AUTONOMOUS MODEL QUANTIZATION WITH EXPLORER–REVIEWER AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Model quantization is critical for efficient deployment of large language models, but developing a strong quantization recipe requires decisions about methods, data, and experimental evidence. We study frontier coding agents for this task.

When given a quantization method and its target configuration, a strong standalone agent can set up the environment, debug the implementation, and reproduce the reported results. In this setting, however, the researcher still chooses the method. Autonomous quantization optimization instead requires the agent to select methods, design experiments, and decide which directions to pursue. Although the agent can develop competitive recipes, we observe four problems during autonomous exploration: overlooked prior work, inadequate experimental design, unsupported conclusions, and premature stopping. Based on these observations, we propose an Explorer–Reviewer system. The Explorer develops and executes candidate improvements, while an independent Reviewer evaluates research decisions at four checkpoints: proposals, experimental plans, conclusions, and requests to stop. The agents communicate through a shared research log. Our study covers 2-bit weight-only quantization of Llama-3-8B, Qwen3-8B, and Mistral-7B. On Llama-3-8B, Qwen3-8B, and Mistral-7B, our system improves average zero-shot accuracy across seven tasks by 0.88, 0.60, and 0.19 percentage points, respectively, over the standalone agent. It also outperforms the evaluated published quantization baselines on this metric for all three models. Case studies show how the Reviewer guides data selection, requests stronger experimental controls, and challenges unsupported conclusions.

1 INTRODUCTION

Quantization reduces the memory footprint and inference cost of large language models (LLMs). Methods such as GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024), QTIP (Tseng et al., 2024b), and YAQA (Tseng et al., 2026) provide effective ways to preserve model quality at low precision. Obtaining a strong quantized model, however, requires more than implementing a published method. The researcher must decide which techniques suit the target model and how to adapt or combine them. A user may know which model to compress and how to evaluate it without knowing which methods to try.

Frontier coding agents offer a practical starting point for automating this work. Execution-grounded agents can navigate repositories, modify programs, and resolve errors through tool feedback (Yang et al., 2024); research agents extend these capabilities to implementing and evaluating machine learning ideas (Jiang et al., 2025; Yamada et al., 2025). In our specified-method reproduction experiments, a standalone agent sets up the environment, debugs the implementation, and reproduces published quantization results. In this setting, the researcher has already decided what method to pursue. Given only a model, a quality objective, and compression constraints, the agent must also choose research directions, design experiments, and judge what their results establish.

The agent can make substantial progress in this open-ended setting, finding recipes that outperform published quantization baselines in our comparisons. Yet successful execution does not ensure sound research decisions. Our exploratory runs reveal four problems: overlooked prior work, inadequate experimental design, unsupported conclusions, and premature stopping (Section 3.1). Because the same agent conducts the research and judges its own evidence, these errors can go unchecked.

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.

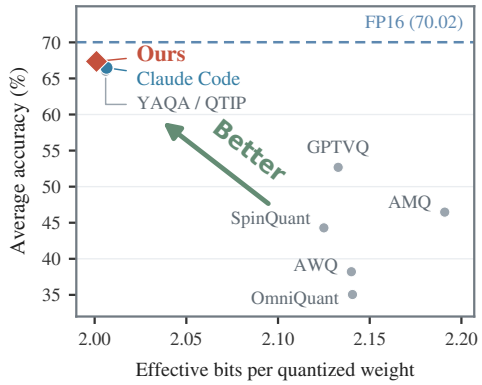


Figure 1: Llama-3-8B at nominal 2 bits. Ours (Explorer–Reviewer) achieves the highest seven-task average accuracy. Claude Code is the single-agent baseline. The dashed line shows FP16 accuracy. Effective bits include auxiliary storage, excluding the high-precision embedding and output head.

We propose an *Explorer–Reviewer* system that separates autonomous experimentation from high-level review. The Explorer iterates through literature search, method development, experiments, and analysis. The Reviewer provides feedback on proposals, experimental plans, and conclusions. When the Explorer asks to stop, the Reviewer checks the research log and either approves termination or requests further exploration. Each review starts with a fresh context and uses the shared log, keeping high-level feedback separate from the Explorer's coding and debugging history.

We study 2-bit weight-only quantization on Llama-3-8B (Grattafiori et al., 2024), Qwen3-8B (Yang et al., 2025), and Mistral-7B (Jiang et al., 2023), evaluating WikiText-2 perplexity and seven zero-shot tasks. On Llama-3-8B, Explorer–Reviewer reaches 67.36% average accuracy, compared with 66.48% for the standalone agent (Figure 1); on Qwen3-8B, it reaches 68.92% versus 68.32%. On Mistral-7B, it reaches 68.92% versus 68.73%. These gains of 0.88, 0.60, and 0.19 percentage points place the system above both the standalone agent and the compared published quantization baselines on all three models. Review-stage ablations on Qwen3-8B and case studies further examine how review changes the search and its outcomes.

In summary, this paper makes the following contributions:

- We study the capabilities and limitations of frontier coding agents for model quantization, demonstrating successful specified-method reproduction and identifying failures in autonomous exploration.
- Based on these observations, we propose Explorer–Reviewer, which adds independent review at four key decision points in an autonomous research loop.
- Our quantization experiments demonstrate the system's effectiveness in improving model quality. Review-stage ablations and case studies further examine how review contributes to the research process.

2 Related Work

Low-bit LLM quantization. Post-training quantization offers several mechanisms for preserving model quality under low-precision constraints. GPTQ (Frantar et al., 2023) uses approximate second-order information for weight reconstruction, while AWQ (Lin et al., 2024) uses activation-aware scaling to protect salient weights. OmniQuant (Shao et al., 2024) learns clipping and equivalent transformations, and SpinQuant (Liu et al., 2025) learns rotations that make activations and weights easier to quantize. Structured representations further expand the low-bit design space. GPTVQ (van Baalen et al., 2024) combines vector quantization with second-order information; AQLM (Egiazarian et al., 2024) learns additive codebooks; QuIP# (Tseng et al., 2024a) combines incoherence processing with lattice-based vector quantization; and QTIP (Tseng et al., 2024b) uses trellis-coded quantization to increase effective quantization dimension without an exponentially large codebook. YAQA (Tseng et al., 2026) improves the rounding objective using Kronecker-factored approximations to a model-level Hessian. GuidedQuant (Kim et al., 2025) incorporates end-loss guidance into reconstruction and can be combined with existing quantizers. PV-Tuning (Malinovskii et al., 2024) alternates updates to discrete assignments and continuous parameters. These methods provide different mechanisms for quantized representations, reconstruction, and recovery.

Automated quantization and deployment. Quantization has long involved automated search. HAQ (Wang et al., 2019) uses reinforcement learning and hardware feedback to choose mixed-precision policies. More recently, AMQ (Lee et al., 2025) searches layer-wise bit-width assignments through surrogate-assisted multi-objective optimization, using a quantization proxy and a quality predictor. HAQA (Deng et al., 2026) applies LLM agents to tune quantization, fine-tuning, and hardware deployment settings within supplied frameworks and hyperparameter ranges. Whereas these approaches optimize configurations within specified search spaces, our task leaves the choice of methods, calibration and recovery data, and implementation to the agent. The agent can retrieve existing methods, modify or combine their components, and revise its experiments as it learns about the target model.

Coding agents and autonomous ML research. SWE-agent (Yang et al., 2024) develops interfaces for agents that edit repositories and execute programs. AIDE (Jiang et al., 2025) formulates machine learning engineering as search over candidate implementations, while MLE-bench (Chan et al., 2025) evaluates agents on end-to-end ML tasks. The AI Scientist (Lu et al., 2024) and its successor (Yamada et al., 2025) automate research workflows spanning ideas, experiments, and scientific reports. Our study focuses on model quantization, distinguishing the ability to reproduce a specified method from the ability to select and improve a recipe through experiments.

Feedback and review in research agents. Iterative feedback helps agents refine their outputs and plans. Self-Refine (Madaan et al., 2023) generates feedback and revisions with a single model, and Reflexion (Shinn et al., 2023) uses verbal reflections to inform subsequent attempts. ResearchAgent (Baek et al., 2025) employs reviewing agents to refine literature-grounded research ideas, including proposed methods and experimental designs, rather than executing those experiments. Agent Laboratory (Schmidgall et al., 2025) reviews research reports and can return to planning, experimentation, or interpretation in response to reviewer feedback. AutoML-Agent (Trirat et al., 2025) verifies user requests, planned solutions, and executed implementations, revising its plans when verification fails. Our system applies independent review to research decisions throughout quantization optimization. Log-based communication separates high-level review from code execution, and final review determines whether exploration should continue.

3 Explorer–Reviewer Quantization

3.1 From Execution to Research Decisions

Research logs from our early standalone explorations reveal four types of research judgment that can fail despite successful execution. Figure 2 pairs each example with a corresponding review question; Appendix E provides the experimental details.

Overlooked prior work. The agent observed that a small fraction of calibration tokens dominates the output-gradient energy and proposed a token-weighted Hessian. GuidedQuant (Kim et al., 2025) and YAQA (Tseng et al., 2026) had already incorporated end-loss information into quantization, but the agent did not connect these findings to its proposal. It spent several rounds developing the mechanism from scratch, rather than using prior work to guide its adaptation to the target model.

Inadequate experimental design. The agent retained a rank-1 correction after it improved an earlier baseline. When the recipe changed, it did not check whether the correction still helped. Follow-up controlled experiments found no additional benefit before fine-tuning. The earlier tests had not established the component's value in the final recipe.

Unsupported conclusions. After one short failed trial and a few small-batch diagnostics, the agent abandoned PV-inspired discrete refinement. It treated the instability of the tested setup as evidence against the entire direction, without adequately testing alternative configurations.

Premature stopping. The agent proposed loss-aware quantization-code selection but neither tested it nor recorded a reason to abandon it. It later declared convergence when changes to its current recipe stopped improving performance, leaving this earlier direction unresolved.

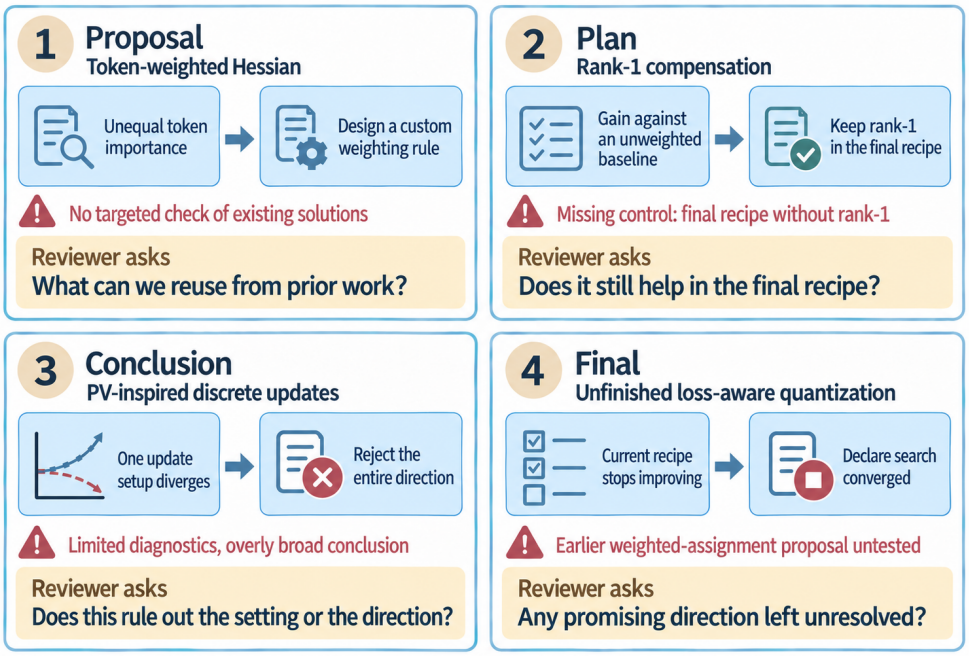


Figure 2: Four judgments that motivate independent review. Each panel shows a standalone agent's decision, the gap in its reasoning, and the question a Reviewer would ask at the corresponding checkpoint.

These cases motivate independent review of proposals, experimental plans, conclusions, and requests to stop. Review checks whether the evidence supports the research decision, while leaving implementation and experimentation to the Explorer.

3.2 Task and Division of Responsibilities

The input is a pretrained model, an evaluation objective, and quantization and deployment constraints. The user supplies neither a quantization method nor calibration or recovery-training data. The agent must select these and deliver a quantized model, loading and inference code, and a research record supporting the selected recipe.

The *Explorer* has access to the model, a GPU, the shell, and the web. It surveys prior work, maintains a roadmap, implements candidates, and runs experiments. The *Reviewer* assesses research decisions from the log and relevant literature, without inspecting source code or taking over implementation. This division keeps review focused on directions and evidence rather than coding and debugging.

The Explorer retains its own loop of proposing, testing, and revising ideas. Review adds feedback at key research decisions while leaving implementation to the Explorer. The Reviewer can ask for a different comparison or a narrower conclusion while leaving the Explorer to decide how to carry out the work. Apart from the task specification, the loop and review prompts focus on general research judgments rather than quantization-specific procedures.

3.3 The Research Log and Review Protocol

The agents communicate through a chronological research log containing proposals, plans, data and commands, results, and decisions. Reviewer feedback is appended verbatim, followed by the Explorer's response and actions. Every review starts with a fresh context, receives the original task and review request, and reads the complete log. It does not inherit the Explorer's coding conversation. Figure 3 summarizes the protocol.

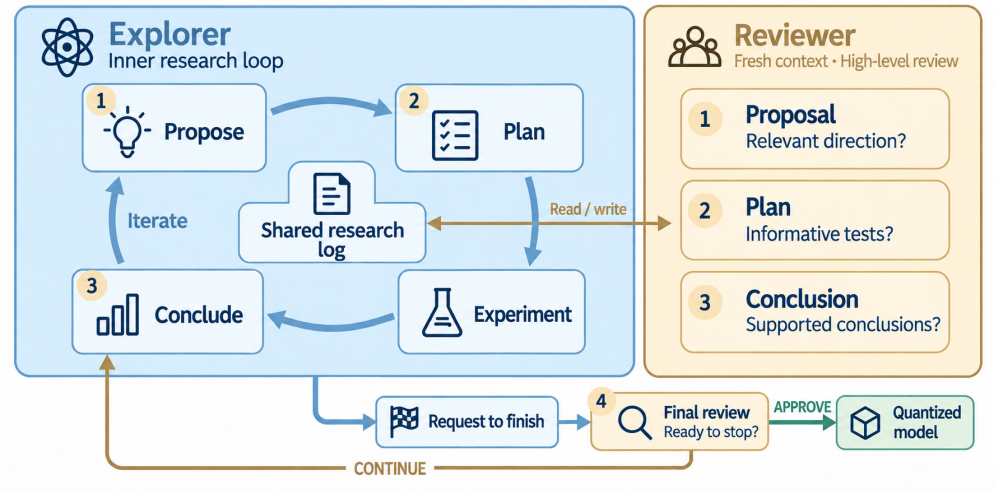


Figure 3: Explorer–Reviewer: an inner research loop with four review checkpoints. Proposal (1), plan (2), and conclusion (3) reviews advise the Explorer through a shared research log. Final review (4) either approves termination or returns the Explorer to the loop.

Keeping the research record while separating the conversations serves two purposes: previous results and unresolved directions remain available for review, but low-level debugging history does not become part of the Reviewer's context. Fresh context therefore does not mean starting without the evidence accumulated by the Explorer.

Proposal review. Before committing to a direction, the Explorer records its proposal and expected benefit. The Reviewer searches for relevant findings and implementations, assesses their applicability, and compares the proposal with alternatives in the log. This review guides method selection and adaptation using existing knowledge.

Plan review. Before launching experiments, the Explorer records the comparisons, data, and expected outcomes. The Reviewer checks whether the plan can answer the intended question. For example, attributing a gain to one component requires a control that isolates its contribution; comparing complete recipes does not require isolating every component.

Conclusion review. After an attempt, the Reviewer checks whether fair comparisons support the claimed gains and whether negative results justify rejecting a configuration or an entire direction. It also checks consistency with earlier results and whether combined methods were tested together, identifying missing evidence needed for the next decision.

These three reviews are advisory. The Explorer may disagree, provided it records its reasoning; repeated exchanges on the same point are bounded. Requests for additional experiments should be proportionate to their expected value and cost.

3.4 Reviewer-Governed Termination

When the Explorer attempts to finish, a stop hook requests *final review*. The Reviewer checks the final results and deliverables and whether promising directions in the roadmap have been explored or reasonably retired. It revisits unsupported conclusions using the complete research record. The criterion is a justified stopping decision, not global optimality or exhaustive search.

The Reviewer returns `APPROVE` or `CONTINUE`. Approval permits normal termination. Continuation blocks the stop attempt and returns specific unresolved issues to the Explorer in the same research loop. The Reviewer must explain why the existing evidence is insufficient and further work is worth the cost. An unavailable review does not count as approval. Operational safeguards, including the cap on consecutive stop checks, are described in Appendix A.

4 EXPERIMENTS

We evaluate specified-method reproduction, autonomous quantization quality, and the research decisions affected by review.

4.1 EXPERIMENTAL SETUP

Models, constraints, and metrics. We study Llama-3-8B, Qwen3-8B, and Mistral-7B-v0.1 at nominal 2-bit weight-only quantization. Agents must use at most 2.01 effective bits per quantized Transformer-block weight, including auxiliary storage such as scales, transforms, and codebooks. This allows a 0.01-bit margin above the nominal target. Embedding and `lm_head` remain high precision in the primary comparisons and are excluded from this denominator. Baselines use their respective nominal 2-bit configurations; Appendix B reports their effective storage rates. Agents select their own calibration and recovery-training data.

The primary metric is the unweighted average (AVG-7) of seven zero-shot tasks: ARC-Easy and ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), PIQA (Bisk et al., 2019), OpenBookQA (Mihaylov et al., 2018), and BoolQ (Clark et al., 2019). We use normalized accuracy for ARC, HellaSwag, PIQA, and OpenBookQA, and accuracy for WinoGrande and BoolQ. WikiText-2 (Merity et al., 2016) perplexity with non-overlapping 2048-token windows is the secondary metric. Zero-shot refers to evaluation without in-context demonstrations; agents may select external training data for recovery.

Agents and comparisons. The Explorer uses Claude Code with `claude-fable-5-1`; the Reviewer uses Codex with `gpt-5.6-sol` at high reasoning effort. Each autonomous run uses one H200 in a separate Docker container, with isolated workspaces, home directories, and agent memory. The standalone baseline receives the same core prompt without Reviewer instructions or hooks; its native tools, including subagents, remain available. All quantization baseline scores come from our own reproductions.

We run each configuration once, without a preset time or token budget. Each system follows its own stopping protocol. We therefore compare outcomes under autonomous stopping, not compute-matched search efficiency. Appendix A gives configuration and protocol details.

4.2 SPECIFIED-METHOD REPRODUCTION

Table 1: Specified-method reproduction using each paper's model, quantization configuration, and evaluation protocol. Wiki and C4 report perplexity; lower is better. FT denotes recovery fine-tuning.

			Wiki ↓		C4 ↓	
Method	Model	Context	Paper	Agent	Paper	Agent
QTIP + recovery FT	Llama-3-8B	8192	7.33	7.14	8.62	8.43
GuidedQuant, no FT	Llama-2-7B	4096	6.19	6.20	8.06	8.05
YAQA, no FT	Llama-3.1-8B-Inst.	8192	8.39	8.39	9.83	9.83
YAQA + recovery FT	Llama-3.1-8B-Inst.	8192	7.60	7.51	8.96	8.95

Given a published method and its reference configuration, the standalone agent sets up the environment, adapts and debugs the implementation, and reproduces the reported results. It completes all three workflows—QTIP (Tseng et al., 2024b), GuidedQuant (Kim et al., 2025), and YAQA (Tseng et al., 2026)—with results close to the references (Table 1), including YAQA before and after recovery fine-tuning. These results demonstrate successful execution when the method and configuration are specified.

4.3 AUTONOMOUS QUANTIZATION RESULTS

The standalone agent exceeds the reproduced baselines in AVG-7 on all three models (Table 2). Explorer–Reviewer improves further on all three: from 66.48 to 67.36 on Llama-3-8B, from 68.32 to 68.92 on Qwen3-8B, and from 68.73 to 68.92 on Mistral-7B. These gains are over an agent that

Table 2: Nominal 2-bit weight-only quantization with high-precision embedding and output head. PPL is WikiText-2 perplexity at context 2048; AVG-7 is average zero-shot accuracy (%). Bold marks the best quantized PPL and AVG-7 for each model. Per-task scores and effective bit widths appear in Appendix B.

	Llama-3-8B		Qwen3-8B		Mistral-7B	
Method	PPL ↓	AVG-7 ↑	PPL ↓	AVG-7 ↑	PPL ↓	AVG-7 ↑
16-bit reference	6.14	70.02	9.73	69.51	5.25	71.50
AWQ	1.70×10^6	38.21	1.37×10^9	38.96	37977	35.41
OmniQuant	332.84	35.05	25.62	52.11	100.65	38.96
SpinQuant	25.96	44.29	15.38	51.06	11.97	52.35
AMQ	81.33	46.47	40.82	50.26	14.87	55.97
GPTVQ	11.91	52.68	14.63	55.91	10.82	49.30
YAQA	7.75	66.30	10.44	67.36	5.82	68.64
QTIP	8.05	65.99	10.67	67.99	5.89	68.67
Standalone agent	7.58	66.48	**9.99**	68.32	**5.79**	68.73
Explorer–Reviewer	**7.56**	**67.36**	10.52	**68.92**	5.81	**68.92**

already finds and refines strong quantization methods. On Mistral, Explorer–Reviewer also exceeds the strongest reproduced baseline, QTIP (68.67). On Qwen and Mistral, higher AVG-7 comes with slightly higher PPL: 10.52 versus 9.99 and 5.81 versus 5.79, respectively. Optimizing downstream accuracy need not also minimize perplexity. We examine the selected recipes and the decisions behind them below.

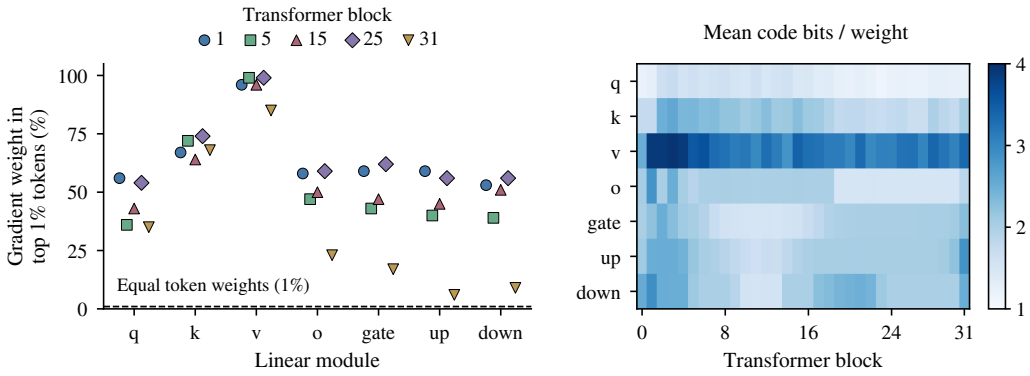
4.4 What Recipes Do the Agents Discover?

We use the final Llama-3-8B recipe as an example of what Explorer–Reviewer discovers. It combines QTIP with token-weighted Hessians and groupwise mixed precision, followed by recovery fine-tuning. These adaptations address two measured properties of the model: uneven token importance and variation in quantization sensitivity across weight groups.

Accounting for uneven token importance. The agent profiles gradients of the full-precision model's next-token loss. In the sampled value projections, the highest-weight 1% of tokens accounts for 85–99% of the squared-gradient weight (Figure 4(a)). This profile motivates giving greater weight to tokens with higher estimated loss sensitivity, rather than treating every token's reconstruction error equally. Following loss-aware quantization (Kim et al., 2025; Tseng et al., 2026), the recipe uses a token-weighted Hessian with square-root tempering to limit the dominance of individual tokens.

Allocating precision where it matters. Profiling also finds near-Gaussian weight statistics after rotation, yet quantization sensitivity still varies across layers and weight groups. Similar weight distributions do not imply that quantization errors have equal effects on model quality. The agent therefore uses estimated loss sensitivity to guide bit allocation.

The agent estimates local sensitivity from the output errors induced by quantized weights, weighted by the full-precision model's squared output gradients. A budget-constrained allocator assigns 16-row groups code rates from $\{1, 1.5, 2, 2.5, 3, 4\}$ bits per weight. Value projections receive 3.26 code bits per weight on average, compared with 1.39 for query projections, with further variation across blocks (Figure 4(b)). The total remains approximately 2 effective bits per quantized weight, including auxiliary storage. Higher rates for sensitive groups are paid for by lower rates elsewhere: the adaptation redistributes a fixed storage budget rather than increasing precision throughout the model. Together, these choices adapt reconstruction and storage allocation to the target model: token gradients guide which errors to protect against, while groupwise sensitivity guides where to spend bits. Appendix C gives the profiling setup and allocation details.



(a) Token-weight concentration

(b) Mixed-precision allocation

Figure 4: Model-specific adaptations in the Llama-3-8B recipe. (a) Share of squared-gradient weight carried by the highest-weight 1% of tokens before square-root tempering, measured on 16 sequences of 2048 tokens. Each marker represents one module/block; the dashed line shows equal token weights. (b) Mean code bits per weight across 16-row groups in each linear module. Code bits exclude auxiliary storage, which is included in the global effective-bit budget.

4.5 Ablation Study

On Qwen3-8B, we compare the standalone agent with final-only review, then progressively enable proposal, plan, and conclusion review. Table 3 reports average zero-shot accuracy and WikiText-2 perplexity for the resulting configurations.

Table 3: Review-stage ablation on Qwen3-8B. All configurations retain high-precision embedding and output head. Each configuration is evaluated in one independent research trajectory. ΔAVG-7 is the gain over the standalone agent, in percentage points.

Model	Enabled review stages	AVG-7 ↑	PPL ↓	ΔAVG-7
Qwen3-8B	None (standalone)	68.32	9.99	–
	Final	68.59	10.26	+0.27
	Proposal + Final	68.63	10.50	+0.31
	Proposal + Plan + Final	68.62	**9.97**	+0.30
	Proposal + Plan + Conclusion + Final	**68.92**	10.52	+0.60

Every configuration with review exceeds the standalone agent in AVG-7. The full protocol achieves the highest AVG-7, our primary metric, at 68.92—a gain of 0.60 percentage points over standalone. Proposal-plus-plan-plus-final review achieves the lowest PPL at 9.97.

4.6 Case Studies

We examine the research logs to understand how review changes what the Explorer tests and which results it accepts. The following cases trace a review suggestion through the resulting experiment or correction.

Adapting recovery data to downstream tasks. During Qwen exploration, changes to the quantization objective improve perplexity without improving downstream accuracy. At a conclusion checkpoint, the Reviewer suggests adding QA-style data to recovery training while retaining general text. A subsequent plan review checks the proposed comparisons and clarifies which conclusions they can support. The Explorer tests a mixture of 512 RedPajama sequences and 256 QA-formatted sequences from the training splits of SciQ, CommonsenseQA, and QASC, using the full-precision model's predictions as targets. Both runs use 768 sequences and four epochs: the QA mixture achieves 68.58 AVG, compared with 68.18 for general text, while PPL is 10.61 versus 10.54. The

Explorer retains QA data in the final recipe. Review thus introduces a concrete way to improve downstream accuracy when better perplexity alone has not translated into better task performance. Appendix D.1 gives the comparison settings.

Preventing validation leakage. In the Llama run, the Explorer plans to enlarge a recovery dataset and reserve its last 128 sequences for validation. Since the original dataset forms the beginning of the enlarged dataset, this would move the old validation examples into training. At the plan checkpoint, the Reviewer catches the problem and requires the original validation set to remain held out. The Explorer instead constructs 2432 training sequences with the same 128 validation sequences. The correction prevents validation leakage before training begins.

Testing a prematurely dismissed direction. In the Mistral run, the Explorer estimates the benefit of mixed precision from noise-sensitivity measurements and decides that the expected gain is too small to justify a test. At the conclusion checkpoint, the Reviewer points out that injected noise does not establish how actual quantization will affect model quality. It requests a direct comparison for a candidate that fits the bit budget. The Explorer increases five sensitive value projections from 2 to 3 bits, requantizes the affected blocks, and applies the same recovery-training recipe. PPL decreases from 5.823 to 5.813, while AVG-7 rises slightly from 68.55 to 68.62. Both configurations remain within the 2.01-bit budget. Review thus leads the Explorer to test a direction it had dismissed on an estimate, yielding a small measured improvement.

These cases show how review changes both the recipes explored and the experiments used to judge them. Appendix D provides the corresponding log excerpts. Review does not replace implementation checks: the Reviewer relies on the research log and can miss code-level errors.

5 Conclusion

We studied frontier coding agents for model quantization and found a gap between reproducing a specified method and conducting open-ended optimization. A standalone agent can implement and debug published methods, but can also overlook relevant work, draw conclusions from incomplete experiments, or stop before testing promising directions. We introduced Explorer–Reviewer to separate exploration from independent high-level review at four checkpoints: proposals, experimental plans, conclusions, and requests to stop. The reported results show improved average zero-shot accuracy over the standalone agent and the evaluated published-method baselines on all three models. The case studies show how review guides recovery-data selection, prevents validation leakage, and reopens directions dismissed without direct testing.

Beyond the task specification, our research loop and review prompts focus on general research decisions rather than quantization-specific procedures. This design offers a starting point for extending Explorer–Reviewer to autonomous research in other domains.

Ethics Statement

This work studies autonomous quantization of publicly available language models using existing datasets and benchmarks. Lower deployment costs may also make harmful applications more accessible, and quantization does not guarantee that a model's safety behavior is preserved. Autonomous agents can produce incorrect code or unsupported conclusions; independent review does not eliminate these risks. Our review mechanism supports research decisions rather than certifying model safety, and the authors remain responsible for the reported findings.

Reproducibility Statement

Section 3 describes the agent roles, research-log interface, review checkpoints, and termination protocol. Section 4.1 specifies the agent models, hardware, isolation, quantization constraints, and evaluation metrics. The research logs record experimental plans, commands, results, and Reviewer feedback, including unsuccessful directions. Each quantitative configuration uses one independent research trajectory. Agent stochasticity and changes to hosted models can lead to different trajectories when the workflow is repeated. We will release the code upon acceptance. Appendix F provides

the Qwen Explorer prompt and the Reviewer template, and specifies the suffix removed for the standalone condition.

AI USE STATEMENT

We used generative AI in two ways. (i) As the core of the studied system, the Explorer runs on Claude Code with `claude-fable-5-1` and the Reviewer runs on Codex with `gpt-5.6-sol` (Section 4.1); these agents performed the literature search, method implementation, experimental design, and result interpretation reported in Sections 3–4, and their outputs are presented as experimental results rather than as contributions of the authors. (ii) LLM-based assistants helped edit and restructure the manuscript text and prepare figures. The authors reviewed all AI-assisted text, checked the reported numbers against the experiment logs, and take full responsibility for the content of this paper.

REFERENCES

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. ResearchAgent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6709–6738. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.342. URL https://aclanthology.org/2025.naacl-long.342/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *arXiv preprint arXiv:1911.11641*, 2019. URL https://arxiv.org/abs/1911.11641.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.07095.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019. URL https://arxiv.org/abs/1905.10044.

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 preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Kaiyuan Deng, Hangyu Zheng, Minghai Qing, Kunxiong Zhu, Gen Li, Yang Xiao, Lan Emily Zhang, Linke Guo, Bo Hui, Yanzhi Wang, Geng Yuan, Gagan Agrawal, Wei Niu, and Xiaolong Ma. From bits to chips: An LLM-based hardware-aware quantization agent for streamlined deployment of LLMs. *arXiv preprint arXiv:2601.03484*, 2026. URL https://arxiv.org/abs/2601.03484.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12284–12303. PMLR, 2024. URL https://proceedings.mlr.press/v235/egiazarian24a.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.17323.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

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. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025. URL https://arxiv.org/abs/2502.13138.

Jinuk Kim, Marwa El Halabi, Wonpyo Park, Clemens JS Schaefer, Deokjae Lee, Yeonhong Park, Jae W. Lee, and Hyun Oh Song. GuidedQuant: Large language model quantization via exploiting end loss guidance. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 30011–30037. PMLR, 2025. URL https://proceedings.mlr.press/v267/kim25d.html.

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 *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35532–35550. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1799. URL https://aclanthology.org/2025.emnlp-main.1799/.

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 on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2405.16406.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023. URL https://arxiv.org/abs/2303.17651.

Vladimir Malinovskii, Denis Mazur, Ivan Ilin, Denis Kuznedelev, Konstantin Burlachenko, Kai Yi, Dan Alistarh, and Peter Richtárik. PV-Tuning: Beyond straight-through estimation for extreme LLM compression. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/091166620a04a289c555f411d8899049-Abstract-Conference.html.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. URL https://arxiv.org/abs/1609.07843.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018. URL https://arxiv.org/abs/1809.02789.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019. URL https://arxiv.org/abs/1907.10641.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.320. URL https://aclanthology.org/2025.findings-emnlp.320/.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.13137.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. AutoML-Agent: A multi-agent LLM framework for full-pipeline AutoML. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 60099–60146. PMLR, 2025. URL https://proceedings.mlr.press/v267/trirat25a.html.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48630–48656. PMLR, 2024a. URL https://proceedings.mlr.press/v235/tseng24a.html.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. In *Advances in Neural Information Processing Systems*, volume 37, 2024b. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6de2e84b8da47bb2eb5e2ac96c63d2b0-Abstract-Conference.html.

Albert Tseng, Zhaofeng Sun, and Christopher De Sa. Model-preserving adaptive rounding. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2505.22988.

Mart van Baalen, Andrey Kuzmin, Ivan Koryakovskiy, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. GPTVQ: The blessing of dimensionality for LLM quantization. *arXiv preprint arXiv:2402.15319*, 2024. URL https://arxiv.org/abs/2402.15319.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/papers/Wang_HAQ_Hardware-Aware_Automated_Quantization_With_Mixed_Precision_CVPR_2019_paper.pdf.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025. URL https://arxiv.org/abs/2504.08066.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024. URL https://arxiv.org/abs/2405.15793.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019. URL https://arxiv.org/abs/1905.07830.

A Implementation and Resources

Task and isolation. Each run starts in an isolated workspace and home directory, without native agent memory from another run. The agent selects calibration and recovery data, establishes baselines, and keeps an append-only research log. The standalone condition uses the same Explorer prompt but removes the final *Independent reviewer* section (Appendix F). Native coding-agent delegation is not disabled; "standalone" denotes the coding-agent system without our independent Reviewer, rather than a single model call.

Review calls and stopping. Each review starts a fresh Codex process with web search enabled, `gpt-5.6-sol`, and high reasoning effort. The Reviewer receives the task, log path, and current request, and is instructed to read the whole log without inspecting code. Its response is appended to the log. Intermediate advice is non-binding; disagreements permit at most three exchanges before the Explorer records a decision and continues. A stop hook requests final review, including the Explorer's stopping statement. A final line of `APPROVE` permits normal exit; otherwise the Explorer must continue. Failed review calls do not count as approval. An operational safeguard allows exit after more than 25 consecutive stop checks; this is not a research-approval decision.

Accounts and run accounting. The study used two Claude Code Max $20\times$ accounts and one Codex Pro $5\times$ account. These are subscription configurations, not measured per-run token or compute costs. Each quantitative configuration has one independent research trajectory. The task imposes no fixed elapsed-time or token budget. Evaluation-item bootstraps and repeated training within a trajectory do not constitute independent repetitions of the agent search.

Evaluation and storage. Benchmark results are available as feedback during exploration; the reported scores describe the selected artifacts, not an untouched held-out assessment of the search procedure. All seven tasks use zero in-context demonstrations and the metrics in Section 4.1. Effective bit width counts codes and auxiliary storage for quantized Transformer-block linear weights. Embedding and output-head weights remain high precision and are excluded from this denominator. They may still be updated during recovery: remaining high precision does not imply being frozen.

B Full Results with Effective Bit Widths

Table 4 expands Table 2 with individual task scores and effective bit widths, rounded to three decimal places. Rounding is for display only; budget checks use the unrounded storage measurements.

Table 4: Nominal 2-bit weight-only quantization with high-precision embedding and output head. Bits reports effective storage per quantized weight, including auxiliary storage and excluding embedding and output head. PPL is WikiText-2 perplexity at context 2048. Zero-shot scores are percentages: ARC-E/C, HellaSwag (HS), PIQA, and OpenBookQA (OBQA) use normalized accuracy; WinoGrande (WG) and BoolQ use accuracy. AVG is the unweighted seven-task mean, computed before rounding. Lower PPL and higher accuracy are better. Bold marks the highest quantized AVG for each model. Baseline order follows increasing mean AVG across the three models; the same order is used in every block.

Method	Bits	PPL ↓	ARC-E	ARC-C	HS	WG	PIQA	OBQA	BoolQ	AVG ↑
Llama-3-8B										
16-bit reference	16.000	6.14	77.78	53.33	79.17	72.69	80.79	45.00	81.38	70.02
AWQ	2.140	1.70×10^{6}	25.00	24.74	26.52	49.01	51.03	29.00	62.17	38.21
OmniQuant	2.141	332.84	27.27	22.95	27.43	50.43	52.10	27.40	37.80	35.05
SpinQuant	2.125	25.96	41.92	24.74	37.53	55.56	60.07	26.60	63.64	44.29
AMQ	2.191	81.33	45.70	42.60	42.00	55.90	63.80	28.60	46.70	46.47
GPTVQ	2.133	11.91	54.97	29.78	54.51	60.30	66.81	32.60	69.76	52.68
YAQA	2.006	7.75	73.91	45.65	74.61	70.56	77.58	42.20	79.60	66.30
QTIP	2.006	8.05	73.30	47.40	74.10	69.70	77.80	40.60	79.00	65.99
Standalone agent	2.007	7.58	71.76	47.44	75.42	70.24	78.89	41.80	79.82	66.48
Explorer–Reviewer	2.001	7.56	76.50	49.10	75.30	72.40	79.10	42.00	77.10	**67.36**
Qwen3-8B										
16-bit reference	16.000	9.73	80.89	56.74	74.90	68.03	77.80	41.60	86.61	69.51
AWQ	2.140	1.37×10^{9}	25.93	27.47	26.02	49.80	51.74	29.60	62.17	38.96
OmniQuant	2.141	25.62	56.44	33.11	46.08	58.01	66.21	32.00	72.91	52.11
SpinQuant	2.125	15.38	46.55	30.03	50.19	60.38	64.74	29.60	75.90	51.06
AMQ	2.201	40.82	51.73	31.14	48.51	55.01	66.43	29.80	69.20	50.26
GPTVQ	2.133	14.63	55.51	37.63	55.26	63.22	68.88	32.60	78.26	55.91
YAQA	2.006	10.44	78.62	51.79	71.41	67.96	76.06	39.80	85.87	67.36
QTIP	2.006	10.67	79.50	53.24	71.52	68.35	76.44	41.40	85.47	67.99
Standalone agent	2.007	9.99	78.87	52.99	72.33	69.69	76.88	42.00	85.50	68.32
Explorer–Reviewer	2.007	10.52	80.43	55.80	71.99	69.53	76.88	41.80	86.02	**68.92**
Mistral-7B										
16-bit reference	16.000	5.25	80.30	54.78	80.96	74.35	81.94	44.40	83.76	71.50
AWQ	2.140	37977	26.26	29.18	25.80	49.72	51.09	28.00	37.83	35.41
OmniQuant	2.141	100.65	32.37	23.38	33.59	49.96	55.06	26.20	52.14	38.96
SpinQuant	2.125	11.97	56.19	31.48	53.15	59.75	66.76	30.00	69.11	52.35
AMQ	2.191	14.87	59.76	33.02	60.05	61.40	72.47	35.60	69.48	55.97
GPTVQ	2.133	10.82	51.39	30.03	47.28	59.51	64.74	32.00	60.12	49.30
YAQA	2.006	5.82	76.30	49.74	77.64	72.30	79.71	42.80	82.02	68.64
QTIP	2.006	5.89	76.09	50.94	77.99	71.82	80.20	42.40	81.28	68.67
Standalone agent	2.001	5.79	75.34	48.98	78.22	71.98	80.74	44.20	81.62	68.73
Explorer–Reviewer	2.009	5.81	76.21	50.21	78.27	71.74	80.85	43.20	81.98	**68.92**

C LLAMA RECIPE: PROFILING AND ALLOCATION

Token-gradient statistics. For token t at a linear module, let x_t be the input activation and g_t the gradient of the teacher's observed-label next-token cross-entropy with respect to the module output. The profiling weight is $q_t = \|g_t\|_2^2/d_{\text{out}}$. Figure 4(a) shows the fraction of $\sum_t q_t$ carried by the largest 327 weights among 32768 tokens: 16 SlimPajama sequences of length 2048, sampled with seed 0. Each marker represents a module/block, not an independent repeat. The plotted values are the integer percentages recorded by the profiling script. Equal weights give $327/32768 \approx 1\%$.

Weighted reconstruction. The recipe uses the reconstruction Hessian

$$H_w \propto \sum_t \sqrt{q_t}\, x_t x_t^\top .$$

Square-root tempering reduces the concentration of the weights without changing their ordering. This is a loss-informed reconstruction proxy, not the exact Hessian of the end-to-end loss.

Mixed-precision allocation. For reference quantization error $E = W_q - W$, the local rowwise cost is estimated as $\sum_t g_{tr}^2 (Ex_t)_r^2$ and aggregated in groups of 16 rows. It uses teacher activations and gradients, rather than measuring an end-to-end loss change after quantization. The allocator combines these costs with codebook distortion estimates and searches a storage penalty to meet the global budget. Figure 4(b) shows the mean code bits per weight across 16-row groups for each linear module in each of the 32 blocks. Code rates describe the weight codes; scales, transforms, codebooks, rate tags, and any rank corrections are counted separately in effective storage.

D REVIEW CASE STUDIES: LOG EXCERPTS

The following excerpts document the cases in the main text. Each box shows the Explorer's initial judgment, the relevant Reviewer response, or the Explorer's subsequent action and result. Wording is preserved; Markdown emphasis is rendered as text, and [...] marks omissions. Experiment identifiers refer to the original logs.

D.1 QWEN: ADAPTING RECOVERY DATA

The initial comparison changed the quantization objective but improved PPL rather than AVG. The Reviewer suggested task-oriented recovery data. The Explorer tested token-level teacher targets on QA-formatted text, not the answer-choice/ranking loss initially suggested by the Reviewer; the subsequent plan review explicitly distinguishes the two. In the excerpts, A/B denote the plain- and two-sided-objective pipelines; suffix 3 includes recovery fine-tuning.

Explorer: current diagnosis (log §10.5)

Conclusion for the roadmap: at the PTQ stage the choice of rounding objective is a ppl/KL lever, not an AVG lever; the AVG lever has to come from elsewhere (fine-tuning stages, and possibly from what the calibration objective targets).

Reviewer: direction and test scope (R3/R4)

R3, CONCLUDE: If A3/B3 remain tied or favor A, a proportionate next test is general-text KL versus a mixture adding teacher answer-choice score/ranking distillation on disjoint QA-style data. Do not infer benefit from generic ranking-KD papers without this controlled test.

R4, PLAN: Standard token-level soft-target CE on QA-formatted sequences does not test the roadmap's proposed answer-choice/ranking distillation. A negative E3 only rejects this QA-prompt CE mixture, not task-aware distillation generally.

Explorer: experiment and result (log §17)

E3 = A2 + e2e on 512 RedPajama + 256 QA-style sequences (SciQ/CSQA/QASC train, lm-eval prompt format, teacher soft targets only) $\times$ 4 epochs = same update count as E2.

Selected columns from the recorded result table:

Arm	Recovery data	PPL@2048	AVG
E2	768 RP $\times$ 4	10.541	68.18
E3	(512 RP +256 QA) $\times$ 4	10.614	68.58

Both arms start from the same layer-wise-recovered checkpoint, use 4096-token sequences, and train for four epochs. QA sequences come from the training splits of SciQ, CommonsenseQA, and QASC, not the seven evaluated tasks. Both retained checkpoints are from epoch 3. The general-text run's last epoch resumed with a fresh Adam optimizer after preemption, so nominal update counts match but optimizer histories are not identical. The comparison supports the tested recovery recipes, not a perfectly isolated estimate of the data effect.

D.2 LLAMA: PREVENTING VALIDATION LEAKAGE

B2 denotes the original recovery dataset of 640 sequences, with the last 128 reserved for validation. C2 expands this dataset to 2560 sequences. Keeping only the enlarged dataset's last 128 as validation would move the original validation examples into training.

Explorer: proposed split (R2 batch plan)

C2 4$\times$ data: 2560 $\times$ 4096 sequences (seed 0 sampling; the first 640 coincide with B2's devset; the same last-128 validation split), 2 epochs, otherwise identical [. . .]

Reviewer: leakage identified (R8, PLAN)

Fix C2's split definition. "First 640 coincide with B2" conflicts with using the expanded set's last 128 for validation. Preserve B2 indices 512–639 as the identical validation set and explicitly exclude them from C2 training; otherwise B2 validation examples enter C2 training and the comparison is biased.

Explorer: corrected split (decision after R8)

C2's devset is built so that B2's exact validation sequences (indices 512–639 of the seed-0 draw) remain the validation split and are excluded from C2's training: C2 devset = seed-0 draw of 2560 sequences with rows 512–639 moved to the end (2432 train + 128 val; verified that the first 640 rows of the 2560-row draw equal the B2 devset) [. . .]

D.3 MISTRAL: TESTING A DISMISSED DIRECTION

R7 proposes spending unused bit-budget headroom on five sensitive value projections. The Explorer initially rejects it using a predicted gain from noise-injection measurements; the Reviewer requests a direct test.

Explorer: initial decision (log §25)

R7 retired on this quantitative evidence (expected gain below primary-metric resolution at this budget) [. . .]

Reviewer: direct evidence needed (R13, CONCLUDE)

The five-v_proj configuration fits the budget and uses supported K=3 kernels. With an estimated 7% reduction in remaining KL and only about six GPU-hours, its decision value is high enough that retirement on the surrogate alone is premature.
A direct mixed-rate endpoint is needed to learn whether this improves the required seven-task AVG.

Explorer: revised decision and result (log §§25, 27)

accepted — R7 is reinstated as a run: the noise slope is only an allocation surrogate [. . .]
[. . .] spending the 0.0035-bpw headroom on the five most sensitive v_proj layers (K=3) gives a small, consistent perplexity/KL improvement (Wiki2 -0.010, C4 -0.006, val CE -0.0007) and no measurable AVG change ($+0.07 \pm 0.37$) [. . .]

The resulting recipe changes five value projections from 2 to 3 bits, requantizes the affected blocks, and applies the same recovery schedule. Its effective rate is 2.00954 bits per quantized weight. Relative to R6, PPL changes from 5.823 to 5.813 and AVG from 68.55 to 68.62. The small accuracy difference is descriptive; the case demonstrates a change in research decision followed by a direct measurement.

E Failure Cases in Standalone Exploration

These cases from early Llama-3-8B exploration underlie the four failure modes in Section 3.1. PPL uses WikiText-2 with 2048-token contexts.

E.1 Proposal: Overlooked Prior Work

The agent found that the top 1% of tokens carried 55–99.7% of output-gradient energy, depending on the layer. It proposed a gradient-weighted reconstruction Hessian, with tempering and uniform mixing to control concentration. After tuning these choices, no-recovery PPL decreased from 10.03 to 9.50; the agent presented the mechanism as an algorithmic innovation.

GuidedQuant (Kim et al., 2025) already uses squared output gradients to weight reconstruction within channel groups. YAQA (Tseng et al., 2026) uses a related gradient-weighted input factor in a two-sided objective. These are relevant precedents, not identical methods. The initial search had encountered both, but the concrete proposal was not checked against them. Tempering and mixing may still merit testing; the missed opportunity was to start from the closest existing objectives, implementations, and evidence instead of rediscovering the underlying mechanism.

E.2 Plan: Inadequate Experimental Design

The agent proposed a rank-1 correction for quantization error along an influential input direction. A value-projection-only pilot improved PPL from 7.109 to 6.409; a whole-model test under an unweighted recipe also improved PPL, from 9.530 to 9.102. However, the eventual recipe used a stronger weighted Hessian. Neither test established whether rank-1 still added value in that setting.

Later matched controls, all at 2.000 bits per quantized weight and without recovery fine-tuning, gave:

	Plain Hessian	Weighted Hessian
With rank-1	8.511	8.406
Without rank-1	8.570	8.365

The correction helps with the plain Hessian but not with the weighted Hessian in this comparison. No matched post-recovery control was available. The issue is not that rank-1 never helps, but that evidence from a weaker recipe does not establish its value in the final one.

E.3 Conclusion: Unsupported Conclusions

The agent tested a tile-wise trellis refinement inspired by PV-Tuning (Malinovskii et al., 2024). Each iteration estimated gradients from 16 sequences and updated about 25% of the weight tiles. Training KL rose from 0.155 to 57.7 within ten iterations. Diagnostics suggested noisy gradients and large updates, but the agent generalized the result to abandonment of end-to-end-gradient-driven discrete refinement, stating that it "provably fails at feasible batch sizes."

The record did not systematically test smaller updates, fewer updated tiles, or acceptance safeguards. It supports rejecting the implemented setting, not the entire direction; it also does not establish that an untested variant would succeed.

E.4 FINAL: PREMATURE STOPPING

The agent proposed both Fisher-weighted codeword assignment and sensitivity-based bit allocation. It pursued allocation and recovery, but did not test the weighted-assignment component; an optional implementation interface remained disabled. After several later experiments failed, the agent declared convergence and estimated less than 0.05 PPL of remaining improvement.

The earlier proposal had neither a result nor an explicit reason for retirement. Exhausting recent experiments did not resolve that open item in the roadmap. This motivates checking the complete research record at final review, rather than requiring an exhaustive search of every conceivable method.

F AGENT PROMPTS

We provide the two prompts from the Qwen3-8B configuration: the Explorer task prompt and the Reviewer template. The standalone agent receives the *same Explorer prompt with the final Independent reviewer section removed*; all preceding research instructions are unchanged. The suffix is marked explicitly below.

The prompt wording is preserved, with Markdown rendered for readability and launcher comments omitted. The initial user-provided reference scores remain as supplied to the agent; they are not substituted for the measured reference scores in Table 4. In the Reviewer template, `{{MODE}}` selects the checkpoint, `{{LOG_FILE}}` supplies the log path, `{{TASK}}` inserts the Explorer task, and `{{NOTE}}` inserts the current review request.

F.1 EXPLORER PROMPT

Explorer prompt: Qwen3-8B

You are an autonomous ML research agent with access to single GPU, web access, and full permission to write and run code.

Your task is to achieve the best quantization quality you can for the Qwen3-8B model located at: /models/qwen3-8b

There are no constraints on the method family, as long as it can achieve the goal and remains practically deployable for GPU inference (see requirement below). The goal is to maximize average zero-shot accuracy subject to an average budget of at most 2.01 bits per weight. Evaluate ARC-Easy, ARC-Challenge, HellaSwag, WinoGrande, PIQA, OpenBookQA, and BoolQ with zero few-shot examples. Use `acc_norm` for ARC-Easy, ARC-Challenge, HellaSwag, PIQA, and OpenBookQA; use `acc` for WinoGrande and BoolQ. AVG is the unweighted arithmetic mean of these seven scores. Report all seven scores and AVG. WikiText perplexity (seqlen=2048, lower is better) is a secondary metric; report it separately rather than combining it with AVG. Set up and verify the evaluations yourself, using a consistent evaluation protocol for comparisons and recording its version and settings.

User-provided FP16 reference results for the unquantized model are WikiText PPL = 9.73 (seqlen=2048) and zero-shot AVG = 69.44% across the seven tasks above. Reproduce the FP16 evaluation to validate your setup, record your measured results, and investigate material discrepancies before relying on quantized-model comparisons.

Identify and reproduce competitive published methods, and aim to outperform the strongest relevant baselines under comparable conditions. Reproductions, hyperparameter tuning, combinations of existing methods, and new designs are all acceptable. Choose what to pursue and deliver based on final model quality and practical value.

A requirement: hardware realism. Your quantized format must remain practically deployable on GPUs — dequantization must be cheap, and any lookup structures must fit in fast on-chip memory. Published methods often cap certain design parameters for exactly this reason. Relaxing such a cap because "we don't measure speed here" is discarding a constraint that prior work deliberately respects.

When you need an existing method — as a baseline, a starting point, or a component — run its official implementation: clone the repository and record the commit. Re-implementations introduce unnecessary error and make comparisons against published numbers unreliable. Two rules follow from this:

- **Reproduce before you compare.** If the paper reports a number for *this* model, reproduce it *under the evaluation protocol the paper reports*, including context length for PPL and task metrics and few-shot settings for accuracy, and record the deviation; until that check is in your log you may not claim to match or beat that method. Also check any user-provided reference results under their stated protocol. If the paper reports no number for this model or protocol, say so explicitly and label every comparison to it as "against my own run of their code, not against a published number" — do not bridge protocol differences with estimated conversions. Use the task's evaluation protocol for the final comparison, and distinguish that comparison from the paper-protocol reproduction.
- **Re-implement only when you must.** You may re-implement a method only when the official code structurally cannot do what you need; then say which capability is missing, and show that your implementation matches the official one at a setting where both apply, with the same calibration data and the same pipeline stages enabled.

To get there, follow this research methodology:

1. Literature survey. Search for and read the relevant papers on low-bit LLM quantization, especially work at or near the 2-bit regime. For each paper, summarize in your log its core insight (why does the

Explorer prompt: Qwen3-8B (continued)

method work?) and its core mechanism (what does it actually do?), separately. The insight matters more than the mechanism — insights can be recombined in ways the original mechanisms cannot. Also note any design limits the paper imposes for hardware reasons — these limits apply to you too.

2. Profiling and diagnosis. Profile the model where it helps explain failures or identify headroom: per-layer weight distributions, outlier structure, layer sensitivity, and where quantization error actually concentrates at this bit budget. Combine the survey, baseline results, and profiling to form explicit hypotheses about what could improve quality at 2 bits. Write these hypotheses down before implementing the corresponding changes; baseline reproduction need not wait for comprehensive profiling.

3. Roadmap. After the initial survey and diagnosis, write a roadmap section in your log: which modules you plan to test which methods on, why (each item traceable to the survey or your diagnosis), what result you expect, what a success would unlock next, and what a failure would rule out or fall back to. This roadmap is the backbone of the whole exploration, but it is not frozen — it evolves with the evidence. Keep its history intact: once written, never rewrite or delete the original text; record every change as a dated entry in a "Roadmap revisions" list beneath it (what changed, why, and which result triggered it). You may strike through a retired item in place (marking it "superseded by Rev N") so the current state stays readable, but the words must remain. When picking the next experiment, consult the roadmap rather than only the most recent results, and prioritize the most promising improvements, whether they require refining a successful approach or exploring another direction.

4. Propose and implement. Pick the next item from the roadmap and instantiate it, together with your hypotheses, into a concrete proposal, and implement it to test whether your reasoning holds. The roadmap says which module and method family and why; the proposal is the specific mechanism, implementation plan, and expected result. If a new idea arises outside the roadmap, you may pursue it, but first add it to the roadmap as a revision so the roadmap remains the complete map of the search. Design decisions should be traceable to the literature or your diagnosis. Before committing to any proposal, explain why it could improve final model quality and whether it respects the hardware constraints.

5. Iterate. Use the experimental results to revise your hypotheses and refine or replace your approach. Failed hypotheses are valuable — record them and what they ruled out, then continue the loop.

If an approach underperforms the strongest reproduced baseline, report both honestly and retain the better model. Scope conclusions to the evidence: an improvement or failure in a limited experiment does not automatically establish the value of an entire direction for the final model.

There is no time limit. Stop when there is sufficient experimental, applicable literature, or profiling evidence to justify no further investment in the major identified directions with a reasonable prospect of improving final model quality. You need not prove global optimality or exhaust every imaginable method. When you stop, state explicitly what you tried, what the last several attempts changed, and why the evidence supports retiring the remaining directions rather than merely explaining away failures. Report honestly whether you surpassed the strongest relevant baselines you established.

Waiting for jobs to finish is not a reason to end your turn. While they run, do the work that does not need a GPU: analyse the results already in hand, submit the next experiment plan for review, extend the survey, update the roadmap, write up. End your turn only when you believe the search has converged.

Your final deliverables should include:
- The final quantized model.
- The corresponding loading / inference code.
- A log documenting your exploration process, including methods tried, results, failed attempts, and final decisions. Keep this log at `docs/research-log.md` and update it as you go; it is the record a human will review afterwards. For every run you record, log the exact command line, the calibration / fine-tuning data file it used, and the number and identity of the trainable parameters. Where these differ between runs, say so where you draw the comparison, and distinguish comparing complete approaches from attributing an improvement to a particular change.
- A final summary document that reorders the log by topic: the delivered method, the results table, the evidence behind the main design decisions, the negative results, and the convergence argument. The log itself stays append-only; the summary is what a reader starts from.

I have granted you all necessary permissions. From this point on, make all subsequent decisions autonomously.

[Reviewer-only suffix; omitted for the standalone agent.]

Independent reviewer

An independent reviewer is available: a separate model with no access to this conversation. It reads your research log and searches the literature; it does not read your code. It knows the task statement above. Its

Explorer prompt: Qwen3-8B (continued)

role is a mentor on research direction and evidence, helping you decide what is worth pursuing or retiring while leaving implementation decisions to you. Consult it at three moments in the methodology:

- **Step 4, before committing to a proposal** — after you have written the proposal and why it could improve final model quality in the log, run:

```
review/run-review.sh propose "<the proposal, its motivation from
literature or diagnosis, and the expected improvement>"
```

It provides relevant literature and implementations, what those works concluded, and what can be borrowed from them to guide the proposal.

- **Steps 4–5, before launching a batch of experiments** — once you have written down what you are about to run (the arms, what each one changes relative to the others, the data and seeds, and what result you expect), and before the jobs start, run:

```
review/run-review.sh plan "<the arms, what each one changes, what you
expect>"
```

It judges whether the plan as written can support the conclusion you intend to draw from it.

- **Step 5, when a hypothesis is resolved** — once you have written the conclusion for an attempt in the log (did it help, what did it rule out, should the direction be continued or retired), run:

```
review/run-review.sh conclude "<the conclusion and where the evidence
is in the log>"
```

It checks whether the evidence supports the conclusion and the decision to continue or retire the direction, whether comparisons are fair, and whether the metric, budget, and hardware requirements are met.

Its reply is appended to the log automatically and printed for you. It is an opinion, not an instruction: decide for yourself what to do with it and record your decision in the log — including what you decline, and why. If you think an opinion is wrong, or was based on a misreading of the log, you may put your reasoning back to the reviewer and ask again on the same point; at most three exchanges per point, after which record the disagreement and your decision and carry on. A proposal whose mechanism or objective has materially changed needs a new propose review. Every batch of experiments needs a plan review, including batches inside a proposal that was already reviewed.

When you decide the search has converged and try to stop, the reviewer reads the log once more and checks your convergence argument and the task requirements. If it answers CONTINUE you get its reasons back; if it answers APPROVE, you stop.

F.2 Reviewer Prompt

Reviewer prompt: shared template for all four checkpoints

Independent Reviewer — mode: {{MODE}}

You are an independent reviewer and research mentor for a model-quantization project. A researcher agent (Claude) works autonomously in this repository and keeps a research log. You are called at a few points to give a second opinion. Your purpose is to help direct effort toward the most promising improvements in final model quality. You have no memory between calls; the log is your context.

Your scope

- Read the whole research log first: `{{LOG_FILE}}`. Your earlier opinions appear in it as blocks marked `<!- review:R... ->`.
- You may use web search to check the literature.
- Do NOT review or read source code. You judge research direction, evidence, and conclusions — not implementation. Leave implementation and detailed experiment choices to the researcher. If a conclusion looks unsupported, explain the gap and what evidence would resolve it.
- Judge whether the evidence is sufficient for the decision at hand: continue, retain, or retire a direction. Scrutinize both claimed improvements and reasons for giving up; the scope of a conclusion should not exceed that of its evidence. Prioritize issues that could change the research decision, and keep requests for additional evidence proportionate to their decision value and cost.
- You give an opinion; the researcher decides what to do with it. Be concrete and brief (aim for under 30 lines). Cite literature with URLs.

Reviewer prompt: shared template for all four checkpoints (continued)

- Beyond the question you were asked: if reading the log as a whole reveals something *earlier* that now looks wrong — a conclusion drawn from evidence that does not apply at its claimed scope, a comparison that a later result makes unfair, a direction retired for a reason that no longer holds, a claim contradicted by a later number — raise at most two such items and say why they matter now.

The task given to the researcher

<!- TASK START ->
{{TASK}}
<!- TASK END ->
Hold the researcher's proposals and conclusions to the requirements stated there (e.g. model quality, hardware constraints, the metrics and budget) — not to standards of your own.

The researcher's message for this review

<!- RESEARCHER MESSAGE START ->
{{NOTE}}
<!- RESEARCHER MESSAGE END ->

What to do in mode {{MODE}}

PROPOSE — the researcher is about to start a new direction.
Search the literature for methods and experience that can guide this direction. For each relevant work (at most ~5): URL, official implementation if available, one line on what they did, one line on what they concluded, and what the researcher can borrow or learn from it. If the literature already answers the question, explain the answer and whether its conditions apply here. Assess whether the direction is promising relative to the alternatives in the log. Do not decide for the researcher.

PLAN — the researcher is about to launch a batch of experiments.
Judge whether the plan can inform the next research decision. Are the comparisons appropriate to the intended conclusion, and can the outcomes justify continuing or retiring the direction? Distinguish testing whether a complete approach works better from attributing an improvement to one component; only the latter requires isolating that component. Check whether the measurements and assumptions can support conclusions about final model quality. Say plainly whether the plan as written can support the conclusion the researcher intends to draw, and prioritize any gap that would change the decision.

CONCLUDE — the researcher finished an attempt and states a conclusion and what to do with the direction.
Check the conclusion against the evidence in the log: do the numbers support it, are the comparisons fair (comparable settings, official baselines), and is the improvement valuable enough to justify further investment? For negative results, does the evidence justify retiring the direction, or only rejecting the tested configuration? Check whether the result applies at the scope being claimed. Also look at the log as a whole: does this conclusion contradict or depend on earlier results; if directions are being combined, was the combination itself tested. Say plainly whether the conclusion and next decision are supported, and if not, what evidence would change that judgment.

FINAL — the researcher wants to stop, claiming the work is done.
Judge the convergence argument against the task's stopping condition. Read the roadmap against what was actually done: have the major identified promising directions been explored or reasonably retired on experimental, applicable literature, or profiling evidence? Check whether retirement reasons apply to the directions they dismiss, revisit conclusions that fail the CONCLUDE checks above, and identify at most one or two clearly valuable directions the plan missed. Check the final results and deliverables against the task requirements. Do not require global optimality or an exhaustive search. For CONTINUE, identify the unresolved direction or task requirement, explain why the existing evidence is insufficient, and why addressing it is worth the cost; the mere possibility of another experiment is not enough. The task requires fully autonomous decisions: if the researcher stopped to ask a question rather than because the work is done, answer CONTINUE and tell it to decide on its own and carry on.
End your reply with exactly one of these as the LAST line: `APPROVE` (the researcher may stop) or `CONTINUE` (list what to address first).