TRAIN WHERE THE QUANTIZED MODEL GOES: ON-POLICY DISTILLATION FOR LOW-BIT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantization-aware distillation (QAD) restores much of the short-form question-answering performance lost to sub-3-bit quantization, yet leaves mathematical and code reasoning substantially impaired. Long generations often degenerate into repetitive loops, exhausting the decoding budget without completing a solution. We trace this gap to quantization-amplified exposure bias: QAD trains on fixed corpus prefixes, while quantization-induced deviations compound along the model's own autoregressive trajectories. To address this mismatch, we introduce an on-policy distillation (OPD) stage that places teacher supervision where the quantized model actually goes. Starting from a QAD checkpoint, the student generates through the quantized forward path used at deployment and receives feedback from a frozen full-precision teacher on its own prefixes, combining dense token-level guidance with task-verifier rewards. Across four models at 2.79 and 1.88 effective bits, OPD raises average BF16 performance retention from 35% to 70% on MATH-500 and from 66% to 91% on HumanEval while preserving short-form performance, with reasoning gains substantially exceeding those of continued teacher-forced QAD in matched-budget comparisons. By coupling QAD's stable low-bit initialization with OPD's on-policy reasoning recovery, our framework provides a comprehensive sub-3-bit solution that preserves broad capabilities while restoring long-form reasoning.

1 INTRODUCTION

Large language models (DeepSeek-AI, 2025) impose substantial memory and bandwidth demands, limiting inference efficiency and deployment on constrained hardware. Quantization (Dettmers et al., 2022) reduces these costs by representing model weights at lower precision while retaining the original architecture. Pushing precision below three bits offers substantial compression, making extreme low-bit quantization an attractive route to deploying capable models with fewer resources.

At four bits and above, post-training quantization (PTQ) often preserves performance with only a small calibration set. Below three bits, however, quantization errors become harder to compensate for, and direct PTQ can severely degrade or even collapse model performance. This makes quantization-aware training (QAT) essential for adapting the model to low-bit computation. Conventional QAT typically relies on pretraining-style data and objectives, while modern models acquire much of their instruction-following and reasoning capabilities during post-training. Quantization-aware distillation (QAD) provides a practical route to recovering these capabilities: a full-precision teacher supervises its quantized counterpart on a manageable distillation corpus.

Our study reveals a sharp imbalance in what QAD recovers. Across four models and two effective bit widths, QAD retains an average of 88% of BF16 performance on short-form question answering, but only 35% on MATH-500. Long generations expose a striking failure mode: the quantized model enters repetitive loops and exhausts its decoding budget without completing a solution. The recovery deficit becomes more pronounced on tasks requiring longer generations, making the ability to sustain and complete extended reasoning a central target of low-bit recovery.

We trace this gap to quantization-amplified exposure bias. QAD trains the quantized student to match its teacher on prefixes drawn from a fixed corpus. During deployment, the student instead conditions on its own previous predictions. Quantization perturbs the next-token distribution at every step, and each departure changes the context for the predictions that follow. Even a locally plausible

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

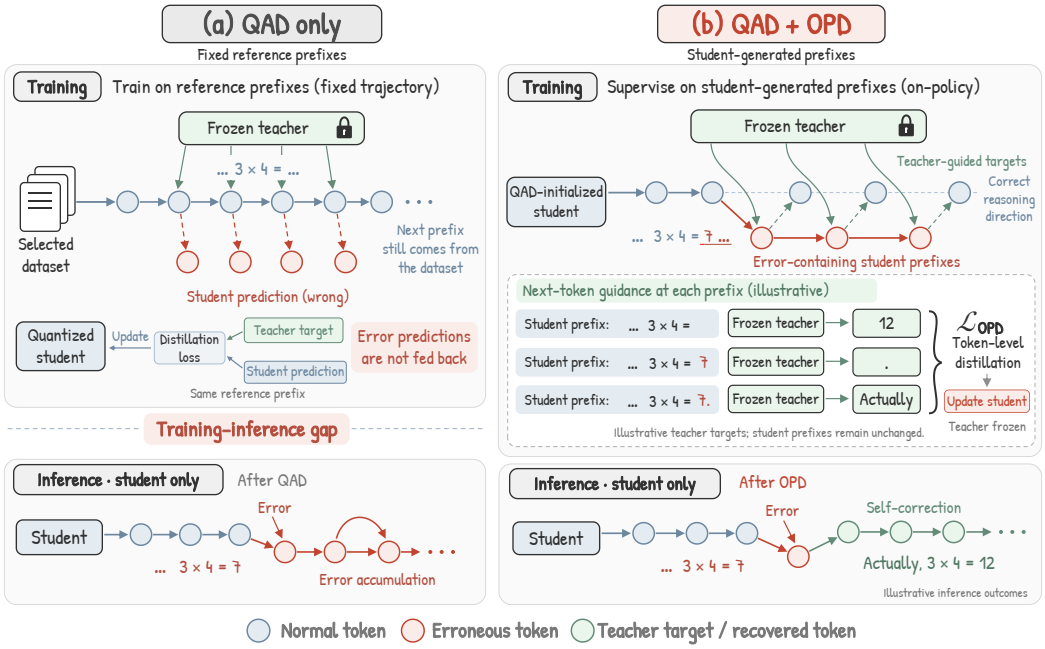


Figure 1: QAD provides a stable low-bit initialization through teacher supervision on fixed corpus prefixes, but leaves a training–inference gap on reasoning: the model is never trained on the trajectories it actually generates at deployment. OPD closes this gap by sampling through the deployment quantized path and combining token-level teacher guidance on student prefixes with verifier rewards.

continuation can move the model away from the trajectories covered during training. Over a long generation, these deviations accumulate, exposing the student to states on which it has received little supervision. The missing guidance therefore lies along the reasoning trajectories the quantized model actually generates.

We address this gap with a two-stage recovery framework that follows QAD initialization with on-policy distillation (OPD). QAD first restores broad capabilities and provides a viable low-bit policy. OPD then places teacher supervision on that policy's own trajectories. The student samples completions through the quantized forward path used at deployment, and a frozen BF16 teacher provides token-level feedback on the prefixes it produces. Generating through the low-bit path makes quantization-induced changes in the student's behavior part of the training distribution itself. Task verifiers supply complementary rewards for final-answer correctness in mathematics and successful test execution in code. By shifting teacher supervision from fixed corpus prefixes to the quantized student's own trajectories, OPD targets the accumulated deviations that disrupt long-form reasoning.

We evaluate this framework on Qwen3-0.6B, Qwen3-1.7B, Qwen3-4B, and Falcon3-1B at 2.79 and 1.88 effective bits. Building on QAD initialization, OPD raises average BF16 retention from 35% to 70% on MATH-500 and from 66% to 91% on HumanEval while preserving short-form performance, with reasoning gains substantially exceeding those of continued teacher-forced QAD under matched corpora and optimizer-step budgets. OPD accounts for a larger share of recovery at lower bit widths, making on-policy recovery increasingly valuable under aggressive quantization. With only a few hundred additional optimizer steps, the combined pipeline couples QAD's stable low-bit initialization with OPD's reasoning recovery to preserve broad capabilities and restore long-form reasoning below three bits.

2 Related Work

2.1 Extreme Low-Bit Quantization and Reasoning Recovery

Post-training quantization (PTQ) compresses pretrained models using a small calibration set (Frantar et al., 2023; Shao et al., 2024a). Techniques such as activation-aware scaling

(Xiao et al., 2023; Lin et al., 2024), rotations (Ashkboos et al., 2024), mixed precision, and vector quantization (Chee et al., 2023; Egiazarian et al., 2024) reduce quantization error and preserve performance at moderate precision. More aggressive compression motivates quantization-aware training (QAT), which updates model parameters through a simulated low-bit forward pass so that the model adapts to quantization during training (Liu et al., 2024; Ma et al., 2024). Quantization-aware distillation (QAD) augments this process with a full-precision teacher, transferring output distributions or intermediate representations to recover the quantized student's capabilities (Du et al., 2024). In teacher-forced QAD, this supervision is evaluated on prefixes drawn from a fixed corpus.

Recent studies examine the particular challenges of reasoning under extreme quantization. Lee et al. (2026) identify how quantization errors concentrated on low-entropy tokens propagate along a chain of thought. Lv et al. (2026) show that reinforcement learning applied to a collapsed low-bit model requires a distillation cold start to establish a viable policy.

2.2 Exposure bias and on-policy distillation

Exposure bias arises when a model is trained on reference prefixes but conditions on its own predictions during autoregressive inference. Once generation departs from the reference trajectory, subsequent predictions depend on contexts that training may not have covered. Work in imitation learning (Ross et al., 2011) and sequence modeling addresses this mismatch by exposing the learner to its own outputs, including through scheduled sampling (Bengio et al., 2015) and sequence-level training (Ranzato et al., 2016).

On-policy distillation brings teacher supervision directly onto these student-generated trajectories (Gu et al., 2024; Agarwal et al., 2024). The student samples completions, and the teacher supplies targets on the prefixes the student actually visits. Relative to teacher-forced distillation (Kim & Rush, 2016), OPD changes the distribution of prefixes receiving supervision (Ko et al., 2024). Relative to reward-only training, it provides token-level teacher feedback throughout each completion. This combination supports reasoning recovery by aligning dense supervision with the student's own generation behavior. We adapt this idea, established in full-precision post-training (Lu & Thinking Machines Lab, 2025), to extreme low-bit quantization.

3 What teacher-forced recovery leaves behind

To understand what QAD recovers and what it leaves behind, we quantize Qwen3-0.6B, Qwen3-1.7B, and Qwen3-4B using round-to-nearest (RTN) at effective weight precisions of 2.79 and 1.88 bits, with 8-bit activations and 4-bit embedding and output-head weights. We then recover the resulting six quantized checkpoints using the EdgeRazor QAD recipe (Zhang et al., 2026). Evaluation covers mathematical reasoning on GSM8K, MATH-500, and AMC23, code generation on HumanEval and MBPP, and 9 short-form question-answering benchmarks (QA9), with retention defined as each checkpoint's performance as a percentage of the corresponding BF16 reference.

RTN leaves these models with zero accuracy on all five generative benchmarks. QAD restores useful behavior, but the recovery is strikingly uneven across short answers and extended reasoning.

3.1 Short answers recover, long derivations do not

The performance recovered by QAD separates into three distinct bands in Figure 2. Averaged over the six settings, the QAD checkpoints retain 86% of BF16 performance on QA9. Retention falls to 56% on GSM8K and 59% on MBPP, then drops further to 27% on MATH-500 and 16% on AMC23.

What distinguishes these bands is how much the model must generate for itself. QA9 scores answer options by likelihood and requires no autoregressive generation. The BF16 reference generates roughly 130 tokens on GSM8K and MBPP, compared with about 410 on MATH-500 and 3,200 on AMC23. GSM8K and MBPP require different outputs, an arithmetic derivation and a working program, yet their comparable generation lengths are accompanied by similar retention.

Figure 2(b) shows the same decline from short to medium to long responses in every model–bit-width setting. The recovery gap therefore grows where the student must sustain a longer chain of its own predictions.

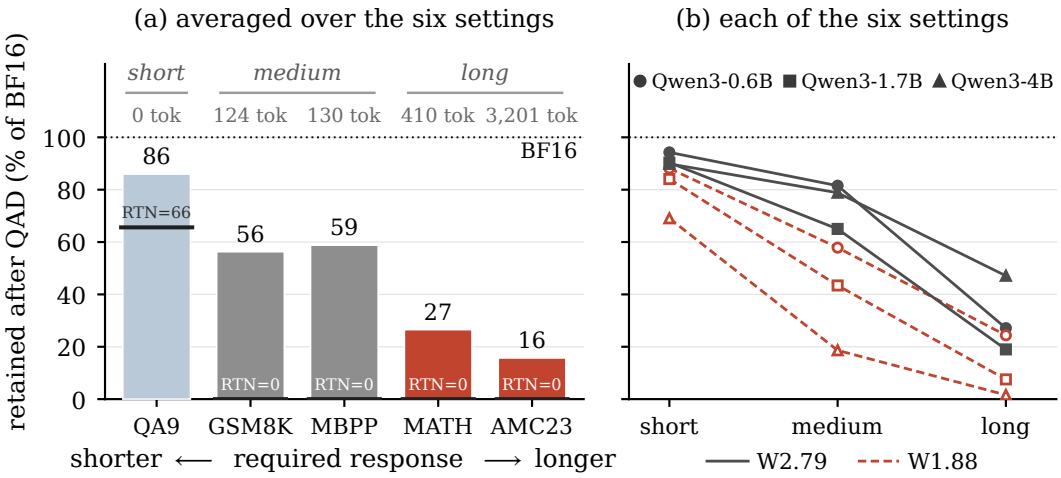


Figure 2: QAD retention relative to BF16 across three Qwen3 scales and two bit widths. (a) Benchmarks ordered by mean BF16 generation length (shown above the bars); horizontal marks indicate RTN retention. QA9 uses answer-option likelihoods without generation; MATH denotes MATH-500. (b) Mean retention decreases from short- to medium- to long-form tasks in every setting.

3.2 Quantization amplifies exposure bias

This dependence on the student's own predictions exposes a mismatch in teacher-forced QAD. Its teacher supervision is evaluated on fixed prefixes taken from the training corpus, while deployment visits prefixes the quantized student generates for itself. Quantization amplifies this mismatch by changing the trajectories the student follows.

At each generation step, quantization perturbs the next-token distribution. Once this changes the selected token, every subsequent prediction conditions on a different prefix. The continuation need not be incorrect: even a plausible alternative can carry the student beyond the trajectories covered by teacher forcing. Further predictions then combine quantization error with the effects of that altered context, allowing deviations to accumulate along the sequence. Lower precision increases the disruption, and longer generations provide more opportunities for it to compound.

This mechanism predicts a breakdown that unfolds during generation: as deviations accumulate, responses become harder to terminate and increasingly prone to repetition.

3.3 QAD loses the trajectory, not the arithmetic

We track these two behaviors in the BF16 and W2.79 QAD checkpoints of Qwen3-0.6B, using greedy generation on 200 problems each from GSM8K and MATH-500. Figure 3(a, b) shows the fraction of responses still generating after t tokens, while Figure 3(c) shows the fraction whose closing words repeat an 8-gram.

The clearest breakdown appears on MATH-500: 95% of QAD generations exhaust the decoding budget, compared with 27% for BF16. Moreover, 70% of QAD outputs end in repeated 8-grams, against 12% for BF16. The model continues producing tokens, but its reasoning becomes trapped in repetitions that prevent it from reaching a conclusion.

The curves reveal how this gap develops. BF16 and QAD have similar termination patterns over the first few dozen tokens, but their curves separate as generation continues. Shorter GSM8K responses follow the same pattern with smaller gaps: 32% of QAD responses exhaust the budget, versus 5% for BF16, while repetition rates are 30% and 2%. The longer reasoning required by MATH-500 exposes a much greater breakdown in the ability to sustain and complete a solution.

These failed generations make the supervision gap concrete. QAD teaches the student how to continue demonstrated trajectories, but the model must complete its reasoning from prefixes produced

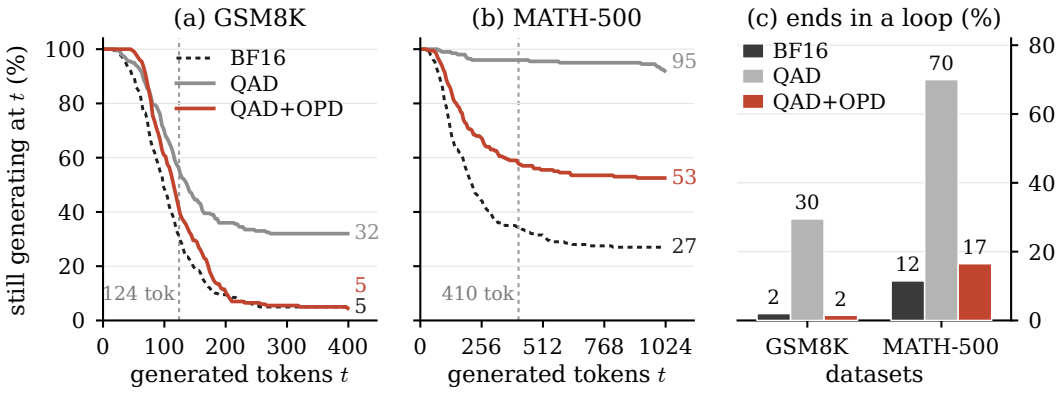


Figure 3: OPD restores trajectory control after QAD. Qwen3-0.6B at W2.79 is evaluated with greedy decoding on 200 problems per benchmark. (a, b) Fraction of responses still generating after t tokens on GSM8K and MATH-500. Vertical dashed lines mark mean BF16 response lengths; endpoint labels report budget exhaustion rates. (c) Fraction of responses ending in repeated 8-grams.

by its own perturbed predictions. Recovering this missing ability calls for extending teacher supervision to the reasoning trajectories the quantized model actually generates.

4 On-policy distillation for low-bit reasoning

Long-form reasoning breaks down along the quantized model's own trajectories; recovering it calls for teacher guidance along those same paths. We therefore combine QAD initialization with on-policy distillation. QAD first restores broad capabilities and establishes a viable low-bit policy, providing the foundation for rollout-based recovery: a round-to-nearest model at these widths has nothing worth sampling and earns no reward to bootstrap from. OPD then trains the student on trajectories sampled through its deployment quantized forward path, with a frozen BF16 teacher guiding its continuations and task verifiers rewarding successful solutions. The two stages divide the work of recovery: QAD restores policy viability; OPD restores trajectory control.

4.1 From teacher forcing to student forcing

Let π_{T} denote a frozen BF16 teacher and π_θ the student policy evaluated through the quantized forward path used at deployment. Teacher-forced QAD learns from a corpus $\mathcal{D}$ of prompt–response pairs by minimizing

$$\mathcal{L}_{\mathrm{TF}}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\sum_{t=1}^{|y|} \mathrm{KL}(\pi_{\mathrm{T}}(\cdot \mid x, y_{<t}) \,\|\, \pi_\theta(\cdot \mid x, y_{<t}))\right]. \tag{1}$$

Here, each prefix $y_{<t}$ comes from a fixed reference response. Student forcing instead draws prompts x from a set $\mathcal{X}$ and samples a completion $\hat{y} \sim \pi_\theta(\cdot \mid x)$, placing teacher supervision on the resulting student-generated prefixes:

$$\mathcal{L}_{\mathrm{SF}}(\theta) = \mathbb{E}_{x\sim\mathcal{X}}\, \mathbb{E}_{\hat{y}\sim\pi_\theta(\cdot|x)}\left[\sum_{t=1}^{|\hat{y}|} \mathrm{KL}(\pi_\theta(\cdot \mid x, \hat{y}_{<t}) \,\|\, \pi_{\mathrm{T}}(\cdot \mid x, \hat{y}_{<t}))\right]. \tag{2}$$

Sampling through the quantized forward path makes the effects of low-bit computation part of the training distribution itself. The teacher therefore guides the student on continuations shaped by quantization, including departures from the reference trajectories. We use reverse KL because it can be estimated from student samples and penalizes continuations that the student favors but the teacher assigns low probability to.

4.2 LEARNING FROM QUANTIZED ROLLOUTS

To translate this supervision into successful reasoning, we combine on-policy teacher feedback with task-verifier rewards. For each prompt $x \in \mathcal{X}$, the student generates G completions through the quantized forward path. The frozen teacher scores those same token sequences, and a verifier assigns each completion a reward $r(x, \hat{y})$: final-answer correctness for mathematics and test execution for code. We optimize

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = -\,\mathbb{E}\left[\sum_t \hat{A}(x,\hat{y})\,\log \pi_\theta(\hat{y}_t \mid x, \hat{y}_{<t})\right] \;+\; \beta\,\mathbb{E}\left[\sum_t \log \frac{\pi_\theta(\hat{y}_t \mid x, \hat{y}_{<t})}{\pi_{\mathrm{T}}(\hat{y}_t \mid x, \hat{y}_{<t})}\right], \quad (3)$$

where $\hat{A}$ is the advantage of $r(x, \hat{y})$ relative to the other completions for the same prompt (Shao et al., 2024b), both expectations are over $x \sim \mathcal{X}$ and $\hat{y} \sim \pi_\theta(\cdot \mid x)$, and β controls the strength of teacher guidance. We use $\beta = 1$ in every run reported here, so the balance between the two terms is never tuned per setting. The two signals guide recovery at complementary scales: the sampled reverse-KL term supplies token-level feedback on the prefixes the student actually generates, while the group-relative advantage promotes completions that reach verified solutions.

4.3 OPD RESTORES TRAJECTORY CONTROL

Training on these trajectories directly improves the generation behaviors that break down after QAD. For the Qwen3-0.6B W2.79 checkpoint examined above, Figure 3(a–c) shows the clearest recovery on MATH-500. OPD reduces the loop rate from 70% to 17%, close to the BF16 rate of 12%, and cuts budget exhaustion from 95% to 53%. Accuracy rises alongside this behavioral recovery, from 10.4% to 24.2%, approaching the BF16 reference of 27.2%.

On GSM8K, the loop rate falls from 30% to 2% and budget exhaustion from 32% to 5%, matching the BF16 reference on both measures. The larger improvement on MATH-500 mirrors the greater disruption on longer generations: OPD recovers more ground where QAD's trajectory failures are most severe. By extending teacher guidance onto the student's own prefixes, OPD helps the low-bit model sustain a derivation and bring it to a conclusion.

5 EXPERIMENTS

We evaluate reasoning recovery across four models and two effective bit widths, examining the preservation of broad capabilities, the contribution of OPD at lower precision, OPD's recovery efficiency relative to QAD, and how the recovered models compare with existing methods.

5.1 EXPERIMENTAL SETUP

Models and quantization. We evaluate Qwen3-0.6B, 1.7B, and 4B (Qwen Team, 2025), together with Falcon3-1B-Instruct (Falcon-LLM Team, 2024), spanning model scales and architectures. Following Zhang et al. (2026), we use 2.79 and 1.88 effective bits per weight, allocating 50% and 12.5% of weight groups to 4 bits, respectively, and the remainder to 1.58 bits. Embedding and output-head weights use 4 bits and activations 8 bits. Appendix A details how we reproduced the QAD stage.

Implementation. Every OPD run starts from the reported QAD checkpoint, which keeps the QAD and OPD numbers measured from a common starting point. Recovery then proceeds in two phases, a mathematics phase followed by a code phase that resumes from the selected mathematics

Table 1: OPD phases. The code phase resumes from the selected mathematics checkpoint. Each optimizer step draws 8 prompts and G completions. GSM8K, MATH L1–3, and MBPP are taken from the training splits and do not overlap the evaluation sets.

Phase	Training corpus	Steps	LR	G	Rollouts
Math	GSM8K, MATH L1–3, DAPO-Math	80–140	3×10^{-6}	4	32
Code	MBPP, KodCode	80–150	3×10^{-6}	8	64

Table 2: Four models at two effective bit widths, from the BF16 reference down to RTN and back up through QAD and OPD; the shaded row is the OPD result for each line.

			GSM8K	MATH-500	AMC23	MBPP	HumanEval	QA9 (avg)
Qwen3-0.6B	BF16		41.62	27.20	7.81	40.0	36.6	46.08
	W2.79	RTN	0.00	0.00	0.00	0.0	0.0	34.87
		QAD	32.75	10.40	1.25	33.7	29.9	43.43
		+ OPD	**43.14**	**24.20**	**4.69**	**37.7**	**35.4**	**43.64**
	W1.88	RTN	0.00	0.00	0.00	0.0	0.0	34.74
		QAD	23.05	2.40	3.12	24.1	22.0	40.62
		+ OPD	**37.65**	**15.00**	3.12	**35.7**	**35.4**	**41.83**
Qwen3-1.7B	BF16		68.76	54.40	31.72	54.0	67.1	54.54
	W2.79	RTN	0.00	0.00	0.00	0.0	0.0	35.49
		QAD	44.50	19.60	0.62	35.3	41.5	49.28
		+ OPD	**54.28**	**45.60**	**19.38**	**52.0**	**59.1**	**51.81**
	W1.88	RTN	0.00	0.00	0.00	0.0	0.0	35.22
		QAD	26.46	8.20	0.00	26.1	24.4	45.85
		+ OPD	**44.05**	**24.20**	**3.12**	**41.3**	**45.1**	**47.90**
Qwen3-4B	BF16		86.35	68.80	55.00	67.6	82.3	61.63
	W2.79	RTN	0.00	0.00	0.00	0.0	0.0	34.80
		QAD	69.45	39.80	20.00	52.2	62.2	55.38
		+ OPD	**77.86**	**54.40**	**39.38**	**57.8**	**64.0**	**57.37**
	W1.88	RTN	0.00	0.00	0.00	0.0	0.0	34.78
		QAD	17.36	2.20	0.00	11.6	11.0	42.56
		+ OPD	**64.59**	**36.00**	**16.25**	**48.7**	**60.4**	**52.83**
Falcon3-1B -Instruct	BF16		41.55	23.00	6.25	26.8	17.7	54.17
	W2.79	RTN	0.00	0.40	0.00	0.0	0.0	37.74
		QAD	36.32	16.00	5.00	**24.6**	20.7	52.89
		+ OPD	**38.89**	**19.20**	2.50	24.1	20.7	**53.03**
	W1.88	RTN	0.00	0.60	0.00	0.0	0.0	35.36
		QAD	31.61	12.40	2.50	21.0	14.6	49.53
		+ OPD	**35.03**	**16.20**	**6.25**	**24.3**	**20.1**	**49.58**

checkpoint; Table 1 lists the corpora, step budgets, and rollout settings, including the DAPO-Math (Yu et al., 2025) and KodCode (Xu et al., 2025) training sets. We train with verl (Sheng et al., 2025) and generate with vLLM (Kwon et al., 2023), using non-thinking mode at temperature 1. The two precisions stay separate within a step: BF16 master weights receive the updates, while rollouts pass through the deployment quantized forward path. Each student is supervised by its own BF16 counterpart as the frozen teacher, with a shared tokenizer; Qwen3-0.6B is the one exception and takes Qwen3-1.7B. Appendix F examines teacher choice. The learning rate is 3×10^{-6}, except for the Falcon3 code phase, which uses 1×10^{-6}. We checkpoint every 20 steps during OPD and select by validation on the current phase's metric. Appendix E plots the phase-1 curves as an example.

Evaluation. GSM8K (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2024) follow the lm-eval protocol (Gao et al., 2024). AMC23 uses avg@16 over 40 problems. Code generation is measured by pass@1 on 448 MBPP problems (Austin et al., 2021) and 164 HumanEval problems (Chen et al., 2021). QA9 averages ARC-Easy, ARC-Challenge, HellaSwag, SocialIQA, OpenBookQA, PIQA, WinoGrande, TruthfulQA and MMLU, with per-task scores in Appendix B.

5.2 Recovering reasoning while preserving broad capabilities

For each configuration, Table 2 compares four arms: the BF16 reference, RTN, QAD, and QAD+OPD. Building on QAD initialization, OPD substantially restores mathematical and code reasoning. OPD doubles average BF16 retention on MATH-500 from 35.3% to 69.7% and raises GSM8K retention from 62.7% to 85.0%. Across the six Qwen3 configurations, GSM8K accuracy improves by 8.41–47.23 percentage points. Code generation improves alongside mathematical rea-

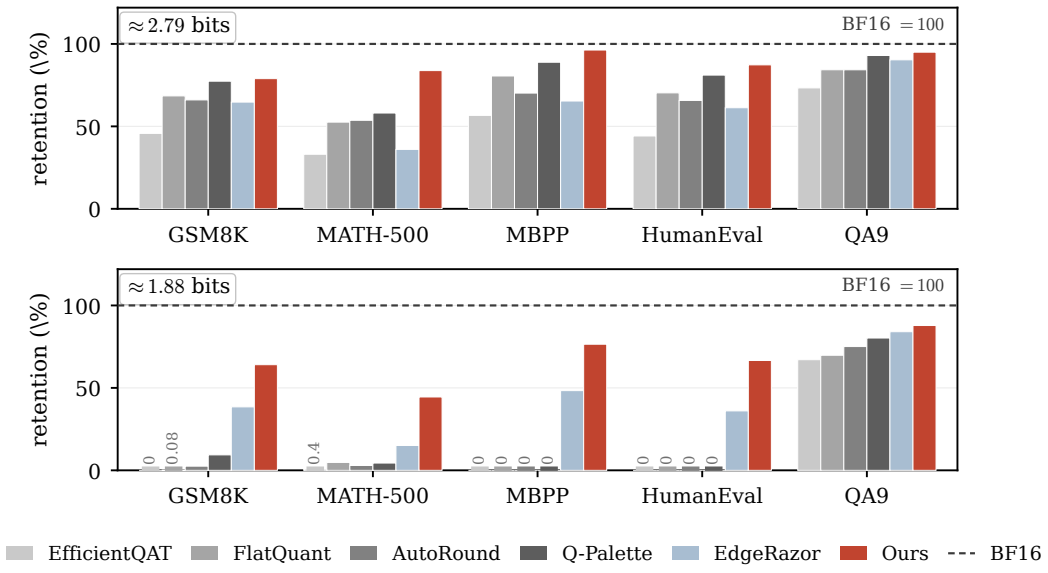


Figure 4: Qwen3-1.7B as retention of the BF16 reference. Most baselines are aligned to our deployment setting of W2.79/W1.88 with INT8 activations, and to INT4 embedding and lm_head; Appendix C gives the exceptions and the builds. Methods without mixed-precision support fall back to the next whole width, 3 and 2 bits in the two bands, which leaves them above our budget.

soning. At 2.79 bits, Qwen3-1.7B raises MBPP from 35.3% to 52.0% and HumanEval from 41.5% to 59.1%, approaching the corresponding BF16 scores of 54.0% and 67.1%.

OPD successfully preserves the broad capabilities restored by QAD, with mean QA9 retention increasing from 88.1% to 92.1%, so the combined pipeline restores reasoning while maintaining short-form performance. OPD's contribution grows as precision falls. Across all four models, OPD accounts for a larger share of total GSM8K recovery from RTN at 1.88 bits than at 2.79 bits. For Qwen3-4B, this share rises from 11% to 73%: at 1.88 bits, OPD raises GSM8K accuracy from 17.36% to 64.59% and MBPP from 11.6% to 48.7%.

5.3 Recovery efficiency

Table 3: Training cost of QAD initialization and OPD recovery.

Model	Stage	Steps	Wall-clock	GPU-hours
Qwen3-4B (W1.88)	QAD	6,400	102 h	820
	OPD	260	14 h	57
Falcon3-1B (W2.79)	QAD	6,400	32 h	253
	OPD	245	2.8 h	11

OPD converts training steps into reasoning gains far more efficiently than continued teacher-forced QAD, yielding more GSM8K points per thousand optimizer steps from the same starting checkpoint by up to 42× where continued QAD still gains at all. This translates into substantially lower cost. Table 3 reports the two stages from the training logs: the OPD stage uses 57 GPU-hours against 820 for QAD on Qwen3-4B at 1.88 bits, and 11 against 253 on Falcon3-1B at 2.79 bits, completing recovery with approximately 14–23× fewer GPU-hours than QAD initialization.

5.4 Comparison with quantization baselines

The comparison in Figure 4 places our pipeline on Qwen3-1.7B against baselines spanning all three regimes of low-bit recovery: PTQ, QAT, and QAD. Three PTQ methods cover signed-gradient rounding (Cheng et al., 2024), learned affine transformations (Sun et al., 2025), and fractional-bit codebook quantizers (Lee & Song, 2025), while a block-wise method (Chen et al., 2025) represents QAT, and the QAD arm is EdgeRazor, which our own recovery also starts from.

At 2.79 bits the ordering separates short-form from long-form ability. The strongest baseline, Q-Palette, retains 93% of BF16 on QA9 and 89% on MBPP, confirming that a well-designed quantizer

Table 4: Controlled comparison of OPD and continued QAD under a matched budget. Both arms share starting checkpoints, corpora, phase schedules, step counts, learning rates, and samples per step. OPD leads in all 16 comparisons. Baselines are re-evaluated separately for this control.

Model	Width	Arm	GSM8K	MATH-500	MBPP	HumanEval
Qwen3-0.6B	W2.79	QAD start	33.21	10.00	33.7	29.9
		+ QAD, matched	35.10	15.60	31.7	30.5
		+ OPD	**43.14**	**24.20**	**37.7**	**35.4**
	W1.88	QAD start	22.52	2.60	24.1	22.0
		+ QAD, matched	27.41	8.20	24.3	18.3
		+ OPD	**37.65**	**15.00**	**35.7**	**35.4**
Qwen3-1.7B	W2.79	QAD start	44.58	20.60	35.3	41.5
		+ QAD, matched	44.28	23.20	39.7	42.7
		+ OPD	**54.28**	**45.60**	**52.0**	**59.1**
	W1.88	QAD start	26.54	8.40	26.1	24.4
		+ QAD, matched	31.01	15.80	31.7	32.3
		+ OPD	**44.05**	**24.20**	**41.3**	**45.1**

preserves knowledge and short code completions. Its retention falls to 58% on MATH-500, while OPD reaches 84%. The pattern repeats across the other baselines: on MATH-500 they retain 33–58%, and none exceeds 59%, whereas OPD retains 84% while running at 2.79 bits.

Below two bits the baselines stop producing usable derivations altogether. Every post-training and quantization-aware baseline scores zero on MBPP and HumanEval, and at most 9% retention on GSM8K, while still retaining 67–80% on QA9: the knowledge a likelihood-scored suite measures survives a width at which derivations do not. EdgeRazor lifts the generative benchmarks off zero, and OPD then multiplies what QAD recovers by 1.6–$2.9\times$ on each of them, reaching 64% on GSM8K and 77% on MBPP. Short-form performance is unaffected by this shift: QA9 retention moves from 84% to 88%. The methods that compete with us on knowledge retention are therefore not the ones that recover reasoning.

6 Ablations

To quantify the benefit of switching to on-policy recovery after QAD initialization, we compare OPD with continued teacher-forced QAD under matched training conditions. Both arms use the same starting checkpoints, corpora, mathematics-to-code schedule, and optimizer-step budgets. For continued QAD, we generate reference responses with the same teacher on the training prompts.

Across Qwen3-0.6B and Qwen3-1.7B at both bit widths, OPD outperforms continued QAD on all four benchmarks, winning all 16 comparisons (Table 4). For Qwen3-1.7B at 2.79 bits, OPD nearly doubles the MATH-500 accuracy of continued QAD, reaching 45.6% versus 23.2%. The same configuration gains 9.70 GSM8K percentage points with OPD, while continued QAD leaves accuracy essentially unchanged. More importantly, extending the QAD mathematics phase to two and three times the matched budget yields essentially no further gain across the benchmarks (Appendix G). By contrast, switching to on-policy training after QAD initialization unlocks reasoning gains that continued teacher forcing fails to reach.

7 Conclusion

Teacher-forced QAD restores broad short-form capabilities after extreme quantization, yet leaves long reasoning trajectories vulnerable to accumulated deviations and repetitive loops. OPD extends teacher supervision to prefixes generated through the deployment quantized path, targeting the trajectories where these failures arise. Across four models at 2.79 and 1.88 effective bits, OPD doubles average BF16 retention on MATH-500 from 35% to 70%, improves code generation, and preserves the broad capabilities restored by QAD. These gains arrive in a few hundred optimizer steps, alongside improved termination and reduced repetition.

AI USE STATEMENT

We used generative AI tools as a writing aid for the manuscript text and for routine code and plotting assistance. We did not use them to generate research ideas, experimental designs, or results, and all reported measurements come from the experiments described in the paper. We have reviewed all AI-assisted text and code, and we take responsibility for the final content of this work.

ETHICS STATEMENT

This work compresses existing open-weight models and does not introduce new data collection involving human subjects. Aggressive quantization changes model behaviour unevenly across capabilities, as our results show, and a model that passes a short-form benchmark suite may still fail at multi-step reasoning; we report per-task rather than aggregate numbers for that reason. Compression reduces the cost of deploying capable models, which lowers the barrier to both beneficial and harmful uses.

REPRODUCIBILITY STATEMENT

Section 5.1 gives the models, mixed-precision configurations, optimizer settings, rollout counts, step budgets and evaluation protocols; Section 4 gives the objective and the constraint on the quantized rollout path, and Appendix A gives the checkpoint forms a reimplementation must respect. The matched-budget control in Section 6 is specified by the list of quantities held fixed. The code is provided in the supplementary material.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient quantization-aware training for large language models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Wenhua Cheng, Weiwei Zhang, Haihao Shen, Yiyang Cai, Xin He, Kaokao Lv, and Yi Liu. Optimize weight rounding via signed gradient descent for the quantization of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP*, pp. 11332–11350, 2024.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. BitDistiller: Unleashing the potential of sub-4-bit LLMs via self-distillation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *International Conference on Machine Learning (ICML)*, 2024.

Falcon-LLM Team. The Falcon 3 family of open models. `https://huggingface.co/blog/falcon3`, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, et al. The language model evaluation harness. `https://github.com/EleutherAI/lm-evaluation-harness`, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

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

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards streamlined distillation for large language models. In *International Conference on Machine Learning (ICML)*, 2024.

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. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Deokjae Lee and Hyun Oh Song. Q-Palette: Fractional-bit quantizers toward optimal bit allocation for efficient LLM deployment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Janghwan Lee, Sihwa Lee, Jinseok Kim, Yongjik Kim, Jieun Lim, Jinwook Oh, and Jungwook Choi. ReQAT: Achieving full-precision reasoning accuracy with 4-bit floating-point quantization-aware training. In *International Conference on Machine Learning (ICML)*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

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. *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2024.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. `https://thinkingmachines.ai/blog/on-policy-distillation/`.

Keyu Lv, Manyi Zhang, Xiaobo Xia, Jingchen Ni, Shannan Yan, Xianzhi Yu, Lu Hou, Chun Yuan, and Haoli Bai. What makes low-bit quantization-aware training work for reasoning LLMs? a systematic study. *arXiv preprint arXiv:2601.14888*, 2026.

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. *arXiv preprint arXiv:2402.17764*, 2024.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations (ICLR)*, 2024a.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *European Conference on Computer Systems (EuroSys)*, 2025.

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. In *International Conference on Machine Learning (ICML)*, pp. 57587–57613, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning (ICML)*, 2023.

Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, 2025.

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

Shu-Hao Zhang, Le-Tong Huang, Xiang-Sheng Deng, Xin-Yi Zou, Chen Wu, Nan Li, Shao-Qun Zhang, and Zhi-Hua Zhou. EdgeRazor: A lightweight framework for large language models via mixed-precision quantization-aware distillation. *arXiv preprint arXiv:2605.04062*, 2026.

A THE EDGERAZOR QAD INITIALIZATION

Every arm in this paper starts from a checkpoint produced by EdgeRazor's mixed-precision QAD recipe (Zhang et al., 2026), so what that stage does and how faithfully we reproduced it bounds every number we report.

The quantizer. Weights are quantized per block of 256 input channels. A block is taken either to ternary values $\{-1, 0, 1\}$, which costs $\log_2 3 \approx 1.58$ bits, or to INT4. The ternary scale is not an absmax: it is twice the block's mean absolute weight, so the rounding clips outliers rather than stretching the grid to reach them, while INT4 blocks do use a per-block absmax. Which blocks get the wider format is decided by position alone. With a mixed-precision proportion p, rows of the output dimension are selected at spacing $1/p$ and a selected row is INT4 across all of its blocks; $p = 0.5$ takes every second row and gives $0.5 \cdot 4 + 0.5 \cdot 1.58 = 2.79$ effective bits, and $p = 0.125$ takes every eighth row for 1.88. Because the rule is positional rather than saliency-based, it needs no calibration data and reproduces exactly. Embedding and output-head weights bypass the mixture and are taken to per-block INT4. Activations on the decoder linears are INT8, symmetric absmax over the same block size. Our arms leave the KV cache at 16 bits, one step looser than the released `a8kv8` configuration; we note it because the direction of that deviation favours us. Mathematics and QA are evaluated on deployment exports of these weights; code is evaluated on the trainable quantized checkpoint.

Our QAD run. We reproduce the QAD stage ourselves, on the same models, from the released repository and recipe, and every OPD arm in this paper starts from a checkpoint of that reproduction. It distils on a general instruction mixture, not on the mathematics and code pools OPD later uses, with the student's own BF16 copy as teacher. The objective is an online logit KL against that teacher plus a small task loss. We follow the released hyperparameters – learning rate 2×10^{-5} held constant after warmup, sequence length 1024, 8-bit AdamW, ZeRO-3 on eight GPUs – with one deviation that matters and one we judged not to. The effective batch is 768, below the 1024 and 1536 the recipe uses for Qwen3-0.6B and Qwen3-1.7B: the KL term materialises logits over a $151{,}936$-token vocabulary, which at the published per-device batch exceeds 70 GB on its own, so we reduced the per-device batch and recovered part of it through gradient accumulation, taking more optimizer steps inside a fixed token budget instead of restoring the batch. We also skipped the optional offline step that regenerates each assistant turn with the teacher, because the released training code reads the original files rather than the regenerated ones, and because the recipe's supervision is the online KL either way.

Table 5 gives the per-arm budgets of that reproduction.

Table 5: QAD budgets for the six Qwen3 arms. Per-device batch times gradient accumulation gives 96 sequences per GPU per optimizer step, so the effective batch is 768 on eight GPUs throughout. Wall-clock is the full QAD run.

Model	Width	Steps	Epochs	Per-device × accum	Effective batch	Wall-clock
Qwen3-0.6B	W2.79	12,800	2.0	12×8	768	55 h
	W1.88	12,800	2.0	12×8	768	55 h
Qwen3-1.7B	W2.79	6,400	1.0	6×16	768	44 h
	W1.88	12,800	2.0	6×16	768	87 h
Qwen3-4B	W2.79	9,600	1.5	6×16	768	124 h
	W1.88	6,400	1.0	6×16	768	102 h

B QA9 PER TASK

Table 6 expands the QA9 column of Table 2 into its nine tasks, for every model, width and recovery stage. The tasks are ARC-Easy and ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), SocialIQA (Sap et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), TruthfulQA (Lin et al., 2022) and MMLU (Hendrycks et al., 2021). Table 7 does the same for the QA9 column of Figure 4.

C BASELINE ALIGNMENT

Every baseline in Figure 4 is brought onto the setting our own arms run at, so that a score difference is the quantization method rather than a difference in what was quantized.

Activations and embeddings. All arms carry INT8 activations on the decoder linears, symmetric absmax over blocks of 256 input channels, the quantizer our configuration uses. On QA9 the embedding and lm_head are also taken to INT4 over blocks of 256, matching the override in our recipe; the generative benchmarks leave them in BF16, which favours the baselines. The isolated cost of the embedding override is small: on Qwen3-1.7B it moves GSM8K by -0.23 points for BF16 weights and -0.45 for FlatQuant at 4 bits.

Width. Our arms mix INT1.58 and INT4 rows, 0.5 of each, for 2.79 effective bits, and the analogous mixture for 1.88. Q-Palette constructs fractional widths natively and is run at the same two widths. EfficientQAT, FlatQuant and AutoRound emit a single width per run, so they fall back to the next whole width, 3 bits against our 2.79 and 2 bits against our 1.88. Rounding up rather than down keeps the fallback in the baseline's favour.

Width-matched QA9. On QA9 the extra bit was large enough to reorder the comparison, so the upper band also matches the width. Our row rule is positional – every second output row of every decoder linear is INT4 – so it can be reproduced without a saliency criterion. Lacking an INT1.58 output, a baseline reaches 2.79 from the widths it does produce, taking a fraction p of its 2-bit solution and $1 - p$ of a wider one: $p = 0.21$ against 3 bits, $p = 0.605$ against 4 bits. Both are built and the better one reported. Mixing is row-wise for AutoRound and EfficientQAT, whose saved weights approximate the original layer directly. FlatQuant folds a per-channel scale into the preceding LayerNorm, so a row taken from one width would be expressed in the other's basis; its mixture is therefore layer-wise, with each layer's norms travelling with its linears.

Reproduction details. Baselines run from their released implementations, with one exception. Q-Palette's official entrypoints are Llama-only and depend on custom CUDA kernels, so its arm is our own pure-PyTorch reimplementation: a Walsh–Hadamard rotation, then output rows split across two adjacent codebook levels so the mean width matches the target. It is data-free, and it is also the arm that leads QA9 and MBPP, so the comparison there is against our reading of the method rather than the authors' code. The methods that do calibrate share one corpus, the 10k-document Pile subset that AutoRound uses by default, which we substituted for FlatQuant's

Table 6: QA9 by task. Columns are ARC-Easy, ARC-Challenge, HellaSwag, SocialIQA, OpenBookQA, PIQA, WinoGrande, TruthfulQA and MMLU, each scored under the protocol of Section 5.1; MMLU is its own 57-subject aggregate and enters the mean as one task. The last column is the equally weighted mean and reproduces the QA9 column of Table 2.

		ARC-e	ARC-c	HellaS	SIQA	OBQA	PIQA	WinoG	TQA	MMLU	QA9
Qwen3-0.6B											
BF16		56.19	33.79	47.26	39.25	31.40	67.36	56.27	42.84	40.34	46.08
W2.79	RTN	25.51	25.77	26.44	34.29	27.20	53.32	48.46	48.00	24.82	34.87
W2.79	QAD	53.45	30.20	37.57	40.89	28.20	64.15	54.70	44.01	37.70	43.43
W2.79	+ OPD	53.20	30.97	38.75	40.43	29.00	65.07	53.91	45.23	36.17	43.64
W1.88	RTN	25.97	26.28	26.16	33.16	27.20	50.92	48.78	49.65	24.56	34.74
W1.88	QAD	47.90	28.07	33.53	37.67	28.00	63.06	52.49	43.98	30.89	40.62
W1.88	+ OPD	50.63	28.67	35.53	40.43	27.20	64.31	53.75	41.31	34.65	41.83
Qwen3-1.7B											
BF16		70.20	43.43	60.37	45.09	37.40	72.25	60.85	45.84	55.45	54.54
W2.79	RTN	26.05	25.00	26.55	33.52	29.60	52.39	50.59	49.84	25.85	35.49
W2.79	QAD	63.80	37.12	49.65	43.65	32.20	69.15	54.54	48.84	44.54	49.28
W2.79	+ OPD	68.98	41.21	56.77	48.26	33.00	71.33	59.27	44.30	43.21	51.81
W1.88	RTN	26.30	25.94	26.10	34.08	28.20	51.69	48.62	50.23	25.86	35.22
W1.88	QAD	58.00	32.94	44.41	41.66	31.00	66.21	55.41	43.44	39.56	45.85
W1.88	+ OPD	61.99	36.86	46.59	43.81	34.00	68.61	54.22	44.64	40.34	47.90
Qwen3-4B											
BF16		78.58	53.75	68.40	49.95	40.40	75.03	65.59	54.68	68.30	61.63
W2.79	RTN	25.46	25.26	26.43	33.21	27.60	51.36	50.83	49.07	24.02	34.80
W2.79	QAD	73.53	45.73	57.83	47.13	37.00	72.58	61.56	51.16	51.93	55.38
W2.79	+ OPD	75.59	49.23	60.08	48.52	38.00	74.05	64.72	49.62	56.54	57.37
W1.88	RTN	24.96	26.02	26.57	33.93	29.00	50.38	48.70	49.34	24.08	34.78
W1.88	QAD	45.41	28.92	37.41	42.63	28.80	64.53	55.09	44.08	36.16	42.56
W1.88	+ OPD	70.79	44.28	52.76	47.95	33.00	72.52	59.91	49.52	44.74	52.83
Falcon3-1B-Instruct											
BF16		68.18	45.56	63.10	45.60	40.40	74.92	60.30	45.59	43.85	54.17
W2.79	RTN	37.88	25.85	35.80	35.57	25.60	58.49	50.59	45.59	24.25	37.74
W2.79	QAD	70.33	44.03	57.97	49.18	38.00	73.72	59.67	42.02	41.13	52.89
W2.79	+ OPD	71.13	44.71	57.65	50.05	37.80	73.07	59.51	42.05	41.28	53.03
W1.88	RTN	30.56	22.53	26.69	34.03	27.60	52.50	51.62	48.99	23.71	35.36
W1.88	QAD	66.71	39.33	51.55	47.54	34.60	72.09	56.99	40.28	36.72	49.53
W1.88	+ OPD	66.75	39.59	51.35	47.44	34.80	72.31	57.30	40.20	36.51	49.58

WikiText-2 and EfficientQAT's RedPajama defaults so that no arm differs from another in what it saw; weight group size is 128 wherever a method exposes it, and sample counts and epoch budgets follow each method's published recipe. One asymmetry is left uncorrected because it favours the baselines: on the generative benchmarks their embedding and lm_head stay in BF16 while ours are INT4.

D Training efficiency

Figure 5 gives the per-step comparison behind Section 5: across the four models, OPD yields more GSM8K points per thousand optimizer steps than continued teacher-forced QAD from the same starting checkpoint. For Qwen3-4B at 2.79 bits, just 20 OPD steps gain 4.32 points, exceeding the 4.09-point gain from 800 QAD steps.

E OPD training curves

Figure 6 shows the phase-1 run for Qwen3-4B at 1.88 bits, the arm whose QAD start is the most degraded of the six and where the recovery is therefore easiest to read. The run divides into a short repair phase and a long plateau. Over the first thirty steps the policy entropy falls from about 10

Table 7: QA9 by task for the Qwen3-1.7B arms of Figure 4. Columns are as in Table 6; Ret. is the mean as a percentage of the BF16 row and is what the figure plots. The BF16 row is the same unquantized reference as in Table 6, and the EdgeRazor and Ours rows are that table's QAD and + OPD rows for this model.

	ARC-e	ARC-c	HellaS	SIQA	OBQA	PIQA	WinoG	TQA	MMLU	QA9	Ret.
BF16	70.20	43.43	60.37	45.09	37.40	72.25	60.85	45.84	55.45	54.54	—
≈ 2.79 *bits*											
EfficientQAT	40.32	27.99	42.90	37.31	26.60	62.84	51.30	45.80	24.88	39.99	73.3
FlatQuant	52.57	33.70	48.98	38.54	30.00	66.87	56.04	44.22	42.99	45.99	84.3
AutoRound	54.08	32.17	49.06	41.86	32.60	66.87	56.35	44.19	36.65	45.98	84.3
Q-Palette	59.39	39.33	55.77	41.56	36.20	69.70	59.04	43.69	51.69	50.71	93.0
EdgeRazor	63.80	37.12	49.65	43.65	32.20	69.15	54.54	48.84	44.54	49.28	90.4
Ours	68.98	41.21	56.77	48.26	33.00	71.33	59.27	44.30	43.21	51.81	**95.0**
≈ 1.88 *bits*											
EfficientQAT	33.75	23.81	33.09	35.06	25.80	58.87	50.67	45.29	22.95	36.59	67.1
FlatQuant	39.39	25.09	34.28	35.41	26.40	58.81	52.96	47.02	23.14	38.06	69.8
AutoRound	44.78	27.22	40.03	38.64	30.20	61.48	55.49	41.56	29.11	40.94	75.1
Q-Palette	48.65	31.31	45.45	39.92	31.60	64.91	54.22	42.31	35.42	43.76	80.2
EdgeRazor	58.00	32.94	44.41	41.66	31.00	66.21	55.41	43.44	39.56	45.85	84.1
Ours	61.99	36.86	46.59	43.81	34.00	68.61	54.22	44.64	40.34	47.90	**87.8**

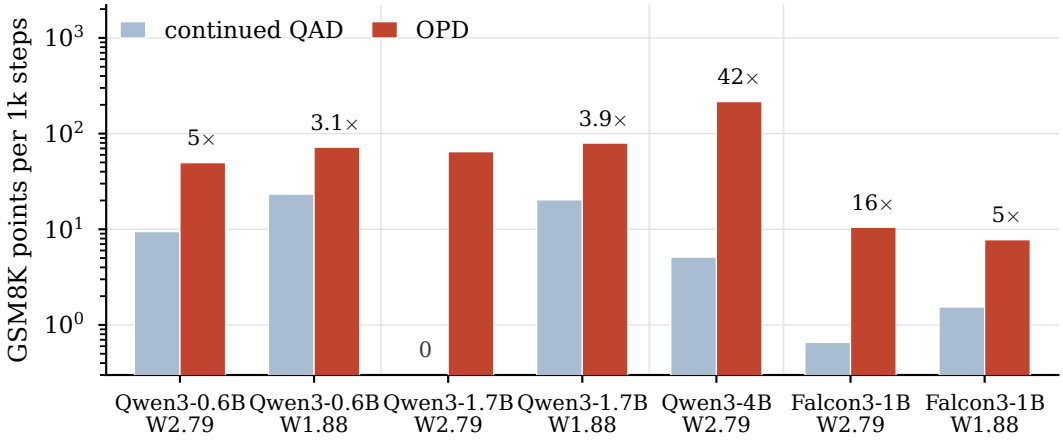


Figure 5: OPD versus continued QAD on GSM8K, in points per thousand optimizer steps from the same starting checkpoint. Qwen3-0.6B and Qwen3-1.7B use the matched step budget of Table 4; Qwen3-4B and Falcon3-1B use a late QAD segment. The vertical axis is logarithmic. A zero marks a non-positive QAD gain.

nats per token, an effective support of tens of thousands of tokens, to 0.35: the quantized start is close to emitting arbitrary text, and teacher supervision on its own prefixes restores a usable output distribution almost immediately. The distillation term falls from 2.07 to 0.25 over the same interval and the verifier reward rises from -0.39 to roughly $+0.15$. The remaining ninety steps hold the reward between 0.1 and 0.2.

The two terms of Equation 3 sit on very different scales. With $\beta = 1$ the distillation term accounts for almost all of the total loss, while the policy-gradient term stays near 0.05 throughout. Single-step reward spans roughly ± 0.5 because a step scores only 32 sequences, so the smoothed curve rather than the raw trace carries the trend.

F Teacher choice

A natural worry is that OPD's gain is borrowed capacity from a stronger teacher rather than the on-policy objective. The default recipe uses each student's own BF16 weights; Qwen3-0.6B is the

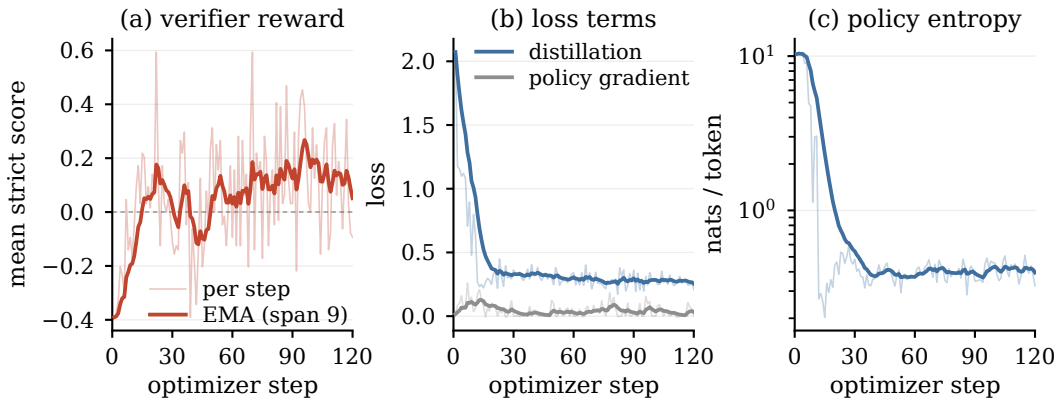


Figure 6: OPD phase-1 training for Qwen3-4B at W1.88: 120 optimizer steps on the clean mathematics pool. Faint lines are per-step values and solid lines an exponential moving average over nine steps. (a) mean strict verifier score over the 32 sequences of a step. (b) the two terms of Equation 3. (c) policy entropy, logarithmic axis.

one exception, because that model's BF16 copy is too weak to supervise. Table 8 separates the two cases.

From 1.7B up, a larger teacher does not help. On Qwen3-1.7B at 2.79 bits, self-distillation matches or beats 4B and 8B teachers on both GSM8K and MATH-500, within run-to-run variation on GSM8K and ahead by 1.9 points on MATH-500. On Qwen3-4B at 2.79 bits, switching the teacher to 8B after the self-teacher mathematics checkpoint does not improve either metric. At 1.88 bits the same student's own BF16 weights are not merely sufficient but better: matched 80-step arms from the same QAD start gain +18.88 GSM8K with the self teacher against +15.62 and +15.84 for the 4B and 8B teachers, and lead MATH-500 by 6.4 and 6.8 points. On Qwen3-4B at 1.88 bits the two teachers are level instead: from a start that quantization had driven down to 17.36 GSM8K, the self teacher recovers +48.60 against +47.77 for the 8B teacher, and MATH-500 splits the other way, +34.20 against +35.40. Across both widths and all three students, then, no teacher larger than the student buys anything its own BF16 copy does not: the recovery reported in the main text is the method, not a stronger model.

Qwen3-0.6B is a special case. A probe on the mathematics training pool, four samples per problem, gives teacher pass@4 of 62% for Qwen3-0.6B against 83% for 1.7B and 89% for both 4B and 8B, while the quantized 0.6B student sits at 36–39%. Its own BF16 teacher is barely ahead of the student, so there is little to transfer. Matched OPD arms confirm the gap: self-distillation gains only +2.80 and +6.44 GSM8K points at the two widths, whereas every teacher of 1.7B or above gains +11.0 to +14.1. Scaling past 1.7B buys nothing: at 2.79 bits the 8B teacher adds 0.23 GSM8K points, below the spread between adjacent checkpoints of a single arm, and at 1.88 bits the 4B and 8B teachers fall 1.8 and 3.1 points behind it, losing MATH-500 at both widths. We therefore keep Qwen3-1.7B as the 0.6B teacher and use every larger student's own BF16 weights.

G Extended QAD budget

The matched-budget control in Table 4 equalizes optimizer steps, so a remaining objection is that teacher-forced QAD simply needs a longer run. We therefore keep the same starting checkpoints and mathematics corpus, and extend the QAD mathematics phase to 2× and 3× the matched step budget; the 3× run covers roughly one epoch of the training pool. Table 9 compares these arms with the OPD mathematics phase alone.

Extra teacher-forced steps do not close the gap. At 3×, QAD still trails OPD by 8.9–18.1 GSM8K points on every configuration, and on Qwen3-0.6B at 2.79 bits the 3× run falls below the 1× baseline. The advantage of switching to on-policy recovery after QAD is therefore not an artifact of the matched step count.

Table 8: Teacher size. Entries are gains over a QAD baseline fixed per student and width, so a row reports what OPD recovered rather than what the model scores. Baselines, as GSM8K / MATH-500: Qwen3-0.6B 33.21 / 10.00 at 2.79 bits and 22.52 / 2.60 at 1.88; Qwen3-1.7B 44.35 / 17.80 and 26.46 / 8.20; Qwen3-4B 69.45 / 39.80 and 17.36 / 2.20. A baseline is shared by every row of its block, so the differences between teachers do not depend on it. Steps counts OPD steps. A row's two columns come from a single checkpoint, except for the Qwen3-1.7B rows at 2.79 bits: that line swings enough between adjacent checkpoints that both columns are means over the last four of its 200 steps. Every teacher within a block is trained to the same budget except Qwen3-4B at 2.79 bits, whose two rows are not matched. Shaded rows are the teacher used in the main experiments.

Student	Teacher	Width	Steps	ΔGSM8K	ΔMATH-500
Qwen3-0.6B	self	W2.79	80	+2.80	+9.40
		W1.88	90	+6.44	**+9.20**
	1.7B	W2.79	80	+13.87	**+13.80**
		W1.88	90	**+14.10**	+9.00
	4B	W2.79	80	+11.75	+10.00
		W1.88	90	+12.35	+3.00
	8B	W2.79	80	**+14.10**	+11.00
		W1.88	90	+10.99	+7.40
Qwen3-1.7B	self	W2.79	200	**+9.20**	**+13.55**
		W1.88	80	**+18.88**	**+16.40**
	4B	W2.79	200	+8.77	+11.65
		W1.88	80	+15.62	+10.00
	8B	W2.79	200	+8.34	+13.05
		W1.88	80	+15.84	+9.60
Qwen3-4B	self	W2.79	80	**+8.41**	**+14.60**
		W1.88	120	**+48.60**	+34.20
	8B	W2.79	60	+7.20	+13.40
		W1.88	120	+47.77	**+35.40**

Table 9: Mathematics-only recovery with QAD trained for 2× and 3× the matched budget of Table 4. The OPD column is the mathematics phase only; the gap is OPD minus QAD at 3×. Baselines are re-evaluated separately for this control.

Model	Width	QAD 1×	QAD 2×	QAD 3×	OPD phase 1	gap
Qwen3-0.6B	W2.79	33.43	36.39	27.67	**45.72**	+18.1
	W1.88	30.63	26.16	27.29	**37.23**	+9.9
Qwen3-1.7B	W2.79	45.34	48.75	44.05	**52.92**	+8.9
	W1.88	34.87	38.06	36.32	**45.34**	+9.0