OPQAD: Recovering Capability under KV-Cache Quantization via On-Policy Quantization-Aware Distillation

Anonymous authors
Paper under double-blind review

Abstract

QAD is a common approach for recovering capability loss in post-training quantization (PTQ). However, naive off-policy QAD recovers only 41.5% of the loss for a 1.5B model under TurboQuant K2V3. To understand this bottleneck, we analyze the issue from two perspectives: quantization-induced reasoning degradation and the intrinsic limitations of off-policy QAD. We find that quantization creates a distributional "flattening effect" and suppresses key high-confidence tokens, while off-policy QAD suffers from exposure bias and layer-wise representation drift. Building on these insights, we propose On-Policy Quantization-Aware Distillation (OPQAD), which integrates an on-policy backbone, reverse KL divergence, layer-wise hidden-state alignment, and a gated forward KL to correct suppressed tokens. Evaluated across 1.5B–8B models and long CoT tasks (up to 32K length), OPQAD improves AIME24 math accuracy by 6.0–7.3 percentage points over off-policy QAD under K2V3, raising the 1.5B model's recovery rate to 58.7%, while also yielding modest gains under OSCAR. Trained solely on math data, OPQAD generalizes to out-of-distribution code and instruction-following tasks with 9.7% training overhead. Ablations confirm that OPQAD reduces student-prefix distribution errors, hidden-layer drift, and extreme probability inversions, validating its effectiveness for low-bit KV cache recovery.

1 Introduction

Large language models reuse the KV cache to accelerate attention-based inference (Vaswani et al., 2017; Kwon et al., 2023), but its memory footprint grows with context length. Low-bit KV cache quantization reduces this cost (Liu et al., 2024c; Hooper et al., 2024; Su et al., 2025b) at the risk of capability loss. Although OSCAR outperforms KIVI, TurboQuant+, and RotateKV on benchmarks such as LongBench-E (Bai et al., 2023; Su et al., 2026), it still causes a 6–7% performance drop on Qwen3-4B/8B in our evaluation. Quantization-aware distillation on offline teacher trajectories (off-policy QAD) matches teacher and student output distributions (Liu et al., 2023b; Du et al., 2024; Xin et al., 2026), yet recovers only 41.5% of the quantization loss for Nemotron-Research-Reasoning-Qwen-1.5B (BroRL) (Hu et al., 2025) under TurboQuant K2V3.

We diagnose four obstacles to recovery. Quantization flattens output distributions unevenly and collapses probabilities at a small number of high-confidence tokens, often disrupting exact computation. Quantization error also accumulates across layers; hidden-state alignment helps repair the resulting distribution mismatch. Finally, off-policy QAD suffers from exposure bias: teacher trajectories used in training differ from the quantized student's inference-time prefixes (Bengio et al., 2015; Ranzato et al., 2015; Arora et al., 2022).

These findings motivate OPQAD. It uses on-policy student trajectories to reduce exposure bias (Agarwal et al., 2023; Gu et al., 2023), reverse KL to counteract flattening, layer-wise alignment of hidden-state norms and directions to reduce representation drift (Romero et al., 2014; Yang et al., 2026), and gated squared forward KL to restore severely underestimated token probabilities.

On-policy quantization-aware training remains underexplored, partly because quantized rollouts are difficult to integrate efficiently into inference engines. Our OSCAR 2-bit implementation incurs only 9.7% additional total training cost over off-policy QAD. Generation takes 19.6% less time de-

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.

spite 1.25× longer responses, showing that efficient quantized rollouts can make on-policy recovery practical.

We evaluate TurboQuant K2V3 and OSCAR on 1.5B, 4B, and 8B models across math, code, and instruction-following tasks, including reasoning with generation lengths up to 32K. OPQAD improves AIME24 accuracy by 6.0–7.3 percentage points over off-policy QAD under K2V3, with gains also observed under OSCAR. For BroRL 1.5B, it raises quantization-loss recovery from 41.5% to 58.7%. Math-only training also improves recovery on code and instruction-following tasks. Component ablations link these gains to the diagnosed failure mechanisms.

2 Diagnosing Quantization-Induced Reasoning Failures

2.1 How KV Cache Quantization Interferes with Reasoning

Hypotheses on the interference mechanism of quantization. We first investigate the mechanism through which KV cache quantization affects the reasoning ability of reasoning models. Before conducting experiments, we formulate the following hypotheses:

(1) **Gaussian noise:** quantization adds random Gaussian noise to the output distribution of the BF16 model;
(2) **Mode collapse:** quantization collapses the output onto a few modes of the BF16 model;
(3) **Distribution flattening:** quantization flattens the sharp probability distribution of the model.

For Hypothesis (1), we compute the full-vocabulary log-probability ratio $\ln(q_v/p_v)$ (logit decomposition in Appendix B.3); for Hypotheses (2) and (3), we compare entropy and reverse versus forward KL. Appendix A.5 gives the statistical protocol.

Empirical validation. The full-vocabulary mean of $\ln(q_v/p_v)$ decreases as p_v increases (Figure 1(a)(d)). Adding Gaussian noise to BF16 pre-softmax logits fails to reproduce the entropy and KL changes caused by quantization. Across model scales and quantization methods, entropy generally increases and reverse KL exceeds forward KL; for Qwen3-4B K2V3, entropy rises from 0.27 to 0.47 and reverse KL is 2.5 times forward KL (Figure 1(b)(c)(e)(f)). Together, the probability-dependent shift, failed Gaussian control, increased entropy, and KL asymmetry support Hypothesis (3), distribution flattening.

Two effects of flattening. We further decompose flattening into two effects, **in-place contraction** and **off-position diffusion**, and analyze them on models of three scales (1.5B, 4B, and 8B), using trajectories from the BF16 model and the quantized model separately.

Let $p_t(x)$ and $q_t(x)$ denote the probabilities assigned to word x by the BF16 model and the quantized model, respectively. I_x is the set of positions where the reference trajectory actually outputs x, and O_x is the set of remaining positions; t ranges over all analyzed trajectory positions.

In-place contraction: We define $C_x = \frac{1}{|I_x|}\sum_{t\in I_x}\big[p_t(x) - q_t(x)\big]$, the average probability decrease of x at positions where token x actually appears, to measure how quantization compresses the probability of the token at its own position.

Off-position diffusion: We define $D_x = \frac{\sum_{t\in O_x} q_t(x)}{\sum_{t\in O_x} p_t(x)}$, the ratio of change in the total probability mass of x over all positions where x does not actually appear, to measure the proportional increase in off-position token probability after quantization.

Figure 1(i)–(l) shows in-place contraction across token types but distinct diffusion preferences. Hesitation tokens such as wait diffuse strongly: their probability falls at intended checking positions but rises elsewhere, potentially causing missed checks and unnecessary backtracking.

Probability collapse. For BF16 top-1 tokens with $p_t(v_t^*) \geq 0.7$, we measure severe drops through the ratio $p_t(v_t^*)/q_t(v_t^*)$ (Appendix B.4). Collapse rarely affects hesitation tokens but frequently hits digits, symbols, and operators needed for exact computation (Figure 1(h)).

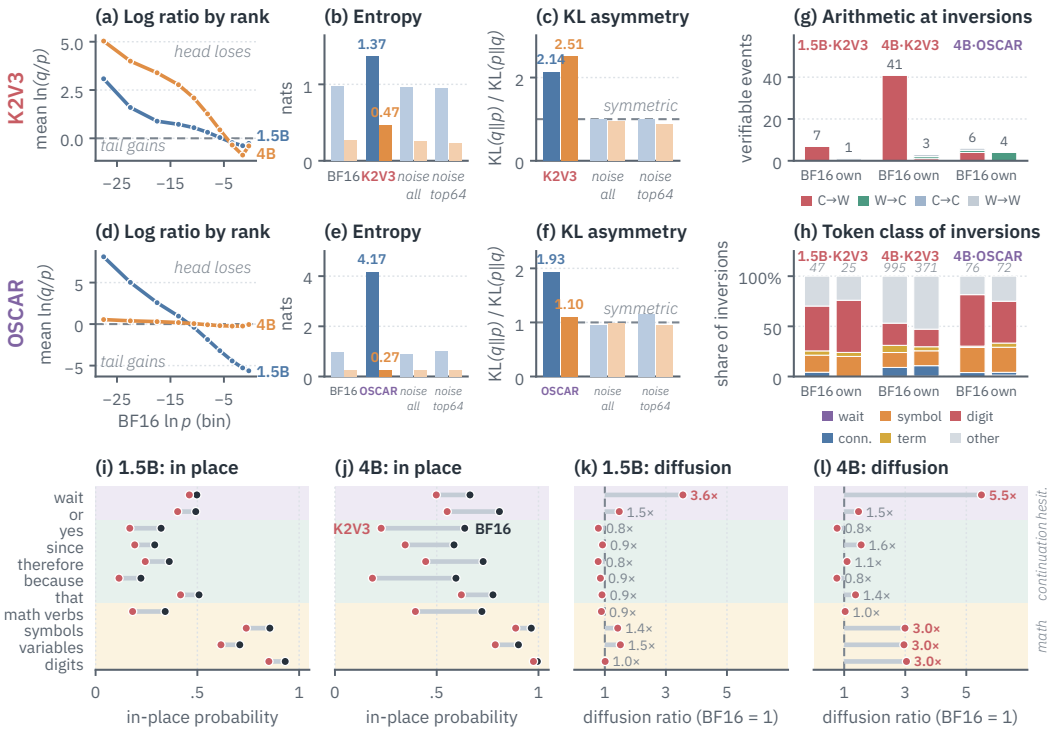

Figure 1: Effect of KV cache quantization. (a)(d) Mean $\ln(q/p)$; (b)(e) entropy; (c)(f) ratio of reverse KL to forward KL. "noise" denotes the reproduction experiment that adds variance-matched i.i.d. Gaussian noise position-wise to BF16 logits. (g) Change in correctness of verifiable arithmetic expressions at extreme-inversion positions before and after quantization; (h) category proportions of extreme-inversion tokens; BF16/own denote BF16 trajectories and the quantized model's self-generated trajectories, respectively. (i)–(l) Per-word in-place probability and off-position diffusion ratio under K2V3 .

Probability collapse correlates with erroneous reasoning. We extract expressions at positions of probability collapse and study how their correctness changes before and after quantization. As shown in (Figure 1(g)), we find that the quantized model tends to turn correct answers into incorrect ones. Table 1 gives two concrete examples for 1.5B and 4B.

Table 1: Arithmetic cases at extreme-inversion positions

Model	Expression and prefix	Correct form	BF16 preference	K2V3 preference	Original token prob.	Ratio
Qwen3-4B	$20+21+63=1$	104	**1**0 ... (100.00%)	**1**4 ... (99.85%)	0.01%	7162.04
BroRL 1.5B	$1126\times13=1463$	14638	1463**8** ... (99.99%)	1463**2** ... (95.11%)	0.02%	4559.40

Therefore, the mechanism by which KV cache quantization interferes with reasoning ability can be summarized in two aspects: the flattening effect based on in-place contraction and off-position diffusion (Finding 1), and extreme probability inversion of sensitive tokens (Finding 2).

2.2 Recovery-Time Bottlenecks of QAD

Layer-wise hidden-state alignment aids output matching. Quantization perturbations propagate across Transformer blocks. To test whether hidden-state alignment repairs output distributions, we replace one quantized block's output at a time with its BF16 counterpart on the same prefix. Replacing deeper blocks progressively reduces KL divergence and repairs flattening (Appendix B.5).

2.3 Off-Policy Distillation and Exposure Bias

Identifying exposure bias in off-policy QAD. Reverse KL can counteract flattening (Section 5), but off-policy reverse-KL QAD recovers BroRL 1.5B K2V3 accuracy from 14.4% to only 31.3% at step300, versus 55.8% for BF16.

We attribute part of the remaining gap to the mismatch between training and inference trajectories. Following Arora et al. (2022), ExAccErr measures the relative cumulative KL difference between teacher and student-generated prefixes.

Figure 2 shows that off-policy training has only slightly higher teacher-prefix error but sharply higher error on student prefixes. On-policy QAD maintains lower KL error and greater stability over longer trajectories.

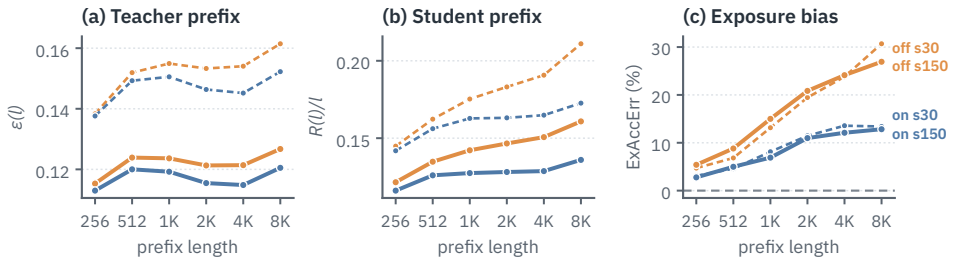

Figure 2: Prefix error and exposure bias of on-policy vs. off-policy training (BroRL 1.5B K2V3; dashed: step30, solid: step150): (a) teacher-prefix error; (b) student-prefix error; (c) ExAccErr.

We summarize the findings from the above analysis of how quantization damage interferes with reasoning, together with the diagnosis of existing QAD recovery methods, as follows: **Finding 1: Quantization flattens the original probability distribution unevenly.**

Finding 2: Quantization causes probability collapse at a small number of high-confidence positions, which often steers the model toward erroneous reasoning.

Finding 3: Quantization error accumulates layer by layer along depth, and layer-wise hidden-state alignment can repair the flattening.

Finding 4: Off-policy training suffers from exposure bias between training and evaluation, with significantly higher error on self-generated prefixes.

3 OPQAD Method

OPQAD targets the four failure patterns diagnosed in Section 2: exposure bias, flattening, representation drift, and critical-token probability collapse (Figure 3).

3.1 On-policy Quantization-aware Distillation

We first adopt on-policy QAD as the overall training framework, which addresses the exposure bias of off-policy training. In addition, we replace the forward KL of classical QAD (Xin et al., 2026) with a full-vocabulary reverse KL to more effectively counteract the flattening effect of quantization. This loss term is given by Eq. (1) (Gu et al., 2023; Agarwal et al., 2023).

$$\begin{aligned}\mathcal{L}_{\text{OPD}}^{\text{rev}} &= \mathbb{E}_{y^S_{<t}\sim q_\theta}\left[D_{\text{KL}}(q_\theta(\cdot \mid y^S_{<t}, \mathbf{x}) \,\|\, p_T(\cdot \mid y^S_{<t}, \mathbf{x}))\right] \\ &= \mathbb{E}_{y^S_{<t}\sim q_\theta}\left[\sum_{v\in\mathcal{V}} q_\theta(v \mid y^S_{<t}, \mathbf{x}) \log \frac{q_\theta(v \mid y^S_{<t}, \mathbf{x})}{p_T(v \mid y^S_{<t}, \mathbf{x})}\right].\end{aligned} \tag{1}$$

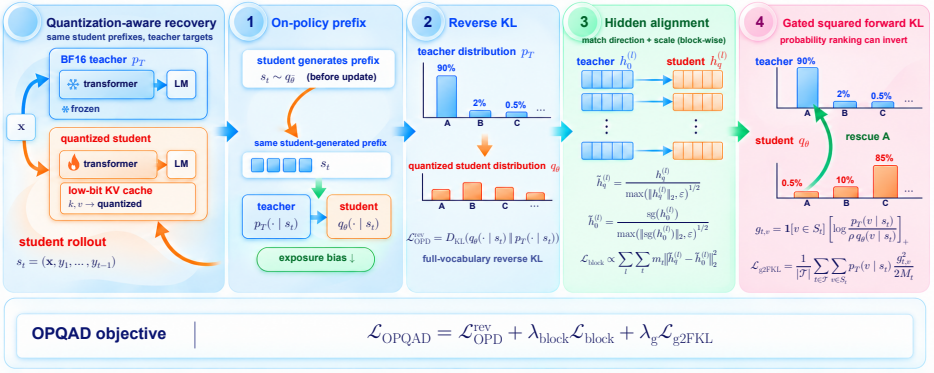

Figure 3: Overall framework of OPQAD. (1) On-policy QAD; (2) full-vocabulary reverse KL; (3) hidden-state alignment; (4) gated squared forward KL.

3.2 Block-wise Hidden-state Alignment

The analysis in Section 2.2 shows that aligning hidden states benefits distribution matching. We therefore use hidden-state alignment as an auxiliary objective. Since the BF16 model and the quantized model share the same architecture, hidden-state alignment is naturally feasible.

The vanilla OPD proposed by OPRD also uses hidden-state alignment as a supervision target, but it cannot constrain norm drift. We treat the exponent of its denominator as a tunable parameter α to balance the alignment of hidden-state norm and direction. Our method uses $\alpha = 1/2$; Appendix D.3 reports other choices of α and the corresponding analysis.

$$\tilde{h}_{q,t}^{(l)} = \frac{h_{q,t}^{(l)}}{\max\left(\|h_{q,t}^{(l)}\|_2, \varepsilon\right)^{1/2}}, \qquad \tilde{h}_{0,t}^{(l)} = \frac{\mathrm{sg}\left(h_{0,t}^{(l)}\right)}{\max\left(\left\|\mathrm{sg}\left(h_{0,t}^{(l)}\right)\right\|_2, \varepsilon\right)^{1/2}}.$$

When the hidden-state norms on both sides are at least $\varepsilon > 0$, the loss of this module is computed as

$$\mathcal{L}_{\mathrm{block}} = \frac{1}{|\mathcal{L}| \sum_t m_t} \sum_{l \in \mathcal{L}} \sum_t m_t \left\|\tilde{h}_{q,t}^{(l)} - \tilde{h}_{0,t}^{(l)}\right\|_2^2, \qquad \mathcal{L}_{\mathrm{align}} = \lambda_{\mathrm{block}} \mathcal{L}_{\mathrm{block}}.$$

Appendix C.3 provides the decomposition of the hidden state alignment term into direction and norm components.

3.3 Gated Squared Forward KL for Critical Token Recovery

Section 2.1 reveals that a small number of mathematical reasoning positions exhibit extreme probability inversion of high-confidence tokens. The reverse KL term alone cannot impose an effective penalty on these tokens to which the student assigns low probability. We therefore apply a gated squared forward KL loss to severely underestimated tokens in the teacher top-k, providing an additional recovery signal at these positions.

Let s_t be the prefix generated by the quantized model, $S_t = \mathrm{TopK}(p_T(\cdot \mid s_t))$, and $M_t = \sum_{v \in S_t} p_T(v \mid s_t)$. Given a tolerance factor $\rho > 1$, we define

$$g_{t,v} = \mathbf{1}[v \in S_t] \left[\log \frac{p_T(v \mid s_t)}{\rho q_\theta(v \mid s_t)}\right]_+, \qquad \mathcal{L}_{\mathrm{g2FKL}} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{\sum_{v \in S_t} p_T(v \mid s_t) g_{t,v}^2}{2M_t}.$$

Here, M_t is the normalizing teacher head mass. To strengthen the penalty on above-threshold log-ratios, we use $g_{t,v}^2$ instead of $g_{t,v}$, yielding a squared KL divergence.

Positions severely underestimated by the quantized model receive stronger recovery gradients. Appendix C.2 provides a gradient analysis of how gated squared forward KL, compared with reverse

KL, recovers extremely low-probability tokens, and Appendix D.4 reports the sensitivity analysis of top-k and ρ.

Therefore, the final training objective of OPQAD is

$$\mathcal{L}_{\text{OPQAD}} = \mathcal{L}_{\text{OPD}}^{\text{rev}} + \lambda_{\text{block}}\mathcal{L}_{\text{block}} + \lambda_{\text{g}}\mathcal{L}_{\text{g2FKL}}.$$

4 Experiments

4.1 Experimental Setup

We evaluate two quantization methods—OSCAR 2-bit (Su et al., 2026) and TurboQuant K2V3 (Zandieh et al., 2025)—across three model scales: BroRL 1.5B (Hu et al., 2025), Qwen3-4B, and Qwen3-8B (Qwen Team, 2025)[1]. By default, we adopt off-policy QAD with reverse KL as our primary baseline, alongside forward KL QAD for the 1.5B model. Training is conducted on the Skywork dataset (He et al., 2025) with a maximum sequence length of 16K. For off-policy QAD, we pre-generate an offline trajectory dataset comparable in size to the on-policy training samples.

Baseline selection. Existing QAD methods employ various distillation targets: LLM-QAT uses soft-label cross-entropy (Liu et al., 2023b), BitDistiller (CAKLD) combines forward and reverse KL (Du et al., 2024), and NVFP4 QAD minimizes forward KL for better computational efficiency (Xin et al., 2026). We thus treat forward and reverse KL distillation as representative, strong baselines for off-policy QAD.

Implementation details and evaluation protocols are elaborated in Appendix A.

4.2 Main Results

Table 2 summarizes the main evaluation results across all models and quantization schemes. On mathematical reasoning tasks, OPQAD consistently outperforms off-policy QAD across all model scales and quantization setups. Under K2V3, OPQAD improves Math Avg over reverse-KL QAD by 4.17–7.29 percentage points, including a 6.04–7.30 percentage point gain on AIME24. Notably, recovery performance depends heavily on the underlying quantization scheme: while OPQAD yields more modest gains on Qwen3-4B/8B under OSCAR (1.80 and 3.54 Math Avg points over direct quantization), it still consistently surpasses reverse-KL QAD.

4.3 Cross-task Capability Recovery

To evaluate cross-task recovery, we additionally evaluate on instruction-following and code tasks. Although the advantage of OPQAD over off-policy QAD in cross-task recovery is considerably smaller than on mathematical tasks, OPQAD still achieves better overall cross-task recovery. Across the 5 model–quantization settings and the two task categories (code and instruction following), OPQAD matches or outperforms off-policy QAD in all cases except instruction following on Qwen3-4B with OSCAR.

5 Analysis and Ablations

5.1 OPQAD Component Ablations

To analyze the role of each OPQAD component, we ablate four components: on/off-policy, reverse/forward KL, hidden state alignment, and gated forward KL. Compared with forward KL QAD, OPQAD improves Math Avg by 5.42 percentage points and raises the quantization-loss recovery rate from 41.5% to 58.7%. Following the analysis methodology in Section 2, we further examine whether each module resolves the problem it is designed to address. Appendix A.5 details the implementation and evaluation protocol.

[1]BroRL 1.5B experiences severe generation degradation under OSCAR quantization, so we report only its K2V3 results (see Appendix B.3 for detailed analysis).

Table 2: Results on mathematics, code, and instruction-following tasks. Math Avg averages AIME24/25/26; Code Avg averages LCB and HE+.

Method	Math				Code			IF
	AIME24 avg@16	AIME25 avg@16	AIME26 avg@16	Avg.	LCB avg@2	HE+ avg@2	Avg.	IFEval avg@2
BroRL 1.5B								
BF16	55.83	35.63	37.08	42.85	30.29	77.44	53.86	76.92
K2V3	14.38	13.33	6.46	11.39	14.86	57.93	36.39	69.00
+ reverse KL QAD	31.25	24.79	19.58	25.21	**24.00**	70.73	47.37	71.10
+ forward KL QAD	30.42	23.96	18.96	24.44	**24.00**	71.34	47.67	70.86
+ OPQAD	**38.54**	**26.46**	**24.58**	**29.86**	23.71	**71.95**	**47.83**	**71.40**
Qwen3-4B								
BF16	73.54	63.33	65.83	67.57	41.43	89.02	65.23	89.15
K2V3	3.33	1.88	1.88	2.36	7.71	55.18	31.45	76.38
+ reverse KL QAD	39.58	30.42	35.21	35.07	29.43	80.49	54.96	**85.55**
+ OPQAD	**46.88**	**37.71**	**42.50**	**42.36**	**30.86**	**81.40**	**56.13**	**85.55**
Oscar	64.38	58.54	**61.25**	61.39	39.14	85.98	62.56	87.41
+ reverse KL QAD	64.17	**59.79**	59.17	61.04	**41.14**	86.59	63.86	**88.67**
+ OPQAD	**70.83**	59.17	59.58	**63.19**	40.00	**88.41**	**64.21**	87.47
Qwen3-8B								
BF16	74.38	65.00	67.29	68.89	44.57	87.50	66.04	90.89
K2V3	16.88	15.83	14.17	15.63	13.71	74.39	44.05	85.37
+ reverse KL QAD	43.13	31.04	37.29	37.15	31.71	**85.06**	58.39	86.63
+ OPQAD	**49.17**	**33.13**	**41.67**	**41.32**	**33.71**	83.84	**58.78**	**87.59**
Oscar	69.17	56.67	60.00	61.94	39.14	86.89	63.02	89.21
+ reverse KL QAD	72.92	58.13	61.88	64.31	**39.71**	84.45	62.08	89.03
+ OPQAD	**73.33**	**60.63**	**62.50**	**65.48**	**39.71**	**87.20**	**63.45**	**90.29**

Table 3: Component ablations and the loss × policy comparison on BroRL 1.5B K2V3. All results are at step300; AIME24/25/26 each use avg16. In the left table, Avg is the equally weighted mean over the three test sets; the right table reports Math Avg (%) over AIME24/25/26. Δ is in percentage points.

(a) Component ablations

Loss / Branch	AIME24	AIME25	AIME26	Avg	Δ Avg
OPQAD	**38.54**	*26.46*	**24.58**	**29.86**	0.00
− hidden state alignment	*36.46*	**26.88**	22.50	*28.61*	-1.25
− gated forward KL	35.42	25.00	*24.17*	28.19	-1.67
− on-policy (off)	31.04	24.17	21.46	25.56	-4.31
Reverse KL (off)	31.25	24.79	19.58	25.21	-4.65
Forward KL (off)	30.42	23.96	18.96	24.44	-5.42

(b) Math Avg: Loss × policy

Loss	Off-policy	On-policy	Δ policy
Reverse KL	25.21	27.36	+2.15
OPQAD	25.56	**29.86**	**+4.31**
Δ loss	+0.35	+2.50	+2.15

On-policy QAD reduces exposure bias. On BroRL 1.5B, on- and off-policy training have similar KL errors on BF16 prefixes but differ sharply on self-generated trajectories (Section 2.3). On-policy exposure bias remains stable with length, whereas off-policy ExAccErr reaches 4.17 times the on-policy value at 8K (Figure 4(a)–(c)). Replacing only OPQAD's on-policy sampling with off-policy sampling lowers Math Avg by 4.31 points and AIME24 by 7.50 points.

Reverse KL controls the mode-spreading errors. Section 2.1 shows that the flattening effect is an important mechanism through which quantization interferes with reasoning. Whereas reverse KL is mode-seeking, forward KL tends to cover all patterns of the teacher and is weaker at restoring in-position probability and at expelling excess diffusion. Figure 5(a)–(c) systematically compares the two, showing that reverse KL better recovers from flattening; its insufficient recovery of extremely underestimated tokens is compensated by the gated squared forward KL.

Hidden state alignment pulls hidden states closer. On shared prefixes, OPQAD reduces the all-layer average half-norm distance to BF16 by 47%, versus 39% without alignment (Figure 4(d));

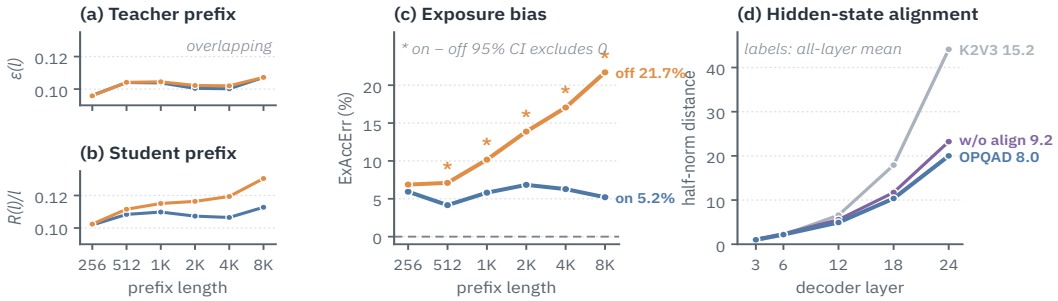

Figure 4: On-policy and hidden state alignment ablations. (a)(b) Per-token KL error on teacher prefixes and student prefixes; (c) ExAccErr as a function of prefix length, where * indicates that the per-problem paired 95% interval of on−off excludes 0; (d) layer-wise half-norm hidden-state distance to BF16, with the all-layer average shown on the right.

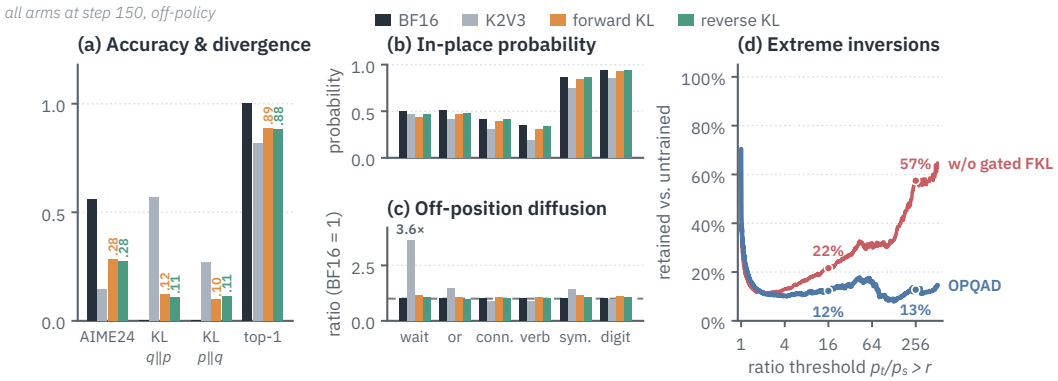

Figure 5: KL direction and gated forward KL ablations (BroRL 1.5B K2V3, same batch of BF16 trajectories). (a)(b)(c): comparison of forward / reverse KL in accuracy and in recovery from the flattening effect; (d) under fixed prefixes, the retention ratio of probability-collapse tokens after OPQAD training relative to the untrained K2V3 model.

removing alignment lowers Math Avg by 1.25 points. With alignment, last-layer, middle-layer, and all-layer average cosine distances are also lower on all 20 trajectories (Appendix D.5).

Gated forward KL cuts extreme inversions. On fixed BF16 prefixes, OPQAD reduces extreme probability ratios across all measured thresholds (Figure 5(d)). Among positions with BF16 top-1 probability above 70%, the number assigned student probability below 1% falls to 1/9 of the untrained count. Without the gate, more than half of the tokens with ratios above 256 remain—4.5 times as many as with the gate (Appendix D.2).

5.2 Training Cost and Throughput

On one NVIDIA B300, Qwen3-4B-Thinking-2507 with OSCAR 2-bit incurs 9.7% more total training cost under on-policy QAD (Table 4). This comparison includes quantized rollouts before each on-policy update and BF16 offline generation for off-policy QAD. Despite 1.25× longer responses and higher update costs, on-policy generation takes 19.6% less time: greater concurrency at low bit widths offsets quantized decoding and step-wise inference overhead. Appendix A.4 gives the protocol.

Table 4: Wall-clock cost of 5 training steps (min), Qwen3-4B-Thinking-2507 with OSCAR on one B300 (batch 224, 32K context). Training stages: mean of 3 steady steps $\times 5$; offline generation measured on 1,120 rollouts.

	BF16 offline gen.	Online gen./load	Teacher scoring	Actor update	Total	Mean resp. length
On-policy	—	97.06	106.02	74.95	278.54	13,644
Off-policy	116.10	0.15	81.35	55.92	253.88	10,912

6 Related Work

KV Cache Quantization. KIVI and KVQuant reduce cache memory through fine-grained quantization and outlier handling (Liu et al., 2024c; Hooper et al., 2024); TurboQuant and OSCAR use randomized rotations and scale optimization (Zandieh et al., 2025; Su et al., 2026). OPQAD complements these schemes by recovering output distributions and long-chain reasoning after quantization.

Post-Training Quantization. GPTQ uses approximate second-order information, AWQ protects salient activation channels, and SmoothQuant balances quantization difficulty through equivalent scaling (Frantar et al., 2022; Lin et al., 2023; Xiao et al., 2022). QuaRot and SpinQuant improve quantizability through rotations (Ashkboos et al., 2024; Liu et al., 2024b). OPQAD instead targets distribution matching on inference-time student trajectories.

Quantization-Aware Training. LLM-QAT distills with soft-label cross-entropy, BitDistiller combines forward and reverse KL, and NVFP4 QAD matches teacher distributions (Liu et al., 2023b; Du et al., 2024; Xin et al., 2026). MiniLLM and GKD use student-generated prefixes to align training with inference (Gu et al., 2023; Agarwal et al., 2023). OPQAD combines this on-policy principle with losses targeting representation drift and critical-token probability collapse under KV cache quantization. Appendix E gives an extended review.

7 Conclusion

OPQAD addresses four barriers to capability recovery under low-bit KV cache quantization: distribution flattening, critical-token probability collapse, representation drift, and off-policy exposure bias. It combines student-generated trajectories, reverse KL, layer-wise hidden-state alignment, and gated squared forward KL. Across TurboQuant K2V3 and OSCAR, multiple model scales, and math, code, and instruction-following tasks, OPQAD improves mathematical recovery and matches or exceeds off-policy baselines in most cross-task settings. These results support designing recovery objectives around quantization failure mechanisms.

AI Use Statement

AI Assistance: Yes—to aid or polish writing; for retrieval and discovery (e.g., finding related work); for research ideation or execution; and to draft sections of the paper. Details are described in the paper.

Reproducibility Statement

Appendix A.1 specifies the model initialization, training data, optimization settings, and checkpoint selection. Appendix A.3 gives the evaluation tasks, sampling settings, and output budgets, and Appendix A.5 describes the diagnostic trajectories and statistics. The recovery-gradient derivation and hidden-state loss decomposition appear in Appendices C.2 and C.3. Training-cost measurements are detailed in Appendix A.4.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes, 2023. URL https://arxiv.org/abs/2306.13649.

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Cheung. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 700–710, 2022. doi: 10.18653/v1/2022.findings-acl.58. URL https://aclanthology.org/2022.findings-acl.58/.

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, 2024. URL https://arxiv.org/abs/2404.00456.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. BinaryBERT: Pushing the Limit of BERT Quantization, 2020. URL https://arxiv.org/abs/2012.15701.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding, 2023. URL https://arxiv.org/abs/2308.14508.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks, 2015. URL https://arxiv.org/abs/1506.03099.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-Bit Quantization of Large Language Models With Guarantees. *arXiv preprint arXiv:2307.13304*, 2023. URL https://arxiv.org/abs/2307.13304.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient Quantization-Aware Training for Large Language Models, 2024. URL https://arxiv.org/abs/2407.11062.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized Clipping Activation for Quantized Neural Networks, 2018. URL https://arxiv.org/abs/1805.06085.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *arXiv preprint arXiv:2208.07339*, 2022. URL https://arxiv.org/abs/2208.07339.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs, 2023a. URL https://arxiv.org/abs/2305.14314.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression. *arXiv preprint arXiv:2306.03078*, 2023b. URL https://arxiv.org/abs/2306.03078.

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, 2024. URL https://arxiv.org/abs/2402.10631.

Dayou Du, Shijie Cao, Jianyi Cheng, Luo Mai, Ting Cao, and Mao Yang. BitDecoding: Unlocking Tensor Cores for Long-Context LLMs with Low-Bit KV Cache, 2025. URL https://arxiv.org/abs/2503.18773.

Haojie Duanmu, Zhihang Yuan, Xiuhong Li, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. SKVQ: Sliding-window Key and Value Cache Quantization for Large Language Models, 2024. URL https://arxiv.org/abs/2405.06219.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme Compression of Large Language Models via Additive Quantization. *arXiv preprint arXiv:2401.06118*, 2024. URL https://arxiv.org/abs/2401.06118.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned Step Size Quantization, 2019. URL https://arxiv.org/abs/1902.08153.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. *arXiv preprint arXiv:2210.17323*, 2022. URL https://arxiv.org/abs/2210.17323.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-Policy Distillation of Large Language Models, 2023. URL https://arxiv.org/abs/2306.08543.

Insu Han, Praneeth Kacham, Amin Karbasi, Vahab Mirrokni, and Amir Zandieh. PolarQuant: Quantizing KV Caches with Polar Transformation, 2025. URL https://arxiv.org/abs/2502.02617.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork Open Reasoner 1 Technical Report, 2025. URL https://arxiv.org/abs/2505.22312.

Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. ZipCache: Accurate and Efficient KV Cache Quantization with Salient Token Identification, 2024. URL https://arxiv.org/abs/2405.14256.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, 2015. URL https://arxiv.org/abs/1503.02531.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization, 2024. URL https://arxiv.org/abs/2401.18079.

Jian Hu, Mingjie Liu, Ximing Lu, Fang Wu, Zaid Harchaoui, Shizhe Diao, Yejin Choi, Pavlo Molchanov, Jun Yang, Jan Kautz, and Yi Dong. BroRL: Scaling Reinforcement Learning via Broadened Exploration, 2025. URL https://arxiv.org/abs/2510.01180.

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, 2025. URL https://arxiv.org/abs/2510.11696.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference, 2017. URL https://arxiv.org/abs/1712.05877.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code, 2024. URL https://arxiv.org/abs/2403.07974.

Zhongping Ji. IsoQuant: Hardware-Aligned SO(4) Isoclinic Rotations for LLM KV Cache Compression, 2026. URL https://arxiv.org/abs/2603.28430.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. GEAR: An Efficient KV Cache Compression Recipe for Near-Lossless Generative Inference of LLM, 2024. URL https://arxiv.org/abs/2403.05527.

Jangho Kim, Yash Bhalgat, Jinwon Lee, Chirag Patel, and Nojun Kwak. QKD: Quantization-aware Knowledge Distillation, 2019. URL https://arxiv.org/abs/1911.12491.

Minsoo Kim, Sihwa Lee, Janghwan Lee, Sukjin Hong, Du-Seong Chang, Wonyong Sung, and Jungwook Choi. Token-Scaled Logit Distillation for Ternary Weight Generative Language Models, 2023a. URL https://arxiv.org/abs/2308.06744.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-Sparse Quantization. *arXiv preprint arXiv:2306.07629*, 2023b. URL https://arxiv.org/abs/2306.07629.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention, 2023. URL https://arxiv.org/abs/2309.06180.

Xing Li, Zeyu Xing, Yiming Li, Linping Qu, Hui-Ling Zhen, Wulong Liu, Yiwu Yao, Sinno Jialin Pan, and Mingxuan Yuan. KVTuner: Sensitivity-Aware Layer-Wise Mixed-Precision KV Cache Quantization for Efficient and Nearly Lossless LLM Inference, 2025a. URL https://arxiv.org/abs/2502.04420.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the Limit of Post-Training Quantization by Block Reconstruction. *arXiv preprint arXiv:2102.05426*, 2021. URL https://arxiv.org/abs/2102.05426.

Zeyu Li, Chuanfu Xiao, Yang Wang, Xiang Liu, Zhenheng Tang, Baotong Lu, Mao Yang, Xinyu Chen, and Xiaowen Chu. AnTKV: Anchor Token-Aware Sub-Bit Vector Quantization for KV Cache in Large Language Models, 2025b. URL https://arxiv.org/abs/2506.19505.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *arXiv preprint arXiv:2306.00978*, 2023. URL https://arxiv.org/abs/2306.00978.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 Quantization and System Co-design for Efficient LLM Serving, 2024. URL https://arxiv.org/abs/2405.04532.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation, 2023a. URL https://arxiv.org/abs/2305.01210.

Shih-Yang Liu, Maksim Khadkevich, Nai Chit Fung, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, Jan Kautz, Yu-Chiang Frank Wang, Pavlo Molchanov, and Min-Hung Chen. EoRA: Fine-tuning-free Compensation for Compressed LLM with Eigenspace Low-Rank Approximation. *arXiv preprint arXiv:2410.21271*, 2024a. URL https://arxiv.org/abs/2410.21271.

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, 2023b. URL https://arxiv.org/abs/2305.17888.

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. *arXiv preprint arXiv:2405.16406*, 2024b. URL https://arxiv.org/abs/2405.16406.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache, 2024c. URL https://arxiv.org/abs/2402.02750.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits, 2024. URL https://arxiv.org/abs/2402.17764.

Asit Mishra and Debbie Marr. Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy, 2017. URL https://arxiv.org/abs/1711.05852.

Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or Down? Adaptive Rounding for Post-Training Quantization. *arXiv preprint arXiv:2004.10568*, 2020. URL https://arxiv.org/abs/2004.10568.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A White Paper on Neural Network Quantization. *arXiv preprint arXiv:2106.08295*, 2021. URL https://arxiv.org/abs/2106.08295.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization, 2018. URL https://arxiv.org/abs/1802.05668.

Qwen Team. Qwen3 Technical Report, 2025. URL https://arxiv.org/abs/2505.09388.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence Level Training with Recurrent Neural Networks, 2015. URL https://arxiv.org/abs/1511.06732.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for Thin Deep Nets, 2014. URL https://arxiv.org/abs/1412.6550.

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. *arXiv preprint arXiv:2308.13137*, 2023. URL https://arxiv.org/abs/2308.13137.

Akshat Sharma, Hangliang Ding, Jianping Li, Neel Dani, and Minjia Zhang. MiniKV: Pushing the Limits of LLM Inference via 2-Bit Layer-Discriminative KV Cache, 2024. URL https://arxiv.org/abs/2411.18077.

Alina Shutova, Vladimir Malinovskii, Vage Egiazarian, Denis Kuznedelev, Denis Mazur, Nikita Surkov, Ivan Ermakov, and Dan Alistarh. Cache Me If You Must: Adaptive Key-Value Quantization for Large Language Models, 2025. URL https://arxiv.org/abs/2501.19392.

Yi Su, Yuechi Zhou, Quantong Qiu, Juntao Li, Qingrong Xia, Ping Li, Xinyu Duan, Zhefeng Wang, and Min Zhang. Accurate KV Cache Quantization with Outlier Tokens Tracing, 2025a. URL https://arxiv.org/abs/2505.10938.

Zunhai Su and Kehong Yuan. KVSink: Understanding and Enhancing the Preservation of Attention Sinks in KV Cache Quantization for LLMs, 2025. URL https://arxiv.org/abs/2508.04257.

Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Linge Li, Huangqi Yu, and Kehong Yuan. RotateKV: Accurate and Robust 2-Bit KV Cache Quantization for LLMs via Outlier-Aware Adaptive Rotations, 2025b. URL https://arxiv.org/abs/2501.16383.

Zunhai Su, Rui Yang, Chao Zhang, Yaxiu Liu, Yifan Zhang, Wei Wu, Jing Xiong, Dayou Du, Xialie Zhuang, Yulei Qian, Yuchen Xie, Yik-Chung Wu, Hongxia Yang, and Ngai Wong. OScaR: The Occam's Razor for Extreme KV Cache Quantization in LLMs and Beyond, 2026. URL https://arxiv.org/abs/2605.19660.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. FlatQuant: Flatness Matters for LLM Quantization. *arXiv preprint arXiv:2410.09426*, 2024. URL https://arxiv.org/abs/2410.09426.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even Better LLM Quantization with Hadamard Incoherence and Lattice Codebooks. *arXiv preprint arXiv:2402.04396*, 2024. URL https://arxiv.org/abs/2402.04396.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, 2017. URL https://arxiv.org/abs/1706.03762.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier Suppression: Pushing the Limit of Low-bit Transformer Language Models. *arXiv preprint arXiv:2209.13325*, 2022. URL https://arxiv.org/abs/2209.13325.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. *arXiv preprint arXiv:2211.10438*, 2022. URL https://arxiv.org/abs/2211.10438.

Meng Xin, Sweta Priyadarshi, Jingyu Xin, Bilal Kartal, Aditya Vavre, Asma Kuriparambil Thekkumpate, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Ido Shahaf, Akhiad Bercovich, Kinjal Patel, Suguna Varshini Velury, Chenjie Luo, Zhiyu Cheng, Jenny Chen, Chen-Han Yu, Wei Ping, Oleg Rybakov, Nima Tajbakhsh, Oluwatobi Olabiyi, Dusan Stosic, Di Wu, Song Han, Eric Chung, Sharath Turuvekere Sreenivas, Bryan Catanzaro, Yoshi Suhara, Tijmen Blankevoort, and Huizi Mao. Quantization-Aware Distillation for NVFP4 Inference Accuracy Recovery, 2026. URL https://arxiv.org/abs/2601.20088.

Shenzhi Yang, Guangcheng Zhu, Bowen Song, Haobo Wang, Mingxuan Xia, Xing Zheng, Yingfan Ma, Zhongqi Chen, Weiqiang Wang, Junbo Zhao, and Gang Chen. OPRD: On-policy representation distillation, 2026.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers. *arXiv preprint arXiv:2206.01861*, 2022. URL https://arxiv.org/abs/2206.01861.

Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. RPTQ: Reorder-based Post-training Quantization for Large Language Models. *arXiv preprint arXiv:2304.01089*, 2023. URL https://arxiv.org/abs/2304.01089.

Amir Zandieh, Majid Daliri, and Insu Han. QJL: 1-Bit Quantized JL Transform for KV Cache Quantization with Zero Overhead, 2024. URL https://arxiv.org/abs/2406.03482.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate, 2025. URL https://arxiv.org/abs/2504.19874.

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. TernaryBERT: Distillation-aware Ultra-low Bit BERT, 2020. URL https://arxiv.org/abs/2009.12812.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models, 2023. URL https://arxiv.org/abs/2311.07911.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients, 2016. URL https://arxiv.org/abs/1606.06160.

A Experimental Settings

A.1 Models and Training

We use Nemotron-Research-Reasoning-Qwen-1.5B-brorl, Qwen3-4B, and Qwen3-8B as BF16 teachers. The student is initialized from the same weights, only its KV cache is quantized, and training updates only the student parameters. The training data is the Skywork math subset (difficulty 1–15, 16K version). We use a global batch size of 64, one response per prompt, AdamW, a learning rate of 1.5×10^{-6} (constant after 5 warmup steps), weight decay 0, prompt/response limits of 512/16,384 tokens, and a rollout temperature and top-p of 1. OPQAD uniformly uses $\lambda_{\text{block}} = 0.02$ (half-norm) and, for the gated forward KL, top-$k = 4$, $\rho = 4$, and $\lambda_{\text{g}} = 1$; the sensitivity analyses are in Appendix D.3 and Appendix D.4. The off-policy baseline uses reverse KL, disables the gating and representation-alignment terms, and trains on 9,600 trajectories pre-generated by the corresponding BF16 model; for BroRL 1.5B we additionally include a forward-KL off-policy baseline. We evaluate the step300 checkpoint, except for Qwen3-8B K2V3 and OSCAR, which use step90 and step120, respectively.

A.2 Quantization Configuration

K2V3 keeps the KV cache of the first and last two layers in BF16 and uses a 2-bit Lloyd-Max codebook for keys and 3 bits for values. OSCAR applies INT2 to all layers and additionally stores 64 sink tokens and 256 recent tokens in BF16; its rotation matrices are calibrated on GPQA Diamond for 4B and on Skywork-hard for 8B. The effective bit rates, including quantization metadata and the BF16 portions, are given in Table 5.

Table 5: Effective bit rate of the KV cache (bits/element, averaged over all layers)

Quantization method	BF16 layers	Context length	BroRL 1.5B (28 layers)	Qwen3-4B / 8B (36 layers)
K2V3	first/last 2 layers	any	4.59	4.17
OSCAR	none	8K	2.79	2.79
OSCAR	none	16K	2.52	2.52
OSCAR	none	32K	2.38	2.38

A.3 Evaluation

Thinking is enabled for all benchmarks, and the engine context is 33,792. Code tasks use LiveCodeBench v6 (Jain et al., 2024) and HumanEval+ (Liu et al., 2023a), and instruction following uses IFEval (Zhou et al., 2023); sampling parameters and output budgets are listed in Table 6. For AIME we report the mean accuracy over 16 responses per problem; for code tasks we estimate pass@1 from two samples per problem; for IFEval we report instruction-level strict accuracy. Math Avg and Code Avg are the unweighted means over the corresponding benchmarks.

Table 6: Evaluation sampling and output budgets. The output budget covers both thinking and the final answer.

Benchmark	**Problems × samples**	**Total**	**Temp.**	**top-p**	**top-k**	**Max output**
AIME24	30 × 16	480	0.7	0.95	disabled	32,768 tokens
AIME25	30 × 16	480	0.7	0.95	disabled	32,768 tokens
AIME26	30 × 16	480	0.7	0.95	disabled	32,768 tokens
IFEval	541 × 2	1,082	0.9	0.95	20	16,384 tokens
LiveCodeBench v6	175 × 2	350	0.9	0.95	20	16,384 tokens
HumanEval+	164 × 2	328	0.9	0.95	20	16,384 tokens

A.4 Training Cost Measurement

Table 4 compares the wall-clock time of 5 training steps for on-policy and off-policy training on a single B300. The student is Qwen3-4B-Thinking-2507 with OSCAR 2-bit KV quantization, and the teacher is the corresponding BF16 model; both use a batch size of 224 and a 32K context limit, with the rollout engine's GPU memory utilization set to the largest runnable value. Each on-policy step consists of online generation by the quantized student, teacher scoring, and the actor update. For off-policy training, the trajectories are generated in bulk by the BF16 model before training, and the training phase only loads trajectories, followed by teacher scoring and the actor update. For each training stage we average the time over 3 stable steps and multiply by 5 to exclude warmup; the off-policy offline generation time is measured directly on $224 \times 5 = 1{,}120$ rollouts, matching the amount used in 5 training steps, and is included in its total time. The total time also includes a small amount of other overhead not listed separately, and is therefore slightly larger than the sum of the stages. Under this protocol, the total times for on-policy and off-policy training are 278.54 and 253.88 minutes, respectively, with the former 9.71% slower.

A.5 Analysis Settings

Trajectory selection. The background analysis uses AIME24 problem IDs 60–69 with samples 0/1 per problem, for a total of 20 trajectories, keeping the first 8,192 response positions of each. The teacher and the quantized student run forward on the same fixed prefix, and probabilities are computed by a softmax over the full logits (temperature 1); trajectories are generated with temperature 0.7, top-p 0.95, and a 32K output limit. BF16 trajectories are used for fixed teacher-prefix comparisons, and quantized trajectories for the prefixes visited by each method itself. Under this setting, the distribution diagnostics cover about 126K and 145K token positions on BroRL 1.5B and Qwen3-4B, respectively.

Full vocabulary and sparse storage. KL, entropy, and top-1 statistics are computed over the full vocabulary during the forward pass. The union data takes the union of the teacher's and student's nuclei, each with cumulative mass 0.9999, keeping at most 8,192 tokens per position; when the limit is exceeded, tokens are kept in order of $\max(p_v, q_v)$, and the actually covered mass on each side is recorded. The stored log-probabilities use the normalization of the full softmax. Word-group probabilities come from the stored nucleus or union entries, with entries outside the set counted as zero; when coverage is incomplete, the error is bounded by the actual missing mass. The full-vocabulary logit-shift test and the Gaussian control do not go through nucleus filtering; their settings are given in Appendix B.3.

Averaging and word groups. KL, entropy, and top-1 agreement are averaged over all retained positions pooled together, so longer trajectories contribute more positions. The in-place probability is averaged over the positions where the word group is actually output, and the diffusion ratio is the sum of the student's mass over the remaining positions divided by the corresponding sum of the teacher's mass. Vocabulary tokens are decoded individually, stripped of leading and trailing whitespace, lowercased, and merged with their variants. The connective group is yes/since/therefore/because/that; the math verbs are substitute/substitution/express/expression/compute/plug; the math symbols are $=, +, -, *, /, \wedge, (,)$ and the backslash; and the digits are 0–9. The digit plot uses the mass of the whole group, which differs from the average of per-digit diffusion ratios.

Extreme inversion and arithmetic checks. Let $v_t^* = \arg\max_v p_t(v)$; an event requires $p_t(v_t^*) \geq 0.7$ and $p_t(v_t^*)/q_t(v_t^*) > r$. The denominator of the background ratio is all retained positions, including low-confidence ones; the continuous curves take $r \in [1, 512]$. Token categorization uses the teacher top-1 token at which the event occurs and counts events. The arithmetic check fixes the remaining characters of the original trajectory, substitutes the teacher and student top-1 tokens in turn, and verifies the local equation via whitelisted expression parsing and exact rational arithmetic; elided, unclosed, or ambiguous expressions are excluded. Of the 1,586 candidate events, 62 are checkable; the correctness changes in the figure describe local substitutions under a fixed suffix.

Hidden-state patching. Figure 9 uses the 20 BF16 trajectories above with BroRL 1.5B K2V3. We first cache the output of every BF16 decoder block, and then, in the quantized forward pass, replace only the residual-stream output of the specified block with the corresponding BF16 output, while subsequent blocks continue along the quantized path. We test each block except the last one; no patching serves as the reference configuration, and patching the last layer directly recovers the BF16

output, so this trivial endpoint is omitted. Per-word probabilities use the nucleus union, and KL takes the full-vocabulary value.

Post-training representation distance. Figure 4 compares the untrained K2V3, OPQAD at step300, and the branch without hidden-state alignment, all using the same 20 BF16 trajectories. We compute the per-position squared distance between hidden vectors after half-norm (summed over the hidden dimension), with a norm floor of 10^{-6}; across trajectories we weight by the number of response positions, and across layers we take the arithmetic mean. We use the decoder hidden states, where the last layer is the state after the final norm. The teacher after training is always the original BF16 model.

KL direction and gated recovery. Figure 5(a)–(c) uses the off-policy step150 checkpoints trained with pure forward/reverse KL, on the same batch of BF16 trajectories with 125,530 positions in total. For BF16 itself, the KL is 0 and the top-1 agreement and diffusion ratio are 1. Figure 5 uses the step300 branches; the retention ratio at threshold r is defined as the number of events after training divided by the number of events for the untrained K2V3, with both counts using the same set of fixed prefixes and the same high-confidence condition. For the trajectory-paired interval of the inversion ratio, we resample the 20 trajectories 10,000 times, add 0.5 to both numerator and denominator, and take the 2.5/97.5 percentiles.

Exposure bias and training dynamics. The step300 exposure-bias analysis uses 30 problems with 16 samples each, at lengths 256/512/1024/2048/4096/8192, pairing problem/sample instances for which both sides reach the target length; the on/off difference is resampled over problems 2,000 times. The formulas are given in Appendix B.6. In the training dynamics, each branch uses its own on-policy batches: overlap is the size of the intersection of the teacher and student top-512 sets divided by 512, and intersection mass is the probability of the original distribution summed over the intersection. The overlap curves take the mean over the most recent 10 logged points, the mass curves keep the original logged points, and the final-step scatter points use the step300 metrics.

B Supplementary Background Analysis

This section provides the complete numbers for Section 2.1. Unless otherwise stated, the subject is the untrained K2V3; the trajectory and statistics settings are given in Appendix A.5.

B.1 Word-Level Analysis of Quantization-Induced Output Lengthening

Output length decomposition. The statistics cover all AIME24 evaluation samples, and the reasoning is split into segments at transition words (wait, but, alternatively, etc.); the degeneration rate is the fraction of samples containing consecutive 8-gram repetition. Results are shown in Table 7. For 1.5B, the lengthening comes almost entirely from frequent transitions: more segments, shorter segments, more wait, less verify, and almost no repetition. 4B and 8B likewise show more transitions, compounded by repetition degeneration; after removing degenerate samples, their median lengths are 16931 and 20080, respectively, still clearly longer than BF16.

Table 7: Output length, segmentation, and degeneration statistics

Model	Method	Accuracy	Median length	Median segs.	Words / seg.	Degen. rate	wait / 1k words	verify / 1k words
1.5B	BF16	0.583	7257	74	51	0	16.2	2.55
1.5B	k2v3	0.143	9818	178	30	0.003	32.3	1.34
4B	BF16	0.734	12988	99	83	0.004	6.2	3.80
4B	k2v3	0.029	32623	176	89	0.713	11.3	0.62
8B	BF16	0.757	13192	104	78	0.002	7.3	4.12
8B	k2v3	0.169	26768	236	51	0.388	16.1	1.97

Fixed-prefix measurement. For each model, we take 20 trajectories generated by the BF16 model itself (the first 10 problems, 2 samples per problem), run teacher-forced on the same prefix, and

read the next-token distributions of BF16 and K2V3 at every position. Probabilities are taken from the nucleus with target cumulative mass 0.9999, with tokens outside the nucleus counted as 0. A "word" comprises all tokens that, after decoding, stripping whitespace, and lowercasing, equal that word. For a word w, let $\mathcal{T}_w$ be the set of positions in the BF16 trajectories where w is actually output: (a) the in-place probability is the mean of $p(w)$ and $q(w)$ over $t \in \mathcal{T}_w$; (b) the diffusion ratio is $\sum_{t \notin \mathcal{T}_w} q_t(w) \,/\, \sum_{t \notin \mathcal{T}_w} p_t(w)$, i.e., the ratio of the expected occurrence counts under the two models at positions where BF16 does not say w. This section uses full trajectories; to align with the trained models, Appendix D.1 truncates each trajectory to its first 8,192 positions, so the wait diffusion ratio of the untrained 1.5B K2V3 is 3.64 there (3.56 here), a difference that stems solely from the statistics range.

For 1.5B/4B/8B, wait diffusion ratios are 3.56/5.50/4.20. At BF16 end positions, the mean end-token probability changes from 0.933 to 0.619 for 1.5B and remains 1.00 for 4B and 8B; these end-position estimates use only about 20 positions per model.

Table 8 and Table 9 give the diffusion ratio and in-place probability of each word group, respectively. The in-place probability decreases for every word, corresponding to the increase in entropy. Diffusion ratios well above 1 concentrate on wait, followed by or, math symbols, and variables; connectives and math verbs have diffusion ratios close to or below 1, i.e., they are only suppressed without compensation. "Digits, total" counts only the probability moved onto "any digit", whereas "digits, per-digit mean" also includes substitutions among digits; the gap between the two corresponds to the probability of writing one digit as another, i.e., calculation errors. The total for 1.5B is about 1, indicating that its digit diffusion comes almost entirely from such substitutions.

Table 8: Off-position diffusion ratio of word groups

Group	**Word**	**1.5B**	**4B**	**8B**
Hesitation	wait	3.56	5.50	4.20
Hesitation	or	1.47	1.47	1.94
Connective	yes	0.79	0.77	1.12
Connective	since	0.92	1.56	1.30
Connective	therefore	0.78	1.09	1.10
Connective	because	0.86	0.76	0.76
Connective	that	0.91	1.37	1.33
Math	Symbols = + − * / ^ ( ) \, total	1.42	2.99	2.56
Math	Variables x/n/m, total	1.51	2.96	2.19
Math	Digits, total	1.01	3.04	2.20
Math	Digits, per-digit mean	~ 1.3	~ 5.5	~ 3.9
Math verbs	substitute/plug/compute/express etc., total	0.89	1.03	1.19

Table 9: In-place probability of word groups: BF16 vs. K2V3

Group	**1.5B**	**4B**	**8B**
wait	0.497→0.461	0.663→0.498	0.690→0.536
or	0.492→0.403	0.808→0.551	0.794→0.554
yes	0.32→0.17	0.64→0.23	0.63→0.26
since	0.290→0.192	0.585→0.345	0.613→0.367
because	0.22→0.12	0.59→0.18	0.51→0.19
therefore	0.36→0.24	0.73→0.45	0.70→0.41
that	0.508→0.417	0.776→0.621	0.825→0.710
Math verbs	0.34→0.18	0.72→0.40	0.73→0.48
Math symbols	0.86→0.74	0.97→0.89	0.97→0.91
Variables x/n/m, total	0.708→0.616	0.902→0.788	0.914→0.824
Digits	0.93→0.85	0.995→0.975	0.994→0.975

Figure 6 shows the per-word in-place probability and diffusion ratio for Qwen3-8B, consistent with the 1.5B and 4B findings in Figure 1(i)–(l).

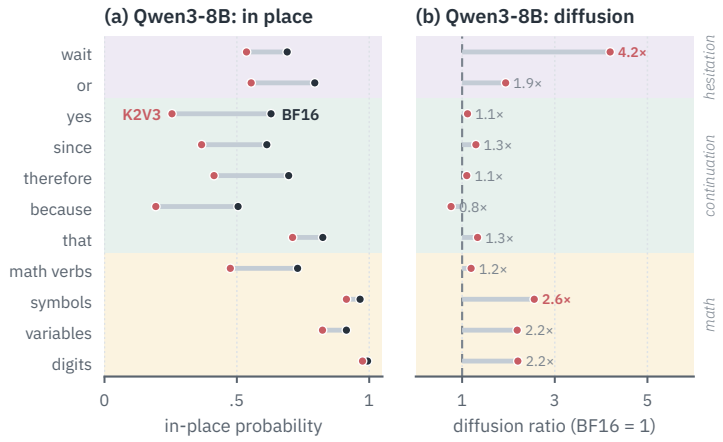

Figure 6: Per-word in-place probability (a) and off-position diffusion ratio (b) for Qwen3-8B K2V3 on BF16 trajectories.

Limitations. Each model has only 20 BF16 trajectories, so low-frequency words (substitute, plug, etc.) only indicate direction; all probabilities come from teacher-forced forward passes with HF fake-quant rather than native vLLM decoding. The corresponding statistics for trained models are given in Table 18.

B.2 Hypotheses 2 and 3: Entropy and KL Direction

Table 10 reports the entropy and bidirectional KL of the untrained quantized models and the BF16 models on the same prefixes, averaged over all retained positions. In "trajectory → method", the former determines the source of the prefix and the latter is the quantization used in the forward pass; statistics are computed on the union of the teacher's and student's 99.99% nuclei, and the last two columns give the actual mass of this union covered on each side.

Table 10: Entropy and KL direction of untrained quantized models. Values are computed on the union of the teacher's and student's 99.99% nuclei, with coverage in the last two columns; they differ slightly from the exact full-vocabulary values in Figure 1 and Table 12 (e.g., 1.5B BF16 entropy 0.955 vs. 0.979), without affecting the direction.

Model	Trajectory → method	$D(q\|p)$	$D(p\|q)$	$H(p)$	$H(q)$	Coverage p	Coverage q
BroRL 1.5B	BF16 → K2V3	0.576	0.270	0.955	1.334	0.999917	0.999719
BroRL 1.5B	K2V3 → K2V3	0.292	0.211	1.163	1.365	0.999830	0.999650
Qwen3-4B	BF16 → K2V3	1.008	0.404	0.263	0.463	0.999996	0.999934
Qwen3-4B	K2V3 → K2V3	0.332	0.288	0.395	0.427	0.999978	0.999939
Qwen3-4B	BF16 → OSCAR	0.072	0.065	0.263	0.270	0.999982	0.999975
Qwen3-4B	OSCAR → OSCAR	0.065	0.068	0.298	0.285	0.999976	0.999978
Qwen3-8B	BF16 → K2V3	0.673	0.300	0.264	0.422	0.999996	0.999944
Qwen3-8B	K2V3 → K2V3	0.306	0.242	0.372	0.414	0.999979	0.999951
Qwen3-8B	BF16 → OSCAR	0.059	0.050	0.264	0.291	0.999986	0.999972
Qwen3-8B	OSCAR → OSCAR	0.051	0.049	0.284	0.296	0.999981	0.999973

At all three scales, K2V3 satisfies $H(q) > H(p)$ and $D(q\|p) > D(p\|q)$, most clearly on BF16 trajectories. $D(q\|p)$ grows when the quantized model places mass on tokens to which the teacher assigns low probability, corresponding to a flatter distribution; if the quantized model instead collapsed onto

a few teacher modes, one would more likely see a larger $D(p\|q)$ and a decrease in entropy. For OSCAR, both the entropy and KL changes are small, and entropy decreases slightly on 4B self-generated trajectories, consistent with a small perturbation.

B.3 Full-Vocabulary Logit Shift and Gaussian Control

Setup. To avoid nucleus selection bias and the Jensen effect of softmax, this section tests over the full vocabulary before the softmax. The models are BroRL 1.5B and Qwen3-4B, with the same trajectories as above (20 BF16 trajectories, first 8,192 response positions); the first position of each trajectory falls within the BF16 prefill and is not quantized, so it is removed, leaving 125,510 (1.5B) and 145,458 (4B) positions. The vocabulary is the tokenizer's actual vocabulary (151,665 for 1.5B and 151,669 for 4B), excluding the padding rows of lm_head. Let z^p, z^q be the pre-softmax logits and $\Delta z_v = z_v^q - z_v^p$ their difference before and after quantization. Then

$$\ln q_v - \ln p_v = \Delta z_v - \big(\mathrm{lse}(z^q) - \mathrm{lse}(z^p)\big),$$

where $\mathrm{lse}(z) = \ln \sum_u e^{z_u}$. Thus, at a given position, $\ln q_v - \ln p_v$ and Δz_v differ only by a constant. All statistics are computed exactly over the full vocabulary during the forward pass; for the scatter plots, we additionally sample 256 vocabulary tokens uniformly per position and add the BF16 top-32.

Shape and dependence. The upper half of Table 11 shows the shape of Δz after per-position standardization and pooling: unweighted, it is close to Gaussian with slightly heavy tails (Figure 7(c)). However, this statistic is determined almost entirely by long-tail tokens; when weighted by p, the standardized mean is -3.46 (1.5B) and -5.79 (4B), and the fraction with $|z| > 3$ is 59% and 90%, i.e., the tokens carrying the probability mass fall predominantly in the left tail. The lower half shows that the shift is negatively correlated with the BF16 logit; on 4B, this contraction is equivalent to multiplying the logits by about 0.66–0.71, i.e., a temperature of about 1.4–1.5. Figure 7(a) bins by BF16 $\ln p$: the more confident BF16 is about a token, the more its logit is pushed down, monotonically and systematically rather than as zero-mean jitter; $\ln(q/p)$ is positive in the tail and negative in the head, which is the result of this contraction after softmax normalization.

Table 11: Shape of the full-vocabulary logit shift Δz and its dependence on the BF16 logit (K2V3, BF16 trajectories)

Statistic	BroRL 1.5B	Qwen3-4B
Per-position std(Δz), median (p10–p90)	0.61 (0.42–0.95)	1.40 (1.02–1.91)
std(Δz) within BF16 top-64, median	1.26	2.63
Pooled skewness	0.31	-0.01
Pooled excess kurtosis	1.21	0.74
Fraction with $\|z\| > 3$ (normal: 0.27%)	0.78%	0.56%
Fraction with $\|z\| > 5$	0.031%	0.024%
corr(Δz, z^p), full-vocabulary per-position mean	-0.15	-0.61
corr, BF16 top-32 only	-0.33	-0.61
Slope a ($z^q \approx az^p + b$), uniform sampling, median (p10–p90)	0.95 (0.86–1.02)	0.71 (0.52–0.85)
Slope a, BF16 top-32 only, median	0.83	0.66
Skewness / excess kurtosis of residual after removing linear term	0.42 / 1.13	0.21 / 0.64

Counterfactual Gaussian control. We add $\varepsilon_v \sim \mathcal{N}(0, \sigma^2)$ to the BF16 logits at each position and then apply softmax, with σ set to the full-vocabulary std(Δz) at that position (vocabulary σ) or the std(Δz) within the BF16 top-64 (top-64 σ). Table 12 shows that under neither setting does independent Gaussian noise reproduce the measurements: K2V3 increases entropy by 0.39 (1.5B) / 0.21 (4B), whereas Gaussian noise leaves entropy unchanged or slightly lower; the measured reverse KL is about 2.1–2.5 times the forward KL, whereas the two are nearly equal under Gaussian noise. Top-64 σ already matches the measurements in total KL and top-1 agreement, yet its entropy and KL direction remain opposite. Therefore, the entropy increase and the KL asymmetry stem from the systematic dependence of the shift on z^p, not from the noise magnitude.

OSCAR uses the same configuration as logitdist (INT2, all layers, a BF16 window of 64 sink + 256 recent tokens, clip 0.96/0.92, rotation calibrated on Skywork-hard with sequence length 20,000). For both models the direction agrees with K2V3: corr$(\Delta z, z^p)$ is -0.45 (1.5B) and -0.23 (4B), reverse KL exceeds forward KL, and entropy increases. The perturbation of 4B OSCAR is itself an order of magnitude smaller than that of K2V3 (entropy increases by only 0.008), consistent with its near-BF16 accuracy, so on its own it is weak evidence; 1.5B OSCAR (all layers) is in a collapsed state (top-1 agreement 0.10), consistent with its PTQ accuracy of 0, and is not a suitable representative of the "flattening" mechanism.

Table 12: Measured quantization vs. variance-matched i.i.d. Gaussian noise control (full vocabulary, BF16 trajectories). q(top-1) is the probability the quantized distribution assigns to the BF16 top-1 token.

Model	Distribution	Entropy	KL($q\|p$)	KL($p\|q$)	reverse / forward	top-1 agreement	q(top-1)
BroRL 1.5B	BF16	0.979	–	–	–	–	0.710
	K2V3 measured	**1.367**	0.572	0.268	**2.13**	0.817	0.597
	Gaussian (vocabulary σ)	0.968	0.076	0.077	0.99	0.861	0.690
	Gaussian (top-64 σ)	0.950	0.332	0.335	0.99	0.748	0.639
	OSCAR measured	4.165	10.36	5.36	1.93	0.096	0.053
Qwen3-4B	BF16	0.266	–	–	–	–	0.899
	K2V3 measured	**0.471**	1.022	0.407	**2.51**	0.823	0.764
	Gaussian (vocabulary σ)	0.254	0.114	0.120	0.95	0.901	0.864
	Gaussian (top-64 σ)	0.232	0.481	0.422	0.88	0.826	0.808
	OSCAR measured	0.274	0.075	0.068	1.10	0.939	0.879

B.4 Extreme Inversion of High-Confidence Tokens

Let $v_t^* = \arg\max_v p_t(v)$. An extreme-inversion event requires $p_t(v_t^*) \geq 0.7$ and $p_t(v_t^*)/q_t(v_t^*) > r$. Table 13 gives the number of events at each threshold and their fraction among all positions; K2V3 is most severe on 4B, and the fraction on self-generated trajectories is lower than on BF16 trajectories. Table 14 categorizes events by the teacher top-1 token at which they occur: hesitation words are almost never extremely inverted, whereas digits, symbols, and connectives account for a considerable share. Table 15 compares, at checkable arithmetic positions, the correctness of the local equation after substituting the teacher versus student top-1 token; on BF16 trajectories, all checkable K2V3 events are "correct→wrong".

Table 13: Extreme-inversion events of high-confidence tokens: counts (rate per thousand positions in parentheses; denominator is all positions)

Model / trajectory→method	All positions N	$p_{\text{top1}} \geq 0.7$	$r > 4$	$r > 16$	$r > 64$	$r > 256$
1.5B BF16→K2V3	125,530	69,798	1,986 (15.8)	440 (3.51)	132 (1.05)	47 (0.374)
1.5B K2V3→K2V3	161,782	78,599	798 (4.93)	193 (1.19)	74 (0.457)	25 (0.155)
4B BF16→K2V3	145,478	122,939	8,059 (55.4)	3,538 (24.3)	1,842 (12.7)	995 (6.84)
4B K2V3→K2V3	163,840	126,694	3,362 (20.5)	1,415 (8.64)	705 (4.30)	371 (2.26)
4B BF16→OSCAR	145,478	122,939	604 (4.15)	221 (1.52)	119 (0.818)	76 (0.522)
4B OSCAR→OSCAR	151,247	124,901	541 (3.58)	198 (1.31)	119 (0.787)	72 (0.476)

B.5 Hidden-State Patching

Figure 9 replaces the output of a specified decoder block with the BF16 output in the quantized forward pass of BroRL 1.5B K2V3 (settings in Appendix A.5). The output KL decreases as the patching depth increases, and the diffusion and in-place probability of each word group recover accordingly.

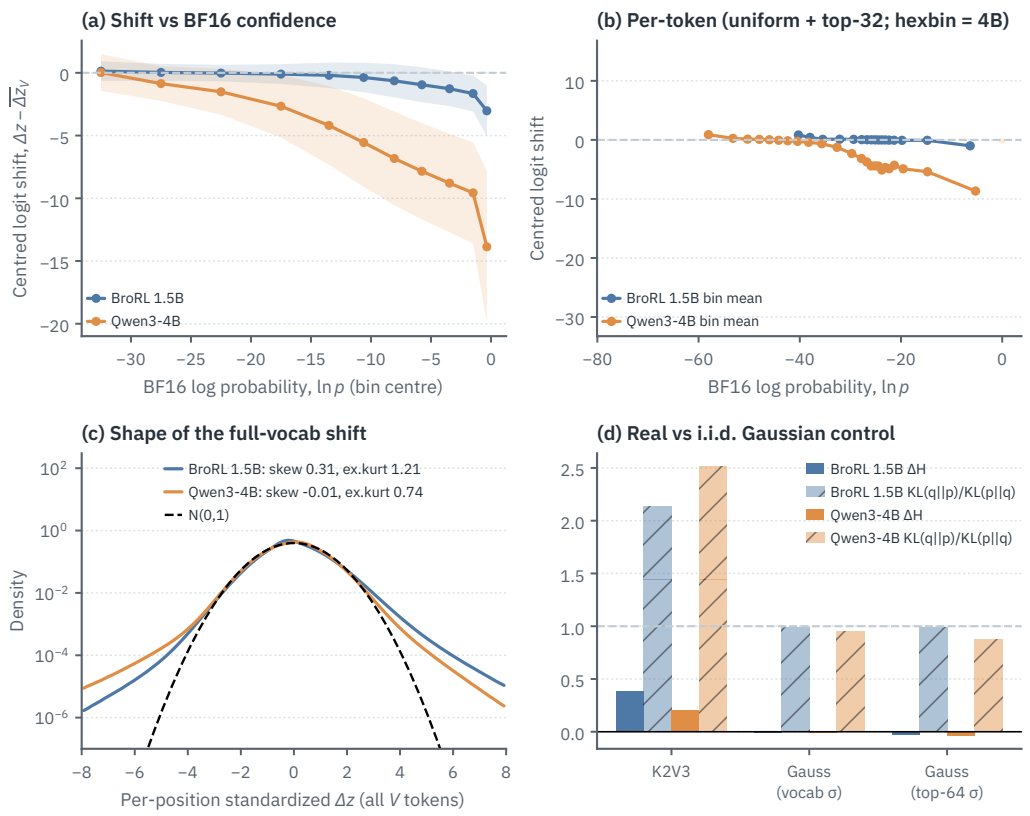

Figure 7: Full-vocabulary logit shift (K2V3, BF16 trajectories). (a) Relative logit shift binned by BF16 $\ln p$, with shading showing ± 1 std; (b) per-token scatter (20 positions per trajectory, 256 uniformly sampled tokens plus the BF16 top-32; hexbin for 4B), with lines showing bin means; (c) distribution of Δz after per-position standardization and pooling (log-scale y-axis), with the dashed line showing $\mathcal{N}(0, 1)$; (d) entropy increase ΔH (solid) and KL ratio (hatched) for measured K2V3 and the two Gaussian controls, with the dashed line marking a ratio of 1.

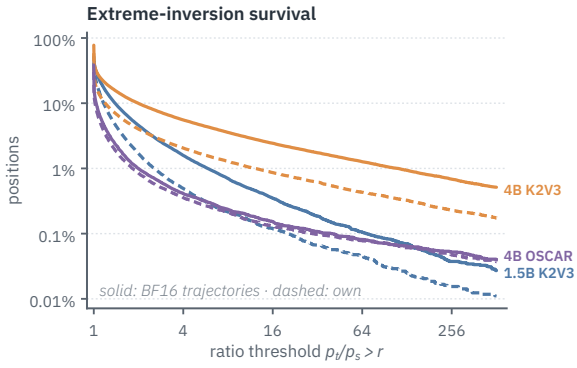

Figure 8: Fraction of BF16 high-confidence tokens ($p_{\text{top1}} \geq 0.7$) that are extremely inverted in the quantized model, as a function of the threshold r; the denominator is all positions. Solid lines: BF16 trajectories; dashed lines: trajectories self-generated by the quantized model.

Table 14: Category counts of extremely inverted tokens

Model / trajectory→method	Total	wait	Connectives	Math symbols	Math words	Digits
4B BF16→K2V3	995	1	89	148	71	220
4B BF16→OSCAR	76	0	3	19	1	39
4B K2V3→K2V3	371	1	39	55	15	65
4B OSCAR→OSCAR	72	1	2	18	3	30
1.5B BF16→K2V3	47	0	2	8	2	21
1.5B K2V3→K2V3	25	0	0	5	1	13

Table 15: Arithmetic check at extreme-inversion positions: counts (fraction of checkable events)

Group	All >256 events	Checkable	Correct → wrong	Wrong → correct	Correct → correct	Wrong → wrong
1.5B BF16→K2V3	47	7	7 (1.0)	0 (0.0)	0 (0.0)	0 (0.0)
1.5B K2V3→K2V3	25	1	0 (0.0)	0 (0.0)	0 (0.0)	1 (1.0)
4B BF16→K2V3	995	41	41 (1.0)	0 (0.0)	0 (0.0)	0 (0.0)
4B K2V3→K2V3	371	3	1 (0.333)	1 (0.333)	0 (0.0)	1 (0.333)
4B BF16→OSCAR	76	6	4 (0.667)	1 (0.167)	0 (0.0)	1 (0.167)
4B OSCAR→OSCAR	72	4	0 (0.0)	4 (1.0)	0 (0.0)	0 (0.0)
Total	1586	62	53 (0.855)	6 (0.097)	0 (0.0)	3 (0.048)

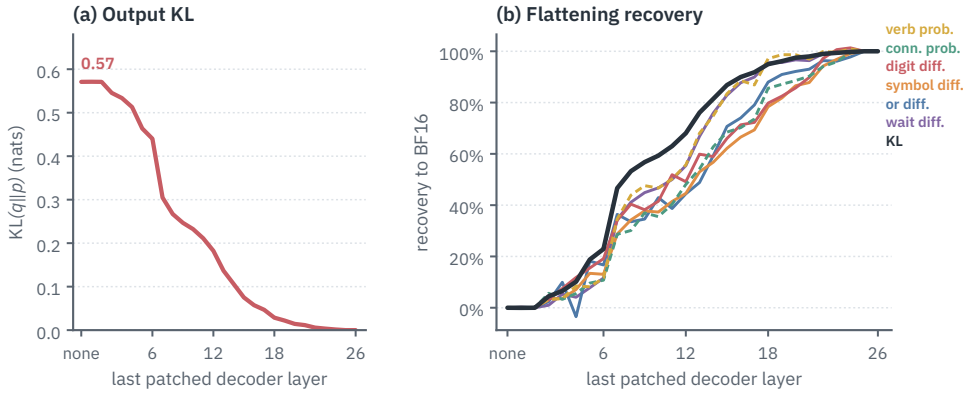

Figure 9: Effect of hidden-state patching on KL divergence and distribution flattening. (a) Output KL$(q\|p)$; (b) recovery fraction of each metric toward BF16 relative to the unpatched configuration. The x-axis is the patched decoder block.

B.6 Definition and Per-Length Statistics of Exposure Bias

Following Arora et al. (2022), ExAccErr within prefix length l is defined as

$$\text{ExAccErr}(l) = \frac{R(l) - l\epsilon(l)}{l\epsilon(l)} \times 100\%,$$

where $R(l)$ is the regret accumulated up to step l when the student generates freely, with the teacher distribution p as the imitation target:

$$R(l) = \sum_{t=1}^{l} \mathbb{E}_{\substack{y_{<t}\sim q_\theta(\cdot|\mathbf{x}) \\ y_t \sim p(\cdot|y_{<t},\mathbf{x})}} \log \frac{p(y_t \mid y_{<t}, \mathbf{x})}{q_\theta(y_t \mid y_{<t}, \mathbf{x})},$$

and $\epsilon(l)$ is the average per-step error between q_θ and p when teacher-sampled prefixes serve as the oracle context:

$$\epsilon(l) = \frac{1}{l}\sum_{t=1}^{l} \mathbb{E}_{\substack{y_{<t}\sim p(\cdot|\mathbf{x}) \\ y_t \sim p(\cdot|y_{<t},\mathbf{x})}} \log \frac{p(y_t \mid y_{<t}, \mathbf{x})}{q_\theta(y_t \mid y_{<t}, \mathbf{x})}.$$

ExAccErr measures the relative difference in cumulative teacher–student KL between student prefixes and teacher prefixes (requiring $\epsilon(l) > 0$). In practice, each branch keeps the problem/sample pairs for which both the teacher and student trajectories reach l, so the included set varies with l and the branch; this statistic describes errors associated with the prefix distribution rather than an isolated causal effect. Table 16 gives the per-length results of on/off-policy at step300: the teacher-prefix errors ϵ of the two are nearly identical, and the difference comes from the cumulative error on student prefixes, which grows with length.

Table 16: Exposure bias as a function of prefix length

l	on ε	off ε	on R/l	off R/l	on ExAccErr	off ExAccErr	95% CI of on − off
256	0.0961	0.0958	0.1018	0.1024	5.9%	6.9%	[−4.2, +2.2]
512	0.1040	0.1041	0.1084	0.1115	4.2%	7.1%	[−5.8, −0.1]
1K	0.1038	0.1045	0.1098	0.1151	5.8%	10.2%	[−7.0, −2.0]
2K	0.1004	0.1022	0.1073	0.1164	6.8%	13.9%	[−9.5, −4.9]
4K	0.1002	0.1020	0.1065	0.1194	6.3%	17.1%	[−14.0, −8.1]
8K	0.1072	0.1072	0.1127	0.1304	**5.2%**	**21.7%**	[−21.2, −11.7]

C Mathematical Analysis

C.1 Notation and Setup

The following analysis fixes the prefix and the frozen teacher, and works in logit space at temperature 1. Let the finite vocabulary be $\mathcal{V}$, $p_v > 0$, and $q = \mathrm{softmax}(z)$. Define

$$u_v = \log(q_v/p_v), \qquad \Delta_v = -u_v = \log(p_v/q_v). \tag{A.1}$$

For a fixed prefix distribution μ, the objective is

$$\mathcal{L}_\mu = \mathbb{E}_{s\sim\mu}[R(s) + \lambda_{\text{g}} G(s)] + \lambda_{\text{block}} L_{\text{block}}, \qquad R = D_{\text{KL}}(q\|p). \tag{A.2}$$

As in the main text, let $S = \mathrm{TopK}(p)$ and $M = \sum_{v\in S} p_v > 0$, and take $\rho > 1$. The gate term is

$$g_v = \mathbf{1}[v \in S][\Delta_v - \log\rho]_+, \qquad G = \frac{1}{2M}\sum_{v\in S} p_v g_v^2. \tag{A.3}$$

S and M are determined by the frozen teacher. All gradients below are for a single fixed prefix; averaging over prefixes follows the main text.

The gate and its gradient vanish whenever all teacher-head tokens satisfy $\Delta_v \le \log\rho$, including a neighborhood of $q = p$.

C.2 Recovery Gradient for Severely Underestimated Tokens

Let $d = R$ and $A = \sum_{v\in S} p_v g_v$. The exact per-logit gradients are

$$\frac{\partial R}{\partial z_j} = q_j(u_j - d) = q_j(-\Delta_j - d), \qquad \frac{\partial G}{\partial z_j} = \frac{q_j A - p_j g_j}{M}.$$

Proof. Differentiating $\sum_i q_i u_i$ using $\partial q_i / \partial z_j = q_i(\mathbf{1}[i=j] - q_j)$ gives the first expression. When the gate is active, $\partial \Delta_i / \partial z_j = q_j - \mathbf{1}[i=j]$; substituting into the derivative of the squared term and summing gives the second expression. □

For a fixed $j \in S$, hold $p_j > 0$ fixed and let $q_j \to 0$, with the other student probabilities remaining positive and bounded away from 0. Then $q_j \log q_j \to 0$, d is bounded, and $q_j A \to 0$; hence

$$\frac{\partial R}{\partial z_j} \longrightarrow 0, \qquad \frac{\partial G}{\partial z_j} \sim -\frac{p_j}{M} \log \frac{p_j}{\rho q_j}. \tag{A.4}$$

A negative-gradient update raises the logit of this token. At severely underestimated positions, the gate term provides a direct recovery signal that grows with the log-probability gap, whereas the reverse KL gradient for this coordinate vanishes. This is a local mechanistic analysis under a fixed prefix; actual recovery during training also depends on parameter sharing in the network, the optimizer, and the other loss terms.

In addition, the forward KL gradient is $\partial D_{\mathrm{KL}}(p\|q)/\partial z_j = q_j - p_j$. When $k=1$, $\nabla_z G = g_{j^*}(q - e_{j^*})$, i.e., the cross-entropy gradient toward the teacher top-1 token weighted by the above-threshold log gap.

C.3 Hidden State Alignment Term: Exact Angle–Norm Decomposition

Let $\tilde{h} = h / \sqrt{\max(\|h\|_2, \varepsilon)}$. When $\|h_q\|_2, \|h_0\|_2 \geq \varepsilon > 0$,

$$\|\tilde{h}_q - \tilde{h}_0\|_2^2 = \underbrace{(\sqrt{\|h_q\|_2} - \sqrt{\|h_0\|_2})^2}_{\text{scale mismatch}} + \underbrace{2\sqrt{\|h_q\|_2 \|h_0\|_2}(1 - \cos \angle(h_q, h_0))}_{\text{direction mismatch}}. \tag{A.5}$$

This identity follows from expanding the squared distance and shows that half-norm carries both scale and direction information; the relative size of the two terms depends on the specific representations. When a norm falls below the truncation threshold, the original definition with max is used.

The radial length of the normalization map, $r / \sqrt{\max(r, \varepsilon)}$, is strictly increasing, so a zero alignment loss is equivalent to equality of the representations at all positions and layers included in the sum. This constrains representations on the measured prefixes; uniqueness of parameters and generalization to unmeasured prefixes require further conditions.

D Supplementary Ablation Results

All results in this section use BroRL 1.5B K2V3. Distribution statistics use the 20 BF16 trajectories from Appendix A.5 (first 8,192 positions, 125,530 positions in total), and AIME24 for each branch is avg16.

D.1 KL Direction and Loss × Policy

The full-vocabulary off-policy comparison uses fixed teacher trajectories with pure forward or reverse KL, and sets the hidden-state alignment and gate-term coefficients to 0. Table 17 reports the results at step300: reverse KL is 0.83 percentage points higher on AIME24, and per-word statistics show that it better recovers from flattening (Table 18); however, it still has 30 extreme inversions with $r > 256$, versus 6 for forward KL. This is consistent with the analysis that the reverse KL gradient vanishes at severely underestimated positions, and it motivates introducing the gated squared forward KL to supply a recovery signal for extreme tokens (Appendix C.2).

Table 19 gives the full loss × policy comparison. Under the same on-policy setting, the OPQAD training objective is 2.50 percentage points higher than pure reverse KL on the average over the three AIME sets; with the objective fixed to OPQAD, on-policy is 4.30 percentage points higher than off-policy. Both comparisons point in the same direction, indicating that both the loss function and the trajectory source contribute to the final result.

Table 17: Accuracy and distribution recovery of the pure-KL off-policy branches at step300. Bold marks the best value among trained branches.

Branch	AIME24	Correct / Total	KL($q\|p$)	KL($p\|q$)	Top-1 agreement	$r > 256$ inversions
K2V3 (untrained)	14.38%	69 / 480	0.571	0.268	0.817	47
forward KL s300	30.42%	146 / 480	0.109	**0.089**	**0.890**	**6**
reverse KL s300	**31.25%**	150 / 480	**0.097**	0.106	0.889	30

Table 18: Per-word distribution after training to step300: off-position diffusion ratio and in-place probability (definitions in Appendix B.1). Statistics use the first 8,192 positions of each trajectory.

	Diffusion ratio				In-place probability			
Branch	wait	or	Symbols	Digits (aggregate / per-digit)	Connectives	Math verbs	Symbols	Digits
BF16	1	1	1	1 / 1	0.42	0.35	0.86	0.936
K2V3 (untrained)	3.64	1.47	1.42	0.95 / 1.31	0.31	0.18	0.74	0.857
forward KL s300	1.14	1.05	1.14	1.11 / 1.23	0.39	0.31	0.84	0.933
reverse KL s300	1.06	0.95	1.07	1.05 / 1.10	0.41	0.33	0.86	0.939

Table 19: Full loss × policy math comparison for BroRL 1.5B K2V3. All branches are at step300; each test set is avg16, in percent; Avg is computed from raw counts.

Loss	**Policy**	**AIME24**	**AIME25**	**AIME26**	**Avg**
Reverse KL	off-policy	31.25	24.79	19.58	25.21
Reverse KL	on-policy	36.04	24.38	21.67	27.36
OPQAD	off-policy	31.04	24.17	21.46	25.56
OPQAD	on-policy	38.54	26.46	24.58	29.86

D.2 Gated forward KL

Table 20 compares OPQAD with the branch that removes the gated forward KL (both on-policy, step300). The two have similar KL($q\|p$) and top-1 agreement; the difference is concentrated in extreme inversions: without the gate term, $r > 256$ events increase from 6 to 27, and high-confidence positions with $p_s < 0.01$ increase from 12 to 32. Table 21 gives trajectory-paired statistics, with ratio intervals obtained by trajectory resampling (Appendix A.5); Table 22 breaks down the $r > 256$ events by token category, showing that the reduction comes mainly from digits and symbols.

Table 20: Accuracy, extreme-inversion counts, and distribution statistics for the gated forward KL. $p_s < 0.01$ is the number of positions where the BF16 top-1 probability is ≥ 0.7 while the student probability is below 0.01.

Branch	AIME24	$r > 4$	$r > 16$	$r > 64$	$r > 256$	KL($q\|p$)	Top-1 agreement	$p_s < 0.01$
K2V3 (untrained)	14.38%	1,986	440	132	47	0.571	0.817	108
OPQAD	**38.54%**	204	**54**	**21**	**6**	0.106	0.889	**12**
OPQAD − gated forward KL	35.42%	258	95	42	27	0.100	0.889	32

D.3 Scale Trade-off and Early Sensitivity of Half-norm

Trade-off between scale and direction. Raw MSE ($\alpha = 0$) retains norm information but is more easily dominated by large-norm layers; unit-normalized alignment ($\alpha = 1$) constrains only direction and cannot constrain norm drift. Half-norm compresses the radial length of a vector from $\|h\|$ to

Table 21: Trajectory-paired statistics for the gated forward KL

Threshold	OPQAD	No gate term	Ratio 95% CI	Trajectories with fewer / more for OPQAD
>16	54	95	[0.41, 0.72]	19 / 1
>64	21	42	[0.33, 0.66]	15 / 1
>256	6	27	[0.05, 0.57]	10 / 2

Table 22: Category counts of extreme-inversion tokens before and after training

Model	Digits	Symbols	Math words	Connectives	wait	Other
K2V3 (untrained)	21	9	6	2	0	9
OPQAD	3	2	0	0	0	1
No gate term	12	6	3	0	0	6

$\sqrt{\|h\|}$, mitigating scale differences while retaining norm information (Appendix C.3). On 20 fixed BF16 prefixes for untrained K2V3 and BF16 (first 256 response positions each), layers 19–27 account for 85.13%, 64.69%, and 39.51% of the alignment loss under $\alpha = 0, 1/2, 1$, respectively, supporting this scale trade-off interpretation.

A 30-step sensitivity check (AIME24/AIME25 avg16, 32K context, temperature 0.7) gives AIME24 scores of 25.83/28.13/27.50% and AIME25 scores of 22.71/21.04/21.04% for $\alpha = 0, 1/2, 1$, respectively. Their mean accuracies are close (24.27/24.58/24.27%); the choice of half-norm is motivated by the scale–direction trade-off rather than an established final-performance advantage.

D.4 Robustness to Gate Parameters

To examine the local sensitivity of the gate hyperparameters, we compare different (k, ρ) settings starting from a common step180 checkpoint. All results are AIME24/AIME25 avg16 with 32K context and temperature 0.7; Avg is the equally weighted mean of AIME24 and AIME25.

Table 23: Local sensitivity to gate parameters. step180 is the common starting point, and the other branches are trained to step210; both benchmarks have 30 problems with 16 samples each. "No gate term" keeps reverse KL and hidden-state alignment and removes the gated forward KL.

Setting	step	AIME24	AIME25	Avg
Common step180 checkpoint	180	35.00%	24.58%	29.79%
$(k, \rho) = (1, 2)$	210	35.42%	24.79%	30.11%
$(k, \rho) = (4, 4)$	210	35.63%	26.46%	31.05%
$(k, \rho) = (16, 10)$	210	35.63%	25.00%	30.32%
No gate term	210	32.29%	25.83%	29.06%

Within this short training window, all three settings that keep the gate term maintain AIME24 at around 35%; $(4, 4)$ achieves the highest two-benchmark average, 31.05%. Removing the gate term gives an average of 29.06%, lower than all three gated settings. This experiment indicates that the gate hyperparameters are insensitive within the range from $(1, 2)$ to $(16, 10)$ and that the gated forward KL has a positive effect on math performance; given the short training, this result does not constitute a general conclusion.

D.5 Hidden-State Alignment: Representation Analysis

Following the analysis of OPRD (Yang et al., 2026), we compare, on the same BF16 prefixes, the cosine distance $1-\cos(h_s, h_t)$ between the student and the BF16 teacher at each layer and its variation with response position (Figure 10). For untrained K2V3, the cosine distance grows with depth, reaching 0.074 at layer 20, with averages of 0.012, 0.044, and 0.065 over layers 2–6, 7–18, and 19–26, respectively; the last-layer distance increases with response position from 0.038 (first 256 positions) to 0.059 (4K–8K), consistent with OPRD's observation that later tokens deviate more. After training with the output KL alone, the middle and deep layers converge substantially (0.034 for layers 7–18 and 0.035 for layers 19–26), while layers 2–6 increase slightly (0.013); hidden-state alignment further reduces the distance on all layer groups (0.012, 0.031, 0.031; all-layer average 0.0279→0.0248), and is lower than the branch without this alignment term on all 20 trajectories (20/20 for the last layer, the layer 7–18 average, and the all-layer average). The reduction is of similar magnitude across position ranges (about 8–15%), and the last-layer distance after training still increases with position.

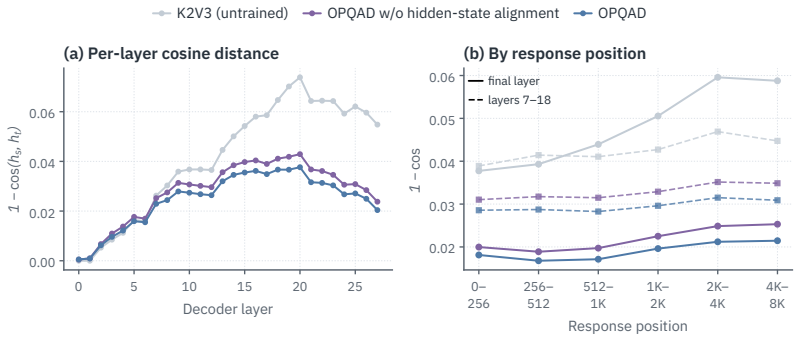

Figure 10: Representation analysis for BroRL 1.5B K2V3 (20 BF16 trajectories, 125,530 positions; trained branches are at step300). (a) Layer-wise cosine distance; (b) cosine distance of the last layer (solid) and the layer 7–18 average (dashed) as a function of response position.

D.6 Training Dynamics

Average top-k output-probability overlap and intersection mass do not reflect training effectiveness. During training, we seek a proxy metric that helps predict final accuracy. A natural idea is to track the top-k overlap between the quantized student and the full-precision teacher, together with the probability mass of their intersection. During training, we monitor the fraction of intersecting tokens within the top-512, as well as the total probability of these tokens under the teacher and student distributions. Although OPQAD attains higher overall accuracy than the control groups, its overlap ratio consistently ranks second, and the intersection mass also fails to predict the corresponding accuracy (Figure 11).

Figure 11 reports top-512 overlap and intersection mass on each branch's own on-policy batches. At step300, OPQAD has higher AIME24 accuracy than the branch without the gate, but lower overlap (0.84496 vs. 0.84802) and student intersection mass (0.99194 vs. 0.99418). These metrics do not rank the branches by accuracy in this experiment; prefixes also differ across branches.

E Implementation Details

Window semantics and training alignment. In OSCAR, the low-bit historical KV and the BF16 sink/recent windows jointly participate in attention. We maintain a separate window region per request, handle request replacement, block reuse, and differing K/V strides, and install the adaptation via a worker plugin. On the training side, rotation matrices are read per layer, and the rotation, magnitude clipping, and rounding rules are aligned; causal attention is split into the high-precision window and the quantized history, which are then merged via log-sum-exp. For padded inputs, windows are located by valid tokens, and checks cover both forward outputs and gradients.

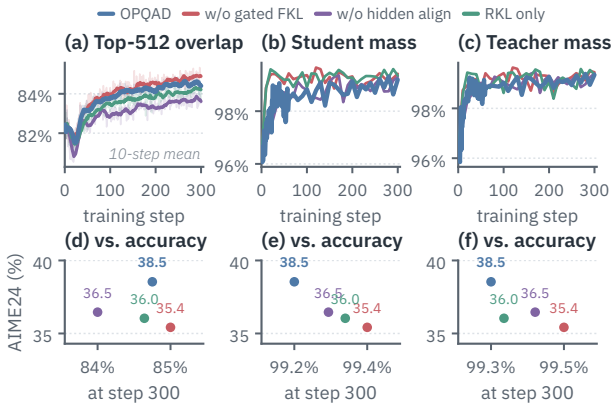

Figure 11: Top-512 overlap, intersection probability mass, and AIME24 score of each on-policy branch. (a)–(c) Evolution over training steps; (d)–(f) metrics versus AIME24 accuracy at step300.

Training-cost settings and accounting are given in Appendix A.4.

Extended Related Work

E.1 KV Cache Quantization

The storage requirement of the KV cache grows with context length, making low-bit representations an important optimization direction for long-sequence inference. Early methods design quantization granularity based on the distributional differences between Keys and Values: KIVI adopts an asymmetric scheme that quantizes Keys per channel and Values per token (Liu et al., 2024c); KVQuant combines pre-RoPE quantization, non-uniform codebooks, and outlier isolation (Hooper et al., 2024). Building on this, SKVQ protects recent information via channel reordering and a sliding window, ZipCache allocates precision according to token saliency, and OTT and KVSink track outlier tokens and attention sinks, respectively (Duanmu et al., 2024; He et al., 2024; Su et al., 2025a; Su & Yuan, 2025). These works share a common design principle: allocating limited representational precision according to the distributional structure of channels and tokens.

Another line of methods improves the quantization space through transformations. QuaRot applies rotations to weights, activations, and the KV cache, and RotateKV further combines outlier-aware rotation with attention-sink protection (Ashkboos et al., 2024; Su et al., 2025b). OScaR analyzes differences in token norms and combines rotation with per-token scaling to improve extremely low-bit cache representations (Su et al., 2026). QJL, PolarQuant, and TurboQuant explore online quantization under random projection, polar-coordinate transformation, and random rotation, respectively, to control inner-product or reconstruction error (Zandieh et al., 2024; Han et al., 2025; Zandieh et al., 2025); IsoQuant studies hardware-oriented block rotations, with its reported validation limited to quantization kernels on synthetic vectors (Ji, 2026).

Cache compression can be further improved through precision allocation, error compensation, and system co-design. MiniKV applies layer-wise differentiated cache compression, KVTuner searches per-layer Key/Value precision combinations, and AnTKV combines vector quantization with protection of highly sensitive tokens (Sharma et al., 2024; Li et al., 2025a;b). GEAR compensates quantization error with low-rank and sparse components, and AQUA-KV exploits cross-layer dependencies to predict cache representations and compress the residuals (Kang et al., 2024; Shutova et al., 2025). QServe and BitDecoding co-design low-bit representations with dequantization and attention kernels to achieve practical inference speedups (Lin et al., 2024; Du et al., 2025). Building on these advances in cache representation and execution, OPQAD further studies how quantization affects the inference output distribution and self-generated trajectories, and recovers reasoning ability through quantization-aware on-policy distillation.

E.2 Post-Training Quantization

Post-training quantization (PTQ) converts an existing model into a low-precision representation through calibration and error compensation, thereby reducing deployment cost (Nagel et al., 2021). Its core problem is to preserve the computational behavior of the floating-point model using limited calibration data. AdaRound optimizes weight rounding decisions, BRECQ handles cross-layer error via block-wise reconstruction, ZeroQuant combines fine-grained quantization with layer-wise knowledge distillation, and GPTQ uses approximate second-order information for efficient layer-wise weight quantization (Nagel et al., 2020; Li et al., 2021; Yao et al., 2022; Frantar et al., 2022).

Outliers and differences in dynamic range across channels have driven the design of quantization scales and representations. LLM.int8() uses a mixed-precision decomposition for outlier dimensions, Outlier Suppression combines scale migration with per-token clipping, SmoothQuant shifts part of the activation quantization difficulty to the weights through equivalent scaling, and RPTQ reorders channels and groups them by value range (Dettmers et al., 2022; Wei et al., 2022; Xiao et al., 2022; Yuan et al., 2023). To address differences in weight importance, AWQ protects salient channels based on activation statistics, while SpQR and SqueezeLLM adopt sparse high-precision representations and dense–sparse decomposition, respectively, to preserve error-sensitive components (Lin et al., 2023; Dettmers et al., 2023b; Kim et al., 2023b).

Another class of methods improves low-bit representations through transformations and structured encoding. QuIP uses incoherence processing to improve the quantization conditioning of weights and second-order information, and QuIP# further introduces Hadamard transforms and lattice codebooks; AQLM models weights with additive codebooks (Chee et al., 2023; Tseng et al., 2024; Egiazarian et al., 2024). QuaRot suppresses outliers via rotation, SpinQuant learns rotation matrices, OmniQuant jointly optimizes clipping and equivalent transformations, and FlatQuant learns affine transformations to improve the quantizability of weights and activations (Ashkboos et al., 2024; Liu et al., 2024b; Shao et al., 2023; Sun et al., 2024). In addition, EoRA constructs low-rank compensation in the eigenspace of input activations to recover information lost to model compression (Liu et al., 2024a).

These studies improve quantization accuracy mainly from the perspectives of numerical representation, local reconstruction, and error compensation. OPQAD focuses on autoregressive reasoning behavior under KV cache quantization: it aligns output distributions and hidden representations in contexts generated by the student itself, and designs a recovery objective targeting extreme probability inversions of high-confidence tokens. This direction treats the post-quantization trajectory distribution and the probability preservation of critical tokens as optimization targets, complementing improvements to the underlying quantized representations.

E.3 Quantization-Aware Training

Quantization-aware training (QAT) simulates low-precision arithmetic in the training forward pass so that model parameters adapt to the quantization error at deployment. Early work jointly designed quantization and the training pipeline to support integer-only inference(Jacob et al., 2017); PACT and LSQ learn activation clipping ranges and quantization step sizes, respectively(Choi et al., 2018; Esser et al., 2019). Another line incorporates low-precision computation into model training itself: DoReFa-Net studies low-bit weights, activations, and gradients jointly, and BitNet b1.58 directly trains language models with ternary weights(Zhou et al., 2016; Ma et al., 2024). These methods reflect the co-design of training and quantization, and their training objectives differ in emphasis from post-quantization accuracy recovery of existing models.

QAD guides a low-precision student with soft targets from a full-precision teacher(Hinton et al., 2015). Apprentice, quantized distillation, and QKD study the combination of distillation with low-precision training and its training-stage scheduling in vision models(Mishra & Marr, 2017; Polino et al., 2018; Kim et al., 2019); TernaryBERT and BinaryBERT apply distillation or initialization designs to ultra-low-bit language encoders(Zhang et al., 2020; Bai et al., 2020). For generative language models, LLM-QAT distills on model-generated data and jointly considers weight, activation, and KV cache quantization(Liu et al., 2023b); TSLD adjusts distillation supervision per token, and BitDistiller combines asymmetric quantization with a confidence-aware KL objective(Kim et al., 2023a; Du et al., 2024). EfficientQAT reduces training cost through block-wise full-parameter training followed by end-to-end fine-tuning of quantization parameters(Chen et al., 2024).

Quantization has also been used to reduce the cost of post-training. QLoRA freezes a quantized base model and trains low-rank adapters, and QeRL combines NVFP4 with low-rank adaptation for reinforcement learning and studies the effect of quantization noise on exploration(Dettmers et al., 2023a; Huang et al., 2025). For models that have already undergone multi-stage post-training, NVFP4 QAD directly matches the full-precision teacher distribution and recovers quantized accuracy across a variety of language and vision-language models(Xin et al., 2026). This result shows that the teacher distribution can provide direct supervision for preserving post-trained capabilities.

The source of training prefixes also affects distillation. MiniLLM uses reverse KL with on-policy optimization, and GKD queries teacher feedback on student-generated sequences, adapting to the contexts the student visits at inference time(Gu et al., 2023; Agarwal et al., 2023). OPQAD applies this idea to long-chain reasoning after KV cache quantization and combines it with block-level representation alignment and gated probability recovery, focusing on distribution shift in the quantized student's own trajectories and extreme probability inversions of critical tokens.