# QRPE: Taming Quantization Errors via Quantization-guided Relative Prompt Evolution

Anonymous ACL submission

## Abstract

Post-training quantization (PTQ) is crucial for efficient LLM deployment but often suffers from activation outliers, leading to significant accuracy degradation. Traditional calibration methods rely on static datasets, which may fail to capture the model's most quantization-sensitive features. In this work, we propose Quantization-guided Relative Prompt Evolution (QRPE), a training-free framework that autonomously synthesizes calibration data by optimizing prompts based on quantization feedback. Instead of relying on ground-truth labels, QRPE leverages a relative comparison mechanism: it generates a group of candidate outputs, evaluates their quantization robustness by using quantization loss as a reward signal, and evolves the generation prompt through natural language feedback to favor "quantization-friendly" patterns. Extensive experiments on the Qwen3 family demonstrate that QRPE significantly outperforms static calibration baselines, reducing INT4 quantization accuracy loss by 44.35% on Common Sense Reasoning and Scientific Knowledge benchmarks, and achieving a 22.92% reduction in accuracy loss compared to the current state-of-the-art generation-based calibration methods. Our analysis reveals that QRPE can effectively tame outliers by guiding model activations toward smoother regions of the activation manifold, without updating any model parameters. Notably, we characterize the scaling behavior between search space size (O) and calibration data volume (N), which further illuminates promising directions for optimizing low-bit quantization.

## 1 Introduction

Large Language Models (LLMs) (Dettmers et al., 2022) have demonstrated remarkable capabilities across diverse tasks, ranging from complex reasoning to code generation. However, their massive parameter counts pose significant challenges for deployment, particularly in resource-constrained environments. Post-Training Quantization (PTQ) has emerged as a promising solution to reduce memory footprint and computational cost without the need for extensive retraining. By mapping high-precision weights and activations to lower-bit representations (e.g., INT4), PTQ enables efficient inference on consumer-grade hardware.

Early quantization research identified that extreme activation values (outliers) severely degrade quantization quality. To mitigate this, various techniques have been proposed, including activation-weight scaling (e.g., AWQ (Lin et al., 2024), SmoothQuant (Xiao et al., 2023)), rotation-based optimization (e.g., QuaRot (Ashkboos et al., 2024), SpinQuant (Liu et al., 2025)), and outlier-aware mixed precision (e.g., SPQR (Dettmers et al., 2024)). While these methods effectively manage outliers algorithmically, their performance remains fundamentally bounded by the quality of the calibration dataset. Most existing approaches rely on static generic datasets (e.g., WikiText-2 (Merity et al., 2017), C4 (Raffel et al., 2020)), which suffer from significant distribution shift: they fail to capture the task-specific activation patterns triggered by user prompts during actual inference. To address this, recent works like FAQ (Xiao et al., 2026) proposed generating calibration data directly from the model or its variants. This approach aligns the calibration distribution with the model's intrinsic behavior and has shown promising results in reducing quantization error.

However, relying solely on the original prompt for calibration data generation, while improving distribution alignment, neglects an underestimated upstream optimization dimension: the prompt itself that drives data generation. Existing generation-based calibration methods typically treat the prompt as a fixed input, implicitly assuming that the original prompt is sufficient to elicit the activation patterns required for effective calibration.

The validity of this assumption merits examination. From the computational mechanism of the Transformer (Vaswani et al., 2017), the semantic content of a prompt directly influences the allocation of attention weights and the routing of feed-forward activations. Prompts with varying semantic complexity can guide the model toward markedly different activation states. Consequently, there may be systematic differences in the quality of calibration signals generated by the original prompt and the optimized prompt—differences that existing generation-based methods have not exploited. Building upon this observation, we ask a central question: Can the quality of model quantization, such as quantization loss, be employed as a guiding signal to refine calibration prompts, thereby achieving improved calibration performance beyond what is attainable solely through the data source ?

Motivated by this insight, we propose QRPE (Quantization-guided Relative Prompt Evolution), a training-free upstream framework designed for generation-based post-training quantization (PTQ). The core idea of QRPE is to treat prompt construction as an optimization problem guided by quantization quality. Specifically, QRPE leverages feedback signals from the quantization algorithm to automatically identify semantic advantages that can systematically improve calibration quality, and uses them to continuously refine calibration prompts. The entire process requires no additional training and is decoupled from specific quantization algorithms, enabling it to serve as a plug-and-play upstream module that integrates seamlessly with existing quantizers such as GPTQ (Frantar et al., 2023).Our contributions are:

- We identify that the construction of calibration prompts constitutes an independent optimization dimension in generation-based post-training quantization (PTQ), which has not yet been systematically investigated: existing methods focus on aligning the distribution of the data source, while treating the prompt that drives generation as a fixed input, thus overlooking its direct impact on the quality of calibration signals.

- We propose QRPE, a training-free upstream framework for generation-based PTQ, which formulates calibration prompt construction as an optimization problem guided by quantization quality. It leverages quantization loss as a feedback signal to iteratively evolve prompts, and can be integrated with existing quantizers in a plug-and-play manner. To the best of our knowledge, QRPE is the first framework to use per-layer quantization reconstruction loss as the reward signal for optimizing calibration prompts, closing the loop between the quantizer's internal error metric and the upstream data-generation process.

- We systematically validate the effectiveness of QRPE across multiple model architectures. Under INT4/8 settings, QRPE consistently outperforms standard baselines and existing generation-based calibration methods (e.g., achieving an average accuracy of 86.89 on Qwen3-8B, with a +1.19 point improvement over baseline).

## 2 Related Work

### 2.1 Post-Training Quantization for LLMs

Post-Training Quantization (PTQ) aims to reduce the memory footprint and computational cost of pre-trained LLMs without the need for extensive retraining. Early works like BRECQ (Li et al., 2021) introduced block-wise reconstruction to minimize quantization error. However, as LLMs scale, the presence of extreme outliers in activations poses a significant challenge for low-bit quantization (Krishnamoorthi, 2018; Nagel et al., 2021).To mitigate outlier-induced errors, recent approaches focus on three main strategies:

**Rounding Optimization:** Methods like GPTQ (Frantar et al., 2023) and FlexRound (Lee et al., 2023) utilize Hessian-based information or learnable rounding schemes to optimally map weights to discrete grids. FOEM (Zheng et al., 2026) further improves this by incorporating first-order gradient information for more accurate error compensation.

**Outlier Smoothing & Rotation:** SmoothQuant (Xiao et al., 2023) and AWQ (Lin et al., 2024) employ activation-aware scaling to migrate the difficulty of quantization from activations to weights. QuaRot (Ashkboos et al., 2024) and SpinQuant (Liu et al., 2025) use orthogonal matrices to flatten outlier peaks, creating a more quantization-friendly distribution. FPTQuant (van Breugel et al., 2025) introduces function-preserving transforms to adjust activation distributions, achieving significant speedups.

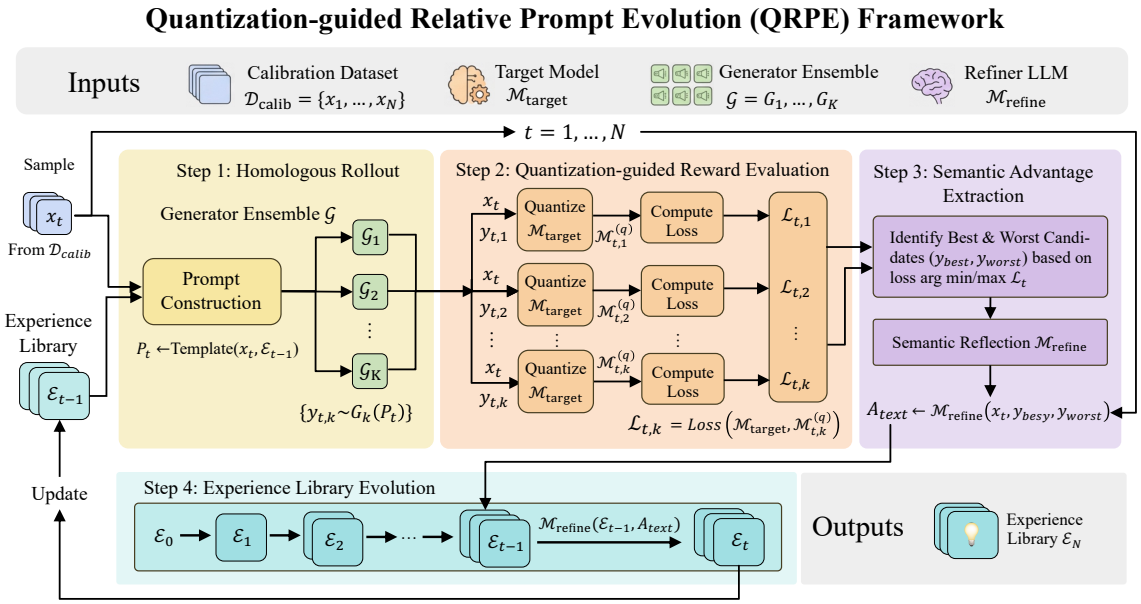


Figure 1: Quantization-guided Relative Prompt Evolution (QRPE) Framework .

**Mixed-Precision & Sparsity:** SpQR (Dettmers et al., 2024) and SqueezeLLM (Kim et al., 2024) adopt a dense-and-sparse strategy, isolating sensitive outliers in high precision while compressing the majority of parameters to low bits. Similarly, SliM-LLM (Huang et al., 2025) and AMLQ (Ou et al., 2024) leverage salience information to allocate bit-widths dynamically.

### 2.2 Data-Centric Quantization Calibration

The quality of calibration data is a bottleneck for PTQ performance. Traditional methods rely on generic corpora, which may not align with the specific activation patterns of the target model. Recent research has shifted towards utilizing the model itself for data generation. ZeroQ (Cai et al., 2020) attempted to synthesize distilled data to match batch normalization statistics. FAQ (Xiao et al., 2026) is a pioneering work that leverages homologous models (e.g., a larger model from family) to regenerate calibration samples. This ensures high distributional alignment and has been shown to significantly reduce quantization error. Similarly, OSTQuant (Hu et al., 2025) optimizes data distribution via learnable transforms, and Fair-GPTQ (Proskurina et al., 2025) generates counterfactual data to mitigate bias during quantization.

## 3 Method

Since calibration quality can only be indirectly observed via downstream quantization feedback and lacks direct supervision signals, we model prompt construction as a training-free black-box optimization problem. Inspired by RL algorithms such as training-free GRPO (Cai et al., 2025), we propose QRPE (Quantization-guided Relative Prompt Evolution), an autonomous high-quality calibration data synthesis framework for low-bit LLM quantization. Instead of improving prompts through gradient-based parameter updates, QRPE operates as an iterative loop consisting of four key stages as illustrated in Figure 1: (1) Homologous Rollout, (2) Quantization-guided Reward Evaluation, (3) Semantic Advantage Extraction, and (4) Experience Library Evolution. The detailed procedure is outlined in Algorithm 1.

### 3.1 Overview and Problem Formulation

Let $\mathcal{M}_{\text{target}}$ denote the target LLM to be quantized. Our goal is to generate a calibration dataset $\mathcal{D}^*_{\text{quant}}$ that minimizes the quantization error of $\mathcal{M}_{\text{target}}$. Unlike traditional methods that rely on static datasets $\mathcal{D}_{\text{calib}}$, QRPE dynamically constructs prompts to elicit diverse and "quantization-friendly" activations. We maintain an Experience Library $\mathcal{E}_t$, which stores accumulated semantic strategies to guide the generation process. At each iteration $t$, a query $x_t$ is sampled from a seed

dataset $\mathcal{D}_{\text{calib}}$ of size N, and concatenated with $\mathcal{E}_{t-1}$ to form an evolved prompt $P_t$.

### 3.2 Step 1: Homologous Rollout

The first step aims to generate diverse candidate responses that cover the potential activation space of the target model. We construct the input prompt $P_t$ using a template function:

$$P_t = Template\left(x_t, \mathcal{E}_{t-1}\right), \mathcal{E}_0 = \emptyset. \quad (1)$$

Where $\mathcal{E}_{t-1}$ provides high-level instructions (e.g., "Reason step-by-step", "Consider edge cases") derived from previous successful iterations.

To ensure the generated data distribution aligns with $\mathcal{M}_{\text{target}}$, we employ a Homologous Generator Ensemble $\mathcal{G} = \{G_1, \ldots, G_K\}$. As posited by Homologous Calibration Hypothesis (Xiao et al., 2026), models from the same family share similar activation landscapes but possess hierarchical reasoning capabilities. By using stronger homologous models (or the target model itself) as generators, we obtain a set of candidate answers $\mathcal{Y}_t = \{y_{t,1}, \ldots, y_{t,K}\}$ that are both distributionally aligned and semantically richer than what a single greedy decoding could produce.

$$y_{t,k} = G_k(P_t), k = 1, \ldots, K, t = 1, \ldots, N. \quad (2)$$

### 3.3 Step 2: Quantization-guided Reward Evaluation

A core innovation of QRPE is its reward mechanism. Instead of using linguistic metrics (e.g., perplexity or coherence) which correlate poorly with quantization robustness, we directly use Quantization Loss as the reward signal. For each candidate answer $y_{t,k} \in \mathcal{Y}_t$, we form a temporary calibration sample $S_{t,k} = (x_t, y_{t,k})$. We then perform a "trial quantization" on $\mathcal{M}_{\text{target}}$ using $S_{t,k}$ to obtain a quantized model $\mathcal{M}_{t,k}^{(q)}$. The reward is defined as the reconstruction loss $\mathcal{L}_{t,k}$ between the original and quantized model:

$$\mathcal{L}_{t,k} = Loss(\mathcal{M}_{\text{target}}(S_{t,k}), \mathcal{M}_{t,k}^{(q)}(S_{t,k})). \quad (3)$$

This loss (e.g., HW-MSE in Appendix A) explicitly measures how well the quantization parameters estimated from $y_{t,k}$ preserve the model's precision. A lower $\mathcal{L}_{t,k}$ suggests that $y_{t,k}$ facilitates the quantizer in estimating more optimal quantization parameters (e.g., scale, zero).

**Algorithm 1** Quantization-guided Relative Prompt Evolution (QRPE)

**Input:** Calibration dataset $\mathcal{D}_{\text{calib}}$, Target model $\mathcal{M}_{\text{target}}$, Generator ensemble $\mathcal{G}$, Refiner LLM $\mathcal{M}_{\text{refine}}$, Iterations $N$, Initialize experience library $\mathcal{E}_0 \leftarrow \emptyset$

**Output:** Optimized experience library $\mathcal{E}^*$

```
for t = 1, 2, ..., N do
    x_t ~ D_calib
    P_t ← Template(x_t, E_{t-1})
    Y_t ← G(P_t), {y_{t,1}, ..., y_{t,K}} ← {G_1(P_t), ..., G_K(P_t)}
    for k = 1, ..., K do
        S_{t,k} ← (x_t, y_{t,k})
        M^(q)_{t,k} ← Quantize(M_target, S_{t,k})
        L_{t,k} ← Loss(M_target, M^(q)_{t,k})
    end
    L_t ← {L_{t,1}, ..., L_{t,k}}
    A_text ← M_refine(x_t, Y_t, L_t)
    E_t ← M_refine(E_{t-1}, A_text)
end
return E_N
```

### 3.4 Step 3: Semantic Advantage Extraction

After evaluating all candidates, we identify the Best Candidate $y_{\text{best}}$ (lowest loss) and the Worst Candidate $y_{\text{worst}}$ (highest loss). Simple filtering is not enough; we aim to understand why $y_{\text{best}}$ yielded better quantization results. We employ a Refiner LLM $\mathcal{M}_{\text{refine}}$ to perform a comparative analysis:

$$A_{\text{text}} = \mathcal{M}_{\text{refine}}(x_t, \mathcal{Y}_t, \mathcal{L}_t). \quad (4)$$

Here, $A_{\text{text}}$ represents the extracted "Semantic Advantage"—a natural language description of the features (e.g., "used more complex syntax," "included mathematical derivations") that contributed to the superior calibration performance. This contrastive approach filters out noise and distills actionable insights.

### 3.5 Step 4: Experience Library Evolution

The final step is to update the Experience Library. We treat $\mathcal{E}$ as a dynamic knowledge base that evolves over time. The Refiner LLM integrates the newly found advantage $A_{\text{text}}$ into the existing library $\mathcal{E}_{t-1}$.

$$\mathcal{E}_t = \mathcal{M}_{\text{refine}}(\mathcal{E}_{t-1}, A_{\text{text}}). \quad (5)$$

The update process follows a strict set of maintenance principles: merge similar rules to prevent

| Model | Method | #W | Arc_E | Arc_C | HellaS | WinoG | PiQA | Avg | Loss |
|---|---|---|---|---|---|---|---|---|---|
| Qwen3-8B | BF16 | 16 | 94.70 | 94.28 | 79.17 | 83.03 | 90.75 | 88.39 | - |
| | GPTQ | 4 | 94.36 | 92.83 | 75.40 | 78.30 | 87.60 | 85.70 | 2.69 |
| | +FAQ | 4 | 94.19 | 93.17 | 75.93 | 79.40 | 88.68 | 86.27 | 2.11 |
| | +QRPE | 4 | **94.49** | **93.43** | **77.00** | **79.87** | **89.66** | **86.89** | **1.50** |
| | GPTQ | 8 | 94.32 | 93.69 | 78.58 | 80.82 | 89.34 | 87.35 | 1.04 |
| | +FAQ | 8 | 94.44 | 93.87 | 78.84 | 81.65 | 90.18 | 87.80 | 0.59 |
| | +QRPE | 8 | **94.57** | **94.28** | **79.17** | **82.64** | **90.26** | **88.18** | **0.20** |
| Llama3.1-8B | BF16 | 16 | 88.97 | 80.63 | 73.27 | 61.80 | 83.08 | 77.55 | - |
| | GPTQ | 4 | 85.11 | 75.15 | 56.22 | 57.50 | 76.28 | 70.05 | 7.50 |
| | +FAQ | 4 | 87.08 | 77.99 | 71.95 | 58.01 | 80.47 | 75.10 | 2.45 |
| | +QRPE | 4 | **88.13** | **78.50** | **72.26** | **60.30** | **82.26** | **76.29** | **1.26** |
| | GPTQ | 8 | 85.90 | 76.54 | 57.90 | 58.58 | 78.44 | 71.47 | 6.08 |
| | +FAQ | 8 | 87.88 | 79.01 | 72.35 | 58.64 | 80.79 | 75.73 | 1.82 |
| | +QRPE | 8 | **88.60** | **79.96** | **72.80** | **61.06** | **82.05** | **76.89** | **0.66** |
| Phi-4-mini | BF16 | 16 | 81.69 | 80.89 | 59.85 | 69.30 | 71.71 | 72.69 | - |
| | GPTQ | 4 | 74.92 | 68.26 | 40.65 | 54.22 | 61.26 | 59.86 | 12.83 |
| | +FAQ | 4 | 69.40 | 63.99 | 49.16 | **68.82** | 63.22 | 62.92 | 9.77 |
| | +QRPE | 4 | **79.29** | **73.55** | **53.74** | 64.96 | **67.36** | **67.78** | **4.91** |
| | GPTQ | 8 | 80.43 | 79.18 | 59.13 | 67.01 | 69.80 | 71.11 | 1.58 |
| | +FAQ | 8 | 81.58 | **81.60** | **59.42** | 68.28 | **71.25** | 72.43 | 0.26 |
| | +QRPE | 8 | **81.65** | 81.08 | 59.02 | **69.88** | 70.93 | **72.51** | **0.18** |
| MiMo-7B | BF16 | 16 | 93.43 | 92.06 | 74.25 | 82.72 | 89.17 | 86.33 | - |
| | GPTQ | 4 | 91.88 | 89.33 | 71.12 | 79.08 | 86.13 | 83.51 | 2.82 |
| | +FAQ | 4 | **92.62** | 90.28 | **71.94** | **80.10** | **86.96** | **84.38** | **1.95** |
| | +QRPE | 4 | 92.13 | **90.44** | 71.36 | 79.95 | 86.89 | 84.15 | 2.17 |
| | GPTQ | 8 | 92.68 | 90.02 | 72.97 | 79.48 | 87.60 | 84.55 | 1.78 |
| | +FAQ | 8 | 92.87 | 91.03 | 73.60 | 81.25 | 88.18 | 85.38 | 0.94 |
| | +QRPE | 8 | **93.27** | **92.06** | **73.82** | **82.16** | **88.52** | **85.97** | **0.36** |

Table 1: Comparison of quantization methods across models and benchmarks. All quantized results are based on GPTQ. "+FAQ" indicates that the calibration data for GPTQ is produced via FAQ. "+QRPE" further applies QRPE preprocessing prior to FAQ-based calibration data generation (i.e., GPTQ+FAQ+QRPE).

redundancy, revise existing strategies, add novel insights, delete outdated entries, and maintain brevity. This ensures that $\mathcal{E}_t$ remains concise and generalizable. As the loop progresses, $\mathcal{E}_t$ accumulates robust prompt strategies that consistently guide the generators to produce high-value calibration data.

Upon completion of the loop, the resulting $\mathcal{E}_N$ serves as a prompt (as shown in Appendix F) within the FAQ process to guide the model in synthesizing a high-quality calibration dataset $\mathcal{D}^*_{\text{quant}}$ , subsequently used for the final model quantization.

**Scope of the Plug-and-Play Claim.** QRPE treats the quantizer as a black box that returns a per-layer reconstruction loss as the reward signal. In this work, we instantiate and validate this design with GPTQ (Frantar et al., 2023), whose Hessian-weighted reconstruction error is computed as a natural byproduct of the quantization pass. Extending QRPE to other reconstruction-based quantizers (e.g., AWQ (Lin et al., 2024), QuaRot (Ashkboos et al., 2024)) only requires exposing an analogous scalar loss; we leave a systematic study of such extensions to future work.

## 4 Experiments

### 4.1 Experimental Setup

To demonstrate the effectiveness and generalizability of QRPE, we conduct comprehensive evaluations across diverse model architectures and reasoning tasks. In our main experiments,

| O | N | I | Select Model | Avg |
|---|---|---|---|---|
| BF16 | | | | 74.78 |
| 3 | 64 | 4 | 8, 32, 30 | 70.45 |
| 3 | 128 | 4 | 8, 32, 30 | 69.17 |
| 3 | 256 | 4 | 8, 32, 30 | 68.73 |
| 4 | 64 | 4 | 8, 14, 32, 30 | 70.84 |
| 5 | 64 | 4 | 4, 8, 14, 32, 30 | 69.10 |
| 5 | 128 | 4 | 4, 8, 14, 32, 30 | 69.00 |
| 5 | 256 | 4 | 4, 8, 14, 32, 30 | 70.02 |
| 3 | 64 | 8 | 8, 32, 30 | 68.63 |
| 3 | 128 | 8 | 8, 32, 30 | 70.06 |
| 3 | 256 | 8 | 8, 32, 30 | 69.10 |
| 4 | 64 | 8 | 8, 14, 32, 30 | 69.51 |
| 4 | 128 | 8 | 8, 14, 32, 30 | 70.02 |
| 4 | 256 | 8 | 8, 14, 32, 30 | **71.53** |
| 5 | 64 | 8 | 4, 8, 14, 32, 30 | 69.95 |
| 5 | 128 | 8 | 4, 8, 14, 32, 30 | 70.29 |
| 5 | 256 | 8 | 4, 8, 14, 32, 30 | 70.14 |

Table 2: Hyperparameter analysis on Qwen3-8B quantized to 4-bit. The BF16 row serves as the full-precision baseline. The best value in each column is shown in **bold**, and the best overall configuration is highlighted in blue .Per-dataset results are reported in Table 6.

we adopt five widely-used common sense reasoning and scientific knowledge benchmarks: ARC-Easy/Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), and PiQA (Bisk et al., 2020). We evaluate them on four representative LLM architectures: Qwen3 [1], Llama3 [2], Phi-4 [3], and MiMO [4]. Detailed implementation information is provided in Appendix B.

### 4.2 Main Results

Table 1 presents our comprehensive evaluation results across four diverse LLM architectures and five common sense reasoning benchmarks. We compare QRPE against the standard GPTQ baseline and the FAQ method under both INT4 and INT8 quantization settings.

[1]https://huggingface.co/Qwen/Qwen3-8B
[2]https://huggingface.co/Jackrong/GPT-5-Distill-llama3.1-8B-Instruct
[3]https://huggingface.co/microsoft/Phi-4-mini-reasoning
[4]https://huggingface.co/XiaomiMiMo/MiMo-7B-RL

**Overall Performance Gains.** Across all experimental configurations, QRPE consistently outperforms both baselines, demonstrating its effectiveness as a universal enhancement for quantization calibration. On average, QRPE achieves 86.89 average accuracy for Qwen3-8B under INT4 quantization (vs. 85.70 for GPTQ), representing a 44.35% reduction in accuracy loss relative to BF16, while under INT8 quantization it delivers near-lossless compression with only 0.23-0.85% average accuracy drop across models, substantially better than FAQ's 0.36-2.34% drop. Most remarkably, for Phi-4-mini-reasoning under INT8 quantization, QRPE achieves best accuracy recovery (72.51 vs. 72.69 BF16). This result validates our hypothesis that evolved prompts can fully capture the model's activation landscape for robust quantization.

**Model-Specific Analysis.**The benefits of QRPE are most pronounced on models that exhibit high sensitivity to quantization. Llama3.1-8B shows catastrophic degradation under vanilla GPTQ (70.05 at INT4, a 7.5 drop), which FAQ partially mitigates (75.10). QRPE further pushes the performance to 76.29, recovering 83.2% of the accuracy loss. This pattern suggests that distilled models like Llama3.1, which have compressed knowledge representations, require more sophisticated calibration strategies to preserve their capabilities under quantization.Phi-4-mini presents an interesting case where FAQ actually harms performance on certain tasks at INT4 (e.g., Arc-E drops from 74.92 to 69.40). We hypothesize this occurs because FAQ's family-generated data may not align well with Phi-4's specialized reasoning pathways. In contrast, QRPE's iterative evolution discovers prompts that better activate the model's unique architectural features, achieving 67.78 average accuracy—a substantial 7.92 improvement over base GPTQ.For well-balanced models like Qwen3-8B and MiMo-7B, which already show relatively modest quantization degradation, QRPE still provides consistent gains. The improvements are more uniform across tasks (0.5-1.5% per benchmark), suggesting that QRPE's evolved prompts provide comprehensive coverage of the activation space rather than just addressing specific weaknesses.

### 4.3 Hyperparameter Analysis: Scaling Behavior

To identify the optimal configuration for QRPE and understand the scaling behavior of its components, we conduct a hyperparameter exploration

| O | N | I | Select Model | Avg |
|---|---|---|---|---|
| BF16 | | | | 74.78 |
| 3 | 64 | 4 | 8, 32, 30 | 72.46 |
| 3 | 128 | 4 | 8, 32, 30 | 72.92 |
| 3 | 256 | 4 | 8, 32, 30 | 72.26 |
| 4 | 64 | 4 | 8, 14, 32, 30 | 72.41 |
| 5 | 64 | 4 | 4, 8, 14, 32, 30 | 72.72 |
| 5 | 128 | 4 | 4, 8, 14, 32, 30 | 72.73 |
| 5 | 256 | 4 | 4, 8, 14, 32, 30 | 72.69 |
| 3 | 64 | 8 | 8, 32, 30 | 72.30 |
| 3 | 128 | 8 | 8, 32, 30 | 72.54 |
| 3 | 256 | 8 | 8, 32, 30 | 72.61 |
| 4 | 64 | 8 | 8, 14, 32, 30 | 72.61 |
| 4 | 128 | 8 | 8, 14, 32, 30 | 72.76 |
| 4 | 256 | 8 | 8, 14, 32, 30 | **73.08** |
| 5 | 64 | 8 | 4, 8, 14, 32, 30 | 72.76 |
| 5 | 128 | 8 | 4, 8, 14, 32, 30 | 72.62 |
| 5 | 256 | 8 | 4, 8, 14, 32, 30 | 72.82 |

Table 3: Hyperparameter analysis on Qwen3-8B quantized to 8-bit. Per-dataset results are reported in Table 7.

| O | N | I | Select Model | #W | Avg |
|---|---|---|---|---|---|
| 4 | 256 | 8 | 8, 14, 32, 30 | 8 | **73.08** |
| 4 | 64 | 8 | 0.6, 1.7, 4, 8 | 8 | 72.04 |
| 4 | 128 | 8 | 0.6, 1.7, 4, 8 | 8 | 72.10 |
| 4 | 256 | 8 | 0.6, 1.7, 4, 8 | 8 | 72.22 |
| 4 | 64 | 8 | 8, 8, 8, 8 | 8 | 72.37 |
| 4 | 128 | 8 | 8, 8, 8, 8 | 8 | 72.24 |
| 4 | 256 | 8 | 8, 8, 8, 8 | 8 | <u>72.45</u> |
| 4 | 256 | 8 | 8, 14, 32, 30 | 4 | **71.53** |
| 4 | 64 | 8 | 0.6, 1.7, 4, 8 | 4 | 69.18 |
| 4 | 128 | 8 | 0.6, 1.7, 4, 8 | 4 | 69.92 |
| 4 | 256 | 8 | 0.6, 1.7, 4, 8 | 4 | 69.79 |
| 4 | 64 | 8 | 8, 8, 8, 8 | 4 | 69.36 |
| 4 | 128 | 8 | 8, 8, 8, 8 | 4 | 69.61 |
| 4 | 256 | 8 | 8, 8, 8, 8 | 4 | <u>70.06</u> |

Table 4: Comparison of selected model layer configurations on Qwen3-8B under int4 and int8 quantization. **Bold** denotes the best value per column within each bit-width group. Underlined <u>values</u> denote the second-best.

across three critical dimensions. $O$: The number of homologous generators used in the rollout phase. $N$: The total number of evolved calibration samples. $I$: The bit-width used to compute the quantization loss (Reward) during the evolution loop. The analysis is performed on Qwen3-8B using challenging mathematical and coding benchmarks (AIME2024/2025 (AIME, 2025), MATH-500 (Lightman et al., 2024), and LiveCodeBench_v5 (Jain et al., 2025)). The results for target 4-bit and 8-bit quantization are summarized in Table 2 and Table 3, respectively.

**Impact of Sample Scaling ($N$).**We observe a consistent scaling trend regarding the sample size $N$. As $N$ increases from 64 to 256, the quantization performance generally improves. This indicates that QRPE is highly data-efficient—achieving substantial precision recovery with only 256 high-quality evolved samples.

**Impact of Ensemble Diversity ($O$).**The composition of the generator ensemble $\mathcal{G}$ significantly influences the quality of the rollout. We compared ensembles of size $O = 3$ (8B, 32B, 30B-A3B), $O = 4$ (adding 14B), and $O = 5$ (adding 4B). The configuration with $O = 4$ generators consistently yields the best results, reaching an average score of 71.53 for 4-bit targets and 73.08 for 8-bit targets. This suggests that incorporating a diverse set of mid-to-large homologous models provides a "hierarchical perspective" that captures both complex reasoning and structural linguistic patterns.

**Impact of Bit-width ($I$).**The parameter $I$ represents the precision of the reward signal used to guide prompt evolution. We explored using both INT4 ($I = 4$) and INT8 ($I = 8$) quantization losses as the guiding metric. A pivotal finding is that $I = 8$ consistently outperforms $I = 4$ as an evaluation proxy, regardless of whether the final target quantization is 4-bit or 8-bit.

Based on the scaling analysis, we identify the $O = 4, N = 256, I = 8$ configuration (highlighted in blue in Table 2 and 3) as the optimal setup. This configuration leverages a balanced homologous ensemble and uses high-precision INT8 loss as a reliable guide for evolving 256 high-quality calibration samples. Consequently, all subsequent main results are reported using this optimized configuration.

### 4.4 Generator Composition Analysis

To evaluate the influence of generator reasoning capability on quantization robustness, we ex-

| Model | Generator | #W | Arc_E | Arc_C | HellaS | WinoG | PiQA | Avg |
|---|---|---|---|---|---|---|---|---|
| Llama3.1-8B | Qwen3-8B | 4 | 84.18 | 74.32 | 56.14 | 55.56 | 77.15 | 69.47 |
| | Qwen3-8B | 8 | 86.07 | 77.30 | 58.47 | 58.72 | 78.62 | 71.84 |

Table 5: Results with Qwen3-8B generator in the QRPE process for Llama3.1-8B.

tend our analysis to the entire Qwen3 model family (0.6B, 1.7B, 4B, 8B, 14B, 32B, and 30B-A3B variants (Yang et al., 2025)), and compare three ensemble configurations on Qwen3-8B: Stronger-Homologous ($O_{strong}$), Weaker-Homologous ($O_{weak}$), and Self-Homologous ($O_{self}$). As summarized in Table 4, the $O_{strong}$ configuration (utilizing models up to 32B) achieves the highest performance. Crucially, the $O_{self}$ configuration (using the 8B target model itself) consistently ranks as the second-best, outperforming the $O_{weak}$ ensemble. This demonstrates that while stronger guidance is optimal, QRPE remains highly effective through self-evolution even when superior homologous models are unavailable. A comprehensive analysis of these results and their implications is provided in Appendix C.

### 4.5 Homologous vs Heterologous Generator

To investigate the cross-architecture transferability of QRPE, we evaluate the performance of Llama3.1-8B using a heterologous generator (Qwen3-8B). As shown in Table 5, this cross-family configuration leads to significant performance degradation compared to the homologous (self-guided) results in Table 1: average accuracy drops from 76.29 to 69.47 for INT4, and from 77.51 to 71.84 for INT8. This substantial gap suggests a "semantic-activation mismatch" between different model families; the reasoning trajectories and internal activation distributions elicited by Qwen3-8B do not effectively align with the specific quantization-sensitive outliers of the Llama-3.1 architecture. These results reinforce the necessity of using homologous models—which share similar pre-training distributions and architectural priors—to ensure that the evolved calibration data remains highly relevant to the target model's internal representation space.

## 5 Conclusions

In this paper, we presented QRPE, a novel quantization-robust preprocessing framework that leverages homologous generator ensembles to iteratively evolve high-quality calibration data. By incorporating quantization loss as a reward signal, QRPE effectively distills semantic prompts that elicit representative activation patterns, bridging the gap between full-precision and low-bit models. Our extensive evaluations across diverse architectures—including Qwen3, Llama3.1, Phi-4, and MiMo—demonstrate that QRPE consistently achieves state-of-the-art results, notably delivering near-lossless INT8 performance and significant accuracy recovery at INT4. Through comprehensive scaling and ablation analyses, we further established the critical importance of homologous guidance in maximizing quantization robustness. Ultimately, QRPE offers a universal and data-efficient solution for the high-fidelity deployment of large language models in resource-constrained environments.

## Limitations

The effectiveness of QRPE can be influenced by specific choices in the iterative evolution process, such as the Refiner LLM, the candidate count per iteration, and the composition of the homologous generator ensemble. While our default configuration demonstrates broad robustness across model families and bit-widths, low-bit regimes (e.g., INT4) inherently introduce volatility. Consequently, the optimal generator composition can shift with the compute budget—for instance, under restricted sample counts a weaker homologous ensemble may occasionally match or exceed a stronger one—suggesting a valuable avenue for future research into adaptive, budget-aware calibration scheduling.

## References

AIME. 2025. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIMEProblemsandSolutions.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman.

2024. Quarot: Outlier-free 4-bit inference in rotated llms. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Zeroq: A novel zero shot quantization framework. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 13166–13175. Computer Vision Foundation / IEEE.

Yuzheng Cai, Siqi Cai, Yuchen Shi, Zihan Xu, Lichao Chen, Yulei Qin, Xiaoyu Tan, Gang Li, Zongyi Li, Haojia Lin, Yong Mao, Ke Li, and Xing Sun. 2025. Training-free group relative policy optimization. *CoRR*, abs/2510.08191.

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2024. Spqr: A sparse-quantized representation for near-lossless LLM weight compression. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. OPTQ: accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Xing Hu, Yuan Cheng, Dawei Yang, Zhixuan Chen, Zukang Xu, Jiangyong Yu, Chen Xu, Zhihang Yuan, Zhe Jiang, and Sifan Zhou. 2025. Ostquant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and Xiaojuan Qi. 2025. Slim-llm: Salience-driven mixed-precision quantization for large language models. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. 2024. Squeezellm: Dense-and-sparse quantization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Raghuraman Krishnamoorthi. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342.

Jung Hyun Lee, Jeonghoon Kim, Se Jung Kwon, and Dongsoo Lee. 2023. Flexround: Learnable rounding based on element-wise division for post-training quantization. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 18913–18939. PMLR.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. 2021. BRECQ: pushing the limit of post-training quantization by block reconstruction. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Guangxuan Xiao, and Song Han. 2024. AWQ: activation-aware weight quantization for on-device LLM compression and acceleration. *GetMobile Mob. Comput. Commun.*, 28(4):12–17.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2025. Spinquant: LLM quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *CoRR*, abs/2106.08295.

Lin Ou, Jinpeng Xia, Yuewei Zhang, Chuzhan Hao, and Hao Henry Wang. 2024. Adaptive quantization error reconstruction for llms with mixed precision. In *First Conference on Language Modeling*.

Irina Proskurina, Guillaume Metzler, and Julien Velcin. 2025. Fair-gptq: Bias-aware quantization for large language models. *CoRR*, abs/2509.15206.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

ModelScope Team. 2024. EvalScope: Evaluation framework for large models.

Boris van Breugel, Yelysei Bondarenko, Paul N. Whatmough, and Markus Nagel. 2025. Fptquant: Function-preserving transforms for LLM quantization. *CoRR*, abs/2506.04985.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR.

Haiyang Xiao, Weiqing Li, Jinyue Guo, Guochao Jiang, Guohua Liu, and Yuewei Zhang. 2026. FAQ: mitigating quantization error via regenerating calibration data with family-aware quantization. *CoRR*, abs/2601.11200.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark W. Barrett, and Ying Sheng. 2024. Sglang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Xingyu Zheng, Haotong Qin, Yuye Li, Haoran Chu, Jiakai Wang, Jinyang Guo, Michele Magno, and Xianglong Liu. 2026. First-order error matters: Accurate compensation for quantized large language models. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pages 28883–28891. AAAI Press.

## Appendix Overview

- Section A: Quantization loss
- Section B: Implementation details
- Section C: Generator composition analysis
- Section D: LLM prompts
- Section E: Experience library case

## A Quantization Loss

### A.1 Problem Formulation

Given a weight matrix $\mathbf{W} \in R^{m \times n}$, the goal of quantization is to find a quantized weight matrix $\mathbf{Q}$ that minimizes the reconstruction error under the constraint that each element of $\mathbf{Q}$ must belong to a discrete set of quantization levels.

### A.2 Hessian-Weighted Loss Function

The quantization loss for each layer is computed as a Hessian-weighted mean squared error:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{(W_{ji} - Q_{ji})^2}{2 \cdot [\mathbf{H}^{-1}]_{ii}} \tag{6}$$

where:

- $\mathbf{W} \in R^{m \times n}$ denotes the original weight matrix with $m$ output features and $n$ input features,
- $\mathbf{Q} \in R^{m \times n}$ denotes the quantized weight matrix,
- $\mathbf{H} \in R^{n \times n}$ denotes the Hessian matrix of the loss function with respect to the weights,
- $[\mathbf{H}^{-1}]_{ii}$ denotes the $i$-th diagonal element of the inverse Hessian matrix,
- $N$ denotes the total number of calibration samples.

### A.3 Per-Element Loss Computation

Following the GPTQ algorithm (Frantar et al., 2023), the loss is accumulated column-by-column during the greedy quantization process:

$$\mathcal{L}_{ji} = \frac{(W_{ji} - Q_{ji})^2}{2 \cdot [\mathbf{H}^{-1}]_{ii}} \tag{7}$$

The total loss for a module is then computed as:

$$\mathcal{L}_{\text{module}} = \frac{1}{N} \sum_{j=1}^{m} \sum_{i=1}^{n} \mathcal{L}_{ji} \tag{8}$$

### A.4 Quantization Constraint

Each quantized weight $Q_{ji}$ is obtained via a quantize-dequantize procedure. The quantization step maps a floating-point weight to a discrete integer:

$$q_{ji} = \text{clamp}\left( \left\lfloor \frac{W_{ji}}{s} \right\rceil + z,\ 0,\ 2^b - 1 \right) \tag{9}$$

and the dequantization step maps the integer back to a floating-point representation:

$$Q_{ji} \in \left\{ s \cdot (q_{ji} - z) \mid q_{ji} \in \{0, 1, \ldots, 2^b - 1\} \right\} \tag{10}$$

where:

- $b$ denotes the number of quantization bits (e.g., $b = 8$ for 8-bit quantization),
- $s \in R^+$ denotes the scaling factor,
- $z \in Z$ denotes the zero-point offset.
- $\lfloor \cdot \rceil$ denotes rounding to the nearest integer,
- $\text{clamp}(x,\ 0,\ 2^b - 1)$ clips $x$ to the range $[0,\ 2^b - 1]$.

### A.5 Group-wise Quantization

When group-wise quantization is employed with group size $g$, the scaling factor $s$ and zero-point $z$ are shared among weights within the same group:

$$\mathbf{W}^{(k)} \in R^{m \times g} \rightarrow \{s_k, z_k\}, \quad k = 1, \ldots, \lceil n/g \rceil \tag{11}$$

### A.6 Interpretation

The Hessian-weighted loss in Equation 6 can be interpreted as a second-order Taylor approximation of the impact of weight quantization on the overall model loss function. Specifically, given a perturbation $\delta\mathbf{w} = \mathbf{Q} - \mathbf{W}$ introduced by quantization, the change in the model loss $\mathcal{L}$ can be approximated as:

$$\delta\mathcal{L} \approx \delta\mathbf{w}^\top \mathbf{g} + \frac{1}{2} \delta\mathbf{w}^\top \mathbf{H}\, \delta\mathbf{w} \tag{12}$$

where $\mathbf{g} = \nabla_{\mathbf{w}} \mathcal{L}$ denotes the gradient and $\mathbf{H} = \nabla^2_{\mathbf{w}} \mathcal{L}$ denotes the Hessian of the loss with respect to the weights. Since the model is assumed to be at or near a local optimum after training, the gradient term vanishes (i.e., $\mathbf{g} \approx \mathbf{0}$), and Equation 12 reduces to:

$$\delta\mathcal{L} \approx \frac{1}{2} \delta\mathbf{w}^\top \mathbf{H}\, \delta\mathbf{w} \tag{13}$$

Under the assumption that the Hessian is diagonal, Equation 13 further simplifies to a sum over individual weight perturbations:

$$\delta\mathcal{L} \approx \frac{1}{2} \sum_i H_{ii} \cdot (\delta w_i)^2 \tag{14}$$

This motivates the per-element loss in Equation 7, where each quantization error $(W_{ji} - Q_{ji})^2$ is weighted by $H_{ii}$, or equivalently divided by $[\mathbf{H}^{-1}]_{ii}$ in the optimal weight update formulation of GPTQ (Frantar et al., 2023).

The diagonal elements of $\mathbf{H}^{-1}$ serve as importance weights, where *smaller* values indicate that

the corresponding weights are more sensitive to quantization errors, as the loss contribution is inversely proportional to $[\mathbf{H}^{-1}]_{ii}$.

In the special case where the Hessian information is not available (failsafe mode), the loss reduces to the standard mean squared error:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{m \cdot n} \sum_{j=1}^{m} \sum_{i=1}^{n} (W_{ji} - Q_{ji})^2 \qquad (15)$$

# B Implementation Details

## B.1 Calibration Data Source.

We utilize the NuminaMath-1.5 dataset as our seed calibration source. This choice is motivated by two key factors: (1) Mathematical reasoning represents a core capability of modern LLMs, making NuminaMath-1.5 highly aligned with practical deployment scenarios; (2) Mathematical problems naturally elicit extended chain-of-thought reasoning, generating longer response sequences with rich intermediate steps. This characteristic helps produce calibration samples with more representative activation distributions compared to short-form QA datasets.

## B.2 Quantization Configuration.

All quantization experiments employ the GPTQ algorithm with group size 128. For QRPE, we use the optimal configuration of o4n256 (4 generators, 256 evolved samples per model), determined through our scaling analysis in 4.3.

## B.3 Infrastructure and Evaluation.

All experiments are conducted on NVIDIA H20-3e GPUs, with each model's quantization and deployment fitting within a single GPU's memory. We leverage the SGLang (Zheng et al., 2024) for efficient model serving and the EvalScope (Team, 2024) toolkit for standardized evaluation. To ensure statistical reliability, we report averaged results over 16-32 runs for each benchmark. Performance is measured using the pass@1 metric, representing the model's single-attempt success rate without multiple sampling.For each invocation of the SGLang deployment service interface, the parameters are consistently set to "temperature": 0.6, "top_k": 20, and "top_p": 0.9, ensuring that the only variable across experiments is the model itself.

# C Generator Composition Analysis

The core of QRPE lies in using a Homologous Generator Ensemble $\mathcal{G}$ to elicit diverse and quantization-sensitive activations. To verify the impact of the ensemble's reasoning capability on the final calibration quality, we compare three distinct configurations on the Qwen3-8B target model:

1. Stronger-Homologous (S-Hom): $\{8B, 14B, 32B, 30B\text{-}A3B\}$, incorporating models with higher parameter counts and stronger reasoning logic.

2. Weaker-Homologous (W-Hom): $\{0.6B, 1.7B, 4B, 8B\}$, consisting of smaller, less capable models.

3. Self-Homogeneous (Self-Hom): $\{8B, 8B, 8B, 8B\}$, using only the target model itself with different sampling temperatures.

The results are summarized in Table 4, providing several key insights into the "teacher-student" dynamics in prompt evolution.

## C.1 The Superiority of Stronger Guidance

As shown in Table 4, the Stronger-Homogeneous configuration achieves the highest performance across all bit-widths and benchmarks. Specifically, in the INT8 setting, it reaches an average score of 73.08, significantly outperforming the weaker ensemble (72.22). This "Stronger-is-Better" effect stems from the semantic complexity of the rollout. Larger models (e.g., 32B or 30B MoE) tend to generate more intricate chain-of-thought reasoning and diverse linguistic structures. Consequently, the resulting calibration parameters are more robust to the precision loss inherent in low-bit quantization.

## C.2 Effectiveness of Self-Evolution

A crucial observation is that the Self-Homogeneous configuration (using the model itself) consistently ranks as the second-best (underlined in Table 4), outperforming the weaker heterogeneous ensemble. For instance, at INT4 with $N = 256$, the self-guided configuration achieves 70.06, whereas the ensemble with smaller models only reaches 69.79. This result is of significant practical importance. It demonstrates that QRPE does not strictly require a "stronger teacher" model to be effective. The target model possesses enough inherent knowledge

to explore its own activation landscape through iterative prompt evolution. This explains why QRPE remains highly effective for cutting-edge or specialized models—such as Llama3.1-8B and Phi-4-mini in our main experiments (Table 1)——where stronger homologous generators may not always be available. In such cases, the model acts as its own guide, refining the experience library to maximize its own quantization robustness.

### C.3 The Pitfall of Weaker Generators

Conversely, the Weaker-Homogeneous configuration yields the lowest performance. We hypothesize that generators with significantly lower reasoning capabilities (e.g., 0.6B or 1.7B) produce repetitive, simplistic, or logically flawed responses. These samples fail to trigger the complex activation patterns that the 8B target model is capable of, leading to "under-fitted" calibration data.

In summary, the ablation study confirms two fundamental design principles for QRPE:

1. Prioritize Strength: Whenever possible, leveraging stronger models in the ensemble provides the most robust semantic signal for evolution.

2. Self-Sufficiency: In the absence of stronger peers, using the target model's own reasoning capability is a robust and effective fallback, ensuring QRPE's broad applicability across any model family.

## D Per-Dataset Results

## E LLM Prompts

We employ the Qwen3-max model as our Refiner LLM, with the prompts for 3.4 and 3.5 presented below.

**Prompt for 3.4: Semantic Advantage Extraction**

# ROLE:
You are a top computational linguist specializing in analyzing the impact of text on neural network activations.
# CONTEXT:
Your task is to evaluate a set of different answers to the same question. High-quality answers can maximize the activation distribution of the model, which is generally positively correlated with lexical rarity, syntactic complexity, logical depth, and information density.
In this task, we use a quantitative score ('score) to measure its potential — **the lower the score, the better the answer performs as calibration data**.
# INPUT:
1. **[Query]:** {{query}}
2. **[List of Answers and their computed scores]:**
{{answers_json}}
# TASK:
1. **Evaluation and Ranking:** Carefully analyze the quality of each Answer (based on vocabulary, syntax, logic, and information density). Then, based on the score (**lower scores are better**), assess its potential as quantization calibration data, and rank this set of Answers from **highest potential to lowest potential**.
2. **Semantic Advantage Extraction:**
- Select **the top-ranked (best Answer)** and **the lowest-ranked (worst Answer)**.
- Perform a detailed comparison and analysis: Explain why the successful answer is better than the unsuccessful one. What key decisions or strategies led to this difference? Produce a natural-language "semantic advantage" text ('A_text') that can be used to optimize future answering strategies.
A_text:

| O | N | I | Select Model | AIME2024 | AIME2025 | MATH-500 | LiveCodeBench | Avg |
|---|---|---|---|---|---|---|---|---|
| BF16 | | | | 76.67 | 72.50 | 94.25 | 55.69 | 74.78 |
| 3 | 64 | 4 | 8, 32, 30 | 70.94 | 64.69 | 94.29 | 51.87 | 70.45 |
| 3 | 128 | 4 | 8, 32, 30 | 70.00 | 61.36 | 93.90 | 51.42 | 69.17 |
| 3 | 256 | 4 | 8, 32, 30 | 70.00 | 59.90 | 94.17 | 50.84 | 68.73 |
| 4 | 64 | 4 | 8, 14, 32, 30 | 71.67 | **66.15** | 94.00 | 51.54 | 70.84 |
| 5 | 64 | 4 | 4, 8, 14, 32, 30 | 70.11 | 61.78 | 94.05 | 50.47 | 69.10 |
| 5 | 128 | 4 | 4, 8, 14, 32, 30 | 69.38 | 61.77 | 94.10 | 50.77 | 69.00 |
| 5 | 256 | 4 | 4, 8, 14, 32, 30 | 70.63 | 64.16 | 94.15 | 51.16 | 70.02 |
| 3 | 64 | 8 | 8, 32, 30 | 68.96 | 59.58 | **94.40** | 51.58 | 68.63 |
| 3 | 128 | 8 | 8, 32, 30 | 71.04 | 63.13 | 93.94 | 52.13 | 70.06 |
| 3 | 256 | 8 | 8, 32, 30 | 71.46 | 60.63 | 93.98 | 50.36 | 69.10 |
| 4 | 64 | 8 | 8, 14, 32, 30 | 70.83 | 61.14 | 94.29 | 51.76 | 69.51 |
| 4 | 128 | 8 | 8, 14, 32, 30 | 72.81 | 62.64 | 93.78 | 50.84 | 70.02 |
| 4 | 256 | 8 | 8, 14, 32, 30 | **73.09** | 66.07 | 94.16 | **52.80** | **71.53** |
| 5 | 64 | 8 | 4, 8, 14, 32, 30 | 70.52 | 63.65 | 94.06 | 51.56 | 69.95 |
| 5 | 128 | 8 | 4, 8, 14, 32, 30 | 71.67 | 63.10 | 94.39 | 51.99 | 70.29 |
| 5 | 256 | 8 | 4, 8, 14, 32, 30 | 70.21 | 64.48 | 94.21 | 51.65 | 70.14 |

Table 6: Full per-dataset breakdown of Table 2 (INT4)

Prompt for 3.5: Experience Library Evolution

# ROLE:
You are a top-tier AI strategy expert, akin to a senior programmer maintaining a codebase, responsible for iterating and optimizing an Experience Library that guides AI behavior. Your core objectives are to keep the library concise, non-redundant, generalizable, and actionable.
# PRINCIPLES (Maintenance Guidelines):
1. **Merge Similar Entries:** If a new lesson is similar to an existing experience, merge them and distill a more general rule.
2. **Revise Existing Entries:** If a new lesson improves or corrects an existing experience, modify that entry directly.
3. **Add Necessary New Entries:** If a new lesson is entirely new and valuable, add it as a new experience.
4. **Delete Outdated Entries:** (If applicable) If the new lesson shows that an existing experience is wrong, remove it.
5. **Maintain Brevity:** The final Experience Library must not contain any duplicate or contradictory content, and the total number of experience entries must not exceed 10.
# INPUT:
1. **[Current Experience Library (E_current)]:**
{{E_current}}
2. **[Newly Discovered Lesson (A_text)]:**

# TASK:
Your task is to use the "Newly Discovered Lesson" and follow the above Maintenance Guidelines to iterate on the "Current Experience Library" and output a **complete, standalone, optimized new version 'E_new'**. 'E_new' will fully replace the previous version in the next round.
Make sure the newly generated Experience Library does not contain overly obscure words, and that the sentences remain smooth and unambiguous.
Produce your output strictly in the following format, without any additional conversation or explanation:
<THOUGHT>
Show your step-by-step thought process here:
1. Analyze the core idea(s) in A_text.
2. Compare A_text with each entry in E_current.
3. Decide which operation to take (merge, revise, add, or no action).
4. Draft the final new version of the Experience Library and verify it meets all principles.
</THOUGHT>
<FINAL_EXPERIENCE_LIBRARY>
1. [Optimized first experience]
2. [Optimized second experience]
...
n. (If applicable)[New or revised experience derived from A_text]
</FINAL_EXPERIENCE_LIBRARY>

| O | N | I | Select Model | AIME2024 | AIME2025 | MATH-500 | LiveCodeBench | Avg |
|---|---|---|---|---|---|---|---|---|
| BF16 | | | | 76.67 | 72.50 | 94.25 | 55.69 | 74.78 |
| 3 | 64 | 4 | 8, 32, 30 | **75.52** | 65.00 | 94.29 | 55.03 | 72.46 |
| 3 | 128 | 4 | 8, 32, 30 | 74.90 | 67.91 | 94.32 | 54.57 | 72.92 |
| 3 | 256 | 4 | 8, 32, 30 | 74.48 | 65.32 | 94.14 | 55.11 | 72.26 |
| 4 | 64 | 4 | 8, 14, 32, 30 | 73.75 | 66.67 | 94.27 | 54.96 | 72.41 |
| 5 | 64 | 4 | 4, 8, 14, 32, 30 | 74.17 | 67.50 | 94.30 | 54.92 | 72.72 |
| 5 | 128 | 4 | 4, 8, 14, 32, 30 | 75.10 | 65.94 | 94.30 | 55.60 | 72.73 |
| 5 | 256 | 4 | 4, 8, 14, 32, 30 | 74.31 | 66.66 | 94.41 | 55.37 | 72.69 |
| 3 | 64 | 8 | 8, 32, 30 | 73.54 | 66.87 | 94.16 | 54.64 | 72.30 |
| 3 | 128 | 8 | 8, 32, 30 | 73.13 | 67.92 | 94.18 | 54.94 | 72.54 |
| 3 | 256 | 8 | 8, 32, 30 | 74.27 | 66.67 | 94.10 | 55.39 | 72.61 |
| 4 | 64 | 8 | 8, 14, 32, 30 | 74.06 | 66.46 | 94.28 | **55.65** | 72.61 |
| 4 | 128 | 8 | 8, 14, 32, 30 | 74.76 | 66.90 | 94.25 | 55.13 | 72.76 |
| 4 | 256 | 8 | 8, 14, 32, 30 | 75.14 | **68.33** | 93.95 | 54.92 | **73.08** |
| 5 | 64 | 8 | 4, 8, 14, 32, 30 | 75.10 | 66.35 | 94.28 | 55.30 | 72.76 |
| 5 | 128 | 8 | 4, 8, 14, 32, 30 | 74.27 | 67.08 | **94.45** | 54.70 | 72.62 |
| 5 | 256 | 8 | 4, 8, 14, 32, 30 | 74.27 | 68.02 | 94.33 | 54.68 | 72.82 |

Table 7: Full per-dataset breakdown of Table 3 (INT8)

## F Experience Library Case

Here we present the final contents of the Experience Library in the QRPE algorithm.

**Experience Library of Qwen3-8B for QRPE O4N256I8**

1. In geometric and kinematic problems, anchor reasoning in direct physical or geometric correspondence (e.g., rolling without slipping implies path length equals rotations times base circumference) to derive minimal, exact equations. Immediately fix the configuration using explicit constraints to set scale and orientation unambiguously, choosing coordinates aligned with intrinsic symmetries. Decompose complex figures into additive components (areas, vectors) to simplify analysis. Avoid introducing extraneous parameters or hidden periodicities not implied by the problem statement—treat explicit event descriptions (e.g., "first returns after 17 rotations") as literal, single-cycle conditions. Validate assumptions through direct computation, use early symbolic simplifications, prioritize elementary axioms, and assert exact values when justified.
2. In problems blending discrete and geometric constraints, identify robust invariants—especially parity or modular conditions—that yield sharp numerical bounds or contradictions (e.g., requiring a sum to be even yet exceed a fixed total). Use these invariants to derive global obstructions before constructing solutions. Crucially, re-evaluate whether the same invariant produces a contradiction when problem parameters change (e.g., board size); avoid over-generalizing conclusions across instances

inductive or direct proof that establishes correctness and optimality simultaneously. Recast the problem into its most revealing framework (e.g., factored products, squared sums, residue systems) and prioritize transformations that collapse complexity over brute-force computation.
4. For combinatorial problems, translate constraints into their exact structural implications as early as possible—such as recognizing that a strict ordering condition enforces a rooted tree—and immediately map the problem to the corresponding canonical model or counting formula. Validate with minimal cases (e.g., $n = 2, 3$) and avoid overgeneralizing into irrelevant classes. Use explicit constructions when the structure is clear; otherwise, prefer global contradiction arguments.
5. For problems requiring construction of maximal or optimal discrete configurations, begin by proposing a candidate solution guided by symmetry or geometric naturalness (e.g., midpoints or centroids). Decompose the problem by verifying feasibility, isolating critical boundaries, and analyzing objective behavior. Partition verification into essential interaction categories, validate with representative cases, and provide a decisive maximality argument showing any extension violates constraints.
6. In problems requiring enumeration of solutions under explicit constraints, derive a complete set of necessary and sufficient symbolic conditions first, then use them to restrict the candidate space to only viable options. Validate each candidate only once against the original constraints, avoiding repetitive checks or redundant intermediate verifications. Preserve exact expressions throughout and confirm correctness through minimal, decisive comparisons—not exhaustive re-evaluation.
7. In constrained inequality or optimization problems, first recognize and exploit structural symmetries or interdependencies to reduce dimensionality through judicious variable substitution (e.g., $S = x + y$, $D = x - y$). Reformulate the objective by leveraging inherent algebraic structure—particularly quadratic forms—using discriminant analysis or vertex properties to derive sharp bounds. Then proceed in distinct phases: (1) global constraint synthesis, (2) aggressive pruning of infeasible regions using dominant bounds, and (3) minimal verification of remaining candidates via a single representative instance. Avoid unfocused trials, redundant algebra, or late formulation of the core objective; prioritize early abstraction that reveals the problem's essential structure and enables a concise, globally valid optimality proof.
8. Deliver solutions as a single, modular, linear exposition that maximizes information density and expository efficiency: decompose the problem immediately into clean subproblems, use algebraic generalizations (e.g., $a^m \cdot a^n = a^{m+n}$) to avoid brute-force arithmetic, and work exclusively in exact symbolic form—never resorting to decimal approximations or mixed-number speculation. Eliminate all metacognitive noise ("Wait...", "Maybe I'm wrong...") and redundant self-verification; every sentence must either advance the deduction, clarify a transformation, or confirm a necessary condition. Structure the argument irreversibly from premise to conclusion, asserting consequences with confidence and embedding validation as concise, integrated checks—not repetitive afterthoughts.

**Experience Library of GPT-5-Distill-llama3.1-8B-Instruct for QRPE O4N256I8**

1. Begin problem-solving by stating relevant general principles—including conservation laws (e.g., net displacement = total forward  total backward) and invariants—and define variables anchored to elementary, repeatable actions (e.g., "Let $x$ = backward distance per gust"). Use strategic normalization (e.g., setting total sum to 1 in partition problems) when it preserves generality and simplifies analysis. Express aggregate effects by scaling unit behaviors linearly, then equate the resulting algebraic sum to the observed outcome to ensure physical fidelity and generalizability.

2. Start by accurately parsing the problem's mathematical and spatial structure: identify all given quantities, constraints, and conditions—including whether values are cumulative or per-step, and whether thresholds apply to totals or individuals. Maintain strict unit and type consistency, avoid unwarranted assumptions, and derive conclusions through stepwise logical reasoning. Always verify the final answer by substituting it back into the original conditions to confirm correctness.
3. When symmetry is broken or constraints must be tightly satisfied, construct solutions by focusing on extremal elements—such as maximum/minimum values, boundary points, or outlier cases—and apply balanced, pairwise, or limiting reasoning to the remaining structure to meet all conditions.
4. When confronting opaque recursive rules or iterative processes, first compute initial terms explicitly to detect empirical patterns—such as periodicity, symmetry, or invariant sums—before attempting abstract formalism. Only after observing concrete behavior should you decompose the problem symbolically or structurally. Mirror real-world measurement processes when needed (e.g., count events $\rightarrow$ compute volume $\rightarrow$ convert units), delay unit conversions when safe, and maintain logical continuity through natural conditional phrasing ("If… then…"), active verbs ("assume," "infer," "conclude"), and consistent notation.
5. Anchor reasoning in precise, domain-specific definitions and explicit user instructions. Rigorously maintain type discipline—distinguishing symbols from numeric values—and never perform arithmetic on symbolic labels unless defined by the task. Avoid conflating representation with value or violating instruction-specified mappings.
6. When a problem lacks sufficient information for a definitive conclusion, avoid asserting unverifiable claims or circular reasoning. Instead, explicitly acknowledge the missing information and pivot to providing a clear, general method for verifying the desired property—using precise definitions, step-by-step criteria, and illustrative examples.
7. Model the space of possible solutions explicitly using well-defined mathematical structures—such as sets, prime factorizations, or parametric forms—and analyze their inherent constraints (e.g., cardinality, mutual exclusivity, or invariants like minimum prime exponents in GCDs) to deduce necessary outcomes. Favor this structural reasoning over brute-force enumeration, ensuring symbolic representations capture essential relationships and eliminate redundant or infeasible cases.
8. When evaluating mathematical expressions involving multiple operations (e.g., radicals and division), explicitly enumerate all validity constraints—such as non-negative radicands, nonzero denominators, and domain restrictions—and resolve their logical conjunction. The final domain or solution set is governed by the strictest combined condition, not individual requirements in isolation. Always verify that candidate values satisfy every operational constraint simultaneously.
9. Begin responses with a correct, problem-specific transformation that reveals underlying structure—such as applying a known identity, factoring, or recognizing a telescoping pattern—to activate deep symbolic reasoning early. Avoid starting with unsimplified repetition or generic placeholders; even brief initial precision significantly improves reasoning quality and information density.
10. Ensure every claim introduces a non-trivial, falsifiable constraint that meaningfully narrows the solution space—never rely on tautologies or definitional truths. Support key assertions with concrete examples or direct verification, prioritizing specificity and discriminative power over generic restatements.