TOWARDS DENSE CAPABILITY RECOVERY OF TERNARY QUANTIZED LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Ternary quantization offers extreme compression, but the resulting perturbation is substantially more severe than conventional low-bit quantization. The key challenge is not merely reducing one-shot projection error, but finding a ternary representation that remains efficiently recoverable through subsequent training. We demonstrate that under matched knowledge distillation (KD), quantizers with vastly different initial errors converge to remarkably similar functional endpoints. Motivated by this, we introduce a calibration-free, energy-preserving quantizer as a stable starting point designed specifically to facilitate subsequent training. However, same-size KD restores broad post-trained capability while leaving substantial and highly nonuniform performance gaps across tasks. To address this, we first exploit test-time sampling to expose substantial residual headroom on several reasoning and coding tasks. We then apply quantization-aware reinforcement learning (RL) directly to the ternary policy, converting part of this headroom into stronger single-shot performance. We evaluate our approach across several LLMs sizes: Qwen3-1.7B, 4B, and 8B. The first math-RL stage substantially improves single-shot performance, while broader gains transfer across knowledge, reasoning, instruction following, mathematics, code, and the unoptimized thinking mode. At 8B, the final 2.03-GB model achieves 99.2% mean benchmark-wise retention relative to BF16 in non-thinking mode and 94.1% in thinking mode, while decoding at 77.9 tokens/s on an M5 Pro. Together, these results support a recovery-centric view of extreme quantization: preserve a recoverable ternary representation, restore the model distribution, and then optimize the deployed ternary policy directly.

1 INTRODUCTION

Ternary language models replace dense weights with only three values, $\{-1, 0, +1\}$, yielding a much larger reduction in weight storage than conventional 4-bit quantization. For a pretrained language model, however, ternary quantization is an unusually severe intervention. The dense checkpoint already contains knowledge, reasoning procedures, instruction-following behavior, and generation policies acquired through a large pretraining and post-training budget; collapsing each weight group to a signed support and a single magnitude removes most of its within-group magnitude information. Consequently, capabilities that are largely preserved under moderate quantization can degrade sharply after ternary quantization.

This setting differs from both native ternary training and conventional post-training quantization. When a LLM is trained under a ternary constraint from the beginning, its representations can adapt throughout training to the restricted weight space. When converting an already-trained dense LLM, the immediate projection is only the start of the problem. A quantizer may minimize local weight reconstruction error yet still produce a poor starting point for subsequent optimization, while two projections with very different one-shot quality may become functionally similar after sufficient recovery. For pretrained ternary quantization, we therefore ask not only how accurately a projection reconstructs the dense weights, but how much useful structure it preserves and how efficiently the resulting ternary LLM can be recovered.

We study this problem on post-trained Qwen3 instruct LLMs from 1.7B to 8B parameters. The attention and feed-forward weight matrices are ternarized in groups of 128, and the token-embedding

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.

and language-model output matrices are also ternary; parameter tying follows the underlying architecture. Normalization parameters and other small scalar parameters remain in their native precision.

We focus on three questions. First, how strongly does the choice of ternary quantizer affect the eventual LLM once substantial recovery is allowed? Second, how much broad capability can same-size knowledge distillation restore while the student remains hard ternary in every forward pass? Third, can the remaining recoverable capability be improved by reinforcement learning without leaving the ternary policy that will actually be deployed?

Our results suggest a simple recovery pipeline. We first apply a moment-consistent ternary quantizer that jointly controls support density and shared scale. We then use knowledge distillation to recover the teacher distribution under the hard ternary constraint. The resulting LLM recovers broad language-model behavior, but the residual error is highly nonuniform: some tasks remain close to the dense LMM while others retain substantial sampling headroom. We use pass@N as a diagnostic of this remaining headroom and apply RL only to the policy that is actually ternary at training and rollout time.

These observations lead to a recovery-centric view of extreme quantization:

$$\text{ternarize} \rightarrow \text{recover the distribution} \rightarrow \text{identify the remaining failure} \rightarrow \text{recover the policy}.$$

Using this recipe, we recover broad capability across knowledge, reasoning, mathematics, code generation and instruction following, while retaining the memory and runtime advantages of ternary weights. We additionally report tool-use and long-horizon agentic evaluations as stress tests, but do not treat them as recovered capabilities; they remain an important direction for future work.

Our main contributions are:

- **End-to-end capability recovery under the ternary constraint.** We characterize recovery from KD through direct ternary RL across Qwen3-1.7B, 4B, and 8B. Repeated sampling shows that part of the post-KD deficit reflects unreliable selection of reachable behaviors rather than complete capability loss, and direct hard-ternary RL converts part of this headroom into stronger single-shot performance.
- **Recovery-centric analysis of ternary quantization.** We introduce a calibration-free, moment-consistent ternary quantizer and compare it against alternative quantization rules under matched recovery. Large differences immediately after projection contract substantially after long recovery, showing that one-shot reconstruction quality is a poor proxy for the final recovered LLM.
- **Broad evaluation and real-device deployment.** We compare dense, Q4, and recovered ternary LLMs across knowledge, reasoning, instruction following, mathematics, and code, together with diagnostic tool/agentic evaluations and real-device inference. The recovered LLMs define practical quality–memory–throughput operating points, while exposing the remaining limits of extreme ternary quantization.

2 Related Work

Ternary quantization and pretrained conversion. Ternary Weight Networks (TWN) introduced the classical $\{-1, 0, +1\}$ parameterization with thresholded support and a shared scale chosen from a weight-reconstruction objective (Li et al., 2016), while Trained Ternary Quantization (TTQ) learned ternary assignments and magnitudes during training (Zhu et al., 2017). For language models, BitNet and BitNet b1.58 demonstrated strong scaling under native 1-bit/ternary training, with BitNet b1.58 adopting an AbsMean-based ternary rule (Wang et al., 2023; Ma et al., 2024).

Recent work attacks pretrained ternary conversion directly. PT^2-LLM combines iterative fitting with activation-aware grid alignment (Yan et al., 2026); CAT-Q uses learned modulation and softened ternary quantization with a small calibration set (Wang et al., 2026a); and Tequila addresses deadzone trapping during ternary optimization (Huang et al., 2025). ScaleQ-1.58 extends CAT-Q to reasoning models by calibrating on model-generated reasoning traces, substantially improving ternary PTQ on mathematics and code while scaling across dense and MoE Qwen3 models (Wang et al., 2026b). Ternary Bonsai further provides released end-to-end ternary Qwen3 checkpoints at

1.7B, 4B, and 8B scale, demonstrating that practically useful post-trained ternary models can be recovered at the same model scales studied here (Prism ML, 2026).

These results make the existence of useful post-trained ternary models less of an open question. Our focus is instead on the *recovery process*: how much of a post-trained capability stack can be recovered under a hard-ternary constraint, which errors remain after distributional recovery, and whether those residual errors can be reduced by optimizing the deployed ternary policy directly.

Post-quantization recovery and distillation. Low-bit recovery commonly combines quantization-aware training with reconstruction or distillation objectives. BitDistiller couples sub-4-bit QAT with self-distillation to improve language understanding and reasoning (Du et al., 2024), while BitNet Distillation adapts pretrained full-precision models to ternary weights using continual training and intermediate-feature distillation (Wu et al., 2025). BitCPM-CANN further demonstrates that strong recovery is possible with a simple AbsMean-style groupwise ternary quantizer. Starting from dense MiniCPM4 checkpoints, it applies quantization-aware training followed by distillation, and reports approximately 96–97% average retention for its 1B–8B models on an 11-task knowledge and reasoning suite (BitCPM Team, 2026).

Together with our controlled quantizer ablation, this motivates treating the quantizer as an optimization substrate rather than assuming that one-shot reconstruction quality determines the final model.

Our setting therefore does not rely on the claim that a particular ternary quantizer is uniquely necessary for high retention. Instead, we study recovery more directly and over a broader post-trained capability suite, including instruction following, mathematics, code, and thinking-mode behavior, while using tool-use, agentic interaction, and long-horizon generation as diagnostic stress tests.

Reinforcement learning under quantization. Outcome-based RL, including GRPO, has become a standard mechanism for improving reasoning in language models (Shao et al., 2024; DeepSeek-AI, 2025). BitRL recently studied RL with extremely quantized language-model backbones, including a frozen-backbone regime (Sajid et al., 2026). Our setting instead optimizes the ternary backbone itself: continuous latent parameters accumulate gradient updates, while both actor forwards and rollout generation use the corresponding hard-ternary weights. We additionally verify synchronization between the materialized rollout policy and the ternary policy used for gradient computation.

3 From Dense to Ternary Models

We represent each weight group $\mathbf{w} \in \mathbb{R}^G$ as $\widehat{\mathbf{w}} = s\mathbf{q}$, where $q_i \in \{-1, 0, +1\}$ and $s > 0$ is shared within the group. Across Qwen3-1.7B, 4B, and 8B, attention and feed-forward matrices use group size $G = 128$. The token-embedding and language-model output matrices are also ternary; when the base architecture ties them, the shared parameter remains ternary. Normalization parameters and other small scalar parameters remain in their native precision. Full parameterization details are given in Appendix A.1.

3.1 What Should a Ternary Projection Preserve?

A symmetric ternary group has two coupled degrees of freedom: which coefficients remain nonzero and the shared magnitude assigned to them. Different projection objectives preserve different properties, and in general moment matching need not minimize coordinatewise reconstruction error (Orr et al., 2025).

Our goal is therefore not a universally optimal one-shot projection. Instead, we seek a cheap, data-free, calibration-free quantization rule that preserves simple group statistics and provides a stable starting point for subsequent recovery.

For a dense group, define

$$m_1 = \frac{1}{G}\sum_{i=1}^{G} |w_i|, \qquad m_2 = \frac{1}{G}\sum_{i=1}^{G} w_i^2, \tag{1}$$

and let ρ denote the fraction of nonzero ternary codes. Because every nonzero ternary coefficient has magnitude s, the projected group has

$$\widehat{m}_1 = \rho s, \qquad \widehat{m}_2 = \rho s^2. \tag{2}$$

For a fixed support density ρ, the single scale s can in general preserve only one of these moments: matching the first moment requires $s = m_1/\rho$, whereas preserving weight energy requires $s = \sqrt{m_2/\rho}$. This motivates treating the support density itself as a design variable.

3.2 Moment-Consistent Ternary Quantizer

We choose the nonzero density ρ and shared magnitude s jointly so that the ternary group matches both dense-group moments:

$$\rho s = m_1, \qquad \rho s^2 = m_2. \tag{3}$$

Solving gives

$$\boxed{\rho^\star = \frac{m_1^2}{m_2}, \qquad s^\star = \frac{m_2}{m_1}}. \tag{4}$$

This solution is distribution-free and requires no calibration activations. By Cauchy–Schwarz, $0 < \rho^\star \leq 1$ for any nonzero group. Matching m_2 also preserves the squared ℓ_2 norm of the group, so the target projection does not systematically attenuate or amplify its weight energy.

Exact realization of $\rho^\star$ would require selecting a different fraction of coefficients in each group. To obtain a fixed, calibration-free threshold without sorting, we approximate the group distribution as $w \sim \mathcal{N}(0, \sigma^2)$, giving

$$m_1 = \sigma\sqrt{\frac{2}{\pi}}, \qquad m_2 = \sigma^2, \qquad \rho^\star = \frac{2}{\pi} \approx 0.637. \tag{5}$$

The corresponding symmetric threshold is $\Delta \approx 0.592 m_1$, which we round to $0.6 m_1$; this coefficient is derived rather than tuned on downstream tasks.

Our practical quantizer is therefore

$$q_i = \operatorname{sign}(w_i)\, \mathbf{1}\{|w_i| > 0.6 m_1\}, \qquad \widehat{\rho} = \frac{1}{G}\sum_{i=1}^{G} q_i^2, \qquad s_{\mathrm{E}} = \sqrt{\frac{m_2}{\widehat{\rho}}}. \tag{6}$$

During quantization-aware recovery, this rule is reapplied to the latent weights on every forward pass; it therefore governs both the initial dense-to-ternary projection and subsequent support and scale changes as the latent weights move.

Because $\widehat{\rho}$ is measured after thresholding, the practical projection preserves m_2 exactly for the empirical group distribution. It also preserves m_1 when $\widehat{\rho} = \rho^\star$; otherwise only the support approximation introduces mismatch. The Gaussian assumption therefore affects support selection, not the energy-preserving scale. Full derivations are given in Appendix A.2.

3.3 Quantizer Sensitivity Under Matched Recovery

We next test how strongly the eventual recovered model depends on its initial ternary projection. All variants use Qwen3-1.7B with the same dense checkpoint, frozen BF16 teacher, training data, optimizer, group size, vocabulary-matrix quantizer, and recovery budget; only the transformer-backbone quantization rule changes. Full definitions, trajectories, and compute measurements are given in Appendix B.

The initial projection becomes substantially less predictive as recovery proceeds. By 400K updates, all four methods lie within a narrow teacher-fidelity range (KL 0.629–0.670; top-1 agreement 75.61%–76.11%), despite large differences at initialization. Teacher fidelity does not determine downstream recovery: Energy-OPT achieves the lowest KL but only 41.02% on GSM8K, while AbsMean reaches 47.81% despite the highest final KL. Energy reaches 47.01% while remaining close to the best teacher-fidelity endpoint. Thus, we treat the quantizer as an optimization substrate rather than ranking projections solely by one-shot error or endpoint KL.

Table 1: Matched recovery trajectories for ternary quantizers on Qwen3-1.7B. Large differences at initialization contract substantially under recovery, but teacher-fidelity and downstream performance remain differently ordered. GSM8K at 400K is mean ± s.d. over three decoding seeds. This controlled ablation uses the matched full-vocabulary JSD recovery objective described in Appendix B; the primary recovery trajectory in Section 3.4 uses forward KL. Additional metrics, compute cost, and evaluation details are reported in Appendix B.

	Initial		60K		100K		400K		
Projection	**KL ↓**	**GSM8K ↑**	**KL ↓**	**GSM8K ↑**	**KL ↓**	**GSM8K ↑**	**KL ↓**	**Top-1 ↑**	**GSM8K ↑**
Energy (ours)	11.78	0.08	1.4241	28.66	1.2454	16.68	0.6345	76.09	47.01 ± 0.86
Energy-OPT	**10.17**	0.83	**1.4066**	32.22	**1.2432**	20.85	**0.6288**	**76.11**	41.02 ± 0.65
AbsMean	14.81	0.00	1.4896	26.91	1.2666	23.96	0.6700	75.61	$\mathbf{47.81 \pm 0.39}$
TWN	11.65	**0.91**	1.4429	**34.42**	1.2811	**34.04**	0.6360	75.99	43.37 ± 1.11

Energy is additionally convenient for the subsequent hard-ternary RL pipeline because its materialized projection is idempotent, $Q_{\mathrm{E}}(Q_{\mathrm{E}}(w)) = Q_{\mathrm{E}}(w)$, whereas the AbsMean-style projection used for the vocabulary matrices rescales under re-quantization when the nonzero density is below one. We therefore use Energy for the transformer body during RL and freeze the already-ternarized embedding/output matrices. The derivation and implementation details are given in Appendix A.4.

3.4 Knowledge Distillation

Direct ternary quantization substantially perturbs the post-trained model, so we recover the ternary student by distillation from a frozen, same-size BF16 teacher initialized from the same dense checkpoint. The student is ternarized before optimization and remains hard ternary in every forward pass; gradients pass through the projection with an STE. Distillation is restricted to assistant-response positions under teacher forcing.

We optimize full-vocabulary forward KL,

$$\mathcal{L}_{\mathrm{KD}} = \frac{T^2}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} D_{\mathrm{KL}}\left(p_T^{(i)} \,\middle\|\, p_S^{(i)}\right), \tag{7}$$

where $\mathcal{M}$ denotes assistant-response positions and T is the distillation temperature. We anneal T from 4 to 1 over the first two epochs and use $T = 1$ in the final epoch. Distillation uses the full vocabulary: early top-k variants left the probability tail unconstrained and produced rare-token and foreign-script leakage during free generation. Full data and optimization details are given in Appendix E.

4 KD Evaluation & Residual Capability

4.1 Evaluation Protocol

We evaluate every checkpoint through a shared EvalScope-based harness (ModelScope Team, 2024), with models served locally through the vLLM inference path (Kwon et al., 2023). Our suite spans six capability areas: knowledge and reasoning, instruction following, mathematics, code, tool use, and agentic interaction. All checkpoints are evaluated zero-shot in both non-thinking and thinking modes, and a recovered checkpoint is always served with the same decoding configuration as the same-size BF16 Qwen3 reference. We follow the published Qwen3 sampling recommendations (Yang et al., 2025); the exact benchmark composition, sampling parameters, repeated-sampling settings, and simulator configuration are given in Appendix C. Tool-use and agentic benchmarks are reported as diagnostic stress tests and are excluded from Avg. and Retention.

For model M, benchmark b, and its same-size BF16 reference M_{BF16}, we define the average benchmark score and mean benchmark-wise retention as

$$\mathrm{Avg}(M) = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} S(M, b), \qquad \mathrm{Ret}(M) = \frac{100}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \frac{S(M, b)}{S(M_{\mathrm{BF16}}, b)}, \qquad |\mathcal{B}| = 11. \tag{8}$$

Retention is not clipped at 100%.

4.2 Capability Retention after KD

Knowledge distillation substantially restores the post-trained model, but recovery remains scale dependent. Using the same recovery corpus and curriculum, KD E3 reaches 83.8%, 86.2%, and 91.6% mean benchmark-wise retention at 1.7B, 4B, and 8B respectively in non-thinking mode, with the same trend in thinking mode (Table 3). This occurs despite a recovery corpus dominated by general instruction and chat data rather than task-specific supervision; for example, explicit code-reasoning data constitutes less than 0.4% of training tokens. Thus, recovery appears to depend not only on the training distribution but also on how much of the dense model's capability remains representable and recoverable after ternarization.

Low teacher-forced distillation loss nevertheless does not imply identical free-running behavior: once an early token differs, subsequent predictions condition on a different autoregressive trajectory. Appendix D analyzes this effect through teacher uncertainty and token margins.

4.3 Residual Headroom with Pass@N

Repeated sampling provides a direct diagnostic of capability that remains reachable but unreliable after KD. At KD E3, the Pass@8–Pass@1 gap is as large as 25.2 points on Math-500 and 26.1 points on the Code aggregate (Table 2). This suggests that part of the remaining deficit reflects unreliable selection of reachable trajectories rather than complete loss of the corresponding behavior. We therefore use direct ternary RL to test whether this headroom can be converted into stronger single-shot performance.

5 Capability Recovery with Reinforcement Learning

5.1 Direct RL Under the Ternary Constraint

To apply reinforcement learning to extreme ternary weights without gradient stagnation, we wrap the policy in a Quantization-Aware Training (QAT) bridge. Standard RL updates are typically too small to cross discrete $\{-1, 0, +1\}$ boundaries. We therefore maintain continuous BF16 latent parameters that absorb fractional gradient updates via a Straight-Through Estimator (STE), while every forward pass operates strictly on the hard-ternary projection. To prevent the RL loop from collapsing under discrete trajectory shifts, we explicitly verify that the materialized rollout workers and the actor calculating gradients remain closely synchronized ($r \approx 0.9996$ correlation). Finally, we recalibrate hyperparameters for the discrete ternary parameterization and we freeze the already-ternarized embedding and output matrices during RL, restricting adaptation to the ternary transformer body. The complete configuration is reported in Appendix F.3.

5.2 Ternary RL Recovers Latent Capability

Direct ternary RL consistently reduces this reliability gap. Across all three model sizes and both decoding modes, Math RL narrows the Pass@8–Pass@1 gap on GSM8K and Math-500 (Table 2). For example, at 1.7B the non-thinking Math-500 gap falls from 25.2 to 14.6 points, while at 8B the thinking gap falls from 11.6 to 5.5 points. Together with the Pass@1 gains reported in Table 3, this shows that direct ternary RL converts part of the capability visible under repeated sampling into more reliable single-sample behavior.

The gains extend beyond mathematics. At 8B, the 11-benchmark average rises from 63.6 after KD to 65.5 after Math RL and 67.5 after Multi RL, corresponding to 94.8% and 99.2% mean benchmark-wise retention versus BF16 and exceeding Bonsai's 63.9 average; gains also transfer to knowledge, instruction following, code, and the unoptimized thinking mode (see Table 3).

Sequential recovery follows the same principle:

$$\mathrm{KD} \rightarrow \text{math RL} \rightarrow \text{multi-domain RL}.$$

At each stage boundary, we re-materialize the ternary model, verify the grid, and recalibrate the next prompt pool against the updated policy.

Table 2: Residual sampling headroom after KD and its reduction under Math RL. **Gap** denotes Pass@8 − Pass@1 in percentage points. Code is the average of HumanEval+ and MBPP+. For GSM8K and Math-500, arrows show the change from KD E3 to Math RL. Absolute Pass@1 scores are reported in Table 3.

	Code gap at KD E3		**GSM8K gap: KD E3 → Math RL**		**Math-500 gap: KD E3 → Math RL**	
Size	**I**	**T**	**I**	**T**	**I**	**T**
1.7B	18.1	25.2	15.6 → **7.1**	11.3 → **6.3**	25.2 → **14.6**	22.3 → **14.2**
4B	14.2	26.1	6.8 → **3.2**	4.7 → **2.7**	14.1 → **9.0**	13.5 → **6.7**
8B	10.7	16.7	5.5 → **2.4**	3.5 → **2.1**	14.8 → **7.5**	11.6 → **5.5**

I = instruct/non-thinking; T = think. Code excludes LiveCodeBench, which is treated as a long-form diagnostic.

Table 3: Capability recovery across KD and RL stages, reported per individual benchmark. Scores are reported on a 0–100 scale for instruct (I) and think (T) decoding. **Avg.** is the unweighted mean of the 11 reported core benchmarks: Knowledge (MMLU-Pro, MMLU-Redux), Reasoning (GPQA-Diamond, MuSR), Instruction Following (IFEval, IFBench, Multi-IF), Math (GSM8K, Math-500), and Code (HumanEval+, MBPP+). Values in parentheses report mean benchmark-wise retention relative to the corresponding same-size BF16 model in the same decoding mode. AIME25/AIME26, LiveCodeBench, tool-use, and agentic evaluations are reported separately and are excluded from Avg. and Retention.

Size	**Stage**	**Know.**				**Reasoning**				**IF**						**Math**				**Code**				**Avg. (Ret.)**	
		MMLU-P		**MMLU-R**		**GPQA**		**MuSR**		**IFEval**		**IFBench**		**Multi-IF**		**GSM8K**		**Math-500**		**HE+**		**MBPP+**			
		I	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**
1.7B	BF16	45.1	57.6	66.5	74.4	32.8	38.9	50.0	57.8	68.0	71.2	17.3	21.3	57.3	60.7	81.9	90.4	73.0	90.3	57.2	77.7	65.1	66.7	55.8 (100.0)	64.3 (100.0)
	Q4	43.8	55.8	65.9	73.3	23.2	29.8	52.0	56.2	67.1	69.9	18.0	21.7	56.1	60.0	80.8	90.1	71.4	89.0	58.0	75.8	65.2	70.3	54.7 (97.2)	62.9 (97.2)
	Bonsai	29.8	–	52.6	–	20.2	–	49.3	–	55.6	–	14.0	–	42.2	–	77.0	–	50.1	–	43.9	–	48.7	–	44.0 (77.8)	–
	CAT-Q	0.6	2.0	3.8	5.2	1.5	3.0	1.7	16.7	5.2	9.2	2.7	6.7	9.0	9.7	0.2	0.2	0.7	0.8	0.0	0.0	0.0	0.0	2.3 (5.0)	4.9 (9.9)
	KD E1	22.4	21.2	44.7	45.5	20.7	16.7	46.8	45.9	56.4	57.7	15.0	13.0	43.4	43.8	59.3	62.7	37.4	33.8	26.8	23.1	44.0	35.1	37.9 (68.8)	36.2 (56.7)
	KD E2	28.1	31.3	49.7	54.1	24.8	24.2	46.6	48.8	59.3	61.4	17.0	16.7	46.3	48.4	69.3	75.3	47.4	48.5	38.0	36.8	45.8	40.6	42.9 (78.0)	44.2 (69.3)
	KD E3	32.8	37.4	55.0	58.5	25.8	29.3	48.0	52.6	62.1	65.1	16.3	18.0	49.6	51.5	75.1	83.0	57.6	61.9	40.9	45.4	50.5	41.9	46.7 (83.8)	49.5 (77.4)
	Math RL	36.0	41.3	57.2	61.7	27.3	31.3	50.3	54.9	62.1	63.2	19.4	19.7	51.5	54.9	86.7	87.4	66.6	73.8	44.4	51.4	53.2	52.9	50.4 (90.8)	53.9 (84.1)
	Multi RL	38.6	42.2	58.9	61.9	26.8	32.3	49.3	55.2	66.7	66.9	20.7	21.0	54.9	56.9	87.3	88.2	68.2	74.8	47.9	55.3	55.4	58.6	**52.2 (94.2)**	**55.8 (87.3)**
4B	BF16	61.8	69.8	79.4	83.8	40.4	53.5	54.9	60.6	79.5	81.9	26.0	26.7	72.2	73.7	91.8	95.0	84.1	95.1	74.9	88.9	76.4	70.4	67.4 (100.0)	72.7 (100.0)
	Q4	60.2	68.9	78.4	83.5	41.9	49.0	58.2	59.7	78.6	80.0	24.0	26.3	70.6	72.2	91.1	94.3	82.5	93.8	71.2	87.0	76.8	76.7	66.7 (98.9)	71.9 (98.9)
	Bonsai	52.6	–	72.8	–	32.8	–	56.8	–	75.4	–	27.0	–	62.7	–	90.6	–	80.2	–	77.3	–	78.8	–	64.3 (95.2)	–
	CAT-Q	8.7	1.3	18.5	7.6	7.6	1.5	23.8	6.2	14.4	10.7	9.3	14.0	11.8	9.0	4.2	1.4	1.3	0.9	0.0	0.0	0.0	0.0	9.1 (16.0)	4.8 (9.5)
	KD E1	34.6	36.4	56.9	61.0	26.3	25.2	51.4	53.0	68.6	66.0	17.0	16.3	56.7	57.2	80.7	82.8	58.5	58.5	47.1	44.5	57.2	47.9	50.5 (73.8)	49.9 (67.8)
	KD E2	42.3	46.8	62.8	68.2	28.3	27.8	54.1	56.9	72.6	70.6	18.3	18.3	61.0	61.9	86.4	89.6	69.9	76.5	51.7	59.5	57.0	49.1	55.0 (80.3)	56.8 (76.8)
	KD E3	48.0	52.3	68.2	72.8	30.8	33.8	54.6	56.9	76.3	73.9	19.7	22.3	64.1	65.3	89.8	91.9	75.7	78.1	61.2	68.0	60.6	48.8	59.0 (86.2)	60.4 (82.4)
	Math RL	50.4	54.3	69.9	73.9	33.8	38.4	55.7	59.1	73.9	75.6	23.3	21.7	64.9	67.1	93.3	93.3	77.2	85.1	62.2	72.0	61.9	68.3	60.6 (89.6)	64.4 (87.8)
	Multi RL	51.4	55.0	71.5	73.5	30.8	36.4	55.8	57.7	76.2	79.5	21.3	23.7	66.6	69.6	93.5	93.9	76.8	84.0	65.7	73.3	58.2	66.6	**60.7 (88.9)**	**64.8 (88.6)**
8B	BF16	61.0	74.3	74.8	87.3	41.4	59.6	57.0	64.4	82.4	83.5	25.3	29.3	77.4	77.7	93.2	95.7	84.4	95.0	81.9	88.8	80.4	74.5	69.0 (100.0)	75.5 (100.0)
	Q4	63.4	74.0	79.1	87.3	42.9	58.6	56.1	63.1	83.0	83.2	23.7	28.0	77.0	77.0	93.0	95.7	83.0	95.1	79.3	89.2	81.6	78.9	69.3 (100.2)	75.4 (99.7)
	Bonsai	55.0	–	76.6	–	27.8	–	55.0	–	73.9	–	22.7	–	61.8	–	92.6	–	82.5	–	74.9	–	80.6	–	63.9 (91.3)	–
	CAT-Q	12.1	8.9	28.9	21.7	8.6	8.1	28.7	26.5	12.2	16.4	7.0	11.0	12.2	11.5	7.8	6.6	2.4	2.4	0.0	0.0	0.2	0.1	10.9 (18.1)	10.3 (15.7)
	KD E1	39.0	41.5	62.6	63.9	26.3	21.2	52.4	51.1	71.4	73.8	19.7	22.0	58.6	59.2	83.6	86.9	63.4	66.0	57.1	54.7	61.1	60.0	54.1 (77.6)	54.6 (71.4)
	KD E2	47.5	52.8	68.2	74.0	25.8	28.3	54.4	58.9	76.3	74.9	22.3	26.3	66.4	66.7	89.8	92.6	73.5	79.4	62.0	70.4	66.8	70.6	59.4 (85.1)	63.2 (83.1)
	KD E3	52.5	58.5	71.8	78.8	37.4	37.4	56.2	62.4	79.1	78.4	21.7	26.3	70.7	71.4	91.5	93.7	76.8	81.4	71.0	77.4	70.7	72.5	63.6 (91.6)	67.1 (88.4)
	Math RL	55.9	61.2	76.5	79.1	35.9	44.4	57.4	59.5	80.6	79.5	24.7	25.0	70.8	73.5	93.8	94.7	79.5	88.7	71.4	81.9	74.2	81.0	65.5 (94.8)	69.9 (91.6)
	Multi RL	57.4	61.7	77.4	79.7	39.4	45.0	59.4	61.1	82.1	81.7	30.7	29.0	72.5	73.7	94.0	94.8	81.1	88.4	74.5	84.8	74.3	82.9	**67.5 (99.2)**	**71.1 (94.1)**

Bonsai Prism ML (2026) checkpoints do not support think mode.
Avg./retention are computed from full-precision scores prior to rounding; may differ by ±0.1 from values derived from the rounded cells shown.

6 On-Device Inference

Ternary weights have an information-theoretic storage limit of $\log_2 3 \approx 1.585$ bits per weight, but practical deployment depends on packing, group scales, and kernel efficiency. We use the `TQ2_0` representation, which trades slightly higher storage—approximately 2 bits per weight plus metadata—for simple fixed-width packing and efficient GPU unpacking.

We benchmark materialized BF16, `Q4_0`, and ternary models through the same `llama.cpp`-based inference path on a MacBook Pro M5 Pro. We report serialized model size, peak runtime memory, and autoregressive decode throughput.

Table 4: Deployment efficiency on a MacBook Pro M5 Pro. Disk is serialized model size in GB, decode throughput is in tokens/s, and RAM is runtime memory in GB. All measurements use non-thinking decoding.

	1.7B			4B			8B		
Metric	**BF16**	**Q4**	**Ternary**	**BF16**	**Q4**	**Ternary**	**BF16**	**Q4**	**Ternary**
Disk (GB) ↓	3.79	1.19	**0.43**	7.50	2.33	**1.00**	15.26	4.68	**2.03**
Decode (tok/s) ↑	70.4	144.2	**221.2**	31.6	70.4	**105.7**	17.7	48.8	**77.9**
RAM (GB) ↓	4.13	1.54	**0.72**	7.91	2.74	**1.41**	14.27	5.10	**2.45**

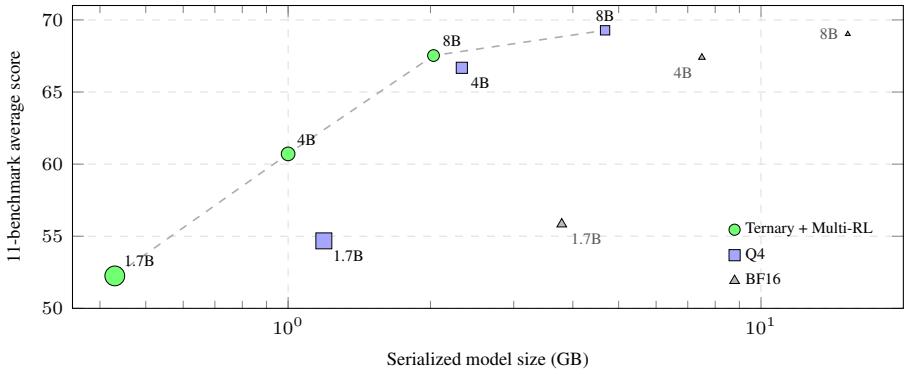


Figure 1: Size–capability frontier on a MacBook Pro M5 Pro under non-thinking decoding. Capability is the unweighted average over the 11 core benchmarks reported in Table 3. Marker area encodes measured decode throughput; ternary points use the final Multi-RL checkpoints. The dashed line denotes the size–capability Pareto frontier.

6.1 The Size–Speed–Capability Frontier

Figure 1 jointly compares non-thinking average capability, serialized size, and measured decode throughput. Across scales, ternary models shift the deployment frontier toward substantially smaller and faster operating points than BF16 or Q4.

Most notably, the recovered 8B ternary model reaches an 11-benchmark average score of 67.5 at 2.03 GB and 77.9 tokens/s, compared with 66.7 for the 4B Q4 model at 2.33 GB and 70.4 tokens/s. Thus, ternary quantization allows a larger-capacity model to occupy—and in this case improve upon—the deployment envelope of a substantially smaller low-bit model.

7 Limitations

The high retention on the 11-benchmark suite does not extend uniformly to long-horizon generation, tool use, or agentic interaction (Table 5). These are reported separately as diagnostic stress tests rather than included in Avg. or Retention. Across all three model sizes, Q4 remains close to the BF16 reference on these tasks, whereas ternary recovery is substantially less complete. This gap is especially pronounced for long thinking trajectories and multi-step interaction.

7.1 Long-Horizon Generation

The largest remaining gap appears on long-form reasoning and code generation. For the final 8B ternary model, long-form performance reaches 16.5 versus 24.8 for BF16 in non-thinking mode and 28.4 versus 66.6 in thinking mode. Our RL stages use non-thinking rollouts and do not explicitly optimize long-form trajectories, so these evaluations are stress tests rather than direct recovery targets.

The gap is consistent with trajectory-level error accumulation: longer generations provide more opportunities for an early local error to alter the subsequent autoregressive state. We also observe

Table 5: Harder diagnostic evaluations under instruct/non-thinking decoding. Long-form is the mean of AIME-25, AIME-26, and LiveCodeBench; Tool Use is the mean of BFCL3, BFCL4, ACE-Bench, and ToolSandbox; Agentic is τ^3. Full per-benchmark results in both decoding modes are reported in Appendix G.

	1.7B			4B			8B		
Diagnostic	**BF16**	**Q4**	**Ternary**	**BF16**	**Q4**	**Ternary**	**BF16**	**Q4**	**Ternary**
Long-form	14.5	12.5	5.1	24.5	23.6	12.6	24.8	24.8	16.5
Tool Use	40.7	40.6	19.2	44.7	45.2	16.2	47.4	46.6	42.0
Agentic	13.6	7.7	0.0	19.2	18.7	9.3	17.6	21.6	8.5

Ternary denotes the final Multi-RL checkpoint.

premature stopping, over-generation, and length-oriented reward exploitation. Recovering long-horizon policies therefore remains future work.

7.2 When Memory-Matched Q4 Is Better

Ternary quantization is most useful when its smaller representation permits a larger model under a fixed deployment budget. For example, the 8B ternary model reaches a non-thinking avg score of 67.5 at 2.03 GB, compared with 66.7 for 4B Q4 at 2.33 GB. However, 4B Q4 remains stronger on long-form generation, tool use, and agentic interaction. Thus Q4 remains preferable when memory is sufficient for the required model and workloads depend heavily on long-horizon or interactive behavior; ternary quantization is most advantageous when the resource budget would otherwise force a smaller model.

7.3 Architectural and Optimization Dependence

Our controlled results are limited to the Qwen3 family, and we do not claim that the exact recovery recipe transfers unchanged across architectures. Direct ternary RL requires learning rates large enough to produce discrete code movement, synchronized actor and rollout policies, complete quantizer coverage, robust reward functions, and sufficient on-policy reward variation. Architecture-specific implementation issues and Qwen3.5 experiments are discussed in the appendix.

8 Conclusion

We study extreme ternary quantization as a capability-recovery problem rather than a one-shot quantization problem. We introduce a moment-consistent ternary target and a practical calibration-free, energy-preserving ternary quantizer, then show that long knowledge-distillation recovery substantially contracts differences between initial projections in teacher-distribution space. Importantly, teacher-fidelity alone does not determine downstream behavior: models with similar KL and top-1 agreement can retain different reasoning capability. This motivates evaluating what remains behaviorally recoverable rather than treating projection error or distillation loss as the final objective.

Across Qwen3 models from 1.7B to 8B, same-size distillation restores broad post-trained capability while test-time sampling exposes substantial residual headroom. Reinforcement learning applied directly to the hard-ternary policy converts part of this headroom into reliable single-sample behavior and further improves knowledge, reasoning, instruction following, mathematics, and code. At 8B, the final model achieves 99.2% mean benchmark-wise retention in non-thinking mode and 94.1% in thinking mode. The resulting ternary models occupy only 0.43–2.03 GB and provide substantial local decoding throughput, shifting larger models into deployment regimes otherwise occupied by much smaller low-bit models.

These results do not imply that ternary quantization is lossless. Long-horizon generation, tool use, and agentic interaction remain substantially harder to recover, and conventional Q4 remains preferable for some workloads. Instead, our results suggest a broader principle for extreme model compression: *preserve a recoverable representation, restore the model distribution, measure the capabilities that remain reachable, and optimize the deployed quantized policy itself.*

AI USE STATEMENT

Generative AI tools were used to assist with software implementation, debugging and manuscript polishing. All experiments, analyses, results, and scientific claims were produced and verified by the authors. The authors reviewed all AI-assisted content and take responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

All base model checkpoints and underlying source datasets used in this work are publicly available, and the experiments do not depend on gated models, proprietary source data, or private inference APIs. The derived recovery corpus is constructed from these sources using the composition and preprocessing described in Appendix E.6. The main text specifies the quantization rule, distillation objective, evaluation protocol, and principal recovery stages. The appendix provides the complete ternary parameterization and derivation, quantizer ablations and matched-recovery trajectories, benchmark composition and decoding settings, knowledge-distillation schedules, reinforcement-learning configurations and integrity checks, negative results, and detailed per-benchmark scores. We additionally report the materialized model sizes and runtime settings used for deployment measurements.

Direct ternary optimization requires integrating the described quantization-aware bridge into the training stack so that continuous latent parameters are optimized while actor and rollout forward passes use the corresponding hard-ternary weights. We describe the required synchronization, quantization boundaries, and optimization conditions in the appendix, but do not rely on an unreleased or gated implementation of this bridge. Together with the publicly available model checkpoints and datasets, these details are intended to make the reported experiments independently reproducible.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. tau^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025. URL https://arxiv.org/abs/2506.07982.

BitCPM Team. BitCPM-CANN: Native 1.58-bit large language model training on ascend npu. Technical report, 2026. URL https://github.com/OpenBMB/MiniCPM/blob/main/docs/BitCPM_CANN.pdf.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Yuefeng Huang, Xiangcheng Liu, Xinzhi Wang, and Wu Liu. ACEBench: A comprehensive evaluation of LLM tool usage. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 12970–12998. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.697.

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. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

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

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile van Krieken, and Pasquale Minervini. Are we done with MMLU? *arXiv preprint arXiv:2406.04127*, 2024. URL https://arxiv.org/abs/2406.04127.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, et al. Multi-IF: Benchmarking LLMs on multi-turn and multilingual instructions following. *arXiv preprint arXiv:2410.15553*, 2024. URL https://arxiv.org/abs/2410.15553.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

Hong Huang, Decheng Wu, Rui Cen, Guanghua Yu, Zonghang Li, Kai Liu, Jianchen Zhu, Peng Chen, Xue Liu, and Dapeng Wu. Tequila: Trapping-free ternary quantization for large language models, 2025. URL https://arxiv.org/abs/2509.23809.

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. In *International Conference on Learning Representations*, 2025.

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 *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016. URL https://arxiv.org/abs/1605.04711.

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. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. *arXiv preprint arXiv:2408.04682*, 2024. URL https://arxiv.org/abs/2408.04682.

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. URL https://arxiv.org/abs/2402.17764.

ModelScope Team. EvalScope: Evaluation framework for large models, 2024. URL https://github.com/modelscope/evalscope.

Douglas Orr, Luka Ribar, and Carlo Luschi. Optimal formats for weight quantisation. *arXiv preprint arXiv:2505.12988*, 2025. URL https://arxiv.org/abs/2505.12988.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Prism ML. Ternary bonsai: 1.58-bit language models at 8b, 4b, and 1.7b scale. Technical report, April 2026. URL https://github.com/PrismML-Eng/Bonsai-demo/blob/main/ternary-bonsai-8b-whitepaper.pdf.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Md. Ashiq Ul Islam Sajid, Mohammad Sakib Mahmood, Md. Tareq Hasan, Md Abdur Rahim, Rafat Ara, and Md. Arafat Hossain. BitRL: Reinforcement learning with 1-bit quantized language models for resource-constrained edge deployment. *arXiv preprint arXiv:2604.24273*, 2026. URL https://arxiv.org/abs/2604.24273.

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*, 2024. URL https://arxiv.org/abs/2402.03300.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barres. tau-knowledge: Evaluating conversational agents over unstructured knowledge. *arXiv preprint arXiv:2603.04370*, 2026. URL https://arxiv.org/abs/2603.04370.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. MuSR: Testing the limits of chain-of-thought with multistep soft reasoning. In *International Conference on Learning Representations*, 2024.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. BitNet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023. URL https://arxiv.org/abs/2310.11453.

Shigeng Wang, Chao Li, Yangyuxuan Kang, Jiawei Fan, and Anbang Yao. CAT-Q: Cost-efficient and accurate ternary quantization for LLMs. *arXiv preprint arXiv:2606.26650*, 2026a. URL https://arxiv.org/abs/2606.26650.

Shigeng Wang, Chao Li, Yangyuxuan Kang, Jiawei Fan, and Anbang Yao. Attend to your own thoughts: Breaking the barrier for post-training quantization of reasoning llms through the lens of 1.58-bit quantization. *arXiv preprint arXiv:2608.01078*, 2026b.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024. URL https://arxiv.org/abs/2406.01574.

Xun Wu, Shaohan Huang, Wenhui Wang, Ting Song, Li Dong, Yan Xia, and Furu Wei. BitNet distillation. *arXiv preprint arXiv:2510.13998*, 2025. URL https://arxiv.org/abs/2510.13998.

Xianglong Yan, Chengzhu Bao, Zhiteng Li, Tianao Zhang, Kaicheng Yang, Haotong Qin, Ruobing Xie, Xingwu Sun, and Yulun Zhang. PT^2-LLM: Post-training ternarization for large language models. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7QZanjCD6M.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1_pAu9xl.

A TERNARY PARAMETERIZATION AND ENERGY DERIVATION

A.1 END-TO-END TERNARY PARAMETERIZATION

For the dense Qwen3 models studied in this work, all compute-dominant learned linear maps in the transformer are ternarized. This includes the attention projections and feed-forward projections. The tied token embedding/output matrix is also ternary, so the vocabulary interface is not retained as a higher-precision exception. Normalization parameters and other small scalar parameters account for a negligible fraction of model storage and remain in their native precision.

Transformer matrices are partitioned along the input dimension into groups of $G = 128$ weights. Each group is represented as

$$\widehat{w}_i = s_G q_i, \qquad q_i \in \{-1, 0, +1\},$$

with one continuous scale s_G per group. The projection therefore chooses both a discrete support/sign pattern and a group magnitude.

The group size determines the granularity at which this magnitude can follow the dense weight distribution. Smaller groups provide more local scales at greater metadata cost, whereas larger groups share one magnitude across a broader set of coefficients. We keep the group size fixed in matched quantizer ablations so that the projection rule is the only changing variable.

Data-free rules determine support and scale directly from dense weight statistics. AbsMean uses the local mean absolute magnitude, whereas AbsMax uses the largest absolute coefficient in the group. More expensive alternatives can instead determine support or scale from activation calibration, quantile selection, or explicit reconstruction objectives. The complete matched comparison is reported in Appendix B.

The vocabulary matrix uses a separate fixed ternary projection in all backbone-quantizer ablations. For a vocabulary group G,

$$a_G = \frac{1}{|G|} \sum_{i \in G} |w_i|,$$

and

$$q_i = \operatorname{sign}(w_i) \mathbf{1}\{|w_i| > 0.5 a_G\}, \qquad \widehat{w}_i = a_G q_i.$$

Thus, both the input embedding and language-model output projection remain ternary throughout recovery. Holding this projection fixed ensures that differences in the quantizer ablation arise from the transformer-backbone projection rather than from different treatments of the vocabulary interface.

A.2 DISTRIBUTION-FREE MOMENT MATCHING

Consider a group

$$\mathbf{w} = (w_1, \ldots, w_G)$$

and its ternary approximation

$$\widehat{w}_i = s q_i, \qquad q_i \in \{-1, 0, +1\}.$$

Define

$$m_1 = \frac{1}{G} \sum_{i=1}^{G} |w_i|, \qquad m_2 = \frac{1}{G} \sum_{i=1}^{G} w_i^2, \qquad \rho = \frac{1}{G} \sum_{i=1}^{G} q_i^2.$$

Since $q_i^2 = 1$ for a nonzero ternary coefficient and 0 otherwise, ρ is exactly the nonzero fraction of the group.

Every nonzero ternary coefficient has magnitude s. The first absolute moment and second moment of the ternary group are therefore

$$\widehat{m}_1 = \rho s, \qquad \widehat{m}_2 = \rho s^2.$$

Matching these to the corresponding dense moments gives

$$\rho s = m_1, \qquad \rho s^2 = m_2.$$

For any nonzero group, solving the two equations yields

$$\rho^\star = \frac{m_1^2}{m_2}, \qquad s^\star = \frac{m_2}{m_1}. \tag{9}$$

No assumption on the distribution of the weights is required. By Cauchy–Schwarz,

$$m_1^2 \le m_2,$$

and therefore

$$0 < \rho^\star \le 1.$$

Equivalently,

$$\rho^\star = \frac{\|\mathbf{w}\|_1^2}{G\|\mathbf{w}\|_2^2}.$$

The desired ternary density is thus determined directly by the ℓ_1/ℓ_2 geometry of the dense group.

The same solution preserves the squared ℓ_2 norm:

$$\|\widehat{\mathbf{w}}\|_2^2 = G\rho^\star(s^\star)^2 \tag{10}$$

$$= Gm_2 \tag{11}$$

$$= \|\mathbf{w}\|_2^2. \tag{12}$$

If groups partition a matrix W, preserving the second moment of every group also preserves the Frobenius norm $\|W\|_F$.

This gives a signal-propagation interpretation of the construction. For a linear transformation

$$\mathbf{y} = W\mathbf{x},$$

the expected squared output norm is

$$\mathbb{E}\left[\|W\mathbf{x}\|_2^2\right] = \operatorname{tr}\left(W\Sigma_x W^\top\right),$$

where Σ_x is the input covariance. For approximately isotropic normalized activations,

$$\Sigma_x \approx \sigma_x^2 I,$$

so

$$\mathbb{E}\left[\|W\mathbf{x}\|_2^2\right] \approx \sigma_x^2 \|W\|_F^2.$$

Preserving weight energy therefore approximately preserves the expected signal energy of the transformation. This interpretation is not required for the moment-matching derivation in Equation 9.

A.3 Gaussian Approximation to the Support

Equation 9 determines the desired nonzero density for an arbitrary weight distribution. Realizing this density exactly, however, requires selecting or ranking the corresponding fraction of coefficients in every group. We use a Gaussian model only to derive a cheap threshold approximation.

For

$$w \sim \mathcal{N}(0, \sigma^2),$$

the first absolute moment and second moment are

$$m_1 = \sigma\sqrt{\frac{2}{\pi}}, \qquad m_2 = \sigma^2.$$

Substituting into Equation 9 gives

$$\rho^\star = \frac{m_1^2}{m_2} = \frac{2}{\pi} \approx 0.6366. \tag{13}$$

Thus, under the Gaussian approximation, the moment-matched ternary representation retains approximately 63.7% of coefficients and maps approximately 36.3% to zero.

For a symmetric magnitude threshold Δ,

$$\Pr(|w| > \Delta) = 2\left[1 - \Phi\left(\frac{\Delta}{\sigma}\right)\right].$$

Setting this probability equal to $2/\pi$ gives

$$\Delta = \sigma\Phi^{-1}\left(1 - \frac{1}{\pi}\right) \approx 0.47243\,\sigma.$$

Since

$$m_1 = \sigma\sqrt{\frac{2}{\pi}},$$

we obtain

$$\Delta \approx 0.59210\, m_1 \approx 0.6\ \mathrm{AbsMean}(\mathbf{w}). \tag{14}$$

The coefficient 0.6 is therefore an inexpensive approximation to the support implied by distribution-free moment matching under Gaussian-like group weights; it is not a fitted sparsity hyperparameter.

Because real groups need not be exactly Gaussian, the practical quantizer does not assume that the realized density equals $2/\pi$. After thresholding, we measure

$$\widehat{\rho} = \frac{1}{G}\sum_{i=1}^{G} \mathbf{1}\{|w_i| > 0.6 m_1\}$$

and choose

$$s_{\mathrm{E}} = \sqrt{\frac{m_2}{\widehat{\rho}}}. \tag{15}$$

The resulting second moment is

$$\frac{1}{G}\sum_{i=1}^{G} \widehat{w}_i^2 = \widehat{\rho} s_{\mathrm{E}}^2 = m_2.$$

Thus, the practical projection preserves group energy exactly for any realized support mask. When $\widehat{\rho} = \rho^\star$, Equation 15 reduces to

$$s_{\mathrm{E}} = \frac{m_2}{m_1},$$

recovering the exact moment-matched solution.

A.4 Re-quantization Idempotency

A useful distinction between the Energy projection and the AbsMean-style vocabulary projection is their behavior when an already materialized ternary tensor is passed through the quantizer again. This occurs at training-stage boundaries and during actor/rollout materialization, where we require a baked ternary checkpoint to remain unchanged if it is reloaded through the quantization path.

Energy projection. For a nonzero group w, let

$$Q_{\mathrm{E}}(w) = sq, \qquad q_i = \mathrm{sign}(w_i)\mathbf{1}[|w_i| > \alpha m_1(w)], \qquad \alpha = 0.6,$$

with realized nonzero density

$$\rho = \frac{1}{G}\sum_{i=1}^{G} q_i^2$$

and energy-preserving scale

$$s = \sqrt{\frac{m_2(w)}{\rho}}.$$

Let $y = Q_{\mathrm{E}}(w) = sq$. Since all nonzero entries of y have magnitude s,

$$m_1(y) = \rho s, \qquad m_2(y) = \rho s^2.$$

On re-quantization, the new support threshold is

$$\alpha m_1(y) = \alpha\rho s.$$

Because $0 < \rho \leq 1$ and $0 < \alpha < 1$,

$$\alpha\rho s < s,$$

so every existing nonzero coefficient remains nonzero and every zero coefficient remains zero. The ternary code q, and therefore ρ, are unchanged. The recomputed scale is

$$s' = \sqrt{\frac{m_2(y)}{\rho}} = \sqrt{\frac{\rho s^2}{\rho}} = s.$$

Hence the materialized Energy projection is idempotent:

$$\boxed{Q_{\mathrm{E}}(Q_{\mathrm{E}}(w)) = Q_{\mathrm{E}}(w)}.$$

For an all-zero group we define $Q_{\mathrm{E}}(0) = 0$, for which the identity is immediate.

AbsMean-style vocabulary projection. The embedding/output matrices use the fixed projection

$$Q_{\mathrm{A}}(w) = aq, \qquad a = m_1(w), \qquad q_i = \mathrm{sign}(w_i)\mathbf{1}[|w_i| > \beta a], \qquad \beta = 0.5.$$

Let the realized nonzero density again be ρ, and write $y = Q_{\mathrm{A}}(w) = aq$. Then

$$m_1(y) = \rho a.$$

Reapplying the support rule leaves the ternary code unchanged, since the new threshold is

$$\beta m_1(y) = \beta\rho a < a,$$

whereas every existing nonzero coefficient has magnitude a. However, the scale is recomputed from the new mean absolute value:

$$a' = m_1(y) = \rho a.$$

Therefore

$$Q_{\mathrm{A}}(Q_{\mathrm{A}}(w)) = \rho aq = \rho Q_{\mathrm{A}}(w).$$

For $0 < \rho < 1$,

$$\boxed{Q_{\mathrm{A}}(Q_{\mathrm{A}}(w)) \neq Q_{\mathrm{A}}(w)}.$$

Repeated application preserves the support but contracts the materialized magnitude:

$$Q_{\mathrm{A}}^{k}(w) = \rho^{\,k-1} Q_{\mathrm{A}}(w), \qquad k \geq 1,$$

provided the same group support is retained.

Implication for recovery. This distinction concerns re-quantization of an already materialized ternary tensor; it does not imply that Energy codes are fixed during quantization-aware optimization. During KD and RL, the optimizer updates continuous BF16 latent weights and Q_{E} is recomputed from those changing latent parameters, so support changes remain possible. Idempotency instead guarantees that a baked Energy checkpoint is a fixed point of its own quantizer across materialization, reload, and synchronization boundaries.

This property is not needed to obtain strong KD recovery—indeed, Table 1 shows that AbsMean can reach comparable downstream accuracy after long KD. It is useful for our subsequent hard-ternary RL pipeline, where materialized actor checkpoints are repeatedly exported and synchronized with rollout workers. We therefore use the Energy projection for the trainable transformer body during RL. The already-ternarized embedding and output matrices use the AbsMean-style vocabulary projection and are frozen during RL so that their materialized values are not repeatedly re-quantized.

B Full Quantizer Recovery Ablation

B.1 Experimental Design

We compare four ternary projection rules under an otherwise identical knowledge-distillation recovery pipeline. All experiments use Qwen3-1.7B as the pretrained student initialization and a frozen Qwen3-1.7B BF16 model as teacher. The student is converted to a fully ternary model before the first optimization step.

The following quantities are held fixed across experimental arms:

- pretrained student initialization and frozen teacher;
- training corpus and curriculum order;
- maximum sequence length;
- ternary group size;
- embedding and output-head quantizer;
- distillation objective;
- optimizer and learning-rate schedule;
- global batch size and distributed topology;
- random seed;
- checkpoint cadence; and
- held-out evaluation set and evaluation implementation.

The backbone ternary projection rule is therefore the only intended experimental variable.

B.2 Backbone Ternary Projections

All attention and feed-forward weight matrices are partitioned along their input dimension into groups of $G = 128$ weights. For each group,

$$\widehat{w}_i = s_G t_i, \qquad t_i \in \{-1, 0, +1\},$$

where t_i is the ternary code and s_G is the groupwise scale.

Energy (ours). For group G, define

$$a_G = \frac{1}{|G|} \sum_{i \in G} |w_i|.$$

We use the threshold

$$\Delta_G = 0.6 a_G$$

and ternary codes

$$t_i = \operatorname{sign}(w_i) \mathbf{1}[|w_i| > \Delta_G] \,.$$

Let the realized nonzero fraction be

$$p_G = \frac{1}{|G|} \sum_{i \in G} \mathbf{1}[t_i \neq 0],$$

and define the dense group RMS

$$r_G = \sqrt{\frac{1}{|G|} \sum_{i \in G} w_i^2}.$$

The reconstruction scale is

$$s_G = \frac{r_G}{\sqrt{p_G}}.$$

Consequently,

$$\frac{1}{|G|} \sum_{i \in G} \widehat{w}_i^2 = p_G s_G^2 = \frac{1}{|G|} \sum_{i \in G} w_i^2.$$

Thus, apart from numerical clamping, Energy preserves the groupwise second moment exactly while using a simple local magnitude threshold to select the ternary support.

Energy-OPT. Energy-OPT explicitly prescribes both sparsity and reconstructed energy. Its support is obtained from the 0.45 quantile of the absolute weight magnitudes,

$$t_i = \operatorname{sign}(w_i)\mathbf{1}\left[|w_i| > Q_{0.45}(|w_G|)\right],$$

giving approximately 45% zero weights.

Let m_G denote the number of retained weights. The reconstruction scale is

$$s_G = \sqrt{\frac{0.98\sum_{i\in G} w_i^2}{m_G}}.$$

The resulting projection therefore targets

$$\frac{\|\widehat{\mathbf{w}}_G\|_2^2}{\|\mathbf{w}_G\|_2^2} \approx 0.98.$$

Unlike Energy, which lets the realized support follow directly from the weight distribution, Energy-OPT fixes an external sparsity target and requires a groupwise quantile computation.

AbsMean. We use the standard AbsMean ternary quantizer used by BitNet b1.58 (Ma et al., 2024). For each group G, let

$$a_G = \frac{1}{|G|}\sum_{i\in G}|w_i|.$$

For $a_G > 0$, the ternary codes are obtained by scaling by the mean absolute magnitude, rounding to the nearest ternary value, and clipping:

$$t_i = \operatorname{RoundClip}\left(\frac{w_i}{a_G}, -1, 1\right),$$

where

$$\operatorname{RoundClip}(x,-1,1) = \max\{-1, \min\{1, \operatorname{round}(x)\}\}.$$

The reconstructed weights are

$$\hat{w}_i = a_G t_i.$$

Equivalently, up to the measure-zero rounding boundary,

$$t_i = \operatorname{sign}(w_i)\mathbf{1}\left[|w_i| > \frac{1}{2}a_G\right].$$

An all-zero group is mapped to zero. For consistency with the other ablation arms, we apply the rule independently to the same $G = 128$ groups rather than over an entire weight matrix.

Ternary Weight Networks (TWN). We implement the thresholded ternary projection of Li et al. (2016) using the rule-of-thumb threshold proposed in the original TWN paper. For each group G, define

$$a_G = \frac{1}{|G|}\sum_{i\in G}|w_i|, \qquad \Delta_G = 0.7a_G.$$

The ternary support and signs are

$$t_i = \begin{cases} +1, & w_i > \Delta_G, \\ 0, & |w_i| \le \Delta_G, \\ -1, & w_i < -\Delta_G. \end{cases}$$

Let

$$I_G = \{i \in G : |w_i| > \Delta_G\}$$

denote the retained support. Following TWN, the reconstruction scale is the mean absolute magnitude of the retained weights,

$$\alpha_G = \frac{1}{|I_G|}\sum_{i\in I_G}|w_i|,$$

and the projected weights are

$$\hat{w}_i = \alpha_G t_i.$$

For an all-zero group we define $\hat{w}_i = 0$. As with the other ablation methods, TWN is applied independently to the same $G = 128$ groups.

B.3 VOCABULARY MATRICES

The treatment of the input embedding and LM head is held fixed across all quantizer-ablation runs. For each 128-element vocabulary-matrix group,

$$a_G = \frac{1}{|G|} \sum_{i \in G} |w_i|,$$

$$t_i = \operatorname{sign}(w_i) \mathbf{1} \left[|w_i| > 0.5 a_G \right],$$

and

$$\widehat{w}_i = a_G t_i.$$

The embedding and output matrices are therefore ternary from the beginning of recovery. Because this treatment is identical across arms, differences in the ablation are attributable to the transformer-backbone projection rule. Thus, all four ablation arms use identical group boundaries and differ only in how the ternary support and shared reconstruction scale are selected.

B.4 QUANTIZATION-AWARE OPTIMIZATION

Every quantized module maintains latent BF16 parameters w, but its effective forward weight is

$$w_{\text{eff}} = w + \operatorname{stopgrad} \left(Q(w) - w \right),$$

where $Q(w)$ is the hard ternary projection.

The forward pass therefore uses

$$w_{\text{eff}} = Q(w),$$

while the backward pass satisfies

$$\frac{\partial w_{\text{eff}}}{\partial w} = 1.$$

AdamW updates the latent BF16 parameters using gradients generated by the ternary forward model. The ternary support, signs, and groupwise scales are recomputed from the updated latent parameters on every forward pass.

Thus, all recovery occurs under the ternary constraint. None of the reported models are obtained by first training a dense model and ternarizing it after optimization.

B.5 KNOWLEDGE-DISTILLATION OBJECTIVE

All ablation arms are trained against the same frozen Qwen3-1.7B BF16 teacher using knowledge distillation only. Hard-label cross-entropy is computed for diagnostics but does not contribute to the optimized loss.

Teacher and student process the same packed sequences under teacher forcing. Only assistant-response positions contribute to the distillation loss; prompt and system tokens are used as conditioning context.

Let

$$p_T^{(i)} = \operatorname{softmax}\left(\frac{z_T^{(i)}}{\tau} \right), \qquad p_S^{(i)} = \operatorname{softmax}\left(\frac{z_S^{(i)}}{\tau} \right),$$

and

$$m_i = \frac{1}{2} \left(p_T^{(i)} + p_S^{(i)} \right).$$

The per-token Jensen–Shannon divergence is

$$D_{\text{JS}}^{(i)} = \frac{1}{2} D_{\text{KL}} \left(p_T^{(i)} \middle\| m_i \right) + \frac{1}{2} D_{\text{KL}} \left(p_S^{(i)} \middle\| m_i \right).$$

The training objective is

$$\mathcal{L}_{\text{KD}} = \frac{\tau^2}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} D_{\text{JS}}^{(i)},$$

Table 6: Optimization configuration shared by all quantizer-ablation runs.

Setting	Value
Student	Qwen3-1.7B
Teacher	Qwen3-1.7B BF16, frozen
Ternary group size	128
Maximum sequence length	8192
GPUs	$8\times$ H100
Global batch size	8 packed sequences
Gradient accumulation	1
Distributed optimizer	DeepSpeed ZeRO-2
Arithmetic	BF16
Optimizer	AdamW
Peak learning rate	2×10^{-5}
Minimum learning rate	0
Adam β values	$(0.9, 0.95)$
Adam ϵ	10^{-8}
Weight decay	0.1
Gradient clipping	0.5
Learning-rate schedule	3% warmup, cosine decay
Distillation temperature	linear $4 \rightarrow 2$
Training objective	full-vocabulary JSD
JSD token chunk	1024
Training seed	42
Checkpoint interval	20,000 steps

where $\mathcal{M}$ denotes assistant-response positions.

All $151{,}936$ vocabulary entries participate in the divergence. No teacher top-k approximation is used. For memory efficiency, the full-vocabulary calculation is evaluated in chunks of 1024 token positions with activation checkpointing; this does not alter the mathematical objective.

B.6 Training Data and Compute Matching

All methods use the same pretokenized and prepacked corpus. The curriculum gradually moves from predominantly 2K-token packs toward 4K- and 8K-token packs.

Each completed run performs

$$400{,}002$$

optimizer steps with eight packed sequences per step, for a total of

$$3{,}200{,}016$$

packed sequences.

Each was trained on $\approx$ 6B input tokens with mean token per sequence of $\approx$ 1896. The largest difference in total token count is below 0.03%, so the completed experiments are effectively compute-matched in training data.

B.7 Optimization Setup

The temperature decreases linearly with global optimizer step g:

$$\tau(g) = 4 - 2\frac{g}{400{,}002}.$$

B.8 Held-Out Evaluation

Each reported checkpoint is reconstructed using its corresponding training quantizer and evaluated after applying the hard ternary projection.

Evaluation uses a fixed split containing 512 sequences and $1{,}011{,}720$ supervised assistant tokens. We report:

Table 7: Matched recovery of four ternary quantizers on Qwen3-1.7B. Despite large differences immediately after projection, endpoint teacher fidelity contracts substantially after 400K matched recovery updates, while downstream GSM8K performance remains differently ordered. 400K GSM8K values are mean ± s.d. over three decoding seeds.

Projection	Step	MMLU-Pro ↑	GSM8K ↑	KL ↓	JSD ↓	Top-1 ↑	NLL ↓	Zero %
Energy (ours)	0	10.78	0.08	11.78	0.6899	0.21	12.74	35.53
Energy-OPT	0	10.56	0.83	**10.17**	0.6866	0.45	**11.25**	40.78
AbsMean	0	10.90	0.00	14.81	0.6757	1.30	16.08	31.49
TWN	0	10.71	**0.91**	11.65	0.6906	0.05	12.61	41.27
Energy (ours)	60K	11.88	28.66	1.4241	0.1608	66.06	4.44	38.36
Energy-OPT	60K	12.97	32.22	1.4066	0.1612	65.81	4.41	43.06
AbsMean	60K	11.68	26.91	1.4896	0.1653	65.50	4.52	34.42
TWN	60K	**14.55**	**34.42**	1.4429	0.1614	65.98	4.49	43.90
Energy (ours)	100K	**14.70**	16.68	1.2454	**0.1472**	**67.77**	4.22	38.75
Energy-OPT	100K	13.18	20.85	**1.2432**	0.1482	67.53	4.20	43.20
AbsMean	100K	11.67	23.96	1.2666	0.1517	67.06	**4.15**	34.90
TWN	100K	12.06	**34.04**	1.2811	0.1505	67.32	4.26	44.20
Energy (ours)	400K	14.38 ± 0.00	47.01 ± 0.86	0.6345	0.0901	76.09	3.48	39.32
Energy-OPT	400K	14.84 ± 0.00	41.02 ± 0.65	**0.6288**	**0.0899**	**76.11**	**3.46**	43.64
AbsMean	400K	$\mathbf{14.95 \pm 0.00}$	$\mathbf{47.81 \pm 0.39}$	0.6700	0.0935	75.61	3.51	35.64
TWN	400K	14.28 ± 0.00	43.37 ± 1.11	0.6360	0.0906	75.99	3.47	44.56

At 400K, downstream values are mean ± sample standard deviation over three decoding seeds. The zero standard deviation reported for MMLU-Pro means that aggregate accuracy was identical across seeds; it should not be interpreted as zero statistical uncertainty. Finite-example standard errors are approximately 0.32–0.33 percentage points for MMLU-Pro and 1.36–1.38 points for GSM8K. Teacher-fidelity metrics are deterministic evaluations of the same materialized checkpoint and are therefore reported once.

Table 8: Measured training cost for the energy-preserving 400K-step runs.

Method	Wall time	H100 GPU-h	Tokens/GPU/s
Energy (ours)	**50.96 h**	**407.72**	**4,134**
Energy-OPT	70.33 h	562.66	2,996

- full-vocabulary teacher-to-student KL at $T = 1$;
- full-vocabulary JSD;
- teacher/student top-1 agreement;
- student negative log-likelihood; and
- post-materialization zero fraction.

Training and evaluation objectives are intentionally distinguished: training minimizes temperature-annealed full-vocabulary JSD, whereas the primary recovery trajectory below reports teacher-to-student KL at $T = 1$ on a fixed held-out set.

B.9 Full Recovery Trajectory

The four completed methods converge to a narrow teacher-fidelity range despite substantially different initial projections and differently ordered downstream trajectories.

B.10 Training Cost

Relative to Energy, Energy-OPT requires approximately 38.0% more. At the same time, Energy-OPT improves final held-out KL over Energy by only 0.0057 (0.9% relative).

Because these long runs use a single random seed, the small endpoint differences should not be interpreted as statistically significant rankings between quantizers. Energy reaches essentially the same teacher-fidelity endpoint as Energy-OPT while requiring substantially less computation; Table 7 shows that all four projections nevertheless remain differently ordered on downstream GSM8K.

Table 9: Benchmark suite used throughout the paper. Tool-use and agentic evaluations are diagnostic and are excluded from the core recovery macro.

Capability	Benchmarks
Knowledge & reasoning	GPQA-Diamond (Rein et al., 2024), MMLU-Pro (Wang et al., 2024), MMLU-Redux (Gema et al., 2024), and MuSR (Sprague et al., 2024)
Instruction following	IFEval (Zhou et al., 2023), IFBench (Pyatkin et al., 2025), and Multi-IF (He et al., 2024)
Mathematics	GSM8K (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021), AIME 2025, and AIME 2026
Code	HumanEval+ and MBPP+ (Liu et al., 2023), and LiveCodeBench (Jain et al., 2025)
Tool use	BFCL v3/v4 (Patil et al., 2025), ACEBench (Chen et al., 2025), and ToolSandbox (Lu et al., 2024)
Agentic	τ-benchmark family across airline, retail, telecom, and banking (Yao et al., 2024; Barres et al., 2025; Shi et al., 2026)

C Evaluation Details

Harness and serving. EvalScope (ModelScope Team, 2024) handles benchmark prompts, answer extraction, and native scoring, while our wrapper controls serving, sampling, job isolation, and capability aggregation. Every checkpoint is served locally with vLLM (Kwon et al., 2023) behind an OpenAI-compatible endpoint using the model's native chat template and, where applicable, its reasoning-content and tool-call parsers. Each $(\text{model}, \text{benchmark})$ pair runs as an isolated job and records both the score and exact serving configuration.

Sampling configuration. All models are evaluated at a 32,768-token context window in both non-thinking and thinking modes. Following Qwen3's published recommendations (Yang et al., 2025), non-thinking generations use temperature 0.7, top-p 0.8, and top-k 20 with an 8,192-token generation budget; thinking generations use temperature 0.6, top-p 0.95, and top-k 20 with a 16,384-token budget. AIME 2025/2026 and MATH-500 use a 30,000-token budget so that an otherwise-correct derivation is not scored as incorrect solely because generation terminates before the final answer. These settings are identical for a recovered checkpoint and its BF16 reference.

Repeated sampling. Benchmarks with substantial single-sample variance, particularly mathematics and code, are evaluated with repeated sampling where specified in our evaluation configuration. Where test-time scaling is analyzed, we report pass@1 together with pass@k; retention comparisons always use the same statistic for the recovered and BF16 checkpoints. Other benchmarks use one sample per prompt.

Dialogue simulator. Benchmarks requiring a dialogue partner use one fixed user simulator throughout: a self-hosted, non-thinking Qwen3.8-27B sampled near-deterministically with temperature 0.001, top-p 1.0, and a 4,096-token turn budget. The same simulator and configuration are used for all model comparisons.

Capability aggregation. Each benchmark is mapped to its native 0–100 score. The primary capability average is the unweighted mean of the 11 core benchmarks used in Table 3. Retention is computed benchmark-wise relative to the corresponding same-size BF16 model and then averaged over the same 11 benchmarks, as defined in Eq. 8. AIME-25, AIME-26, LiveCodeBench, tool-use, and agentic evaluations are diagnostic and do not contribute to either Avg. or Retention.

D Teacher-Forced KD and Free-Running Retention

During distillation, the teacher and ternary student are compared at the same reference state:

$$\mathcal{L}_{\mathrm{KD}} = \sum_{t} D(p_T(\cdot \mid x, y_{<t}), p_S(\cdot \mid x, y_{<t}))\,, \tag{16}$$

where D denotes the distillation divergence used by the corresponding experiment. At inference, the student instead samples from $p_S(\cdot \mid x, \tilde{y}_{<t})$. Hence a token error at step t changes the state at which all later predictions are made.

Let $z_{(1)}$ and $z_{(2)}$ be the teacher's two largest logits and let $\gamma = z_{(1)} - z_{(2)}$. Under distillation temperature T,

$$\log \frac{p_T^{(1)}}{p_T^{(2)}} = \frac{\gamma}{T}. \tag{17}$$

Temperature therefore leaves the teacher's ordering unchanged but reduces the log-odds separating its first and second choices. For large γ, a ternary student can incur appreciable probability error without changing the argmax. For small γ, a smaller perturbation can produce a p_1/p_2 inversion. In an autoregressive rollout, such an inversion can redirect the subsequent trajectory. We use this distinction to interpret why similar teacher-forced losses can correspond to different downstream retention, without assuming that any benchmark category is intrinsically high- or low-margin.

E Knowledge Distillation Details

E.1 Student and Teacher Setup

The ternary student and BF16 teacher are initialized from the same pretrained Qwen3-8B checkpoint. The teacher is frozen and run in evaluation mode. The student is ternarized before recovery and uses the hard ternary projection in every forward pass. Transformer attention and feed-forward matrices use the ternary projection described in Section 3.2, while the vocabulary matrices use the fixed ternary projection described in Appendix A.1. RMSNorm parameters and activations remain in BF16.

For every packed sequence, teacher and student receive identical tokens under teacher forcing. Prompt and system tokens provide context, while only assistant-response positions contribute to the distillation objective.

E.2 Distillation Objective

At supervised position i, let $z_T^{(i)}$ and $z_S^{(i)}$ denote the teacher and student logits. At temperature T,

$$p_T^{(i)} = \mathrm{softmax}\left(\frac{z_T^{(i)}}{T}\right), \qquad p_S^{(i)} = \mathrm{softmax}\left(\frac{z_S^{(i)}}{T}\right). \tag{18}$$

The student minimizes the full-vocabulary forward KL objective from Equation 7. No hard-label cross-entropy term is added. The T^2 factor compensates for the reduction in gradient magnitude induced by temperature smoothing.

E.3 Training Through the Hard Ternary Projection

Each quantized module maintains latent BF16 parameters w, while the effective weight used in the forward pass is

$$w_{\text{eff}} = w + \mathrm{stopgrad}(Q(w) - w)\,, \tag{19}$$

where $Q(w)$ denotes the hard ternary projection.

The forward pass therefore uses

$$w_{\text{eff}} = Q(w),$$

while the backward pass uses

$$\frac{\partial w_{\text{eff}}}{\partial w} = 1.$$

Optimization updates the latent BF16 parameters, and the ternary codes, thresholds, occupancies, and scales are recomputed from the updated weights during training. Thus, recovery takes place under the ternary forward constraint rather than by optimizing a dense model and quantizing it afterward.

E.4 Full-Vocabulary Distillation

Qwen3 uses a vocabulary of $151{,}936$ tokens. The per-token forward KL is

$$D_{\mathrm{KL}}(p_T \| p_S) = \sum_{v=1}^{|\mathcal{V}|} p_T(v) \log \frac{p_T(v)}{p_S(v)}, \qquad |\mathcal{V}| = 151{,}936. \tag{20}$$

Early experiments instead distilled only the teacher's highest-probability tokens. Under this top-k objective, vocabulary entries outside the retained set receive no direct distillation constraint. Quantization-induced perturbations can therefore increase the relative probability of otherwise unlikely tokens. We observed this failure mode as occasional rare-token and foreign-script leakage during autoregressive generation.

Full-vocabulary distillation constrains both the teacher's preferred tokens and its suppression of the distributional tail. In our generation audit, foreign-script leakage was reduced to 0.38% under both thinking and non-thinking decoding. Relative to the earlier top-k model, full-vocabulary distillation also improved MATH-500, MMLU, and honest-answer GSM8K.

Materializing the full teacher and student logits over an 8192-token sequence produces large temporary tensors. We therefore evaluate the KL in chunks of 1024 sequence positions. Every chunk still includes all $151{,}936$ vocabulary entries. Because token-position divergences are independent before the final masked reduction, this sequence-wise chunking is mathematically equivalent to evaluating the complete sequence tensor at once.

E.5 Temperature and Learning-Rate Schedules

The distillation temperature is annealed linearly from 4 to 1 over the first two epochs and held at 1 for the final epoch:

$$T(g) = 4 - 3 \min\left(1, \frac{g}{266{,}668}\right), \tag{21}$$

where g is the global optimizer step. Thus, $T = 4$ at initialization, approximately $T = 2.5$ after the first epoch, and $T = 1$ after the second epoch.

The learning rate is linearly warmed to $\eta_{\max} = 2 \times 10^{-5}$ during the first $12{,}000$ steps and then decayed to zero with a cosine schedule. For $g \geq g_{\mathrm{warm}}$,

$$\eta(g) = \frac{\eta_{\max}}{2}\left[1 + \cos\left(\pi \frac{g - g_{\mathrm{warm}}}{g_{\mathrm{total}} - g_{\mathrm{warm}}}\right)\right], \tag{22}$$

where

$$g_{\mathrm{warm}} = 12{,}000, \qquad g_{\mathrm{total}} = 400{,}002.$$

E.6 Training Data and Sequence-Length Curriculum

Common recovery corpus. All Qwen3 recovery runs reported in this work use the same fixed, pre-tokenized training corpus rather than independently sampled mixtures for different model sizes. Qwen3-1.7B, 4B, and 8B therefore share the same post-deduplication example pool, prepacked shards, curriculum ordering, tokenizer, supervision mask, maximum sequence length, and held-out validation set. Differences between model sizes arise from distributed topology, optimization configuration, quantization staging, and the number of passes over the curriculum, rather than from dataset composition.

After cross-source deduplication, the training pool contains $6{,}881{,}627$ examples and approximately 6.069 billion tokens. A total of $4{,}698{,}755$ cross-source duplicate examples were removed, corresponding to a 40.58% removal rate. All results in this paper use the final 6.88M-example corpus.

Examples are rendered with the Qwen3 chat template, tokenized with the Qwen3 tokenizer, and stored as prepacked NPZ sequences. Loss is applied only to assistant-response tokens; prompt and system tokens remain in context but are excluded from the supervised objective. Training uses a maximum sequence length of 8192 tokens and a fixed random seed of 42.

Table 10: Post-deduplication training-data composition. Example share measures rows in the final corpus; token share better reflects optimization exposure for domains with longer sequences.

Domain	Example share	Token share
General instruction	49.55%	32.83%
Chat	25.91%	40.73%
Math	12.86%	12.36%
Safety	4.50%	2.13%
Science	2.98%	6.99%
Rewrite / summarization	2.63%	1.35%
Reasoning mixture	0.66%	0.28%
Instruction following	0.57%	0.91%
Diverse reasoning KD	0.24%	1.72%
Code reasoning	0.06%	0.39%
Agentic reasoning	0.03%	0.31%
Total	**100%**	**100%**

Table 11: Length distribution of the post-deduplication source pool before packing.

Length bucket	Example share	Token share
$\leq$2k	92.69%	70.96%
2k–4k	5.67%	18.27%
4k–8k	1.63%	10.62%
8k–16k	0.014%	0.14%

Domain composition. Table 10 reports the post-deduplication mixture. We report both example and token shares because the example distribution substantially understates exposure to longer reasoning and science examples.

Length distribution and curriculum. The underlying post-deduplication corpus is dominated by short and medium-length examples, but a substantial fraction of training tokens comes from longer sequences. Approximately 70.96% of tokens occur in examples of at most 2k tokens, 18.27% in the 2k–4k range, and 10.62% in the 4k–8k range. Only 0.14% of source tokens occur in examples above 8k tokens. Because all recovery runs use an 8192-token training limit, those longer source examples are truncated to the configured maximum length.

The materialized curriculum contains 3,126 ordered shards: 2,634 2k shards, 381 4k shards, 110 8k shards, and one 16k-labelled source shard that is still subject to the model's 8192-token training limit. The curriculum is short-to-long rather than a sequence of disjoint length blocks. The first curriculum decile contains only 2k shards; 4k shards are introduced in subsequent deciles; 8k shards are introduced from the middle of the curriculum onward; and shorter 2k shards continue to be replayed throughout. Thus, long-context exposure increases gradually without replacing the shorter data.

Source provenance. Table 12 gives the post-deduplication contribution of each source by number of examples. Source-level token counts were not retained in the final source manifest, so these percentages should not be interpreted as token-mixture weights.

Consistency across model sizes. The same prepared curriculum directory and ordered shard sequence are used for all Qwen3 model sizes in this study. Consequently, the 1.7B, 4B, and 8B recovery runs do not draw independently from a larger source pool and do not use size-specific domain mixtures. Differences in the number of optimizer steps representing one curriculum pass arise from distributed world size and sharding. A shared seed of 42 is used throughout; because worker count can differ between runs, this does not guarantee identical row-level execution order across distributed configurations, but the underlying examples and curriculum ordering are unchanged. The same held-out validation pool is also used across model sizes.

Table 12: Post-deduplication source composition by example count.

Source	Examples	Share
Infinity-Instruct	2,406,270	34.97%
Nemotron Chat v2	1,305,977	18.98%
OpenHermes 2.5	701,466	10.19%
Open Math Instruct 2	592,404	8.61%
Magpie Qwen2.5 filtered	299,946	4.36%
Tulu3 WildJailbreak	246,443	3.58%
Tulu3 WildChat	238,224	3.46%
Nemotron Science v1	204,778	2.98%
UltraChat 200K	193,154	2.81%
Tulu3 Personas Math	149,927	2.18%
Nemotron Math v2	122,790	1.78%
SmolTalk2 Summarization	90,190	1.31%
SmolTalk2 Rewriting	77,655	1.13%
SmolTalk2 Mixture of Thoughts	45,725	0.66%
Nemotron Safety v1	44,701	0.65%
SmolTalk2 Instruction Following	28,124	0.41%
SmolTalk2 System Chats	24,211	0.35%
Tulu3 Personas Algebra	20,000	0.29%
Tulu3 WildGuard	18,577	0.27%
Diverse Reasoning KD	16,635	0.24%
SmolTalk2 Table-GPT	13,140	0.19%
OASST2	12,358	0.18%
Tulu3 CoCoNot	11,331	0.16%
Tulu3 No Robots	9,384	0.14%
Code CoT	3,997	0.06%
Agentic Core	2,386	0.03%
SmolTalk2 Tulu3 Personas	1,834	0.03%
Total	**6,881,627**	**100%**

Table 13: Knowledge-distillation configuration for Qwen3-8B.

Parameter	Value
Maximum sequence length	8192
Steps per epoch	133,334
Total optimizer steps	400,002
GPUs	24× H100
Per-GPU batch size	1 packed sequence
Global batch size	24 packed sequences
Gradient accumulation	1
Peak learning rate	2×10^{-5}
Minimum learning rate	0
AdamW β values	$(0.9, 0.95)$
AdamW ϵ	10^{-8}
Weight decay	0.1
Gradient-norm limit	0.5
Warmup	3%
Training precision	BF16
Distillation vocabulary	151,936 tokens
KL sequence chunk	1024 positions
Random seed	42

E.7 Optimization and Distributed Training

Training uses three nodes with eight H100 GPUs per node. The student is distributed with DeepSpeed ZeRO-3, partitioning parameters, gradients, and optimizer state across workers. The frozen BF16 teacher is replicated on each GPU.

Student activation checkpointing is enabled, communication uses BF16, and attention uses PyTorch scaled dot-product attention. No CPU optimizer or parameter offloading is used. Gradients are clipped to a global norm of 0.5 before each optimizer update.

Table 14: Representative ternary code movement under controlled latent drift.

Checkpoint / setting	Latent drift	Codes flipped
Qwen3-1.7B ternary	10^{-3}	0.16%
Qwen3-1.7B ternary	10^{-2}	1.28%
Qwen3.5-2B, lr 10^{-5}	2.33×10^{-3}	0.0001%
Qwen3.5-2B, lr 5×10^{-5}	1.52×10^{-2}	0.0401%

E.8 Validation Behavior

Validation is performed every $13{,}333$ optimizer steps using 40 held-out batches. The temperature-scaled validation objective is 1.440 at the end of epoch 1 and 0.419 at the end of epoch 2. After the temperature reaches $T = 1$, the full-vocabulary validation KL decreases from approximately 0.420 near the beginning of epoch 3 to 0.349 by step $386{,}657$.

Losses from the first two epochs are not directly comparable because both the temperature and the T^2 scaling factor change during training. There is no discrete quantization transition at epoch boundaries because the student remains ternary throughout recovery.

F Reinforcement Learning Directly on Ternary Weights

F.1 Quantization Boundaries and Update Erasure

The transformer projections use the energy-preserving ternary quantizer from Section 3.2. For each group of 128 weights,

$$\Delta = 0.60 \ \mathrm{mean}(|w|), \qquad c_i = \mathrm{sign}(w_i)\mathbf{1}\{|w_i| > \Delta\}, \qquad s = \frac{\mathrm{rms}(w)}{\sqrt{p}}, \tag{23}$$

where p is the realized nonzero fraction and $w_{\mathrm{eff}} = s\,c$. A discrete code changes only when its latent parameter crosses a support boundary or changes sign, so flip probability is governed by the margin to those boundaries.

Table 14 gives representative measurements. These values should be interpreted as checkpoint-specific diagnostics rather than as a model-size law: the margin distribution depends on both the quantizer and the checkpoint's previous optimization history.

At learning rate 10^{-6}, the monitored Qwen3-1.7B ternary weights showed no code changes over 231 steps and the validation metric remained flat. We therefore treat learning rate as a checkpoint-dependent quantity and require nonzero code movement in a short pre-flight run before launching full RL.

F.2 Actor–Rollout Bridge and Integrity Checks

We distinguish three representations of every quantized parameter:

- **latent**: the BF16 parameter updated by the optimizer;
- **effective**: the hard-ternary value used by actor forwards;
- **baked**: the materialized ternary value exported to rollout and deployment.

Every actor forward uses the effective weight, while rollout synchronization exports the baked weight rather than the latent parameter. This keeps the rollout policy aligned with the ternary actor used for gradient computation and avoids a systematic train–rollout mismatch caused by exporting latent weights.

We enforce four integrity checks:

1. all intended two-dimensional student weights are quantized while teacher weights are not;
2. re-quantizing the materialized starting checkpoint is idempotent;

Table 15: Qwen3-1.7B direct ternary math-RL configuration.

Setting	Value
Objective	GRPO with DAPO reward manager
Reward	binary boxed-answer correctness
Learning rate	5×10^{-5}
Weight decay	0
Warmup ratio	0.02
Rollouts per prompt	8
Rollout temperature	1.0
Top-p	1.0
Train batch size	512
PPO mini-batch	128
Max prompt / response	512 / 2048 tokens
Clip ratio high	0.28
Entropy coefficient	0
KL term	none
Mode	no-think
Horizon	498 steps
Checkpoint selection	held-out greedy pass@1

3. merged checkpoints remain on the ternary grid; and
4. discrete codes actually change during training.

On the Qwen3-1.7B run, rollout/actor Pearson correlation stays near 0.9996 and $|\Delta \log \mathrm{ppl}|$ near 6×10^{-4} for the full horizon.

The implementation routes model-state exports through the baked representation. Quantizer instrumentation buffers are non-persistent so they do not appear in checkpoint or rollout state. Weight tying is restored after module conversion. When the tied embedding/output parameter is frozen, the fake-quant wrapper is removed as well so that the fixed vocabulary grid is not silently re-quantized with the transformer-body recipe.

F.3 Pre-flight Calibration and 1.7B RL Recipe

Before full training, we check quantizer coverage, re-quantization idempotence, rollout temperature, learning-rate/code movement, and verifier behavior on degenerate outputs. Difficulty is calibrated on the actual ternary checkpoint rather than inherited from another model.

For a prompt with $n = 8$ rollouts, an all-correct or all-wrong group has no within-group reward variation and therefore provides no useful GRPO advantage. We score the candidate pool using the target policy, rollout temperature, and response budget, and preferentially retain informative groups.

The final Qwen3-1.7B math-RL configuration is summarized in Table 15.

The prompt pool contains 128k candidates scored by the current policy at the same 2048-token response budget used during RL. Filtering increases the informative-group fraction substantially: for Qwen3-4B, $40.8\% \rightarrow 72.6\%$; for Qwen3-8B, $33.6\% \rightarrow 71.7\%$. This makes difficulty calibration part of the optimization procedure rather than a preprocessing convenience.

F.4 Training Diagnostics and Reward Failure

Table 16 shows the Qwen3-1.7B math-RL trajectory. The healthy pattern is rising validation accuracy, answer completion approaching one, early response-length contraction, near-zero truncation, and stable train–rollout agreement. Performance peaks before the final step but retains most of the gain.

Training reward can move in the opposite direction from capability. Under an overlength penalty, one run increased reward from 0.147 to 0.362 while held-out accuracy fell by 13 percentage points. A termination reward can similarly turn truncated-wrong outputs into terminated-wrong outputs without improving correctness.

Table 16: Qwen3-1.7B ternary math-RL trajectory.

Step	Val p@1	Has ans.	Resp. len.	Clip	Entropy	Pearson	$\lvert\Delta \log \mathrm{ppl}\rvert$
0	0.5186	0.898	731	0.074	0.340	0.9998	0.0006
40	0.5656	0.971	591	0.003	0.330	0.9998	0.0006
100	0.5910	0.986	555	0.001	0.300	0.9998	0.0007
200	0.6497	0.998	605	0.001	0.286	0.9997	0.0007
300	0.6634	0.996	674	0.001	0.260	0.9995	0.0007
420	0.7025	0.994	748	0.000	0.253	0.9996	0.0007
498	0.6947	0.998	873	0.000	0.284	0.9996	0.0007

We therefore use held-out pass@1 as the primary selection metric, with answer-completion and response-length statistics as diagnostics. Entropy, PPO KL, and train–rollout agreement are used to detect optimizer or bridge instability, not to substitute for task accuracy.

F.5 Verifier Validation and Data Failures

Every reward function is tested on degenerate outputs before training. This caught several severe failure modes.

A TriviaQA verifier using substring containment assigned near-perfect scores to one-character answers. An MBPP prompt builder omitted the required function signature for most examples, and a subset of KodCode reference solutions failed their own tests. These cases are removed or corrected before RL.

Difficulty filtering is applied only where it matches the learning objective. For judged corpora, aggressively dropping unanimous groups degraded MT-Bench, so those prompts are retained and handled by on-policy resampling instead.

F.6 Sequential RL and Stage Boundaries

Our recovery pipeline is

$$\mathrm{KD} \rightarrow \text{math RL} \rightarrow \text{multi-domain RL}.$$

At every stage boundary we:

1. merge distributed actor shards;
2. verify that the materialized checkpoint remains on the ternary grid;
3. neutralize output rows that received no training signal;
4. re-check the resulting checkpoint; and
5. recalibrate the next stage's prompt pool against the new policy.

Reusing a pool calibrated for a different checkpoint changes the fraction of informative groups and can silently alter the effective training horizon.

The math stage is necessary empirically: applying the same mixed-reward stage directly to the KD checkpoint reaches MT-Bench 5.51, compared with 6.35 when it starts from the math-RL checkpoint.

Within-stage curricula must also remain relative to the current policy. An absolute "frontier" threshold eventually collapsed into an almost shared unsolvable floor across model sizes, producing little reward variation and heavy truncation.

F.7 Qwen3.5 / Gated-DeltaNet Forensic Study

Full attention receives explicit sequence boundaries, but the recurrent GDN state initially did not; the final state of one packed sample therefore leaked into the next. On a ternary Qwen3.5-4B checkpoint this produced large relative log-probability errors on later samples in a pack.

Table 17: Selected negative results that informed the final RL recipe.

Attempt	Outcome
Dense-style lr 10^{-6} on ternary policy	No observed code movement; flat validation.
Increasing KL to fix the Qwen3.5 "cliff"	Wrong diagnosis; packed recurrent-state leakage was the main cause.
Termination/brevity reward	Improved termination but not correctness.
Larger response budgets alone	Converted some truncation to termination, not reliable accuracy gains.
Decode-time early stopping	No measurable gain in the tested setting.
Difficulty-filtering judged data	Degraded MT-Bench relative to retaining the on-policy frontier.
Untuned SFT on dense traces	Weaker greedy pass@1 than RL in the matched comparison.

Passing sequence boundaries through the recurrent kernels restored rollout/actor agreement to approximately 0.999 and changed a previously collapsing run into stable learning, improving validation from 0.5382 to 0.7397 by step 95.

This family also benefits from sequence-level rollout importance correction, a lower rollout temperature, and an explicit closure stage before math RL. These should be interpreted as architecture-specific optimization requirements, not as evidence that ternary RL is intrinsically unstable on gated-DeltaNet models.

F.8 Negative Results

Direct ternary RL required several iterations to isolate implementation and optimization failures. Table 17 shows some of our attempts and outcomes that guided the successful runs.

F.9 Reproducibility and Run Provenance

For every reported RL checkpoint we retain:

- the exact materialized input checkpoint;
- the QAT and RL configuration;
- the prompt-pool calibration summary;
- the training run identifier;
- the selected actor checkpoint; and
- the post-merge ternary grid check.

A run is considered valid only if ternary QAT is active on all intended student modules, rollout and actor policies remain aligned, and the merged deployment artifact remains on the ternary grid.

G Qwen3 Raw Results

Tables 18–20 report the individual benchmark scores underlying the Qwen3 recovery results in the main text. The first eleven benchmarks constitute the core suite used for Avg. and Retention in Table 3; AIME-25, AIME-26, LiveCodeBench, tool-use, and τ^3 are diagnostic evaluations and are excluded from those aggregates. Bonsai and Cat-Q core results are reported in Table 3; the tables below contain the matched BF16, Q4, and recovery trajectories for which the full diagnostic suite is available.

H Qwen3.5 Raw Results

Tables 21–22 report the individual benchmark scores underlying the Qwen3.5 recovery results

Table 18: Qwen3-1.7B per-benchmark scores (0–100) across BF16, Q4, KD, Math-RL, and Multi-RL checkpoints under instruct (I) and think (T) decoding. The first 11 benchmarks form the core suite used in Table 3; the remaining benchmarks are diagnostic and excluded from Avg. and Retention.

Benchmark	BF16		Q4		KD E1		KD E2		KD E3		Math RL		Multi RL	
	I	T	I	T	I	T	I	T	I	T	I	T	I	T
MMLU-Pro	45.1	57.6	43.8	55.8	22.4	21.2	28.1	31.3	32.8	37.4	36.0	41.3	38.6	42.2
MMLU-Redux	66.5	74.4	65.9	73.3	44.7	45.5	49.7	54.1	55.0	58.5	57.2	61.7	58.9	61.9
GPQA-Diamond	32.8	38.9	23.2	29.8	20.7	16.7	24.8	24.2	25.8	29.3	27.3	31.3	26.8	32.3
MuSR	50.0	57.8	52.0	56.2	46.8	45.9	46.6	48.8	48.0	52.6	50.3	54.9	49.3	55.2
IFEval	68.0	71.2	67.1	69.9	56.4	57.7	59.3	61.4	62.1	65.1	62.1	63.2	66.7	66.9
IFBench	17.3	21.3	18.0	21.7	15.0	13.0	17.0	16.7	16.3	18.0	19.4	19.7	20.7	21.0
Multi-IF	57.3	60.7	56.1	60.0	43.4	43.8	46.3	48.4	49.6	51.5	51.5	54.9	54.9	56.9
GSM8K	81.9	90.4	80.8	90.1	59.3	62.7	69.3	75.3	75.1	83.0	86.7	87.4	87.3	88.2
Math-500	73.0	90.3	71.4	89.0	37.4	33.8	47.4	48.5	57.6	61.9	66.6	73.8	68.2	74.8
HumanEval+	57.2	77.7	58.0	75.8	26.8	23.1	38.0	36.8	40.9	45.4	44.4	51.4	47.9	55.3
MBPP+	65.1	66.7	65.2	70.3	44.0	35.1	45.8	40.6	50.5	41.9	53.2	52.9	55.4	58.6
AIME-25	9.8	37.1	8.5	36.5	1.0	1.7	1.7	2.1	3.3	6.9	4.4	7.3	2.7	9.8
AIME-26	9.6	35.8	7.3	33.1	0.6	0.0	1.5	1.7	3.1	4.0	2.7	7.9	3.1	7.3
LiveCodeBench	24.1	50.7	21.7	47.3	5.2	2.7	7.6	7.3	10.1	10.3	10.2	13.3	9.4	–
BFCL3	53.1	56.0	52.9	54.2	19.7	19.7	24.4	22.0	25.2	23.5	25.8	23.4	25.9	22.1
BFCL4	28.5	32.1	30.5	29.9	11.9	12.0	14.8	12.7	14.6	13.4	15.7	13.4	14.9	13.4
ACE-Bench	40.9	52.2	39.5	51.0	0.4	0.6	13.6	15.6	19.3	25.6	16.6	22.8	16.5	22.7
ToolSandbox	40.1	51.5	39.4	51.4	28.4	25.3	21.1	21.1	19.3	20.7	18.2	19.8	19.3	21.2
τ^3	13.6	9.9	7.7	10.9	4.0	6.1	1.3	1.9	0.0	1.1	0.5	0.3	0.0	0.8

Table 19: Qwen3-4B per-benchmark scores (0–100) across BF16, Q4, KD, Math-RL, and Multi-RL checkpoints under instruct (I) and think (T) decoding. Core and diagnostic benchmarks follow the definitions in Table 18.

Benchmark	BF16		Q4		KD E1		KD E2		KD E3		Math RL		Multi RL	
	I	T	I	T	I	T	I	T	I	T	I	T	I	T
MMLU-Pro	61.8	69.8	60.2	68.9	34.6	36.4	42.3	46.8	48.0	52.3	50.4	54.3	51.4	55.0
MMLU-Redux	79.4	83.8	78.4	83.5	56.9	61.0	62.8	68.2	68.2	72.8	69.9	73.9	71.5	73.5
GPQA-Diamond	40.4	53.5	41.9	49.0	26.3	25.2	28.3	27.8	30.8	33.8	33.8	38.4	30.8	36.4
MuSR	54.9	60.6	58.2	59.7	51.4	53.0	54.1	56.9	54.6	56.9	55.7	59.1	55.8	57.7
IFEval	79.5	81.9	78.6	80.0	68.6	66.0	72.6	70.6	76.3	73.9	73.9	75.6	76.2	79.5
IFBench	26.0	26.7	24.0	26.3	17.0	16.3	18.3	18.3	19.7	22.3	23.3	21.7	21.3	23.7
Multi-IF	72.2	73.7	70.6	72.2	56.7	57.2	61.0	61.9	64.1	65.3	64.9	67.1	66.6	69.6
GSM8K	91.8	95.0	91.1	94.3	80.7	82.8	86.4	89.6	89.8	91.9	93.3	93.3	93.5	93.9
Math-500	84.1	95.1	82.5	93.8	58.5	58.5	69.9	76.5	75.7	78.1	77.2	85.1	76.8	84.0
HumanEval+	74.9	88.9	71.2	87.0	47.1	44.5	51.7	59.5	61.2	68.0	62.2	72.0	65.7	73.3
MBPP+	76.4	70.4	76.8	76.7	57.2	47.9	57.0	49.1	60.6	48.8	61.9	68.3	58.2	66.6
AIME-25	21.5	63.5	20.6	61.5	2.3	2.9	7.5	15.0	13.5	17.7	11.0	19.6	10.0	19.4
AIME-26	16.5	62.1	15.4	62.5	3.3	4.0	8.5	10.4	10.0	15.4	6.7	18.3	6.2	14.4
LiveCodeBench	35.4	66.6	34.9	63.6	11.1	10.5	15.1	17.4	18.5	22.6	17.6	24.4	21.5	28.7
BFCL3	56.8	61.3	56.8	60.7	19.7	19.7	29.8	32.3	20.7	20.6	20.4	20.4	20.7	20.6
BFCL4	31.9	41.8	30.5	41.0	11.4	11.4	20.4	20.6	11.8	11.3	11.8	12.9	11.2	12.3
ACE-Bench	42.2	66.2	47.1	62.1	2.0	5.2	24.0	32.5	4.7	15.9	3.5	15.2	4.1	14.3
ToolSandbox	47.8	59.8	46.4	61.3	28.6	28.1	23.7	27.3	28.8	28.4	16.7	28.8	28.9	28.7
τ^3	19.2	26.4	18.7	23.5	6.1	3.5	5.9	7.7	6.9	8.3	8.3	7.7	9.3	9.1

Table 20: Qwen3-8B per-benchmark scores (0–100) across BF16, Q4, KD, Math-RL, and Multi-RL checkpoints under instruct (I) and think (T) decoding. Core and diagnostic benchmarks follow the definitions in Table 18.

Benchmark	**BF16**		**Q4**		**KD E1**		**KD E2**		**KD E3**		**Math RL**		**Multi RL**	
	I	T	I	T	I	T	I	T	I	T	I	T	I	T
MMLU-Pro	61.0	74.3	63.4	74.0	39.0	41.5	47.5	52.8	52.5	58.5	55.9	61.2	57.4	61.7
MMLU-Redux	74.8	87.3	79.1	87.3	62.6	63.9	68.2	74.0	71.8	78.8	76.5	79.1	77.4	79.7
GPQA-Diamond	41.4	59.6	42.9	58.6	26.3	21.2	25.8	28.3	37.4	37.4	35.9	44.4	39.4	45.0
MuSR	57.0	64.4	56.1	63.1	52.4	51.1	54.4	58.9	56.2	62.4	57.4	59.5	59.4	61.1
IFEval	82.4	83.5	83.0	83.2	71.4	73.8	76.3	74.9	79.1	78.4	80.6	79.5	82.1	81.7
IFBench	25.3	29.3	23.7	28.0	19.7	22.0	22.3	26.3	21.7	26.3	24.7	25.0	30.7	29.0
Multi-IF	77.4	77.7	77.0	77.0	58.6	59.2	66.4	66.7	70.7	71.4	70.8	73.5	72.5	73.7
GSM8K	93.2	95.7	93.0	95.7	83.6	86.9	89.8	92.6	91.5	93.7	93.8	94.7	94.0	94.8
Math-500	84.4	95.0	83.0	95.1	63.4	66.0	73.5	79.4	76.8	81.4	79.5	88.7	81.1	88.4
HumanEval+	81.9	88.8	79.3	89.2	57.1	54.7	62.0	70.4	71.0	77.4	71.4	81.9	74.5	84.8
MBPP+	80.4	74.5	81.6	78.9	61.1	60.0	66.8	70.6	70.7	72.5	74.2	81.0	74.3	82.9
AIME-25	20.6	66.5	21.5	66.9	7.3	9.0	12.1	17.7	14.6	17.9	9.2	25.0	13.1	26.9
AIME-26	15.4	65.4	16.9	65.6	7.9	8.8	9.8	13.5	10.6	21.5	8.5	20.0	10.0	22.3
LiveCodeBench	38.4	67.9	36.1	66.6	12.2	14.4	18.8	24.2	21.2	29.0	23.7	34.8	26.3	36.1
BFCL3	58.2	63.0	58.1	62.1	23.5	27.0	51.4	52.4	51.4	55.0	50.9	53.5	51.5	53.4
BFCL4	35.8	40.8	36.1	40.5	13.3	14.7	26.9	29.3	29.6	31.0	30.8	31.9	31.1	31.2
ACE-Bench	46.7	71.2	43.8	69.8	12.6	20.8	34.1	52.0	37.6	60.2	37.1	57.4	35.7	59.0
ToolSandbox	48.7	64.3	48.2	64.2	28.2	29.5	38.8	48.8	45.4	54.5	48.7	56.0	49.8	56.1
τ^3	17.6	28.5	21.6	30.1	5.9	4.8	6.4	7.5	10.1	9.6	9.1	9.9	8.5	10.4

Table 21: Qwen3.5 4B: per-benchmark scores (0–100 scale) across the BF16, Q4, KD E3, Math RL and Multi RL stages, for instruct (I) and think (T) decoding.

Benchmark	**BF16**		**Q4**		**KD E3**		**Math RL**		**Multi RL**	
	I	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**
BFCL3	**61.7**	**63.9**	60.9	63.6	23.1	29.5	23.0	30.0	24.1	32.2
BFCL4	**52.0**	**55.2**	49.7	54.4	13.5	16.8	12.5	18.0	11.9	18.3
τ^3	62.4	57.3	**62.7**	**57.6**	3.5	8.0	1.1	3.7	0.8	2.4
ACE-Bench	58.9	**74.0**	**60.0**	72.8	19.3	29.6	18.5	30.9	16.9	32.4
ToolSB	**56.7**	**60.8**	56.5	56.3	18.1	11.6	16.7	15.0	18.0	15.3
GPQA	**75.8**	77.8	70.7	**79.3**	33.8	36.4	37.9	42.9	38.9	43.9
MMLU-P	**77.7**	**79.5**	77.1	79.1	-	-	55.5	57.7	56.6	58.3
MMLU-R	**87.3**	88.6	86.7	**88.8**	74.0	76.4	74.6	76.2	74.6	77.3
MuSR	**60.1**	63.4	59.7	**63.8**	57.7	58.7	58.7	61.6	59.4	59.7
IFEval	**80.8**	**89.3**	**80.8**	87.1	75.2	76.9	75.4	74.5	77.5	77.6
IFBench	**27.7**	**53.0**	27.0	52.0	25.3	32.7	26.7	29.3	22.7	30.3
Multi-IF	68.7	78.6	**70.9**	**79.7**	62.3	63.1	62.7	62.5	65.5	65.1
GSM8K	**94.1**	**95.7**	94.0	95.2	91.2	92.6	90.9	92.6	91.7	92.5
Math-500	94.9	**98.0**	**95.0**	97.9	-	-	79.6	80.9	79.3	80.7
AIME-25	46.9	**80.6**	**50.0**	75.6	-	-	13.3	17.3	14.8	16.2
AIME-26	55.2	**82.1**	**57.9**	81.5	-	23.3	12.9	12.9	12.5	12.7
HE+	**78.3**	82.5	77.1	**84.2**	61.0	67.1	62.7	65.8	66.8	71.5
MBPP+	**83.7**	83.3	80.7	**83.7**	68.5	66.9	69.8	69.8	72.1	74.1
LCB	**54.1**	**68.2**	52.6	64.3	17.3	21.8	18.1	21.1	20.9	23.4
Avg	**67.2**	**75.4**	66.8	74.6	42.9	44.5	42.7	45.4	43.4	46.5

Missing ("-") benchmark cells are due to runaway/repetitive generations at that checkpoint.

Table 22: Qwen3.5 9B: per-benchmark scores (0–100 scale) across the BF16, Q4, KD E3, Math RL and Multi RL stages, for instruct (I) and think (T) decoding.

Benchmark	**BF16**		**Q4**		**KD E3**		**Math RL**		**Multi RL**	
	I	**T**	**I**	**T**	**I**	**T**	**I**	**T**	**I**	**T**
BFCL3	**62.6**	**63.5**	61.4	63.4	33.5	36.9	33.9	36.9	35.9	39.5
BFCL4	**55.4**	**60.7**	55.2	60.5	16.9	21.6	16.3	20.8	17.9	22.5
τ^3	**63.2**	60.0	60.0	**60.8**	4.3	6.9	4.5	9.3	5.9	7.7
ACE-Bench	**74.9**	**75.5**	71.4	72.6	27.5	45.7	26.7	43.2	28.7	45.2
ToolSB	61.0	48.8	**62.9**	**53.6**	21.0	21.9	22.9	22.1	25.7	25.9
GPQA	**82.8**	**82.8**	78.8	81.3	43.9	46.5	51.5	46.5	50.0	52.0
MMLU-P	**81.0**	**82.9**	80.5	82.2	64.8	-	63.2	65.8	64.5	66.8
MMLU-R	89.9	**91.3**	**90.0**	91.1	81.3	82.2	80.2	82.1	80.8	82.7
MuSR	**65.0**	**66.7**	64.1	64.8	63.1	64.3	58.9	63.2	59.7	62.2
IFEval	83.9	**89.5**	**85.4**	88.3	80.4	80.8	78.6	79.8	83.4	79.8
IFBench	33.3	53.0	**37.0**	**54.3**	26.7	38.3	26.7	33.0	31.3	34.8
Multi-IF	74.5	81.8	**75.1**	**83.3**	68.6	71.5	68.4	67.9	71.5	70.5
GSM8K	**95.5**	**96.2**	95.2	**96.2**	93.9	94.8	93.2	94.0	93.5	94.8
Math-500	96.4	**98.7**	**96.7**	98.6	87.6	92.0	84.9	85.7	86.2	87.4
AIME-25	57.1	**84.0**	**59.6**	83.5	30.2	39.6	20.2	22.1	21.9	24.6
AIME-26	**65.6**	**87.5**	64.8	86.0	31.0	43.3	16.9	20.0	19.8	22.1
HE+	**83.5**	**88.4**	82.4	85.9	73.1	78.9	72.6	75.9	79.1	80.4
MBPP+	**85.9**	**87.4**	85.5	86.7	77.8	75.4	78.0	77.4	80.8	81.3
LCB	**62.3**	**73.0**	59.0	69.1	26.6	30.3	25.0	27.8	29.4	32.7
Avg	**72.3**	**77.5**	71.8	77.0	50.1	53.9	48.6	51.2	50.8	53.3

Missing ("-") benchmark cells are due to runaway/repetitive generations at that checkpoint.