# CoNestQ: Coordinated Nested Quantization for Low-Bit LLM Deployment

**Anonymous authors**
Paper under double-blind review

## Abstract

Low-latency and data-locality requirements are shifting LLM inference from centralized clouds toward distributed, multi-tier architectures such as the Intelligence Delivery Network (IDN), which distributes intelligence across cloud, regional, edge, and local nodes on demand. Serving such heterogeneous nodes economically requires a single model representation from which multiple precision levels can be derived. Fixed-precision quantization cannot meet diverse node budgets without storing a separate model per bit width, while existing nested methods leave the cross-bit coupling induced by representation sharing largely unaddressed, with several bit-widths unusable on some models. To address these limitations, we propose CoNestQ (Coordinated Nested Quantization), a post-training quantization framework that derives all target precisions from a shared parent representation and coordinates the three coupled stages that prior nested methods address only partially: (i) joint quantization-parameter calibration across target bit-widths, (ii) balanced parent-code selection guided by normalized excess error, and (iii) branch-wise error compensation with periodic cross-precision synchronization. Across the full 3–8-bit range on four LLMs, CoNestQ attains a 100% validity rate with strong low-bit robustness. Relative to the non-linear nested baseline, it reduces multi-precision storage by 26–34%, lowers peak GPU memory by about 38% on average, and improves inference throughput by about 16%; relative to six independently quantized models, multi-precision storage shrinks by 5.5–6.2×.

## 1 Introduction

Large language models (LLMs) impose substantial computational and memory costs during inference, while heterogeneous devices and service requests often demand different trade-offs among model quality, latency, and resource consumption (Xiao et al. (2023); Jiang et al. (2024); Sheng et al. (2023)). A single fixed-precision model is therefore insufficient for diverse deployment environments(Park et al. (2024; 2025)). Emerging architectures such as the Intelligence Delivery Network (IDN) (Wang et al. (2026)), which distributes intelligence across cloud, regional, edge, and local nodes on demand, further highlight the need to derive multiple precision levels from a unified model representation.

Multi-precision quantization supports multiple bit widths within a unified framework, but maintaining separate models for each precision incurs substantial storage overhead. Nested quantization reduces this cost by sharing a high-bit integer representation, from which lower-precision sub-models are deterministically derived via nested mapping and dequantization (Yu et al. (2021); Nair et al. (2025)). However, the sharing tightly couples quantization decisions across bit widths, preventing each precision from being optimized independently. The central challenge is therefore to preserve reconstruction quality across all target bit widths under the strict nesting constraint.

Existing quantization methods such as GPTQ (Frantar et al. (2022)), OmniQuant (Shao et al. (2024)), and QuaRot (Ashkboos et al. (2024)) are optimized for a single precision configuration and do not construct a representation shared across bit-widths. Nested approaches go further but retain important limitations. Any-Precision LLM (Park et al. (2024)) refines from low-bit clustering, so early partitioning decisions restrict the feasible choices at later bit-widths. MatGPTQ (Kleinegger et al. (2026)) jointly selects parent codes but propagates averaged residuals across bit-widths,



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.

which can obscure their differing error scales and compensation requirements. Together, these observations expose three key issues: (1) how to optimize shared quantization parameters for multiple precisions; (2) how to balance heterogeneous errors when selecting parent codes; and (3) how to preserve bit-width-specific compensation trajectories during error propagation.

To address these challenges, we propose CoNestQ (Coordinated Nested Quantization), a post-training framework for nested quantization specifically designed to coordinate the three coupled stages underlying these limitations: grid calibration, parent-code selection, and error propagation. First, joint quantization-parameter calibration selects shared quantization parameters through grid search by minimizing the sum of weight reconstruction errors across the target bit-widths. Second, balanced parent-code selection evaluates each candidate using the worst normalized excess error across branches, the total absolute reconstruction error, and a soft error-ordering regularizer. Third, branch-wise error compensation preserves distinct compensation states within each compensation window, updates subsequent quantization targets using second-order information, and synchronizes the remaining states at window boundaries. Together, these components coordinate grid calibration, code assignment, and error propagation while producing a single parent-code weight representation and shared quantization parameters.

We evaluate CoNestQ across five LLMs and various bit-widths, covering quantization quality, multi-precision usability, and serving efficiency. CoNestQ preserves model quality across precisions, achieves strong low-bit robustness, and reduces multi-precision storage overhead with minimal inference overhead. Ablation studies further verify the contribution of each component.

Our contributions are (1) We identify the coupling among shared quantization parameters, parent-code selection, and error propagation as the key challenge in nested multi-bit quantization. (2) We propose CoNestQ, which combines joint quantization-parameter calibration, balanced parent-code selection, and branch-wise error compensation to improve multi-bit reconstruction under a shared representation. (3) Extensive experiments across multiple LLMs and bit-widths demonstrate robust cross-bit quality and efficient multi-precision serving.

## 2 Background and Related Work

Despite their strong capabilities, large language models (LLMs) incur substantial storage and inference costs due to their massive parameter sizes. Quantization reduces the precision of weights or activations, e.g., to INT8 or INT4, offering an effective trade-off between model quality and efficiency (Shao et al. (2024); Liu et al. (2024a)). Existing studies mainly follow two directions: Quantization-Aware Training (QAT) ((Liu et al. (2024b); Chen et al. (2025))) and Post-Training Quantization (PTQ) (Lin et al. (2024); Ding et al. (2025)).

**Post-Training Quantization.** Unlike QAT, which requires additional training or fine-tuning, PTQ directly compresses pretrained models at much lower offline cost (Zhu et al. (2024)). Representative methods include GPTQ (Frantar et al. (2022)), which uses second-order information for error compensation; OmniQuant (Shao et al. (2024)), which learns clipping ranges and equivalent transformations; and QuaRot (Ashkboos et al. (2024)), which applies invariant rotations to suppress outliers and improve quantizability. Although effective at individual target precisions, these methods typically optimize each precision independently and do not enforce a shared integer representation across bit-widths.

**Multi-Precision and Nested Quantization.** Multi-precision quantization enables a model to support multiple target bit widths, while nested quantization, as a specialized form, derives these precisions from a shared representation. Any-Precision LLM (Ashkboos et al. (2024)) progressively refines low-bit clusters into higher-bit representations using nested indices and bit-width-specific centroids, but its hierarchical construction makes later stages dependent on earlier partitions. MatQuant (Nair et al. (2025)) jointly optimizes nested integer codes across bit-widths under a shared parent representation. MatGPTQ (Kleinegger et al. (2026)) further brings this idea to one-shot PTQ by selecting shared parent codes with a multi-bit reconstruction objective and using GPTQ-based residuals for error compensation.

Despite these advances, existing methods primarily focus on constructing nested representations, while the cross-bit coupling introduced by representation sharing remains underexplored. A shared quantization grid and parent code may not simultaneously satisfy the distinct reconstruction require-

ments of all target bit widths, and the resulting branch-specific errors are difficult to correct with a single shared compensation signal. These factors are further coupled: the quantization grid constrains parent-code selection, and together they determine the errors to be compensated. CoNestQ addresses this coupling through joint quantization-parameter calibration, balanced parent-code selection, and branch-wise error compensation, each developed in Section 3, enabling a shared parent representation to better accommodate multiple bit widths.

# 3 METHODOLOGY

We propose CoNestQ (Coordinated Nested Quantization), a post-training quantization method for multi-bit-width deployment. CoNestQ stores a single integer weight matrix at the highest target bit-width and deterministically derives lower-bit representations through nested mapping. To account for the cross-bit coupling induced by this shared representation, CoNestQ coordinates its construction and reconstruction through Joint Quantization-Parameter Calibration, Balanced Parent-Code Selection, and Branch-Wise Error Compensation.

## 3.1 PROBLEM FORMULATION

We first introduce the nested quantization representation used by CoNestQ and then formulate the joint multi-bit-width reconstruction problem under nested constraints.

Let $W \in \mathbb{R}^{d_o \times d_i}$ denote the original full-precision weight matrix of a linear layer, and $X \in \mathbb{R}^{d_i \times N}$ its corresponding calibration inputs.

### 3.1.1 NESTED QUANTIZATION REPRESENTATION

Let $\mathcal{B} = \{b_1 < b_2 < \cdots < b_K = M\}$ be the target bit-width set, with $M = 8$. The corresponding $M$-bit parent-code set is $\mathcal{C}_M = \{0, 1, \ldots, 2^M - 1\}$. CoNestQ maintains a single $M$-bit parent integer weight matrix $Q$, where each element $q \in \mathcal{C}_M$ represents an individual parent code.

For $b \in \mathcal{B}$, let $\kappa_b = 2^{M-b}$. The nested mapping from parent codes to $b$-bit child codes is defined as

$$Q^{(b)} = \rho_b(Q) = \mathrm{clip}_{[0,\, 2^b - 1]} \left( \left\lfloor \frac{Q}{\kappa_b} + \frac{1}{2} \right\rfloor \right). \tag{1}$$

For $b < M$, the mapping projects parent codes onto the lower-bit integer grid. When $b = M$, $\kappa_M = 1$ and $\rho_M(Q) = Q$. Under nested quantization, the dequantized $b$-bit weights are defined as

$$\widehat{W}^{(b)}(Q, s, z) = s \left( \kappa_b Q^{(b)} - z \right). \tag{2}$$

Thus, a single set of parent integer weights $Q$ and quantization parameters $(s, z)$ uniquely determines the integer representations $\{Q^{(b)}\}_{b \in \mathcal{B}}$ and dequantized weights $\{\widehat{W}^{(b)}\}_{b \in \mathcal{B}}$ at all target bit-widths, eliminating the need to store independent quantized weights for each target bit-width.

### 3.1.2 JOINT OPTIMIZATION ACROSS BIT-WIDTHS UNDER NESTED CONSTRAINTS

Although the shared parent-code representation reduces storage redundancy in multi-bit-width quantized models, it also couples quantization decisions across bit-widths. Since all target bit-widths are derived from the same parent-code matrix, updating a parent code simultaneously affects multiple bit-width branches, preventing them from being optimized independently.

Following the output reconstruction objective of GPTQ, we define the layer output reconstruction error for target bit-width $b \in \mathcal{B}$ as

$$\mathcal{L}_b(\widehat{W}) = \left\| WX - \widehat{W}^{(b)} X \right\|_F^2. \tag{3}$$

The overall objective of CoNestQ is to simultaneously minimize the reconstruction errors of all target bit-widths under the nested representation constraints, which can be formulated as

$$\begin{aligned} \min_{Q, s, z} \quad & \left( \mathcal{L}_b\left( \widehat{W}^{(b)} \right) \right)_{b \in \mathcal{B}} \\ \text{s.t.} \quad & (1), (2). \end{aligned} \tag{4}$$

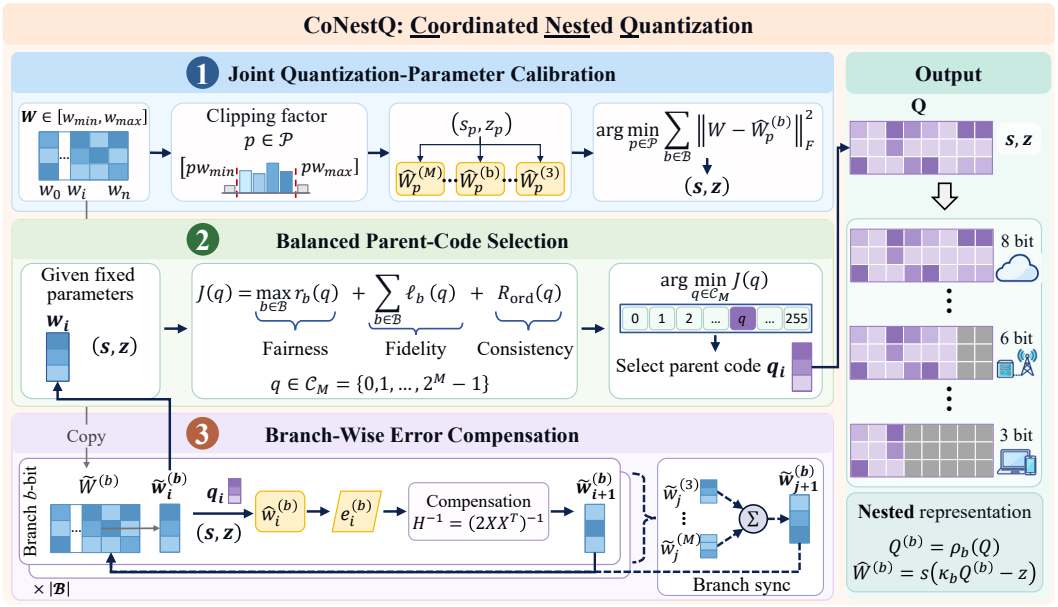



Figure 1: Overview of CoNestQ. During branch-wise error compensation, branch-specific errors are propagated, followed by branch synchronization via Branch sync after each compensation window.

Eq. (4) describes a multi-objective optimization problem with coupled mixed variables under nested constraints. The optimization variables include both the discrete variable $Q, z$ and the continuous variable $s$. Meanwhile, rounding, clipping, and the shared parent-code mapping couple the target bit-widths, making direct joint optimization difficult. To obtain a tractable solution, we decompose the joint optimization into coordinated subproblems for shared quantization-grid construction, parent-code selection, and reconstruction-error correction, which capture the key sources of cross-bit coupling in the nested quantization process.

### 3.2 CoNestQ

To approximately solve the joint multi-bit-width reconstruction problem under nested constraints, we propose CoNestQ, a PTQ method with three components: Joint Quantization-Parameter Calibration, Balanced Parent-Code Selection, and Branch-Wise Error Compensation, which jointly optimize a single parent-code matrix for reconstruction across all target bit-widths.

As shown in Figure 1, given full-precision weights $W$, calibration data $X$ and target bit-widths $\mathcal{B}$, CoNestQ outputs a nested quantization model consisting of a nested parent-code $Q$ and quantization parameters $(s, z)$, which supports efficient reconstruction at different target bit-widths. First, Joint Quantization-Parameter Calibration selects shared quantization parameters by evaluating candidate clipping ranges according to reconstruction errors across all target bit-widths and fixes them thereafter. Then, Balanced Parent-Code Selection determines shared parent codes based on local reconstruction errors across bit-width branches. Finally, Branch-Wise Error Compensation propagates branch-specific quantization errors to subsequent weights and synchronizes branch states at the end of each compensation window. This alternating process continues until the complete parent-code matrix is constructed.

#### 3.2.1 Joint Quantization-Parameter Calibration

In nested quantization, quantization parameters obtained by the min-max scheme for one bit width may be suboptimal for others. CoNestQ introduces joint quantization-parameter calibration, inspired by prior clipping methods (Liu et al. (2022); Shao et al. (2024)), which jointly searches the weight dynamic range across bit widths to obtain shared $(s, z)$ for multiple precisions.

Let $w_{\min}$ and $w_{\max}$ denote the minimum and maximum values of $W$, respectively. We consider clipping factors $\mathcal{P} = \{0.80, 0.81, \ldots, 1.00\}$, each applied uniformly to both quantization bounds. For $M$-bit quantization, the corresponding candidate quantization parameters are given by

$$s_p = \frac{pw_{\max} - pw_{\min}}{2^M - 1}, \qquad z_p = \mathrm{clip}_{[0,2^M-1]}\left(\left\lfloor -\frac{pw_{\min}}{s_p} \right\rceil\right). \tag{5}$$

Decreasing $p$ improves quantization resolution but increases clipping error, so searching over $\mathcal{P}$ balances the two effects.

For each $(s_p, z_p)$, we apply the uniform quantization rule to obtain the temporary parent codes $Q_p^{(b)}$, and use nested quantization rule Eq.(1) and (2) to obtain the corresponding dequantized weights $\widehat{W}^{(b)}p$. The shared quantization parameters are selected by

$$\arg\min_{p\in\mathcal{P}} \sum_{b\in\mathcal{B}} \left\| W - \widehat{W}_p^{(b)} \right\|_F^2. \tag{6}$$

The parameters associated with the optimal $p$ are used as the final shared $(s, z)$, and the temporary parent codes are used only for parameter calibration.

### 3.2.2 Balanced Parent-Code Selection

With $(s, z)$ fixed, CoNestQ selects a unique parent code $q \in \mathcal{C}_M$ for each full-precision weight $w$. Since different bit-widths may prefer different parent codes, no single code can generally minimize all branch-wise quantization errors simultaneously. We therefore select $q$ by minimizing a balanced objective across bit-widths:

$$J(q) = \max_{b\in\mathcal{B}} r_b(q) + \lambda \sum_{b\in\mathcal{B}} \ell_b(q) + \mu R_{\mathrm{ord}}(q) \tag{7}$$

where $\lambda, \mu \geq 0$, with $\lambda = 0.1$ and $\mu = 0.05$ in experiments. The first two terms jointly balance branch-wise fairness and overall quantization accuracy, while the last term encourages the selected code to respect the expected error ordering across bit-widths. Specifically, for a candidate $q$, we define its local quantization error at $b$-bit and normalized excess over the branch-wise optimum as

$$\ell_b(q) = \left[w - s\left(\kappa_b \rho_b(q) - z\right)\right]^2, \qquad r_b(q) = \frac{\ell_b(q) - \ell_b^{\min}}{\max(\ell_b^{\min}, \varepsilon)}, \quad \ell_b^{\min} = \min_{q'\in\mathcal{C}_M} \ell_b(q'), \tag{8}$$

where $\varepsilon > 0$. The normalized excess $r_b(q)$ measures the relative deviation from the branch-wise optimum; thus, $\max_b r_b(q)$ promotes cross-bit fairness, while $\sum_b \ell_b(q)$ preserves overall quantization accuracy. The remaining term introduces a structural prior on the relation between errors at different resolutions. For ordered bit-widths $b_1 < \cdots < b_K$, we define

$$R_{\mathrm{ord}}(q) = \sum_{k=2}^{K} \frac{\left[\ell_{b_k}(q) - \ell_{b_{k-1}}(q)\right]_+}{\max\left(\ell_{b_{k-1}}(q), \varepsilon\right)}, \qquad [u]_+ = \max(u, 0), \tag{9}$$

which softly penalizes candidates that violate the expected error ordering induced by bit-width resolution. Since $\mathcal{C}_M$ is finite, the optimal parent code can be obtained efficiently by enumeration. With the default $M = 8$, only $2^8$ candidates need to be evaluated for each weight.

### 3.2.3 Branch-Wise Error Compensation

Based on GPTQ (Frantar et al. (2022)), CoNestQ designs branch-wise error compensation to reduce reconstruction error. Since different bit widths incur distinct quantization residuals, fully shared propagation can obscure branch-specific errors, while fully independent propagation may weaken cross-bit consistency. CoNestQ therefore preserves branch-specific compensation locally and periodically aggregates branch states to balance error specialization with cross-bit coordination.

For each $b \in \mathcal{B}$, we maintain compensated weights $\widetilde{W}^{(b)}$, initialized from $W$, and use a shared Hessian $H$. Branch states are aggregated every $G$ input channels, forming a compensation window. Given $(s, z)$, Balanced Parent-Code Selection uses $\{\widetilde{w}_i^{(b)}\}_{b\in\mathcal{B}}$ to determine the shared parent code $q_i = Q_{:,i}$. Its reconstruction and branch-wise error are

$$\widehat{w}_i^{(b)} = s\left(\kappa_b \rho_b(q_i) - z\right), \quad e_i^{(b)} = \widetilde{w}_i^{(b)} - \widehat{w}_i^{(b)}. \tag{10}$$

Following GPTQ, each branch propagates its error to subsequent unquantized weights:

$$\widetilde{w}_j^{(b)} \leftarrow \widetilde{w}_j^{(b)} - \frac{e_i^{(b)}}{[H^{-1}]_{ii}}[H^{-1}]_{ij}, \qquad j > i, b \in \mathcal{B}. \tag{11}$$

The updated branch states then serve as the reconstruction targets for subsequent parent-code selection. At the end of each compensation window, the branch states are aggregated and subsequently synchronized to the subsequent weights:

$$\overline{w}_j = \frac{1}{|\mathcal{B}|}\sum_{b\in\mathcal{B}} \widetilde{w}_j^{(b)}, \quad \widetilde{w}_r^{(b)} \leftarrow \overline{w}_j, \qquad b \in \mathcal{B}, j < r. \tag{12}$$

Thus, branch-wise propagation retains local bit-width-specific corrections, while periodic aggregation prevents persistent divergence across branches.

# 4 EXPERIMENTS

**Models and Datasets.** We evaluate CoNestQ on four LLMs from diverse model families and training variants: Meta-Llama-3-8B-Instruct (Grattafiori et al. (2024)), Qwen2.5-14B-Instruct (Qwen et al. (2025)), and the Qwen3 Base models (8B and 14B) (Yang et al. (2025)). We report perplexity (PPL) on WikiText-2 (Merity et al. (2016)), C4 (Raffel et al. (2020)), and PTB (Marcus et al. (1994)) for language modeling, and zero-shot accuracy on PIQA (Bisk et al. (2020)), ARC-Challenge (Clark et al. (2018)), and HellaSwag (Zellers et al. (2019)) for downstream evaluation. For each category, we additionally report the arithmetic mean over the three datasets.

**Implementation.** We focus on weight-only linear quantization and evaluate all methods over the target bit-width range $\mathcal{B} = \{3, 4, 5, 6, 7, 8\}$. For all linear quantization methods, we use a group size of 128. For CoNestQ, the compensation window size is also set to $G = 128$. We use 600 samples from C4 for calibration and adopt a single set of hyperparameters across all evaluated models. All experiments are conducted on a single NVIDIA H100 GPU under the same software environment.

**Baselines.** We compare CoNestQ with three representative baselines covering fixed-precision and multi-precision quantization. GPTQ serves as the fixed-precision baseline, with a separate model independently quantized for each target bit-width. Any-Precision LLM (APL) represents non-linear multi-precision quantization based on nested clustering and bit-width-specific centroids, providing a non-linear counterpart to CoNestQ's linear integer representation. MatGPTQ is the most directly comparable recent baseline, as it likewise constructs nested linear integer representations within a one-shot PTQ framework. Together, these baselines cover independently optimized fixed-precision, non-linear shared multi-precision, and shared linear multi-precision quantization.

## 4.1 QUANTIZATION QUALITY

**Perplexity.** Table 1 compares the PPL of four quantization methods across six bit-widths. CoNestQ consistently achieves low and stable PPL across models and precision levels, demonstrating effective language-modeling preservation under multi-precision quantization. Among nested methods, CoNestQ achieves the best or within 1% of the best PPL in 56.7% of all configurations and 94.4% of Qwen configurations. Compared with MatGPTQ, CoNestQ matches or improves PPL in 70% of configurations, with more pronounced gains at 3 bits, reducing PPL by 16.3% on average and up to 41.1%. These results highlight CoNestQ's strong cross-model robustness and low-bit advantage. In contrast, GPTQ and APL exhibit model-dependent degradation under challenging settings. GPTQ suffers severe low-bit degradation on Llama-3-8B-Instruct, while APL becomes unstable on Qwen models. CoNestQ avoids such failures through joint quantization parameter calibration and branch error compensation, maintaining reliable performance across models and bit-widths.

**Zero-Shot Accuracy.** Table 1 evaluates downstream accuracy under multi-precision quantization. Among nested methods, CoNestQ achieves the best accuracy in 19 of 24 configurations (79.2%) and outperforms MatGPTQ in 20 configurations (83.3%), consistent with the PPL results. From 8 to 3 bits, CoNestQ incurs a smaller average accuracy drop than MatGPTQ (6.38 vs. 7.15), indicating more stable performance across bit-widths. In contrast, GPTQ suffers larger degradation (10.43) due to severe low-bit collapse, e.g., a 29.2-point drop on Llama at 3 bits, while APL exhibits lower overall accuracy despite smaller changes. These results highlight CoNestQ's low-bit

Table 1: Perplexity (PPL, ↓) and zero-shot accuracy (Acc, ↑) at 3–8 bit-width. Bold denotes the best result among nested quantization methods, excluding GPTQ.

| Method | 3 bit | | 4 bit | | 5 bit | | 6 bit | | 7 bit | | 8 bit | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PPL | Acc | PPL | Acc | PPL | Acc | PPL | Acc | PPL | Acc | PPL | Acc |
| *Meta-Llama-3-8B-Instruct* | | | | | | | | | | | | |
| GPTQ | 2.4e3 | 41.4 | 4.3e2 | 56.4 | 95.7 | 62.5 | 11.2 | 70.3 | 11.0 | 70.2 | 11.0 | 70.6 |
| APL | **14.4** | **65.4** | **11.7** | **69.2** | **11.2** | 69.6 | **11.0** | **69.9** | **11.0** | 70.0 | **11.0** | 70.1 |
| MatGPTQ | 49.7 | 62.3 | 54.0 | 68.6 | 46.6 | 69.5 | 35.9 | 69.7 | 37.0 | 70.0 | 36.0 | 69.9 |
| **CoNestQ** | 29.3 | 62.0 | 22.1 | 68.6 | 20.1 | **69.8** | 19.8 | 69.8 | 19.1 | **70.2** | 19.3 | **70.4** |
| *Qwen2.5-14B-Instruct* | | | | | | | | | | | | |
| GPTQ | 11.0 | 72.7 | 9.5 | 75.3 | 8.8 | 75.8 | 8.7 | 76.1 | 8.7 | 76.0 | 8.7 | 76.1 |
| APL | 1.4e8 | 35.2 | 3.6e7 | 35.0 | 2.7e7 | 34.9 | 2.8e7 | 34.9 | 2.9e7 | 35.0 | 2.9e7 | 34.9 |
| MatGPTQ | 11.1 | 71.4 | **9.3** | 75.0 | **8.9** | 76.2 | **8.8** | 76.4 | **8.8** | 76.0 | **8.8** | 76.5 |
| **CoNestQ** | **10.7** | **71.8** | **9.3** | **75.5** | 9.0 | **76.5** | **8.8** | **76.5** | **8.8** | **76.6** | **8.8** | **76.6** |
| *Qwen3-8B-Base* | | | | | | | | | | | | |
| GPTQ | 11.8 | 66.5 | 10.2 | 70.9 | 9.8 | 71.3 | 9.8 | 71.8 | 9.8 | 71.5 | 9.8 | 71.6 |
| APL | 1.6e3 | 33.6 | 2.0e3 | 33.4 | 3.3e3 | 34.2 | 3.6e3 | 34.1 | 3.6e3 | 33.9 | 3.7e3 | 34.2 |
| MatGPTQ | 14.5 | 65.1 | 11.1 | **71.2** | 10.7 | **71.1** | **10.2** | 71.4 | 10.4 | 71.2 | **10.1** | 71.6 |
| **CoNestQ** | **13.6** | **66.1** | **11.5** | 70.6 | **10.4** | 70.9 | **10.2** | **71.9** | **10.2** | **71.6** | **10.1** | **72.0** |
| *Qwen3-14B-Base* | | | | | | | | | | | | |
| GPTQ | 10.9 | 70.1 | 9.3 | 73.1 | 9.0 | 74.0 | 9.0 | 74.2 | 8.9 | 74.1 | 8.9 | 74.1 |
| APL | 2.0e6 | 35.1 | 1.2e6 | 34.1 | 8.7e5 | 33.6 | 8.2e5 | 33.7 | 8.9e5 | 33.8 | 8.0e5 | 33.8 |
| MatGPTQ | 33.1 | 63.1 | **9.6** | 73.0 | 9.9 | 71.7 | 9.8 | 72.3 | 10.0 | 72.2 | 9.8 | 72.5 |
| **CoNestQ** | **22.9** | **66.3** | **9.6** | **73.3** | **9.8** | **72.6** | **9.6** | **72.5** | **9.6** | **72.7** | **9.6** | **72.7** |

Table 2: Bit-width and model-level validity at $\tau_P = 100$ and $\tau_A = 0.50$. Each six-digit string gives the per-bit validity indicators from 3 to 8 bits (left to right): 1 means both quality thresholds are met. "All" is 1 only when all six bits pass; VR averages "All" over the four models.

| Model | GPTQ | | APL | | MatGPTQ | | CoNestQ | |
|---|---|---|---|---|---|---|---|---|
| | 3–8 bit | All | 3–8 bit | All | 3–8 bit | All | 3–8 bit | All |
| Llama-3-8B-Instruct | 001111 | 0 | 111111 | 1 | 111111 | 1 | 111111 | 1 |
| Qwen2.5-14B-Instruct | 111111 | 1 | 000000 | 0 | 111111 | 1 | 111111 | 1 |
| Qwen3-8B-Base | 111111 | 1 | 000000 | 0 | 111111 | 1 | 111111 | 1 |
| Qwen3-14B-Base | 111111 | 1 | 000000 | 0 | 111111 | 1 | 111111 | 1 |
| VR (%, ↑) | 75 | | 25 | | **100** | | **100** | |

robustness, enabled by balanced parent-code selection and branch error compensation. Moreover, CoNestQ outperforms MatGPTQ in 16 of 18 Qwen configurations (88.9%) and consistently surpasses independently quantized GPTQ on Qwen2.5 at 4–8 bits. Overall, CoNestQ achieves full cross-bit availability with improved accuracy preservation and low-bit robustness.

**Validity.** The above results characterize performance at individual bit-widths. However, heterogeneous deployment requires a single quantization method to provide reliable quality across all target precisions. Strong performance at one bit-width cannot compensate for failures at others. We therefore define the *Validity Rate* (VR) as the fraction of evaluated models that satisfy the quality requirements at every target bit-width.

Let $P_{a,m,b}$ and $A_{a,m,b}$ denote the mean PPL and accuracy reported in Table 1 for method $a$, model $m$, and bit-width $b$. Given the evaluated model set $\mathcal{M}$ and target bit-width set $\mathcal{B} = \{3, 4, \ldots, 8\}$, VR is defined as:

$$v_{a,m}(\tau_P, \tau_A) = \prod_{b \in \mathcal{B}} \mathbf{1}[P_{a,m,b} \le \tau_P \wedge A_{a,m,b} \ge \tau_A],$$

$$\mathrm{VR}_a(\tau_P, \tau_A) = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} v_{a,m}(\tau_P, \tau_A).$$

Table 3: Component ablation on Meta-Llama-3-8B-Instruct in terms of PPL.

| Method | Para | Parent | Branch | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit | Mean |
|---|---|---|---|---|---|---|---|---|---|---|
| MatGPTQ | × | × | × | 49.67 | 54.03 | 46.55 | 35.90 | 37.00 | 35.95 | 43.18 |
| CoNestQ w/o Para | × | ✓ | ✓ | 43.35 | 53.07 | 39.86 | 34.06 | 33.92 | 33.35 | 39.60 |
| CoNestQ w/o Parent | ✓ | × | ✓ | 30.78 | 22.14 | 20.39 | 21.11 | 20.30 | 20.31 | 22.51 |
| CoNestQ w/o Branch | ✓ | ✓ | × | 30.92 | 21.81 | 20.89 | 21.13 | 20.30 | 20.22 | 22.54 |
| **CoNestQ** | ✓ | ✓ | ✓ | **29.28** | **22.11** | **20.14** | **19.75** | **19.08** | **19.25** | **21.60** |

Table 4: Storage cost (GB) for multi-precision deployment. GPTQ reports the six independent 3–8-bit quantized models, with † denoting the 8-bit model size. APL, MatGPTQ, and CoNestQ each store one quantized model for 3–8-bit. The last column reports GPTQ/CoNestQ storage ratio.

| Model | Reference | GPTQ | APL | MatGPTQ | **CoNestQ** | Storage Ratio |
|---|---|---|---|---|---|---|
| Llama-3-8B-Instruct | 32.13 | 42.41 (†9.27) | 10.49 | 7.20 | 7.20 | 5.89× |
| Qwen2.5-14B-Instruct | 29.55 | 74.94 (†16.65) | 18.53 | 13.63 | 13.63 | 5.50× |
| Qwen3-8B-Base | 16.39 | 44.53 (†9.61) | 10.86 | 7.16 | 7.16 | 6.22× |
| Qwen3-14B-Base | 29.55 | 74.92 (†16.65) | 18.45 | 13.63 | 13.63 | 5.50× |

Each method–model pair contributes one only if it meets both thresholds at all target bit-widths; thus, VR measures cross-bit reliability rather than performance at individual precisions. Following empirical criteria used in recent quantization studies (Ashkboos et al. (2024); Shafipour et al. (2025)), we set $\tau_P = 100$ and $\tau_A = 50\%$ as practical screening thresholds for detecting severe quality degradation. These thresholds do not indicate near-full-precision performance. Instead, failing them suggests substantial quality loss at certain precisions and limits the applicability of the quantized model for heterogeneous deployment.

Table 2 shows that CoNestQ and MatGPTQ achieve 100% VR across all four models. GPTQ obtains 75% VR, with failures caused by Llama-3-8B-Instruct exceeding the PPL threshold at 3–4 bits and violating the accuracy threshold at 3 bits. APL achieves only 25% VR, as all Qwen models fail both criteria across evaluated bit-widths. Combined with the accuracy and PPL results, these findings demonstrate that CoNestQ provides complete cross-bit coverage while maintaining competitive quality. Appendix C.4 further analyzes the sensitivity of VR to quality thresholds, showing that CoNestQ maintains stable cross-bit validity across different criteria.

### 4.2 Ablation Study

We ablate Joint Quantization-Parameter Calibration (Para), Balanced Parent-Code Selection (Parent), and Branch-Wise Error Compensation (Branch) on Meta-Llama-3-8B-Instruct by disabling each component individually in Table 3. CoNestQ reduces PPL from 43.18 for MatGPTQ to 21.60, demonstrating the benefit of explicitly coordinating cross-bit coupling. Removing Para increases PPL to 39.60, highlighting its critical role in establishing a well-calibrated space for shared-code optimization. Removing Parent or Branch increases PPL to 22.51 and 22.54, respectively. Without Parent, performance degrades and disrupts the expected cross-bit monotonicity, demonstrating its role in balancing errors across bit-widths. Without Branch, degradation is more pronounced at the precision extremes, as averaging heterogeneous branch errors suppresses bit-width-specific deviations and weakens targeted compensation. Overall, the three components complement each other to effectively optimize the coupled constraints of nested quantization across bit-widths.

### 4.3 Efficiency Analysis

Beyond cross-bit quality, we evaluate multi-precision efficiency in storage, offline quantization cost, and inference.

**Storage Efficiency.** As shown in Table 4, CoNestQ requires only 6.261–13.626 GB to support all 3–8-bit models, yielding a 5.5–6.2× reduction compared with six independently quantized GPTQ models. Compared with APL, CoNestQ reduces storage by 26%–34%, since APL requires additional bit-width-specific lookup information, whereas CoNestQ stores only the highest-bit parent weights and a shared set of quantization parameters. CoNestQ matches MatGPTQ in storage cost,

Figure 2: Offline quantization times (seconds) on Qwen3-8B-Base. The stacked segments of GPTQ show per-bit-width quantization times.

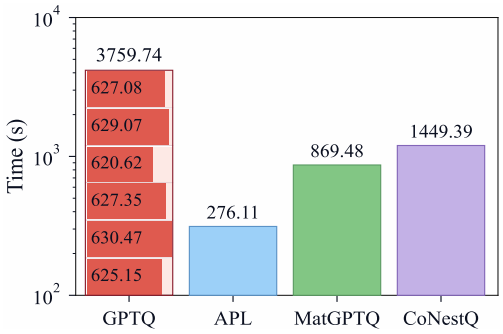



Table 5: Peak GPU memory (GB) and throughput (tokens/s) on Llama-3-8B-Instruct.

| Bit | Method | Peak Memory ↓ | Throughput ↑ |
|---|---|---|---|
| 3bit | GPTQ | 5.23 | 49.32 |
| | APL | 9.97 | 42.40 |
| | MatGPTQ | 4.60 | 47.46 |
| | **CoNestQ** | 4.60 | 47.52 |
| 4bit | GPTQ | 5.67 | 50.21 |
| | APL | 9.97 | 41.87 |
| | MatGPTQ | 5.39 | 47.30 |
| | **CoNestQ** | 5.39 | 47.33 |
| 8bit | GPTQ | 9.00 | 48.97 |
| | APL | 9.97 | 38.21 |
| | MatGPTQ | 8.64 | 46.50 |
| | **CoNestQ** | 8.64 | 47.14 |

reflecting the same compact nested linear representation. Notably, the complete CoNestQ model is even 18%–26% smaller than a single 8-bit GPTQ model, while supporting all six target bit-widths.

**Offline Quantization Cost.** Figure 2 reports the offline quantization cost on Qwen3-8B-Base. Constructing the full 3–8-bit GPTQ family requires six independent runs and 3759.74 seconds, whereas CoNestQ jointly builds all six precisions with 61% lower cost. This comparison therefore reflects multi-precision construction rather than single-precision quantisation. Although MatGPTQ and APL incur lower costs, CoNestQ provides stronger quantization quality than MatGPTQ and more consistent applicability across model families than APL. As quantization is performed offline, the additional construction cost does not affect inference-time efficiency. Overall, CoNestQ offers a favorable trade-off between multi-precision construction cost, quantization quality.

**Inference Efficiency.** We evaluate peak GPU memory and inference throughput on Llama-3-8B-Instruct at 3-, 4-, and 8-bit precision under a unified Transformers-based framework on an NVIDIA H100 GPU, as reported in Table 5. Since CoNestQ and MatGPTQ share similar quantized parameter layouts and storage formats, we adopt the same inference implementation for both, avoiding additional customized inference overhead. Compared with independently quantized GPTQ, CoNestQ reduces throughput by only about 4% on average while lowering peak memory by 7%. Relative to APL, it improves throughput by about 16% and reduces peak memory by 38%. CoNestQ also achieves nearly identical memory usage and comparable throughput to MatGPTQ. Overall, CoNestQ enables flexible multi-precision inference with negligible runtime overhead while maintaining high throughput and low memory consumption.

## 5 Discussion and Limitations

CoNestQ opens several directions for further improvement. First, extending the current local surrogate optimization toward more global cross-bit coordination may further improve quantization quality across all precisions. Second, accelerating joint multi-precision calibration can reduce offline quantization cost. Third, dedicated operator- and kernel-level optimization could further enhance inference throughput.

## 6 Conclusion

In this work, we study nested multi-precision quantization for LLMs, where multiple bit-widths share a unified representation. Existing methods often overlook cross-bit coupling among quantization parameters, parent-code assignments, and error propagation, leading to inconsistent quality across precisions. CoNestQ addresses this by decomposing the coupled optimization into three coordinated stages: joint parameter calibration for shared grids, balanced parent-code selection across precisions, and branch-wise error compensation for bit-specific reconstruction. Extensive experiments show that CoNestQ achieves the highest validity rate among the four methods, together with competitive perplexity and accuracy, strong low-bit robustness, and efficient inference, enabling adaptive LLM deployment under diverse resource and service demands.

## AI USE STATEMENT

LLM-based AI tools were used to polish the manuscript by correcting grammar and spelling and improving sentence fluency. They were also used to assist with code debugging during research execution and to generate scripts for producing figures in the paper. All AI-assisted outputs were reviewed and verified by the authors.

## REFERENCES

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10081–10100, 2025.

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

Xin Ding, Xiaoyu Liu, Zhijun Tu, Yun Zhang, Wei Li, Jie Hu, Hanting Chen, Yehui Tang, Zhiwei Xiong, Baoqun Yin, et al. Cbq: Cross-block quantization for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 7056–7075, 2025.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Youhe Jiang, Ran Yan, Xiaozhe Yao, Yang Zhou, Beidi Chen, and Binhang Yuan. Hexgen: Generative inference of large language model over heterogeneous environment. In *Proceedings of the International Conference on Machine Learning*, volume 235, pp. 21946–21961, 2024.

Maximilian Kleinegger, Elvir Crnčević, and Dan Alistarh. Matgptq: Accurate and efficient post-training matryoshka quantization. *arXiv preprint arXiv:2602.03537*, 2026.

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. *Proceedings of machine learning and systems*, 6:87–100, 2024.

Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. Qllm: Accurate and efficient low-bitwidth quantization for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 15417–15439, 2024a.

Zechun Liu, Kwang-Ting Cheng, Dong Huang, Eric P Xing, and Zhiqiang Shen. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4942–4952, 2022.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. In *Findings of the association for computational linguistics: ACL 2024*, pp. 467–484, 2024b.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Pranav Nair, Puranjay Datta, Jeff Dean, Prateek Jain, and Aditya Kusupati. Matryoshka quantization. *arXiv preprint arXiv:2502.06786*, 2025.

Gunho Park, Jeongin Bae, Beomseok Kwon, Byeongwook Kim, Se Jung Kwon, and Dongsoo Lee. AnyBCQ: Hardware efficient flexible binary-coded quantization for multi-precision LLMs. *arXiv preprint arXiv:2510.10467*, 2025. URL https://arxiv.org/abs/2510.10467.

Yeonhong Park, Jake Hyun, SangLyul Cho, Bonggeun Sim, and Jae W. Lee. Any-precision llm: Low-cost deployment of multiple, different-sized llms. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

A Yang Qwen, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengpeng Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, pp. 2412–15115, 2025.

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

Rasoul Shafipour, David Harrison, Maxwell Horton, Jeffrey Marker, Houman Bedayat, Sachin Mehta, Mohammad Rastegari, Mahyar Najibi, and Saman Naderiparizi. Seedlm: Compressing llm weights into seeds of pseudo-random generators. In *International Conference on Learning Representations*, volume 2025, pp. 12755–12771, 2025.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Gao Peng, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45472–45496, 2024.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 31094–31116. PMLR, 2023. URL https://proceedings.mlr.press/v202/sheng23a.html.

Hanling Wang, Qing Li, Dan Zhao, Yuhong Song, Xingchi Chen, Teng Gao, Peiyuan Zong, Zhuyun Qi, Yue Yu, and Yong Jiang. Intelligence delivery network: Toward an internet architecture for the ai age. *arXiv preprint arXiv:2605.13235*, 2026.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the International Conference on Machine Learning*, pp. 38087–38099, 2023.

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

Haichao Yu, Haoxiang Li, Humphrey Shi, Thomas S Huang, and Gang Hua. Any-precision deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10763–10771, 2021.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12: 1556–1577, 2024.

# A CORE ALGORITHM

Algorithm 1 summarizes joint parameter calibration, balanced parent-code selection, and branch-wise error compensation with synchronization at compensation-window boundaries. The resulting parent codes and shared quantization parameters support all target bit-widths.

**Algorithm 1** CoNestQ Quantization

**Require:** Weight matrix $W$, calibration input $X$, target bit-widths $\mathcal{B}$, window size $G$
**Ensure:** Parent-code matrix $Q$, quantization parameters $(s, z)$

$(s, z) \leftarrow \arg\min_{p \in \mathcal{P}} \sum_{b \in \mathcal{B}} \left\| W - \widehat{W}_p^{(b)} \right\|_F^2$
$H^{-1} \leftarrow \left(2XX^\top\right)^{-1}$
$\widetilde{W}^{(b)} \leftarrow W, \quad \forall b \in \mathcal{B}$
**for** $i = 0,\ G,\ 2G, \ldots$ **do**
  $r \leftarrow \min(i + G - 1, d_i)$
  **for** $j = i, \ldots, r$ **do**
    $Q_{:,j} \leftarrow \arg\min_{q \in \mathcal{C}_M} J(q)$
    $e_j^{(b)} \leftarrow \widetilde{w}_j^{(b)} - s(\kappa_b \rho_b(Q_{:,j}) - z)$
    $\widetilde{W}^{(b)}_{:,\, j+1:r} \leftarrow \widetilde{W}^{(b)}_{:,\, j+1:r} - \frac{e_j^{(b)}}{[H^{-1}]_{jj}} H^{-1}_{j,\, j+1:r}$
  **end for**
  $\widetilde{W}^{(b)}_{:,\, r+1:} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \widetilde{W}^{(b)}_{:,\, r+1:}$
**end for**
**return** $Q, (s, z)$

# B IMPLEMENTATION AND EXPERIMENTAL SETUP

## B.1 HARDWARE AND SOFTWARE

All experiments use a single NVIDIA H100 GPU with 80GB of device memory. The software environment comprises CUDA 12.4, PyTorch 2.6.0 and Transformers 4.56.1.

## B.2 QUANTIZATION AND CALIBRATION SETTINGS

We perform weight-only quantization over $\mathcal{B} = \{3, 4, 5, 6, 7, 8\}$. All linear quantization methods use a group size of 128, and CoNestQ sets the compensation window to $G = 128$ accordingly, aligning the compensation granularity with the quantization groups to maintain a unified partition scale. Baseline methods use their default configurations unless otherwise specified. Calibration is performed using 600 samples from C4, each consisting of 128 tokens. CoNestQ searches clipping factors over $\mathcal{P} = \{0.80, 0.81, \ldots, 1.00\}$ and sets $\lambda = 0.1$ and $\mu = 0.05$ for balanced parent-code selection, weighting the reconstruction error and error-ordering regularizer, respectively. These settings are fixed across all evaluated models.

## B.3 EXPERIMENTAL SETTINGS

PPL is evaluated using the evaluation procedure provided by AutoGPTQ on WikiText-2, C4, and PTB. Zero-shot accuracy is evaluated using lm-evaluation-harness on PIQA, ARC-Challenge, and HellaSwag. The PPL and accuracy values in Table 1 are means over the three corresponding datasets; accuracy is reported as a percentage. Throughput and peak GPU memory are evaluated under a unified Transformers-based inference framework using the same inference implementation for CoNestQ and MatGPTQ. Timing is based on CUDA-synchronized wall-clock time; peak memory is measured independently for each formal run using CUDA memory statistics. Unless otherwise stated, all subsequent experiments use the above environment and default settings.

# C Supplementary Experimental Results

## C.1 Complete Perplexity Results

Tables 6–9 report dataset-level PPL for every model and target bit-width, with the FP16 reference shown for each dataset. They provide the individual measurements underlying the averages in Table 1. Bold marks the lowest PPL among nested methods at each bit-width; GPTQ is an independent-precision reference. Large values are retained in scientific notation.

The dataset-level results localize the pronounced low-bit degradation of GPTQ on Llama to C4. On this dataset, CoNestQ reduces 3-bit PPL from 107.61 for MatGPTQ, accounting for much of the improvement in the corresponding three-dataset average.

Table 6: Perplexity (↓) on Meta-Llama-3-8B-Instruct across 3–8 bits.

| Method | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit |
|---|---|---|---|---|---|---|
| *WikiText-2* (FP16: 8.28) | | | | | | |
| GPTQ | 27.04 | 10.25 | 8.92 | 8.31 | 8.30 | 8.30 |
| APL | **10.98** | **8.83** | **8.46** | **8.32** | **8.30** | **8.29** |
| MatGPTQ | 14.92 | 9.81 | 9.16 | 8.87 | 8.88 | 8.81 |
| CoNestQ | 14.37 | 9.55 | 9.02 | 8.79 | 8.76 | 8.71 |
| *C4* (FP16: 13.10) | | | | | | |
| GPTQ | 7016.31 | 1255.50 | 266.64 | 14.00 | 13.11 | 13.12 |
| APL | **17.12** | **13.97** | **13.36** | **13.15** | **13.11** | **13.10** |
| MatGPTQ | 107.61 | 139.48 | 118.59 | 87.10 | 90.39 | 87.40 |
| CoNestQ | 50.45 | 43.85 | 39.29 | 38.45 | 36.58 | 37.14 |
| *PTB* (FP16: 11.45) | | | | | | |
| GPTQ | 47.75 | 13.45 | 11.58 | 11.41 | 11.46 | 11.47 |
| APL | **14.99** | **12.22** | **11.65** | **11.47** | **11.45** | **11.45** |
| MatGPTQ | 26.47 | 12.80 | 11.90 | 11.74 | 11.74 | 11.65 |
| CoNestQ | 23.03 | 12.93 | 12.11 | 12.00 | 11.90 | 11.89 |

Table 7: Perplexity (↓) on Qwen2.5-14B-Instruct across 3–8 bits.

| Method | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit |
|---|---|---|---|---|---|---|
| *WikiText-2* (FP16: 5.69) | | | | | | |
| GPTQ | 8.33 | 6.58 | 5.84 | 5.72 | 5.70 | 5.70 |
| APL | $1.34 \times 10^8$ | $3.88 \times 10^7$ | $2.89 \times 10^7$ | $3.04 \times 10^7$ | $3.07 \times 10^7$ | $3.07 \times 10^7$ |
| MatGPTQ | 7.91 | **6.32** | **5.98** | **5.81** | **5.82** | **5.79** |
| CoNestQ | **7.67** | 6.41 | 6.04 | 5.92 | 5.90 | 5.88 |
| *C4* (FP16: 10.93) | | | | | | |
| GPTQ | 13.31 | 11.51 | 11.06 | 10.95 | 10.93 | 10.93 |
| APL | $1.12 \times 10^8$ | $2.85 \times 10^7$ | $2.17 \times 10^7$ | $2.21 \times 10^7$ | $2.26 \times 10^7$ | $2.28 \times 10^7$ |
| MatGPTQ | 13.57 | 11.60 | 11.31 | 11.20 | 11.17 | 11.16 |
| CoNestQ | **13.02** | **11.52** | **11.24** | **11.12** | **11.11** | **11.10** |
| *PTB* (FP16: 9.39) | | | | | | |
| GPTQ | 11.36 | 10.28 | 9.45 | 9.40 | 9.40 | 9.39 |
| APL | $1.87 \times 10^8$ | $4.15 \times 10^7$ | $3.07 \times 10^7$ | $3.16 \times 10^7$ | $3.30 \times 10^7$ | $3.29 \times 10^7$ |
| MatGPTQ | 11.87 | 9.94 | **9.53** | **9.42** | **9.44** | **9.40** |
| CoNestQ | **11.52** | **9.85** | 9.64 | 9.47 | 9.45 | 9.43 |

## C.2 Complete Zero-Shot Accuracy Results

Tables 10–13 report accuracy separately for PIQA, ARC-Challenge, and HellaSwag. Bold marks the highest accuracy among nested methods at each bit-width, determined before rounding. On Qwen3-14B-Base at 3 bits, CoNestQ improves all three task accuracies over MatGPTQ, providing task-level support for the aggregate low-bit improvement reported in the main text.

Table 8: Perplexity (↓) on Qwen3-8B-Base across 3–8 bits.

| Method | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit |
|---|---|---|---|---|---|---|
| *WikiText-2* (FP16: 6.99) | | | | | | |
| GPTQ | 9.05 | 7.37 | 7.09 | 7.01 | 7.00 | 7.00 |
| APL | 790.33 | 862.59 | 1143.22 | 1255.87 | 1262.57 | 1276.68 |
| MatGPTQ | 10.76 | **7.90** | 7.52 | **7.17** | 7.32 | **7.15** |
| CoNestQ | **10.06** | 8.12 | **7.44** | 7.22 | **7.25** | 7.16 |
| *C4* (FP16: 12.15) | | | | | | |
| GPTQ | 14.26 | 12.54 | 12.23 | 12.17 | 12.16 | 12.15 |
| APL | 3308.15 | 4624.49 | 7927.02 | 8601.14 | 8784.82 | 8846.15 |
| MatGPTQ | 18.01 | **13.76** | 13.41 | 12.78 | 13.08 | 12.71 |
| CoNestQ | **16.62** | 13.96 | **12.99** | **12.69** | **12.75** | **12.56** |
| *PTB* (FP16: 10.12) | | | | | | |
| GPTQ | 12.21 | 10.55 | 10.20 | 10.15 | 10.13 | 10.12 |
| APL | 545.13 | 625.07 | 854.53 | 860.98 | 873.93 | 875.85 |
| MatGPTQ | 14.57 | **11.70** | 11.06 | **10.55** | 10.71 | 10.52 |
| CoNestQ | **14.16** | 12.35 | **10.85** | 10.56 | **10.58** | **10.48** |

Table 9: Perplexity (↓) on Qwen3-14B-Base across 3–8 bits.

| Method | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit |
|---|---|---|---|---|---|---|
| *WikiText-2* (FP16: 6.38) | | | | | | |
| GPTQ | 8.17 | 6.75 | 6.45 | 6.39 | 6.38 | 6.38 |
| APL | $2.71 \times 10^6$ | $1.51 \times 10^6$ | $1.14 \times 10^6$ | $1.10 \times 10^6$ | $1.22 \times 10^6$ | $1.07 \times 10^6$ |
| MatGPTQ | 30.21 | 6.86 | 7.04 | 6.98 | 7.14 | 6.96 |
| CoNestQ | **20.73** | **6.85** | **7.01** | **6.87** | **6.88** | **6.81** |
| *C4* (FP16: 11.21) | | | | | | |
| GPTQ | 13.25 | 11.57 | 11.28 | 11.22 | 11.21 | 11.21 |
| APL | $1.99 \times 10^6$ | $1.34 \times 10^6$ | $1.00 \times 10^6$ | $9.52 \times 10^5$ | $1.03 \times 10^6$ | $9.17 \times 10^5$ |
| MatGPTQ | 39.74 | 12.05 | 12.63 | 12.48 | 12.71 | 12.46 |
| CoNestQ | **28.65** | **12.03** | **12.46** | **12.18** | **12.18** | **12.10** |
| *PTB* (FP16: 9.22) | | | | | | |
| GPTQ | 11.27 | 9.58 | 9.29 | 9.23 | 9.22 | 9.22 |
| APL | $1.41 \times 10^6$ | $7.39 \times 10^5$ | $4.62 \times 10^5$ | $4.17 \times 10^5$ | $4.35 \times 10^5$ | $4.14 \times 10^5$ |
| MatGPTQ | 29.27 | 9.86 | **9.99** | 9.91 | 10.03 | 9.87 |
| CoNestQ | **19.27** | **9.85** | 10.07 | **9.82** | **9.80** | **9.74** |

Table 10: Zero-shot accuracy (%, ↑) on Meta-Llama-3-8B-Instruct across 3–8 bits.

| Method | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit |
|---|---|---|---|---|---|---|
| *PIQA* (FP16: 78.62%) | | | | | | |
| GPTQ | 60.66 | 75.68 | 76.66 | 78.51 | 78.45 | 78.84 |
| APL | **77.64** | **78.84** | 78.02 | 78.84 | 79.00 | 78.94 |
| MatGPTQ | 74.27 | 78.24 | **78.73** | 79.11 | 79.33 | 78.84 |
| CoNestQ | 73.83 | 78.40 | 78.24 | **79.27** | **79.49** | **79.60** |
| *ARC-Challenge* (FP16: 56.74%) | | | | | | |
| GPTQ | 27.73 | 48.04 | 54.27 | 57.08 | 56.31 | 57.00 |
| APL | **46.76** | **53.84** | 55.12 | **55.20** | 55.46 | 55.63 |
| MatGPTQ | 41.98 | 52.90 | 54.61 | 54.69 | 55.03 | 55.29 |
| CoNestQ | 42.15 | 53.50 | **56.23** | 54.95 | **55.80** | **56.14** |
| *HellaSwag* (FP16: 75.81%) | | | | | | |
| GPTQ | 35.74 | 45.53 | 56.70 | 75.39 | 75.80 | 75.86 |
| APL | **71.76** | **74.94** | **75.58** | **75.59** | **75.56** | **75.69** |
| MatGPTQ | 70.76 | 74.67 | 75.09 | 75.25 | 75.50 | 75.52 |
| CoNestQ | 70.00 | 73.86 | 75.03 | 75.15 | 75.35 | 75.33 |

Table 11: Zero-shot accuracy (%, ↑) on Qwen2.5-14B-Instruct across 3–8 bits.

| Method | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit |
|---|---|---|---|---|---|---|
| *PIQA* (FP16: 81.83%) | | | | | | |
| GPTQ | 79.98 | 82.54 | 81.34 | 81.66 | 81.61 | 81.72 |
| APL | 51.52 | 51.09 | 51.47 | 51.47 | 51.58 | 51.58 |
| MatGPTQ | **79.33** | 81.01 | 82.21 | 81.61 | 81.77 | 81.94 |
| CoNestQ | **79.33** | **81.99** | **82.26** | **82.21** | **82.26** | **82.21** |
| *ARC-Challenge* (FP16: 62.03%) | | | | | | |
| GPTQ | 57.76 | 59.64 | 61.95 | 62.20 | 62.12 | 62.20 |
| APL | 27.99 | 27.99 | 27.22 | 27.30 | 27.47 | 27.13 |
| MatGPTQ | 55.29 | 60.49 | 62.80 | **63.40** | 62.12 | **63.48** |
| CoNestQ | **55.80** | **60.75** | **63.40** | 63.23 | **63.65** | **63.48** |
| *HellaSwag* (FP16: 84.33%) | | | | | | |
| GPTQ | 80.27 | 83.71 | 84.04 | 84.37 | 84.40 | 84.34 |
| APL | 26.01 | 25.89 | 25.92 | 25.90 | 25.92 | 25.83 |
| MatGPTQ | 79.57 | 83.58 | 83.69 | **84.22** | **84.07** | **84.12** |
| CoNestQ | **80.37** | **83.71** | **83.75** | 83.92 | 83.96 | 84.00 |

Table 12: Zero-shot accuracy (%, ↑) on Qwen3-8B-Base across 3–8 bits.

| Method | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit |
|---|---|---|---|---|---|---|
| *PIQA* (FP16: 79.33%) | | | | | | |
| GPTQ | 76.44 | 79.22 | 79.33 | 78.94 | 79.43 | 79.38 |
| APL | 50.65 | 50.92 | 51.14 | 51.41 | 51.20 | 51.80 |
| MatGPTQ | 75.84 | **78.56** | **80.14** | 79.71 | 79.54 | 79.92 |
| CoNestQ | **77.64** | 78.24 | 79.76 | **80.14** | **80.14** | **80.09** |
| *ARC-Challenge* (FP16: 56.91%) | | | | | | |
| GPTQ | 48.89 | 55.55 | 56.31 | 57.76 | 56.66 | 56.83 |
| APL | 22.95 | 22.01 | 23.89 | 23.55 | 23.12 | 23.63 |
| MatGPTQ | 49.06 | **57.94** | **56.66** | 57.00 | 56.57 | 57.42 |
| CoNestQ | **49.57** | 57.34 | 56.06 | **58.02** | **57.17** | **58.45** |
| *HellaSwag* (FP16: 78.62%) | | | | | | |
| GPTQ | 74.29 | 77.93 | 78.34 | 78.65 | 78.51 | 78.68 |
| APL | 27.10 | 27.13 | 27.63 | 27.25 | 27.27 | 27.02 |
| MatGPTQ | 70.27 | **76.99** | 76.62 | **77.58** | 77.45 | 77.44 |
| CoNestQ | **71.07** | 76.32 | **76.93** | 77.53 | **77.54** | **77.50** |

Table 13: Zero-shot accuracy (%, ↑) on Qwen3-14B-Base across 3–8 bits.

| Method | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit |
|---|---|---|---|---|---|---|
| *PIQA* (FP16: 82.05%) | | | | | | |
| GPTQ | 79.16 | 80.79 | 82.59 | 81.88 | 82.21 | 81.94 |
| APL | 52.18 | 50.44 | 49.35 | 50.00 | 49.78 | 50.33 |
| MatGPTQ | 73.67 | **80.74** | 81.28 | **81.34** | **81.28** | 81.34 |
| CoNestQ | **76.66** | **80.74** | **81.39** | 81.28 | 81.18 | **81.72** |
| *ARC-Challenge* (FP16: 59.22%) | | | | | | |
| GPTQ | 53.07 | 57.85 | 58.02 | 59.47 | 58.79 | 59.04 |
| APL | 27.30 | 26.11 | 25.51 | 25.43 | 25.85 | 25.51 |
| MatGPTQ | 49.23 | 57.94 | 54.69 | 55.89 | 55.97 | **56.31** |
| CoNestQ | **52.99** | **58.87** | **56.66** | **56.31** | **56.83** | **56.31** |
| *HellaSwag* (FP16: 81.41%) | | | | | | |
| GPTQ | 77.98 | 80.66 | 81.30 | 81.36 | 81.39 | 81.42 |
| APL | 25.85 | 25.81 | 25.93 | 25.63 | 25.62 | 25.59 |
| MatGPTQ | 66.45 | 80.20 | 79.25 | 79.64 | 79.37 | 79.72 |
| CoNestQ | **69.19** | **80.23** | **79.64** | **79.96** | **79.96** | **80.05** |

### C.3 Detailed Storage Costs

Table 14 lists the sizes of the six independently quantized GPTQ models and their sum. Table 15 reports one shared checkpoint per nested method, each supporting all six bit-widths. Sizes are reported in GB using *[Pending: byte-to-GB conversion]*; the accounting includes *[Pending: checkpoint files, unquantized parameters, and metadata included in the measurement]*. The storage ratio in Table 4 divides the GPTQ sum by the CoNestQ shared-checkpoint size, with all calculations performed before rounding.

Table 14: Per-bit GPTQ storage and total storage for the 3–8-bit family (GB).

| Model | 3 bit | 4 bit | 5 bit | 6 bit | 7 bit | 8 bit | Total |
|---|---|---|---|---|---|---|---|
| Llama-3-8B-Instruct | 4.871 | 5.750 | 6.629 | 7.509 | 8.388 | 9.267 | 42.414 |
| Qwen2.5-14B-Instruct | 8.329 | 9.994 | 11.658 | 13.323 | 14.987 | 16.651 | 74.943 |
| Qwen3-8B-Base | 5.234 | 6.109 | 6.984 | 7.859 | 8.734 | 9.609 | 44.529 |
| Qwen3-14B-Base | 8.325 | 9.989 | 11.654 | 13.318 | 14.983 | 16.647 | 74.917 |

Table 15: Shared-checkpoint storage for nested methods supporting 3–8 bits (GB).

| Model | APL | MatGPTQ | CoNestQ |
|---|---|---|---|
| Llama-3-8B-Instruct | 10.486 | 7.198 | 7.198 |
| Qwen2.5-14B-Instruct | 18.525 | 13.626 | 13.626 |
| Qwen3-8B-Base | 10.864 | 7.163 | 7.163 |
| Qwen3-14B-Base | 18.447 | 13.625 | 13.625 |

### C.4 Sensitivity to Quality Thresholds

Table 16 recomputes VR for all combinations of PPL thresholds $\{50, 100, 150\}$ and accuracy thresholds $\{45\%, 50\%, 55\%\}$ from the unrounded dataset-level results above, using the same arithmetic averaging and all-bit validity criterion as in the main text. No additional quantization or inference runs are required.

Table 16: VR (%) under alternative quality thresholds.

| $\tau_P$ | $\tau_A$ (%) | GPTQ | APL | MatGPTQ | CoNestQ |
|---|---|---|---|---|---|
| 50 | 45 | 75 | 25 | 75 | **100** |
| 50 | 50 | 75 | 25 | 75 | **100** |
| 50 | 55 | 75 | 25 | 75 | **100** |
| 100 | 45 | 75 | 25 | **100** | **100** |
| 100 | 50 | 75 | 25 | **100** | **100** |
| 100 | 55 | 75 | 25 | **100** | **100** |
| 150 | 45 | 75 | 25 | **100** | **100** |
| 150 | 50 | 75 | 25 | **100** | **100** |
| 150 | 55 | 75 | 25 | **100** | **100** |

CoNestQ retains 100% VR throughout the tested grid. At the strictest PPL threshold of 50, MatGPTQ's 4-bit Llama PPL of 54.03 reduces its VR to 75%, while GPTQ and APL remain at 75% and 25%, respectively. Varying the accuracy threshold within the tested range leaves model-level decisions unchanged because all configurations satisfying the PPL requirement also exceed 55% accuracy.