STEERDISTILL: QUANTIZING LLM AGENTS VIA ONLINE TEACHER-STEERED TRAJECTORY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantization makes multi-turn LLM agents brittle: a single erroneous decision alters every subsequent interaction and can derail the task. We observe that quantization rarely breaks an agent's overall plan; instead, it corrupts a few executable decisions, such as a tool call with the wrong interface or a misgrounded argument. Existing recovery methods either learn from fixed full-precision data that misses the states a quantized agent reaches, or supervise the student's own rollouts without locating the error and rely on later rollouts to avoid it; neither corrects the turn at which a rollout goes wrong. We introduce SteerDistill, a quantization-aware distillation method built on *online teacher steering*. When a quantized student's rollout diverges from a successful full-precision trajectory, the teacher issues a steering prompt at the divergent turn; the student regenerates that turn itself, the environment verifies the correction, and the rollout continues from the corrected state. The steered behavior is then distilled into the student without the prompt, so no teacher is needed at inference. Across five agentic benchmarks and three models with 4-bit weights, SteerDistill outperforms the strongest quantized baseline by 1.67–2.18 average points and stays within one point of BF16. On τ^2-Bench, it also exceeds quantization-aware reinforcement learning with 67% fewer training GPU hours.

1 INTRODUCTION

Large language model (LLM) agents call tools and act on environment feedback over many turns until a task is complete (Yao et al., 2023; Yang et al., 2024; Wu et al., 2024). These repeated model calls make inference cost a central deployment concern (Wang et al., 2024; Zheng et al., 2024; Li et al., 2024), and low-bit quantization is a standard way to reduce it (Frantar et al., 2023; Xiao et al., 2023; Lin et al., 2024). Multi-turn interaction, however, makes quantization error hard to contain: one imprecise decision changes the environment and every observation that follows, so a small per-step error can compound into task failure (Putta et al., 2024; Qi et al., 2025).

Inspecting failed quantized rollouts, we find that many failures keep the agent's overall plan intact and stem from a single corrupted executable decision: the wrong interface, an incomplete call sequence, a dropped invocation, or a misgrounded argument, while the surrounding turns remain valid (Section F.1). The damage is thus concentrated on a few *divergent turns*, where the quantized rollout departs from successful full-precision behavior, rather than spread across the trajectory. Recovering the agent therefore calls for correcting these turns rather than re-supervising every turn uniformly.

Existing recovery methods do not provide effective supervision at these turns. (i) Offline quantization-aware training and distillation, such as EfficientQAT and BitDistiller, learn from fixed full-precision data (Chen et al., 2025; Lee et al., 2026; Du et al., 2024; Xin et al., 2026), which rarely covers the states a quantized agent drifts into during multi-turn interaction (Agarwal et al., 2024). (ii) On-policy methods train online on the student's own rollouts, but their supervision leaves each rollout as it is: QeRL assigns one reward to the whole trajectory (Huang et al., 2026), and on-policy distillation supervises every token alike (NVIDIA, 2026; Agarwal et al., 2024). Neither locates the divergent turn or shows the student a correct continuation from it; both rely on the parameter update to make the student find one by itself in later rollouts. The student rarely does: on BFCL V4 multi-turn tasks, its turn-level Pass@4 at the divergent turns is only 9.6%, whereas the teacher steering of

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.

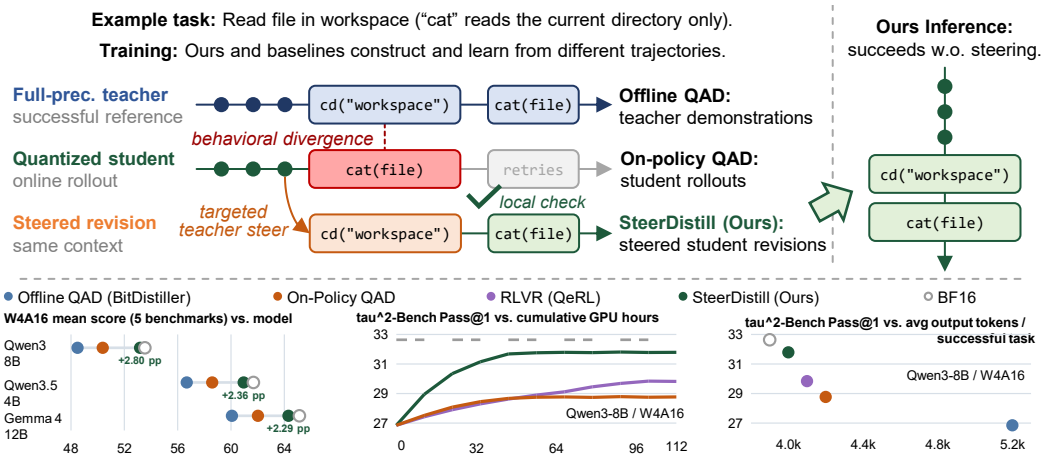


Figure 1: Overview of SteerDistill. The top shows an example of teacher-steered trajectory revision: after the quantized student makes an incorrect `cat(file)` call, a targeted teacher steer is added in the same pre-error context, the student regenerates a locally verified correction, and the corrected behavior is internalized for unassisted inference. The bottom summarizes W4A16 mean score and τ^2-Bench efficiency for the Qwen3-8B setting, with Offline QAD (BitDistiller), On-Policy QAD, RLVR (QeRL), SteerDistill (Ours), and BF16.

our method, SteerDistill, raises it to 82.3% during training.[1] Recovering a quantized agent therefore requires two things that existing methods do not provide: correct behavior constructed at the divergent turns of the student's own rollouts, and a way to make the student produce that behavior without assistance at inference.

Contribution. We introduce **SteerDistill**, a quantization-aware distillation (QAD) method for LLM agents built on *online teacher steering* (Figure 1). Two designs address the two requirements above. (i) *Teacher-steered trajectory revision* aligns each failed student rollout with a successful teacher reference, locates the divergent turn as the first action mismatch that changes the task-relevant environment state, and gives the student a steering prompt at that turn. The student regenerates the turn itself; the correction is retained only if the environment confirms its effect, and the rollout resumes from the corrected state. Because the student writes the correction, the revised trajectory stays close to what it can produce. (ii) *Internalization of steered behavior* trains the student without steering prompts to match the frozen steered student at *intervention turns* (divergent turns with a retained correction) and to match the full-precision teacher elsewhere, so no teacher is needed at inference.

SteerDistill achieves the strongest quantized performance in our comparisons with lower training cost and shorter successful trajectories. **(1) Accuracy.** Across five agentic benchmarks and three models under W4A16[2], it exceeds the strongest quantized baseline by 1.67–2.18 points in average score and stays within one point of BF16; on Qwen3-8B at W2A16, the margin grows to 3.39 points (Table 1). **(2) Training cost.** On τ^2-Bench, with a comparable budget of 32 GPU hours it scores 2.91 points higher than On-Policy QAD; it exceeds QeRL's plateau with 67% fewer GPU hours and finishes 1.97 points higher (Figure 3a). **(3) Inference cost.** On τ^2-Bench, its successful tasks use 4.0K output tokens on average, close to BF16's 3.9K, versus 4.1K–5.2K for the quantized baselines (Figure 3b and Table 6). **(4) Design insights.** Ablations support intervening at the divergent turn, steering the student instead of a teacher takeover, retaining only locally verified revisions, and routing forward and reverse KL by turn (Table 2).

[1] For each divergent turn, we restore the interaction history and environment state, sample four responses for that turn only, and count the turn as solved if any response produces the correct action.

[2] WxAy denotes x-bit weights and y-bit activations.

2 Related Work

Capability Recovery for Low-Bit LLM Quantization. LLM quantization reduces inference cost through lower-precision computation, but can damage the decisions needed for multi-turn interaction. Post-training methods control rounding and scaling (Frantar et al., 2023; Lin et al., 2024), use equivalent transformations (Xiao et al., 2023; Shao et al., 2024a), or rotate representations (Ashkboos et al., 2024; Liu et al., 2025) to limit quantization error. For agentic inference, Mix-Quant (Lu et al., 2026a) accelerates prefilling with NVFP4 while retaining BF16 decoding. Quantization-aware training recovers low-bit performance through parameter updates, with EfficientQAT (Chen et al., 2025) and ReQAT (Lee et al., 2026) targeting efficient training and reasoning recovery. Other approaches distill full-precision predictions into quantized students, including LLM-QAT (Liu et al., 2024), BitDistiller (Du et al., 2024), and NVIDIA QAD (Xin et al., 2026); NeMo QARL (NVIDIA, 2026) and QeRL (Huang et al., 2026) extend quantized training to reinforcement learning. These methods supervise fixed data, whole trajectories, or all tokens uniformly; none targets the individual turns at which a quantized agent's rollout goes wrong.

On-Policy and Privileged-Context Distillation for Agents. On-policy distillation (OPD) supervises sequences generated by the student, keeping training contexts aligned with those encountered at deployment (Agarwal et al., 2024); agent methods organize this experience with trajectory curricula or adaptive supervision (Wang et al., 2026b; Zhou et al., 2026). Several approaches add corrections to student failures. Guided-OPD (Li et al., 2026) combines teacher and student turns with forward and reverse KL, FutureBridge-OPD (Chen et al., 2026) screens teacher corrections with short student continuations, Agent SCoRe (Lyu et al., 2025) trains on teacher-corrected trajectories before short-horizon RL, and SKILL-KD (Shi et al., 2026) stores rerun-validated textual patches in an external skill library. A related line distills from privileged context available only in training, such as reference solutions, feedback, skills, or state-matched references, into an unassisted policy (Zhao et al., 2026; Hübotter et al., 2026; Ye et al., 2026; Wang et al., 2026a; Lu et al., 2026b; Liu et al., 2026; Gou et al., 2026); EDGE-OPD (Lazaridis et al., 2026) and EDGE (Xie et al., 2026) route this supervision by token-level evidence or guided–unassisted reward gaps, and OVCSD (Xia et al., 2026) distills a verified continuation from the first aligned divergence. SteerDistill differs in where the correction comes from and how it is kept: the quantized student itself regenerates the divergent turn under a teacher steering prompt, the environment verifies its effect before the rollout continues, and the frozen steered student, rather than the teacher, supplies the forward-KL target at intervention turns, with full-precision reverse KL elsewhere.

3 Method

Overview. With the quantization operator fixed, SteerDistill seeks an unassisted quantized policy that preserves task success: teacher references and steering may shape the training responses, but at deployment the policy acts from its interaction history alone. Each training cycle has two designs (Figure 2). Teacher-steered trajectory revision corrects the current student's own failed rollouts at their behavioral divergences (Section 3.1); internalization of steered behavior then trains the student to produce the accepted corrections without steering (Section 3.2).

Setup. Let π_T be the frozen full-precision teacher and π_θ the trainable quantized student initialized from the teacher checkpoint. Here π_θ abbreviates $\pi_{Q(\theta)}$, with master weights θ and fixed quantization operator Q. At turn t, the student generates response y_t from history h_t. The history records the task, earlier responses, and environment observations; the response may include reasoning followed by an action or answer. Executing the action produces the observation for the next turn. During training cycle k, a frozen snapshot π_{θ_k} collects and revises trajectories. We update π_θ on the retained responses and use the resulting weights as the next snapshot θ_{k+1}. Collection runs with the actual low-bit runtime, whereas training uses fake quantization (Section A).

3.1 Teacher-Steered Trajectory Revision

For each unsuccessful student rollout, we seek a teacher reference verified to complete the same task from the same initial environment state. We keep this reference fixed throughout revision.

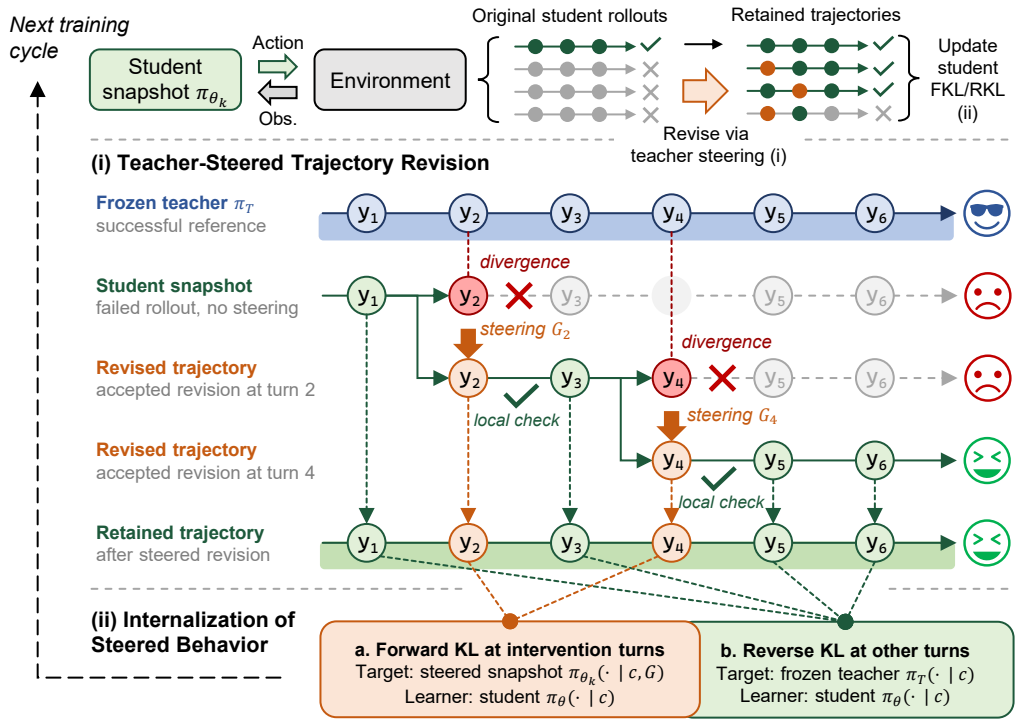


Figure 2: **Trajectory revision and internalization in SteerDistill.** (i) Teacher steering revises failed student rollouts against a fixed successful teacher reference. At each divergence, a steered student response that passes the local check replaces the failed one, and the rollout resumes without steering; gray dashed suffixes are discarded. Revised trajectories that fail later are still retained (top right, last row). (ii) Forward KL transfers steered-snapshot predictions to the unassisted learner at intervention turns; reverse KL provides frozen-teacher supervision elsewhere. Context c comprises the interaction history and retained response prefix, with all steering prompts removed. Only the forward-KL target additionally receives steering prompt G. Response indices denote positions within each row.

Successful student rollouts, and those for which no successful reference is available, enter training unchanged.

Localizing an intervention turn. We align student and teacher trajectories using canonicalized actions and their environment transitions. After the longest shared action–transition prefix, we select the first decision whose environmental effect differs from the reference. We intervene at this *behavioral divergence*, denoting the selected turn by $t^\star$, and regenerate its response from the start.

Constructing a steering prompt. The teacher compares the failed decision with the reference to formulate a *local steer*: an explanation of the discrepancy and a rule for choosing a correct action. To preserve relevant constraints across interventions, it combines this steer with a *trajectory-local steer memory* $\mathcal{M}$ into a single steering prompt $G_{t^\star}$. The memory holds the latest prompt that yielded a locally accepted response, carrying its accumulated guidance into the next intervention.

Generating and retaining steered responses. We restore the history and environment state before turn $t^\star$, sample $y_{t^\star} \sim \pi_{\theta_k}(\cdot \mid h_{t^\star}, G_{t^\star})$, and execute its action. An environment-specific verifier checks the resulting transition against the task-relevant conditions of the reference. If accepted, the response replaces the failed one, its prompt becomes the memory, and the student continues from the new feedback without steering. Otherwise, verifier feedback guides prompt refinement, and we retry from the same history and state up to the per-turn attempt budget. Failed attempts leave the memory unchanged.

After each accepted revision, we realign with the same reference to locate the next intervention. Successive revisions extend the aligned prefix, and we retain the version with the longest such prefix for training. The retained trajectory can therefore contribute verified corrections even when its final rollout does not complete the task. Algorithm 1 and implementation details appear in Section A.

3.2 Internalization of Steered Behavior

We train the student to reproduce the revised behavior without steering. At intervention turns, the target comes from the frozen student under steering; at other turns, we retain full-precision teacher supervision.

Within cycle k, both per-token losses use context c: the interaction history and retained response prefix, with all steering prompts removed. The learner therefore predicts each next token without steering.

Learning from the steered student. At an intervention turn, we retrieve the prompt G that produced the accepted response and use the frozen student π_{θ_k} to score it under that prompt. The target receives (c, G), while the learner receives only c. Minimizing forward KL transfers this steered behavior to the unassisted learner:

$$\ell_{\mathrm{FKL}}(c, G; \theta) = D_{\mathrm{KL}}(\pi_{\theta_k}(\cdot \mid c, G) \,\|\, \pi_\theta(\cdot \mid c)) \,. \tag{1}$$

The target is recomputed with fake quantization and gradients stopped. Supervision covers the entire retained response, including reasoning when present and the resulting action or answer.

Retaining teacher supervision. At other turns, we retain teacher supervision through the reverse-KL variant of on-policy distillation (Agarwal et al., 2024). The frozen teacher scores the same student-generated history and response prefix as the learner, including contexts reached after accepted interventions:

$$\ell_{\mathrm{RKL}}(c; \theta) = D_{\mathrm{KL}}(\pi_\theta(\cdot \mid c) \,\|\, \pi_T(\cdot \mid c)) \,. \tag{2}$$

Training objective. For trajectory i, turn t, and token j, the context is $c_{i,t,j} = (h_{i,t}, y_{i,t,<j})$. Let $\mathcal{I}_i$ be the retained intervention turns, with stored prompts $G_{i,t}$, and $\mathcal{A}_k$ the set of all retained assistant-token positions (i, t, j) in cycle k. We select the loss by turn and average over these tokens:

$$\mathcal{L}^{(k)}(\theta) = \frac{1}{|\mathcal{A}_k|} \sum_{(i,t,j)\in\mathcal{A}_k} \begin{cases} \ell_{\mathrm{FKL}}(c_{i,t,j}, G_{i,t}; \theta), & t \in \mathcal{I}_i, \\ \ell_{\mathrm{RKL}}(c_{i,t,j}; \theta), & t \notin \mathcal{I}_i. \end{cases} \tag{3}$$

This objective learns from locally successful interventions while retaining teacher supervision throughout the rest of each trajectory.

4 Experiments

4.1 Experimental Setup

Benchmarks and Metrics. We evaluate on five agentic benchmarks: BFCL V4 (**?**), τ^2-Bench (Barres et al., 2025), WebShop (Yao et al., 2022), ALFWorld (Shridhar et al., 2021), and Search QA. For BFCL V4, we report the official Overall score aggregated across evaluation categories. For τ^2-Bench, we report Pass@1, the average task success rate over four independent runs. For WebShop, we report Success Rate (*SR*), which measures the fraction of tasks achieving full success. For ALFWorld, we report *SR*, the fraction of successfully completed tasks. For Search QA, we follow the evaluation protocol of Search-R1 (Jin et al., 2025) and consider seven datasets spanning single-hop and multi-hop question answering: Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). We report Exact Match (EM) on each dataset and their unweighted average. Additional metrics are provided in Section D.

Models and Training Data. We evaluate SteerDistill on Qwen3-8B (Yang et al., 2025), Qwen3.5-4B (Qwen Team, 2026), and Gemma 4 12B-it (Gemma Team, 2026). We first post-train each model in BF16 with GRPO (Shao et al., 2024b). The resulting checkpoint is the BF16 reference, the frozen teacher, and the starting point of every quantized method for that model and task. The BF16 GRPO stage and the low-bit training stage use the same task-specific data or environments. For BFCL V4 and τ^2-Bench, we train on executable tool-use tasks from AgentWorldModel (Wang et al., 2026c). For WebShop and ALFWorld, we train in their respective training environments. For Search QA, following Search-R1 (Jin et al., 2025), we use the training sets of NQ and HotpotQA for post-training and evaluate on all seven datasets, treating NQ and HotpotQA as in-domain and the remaining five datasets as out-of-domain.

Baselines. We compare with five representative baselines for quantization and capability recovery: (i) **GPTQ** (Frantar et al., 2023) for post-training quantization (PTQ); (ii) **EfficientQAT** (Chen et al., 2025) for quantization-aware training (QAT); (iii) **BitDistiller** (Du et al., 2024) for offline quantization-aware distillation (QAD); (iv) **On-Policy QAD**, implemented following NVIDIA NeMo-RL's QARL workflow (NVIDIA, 2025); and (v) **QeRL** (Huang et al., 2026) for verifiable-reward reinforcement learning. [3] We train every training-based method to convergence; hyperparameters are listed in Section C.

Quantization and Evaluation Protocol. We evaluate all three models with INT4 weights and BF16 activations (W4A16). To examine performance across quantization settings, we additionally evaluate Qwen3-8B with INT4 weights and INT8 activations (W4A8) and with INT2 weights and BF16 activations (W2A16). At evaluation, SteerDistill runs as the quantized student alone, without the teacher or steering prompts.

4.2 Main Results

Comparison across Methods. Table 1 shows that SteerDistill recovers nearly all of the performance lost to W4A16 quantization. On Qwen3-8B, it reaches an average score of 53.18 compared with 53.54 for BF16, and it outperforms the strongest quantized baseline, QeRL, on each of the five benchmarks. The per-benchmark gap to BF16 stays small across these heterogeneous tasks, so the average gain is not driven by a single benchmark.

Comparison across Precisions. Under W4A8 and W2A16, SteerDistill exceeds the strongest baseline by 1.93 and 3.39 points, respectively, and remains the best method on average in both settings; the margin is largest at W2A16. At W2A16, BitDistiller and On-Policy QAD also exceed QeRL on average, although the ordering varies by benchmark. The only benchmark–precision pair in which SteerDistill is not the best quantized method is Search QA at W2A16, where BitDistiller scores 34.22 versus 34.05. Search QA relies mainly on retrieval and textual reasoning, which may leave fewer tool-use decisions for trajectory revision to correct; our results do not isolate this explanation.

Comparison across Models. The gains are not specific to Qwen3-8B. Under W4A16, SteerDistill outperforms the strongest quantized baseline by 1.71 points on Qwen3.5-4B and 1.67 points on Gemma 4 12B-it, and its average score is within one point of the corresponding BF16 reference for all three models (0.36–0.82).

4.3 Efficiency

Training Efficiency. Figure 3a tracks τ^2-Bench Pass@1 against cumulative GPU hours. On-Policy QAD plateaus after about 40 GPU hours, and with a comparable budget of 32 GPU hours SteerDistill scores 2.91 points higher. QeRL keeps improving for longer but needs about 96 GPU hours to plateau; SteerDistill exceeds that level within 32 GPU hours (67% fewer) and finishes 1.97 points higher. All three methods use the same per-step rollout and update settings (Section C); we

[3] NeMo-RL and QeRL are primarily validated with NVFP4 weight format on Hopper and Blackwell GPUs. For a controlled comparison, we adapt their corresponding On-Policy QAD and QeRL workflows to match our GPTQ-style integer quantization setup on A100 GPUs.

Table 1: Agentic benchmark results across models and quantization settings. Scores are percentages; higher is better. Pass@1 is the average task success rate over four independent runs, and Avg. is the unweighted mean of the five displayed benchmark scores.

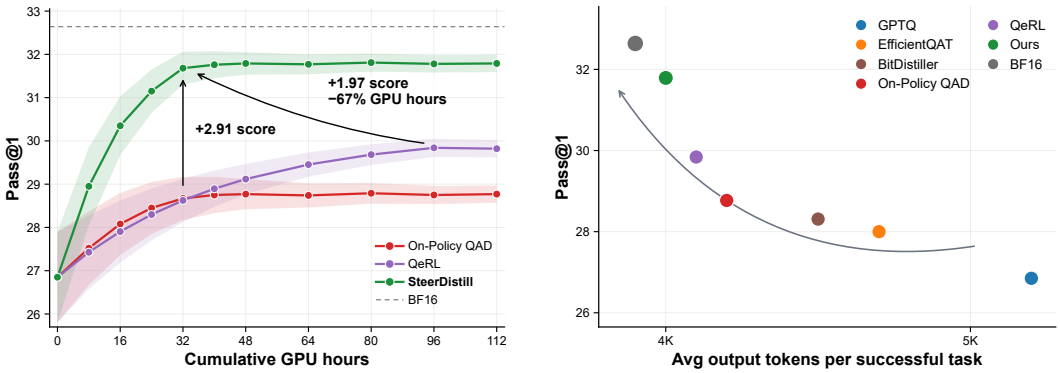
Model	Precision	Method	BFCL V4 Overall	τ^2-Bench Pass@1	WebShop SR	ALFWorld SR	Search QA EM	Avg.
Qwen3 8B	*BF16*	–	*46.89*	*32.64*	*68.58*	*79.92*	*39.65*	*53.54*
	W4A16	GPTQ	40.73	26.85	59.24	71.32	33.64	46.36
		EfficientQAT	41.77	28.00	61.34	73.18	35.08	47.87
		BitDistiller	42.46	28.31	62.07	73.91	35.69	48.49
		On-Policy QAD	44.43	28.77	64.72	76.53	37.47	50.38
		QeRL	44.91	29.84	65.29	77.04	37.94	51.00
		Ours	**46.84**	**31.79**	**68.41**	**79.75**	**39.12**	**53.18**
	W4A8	GPTQ	35.90	22.90	54.70	66.80	30.20	42.10
		EfficientQAT	40.20	27.10	60.00	71.80	34.00	46.62
		BitDistiller	41.20	27.80	60.80	72.90	34.80	47.50
		On-Policy QAD	43.15	27.85	63.35	75.10	36.25	49.14
		QeRL	42.85	29.05	64.10	75.55	36.90	49.69
		Ours	**45.20**	**30.10**	**66.70**	**78.25**	**37.85**	**51.62**
	W2A16	GPTQ	13.84	6.72	30.21	24.63	18.91	18.86
		EfficientQAT	30.92	18.44	49.13	53.87	31.46	36.76
		BitDistiller	33.71	19.86	51.84	57.36	**34.22**	39.40
		On-Policy QAD	32.88	20.57	53.08	58.41	32.96	39.58
		QeRL	31.64	20.91	50.74	55.62	31.58	38.10
		Ours	**38.12**	**23.68**	**55.71**	**63.27**	34.05	**42.97**
Qwen3.5 4B	*BF16*	–	*50.40*	*62.80*	*69.80*	*78.60*	*46.80*	*61.68*
	W4A16	GPTQ	43.91	56.18	60.65	69.78	40.58	54.22
		EfficientQAT	45.36	57.68	62.76	71.84	42.08	55.94
		BitDistiller	46.17	58.33	63.61	72.54	42.77	56.68
		On-Policy QAD	48.02	59.48	66.03	74.92	44.49	58.59
		QeRL	48.53	60.31	66.73	75.63	45.01	59.24
		Ours	**49.82**	**61.91**	**69.08**	**77.82**	**46.11**	**60.95**
Gemma 4 12B-it	*BF16*	–	*64.18*	*61.46*	*72.41*	*84.63*	*42.96*	*65.13*
	W4A16	GPTQ	57.82	54.36	63.48	75.72	36.91	57.66
		EfficientQAT	59.46	56.12	65.72	77.68	38.24	59.44
		BitDistiller	60.08	56.74	66.43	78.31	38.79	60.07
		On-Policy QAD	61.87	58.36	68.61	80.76	40.52	62.02
		QeRL	62.41	59.02	69.33	81.35	41.07	62.64
		Ours	**63.54**	**60.72**	**71.38**	**83.62**	**42.31**	**64.31**

attribute the faster recovery to the more targeted signal at corrected intervention turns, compared with a trajectory-level reward or uniform token supervision.

Inference Efficiency. Because failed tasks may end early or run to the turn limit, we compute inference statistics over each method's own successful tasks; the compared subsets therefore differ across methods. As shown in Figure 3b, SteerDistill achieves the best performance–token trade-off among quantized methods: its successful tasks use 4.0K output tokens on average, close to BF16's 3.9K, versus 4.1K–5.2K for the baselines, and they also take fewer interaction steps and tool calls (Table 6). One possible explanation is fewer erroneous intermediate turns that require later repair, although these statistics do not isolate this cause. Across methods, lower accuracy generally comes with longer successful trajectories. Section E reports the same comparison under W4A8 and W2A16.

4.4 Ablation Studies and Analysis

Table 2 ablates five design choices on BFCL V4; scores below refer to the Overall score.



(a) Training efficiency.

(b) Inference efficiency under W4A16.

Figure 3: Efficiency analysis of Qwen3-8B. (a) Training efficiency on τ^2-Bench. GPU hours are computed as wall-clock training time multiplied by the number of GPUs; all runs use 8 NVIDIA A100 80GB GPUs. Curves show the mean over four evaluation seeds, and shaded regions indicate ± 1 standard deviation. (b) Performance–token trade-off on τ^2-Bench under W4A16: average output tokens per successful task versus Pass@1. The arrow points toward higher performance and fewer tokens. Both panels use a separate evaluation run, so scores may differ slightly from Table 1.

Table 2: Ablation study on BFCL V4 using Qwen3-8B under W4A16. BF16 and On-Policy QAD provide reference scores. Each Q1–Q5 group lists alternatives to its stated default; all defaults together form the full-method row. Final success only adds terminal-success filtering after the local transition check. Section F.3 shows a case study for Q3.

Configuration	Overall	Non-Live	Live	Multi-Turn	Web Search	Memory	Irrel.
BF16	46.89	87.14	81.76	48.53	18.57	19.32	78.63
On-Policy QAD	44.43	85.72	80.19	45.14	16.86	17.16	74.88
Full SteerDistill (Ours)	**46.84**	**88.29**	**81.71**	**48.47**	**18.44**	**18.79**	**78.56**
Q1: Intervention localization (default: Behavioral divergence)							
Random eligible turn	43.52	85.20	79.72	44.00	16.10	16.30	73.48
First action mismatch	45.01	86.20	80.55	45.95	17.20	17.45	76.20
Q2: Intervention strategy (default: Teacher steering)							
Unsteered student retry	44.46	85.81	80.08	45.26	16.93	17.05	74.97
Teacher takeover	45.06	86.31	80.69	46.02	17.31	17.55	75.81
Q3: Steering format and source (default: Teacher / Structured)							
Teacher / Free-form	44.78	87.10	80.85	45.35	17.10	17.05	75.50
Student / Structured	41.74	83.72	79.02	41.48	14.66	14.91	71.08
GPT-5.6-Sol / Structured	45.27	87.05	80.90	46.15	17.55	17.65	75.90
Q4: Revision retention (default: Local transition verified)							
No local verification	42.18	84.10	78.95	42.30	14.95	15.20	71.55
Final success only	46.30	87.91	81.09	47.72	18.16	18.27	78.02
Q5: Distillation routing (default: FKL at verified intervention turns; RKL elsewhere)							
Reverse KL throughout	45.29	86.45	80.93	46.33	17.48	17.78	76.02
Forward KL throughout	44.58	85.98	80.47	45.22	17.28	16.89	75.32
Reversed routing (RKL / FKL)	44.28	85.64	80.28	44.75	16.96	16.89	74.93
FKL from intervention onward	44.35	85.65	80.38	45.15	16.83	16.70	74.96

Q1 (Intervention localization): Intervene at the behavioral divergence. Among canonicalized action mismatches, the behavioral divergence is the first one that also changes the task-relevant environment transition. Intervening there gains 1.83 points over the first action mismatch and 3.32 over a random eligible turn. An earlier mismatch can be behaviorally equivalent to the reference, so revising it spends the attempt budget on a decision that needs no correction.

Q2 (Intervention strategy): Steer the student instead of taking over. All three variants restore the same history and environment state and differ only in how the response at the divergent turn is produced; the student generates all later turns. Unsteered retry resamples the student without a prompt, teacher takeover inserts the teacher's full reasoning-and-action response, and steering gives the student a prompt while it generates both reasoning and action. Steering gains 2.38 points over retry and 1.78 over takeover. The latter gap is consistent with steered corrections staying closer to the student's own response distribution, although the table does not isolate this mechanism.

Q3 (Steering format and source): Use structured teacher prompts. With the intervention turn, reference, and verification fixed, a structured prompt states the discrepancy, the condition under which it applies, and the corrective rule separately. It gains 2.06 points over a free-form prompt from the same teacher, possibly because the required change and its scope are easier to carry across interventions. With the structured format, the teacher also outperforms the student (+5.10) and the external GPT-5.6-Sol (OpenAI, 2026) (+1.57) as prompt sources, all given the same reference. Section F.3 compares the prompts from each variant on one case.

Q4 (Revision retention): Retain locally verified revisions. The local verifier checks whether an action produces the task-relevant state change that the reference requires at that point. Retaining locally verified revisions scores 46.84, versus 42.18 without this check and 46.30 when the revised trajectory must also reach terminal success. Unverified revisions thus hurt substantially, while also requiring final success discards verified corrections from trajectories that fail later for other reasons.

Q5 (Distillation routing): Apply forward KL at intervention turns. All variants use the same retained trajectories, target models, and conditioning, and differ only in which KL direction each turn receives. Our routing gains 1.55 points over reverse KL throughout and 2.26 over forward KL throughout. Swapping the two directions (RKL at intervention turns, FKL elsewhere) loses 2.56 points, and extending FKL from the intervention through the rest of the trajectory loses 2.49. Because FKL is weighted by the steered target's probability mass, it can transfer corrections to which the unassisted student assigns low probability; the scope comparison suggests that this is most useful at the intervention turn itself.

5 Conclusion

SteerDistill recovers the agentic capabilities of quantized LLMs by correcting the student's own rollouts at their behavioral divergences through teacher steering and distilling the verified corrections into the unassisted student. Across five benchmarks and three models under W4A16, it improves over quantized baselines and stays within one point of BF16, while using less training compute than quantization-aware reinforcement learning and producing successful trajectories close to BF16 in length. These results support correcting quantization errors at the turns where they occur and teaching the student to make those corrections without steering. We discuss limitations in Section B.

AI use statement

Generative AI tools were used to assist with interpreting experimental results and to provide feedback on methodology and experimental analysis. They were also used to organize, draft, and edit portions of the manuscript and to help identify and check relevant literature. The authors take responsibility for the final content of this work, including all AI-assisted text, claims, and artifacts.

Reproducibility statement

Section 3 specifies the trajectory revision procedure and distillation objective. Section 4.1 describes the evaluation setup. Section A provides implementation details and a notation reference, while Section C reports the training configurations and hyperparameters. We provide anonymous source code in supplementary materials.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs. *arXiv preprint arXiv:2404.00456*, 2024. URL https://arxiv.org/abs/2404.00456v2.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. tau2-Bench: Evaluating conversational agents in a dual-control environment. 2025. URL https://arxiv.org/abs/2506.07982. Oral presentation at ICML 2026, if retaining the conference-status note.

Chishui Chen, Yaoyou Fan, Te Sun, Yi Yang, Chenghao Sun, Delin Mao, Hongbo Qiao, Zuowei Zhang, Junxi Wang, Chenxing Sun, et al. Look ahead before you distill: Future trajectory validation of teacher guidance for agentic on-policy distillation. *arXiv preprint arXiv:2608.01953*, 2026. URL https://arxiv.org/abs/2608.01953.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10081–10100. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.498. URL https://aclanthology.org/2025.acl-long.498/.

DaYou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. BitDistiller: Unleashing the potential of sub-4-bit LLMs via self-distillation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 102–116. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.7. URL https://aclanthology.org/2024.acl-long.7/.

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/2210.17323v2.

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

Chenhui Gou, Haoqin Tu, Yunhao Fang, Jianfei Cai, and Hamid Rezatofighi. Sample-Efficient Learning from Agent Experience. *arXiv preprint arXiv:2607.21051*, 2026. URL https://arxiv.org/abs/2607.21051v1.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, Yao Lu, Xiaojuan Qi, Song Han, and Yukang Chen. QeRL: Beyond efficiency – quantization-enhanced reinforcement learning for LLMs. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.11696.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement Learning via Self-Distillation. *arXiv preprint arXiv:2601.20802*, 2026. URL https://arxiv.org/abs/2601.20802v2.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Proceedings of the Second Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2503.09516. OpenReview: Rwhi91ideu.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Aristotelis Lazaridis, Dylan Bates, Aman Sharma, Brian King, Vincent Lu, and Jack FitzGerald. EDGE-OPD: Internalizing Privileged Context with Evidence Guided On-Policy Distillation. *arXiv preprint arXiv:2605.23493*, 2026. URL https://arxiv.org/abs/2605.23493v1.

Janghwan Lee, Sihwa Lee, Jinseok Kim, Yongjik Kim, Jieun Lim, Jinwook Oh, and Jungwook Choi. ReQAT: Achieving full-precision reasoning accuracy with 4-bit floating-point quantization-aware training. *arXiv preprint arXiv:2606.15682*, 2026. URL https://arxiv.org/abs/2606.15682.

Gengsheng Li, Mao Zheng, Mingyang Song, Ruiqi Liu, Tianyu Yang, Jie Sun, Qiyong Zhong, Haiyun Guo, Junfeng Fang, Dan Zhang, et al. On-policy distillation with curriculum turn-level guidance for multi-turn agents. *arXiv preprint arXiv:2606.15912*, 2026. URL https://arxiv.org/abs/2606.15912.

Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. Personal LLM agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024. URL https://arxiv.org/abs/2401.05459v2.

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, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Junzhuo Liu, Weiwei Li, Jun Ling, and Peng Wang. When Privileged Guidance Misaligns: State-Matched Routing and Contextualized Self-Distillation for Multi-Turn Agents. *arXiv preprint arXiv:2608.05219*, 2026. URL https://arxiv.org/abs/2608.05219v1.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 467–484, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.26. URL https://aclanthology.org/2024.findings-acl.26/.

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/2405.16406v4.

Haiquan Lu, Zigeng Chen, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Mix-Quant: Quantized Prefilling, Precise Decoding for Agentic LLMs. *arXiv preprint arXiv:2605.20315*, 2026a. URL https://arxiv.org/abs/2605.20315v1.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026b. URL https://arxiv.org/abs/2605.15155.

Yuanjie Lyu, Chengyu Wang, Jun Huang, and Tong Xu. Student-Centered Distillation Narrows the Agentic Gap Between Small and Large LLMs. *arXiv preprint arXiv:2509.14257*, 2025. URL https://arxiv.org/abs/2509.14257v3. Accepted to ICML 2026; camera-ready arXiv version.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

NVIDIA. NeMo-RL: A scalable and efficient post-training library. Software repository, 2025. URL https://github.com/NVIDIA-NeMo/RL. Accessed 2026-09-26.

NVIDIA. Quantization-aware RL (QARL). NeMo-RL documentation, 2026. URL https://docs.nvidia.com/nemo/rl/latest/guides/quantization-aware-rl.html. Accessed 2026-09-26.

OpenAI. Gpt-5.6 system card. https://deploymentsafety.openai.com/gpt-5-6, July 2026.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent Q: Advanced reasoning and learning for autonomous AI agents. *arXiv preprint arXiv:2408.07199*, 2024. URL https://arxiv.org/abs/2408.07199v1.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2411.02337v3.

Qwen Team. Qwen3.5: Towards native multimodal agents. Official blog post, February 2026. URL https://qwen.ai/blog?id=qwen3.5. Accessed 2026-09-26; not a peer-reviewed paper.

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*, 2024a. URL https://arxiv.org/abs/2308.13137v3.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024b. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Qiming Shi, Yibo Dou, Jiawen Zhu, Yulong Tao, Linbo Jin, Zhaolu Kang, Yunfan Zhou, and Di Weng. SKILL-KD: Contrastive Skill Distillation for LLM Agents. *arXiv preprint arXiv:2607.28048*, 2026. URL https://arxiv.org/abs/2607.28048v2.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-SD: Skill-conditioned self-distillation for multi-turn LLM agents. *arXiv preprint arXiv:2604.10674*, 2026a. URL https://arxiv.org/abs/2604.10674.

Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. TCOD: Exploring Temporal Curriculum in On-Policy Distillation for Multi-turn Autonomous Agents. *arXiv preprint arXiv:2604.24005*, 2026b. URL https://arxiv.org/abs/2604.24005v3.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024. doi: 10.1007/s11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. *arXiv preprint arXiv:2602.10090*, 2026c. URL https://arxiv.org/abs/2602.10090.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Ahmed Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Conference on Language Modeling*, 2024. URL https://www.microsoft.com/en-us/research/publication/autogen-enabling-next-gen-llm-applications-via-multi-agent-conversation-framework/.

Xu Xia, Jinghua Piao, Min Yang, Xiaochong Lan, Jiaju Chen, and Yong Li. From Scoring to Acting: Outcome-Verified Comparative Self-Distillation for LLM Agents. *arXiv preprint arXiv:2607.27937*, 2026. URL https://arxiv.org/abs/2607.27937v1.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Can Xie, Yuyi Zhou, Wen Yang, Ziyi Zhang, Siyao Song, Yingzhuo Deng, Shuo Ren, and Jiajun Zhang. EDGE: Experience-Distillation for Guided Exploration in Agentic Reinforcement Learning. *arXiv preprint arXiv:2608.21946*, 2026. URL https://arxiv.org/abs/2608.21946v2. Accepted to EMNLP 2026 (Main Conference).

Meng Xin, Sweta Priyadarshi, Jingyu Xin, Bilal Kartal, Aditya Vavre, Asma Kuriparambil Thekkumpate, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Ido Shahaf, Akhiad Bercovich, et al. Quantization-aware distillation for NVFP4 inference accuracy recovery. *arXiv preprint arXiv:2601.20088*, 2026. URL https://arxiv.org/abs/2601.20088.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang

Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629v3.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-Policy Context Distillation for Language Models. *arXiv preprint arXiv:2602.12275*, 2026. URL https://arxiv.org/abs/2602.12275v2.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models. *arXiv preprint arXiv:2601.18734*, 2026. URL https://arxiv.org/abs/2601.18734v3.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H. Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583, 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Yuhang Zhou, Lizhu Zhang, Yifan Wu, Mingyi Wang, Bo Peng, Jiayi Liu, Xiangjun Fan, and Zhuokai Zhao. SAGE-OPD: Selective agent-guided intervention for multi-turn on-policy distillation. *arXiv preprint arXiv:2606.19659*, 2026. URL https://arxiv.org/abs/2606.19659.

A Implementation Details of SteerDistill

Algorithm 1 revises one student rollout. We describe the local checks, steering state, and training implementation below; Table 3 summarizes their shared notation.

Algorithm 1 Teacher-steered trajectory revision for one rollout.

Require: student rollout τ, frozen student π_{θ_k}, teacher π_T, successful reference τ^T if available, attempt budget $R \geq 1$
Ensure: retained trajectory τ, intervention turns $\mathcal{I}$, and prompt map $\mathcal{G}$
$\mathcal{I} \leftarrow \varnothing$; $\mathcal{G} \leftarrow \varnothing$; $\mathcal{M} \leftarrow \varnothing$
if τ succeeds or no successful reference is available **then**
  return $(\tau, \mathcal{I}, \mathcal{G})$
end if
while τ fails **do**
  Align τ with fixed τ^T to select turn $t^\star$ and reference transition ρ^T
  $F \leftarrow \varnothing$
  for $r = 1, \ldots, R$ **do**
    Teacher π_T forms G from the aligned discrepancy, memory $\mathcal{M}$, and feedback F
    Restore history $h_{t^\star}$ and the environment state before turn $t^\star$
    Sample $y \sim \pi_{\theta_k}(\cdot \mid h_{t^\star}, G)$; execute its action to obtain transition ρ
    $(v, F) \leftarrow \mathrm{LocalVerify}(\rho, \rho^T)$
    if v is true **then break**
  end for
  if v is false **then return** $(\tau, \mathcal{I}, \mathcal{G})$
  Replace turn $t^\star$ and its suffix with y and an unsteered continuation from π_{θ_k}
  Discard intervention records at or after $t^\star$ from $(\mathcal{I}, \mathcal{G})$
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{t^\star\}$; $\mathcal{G}[t^\star] \leftarrow G$; $\mathcal{M} \leftarrow G$
end while
return $(\tau, \mathcal{I}, \mathcal{G})$

Alignment and local checks. After each accepted replacement, we realign against the same successful teacher reference, canonicalizing actions and observations to remove task-irrelevant surface differences. Alignment selects student turn $t^\star$ and reference transition ρ^T; their turn indices need not coincide. An environment-specific verifier compares the candidate transition ρ with the task-relevant conditions of ρ^T, returning an acceptance flag v and feedback F. Comparisons of canonicalized state fields supplement the verifier where its coverage is insufficient. The verifier and state fields depend on the benchmark environment.

Steering prompts and retries. The teacher combines a local steer (discrepancy, state precondition, corrective rule, expected transition, and recovery check) with memory $\mathcal{M}$ into prompt G. Each of up to $R = 4$ attempts restores the same history and environment state. Failed attempts provide feedback for retries. After acceptance, G becomes the memory and is stored in $\mathcal{G}$; unsteered continuation starts from the accepted transition's feedback. Replacing a suffix removes its old intervention records. Revision extends the aligned prefix until success or exhausted retries. We train on either outcome.

Quantized generation and optimization. The teacher and student start from the same BF16 checkpoint. The teacher stays frozen; the student maintains trainable BF16 master weights. Generation in cycle k uses the actual low-bit runtime with snapshot θ_k. For optimization, the FKL target $\pi_{\theta_k}(\cdot \mid c, G)$ is recomputed with a frozen fake-quantized copy of this snapshot and gradients stopped. It receives the stored prompt and the same history and response prefix as the learner. The shared notation identifies the same snapshot and quantization scheme, without assuming numerical identity between the low-bit and fake-quantized runtimes. The learner $\pi_\theta(\cdot \mid c)$ uses fake quantization and propagates gradients through the straight-through estimator to its master weights. Updated weights are quantized for the next cycle.

Loss tokens. The set $\mathcal{A}_k$ includes all retained assistant-response tokens: reasoning when present, tool names and arguments, text responses, and termination tokens. System and user messages and environment observations provide context but receive no loss. Steering prompts condition only the FKL target and also receive no loss. Equation (3) averages uniformly over $\mathcal{A}_k$, so the relative contribution of FKL and RKL follows their token counts. Discarded responses and suffixes are excluded; RKL after an accepted intervention supervises the new continuation.

Table 3: Key notation for SteerDistill. The revision algorithm handles one trajectory; training adds index i.

Symbol	Meaning
Policies and conditioning	
π_T	Frozen full-precision teacher.
θ, π_θ	Student master weights and trainable quantized policy.
k, θ_k, π_{θ_k}	Training cycle, frozen student weights, and corresponding policy.
i, t, j	Trajectory, turn, and response-token indices.
τ, τ^T	Student trajectory and its fixed successful teacher reference.
h_t, y_t	History before turn t and the response generated at that turn.
$c_{i,t,j}$	Token context $(h_{i,t}, y_{i,t,<j})$, with all steering prompts removed.
Revision state and retained supervision	
$t^\star$	Intervention turn selected by alignment at the current revision step.
G, $G_{i,t}$	Current steering prompt and the prompt retained for turn t in trajectory i.
$\mathcal{M}$	Memory holding the latest locally accepted steering prompt.
$\mathcal{G}$	Stored prompt map: $\mathcal{G}[t] = G_{i,t}$ for trajectory i.
$\mathcal{I}_i$	Retained intervention turns in trajectory i; these receive FKL supervision.
Training objective	
$\mathcal{A}_k$	Retained assistant-token positions (i, t, j) in cycle k.
ℓ_{FKL}, ℓ_{RKL}	Per-token losses at intervention and other turns (Equations (1) to (2)).
$\mathcal{L}^{(k)}(\theta)$	Mean routed loss over $\mathcal{A}_k$ (Equation (3)).

B LIMITATIONS

Trajectory revision requires a successful teacher reference, restoration of the environment state, and reliable local verification. Its applicability to environments without these capabilities remains untested. Moreover, local acceptance validates an action's effect but does not establish the correctness of every reasoning step in the generated response.

C TRAINING DETAILS

Table 4 summarizes the shared training budget; method-specific settings are described below.

Table 4: Training configurations of all training-based methods. Method-specific settings not listed here follow the corresponding official implementations.

Method	Epochs	Learning Rate	Batch Size	Samples / Prompt	Updates / Rollout
EfficientQAT	$1+1$	8×10^{-6}	16	–	–
BitDistiller	2	8×10^{-6}	16	–	–
On-Policy QAD	2	8×10^{-6}	16	8	4
QeRL	2	8×10^{-6}	16	8	4
Ours	2	8×10^{-6}	16	8	4

All training-based methods use two epochs, a learning rate of 8×10^{-6}, and a batch size of 16. EfficientQAT splits the epochs evenly between Block-AP and E2E-QP. BitDistiller retains its official AdamW optimizer, constant learning rate, zero weight decay, and quantization-aware distillation objective. On-Policy QAD follows NVIDIA NeMo-RL QARL, retaining its rollout temperature of 1.0, top 64 teacher logits, and on-policy distillation settings. QeRL uses GRPO, temperature 1.0, clipping bounds $(0.2, 0.28)$, AdamW-8bit, and its official adaptive quantization-noise configuration. SteerDistill shares their rollout and update budget (Table 4) and adds teacher-steered revision in verified failure contexts. All quantized configurations use group-wise quantization with group size 64; all experiments run on 8 NVIDIA A100 GPUs.

D Additional Evaluation Metrics

Table 5 reports metrics omitted from the main table: Pass@4 on τ^2-Bench, the fraction of tasks solved in at least one of four runs; Task Score on WebShop, which measures how well the selected product satisfies the user's requirements; and per-dataset EM on the seven Search QA datasets.

Table 5: Additional evaluation metrics on Qwen3-8B under W4A16. We report Pass@4 for τ^2-Bench, Task Score for WebShop, and per-dataset Exact Match (EM) for Search QA. † and ⋆ denote in-domain and out-of-domain Search QA datasets.

Method	τ^2-Bench Pass@4	WebShop Score	NQ†	TriviaQA⋆	PopQA⋆	HotpotQA†	2Wiki⋆	MuSiQue⋆	Bamboogle⋆
BF16	*54.20*	*78.69*	*43.36*	*62.42*	*41.96*	*36.58*	*35.12*	*15.92*	*42.18*
GPTQ	45.50	70.86	37.62	56.04	36.05	30.44	28.39	12.23	34.71
EfficientQAT	47.10	72.73	39.08	57.69	37.48	32.02	30.17	12.88	36.25
BitDistiller	47.55	73.41	39.71	58.31	38.11	32.71	30.86	13.21	36.92
On-Policy QAD	48.20	75.61	41.18	60.22	39.69	34.47	32.86	14.46	39.44
QeRL	50.10	76.08	41.64	60.73	40.13	35.03	33.36	14.79	39.92
Ours	**53.05**	**78.57**	**42.84**	**61.88**	**41.41**	**36.01**	**34.55**	**15.53**	**41.61**

E Detailed Inference Efficiency

Figure 3b in the main paper reports the performance–token trade-off under W4A16, while Figure 4 extends the comparison to W4A8 and W2A16. Across all three precision settings, SteerDistill maintains higher performance with fewer output tokens per successful task than competing quantized methods. As in the main paper, statistics are computed over each method's own successful tasks, so they describe the length of successful trajectories rather than a matched per-task comparison. Table 6 further reports detailed trajectory-level statistics under W4A16.

Table 6: Detailed trajectory and inference statistics on τ^2-Bench using Qwen3-8B under W4A16. We report average input, cached-input, and output tokens per successful task, together with interaction steps and tool calls. Lower is better; K denotes thousands of tokens.

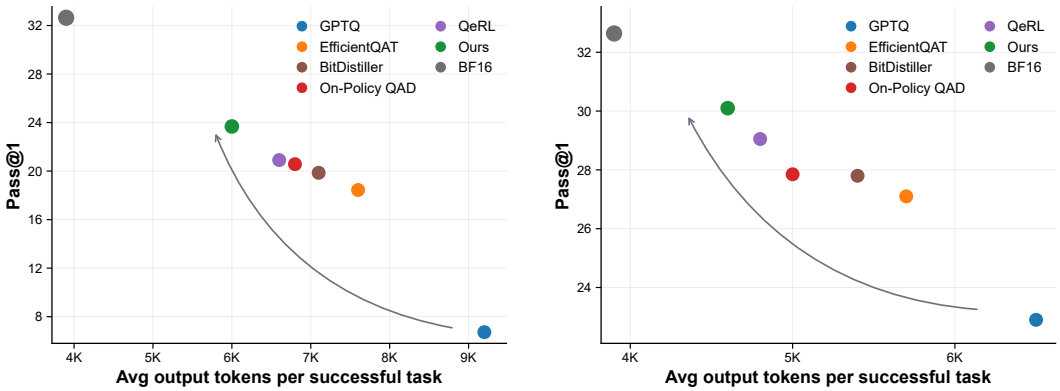
Method	Input Tokens	Cached Input	Output Tokens	Interaction Steps	Tool Calls
BF16	*185K*	*148K*	*3.9K*	*11.4*	*4.2*
GPTQ	270K	223K	5.2K	13.8	5.25
EfficientQAT	243K	199K	4.7K	13.1	4.92
BitDistiller	235K	192K	4.5K	12.9	4.78
On-Policy QAD	214K	173K	4.2K	12.3	4.55
QeRL	206K	166K	4.1K	12.1	4.43
Ours	**192K**	**153K**	**4.0K**	**11.6**	**4.24**

F Additional Case Studies

F.1 Inference Cases

Preserving Executable Actions. The three inference cases share a failure pattern: the quantized models keep the task intent but fail to turn it into the exact executable action. In the Java case, the error is at the interface boundary; in the factorial case, the model recognizes the operation but does not complete the call sequence; in the mixed-domain case, it keeps the subtask but corrupts or omits the required tool call. In all three, SteerDistill matches the BF16 action (Figures 5 to 7). Quantization thus mainly degrades the mapping from intent to the exact tool, arguments, and call structure that the current environment requires, and this mapping is what recovery must preserve.

Reasoning-Prefix Intervention. Swapping reasoning prefixes shows that these action errors do not come from the weights alone. Figure 8 fixes the interaction history, environment state, and the



(a) Inference efficiency under W2A16.

(b) Inference efficiency under W4A8.

Figure 4: Additional performance–token efficiency results of Qwen3-8B on τ^2-Bench under W2A16 and W4A8. The x-axis reports the average output tokens per successful task, while the y-axis reports Pass@4 performance.

Preserving the Java tool interface

Context. Append the node ID from LogEvent logEvent to StringBuilder logBuilder in Elasticsearch. The available function is NodeIdConverter.format(event, toAppendTo).

Method	Decision / reasoning summary	Action / response
Ref (BF16)	Identifies the node-ID formatter and maps logEvent to event and logBuilder to toAppendTo. ***Correct interface***	NodeIdConverter.format(event="logEvent", toAppendTo="logBuilder")
PTQ	Replaces execution of the Java operation with a completion statement. ***No tool call***	"The Java code has been updated successfully."
QAT Offline QAD RLVR	Identifies the node-ID formatter and maps logEvent to event and logBuilder to toAppendTo. ***Tool-name mismatch***	Nodeidconverter.format(event="logEvent", toAppendTo="logBuilder")
On-Policy QAD	Identifies the node-ID formatter and maps logEvent to event and logBuilder to toAppendTo. ***Two interface mismatches***	Nodeidconverter.format(event="logEvent", toappendto="logBuilder")
Ours	Identifies the node-ID formatter and maps logEvent to event and logBuilder to toAppendTo. ***Correct interface***	NodeIdConverter.format(event="logEvent", toAppendTo="logBuilder")

Figure 5: Exact Java interface matching. Given the node-ID formatting request, SteerDistill and BF16 call `NodeIdConverter.format` with `event` and `toAppendTo`; the other quantized methods shown either modify the interface or omit the call.

weights that generate the response, and changes only the reasoning prefix. In both divergent-turn contexts, BF16 and SteerDistill prefixes yield much higher rates of locally verified correct actions, whereas prefixes from weaker quantized models lead the same weights to incorrect actions. In the door-locking case, the prefix changes an argument of the executed call; in the file-ordering case, it changes whether the model queries the environment before answering. Quantization errors can therefore enter through the generated reasoning and alter the subsequent action, not only at the final tool tokens.

Completing every requested computation

Context. The user requests 5!, 7!, and 9! in an environment that requires tool use. math.factorial(number) is available to execute each calculation.

Method	Decision / reasoning summary	Action / response
Ref (BF16)	Executes one factorial call per requested number, then reports all three results. ***All three calls executed***	math.factorial(number=5) math.factorial(number=7) math.factorial(number=9)
PTQ	Treats the calculations as simple enough to answer directly. ***Tool execution skipped***	Direct numerical answer; no computation tool call.
QAT **Offline QAD**	Starts with 5! and 7!, then emits repetitive malformed text instead of the third call. ***Incomplete execution***	math.factorial(number=5) math.factorial(number=7) [malformed third tool-call text]
On-Policy QAD	Concludes that no factorial tool is available without listing the tools. ***Available tool overlooked***	"Please provide access to a suitable tool or clarify if manual computation is acceptable."
RLVR	Calculates the three factorials directly and reports no applicable tool. ***Correct numbers, no tool calls***	5! = 120 7! = 5040 9! = 362880
Ours	Executes one factorial call per requested number, then reports all three results. ***All three calls executed***	math.factorial(number=5) math.factorial(number=7) math.factorial(number=9)

Figure 6: Completing a multi-call request. SteerDistill and BF16 execute all three factorial calls and obtain 120, 5040, and 362880. The other quantized methods shown omit execution, overlook the tool, or fail to generate a valid call for 9!.

F.2 Training Cases

The training cases show that effective steering is local and depends on the current state; it is not a generic retry signal. The three examples involve different failures: a misgrounded argument, a wrong operation after the required category is already known, and a child-object lookup issued before its parent object is known. In each, the steered response keeps what is already valid in the trajectory and changes only the decision that blocks progress. Replacing the whole response would discard valid student-generated context along with the error; revising only the divergent decision keeps the supervision matched to contexts the quantized policy actually reaches.

F.3 Ablation Cases

Figure 12 illustrates the Q3 ablation (Table 2) on two intervention states of one sales-order task. The workflow requires the order header to exist before part retrieval, and the retrieved part to ground line creation. The structured teacher prompt separates these two prerequisites, whereas the free-form prompt conflates header and line creation when merging steers. With the structured format fixed, the student-written prompt focuses on field checks, and the external LLM gives a less state-specific instruction to create or confirm the order, although all sources receive the same reference trajectory. The case illustrates how the prompts differ; the table reports the aggregate effect.

F.4 Recovery Cases

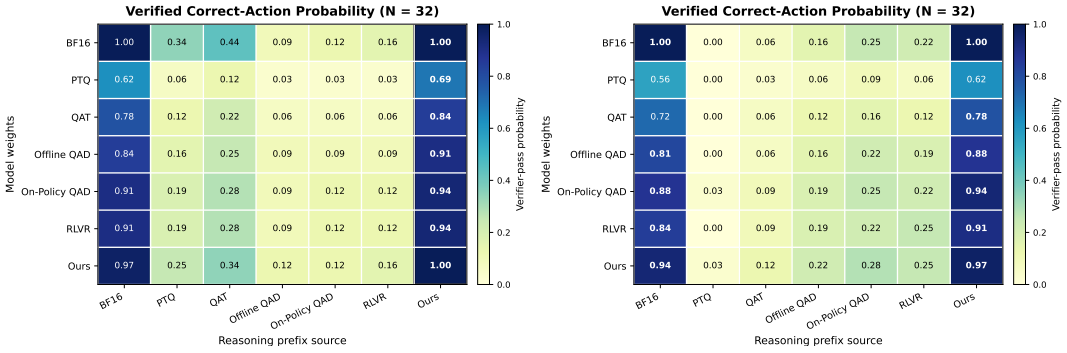
The recovery cases suggest that the learned local repair also applies to inference-time errors that were not explicitly trained. In one case, the environment rejects an infeasible refill and returns the

Keeping the correct tool across a multi-part task

Context. The task requests a vegan main course within 30 minutes, Gravity and Higgs Boson discovery details using the default method, and a gluten-free dessert within 45 minutes; the comparison below focuses on the Higgs Boson lookup.

Method	Decision / reasoning summary	Action / response
Ref (BF16)	Reuses the science-history tool for Higgs Boson and leaves the method at its default. ***Correct tool and argument***	science_history.get_discovery_details(discovery="Higgs Boson")
PTQ	Focuses on the main request and omits secondary requirements. ***Constraint omission***	Answer satisfying only the main requirement.
QAT **Offline QAD**	Attempts a Higgs Boson lookup with method_used="default". ***Tool-name mismatch***	scscience_history.get_discovery_details(discovery="Higgs Boson", method_used="default")
On-Policy QAD	Passes the Higgs Boson query to a different tool identifier. ***Wrong tool identifier***	m/dataTables(discovery="Higgs Boson")
RLVR	Keeps the discovery argument but rewrites the tool identifier. ***Tool-name mismatch***	m_c_p_tool_science_history_get_discovery_details(discovery="Higgs Boson")
Ours	Reuses the science-history tool for Higgs Boson and leaves the method at its default. ***Correct tool and argument***	science_history.get_discovery_details(discovery="Higgs Boson")

Figure 7: Tool selection in a recipe-and-science task. At the illustrated Higgs Boson lookup, SteerDistill and BF16 preserve `science_history.get_discovery_details`; the other quantized methods shown corrupt the tool identifier or omit a required subrequest.



(a) Door-locking case.

(b) File-ordering case.

Figure 8: Reasoning-prefix intervention for two representative divergent-turn contexts. Rows denote the model weights used to generate the next response, and columns denote the source of the injected reasoning prefix. Each cell reports the fraction of locally verified correct actions over 32 trials ($N = 32$) from the same interaction history and environment state.

current state; in the other, it rejects an invalid interface name. In both, SteerDistill uses the returned feedback to fix only the failed part of the action and then resumes the task, rather than restarting or abandoning it. This mirrors revision during training, where a failed transition is followed by a locally verified correction conditioned on the resulting state.

Asset search argument grounding

Shared context at the intervention turn

Task. Check out PRJ-DRN-003 to Maria Lopez (Engineering), record the requested dates, and move it to the Prototype Lab.

Observed state. get_asset_by_tag(asset_tag="PRJ-DRN-003") returned HTTP 500; the exact asset tag is still known.

1 · Ours before steering

search_assets(search="PRJ-DRN-003")

The fallback tool is appropriate, but the known tag is passed through the generic search field.

3 · Ours after steering

search_assets(asset_tag="PRJ-DRN-003")

Preserves the exact tag and corrects the argument key.

2 · Steering process

Steering analysis

Keep the fallback search tool and bind the exact tag to its asset_tag parameter.

Applicable condition

The agent needs to identify an asset by its tag but the 'mcp_server_get_asset_by_tag' tool has returned a 500 error.

Injected steering prompt

Steer 1 · prompt excerpt

When calling 'mcp_server_search_assets' to find an asset with a known tag, use the 'asset_tag' argument key instead of the 'search' argument key.

Figure 9: Asset argument grounding at assistant turn 2. After direct retrieval fails, steering changes the argument key from `search` to `asset_tag`, retaining the exact identifier `PRJ-DRN-003` and the fallback search operation.

Service creation tool selection

Shared context at the intervention turn

Task. Create a 60-minute Deep Tissue Massage service priced at $120, with a Massage Room and the requested add-ons.

Observed state. Downtown Spa is location_id=1; the Massage category was retrieved as service_category_id=1.

1 · Ours before steering

list_resource_types(name="Massage Room")

Ours attempts a resource-type lookup before creating the service required by the workflow.

3 · Ours after steering

create_service(base_price=120, default_duration_minutes=60, name="Deep Tissue Massage", service_category_id=1)

Creates the service using the retrieved category ID and requested values.

2 · Steering process

Steering analysis

The category ID is already grounded; create the service with its known name, price, duration, and category.

Applicable condition

The agent has successfully retrieved the 'service_category_id' for the 'Massage' category (ID: 1).

Injected steering prompt

Steer 1 · prompt excerpt

Immediately call 'mcp_server_create_service' with the required parameters (name: 'Deep Tissue Massage', base_price: 120, default_duration_minutes: 60, service_category_id: 1). Do not attempt to list or verify resource types before creating the service.

Figure 10: Service creation at assistant turn 3. Once the Massage category is known, steering replaces the resource lookup with `create_service`, using category 1, the requested $120 price, and the 60-minute duration.

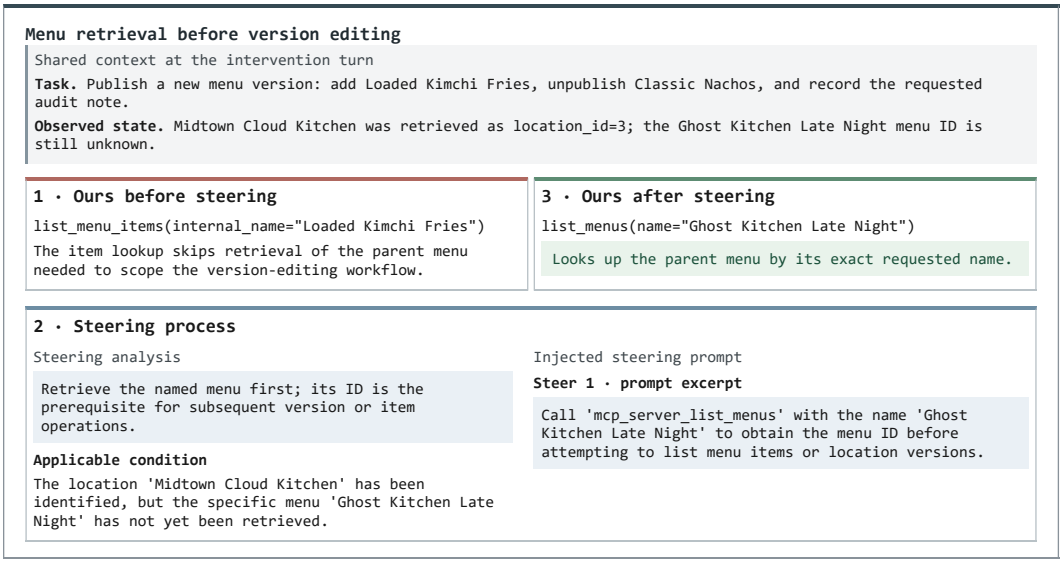


Figure 11: Parent-menu retrieval at assistant turn 2. With the location known but the menu ID missing, steering redirects the policy from premature item lookup to retrieving Ghost Kitchen Late Night, establishing the parent object required for subsequent edits.

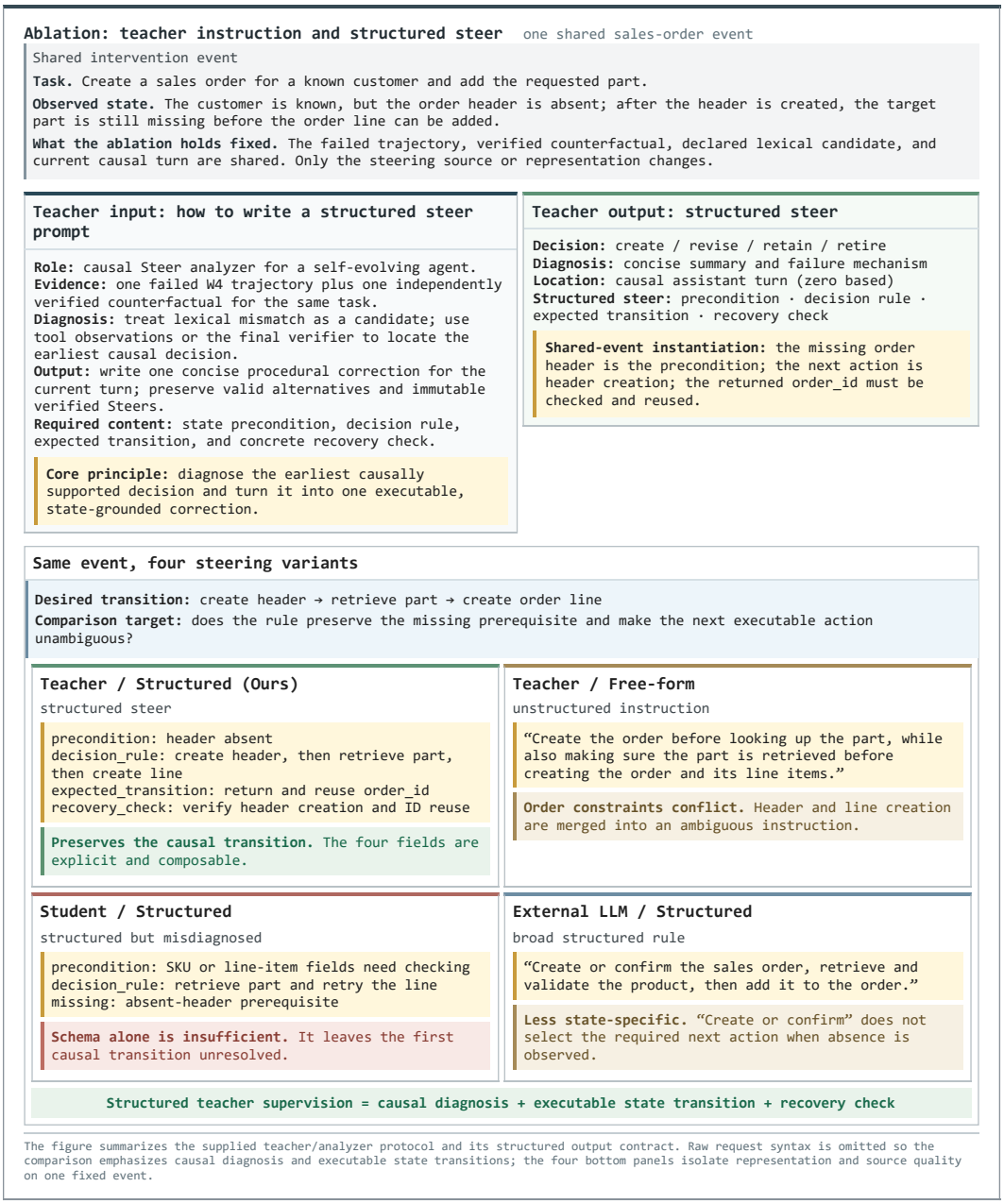


Figure 12: Combined ablations on one sales-order case. Two intervention states compare structured teacher, free-form teacher, student-structured, and external-LLM-structured steering. Structured teacher guidance preserves the header creation → part retrieval → line creation transition; the alternatives either conflate prerequisites or provide less state-specific corrections.

(a) Fuel-capacity recovery

Context. Fill the tank for a road trip; capacity is 50 gal, but the current fuel level has not been checked.

Ref BF16

```
displayCarStatus("fuel")
↳Fuel level: 15 gal
fillFuelTank(fuelAmount=35)
✓Fuel level: 50 gal
```

Ours SteerDistill

```
fillFuelTank(fuelAmount=50)
↳Cannot fill above tank capacity
displayCarStatus("fuel")
↳Fuel level: 15 gal
fillFuelTank(fuelAmount=35)
✓Fuel level: 50 gal
```

Figure 13: Fuel-capacity recovery. After an excessive refill is rejected, SteerDistill observes the existing 15 gallons and adds 35 gallons to reach the 50-gallon capacity; BF16 checks the fuel level before refilling.

(b) Tool-name recovery

Context. Fill the tank for a road trip; the fuel-status tool is named displayCarStatus, and its response contains lengthy metadata.

Ref BF16

```
displayCarStatus("fuel")
↳Fuel level: 15 gal
fillFuelTank(fuelAmount=35)
✓Fuel level: 50 gal
```

Ours SteerDistill

```
display_car_status("fuel")
↳Unknown method 'display_car_status'
displayCarStatus("fuel")
↳Fuel level: 15 gal
fillFuelTank(fuelAmount=35)
✓Fuel level: 50 gal
```

Figure 14: Tool-name recovery. SteerDistill repairs the rejected `display_car_status` call to `displayCarStatus`, obtains the fuel level, and completes the 35-gallon refill; BF16 uses the valid interface on its first attempt.