OPQ: ON-POLICY QUANTIZATION FOR LOW-BIT LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-bit quantization reduces the memory required to deploy large language models. However, learning on fixed sequences can miss the contexts reached by a quantized model's own predictions. We propose on-policy quantization (OPQ), a framework that learns directly on student-generated trajectories. On-policy quantization faces two main challenges: correcting output discrepancies introduced by low precision and maintaining consistent internal attention representations. To address the first, Dynamic Kullback–Leibler (KL) divergence adjusts output supervision using floating and quantized views of the student. To address the second, key/value (K/V) alignment supervises their attention memory. Our theoretical analysis explains the corrective gradients of the output objective and how K/V discrepancies affect attention outputs. With 2-bit and 3-bit weights, OPQ achieves the highest average task accuracy among the evaluated low-bit methods across two model families. Its gains are largest at 2-bit precision, reaching 9.49 percentage points over the strongest baseline. Ablations further show how student trajectories, Dynamic KL and K/V alignment contribute to capability preservation.

1 INTRODUCTION

Large language models (LLMs) acquire broad capabilities, including language understanding, mathematics, reasoning and code generation, through extensive pretraining and post-training. However, their large parameter counts make deployment difficult in environments with limited memory. Reducing the memory required to store and run these models, while retaining their capabilities, is therefore a practical challenge.

Quantization is a common approach to this challenge, representing model weights with fewer bits. Existing quantization approaches include two main families, as illustrated in Figure 1. First, *calibration-based post-training quantization* (PTQ) uses activation statistics or reconstruction objectives to choose low precision weights (Frantar et al., 2023; Lin et al., 2024; Shao et al., 2024). Second, *distillation-based quantization* updates a quantized student to match a high precision teacher on training sequences (Liu et al., 2024; Du et al., 2024). However, the representative calibration and fixed-response distillation methods optimize on inputs that are not refreshed to follow the quantized student's own generation trajectories. At deployment, each predicted token becomes part of the next context. Quantization can therefore move the student to prefixes that these fixed inputs do not cover, leaving errors on later predictions insufficiently corrected.

To address this challenge, we propose to learn quantized weights on the quantized model's own trajectories, which we call *on-policy quantization* (OPQ). The student generates responses, and a frozen high precision teacher provides conditional targets on the same prefixes (Agarwal et al., 2024; Gu et al., 2024). As the student evolves, fresh responses keep training connected to the contexts reached by its low precision computation. However, this approach still faces **two main challenges**.

(1) **Output discrepancies.** Disagreement with the teacher can arise from both parameter updates and quantization. A fixed Kullback–Leibler (KL) divergence objective does not explicitly distinguish these sources. Consequently, it may not respond appropriately when quantization removes support for alternatives preferred by the teacher.

(2) **Latent consistency.** Output matching does not directly constrain representations at different precisions. In attention, keys determine which prefix positions later queries read, while values

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

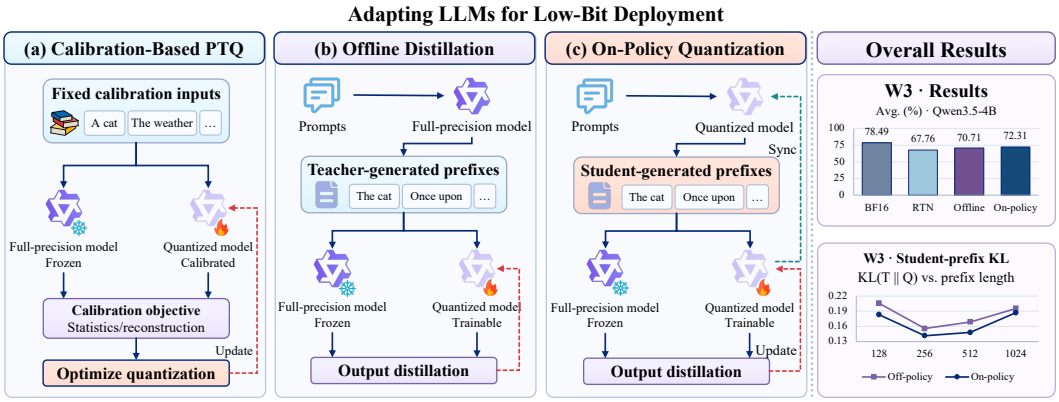


Figure 1: **Low-bit quantization.** Input sources for calibration, distillation and on-policy learning. Right: 4B results with 3-bit weights (W3), showing task averages (top) and student-prefix KL (bottom).

determine the retrieved content. Matching the next-token distribution at one prefix does not require these stored representations to be identical. Later tokens introduce new attention queries, which can expose differences in the stored keys and values even when the predictions are similar.

To address challenge (1), we introduce *Dynamic KL*. A floating view of the student identifies changes in the teacher probability mass covered by its leading predictions after quantization. When that mass decreases, the objective places more emphasis on forward KL to restore support for those alternatives. To address challenge (2), we add *key/value (K/V) alignment* against the same floating student. This provides direct supervision for attention representations in addition to output matching.

Finally, we introduce OPQ, a framework for preserving the capabilities of high precision LLMs during low-bit deployment. It combines on-policy trajectories, Dynamic KL and K/V alignment within a common training process. We update dense student weights through asymmetric integer quantization and derive the final packed weights from the trained model. The teacher and floating reference are needed only during training, leaving a single low-bit model for inference.

In summary, our contributions are as follows.

- **A systematic study of on-policy quantization.** We investigate how student trajectories and internal precision discrepancies shape capability preservation in low-bit models.
- **The OPQ framework.** We bring on-policy learning, precision-aware output supervision and latent alignment into a unified framework for low-bit deployment.
- **Experiments and theoretical analysis.** We connect evaluations across two model architectures and two bit widths with component comparisons and an analysis of attention error propagation.

2 Related Work

Low-bit quantization. Two common families address the loss of accuracy caused by low precision weights. First, *post-training quantization* reduces approximation error using a calibration set without updating the model through a full distillation objective. GPTQ uses approximate second-order information, AWQ scales channels using activations, and OmniQuant learns clipping and equivalent transformations (Frantar et al., 2023; Lin et al., 2024; Shao et al., 2024). Second, *quantization with distillation* updates the student under quantized computation using teacher supervision. LLM-QAT uses synthetic sequences and output distributions, while BitDistiller combines asymmetric clipping with confidence-weighted distillation (Liu et al., 2024; Du et al., 2024). Reasoning-oriented quantization-aware training (QAT) and quantization-aware distillation (QAD) with NVIDIA's 4-bit floating-point format further study accuracy recovery through teacher guidance (Okoshi et al., 2026; Xin et al., 2026). These approaches establish how calibration and learning can reduce quantization

error. In contrast to fixed-sequence training, we refresh supervision on the quantized student's own trajectories and adapt output learning to the discrepancies introduced by low precision.

On-policy learning. On-policy learning collects trajectories from the model and uses them to guide subsequent updates. In reinforcement learning, methods such as proximal policy optimization alternate trajectory collection with updates that improve a reward objective (Schulman et al., 2017). In on-policy distillation, a teacher instead supplies conditional supervision on student-generated sequences. Generalized knowledge distillation (GKD) studies divergence objectives on these sequences, while MiniLLM develops reverse-KL distillation (Agarwal et al., 2024; Gu et al., 2024). Besides general distillation, public NeMo implementations support quantized training and rollout for both reinforcement learning and on-policy distillation (NVIDIA, n.d.). Our focus is capability preservation when the student must be deployed at low precision, rather than reward optimization or compression to a smaller architecture. To this end, we use the student's quantized trajectories and distinguish parameter adaptation from precision loss when constructing its output objective.

3 Problem Statement and Terminology

Given a pretrained high precision model, our goal is to learn a low-bit model that retains its generation capabilities. We call the frozen original model the *teacher* T and denote the student's dense weights by W. These weights define a *quantized student* Q_W and a *floating view* F_W, which evaluates the same weights without quantization. We write Q and F when the dependence on W is clear. Unlike T, this view changes as the student is updated. A *prefix* consists of the input prompt and response tokens preceding the next prediction. Off-policy distillation uses fixed teacher-generated responses, whereas on-policy distillation refreshes responses using the student. Both can match conditional distributions on these prefixes using reverse Kullback–Leibler divergence (RKL), $\mathrm{KL}(Q\|T)$. We distinguish this training loss from the evaluation metric $\mathrm{KL}(T\|Q)$, which measures how well the student preserves the teacher's next-token distribution.

4 Empirical Motivation

In this section, we examine **two questions** using Qwen3.5-4B W3. First, does learning on student trajectories improve output fidelity on prefixes reached by a quantized model? Second, does similar output fidelity imply consistent internal representations? To answer these questions, we compare the two RKL training schemes defined above and an on-policy variant with an additional K/V alignment loss. The compared methods share the asymmetric quantizer, initialization and training budget. Figure 2 connects conditional fidelity and internal alignment with downstream accuracy.

4.1 Off-policy vs On-policy

To isolate trajectory choice, we compare off-policy and on-policy RKL with the same frozen high precision teacher. With fixed teacher responses, the student learns to predict under contexts produced by a different model. At inference, however, its own predictions determine the next context. Even a small change in token choice can introduce a prefix absent from the fixed training sequences, and later predictions are then evaluated on a different input distribution. On-policy training addresses this mismatch by refreshing the responses as the quantized student changes.

To compare the resulting conditional fidelity, both models read identical tokens from a fixed set of sequences generated by the initial quantized model. We evaluate full-vocabulary $\mathrm{KL}(T\|Q)$ at matched prefix lengths. Figure 2(a) displays lengths 128 through 1024. The accompanying data include measurements through 2048 tokens, valid-prompt counts and the teacher-prefix comparison.

Figure 2(a) shows lower KL for on-policy RKL at every measured length. Refreshing student trajectories therefore improves fidelity on contexts reached by low-bit generation.

4.2 Output Fidelity vs Latent Consistency

Quantizing the student's weights changes the hidden activations from which attention keys and values are computed. Consequently, the low and high precision views of the same student can store different

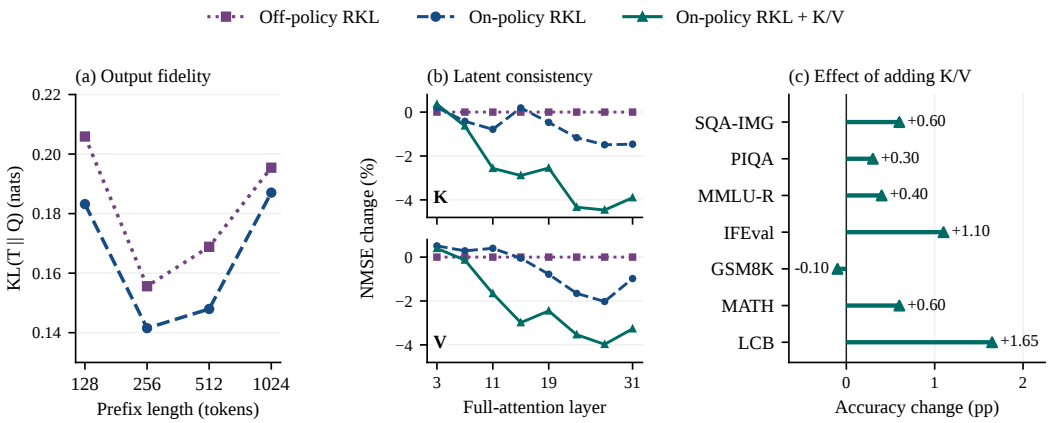


Figure 2: **Qwen3.5-4B W3 motivation.** (a) Teacher KL on initial-student prefixes. (b) K/V NMSE relative to off-policy RKL. (c) Accuracy change from adding K/V to on-policy RKL.

K/V representations for the same tokens. Output matching constrains the final vocabulary distribution, but it does not uniquely determine these intermediate representations. In particular, a key discrepancy can change which prefix positions a later query attends to, while a value discrepancy changes the information retrieved. Thus, *output fidelity and latent consistency are different properties*.

To examine latent consistency, we measure normalized mean squared error (NMSE) between each final quantized student's K/V representations and those of its own floating master on the student's generated prefixes. The comparison uses 23 shared prompt IDs and matched lengths, with each student's own continuation. Figure 2(b) separates keys and values so that their changes can be inspected independently. On-policy RKL alone produces a small reduction in K/V error, while explicit K/V supervision improves the alignment of both keys and values. Meanwhile, the two on-policy variants have similar output fidelity. These observations motivate direct internal supervision beyond output matching.

Finally, Figure 2(c) connects this internal alignment to downstream performance. Adding K/V supervision improves average accuracy, with the clearest gains in instruction following and code generation. On-policy RKL also outperforms off-policy RKL before K/V supervision is added, supporting the trajectory choice independently. Together, these results support K/V alignment as a complementary signal alongside on-policy output learning.

5 Methodology

In this section, we describe OPQ in four steps. We first align output distributions on student trajectories, then align K/V representations, introduce Dynamic KL, and finally explain quantization and export. Following the problem setup, Q_W and F_W share the dense weights W. The frozen teacher T provides output targets, while F_W provides the reference for precision differences. Figure 3 summarizes this training process.

5.1 On-Policy Output Alignment

To keep training connected to the deployed model, we generate $y \sim Q_W(\cdot \mid x)$ for a prompt x using quantized weights. At response position i, all three views read the same prefix $h_i = (x, y_{<i})$. Rollout weights are refreshed after each learner update, so subsequent batches follow the updated low-bit model. Response tokens are fixed during backpropagation, so we do not differentiate through sampling.

To align the output distributions, write $T_i = T(\cdot \mid h_i)$ and similarly define Q_i, F_i. We use full-vocabulary reverse KL as the basic conditional objective,

$$R_i = \mathrm{KL}(Q_i \| \,\mathrm{sg}(T_i)), \qquad \mathcal{L}_{\mathrm{RKL}} = \mathrm{mean}_{i \in \mathcal{R}}\, R_i, \tag{1}$$

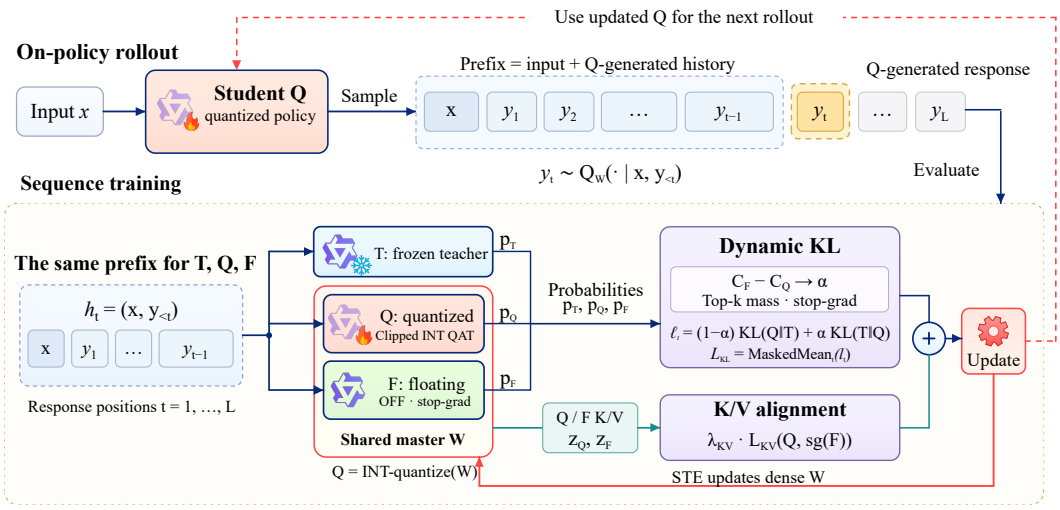


Figure 3: **OPQ training.** Student-generated responses support Dynamic KL and K/V alignment. STE updates the dense weights, from which the deployment quantization state is derived.

where $\mathcal{R}$ contains valid response predictions and sg stops gradients. RKL transfers the teacher's conditional knowledge on alternatives favored by the student (Gu et al., 2024). Thus, output supervision follows the contexts reached by low-bit generation. Appendix C.1 gives the sequence-KL interpretation of this conditional update.

5.2 K/V Alignment

However, output alignment does not directly constrain attention memory. We therefore also align K/V representations between Q_W and F_W. For prediction i, let $Z^Q_{i,\ell,h}, Z^F_{i,\ell,h}$ be key or value matrices on its visible prefix at layer ℓ and head h. For prefix length n_i, define

$$e^Z_{i,\ell,h} = \frac{\|Z^Q_{i,\ell,h} - \mathrm{sg}(Z^F_{i,\ell,h})\|_F^2}{n_i\epsilon + \|\mathrm{sg}(Z^F_{i,\ell,h})\|_F^2}, \qquad L_{\mathrm{KV}} = \mathrm{mean}_{i,\ell,h,Z\in\{K,V\}}\, e^Z_{i,\ell,h}, \tag{2}$$

where $\epsilon > 0$ stabilizes normalization. Prompt and preceding response tokens contribute whenever visible to a valid prediction, while future tokens and padding are excluded. The shared master makes this a direct comparison between precisions. Keys are captured after key normalization and before rotary position encoding, and values use the corresponding attention projection.

K/V discrepancies affect an attention read in two ways. First, keys alter attention probabilities. Second, values alter the content being aggregated. Combining these effects gives the following local bound.

Proposition 1 (K/V errors for a shared query)**.** *Consider a common visible prefix and query q with $\|q\|_2 \le R$. Set $\varepsilon_K = \|K^Q - K^F\|_F$ and $\varepsilon_V = \max_j \|v^Q_j - v^F_j\|_2$, and assume $\max_j \|v^F_j\|_2 \le M$. For $o^A(q) = \mathrm{softmax}(K^A q/\sqrt{d})^\top V^A$,*

$$\|o^Q(q) - o^F(q)\|_2 \le \varepsilon_V + \frac{RM}{\sqrt{d}}\varepsilon_K. \tag{3}$$

The detailed proof and discussion of differing queries are provided in Appendix C.

5.3 Dynamic KL

To adapt output supervision to precision changes, we compare the teacher probability mass covered by the two student views' top-k predictions. Let $S^Q_i = \mathrm{Top}_k(\mathrm{sg}(Q_i))$ and $S^F_i = \mathrm{Top}_k(\mathrm{sg}(F_i))$

Algorithm 1 On-policy quantization

Require: Teacher T, initial dense weights W, calibration inputs, training prompts
Require: Bit width B, group size g, updates N, top-set size k, τ, λ_{KV}
1: Search asymmetric clipping bounds and initialize W once
2: **for** $t = 1, \ldots, N$ **do**
3: Derive (s_t, z_t, q_t) from W using equation 8; refresh rollout
4: Sample responses $y \sim Q_W(\cdot \mid x)$ and detach token IDs
5: Evaluate T, Q_W and detached F_W on each prefix $(x, y_{<i})$
6: Compute R_i, D_i, detached α_i from top-k coverage, and L_{KV}
7: Update W through STE using $\mathrm{mean}_{i \in \mathcal{R}}[(1-\alpha_i)R_i + \alpha_i D_i] + \lambda_{\mathrm{KV}} L_{\mathrm{KV}}$
8: **end for**
9: Recompute and pack (q, z, s) from the final W for deployment

denote their highest-probability token sets. Define

$$C_i^Q = \sum_{v \in S_i^Q} T_i(v), \quad C_i^F = \sum_{v \in S_i^F} T_i(v), \quad \alpha_i = \mathrm{sg}\left[\mathrm{clip}\left(\frac{[C_i^F - C_i^Q]_+}{\tau}, 0, 1\right)\right]. \tag{4}$$

Here $\tau > 0$ controls how strongly a coverage difference changes the mixture. A larger α_i indicates that quantization has reduced teacher mass in the student's leading alternatives. The top-k comparison sets this coefficient, while output alignment uses the full vocabulary.

To restore support for teacher-preferred alternatives, introduce forward KL, $D_i = \mathrm{KL}(\mathrm{sg}(T_i) \| Q_i)$, and form $B_i = (1-\alpha_i)R_i + \alpha_i D_i$. The following lemma explains its corrective effect on underrepresented tokens.

Lemma 1 (Non-vanishing correction for underrepresented tokens)**.** *Fix a prediction prefix and a teacher distribution T_i with $T_i(v) > 0$ on a finite vocabulary. Let $Q_i = \mathrm{softmax}(u_i)$, where u_i denotes the student logits. Treat T_i and $\alpha_i \in [0,1]$ as constants during this update. For any token with $Q_i(v) \le T_i(v)$,*

$$-\frac{\partial B_i}{\partial u_i(v)} \ge \alpha_i \big(T_i(v) - Q_i(v)\big). \tag{5}$$

For fixed T_i and fixed $\alpha_i > 0$, as $Q_i(v) \to 0$,

$$\frac{\partial B_i}{\partial u_i(v)} \to -\alpha_i T_i(v), \qquad \frac{\partial R_i}{\partial u_i(v)} \to 0. \tag{6}$$

Reverse KL provides a diminishing corrective logit gradient for teacher-supported tokens whose student probabilities approach zero. When the precision comparison detects a coverage deficit, Dynamic KL increases the forward KL contribution and preserves this corrective signal. More details can be found in Appendix C.2.

Finally, combining the output and representation terms gives

$$\mathcal{L} = \mathrm{mean}_{i \in \mathcal{R}}\, B_i + \lambda_{\mathrm{KV}} L_{\mathrm{KV}}, \tag{7}$$

where the hyperparameter λ_{KV} is constant throughout training. Teacher predictions, floating targets and α_i are detached in each update and refreshed on later batches. This allows output correction to respond to precision differences while maintaining a stable representation objective.

5.4 Quantization and Optimization

We follow a standard groupwise asymmetric quantization approach with activation-based clipping (Du et al., 2024). Before training, positive and negative clipping bounds are searched using cached activations to reduce layer-output reconstruction error. Clipping initializes the dense master once. We use 2-bit integers (INT2) with group size 32 and 3-bit integers (INT3) with group size 128. Both precisions use uniformly spaced integer codes so that the final weights can be packed in the supported deployment format.

For a group of weights w, let $a = \min w$, $b = \max w$ and $u = 2^B - 1$ at bit width B. Using 16-bit floating-point (FP16) scales, deployment values are

$$s = \mathrm{FP16}\left(\max\left\{\frac{\max(b-a, 10^{-5})}{u}, 2^{-24}\right\}\right), \qquad z = \mathrm{clip}(-\mathrm{round}(a/s), 0, u),$$
$$q = \mathrm{clip}(\mathrm{round}(w/s) + z, 0, u), \qquad \widehat{w} = (q - z)s. \tag{8}$$

During training, the weights determine the scale, zero point and integer codes at each forward pass. The zero point is detached within that pass and recomputed on the next one. Scale and weight rounding use a straight-through estimator (STE), while clipping retains its range-dependent derivative.

Gradients update the dense master and nonquantized language-model parameters; teacher and floating targets are detached. Appendix A.3 specifies the parameter scope. Rollout synchronization requantizes the updated master. After N updates, the same formula produces (q_N, z_N, s_N) for packed inference, with learner, export and runtime consistency checks. Algorithm 1 summarizes training and export.

6 Experiments

6.1 Experimental Setup

In this section, we compare deployed task accuracy, isolate the training components and examine how updates change the low-bit model.

Models and quantization. We study Qwen3.5-4B and Qwen3-8B, covering hybrid and conventional attention (Qwen Team, 2026; 2025). The selected weight masks contain 128 and 252 decoder Linear modules, respectively. To isolate weight quantization, embeddings, norms, output heads, vision and excluded operators remain high precision. We denote 2-bit and 3-bit weight settings by W2 and W3. OPQ uses asymmetric integers with group sizes 32 and 128, respectively. Clipping initializes the dense master once. During training, scales, zero points and codes are derived from its weights as described in Section 5.

Tasks. ScienceQA, PIQA, MMLU-Redux, IFEval, GSM8K, MATH-500 and LiveCodeBench (LCB) cover visual reasoning, knowledge, instruction following, mathematics and code (Lu et al., 2022; Bisk et al., 2020; Gema et al., 2025; Zhou et al., 2023; Cobbe et al., 2021; Hendrycks et al., 2021; Jain et al., 2024). ScienceQA uses the original 2,017 questions with images for the image-capable 4B model. Avg. equally weights seven tasks for 4B and the six applicable tasks for text-only 8B. We compare these averages within each model. Coding uses the fixed release-v6 182 problems and official execution-based scoring. Evaluation uses seed 22, non-thinking and greedy decoding, with native end-of-sequence (EOS) tokens and no extra stop string.

Comparisons. We evaluate two baseline families. Round-to-nearest (RTN), GPTQ, AWQ and OmniQuant represent PTQ, while LLM-QAT and BitDistiller incorporate distillation. Our weight-only adaptations retain their respective data and objectives within the recorded training budgets. OmniQuant optimizes weight clipping, BitDistiller uses deployment-compatible 3-bit integers in place of the nonuniform NormalFloat format (NF3), and LLM-QAT uses narrow-range W2 with three effective values. Appendix A specifies these implementations and their token budgets.

6.2 Main Results

Deployed accuracy. Table 1 shows that OPQ achieves the highest average accuracy among the evaluated low-bit methods in all four model–precision settings. Each paired cell lists W3 followed by W2, with the 16-bit brain floating-point model (BF16) as a common reference. Avg. uses seven tasks for 4B and six for 8B (except for ScienceQA-IMG). Its advantage grows as the weight budget decreases. With 2-bit weights, OPQ exceeds the strongest baseline by **7.75** percentage points on Qwen3.5-4B and **9.49** points on Qwen3-8B. This pattern highlights the value of learning on student trajectories when quantization most strongly perturbs model behavior.

Task-level gains. OPQ's gains extend across instruction following, knowledge, mathematics and code generation. On Qwen3-8B with 3-bit weights, the largest improvement over BitDistiller is in instruction following, alongside gains in every other evaluated task. Retaining instruction following

Table 1: **W3 / W2 deployment accuracy (%).** **Red bold** and **blue bold underlined** mark first and second observed scores per model and bit width. Dashes indicate unavailable values.

	Visual	Knowledge		Instruction	Mathematics		Code	
Method	SQA-IMG	PIQA	MMLU-R	IFEval	GSM8K	MATH	LCB	Avg.
Qwen3.5-4B W3 / W2								
BF16	88.89	86.07	77.11	81.70	94.62	93.00	28.02	78.49
RTN	83.74 / 24.49	81.50 / 38.30	69.26 / 24.23	74.86 / 17.74	85.44 / 1.29	67.40 / 1.60	12.09 / 0.00	67.76 / 15.38
GPTQ	84.48 / —	79.98 / —	70.91 / —	71.90 / —	88.93 / —	69.40 / —	10.99 / —	68.08 / —
AWQ	**84.98** / 69.61	**83.19** / 58.38	**72.88** / 42.49	70.79 / 39.74	87.72 / 22.44	80.00 / 12.40	18.68 / 1.65	**71.18** / 35.24
OmniQuant	**85.32** / 22.56	81.66 / 36.13	70.35 / 14.37	**75.05** / 19.96	89.61 / 5.38	77.80 / 4.00	13.19 / 0.00	70.43 / 14.63
LLM-QAT	82.15 / 21.42	81.56 / 9.36	69.58 / 8.12	68.58 / 13.31	86.13 / 1.82	66.40 / 3.20	13.74 / 0.00	66.88 / 8.18
BitDistiller	84.63 / **74.37**	82.64 / **68.12**	72.40 / **64.09**	60.63 / **44.36**	**92.19** / **78.39**	**80.20** / **49.40**	**19.23** / **9.89**	70.27 / **55.52**
OPQ	84.78 / **75.81**	**83.90** / **78.89**	**73.28** / **65.16**	**75.42** / **71.90**	**92.27** / **83.62**	**85.20** / **55.40**	**20.33** / **12.09**	**73.60** / **63.27**
Qwen3-8B W3 / W2								
BF16	—	85.36	75.39	84.47	92.95	84.20	23.08	74.24
RTN	—	72.85 / 9.47	60.46 / 9.05	69.50 / 8.13	79.91 / 1.97	67.60 / 0.60	12.64 / 0.00	60.49 / 4.87
GPTQ	—	67.14 / —	64.16 / —	77.63 / —	88.86 / —	72.20 / —	12.09 / —	63.68 / —
AWQ	—	**81.94** / 16.65	67.09 / 26.63	**79.30** / 20.89	**90.83** / 1.90	73.60 / 1.40	15.38 / 0.00	68.02 / 11.24
OmniQuant	—	77.42 / 0.92	67.89 / 1.02	70.43 / 4.99	85.29 / 0.61	71.80 / 0.60	19.23 / 0.00	65.34 / 1.36
LLM-QAT	—	79.38 / 1.20	67.56 / 0.96	68.95 / 9.06	85.67 / 0.08	66.40 / 0.80	16.48 / 0.00	64.07 / 2.02
BitDistiller	—	81.10 / **63.22**	**70.30** / **30.93**	75.40 / **49.17**	90.70 / **76.50**	**75.40** / **48.00**	**19.78** / **11.54**	**68.78** / **46.56**
OPQ	—	**82.48** / **64.36**	**71.54** / **55.46**	**82.26** / **75.05**	**91.43** / **80.52**	**76.40** / **48.80**	**20.33** / **12.09**	**70.74** / **56.05**

together with reasoning and coding accuracy is important for preserving the usefulness of the quantized model beyond any single benchmark.

6.3 Language Modeling Quality

We evaluate perplexity (PPL) on WikiText-2 raw test with a 2,048-token context and 297,047 predicted tokens. Table 2 compares Qwen3.5-4B W3 methods with the BF16 reference. Lower PPL indicates better language modeling quality. OPQ reduces PPL by **13.50%** relative to RTN and remains comparable to GPTQ.

Table 2: **WikiText-2 PPL.** Qwen3.5-4B W3; lower is better.

Method	BF16	RTN	GPTQ	AWQ	OmniQuant	LLM-QAT	BitDistiller	OPQ
PPL ↓	9.586	12.220	**10.701**	10.955	11.373	11.175	10.842	**10.570**

Data exposure. GPTQ, AWQ and OmniQuant use $128 \times 2{,}048$ calibration tokens from WikiText-103. The fixed training prompts for our BitDistiller INT adaptation contain approximately one-third WikiText. In contrast, OPQ trains only on 4,096 English8 prompts and uses no WikiText data; Direct/RTN requires no data calibration. WikiText-103 calibration and WikiText-2 test are different splits, but both belong to the WikiText/Wikipedia domain, giving these PTQ baselines domain-matched calibration. Despite this difference, OPQ obtains PPL comparable to GPTQ and lower than the other evaluated PTQ and QAT baselines. This result supports quantization recovery across corpora rather than WikiText-specific fitting.

6.4 Ablation Studies

Components and trajectories. Table 3 compares OPQ with three alternatives. Removing K/V leaves on-policy Dynamic KL; removing Dynamic KL leaves on-policy reverse KL with the same K/V term. Off-policy RKL uses fixed teacher responses and no K/V supervision. The trajectory comparison in Figure 1 holds RKL fixed and shows a clear average improvement from refreshing student responses. Thus, the trajectory choice contributes before either objective component is added.

Table 3: **Qwen3.5-4B W3 component ablation (%).**

Method	SQA-IMG	PIQA	MMLU-R	IFEval	GSM8K	MATH	LCB	Avg.
OPQ	**84.78**	**83.90**	**73.28**	**75.42**	92.27	**85.20**	**20.33**	**73.60**
w/o KV	84.00	83.70	73.10	74.60	**92.30**	84.90	19.78	73.20
w/o Dynamic KL	84.40	83.50	72.70	75.10	91.60	84.20	19.23	72.96
Off-policy RKL	82.80	82.60	71.30	71.40	90.50	81.00	15.38	70.71

Complementary objectives. With the quantizer, initialization and training budget fixed, removing either Dynamic KL or K/V alignment lowers average accuracy. K/V improves instruction following, visual reasoning and code generation, while Dynamic KL also improves mathematics. The complete objective achieves the strongest overall result, indicating that adapting output supervision and aligning attention representations address complementary aspects of quantization error.

6.5 Discussion

Less sensitivity to reduced precision. Reducing weight precision from 3 bits to 2 bits lowers accuracy for both architectures. However, OPQ loses less accuracy than BitDistiller in each case. Its larger advantage at 2 bits therefore reflects improved resilience to a tighter weight budget, rather than a benefit confined to one precision setting.

What changes an attention read? We examine attention memory in Qwen3.5-4B W3 by replacing selected keys, values or both with the frozen teacher's corresponding states while holding the query, mask and token IDs fixed. A discovery set selects the layer and window; 137 independent confirmation prompts evaluate layer 31 and lags 1–32. Figure 4 reports the reduction in teacher KL, so positive values indicate improved conditional fidelity.

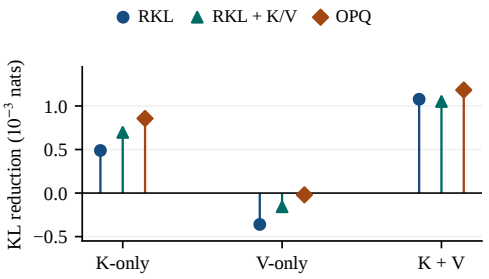


Figure 4: **K/V contributions to output fidelity.** On-policy variants; positive values indicate lower teacher KL.

Keys and values act together. K-only replacement lowers KL for all three conditions, whereas V-only replacement slightly increases it. Replacing both gives a larger improvement than replacing keys alone. A value is useful through the attention weight assigned to it, so jointly supervising K/V respects their coupled role in retrieval.

Many weights can change without many code switches. Training updates nearly all master weights and many deployment scales, while changing relatively few integer codes. Under $\widehat{W} = (q - z)s$, a shared group scale changes reconstructed weights even when their codes stay fixed. Code-switch frequency alone therefore understates how much the deployed model can change. This explains how learning can refine the quantized representation without widespread reassignment of integer values. Detailed measurements and storage costs are provided in Appendix A.4.

7 Conclusion

In summary, we present OPQ, an on-policy quantization framework that learns on the prefixes reached by a low-bit student. A floating view of the student identifies precision-related output discrepancies and provides the reference for K/V alignment. Dynamic KL adjusts output supervision, while a constant K/V term aligns attention memory. Our analysis explains the corrective logit gradient and the effect of K/V error on attention outputs. Across two models with 2-bit and 3-bit weights, OPQ achieves the highest average task accuracy among the evaluated low-bit methods. The results highlight the value of combining student trajectories with output and attention alignment to preserve capabilities under low precision.

AI USE STATEMENT

We used AI tools to aid or polish writing and assist with parts of the code implementation.

REPRODUCIBILITY STATEMENT

The supplement provides manuscript and figure-generation code, measured source data, and the quantizer implementation. Appendix A specifies training, decoding and export.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

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.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2 edition, 2006. doi: 10.1002/047174882X. URL https://onlinelibrary.wiley.com/doi/book/10.1002/047174882X.

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

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017. URL https://arxiv.org/abs/1704.00805.

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? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5069–5096. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.262. URL https://aclanthology.org/2025.naacl-long.262/.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2306.08543v4. Conference-version title; later arXiv versions use a revised title.

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.

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. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

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

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, volume 35, pp. 2507–2521, 2022. doi: 10.52202/068431-0182. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/11332b6b6cf4485b84afadb1352d3a9a-Abstract-Conference.html.

NVIDIA. Quantization-aware RL (QARL). NeMo RL nightly documentation, n.d. URL https://docs.nvidia.com/nemo/rl/nightly/guides/quantization-aware-rl.html. Section: Quantization-Aware Distillation (On-Policy QAD). Accessed September 26, 2026.

Yasuyuki Okoshi, Hikari Otsuka, Daichi Fujiki, and Masato Motomura. Towards quantization-aware training for ultra-low-bit reasoning llms. In *International Conference on Learning Representations*, volume 2026, pp. 19152–19170, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/209423f076b6479ab3a4f45886e30306-Paper-Conference.pdf.

Qwen Team. Qwen3-8b: Official model card. Hugging Face model card, 2025. URL https://huggingface.co/Qwen/Qwen3-8B. Accessed September 26, 2026.

Qwen Team. Qwen3.5-4B: Official model card. Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.5-4B. Accessed September 26, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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

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

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.

A Experimental Details

In this section, we provide implementation details in four parts. We first specify tasks and decoding, then describe quantization and baselines, explain optimization, and finish with export and measured deployment changes.

A.1 Evaluation

Shared decoding settings. Models retain their official tokenizers and non-thinking chat templates. Completed evaluations use EvalScope 1.9.1, vLLM 0.19.1, temperature 0, seed 22 and one response per item.

Table 4: Evaluation denominators, generated-token limits and metrics.

Task	Items	Limit	Metric
ScienceQA-Image / withheld	2,017	64	Accuracy
PIQA	1,838	64	Accuracy
MMLU-Redux	5,700	64	Accuracy
IFEval	541	3,072	Prompt strict
GSM8K	1,319	6,144	Accuracy
MATH-500	500	8,192	Accuracy
LiveCodeBench	182	2,000	Official pass@1

Math answers end in a box, choice tasks request direct choices, and IFEval retains its original instructions. Visible reasoning is permitted where requested, while hidden thinking is disabled. Coding uses release-v6, native EOS and no added stop string. Model-generated programs are evaluated by the official scorer in an isolated container with networking disabled.

Only the image-capable model receives original ScienceQA images. The paired withheld condition removes the image for the same questions. Avg. equally weights the seven displayed tasks for 4B and the six applicable tasks for text-only 8B. The latter has neither an image score nor an image-inclusive average. Comparisons of averages are within each model.

A.2 Quantization

Quantization scope. Our quantizer uses asymmetric clipping selected with activation statistics, followed by groupwise MinMax quantization derived from the weights (Du et al., 2024). Group size is 32 for INT2 and 128 for INT3. The masks contain 252 decoder Linear modules for 8B and 128 for 4B. Embeddings, normalization, output heads, vision, MTP and excluded linear-attention operations remain high precision.

Clipping is selected by searching positive and negative bounds using cached input activations and group-output reconstruction error. The implementation uses a 20-point shrink grid with maximum shrink 0.5, giving 100 two-sided candidate pairs, and at most 512 sampled activation tokens per layer. The selected bounds initialize the master once. Training then updates the master, with scale and zero point derived from its group extrema using equation 8.

OPQ uses English8 prompts without WikiText data. BitDistiller instead uses its Alpaca/WikiText corpus. GPTQ, AWQ and OmniQuant use 128 sequences of 2048 calibration tokens from WikiText-103. This differs from the WikiText-2 test split but shares its Wikipedia domain.

The PTQ baselines differ in how they choose their quantized weights. GPTQ uses Hessian-based reconstruction. AWQ includes scaling and clipping. OmniQuant optimizes learnable weight clipping without learnable equivalent transformations. RTN directly rounds using group MinMax ranges at both bit widths.

LLM-QAT uses fixed teacher-generated sequences from random starts, four near-greedy initial tokens and subsequent teacher sampling. Full-vocabulary forward KL trains a symmetric MinMax weight quantizer through STE. Its narrow-range W2 adaptation uses three effective values, not the original special two-bit branch.

BitDistiller uses fixed prompts from Alpaca and WikiText at roughly 2:1, teacher responses at temperature 0.7, asymmetric clipping and integer quantization. Its objective is fixed CAKLD, $\gamma\,\mathrm{KL}(Q\|T) + (1-\gamma)\,\mathrm{KL}(T\|Q)$. To keep the objective fixed during training, γ is estimated once as the token-weighted teacher probability of the generated response tokens and then frozen. In contrast, OPQ uses student trajectories and a coefficient based on the precision discrepancy. Neither baseline uses OPQ's on-policy objective or K/V loss.

A.3 Optimization

Nonquantized language-model parameters remain trainable at high precision, including token embeddings, normalization layers, the output head and the 4B model's linear-attention modules. The vision branch and multi-token prediction modules are unused by the text-only training objective and receive no updates. The teacher and floating targets receive no gradients.

OPQ configuration. We specify 4096 English8 prompts, global batch 64, microbatch 1, 64 optimizer updates, prompt limit 2048 and response limit 4096. Rollout uses compilation, sampling temperature 1, top-$p = 1$ and no top-k truncation. AdamW uses learning rate 2×10^{-6}, betas $(0.9, 0.999)$, zero weight decay, gradient-norm clipping 1.0, cosine decay and warmup fraction 0.05.

The baseline adaptations use 4096 composite records. LLM-QAT uses initial learning rate 2×10^{-5} and cosine decay. BitDistiller uses constant 8×10^{-6}, both without warmup. Appendix B reports training costs with the observed effective loss-token counts and hardware for each method.

To preserve the same objective across distributed microbatches, let $\mathcal{R}_m$ be valid response positions in microbatch m, N_{resp} the denominator for the distributed update and d_{DP} its data-parallel size. The local contribution is

$$\frac{d_{\mathrm{DP}}}{N_{\mathrm{resp}}} \sum_{i \in \mathcal{R}_m} B_i + \frac{d_{\mathrm{DP}}|\mathcal{R}_m|}{N_{\mathrm{resp}}}\,\lambda_{\mathrm{KV}} L_{\mathrm{KV},m}. \tag{9}$$

This preserves response-token weighting across gradient accumulation. Teacher and floating targets are detached; the floating computation is refreshed from the master.

A.4 Deployment

Objective hyperparameters. We use $k = 20$, $\tau = 0.1$, $\lambda_{\mathrm{KV}} = 0.1$ and $\epsilon = 10^{-5}$, with $N = 64$ updates. In the 4B W3 prefix diagnostics, the teacher's own top-20 tokens contain 96.75–98.83% of its probability mass across the ten prefix-source/length averages. This concentration motivates examining a small leading set. The coverage statistic in equation 4 instead ranks tokens with the two student views and measures their teacher mass. Both KL losses are computed over the full vocabulary.

Weight and scale rounding use numerical straight-through operators. Their backward pass transfers the derivative to the corresponding unrounded input. The scale derivative flows through the group extrema. Integer zero points are detached within each forward pass and recomputed on the next one.

K/V hooks identify normalized keys before RoPE and attention values by corresponding module names. Qwen3.5-4B has full-attention layers 3,7,11,15,19,23,27,31; Qwen3-8B has 36 full-attention layers. This gives 16 and 72 K/V feature modules, respectively. K/V supervision covers these full-attention modules.

The selected weights remain FP32 masters throughout optimization and export; matrix multiplication follows the activation dtype.

Finally, the checkpoint includes master parameters, optimizer and recovery state, plus the quantization configuration and clipping record. Deployment stores final codes, scales and zeros derived from the final master. Metadata alone contributes approximately $(16 + B)/g$ bits per selected weight, beyond the B code bits, before padding and group indices. For INT2 g32, the total is about 2.5625 bits per selected weight. For INT3 g128, it is about 3.1484. Full model size is larger because excluded operators remain high precision.

Table 5 distinguishes master updates, FP32 scales, FP16 deployment scales, integer codes and zero points. Changes to a weight-derived scale follow from updating the master. PPL is supplementary and is excluded from task averages.

Table 5: **OPQ deployment changes.** Change rates are percentages; storage is static weight tensors in GB.

Model	Bits	Master	FP32 s	FP16 s	q	z	GB	PPL↓
Qwen3.5-4B	2	99.998	99.995	69.469	0.320	0.093	5.024	14.410
Qwen3.5-4B	3	99.998	99.993	60.911	0.312	0.095	5.212	10.570
Qwen3-8B	2	99.439	99.439	39.815	0.172	0.079	4.720	19.480
Qwen3-8B	3	99.643	99.640	31.246	0.179	0.087	5.229	11.400

B EFFICIENCY

Table 6 summarizes 64-step training of Qwen3.5-4B with 3-bit weights and group size 128. LLM-QAT and BitDistiller ran on eight A100 GPUs with 40 GB each; OPQ ran on eight A800 GPUs with 80 GB each, split between four student/rollout GPUs and four teacher GPUs. Effective loss-token counts include response positions used by the distillation objective.

Table 6: Training cost for Qwen3.5-4B W3. GPU types are specified above.

Method	Loss tokens (M)	Train time (h)	Reserved GPU·h	Median step (s)	Peak memory (GiB) allocated / reserved
LLM-QAT	2.454	0.50	3.98	23.3	18.6 / 30.0
BitDistiller	2.454	4.95	39.63	231.7	27.3 / 36.4
OPQ	2.193	3.35	26.8	178.5	26.4 / 30.0

- **Training budget.** All methods use 64 updates. The fixed-response baselines process 2.454M effective loss tokens, while OPQ's student-generated responses contribute 2.193M.
- **Recorded cost.** OPQ takes 3.35 h and 26.8 reserved GPU·h. Table 6 reports each method's elapsed time and memory under its stated hardware. Baseline memory peaks are measured on rank 0; OPQ reports actor memory peaks.
- **Step breakdown.** OPQ's median generation, actor-update and weight-synchronization times are 42.0 s, 68.5 s and 58.4 s, respectively. Its median total step is 178.5 s. Actor computation and weight synchronization are the larger optimization opportunities among these stages.
- **Token throughput.** OPQ achieves a median 66.9 token/s per student GPU over the 64 updates. For each step, this is the number of non-padding prompt and response tokens divided by the full step wall time and four student GPUs. This metric includes prompt processing and training, rather than measuring generation alone.

Scope. Hardware and effective token counts differ across methods, so these costs describe the recorded runs rather than an isolated algorithmic speed comparison; separately computed stage medians need not sum to the total-step median.

C THEORETICAL ANALYSIS

We first relate student prefixes to conditional supervision, then derive the Dynamic KL gradients and the propagation of K/V discrepancies through attention.

C.1 ON-POLICY SUPERVISION

To explain why prefixes matter, consider a finite horizon H and compatible autoregressive supports. The chain rule for relative entropy gives (Cover & Thomas, 2006)

$$\mathrm{KL}(Q_{1:H} \| T_{1:H}) = \sum_{i=1}^{H} \mathbb{E}_{y_{<i} \sim Q} \, \mathrm{KL}(Q_i(\cdot \mid y_{<i}) \| T_i(\cdot \mid y_{<i})). \tag{10}$$

To see this, expand the sequence log ratio into conditional log ratios and condition each summand on its prefix. An absorbing end token can represent early termination. Finiteness of the expectations and teacher support on the student's support are required.

The prefix law in this identity is the same Q as the conditional distribution. Sampling temperature, truncation or stale rollout weights can change that law. Our learner optimizes conditional losses on sampled, detached trajectories, not the full derivative of sequence KL. Indeed,

$$\nabla_\theta \mathbb{E}_{P_\theta}[f_\theta] = \mathbb{E}_{P_\theta}[\nabla_\theta f_\theta + f_\theta \nabla_\theta \log P_\theta].$$

Holding trajectories fixed retains the conditional-loss term and omits the trajectory-distribution term. The identity therefore motivates supervision on visited prefixes, while the implemented update differentiates only the conditional loss.

C.2 Dynamic KL Gradients

Proof of Lemma 1. Fix the prediction prefix, the teacher T_i and the detached coefficient α_i. All sums below run over the full vocabulary. For $Q_i = \mathrm{softmax}(u_i)$,

$$\frac{\partial Q_i(j)}{\partial u_i(v)} = Q_i(j)\big(\mathbf{1}_{j=v} - Q_i(v)\big). \tag{11}$$

First, differentiating $D_i = \sum_j T_i(j)\log(T_i(j)/Q_i(j))$ gives

$$\begin{aligned}\frac{\partial D_i}{\partial u_i(v)} &= -\sum_j T_i(j)\big(\mathbf{1}_{j=v} - Q_i(v)\big) \\ &= Q_i(v) - T_i(v).\end{aligned} \tag{12}$$

Second, differentiating $R_i = \sum_j Q_i(j)\log(Q_i(j)/T_i(j))$ gives

$$\begin{aligned}\frac{\partial R_i}{\partial u_i(v)} &= \sum_j Q_i(j)\big(\mathbf{1}_{j=v} - Q_i(v)\big)\left(\log\frac{Q_i(j)}{T_i(j)} + 1\right) \\ &= Q_i(v)\left(\log\frac{Q_i(v)}{T_i(v)} - R_i\right).\end{aligned} \tag{13}$$

Since α_i is detached, the mixture has derivative

$$\frac{\partial B_i}{\partial u_i(v)} = (1-\alpha_i)Q_i(v)\left(\log\frac{Q_i(v)}{T_i(v)} - R_i\right) + \alpha_i\big(Q_i(v) - T_i(v)\big). \tag{14}$$

For an underrepresented token, $Q_i(v) \le T_i(v)$ implies $\log(Q_i(v)/T_i(v)) \le 0$. Together with $R_i \ge 0$ and $1-\alpha_i \ge 0$, this makes the first term nonpositive. Negating the derivative therefore yields equation 5.

For the limit, positivity of the fixed teacher on a finite vocabulary gives

$$0 \le R_i = \sum_j Q_i(j)\log Q_i(j) - \sum_j Q_i(j)\log T_i(j) \le \log\frac{1}{\min_j T_i(j)}. \tag{15}$$

The upper bound follows from $\sum_j Q_i(j)\log Q_i(j) \le 0$ and $-\log T_i(j) \le \log(1/\min_j T_i(j))$. Thus, as $Q_i(v) \to 0$, $Q_i(v)R_i \to 0$, $Q_i(v)\log T_i(v) \to 0$, and $Q_i(v)\log Q_i(v) \to 0$. Equation equation 13 then tends to zero. For fixed $\alpha_i > 0$, the forward contribution tends to $-\alpha_i T_i(v)$, proving equation 6. □

The coefficient is held fixed only for this local objective. Training recomputes it from the precision comparison on subsequent updates.

C.3 Attention Error

We establish the bound in two steps. First, we bound the change in attention probabilities. Second, we separate the effects of keys and values on the attention output. Let $p = \mathrm{softmax}(s)$. For a perturbation h, set $\mu = \sum_j p_j h_j$. Its Jacobian satisfies

$$\|J(s)h\|_1 = \sum_j p_j|h_j - \mu| \le \sqrt{\sum_j p_j(h_j-\mu)^2} \le \sqrt{\sum_j p_j h_j^2} \le \|h\|_2.$$

Integrating along the logit segment yields $\|\mathrm{softmax}(s) - \mathrm{softmax}(s')\|_1 \le \|s - s'\|_2$, with this norm pairing. General Lipschitz properties of softmax are discussed by Gao & Pavel (2017). For a common query, $\|p^Q - p^F\|_1 \le \|K^Q - K^F\|_F \|q\|_2/\sqrt{d}$. Next, splitting the output difference gives

$$\begin{aligned}\|o^Q - o^F\|_2 &\le \|(p^Q)^\top (V^Q - V^F)\|_2 + \|(p^Q - p^F)^\top V^F\|_2 \\ &\le \varepsilon_V + M\|p^Q - p^F\|_1 \le \varepsilon_V + \frac{RM}{\sqrt{d}}\varepsilon_K, \end{aligned} \tag{16}$$

which proves Proposition 1. The first term uses nonnegative attention weights summing to one. The second uses the value-norm bound.

The normalized loss in equation 2 controls a prefix error through $\|Z_i^Q - Z_i^F\|_F = \sqrt{A_i^Z e_i^Z}$, where $A_i^Z = n_i\epsilon + \|Z_i^F\|_F^2$. A common rotary transform preserves the key Frobenius norm. For query and value bounds $R_{q,i} \ge \|q_i\|_2$ and $M_i \ge \max_j \|v_{i,j}^F\|_2$, this gives

$$\|o_i^Q - o_i^F\|_2 \le \sqrt{A_i^V e_i^V} + \frac{R_{q,i}M_i}{\sqrt{d}}\sqrt{A_i^K e_i^K}.$$

The bound weights NMSE by prefix energy and query magnitude. When queries differ, it also includes their discrepancy:

$$\|o^Q(q^Q) - o^F(q^F)\|_2 \le \varepsilon_V + \frac{M}{\sqrt{d}}\left(\varepsilon_K\|q^Q\|_2 + \|K^F\|_{\mathrm{op}}\|q^Q - q^F\|_2\right).$$

This bound describes one attention read; subsequent layers and generated tokens determine how its error propagates.

C.4 Quantizer Gradients

The rounding STE supplies a surrogate derivative for the discrete quantizer. Gradients through scale rounding flow to the group range derived from the weights, while zero-point rounding is detached. At tied extrema, the implementation's reduction rule determines the subgradient.

The fixed K/V coefficient gives the conditional surrogate gradient $\widetilde{\nabla}_W \,\mathrm{mean}_{i\in\mathcal{R}} B_i + \lambda_{\mathrm{KV}}\widetilde{\nabla}_W L_{\mathrm{KV}}$. Export checks compare packed weights and runtime outputs with the learner's quantized forward.