39.5
	Mix	4.4	0.568	0.333	4.4	32.0
gemma-4.4-12B	fp16	-	0.453	0.467	13	62.0
	ResQ	4.4	0.356	0.267	4	50.0
	AWQ	4.4	0.386	0.333	5	47.5
	GPTQ	4.4	0.393	0.200	7	49.0
	SliM-LLM	4.4	0.393	0.333	10	49.5
	QuIP#	4.4	0.340	0.200	11	50.0
	Mix	4.4	0.411	0.400	12	47.5

Table 4: Cross-domain allocation results. Rows indicate probe domains and columns indicate evaluation domains. Domain-aligned allocation performs best. Wiki uses perplexity (lower is better); others use accuracy (higher is better).

Allocation target	GSM8K (math)↑	Wiki (general)↓	GPQA (science)↑	TriviaQA (literature)↑	MMLU-Pro (technology)↑	MMLU-Pro (history)↑
Uniform NVFP4	0.2827	9.6141	0.2588	0.2794	0.2466	0.2179
Math probes	0.4011	9.5138	0.3167	0.3223	0.2814	0.3401
Science probes	0.3550	9.4368	0.3501	0.3545	0.3210	0.2632
History probes	0.3444	9.3947	0.3093	0.3248	0.2632	0.4001
Technology probes	0.3610	9.9375	0.3282	0.2809	0.3797	0.2927
Wiki probes	0.3085	9.0141	0.3205	0.3039	0.2936	0.3065
Literature probes	0.3238	9.3947	0.2954	0.4286	0.3239	0.3257

4.3 Capability-Targeted Allocation Generalizes Across Domains

To evaluate the generality of HiCAP-Q, we construct capability-specific probe sets for six domains: *math*, *general knowledge*Merity et al. (2016), *science*, *history*, *technology*, and *literature*. Each probe set is used to estimate module sensitivity and derive a corresponding mixed-precision allocation, without using evaluation labels. We then evaluate every allocation across all downstream domains, as shown in Table 4.

The results show a clear domain-alignment effect: for each evaluation domain, the allocation guided by the corresponding probe set achieves the best performance. This indicates that quantization sensitivity varies across capabilities, and that capability-specific probes enable targeted preservation. Overall, HiCAP-Q can be readily adapted to new application domains by using representative probes that capture the desired capability.

Table 5: Cumulative ablation of our method components on GSM8K. Each row adds one component on top of the previous configuration, starting from uniform NVFP4 quantization. Δ denotes the relative improvement over the uniform NVFP4 baseline. The full configuration (last row) recovers $+54.9\%$ relative accuracy.

Strategy	**GSM8K↑**	**Δ vs. uniform**
uniform NVFP4	0.249	–
+ random mixed precision	0.265	$+6.19\%$
+ linear-only sensitivity (w/o dataset module)	0.308	$+23.64\%$
+ layer-only sensitivity (w/o dataset module)	0.319	$+27.91\%$
+ multi-level sensitivity	0.334	$+34.19\%$
+ sensitive dataset selection	0.362	$+45.47\%$
+ rank-normalized multi-metric fusion (full)	0.386	$+54.88\%$

4.4 Cumulative Effect of HiCAP-Q Components

We examine whether the components of HiCAP-Q provide complementary benefits. Our method includes capability-aware probe curation, linear- and layer-level sensitivity measurement, hierarchical multi-level allocation, and rank-normalized multi-metric fusion.

Table 5 reports cumulative ablations on GSM8K under the same average bit-width budget. Starting from uniform NVFP4 quantization, we progressively add each component; each row includes all components from the previous row.

Random mixed-precision allocation improves accuracy from 0.249 to 0.265, showing that non-uniform precision allocation is beneficial, but random selection is limited. Adding linear-module sensitivity further improves accuracy to 0.308, confirming the value of fine-grained sensitivity estimation. Layer-level sensitivity increases accuracy to 0.319, suggesting that it captures complementary block-level quantization effects. Hierarchical multi-level allocation further raises performance to 0.334 by assigning precision more flexibly according to sensitivity.

Capability-aware probe curation yields a substantial gain to 0.362, indicating that reliable and capability-relevant responses improve sensitivity estimation. Finally, rank-normalized multi-metric fusion achieves the best accuracy of 0.386, a 54.9% relative improvement over uniform NVFP4 quantization.

Overall, all components provide consistent and complementary improvements, and their combination achieves the strongest capability preservation under the same quantization budget.

5 Conclusion

We introduced HiCAP-Q, a capability-aware mixed-precision PTQ framework for LLMs. It uses capability-specific probes to estimate sensitivity at both layer and module levels. It then assigns higher precision to critical modules under a given bit budget. At only 4.3 average bits, HiCAP-Q outperforms existing 4-bit PTQ baselines across general, mathematical, scientific, and coding benchmarks. Capability-targeted allocations perform best in their corresponding domains, confirming that quantization sensitivity is capability-dependent.

AI use statement

In this work, we used generative AI tools for figure generation, sentence polishing, and manuscript revision.

We did not use generative AI tools for citation generation or for identifying novel contributions.

In addition, we used the generative AI tool Codex to perform code completion for benchmark code generation. We have reviewed all AI-assisted work. The first author verified the LLM-generated code and conducted correctness testing. We take full responsibility for the final content of this work, including text, claims, or results generated with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

We are committed to ensuring the reproducibility of the results reported in this work. To this end, we will publicly release all relevant code on GitHub in the near future. Detailed information regarding the datasets and models used in this work, including their sources and usage instructions, will be clearly documented in the README file of the GitHub repository.

REFERENCES

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. Quip: 2-bit quantization of large language models with guarantees, 2024. URL https://arxiv.org/abs/2307.13304.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models, 2025. URL https://arxiv.org/abs/2407.11062.

Yangyi Chen, Hongcheng Gao, Ganqu Cui, Fanchao Qi, Longtao Huang, Zhiyuan Liu, and Maosong Sun. Why should adversarial perturbations be imperceptible? rethink the research paradigm in adversarial nlp. *arXiv preprint arXiv:2210.10683*, 2022.

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*, 2021a.

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, 2021b. URL https://arxiv.org/abs/2110.14168.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression, 2023. URL https://arxiv.org/abs/2306.03078.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023. URL https://arxiv.org/abs/2210.17323.

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.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021b. URL https://arxiv.org/abs/2009.03300.

Xing Hu, Zhixuan Chen, Dawei Yang, Zukang Xu, Chen Xu, Zhihang Yuan, Sifan Zhou, and Jiangyong Yu. Moequant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance, 2025. URL https://arxiv.org/abs/2505.03804.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and Xiaojuan Qi. Slim-llm: Salience-driven mixed-precision quantization for large language models, 2025. URL https://arxiv.org/abs/2405.14917.

Hyesung Jeon, Yulhwa Kim, and Jae joon Kim. L4q: Parameter efficient quantization-aware fine-tuning on large language models, 2025. URL https://arxiv.org/abs/2402.04902.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, art. arXiv:1705.03551, 2017.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization, 2024. URL https://arxiv.org/abs/2306.07629.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models, 2024. URL https://arxiv.org/abs/2306.02272.

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, 2024. URL https://arxiv.org/abs/2306.00978.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal, 2024. URL https://arxiv.org/abs/2402.04249.

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

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Utkarsh Saxena, Sayeh Sharify, Kaushik Roy, and Xin Wang. Resq: Mixed-precision quantization of large language models with low-rank residuals, 2025. URL https://arxiv.org/abs/2412.14363.

Gemma Team. Gemma 4 technical report, 2026a. URL https://arxiv.org/abs/2607.02770.

SparkLLM Team. Spark-x2.5 4b1.7b: Pushing the limits of agentic capabilities in on-device models, 2026b.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu Huerta, and Hao Peng. Scicode: A research coding benchmark curated by scientists, 2024. URL https://arxiv.org/abs/2407.13168.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024. URL https://arxiv.org/abs/2211.10438.

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, 2025. URL https://arxiv.org/abs/2505.09388.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (aime) 2026, 2026. URL https://huggingface.co/datasets/math-ai/aime26.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi : Plug-and-play 2bit kv cache quantization with streaming asymmetric quantization. 2023. doi: 10.13140/RG.2.2.28167.37282. URL https://rgdoi.net/10.13140/RG.2.2.28167.37282.

A Appendix

.1 Behavioral Discrepancy Metrics

This appendix provides the explicit definitions of the discrepancy metrics used to instantiate the abstract capability loss in Section 3.2.

Let Z and $Z^{(u)}$ denote the full-precision and partially quantized logit tensors evaluated on the same reference sequence. Let D denote the number of scalar elements in the evaluated logit tensor, and let T_x denote the number of evaluated token positions.Cobbe et al. (2021b)

The mean absolute error is defined as

$$e_{\mathrm{MAE}}(Z, Z^{(u)}) = \frac{1}{D} \left\| Z - Z^{(u)} \right\|_1 .$$

The maximum absolute error is

$$e_{\mathrm{MAX}}(Z, Z^{(u)}) = \left\| Z - Z^{(u)} \right\|_\infty .$$

The root mean squared error is

$$e_{\mathrm{RMSE}}(Z, Z^{(u)}) = \sqrt{\frac{1}{D} \left\| Z - Z^{(u)} \right\|_2^2}.$$

To measure directional distortion independently of absolute logit magnitude, we use the average token-level cosine distance:

$$e_{\mathrm{COS}}(Z, Z^{(u)}) = \frac{1}{T_x} \sum_{t=1}^{T_x} \left(1 - \frac{\langle Z_t, Z_t^{(u)} \rangle}{\|Z_t\|_2 \|Z_t^{(u)}\|_2 + \epsilon} \right),$$

where $\epsilon > 0$ is a numerical stability constant.

Finally, let

$$p_t = \mathrm{softmax}(Z_t/\tau), \qquad p_t^{(u)} = \mathrm{softmax}(Z_t^{(u)}/\tau),$$

where τ is a temperature parameter. We define the predictive cross-entropy increase as

$$e_{\mathrm{CE}}(Z, Z^{(u)}) = \frac{1}{T_x} \sum_{t=1}^{T_x} \left[H(p_t, p_t^{(u)}) - H(p_t, p_t) \right].$$

Here,

$$H(p, q) = -\sum_i p_i \log q_i$$

denotes cross entropy. Since

$$H(p_t, p_t^{(u)}) - H(p_t, p_t) = D_{\mathrm{KL}} \left(p_t \| p_t^{(u)} \right),$$

the CE discrepancy is non-negative up to numerical precision and is zero when the two predictive distributions are identical.

The metric set used in the main method is therefore

$$\mathcal{E} = \{\mathrm{MAE}, \mathrm{MAX}, \mathrm{RMSE}, \mathrm{COS}, \mathrm{CE}\}.$$

.2 Parameter-Weighted Bit Budget

The parameter-weighted average bit-width of an allocation $\mathbf{b} = \{b_m\}_{m \in \mathcal{M}}$ is defined as

$$\mathrm{Budget}(\mathbf{b}) = \frac{\sum_{m \in \mathcal{M}} |\theta_m| b_m}{\sum_{m \in \mathcal{M}} |\theta_m|},$$

where $|\theta_m|$ denotes the number of parameters in linear module m. Given a target average bit budget B, the allocation must satisfy

$$\mathrm{Budget}(\mathbf{b}) \le B.$$

For a target capability $c^\star$, the ideal allocation problem is

$$\min_{\mathbf{b}} \Delta_{c^\star}(\mathbf{b}) \quad \text{s.t.} \quad b_m \in \mathcal{B}_L, \quad \mathrm{Budget}(\mathbf{b}) \le B,$$

where $\Delta_{c^\star}(\mathbf{b})$ denotes the capability degradation under allocation $\mathbf{b}$. Since directly evaluating all mixed-precision allocations is combinatorial, HiCAP-Q uses the hierarchical level $h(m, c^\star)$ as a tractable proxy and preferentially allocates additional precision to modules with higher capability sensitivity.

For multiple target capabilities, the corresponding ideal objective is

$$\min_{\mathbf{b}} \sum_{c \in \mathcal{C}} \lambda_c \Delta_c(\mathbf{b}) \quad \text{s.t.} \quad b_m \in \mathcal{B}_L, \quad \text{Budget}(\mathbf{b}) \leq B.$$

Because discrepancy metrics differ in scale and distribution, directly averaging their raw values may bias sensitivity estimates toward metrics with larger magnitudes or variances. We therefore normalize each metric independently for every capability and granularity. For $r \in \{\text{lin}, \text{layer}\}$, the percentile score of group u is

$$P_{r,e}(u, c) = \frac{\text{rank}\big(S^{\text{raw}}_{r,e}(u, c)\big) - 1}{|\mathcal{U}_r| - 1},$$

where the rank is computed in ascending order over $\mathcal{U}_r$, with ties assigned their average rank. Thus, $P_{r,e}(u, c) \in [0, 1]$, where larger values indicate greater sensitivity.

We fuse the normalized scores across metrics as

$$\widehat{S}_r(u, c) = \sum_{e \in \mathcal{E}} \omega_e P_{r,e}(u, c), \qquad \omega_e \geq 0, \quad \sum_{e \in \mathcal{E}} \omega_e = 1.$$

By default, $\omega_e = 1/|\mathcal{E}|$; alternatively, the weights can be determined from each metric's correlation with downstream degradation on validation data.

For multiple target capabilities, the final sensitivity is

$$\overline{S}_r(u) = \sum_{c \in \mathcal{C}} \lambda_c \widehat{S}_r(u, c), \qquad \lambda_c \geq 0, \quad \sum_{c \in \mathcal{C}} \lambda_c = 1,$$

where λ_c controls the preservation priority of capability c. For a single target capability, its weight is set to one. This yields $\overline{S}_{\text{lin}}(m)$ for $m \in \mathcal{M}$ and $\overline{S}_{\text{layer}}(t)$ for $t \in \mathcal{T}$, which are used for all subsequent ranking and allocation.