You Had One Job: Per-Task Quantization Using LLMs' Hidden Representations

Amit LeVi [1 2] **Raz Lapid** [3] **Rom Himelstein** [1] **Chaim Baskin** [4] **Ravid Shwartz-Ziv** [5] **Avi Mendelson** [1]

Abstract

Post-training quantization (PTQ) methods allocate precision without considering the target task. This can waste bits on layers that are less relevant to the task signal while over-compressing layers that are critical for downstream behavior. We propose Task-Aware Quantization (TAQ), a training-free, weight-only mixed-precision PTQ framework that uses a small set of unlabeled task calibration prompts to allocate higher precision to task-relevant transformer layers under a fixed bit budget. TAQ estimates layer importance from hidden representations and output sensitivity, and we instantiate it with three scoring rules: TAQ-IS, based on activation information and stability; TAQ-KL, based on output-distribution sensitivity under a quantization-noise proxy; and TAQ-O, a label-informed oracle diagnostic for analyzing layer sensitivity. Across several benchmarks, TAQ outperforms task-agnostic baselines such in most settings, with especially strong gains in the accuracy–memory ratio. We further validate that these gains translate to real deployment behavior through hardware throughput and latency measurements, and analyze calibration robustness and residual-stream error propagation. Overall, TAQ turns mixed-precision PTQ from a model-centric compression step into a task-conditioned precision-allocation problem. The implementation is available at .

1. Introduction

Large language models (LLMs) continue to scale in size and capability (Vaswani et al., 2017; Kaplan et al., 2020), but this scaling substantially increases inference and memory costs (Pope et al., 2023). At the same time, many deployed applications require only a narrow subset of model capabilities, such as code completion (Chen et al., 2021; Roziere et al., 2023), mathematical reasoning (Cobbe et al., 2021; Hendrycks et al., 2021), or domain-specific question answering. This motivates compression methods that can preserve the capabilities most relevant to the target use case while avoiding task-specific training.

Post-training quantization (PTQ) reduces memory footprint and inference cost by representing model weights or activations with fewer bits (Gholami et al., 2022). Existing LLM PTQ methods include accurate weight reconstruction methods (Frantar et al., 2022; Guan et al., 2024), activation-aware and outlier-mitigation techniques (Dettmers et al., 2022; Xiao et al., 2023), and mixed-precision schemes based on global or layer-sensitive criteria (Dong et al., 2019; Lin et al., 2024; Zhang et al., 2025; Zhao et al., 2025). However, these methods are largely task-agnostic: they optimize generic reconstruction, activation, curvature, or salience proxies rather than asking which layers are most important for a particular downstream task. This mismatch is important because transformer layers contribute heterogeneously to end-task behavior (Tenney et al., 2019; Meng et al., 2022; Geva et al., 2021). Work in representation analysis and mechanistic interpretability suggests that task-relevant information is not uniformly distributed across depth, and that hidden states can expose where task-conditioned computation is concentrated (Cunningham et al., 2023; Heimersheim & Nanda, 2024; Sharkey et al., 2025).

In this work, we introduce a novel task-conditioned precision-allocation approach for post-training mixed-precision quantization. We propose **Task-Aware Quantization** (**TAQ**), a training-free, weight-only PTQ framework that uses task-specific calibration prompts to estimate layer importance with respect to the residual-stream task signal and allocate precision under a fixed budget, an novel precision-allocation policies that can be combined with existing PTQ backends. We instantiate TAQ with three layer-scoring rules: **TAQ-IS** (**I**nformation and **S**tability), **TAQ-KL** (**KL**-divergence-based output sensitivity), and **TAQ-O** (**O**racle sensitivity), a label-informed diagnostic used for analysis rather than deployment. To the best of our knowledge, TAQ is the first training-free, weight-only mixed-precision PTQ framework to allocate per-layer precision from unlabeled target-task prompts, using residual-

[1]Technion – Israel Institute of Technology [2]Zenity [3]Deepkeep [4]Ben-Gurion University of the Negev [5]New York University. Correspondence to: Amit LeVi <amitlevi@campus.technion.ac.il>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

stream hidden-state statistics and output-sensitivity signals. Our contributions are: **(1) Task-aware PTQ.** We formulate mixed-precision PTQ as task-conditioned precision allocation and introduce TAQ. **(2) Layer-importance scoring.** We propose TAQ-IS, TAQ-KL, and diagnostic TAQ-O to identify task-sensitive layers for higher precision. **(3) Accuracy–memory–latency gains.** Across code, reasoning, and knowledge tasks, TAQ improves accuracy–memory trade-offs over GPTQ, AWQ, and SliM-LLM, with hardware validation. **(4) Robustness analysis.** We study calibration size, mixed-task calibration, structural baselines, and inter-layer effects, showing that TAQ rankings are largely stable in practical settings.

2. Background and Related Work

Related Work: PTQ reduces LLM memory and inference cost while preserving model behavior through weight reconstruction, outlier handling, and calibration (Frantar et al., 2022; Xiao et al., 2023; Lin et al., 2024; Shao et al., 2023; Dettmers et al., 2023). Mixed-precision PTQ further exploits non-uniform layer, channel, or group sensitivity, allocating higher precision using model-centric criteria such as curvature, reconstruction error, salience, or average zero-shot fidelity (Dong et al., 2019; Guan et al., 2024; Huang et al., 2024; Zhang et al., 2025; Zhao et al., 2025; Zheng et al., 2024). In contrast, TAQ makes bit allocation task-conditioned: given a small unlabeled calibration set from the target distribution, it identifies layers that preserve task-relevant residual-stream signal and assigns them higher precision under a fixed budget. This is motivated by representation-analysis and mechanistic-interpretability work showing that hidden states encode semantic, factual, task, and behavioral features, and that localized activation interventions can reveal where task-relevant computations concentrate (Turner et al., 2023; Zou et al., 2023; Meng et al., 2022; Shnaidman et al., 2025; Rahimi et al., 2026).

Background: We use this perspective to define task-conditioned layer statistics. Let $D_{\text{calib}} = \{q_j\}_{j=1}^{n}$ be a small calibration set from the target task. For an input prompt q_j with T_j valid tokens, let $H_\ell(q_j) \in \mathbb{R}^{T_j \times d}$ denote the hidden states produced by layer ℓ. Aggregating valid token representations from prompts in D_{calib} gives a layer-wise representation matrix $X_\ell \in \mathbb{R}^{m_\ell \times d}$, where $m_\ell = \sum_{j=1}^{n} T_j$ is the number of collected token vectors. To measure how dispersed these representations are, we form the token–token Gram matrix $G_\ell = X_\ell X_\ell^\top$. Let $\{\lambda_{\ell,i}\}_{i=1}^{m_\ell}$ denote its nonnegative eigenvalues and $\tilde{\lambda}_{\ell,i} = \lambda_{\ell,i} / \sum_k \lambda_{\ell,k}$. The spectral entropy of the layer representation is

$$H(G_\ell) = -\sum_{i=1}^{m_\ell} \tilde{\lambda}_{\ell,i} \log\left(\tilde{\lambda}_{\ell,i} + \epsilon\right). \tag{1}$$

Larger entropy indicates that task prompts induce a more distributed representation at layer ℓ, whereas lower entropy indicates concentration along fewer dominant directions. Together with simple activation statistics such as variance, this provides a label-free way to characterize how task inputs are represented across depth. Since task-useful information is often not uniformly distributed across layers (Tenney et al., 2019; Jawahar et al., 2019; Meng et al., 2022; Hendel et al., 2023; Skean et al., 2025), minimizing task-agnostic quantization distortion may fail to preserve the computations most relevant to a target task. TAQ uses these task-conditioned representation diagnostics to guide per-layer precision allocation.

3. Per-task quantization

We first motivate task-aware precision allocation by examining two empirical properties of LLM representations: (i) task prompts induce different layer-wise representation profiles, and (ii) quantization affects hidden representations non-uniformly across depth. These observations motivate the TAQ framework introduced in the following subsection.

Task-conditioned layer profiles. Prior work suggests that different tasks rely on different regions of an LLM during inference (Skean et al., 2025). We examine this effect by computing the spectral entropy of layer-wise hidden representations (Equation 1) on calibration prompts from several tasks, the entropy profiles vary across tasks and depth, indicating that task inputs induce distinct representational structure across layers. This supports using task-conditioned hidden-state statistics as a signal for precision allocation.

3.1. Layer-wise task-aware quantization framework

Given a downstream task, let $\mathcal{D}_{\text{task}}$ denote the unknown task distribution, and let $\mathcal{D}_{\text{calib}} = \{x_i\}_{i=1}^{n}$ be a small unlabeled calibration set sampled from its input marginal. Our goal is to allocate precision across layers so as to preserve task performance while satisfying a memory or compute budget.

Let f_θ be a language model with L transformer blocks and hidden size d. Given a task loss $\mathcal{L}$ or surrogate objective and a layer-wise cost model $c_\ell(b_\ell)$, we select a per-layer precision assignment $\mathbf{b} = (b_1, \ldots, b_L)$, where $b_\ell \in \{4, 8, 16\}$. The resulting quantized model is denoted by $f_\theta^{\mathbf{b}}$. Formally, we aim to solve:

$$\min_{\mathbf{b}} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{task}}}\left[\mathcal{L}\big(f_\theta^{\mathbf{b}}(x), y\big)\right] \quad \text{s.t.} \quad \sum_{\ell=1}^{L} c_\ell(b_\ell) \le B, \tag{2}$$

where B denotes the overall budget. Directly solving Eq. (2) is expensive because it would require evaluating many candidate bit allocations under the downstream task objective. We therefore approximate it by estimating task-conditioned layer-importance scores s_ℓ from the calibration

set $\mathcal{D}_{\text{calib}}$, and allocate higher precision to layers with larger scores. In our experiments, TAQ-IS and TAQ-KL instantiate this framework with $b_\ell \in \{4, 8\}$, while TAQ-O uses $b_\ell \in \{4, 16\}$.

Quantization operator: weight-only, group-wise affine. For each linear weight matrix, we apply weight-only per-group affine quantization with group size G. For a weight group $w \in \mathbb{R}^G$ and bitwidth $b \in \{4, 8\}$, let $Q_b = 2^b - 1$. The integer quantized weights and their dequantized approximation are given by

$$q = \text{clip}\Big(\text{round}\Big(\frac{w}{\Delta} + z\Big),\ 0,\ Q_b\Big), \qquad \hat{w} = \Delta\,(q - z), \tag{3}$$

where Δ and z are the group-wise scale and zero-point, respectively. Layers assigned $b_\ell = 16$ remain in FP16 and are not quantized. All methods below use the same quantization operator in Eq. (3); they differ only in how the layer importance scores s_ℓ are computed and how these scores determine the final precision assignment $\mathbf{b}$.

3.2. TAQ-IS: Task-aware quantization via information and stability

TAQ-IS estimates task-relevant layer importance scores directly from hidden representations, combining information content and activation stability computed on a small task-specific calibration set.

Layer representations. Given an input x tokenized to length T, let $H_\ell(x) \in \mathbb{R}^{T\times d}$ denote the hidden states output by layer ℓ, restricted to valid (non-padding) tokens. TAQ-IS scores each layer using two calibration-time signals: (i) *information*, quantified via matrix entropy, and (ii) *stability*, measured via activation variance.

Information via matrix entropy. We collect a reservoir of r_ℓ token vectors from $H_\ell(x)$ across $x \sim \mathcal{D}_{\text{calib}}$ and form a matrix $R_\ell \in \mathbb{R}^{r_\ell\times d}$ (rows are sampled token representations). Let $Z_\ell = R_\ell - \mathbf{1}\mu_\ell^\top$ be the centered matrix with mean $\mu_\ell = \frac{1}{r_\ell}\sum_{i=1}^{r_\ell} R_{\ell,i}$. Define the (scaled) covariance

$$C_\ell = \frac{1}{r_\ell}\, Z_\ell^\top Z_\ell \in \mathbb{R}^{d\times d}. \tag{4}$$

Let $\{\lambda_{\ell,i}\}_{i=1}^d$ be the eigenvalues of C_ℓ (nonnegative), and normalize them as $p_{\ell,i} = \lambda_{\ell,i}/\sum_j \lambda_{\ell,j}$. The information score is the entropy of this spectrum:

$$\text{Info}_\ell = -\sum_{i=1}^{d} p_{\ell,i}\,\log(p_{\ell,i} + \varepsilon)\,. \tag{5}$$

Stability via activation variance. Let h denote a scalar activation entry drawn from $H_\ell(x)$, aggregated over all valid tokens, hidden dimensions, and calibration examples. We estimate the empirical variance

$$\text{Var}_\ell = \mathbb{E}[h^2] - \big(\mathbb{E}[h]\big)^2, \qquad \text{Stab}_\ell = -\text{Var}_\ell, \tag{6}$$

so larger Stab_ℓ corresponds to more stable (lower-variance) activations. We z-normalize both signals across layers, $z(\cdot)$, and combine them:

$$s_\ell = \alpha\, z(\text{Info}_\ell) + \beta\, z(\text{Stab}_\ell), \tag{7}$$

with $\alpha, \beta \geq 0$ (we use $\alpha = \beta = 0.5$). We then assign 8-bit to the top $K\%$ layers by s_ℓ and 4-bit to the remaining layers. This yields a task-conditioned mixed-precision model without requiring task labels beyond the calibration prompts.

Residual-stream stability under quantization. TAQ-IS scores should remain meaningful when layers are quantized jointly, not only in isolated per-layer analysis. To test this, we compute TAQ-IS scores on the clean FP16 model, quantize the first eight transformer blocks to 4-bit, and recompute the downstream scores. As shown in Figure 2 for Qwen2.5-7B/TriviaQA, the clean and contaminated scores remain closely aligned. This also suggests that TAQ-IS captures a stable task signal in the residual stream and identifies the layers most relevant to the task, rather than relying on unrelated artifacts. We provide a fuller analysis in Table 4.

3.3. TAQ-O: Task-aware quantization via oracle layer sensitivity

TAQ-O is a label-informed diagnostic reference, not a deployable quantization policy. It estimates local layer sensitivity by measuring how much a task metric changes when a single layer is quantized while all other layers remain in FP16.

Per-layer sensitivity. Let $\mathcal{M}(\cdot)$ be a task metric (e.g., averaged EM/F1) computed on a small held-out subset from $\mathcal{D}_{\text{calib}}$, and let $\mathcal{S}_{\text{base}} = \mathcal{M}(f_\theta)$ be the baseline score in FP16. For each layer ℓ, define a model $f^{(\ell)}$ where *only* layer ℓ is quantized to low precision (4-bit) and all other layers remain FP16; the oracle drop is

$$f^{(\ell)} = f_{\theta(\,b_\ell=4,\ b_{j\neq\ell}=16\,)}, \tag{8}$$

$$\Delta_\ell = \max\Big(0,\ \mathcal{S}_{\text{base}} - \mathcal{M}\Big(f^{(\ell)}\Big)\Big)\,. \tag{9}$$

Larger Δ_ℓ indicates a layer whose low-bit quantization harms task performance more. In practice, Δ_ℓ is estimated over a fixed held-out calibration subset to ensure consistent comparisons across layers.

Allocation. We keep a small set of edge layers $\mathcal{E}$ (first and last blocks) in FP16 for robustness, and additionally keep the top-k layers by Δ_ℓ in FP16. All remaining layers are quantized to 4-bit. Because TAQ-O evaluates layers marginally and uses task labels, it should not be interpreted as a global optimum or theoretical upper bound under joint quantization. We include it only to compare label-free proxies against a task-metric-aligned layer ranking.

3.4. TAQ-KL: Task-aware quantization via output-sensitive KL scoring

TAQ-KL estimates layer importance by how strongly perturbations at a given layer change the model's output distribution, measured via KL divergence. TAQ-KL provides a label-free alternative to TAQ-O. Rather than evaluating task-metric degradation after quantizing each layer, it injects a small quantization-like perturbation at one layer at a time and measures the induced change in the model's next-token distribution. Thus, TAQ-KL is task-conditioned through the calibration prompts but does not require task labels.

Noise model and logits. For an input x, let $z(x) \in \mathbb{R}^{|\mathcal{V}|}$ be the baseline logits at the final position. For each layer ℓ, we inject additive noise into its hidden states to mimic quantization error:

$$H'_\ell(x) = H_\ell(x) + \eta \quad (10)$$

$$\eta \sim \mathcal{U}\left(-\frac{\delta_\ell}{2}, \frac{\delta_\ell}{2}\right) \quad (11)$$

where δ_ℓ is a layer-dependent scale. We set δ_ℓ using a simple range-based proxy: let r_ℓ be the average per-example range of the last-token hidden state at layer ℓ over calibration prompts, then $\delta_\ell \approx r_\ell/(2^4 - 1)$, approximating the step size of a 4-bit uniform quantizer. Let $z'_\ell(x)$ denote the logits produced under the forward pass injected with noise.

KL-based sensitivity. With temperature $T > 0$, define

$$p(x) = \mathrm{softmax}\left(\frac{z(x)}{T}\right), \quad q_\ell(x) = \mathrm{softmax}\left(\frac{z'_\ell(x)}{T}\right).$$

We score each layer by expected KL divergence over calibration prompts:

$$s_\ell = \mathbb{E}_{x \sim \mathcal{D}_{\text{calib}}} \left[\mathrm{KL}(p(x) \,\|\, q_\ell(x))\right]. \quad (12)$$

Higher s_ℓ indicates that perturbations at layer ℓ induce larger changes in the output distribution, suggesting a greater need for higher precision at that layer. As in TAQ-IS, we allocate 8-bit to the top $K\%$ layers by s_ℓ and 4-bit to the rest:

4. Experiments

We evaluate *per-task* PTQ for LLMs through four questions: **RQ1.** Does protecting task-relevant layers improve the accuracy–memory trade-off over standard task-agnostic PTQ baselines? **RQ2.** Can TAQ remain competitive with, or outperform, a stronger less-constrained mixed-precision baseline whose realized precision budget can be higher? **RQ3.** Do label-free signals recover oracle-like layer rankings for precision allocation? **RQ4.** Does TAQ remain effective under imperfect or mixed-task calibration?

Experimental Setting: Tasks and metrics. We evaluate performance across three diverse domains spanning knowledge retrieval, code understanding, and mathematical reasoning: (i) TriviaQA (Knowledge Retrieval) (Joshi et al., 2017), reporting Exact Match (EM) and token-level F1; (ii) CodeMMLU (Code Understanding) (Nguyen et al., 2025), reporting EM; and (iii) MMLU-Pro (Math/Reasoning) (Wang et al., 2024), reporting EM. **Models.** We evaluate four open-weight instruction-tuned LLMs spanning two model families and several parameter scales: Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct (Hui et al., 2024), Qwen2-7B-Instruct (Yang et al., 2024), and Gemma-2-9B-it (Team et al., 2024). **Baselines and comparison regimes.** We benchmark against representative PTQ baselines under three complementary regimes. (i) Standard uniform low-bit PTQ baselines, GPTQ (Frantar et al., 2022) and AWQ (Lin et al., 2024), which represent canonical 4-bit deployment. (ii) A stronger mixed-precision reference, SliM-LLM (Huang et al., 2024), which has greater allocation freedom and, in most cases, a higher realized precision budget than most TAQ variants. (iii) Structural controls (Last25, Uniform 8-bit; subsection 8.1) . We also include W\O (FP16) as the full-precision reference. All methods use identical prompts, splits, metric code, and unified realized-weight-footprint accounting (section 9); we compare them under realistic accuracy–memory (GB) budgets. **Quantization protocol.** We focus on weight-only quantization using group-wise affine quantization with a group size of G=128. Per-layer bitwidths are selected from $\{4, 8, 16\}$, where 16 denotes FP16 retention. The methods differ strictly in their layer ranking and precision allocation policies, as detailed in §3.1. **Calibration data.** For each benchmark, we construct disjoint calibration (D_{calib}) and test sets containing 512 and 2,048 examples, respectively. D_{calib} is used to compute layer-importance scores (TAQ-IS, TAQ-KL), derive oracle sensitivities (TAQ-O), and fit quantizer parameters; no gradient-based updates are applied. For TriviaQA, we use a fixed 512-example subset of the validation split for calibration and a disjoint 2,048-example subset for evaluation. For MMLU-Pro and CodeMMLU, we use stratified subsets from the official evaluation splits; calibration and evaluation examples are disjoint and sampled with a fixed seed. **Memory reporting.** We report W as the realized weight footprint in GB under the unified accounting in section 9: quantized transformer-linear weights, quantization metadata, FP16-retained blocks, embeddings/`lm_head`, and norms. Average bitwidths are reported over transformer-linear layers. **TAQ implementation details. (1) TAQ-IS** ranks layers using $s_\ell = \alpha z(\mathrm{Info}_\ell) + \beta z(\mathrm{Stab}_\ell)$, where $\mathrm{Stab}_\ell = -\mathrm{Var}_\ell$, with $\alpha = \beta = 0.5$, using a reservoir of $r = 256$ token vectors per layer. It assigns 8-bit precision to the top K=25% of layers and 4-bit to the remainder. **(2) TAQ-O** establishes an oracle baseline by quantizing layers individually to 4-bit and retaining FP16 for the top-k=8 most sensitive layers (plus the first/last 2 layers), using $n_{\text{sens}} = 16$ held-out calibration examples. **(3) TAQ-KL** scores layers via the KL divergence between baseline and perturbed outputs (temperature T=1) under uniform noise injection

$\eta \sim \mathcal{U}(-\Delta_\ell/2, \Delta_\ell/2)$. It promotes the top $K{=}25\%$ layers to 8-bit. As baselines have different bit allocations and thus a structural advantage, we report Pareto comparisons across the accuracy–memory frontier rather than budget-matched sweeps.

Experimental results Table 1 compares TAQ-IS, TAQ-O, and TAQ-KL against FP16, GPTQ, AWQ, and SliM-LLM on MMLU-Pro and CodeMMLU. We additionally evaluate on a knowledge-retrieval benchmark (TriviaQA); the corresponding full results table and exact bit allocations are deferred to subsection 6.4 and subsection 6.2, respectively. **Accuracy–memory trade-off vs. task-agnostic PTQ (RQ1).** On MMLU-Pro and CodeMMLU, the best TAQ variant improves the accuracy–memory trade-off over standard task-agnostic baselines on every backbone. On CodeMMLU/Qwen2.5-3B, TAQ-KL reaches 46.83% EM at $2.25\,$GB, compared to GPTQ's 24.66% at $1.93\,$GB and AWQ's 20.61% at $2.50\,$GB. On MMLU-Pro/Qwen2.5-7B, TAQ-KL reaches 35.35% EM at $5.95\,$GB, outperforming AWQ by nearly 12 points with only a $0.76\,$GB premium over AWQ's $5.19\,$GB. Similar patterns hold on Qwen2-7B and Gemma-2-9B. Pareto views of accuracy vs. memory are provided in subsection 6.1. **Comparison against a stronger mixed-precision reference (RQ2).** SliM-LLM is a less-constrained mixed-precision baseline with greater allocation freedom than the fixed TAQ-IS/TAQ-KL policy and, in our setup, a substantially higher realized precision budget. TAQ-IS/TAQ-KL use a constrained $4/8$-bit policy in which only the top 25% of layers receive 8-bit precision, yielding an average of 5.00–5.05 bits per transformer layer. Under the linear-weight effective-bit accounting, SliM-LLM uses approximately 7.17, 6.13, 6.14, and 7.39 bits on Qwen2.5-3B, Qwen2.5-7B, Qwen2-7B, and Gemma-2-9B, respectively, while TAQ-IS/TAQ-KL use approximately 4.87, 4.60, 4.61, and 5.05 bits (Table 2). SliM-LLM is therefore a deliberately challenging reference: more allocation freedom and a higher effective bit budget. Despite this asymmetry, TAQ remains competitive with or better than SliM-LLM on the structured tasks: on MMLU-Pro, the best TAQ variant outperforms SliM-LLM on every backbone (33.01 vs. 30.86 on Qwen2.5-3B; 35.35 vs. 31.01 on Qwen2.5-7B; 39.11 vs. 38.48 on Qwen2-7B; 42.48 vs. 39.89 on Gemma-2-9B); on CodeMMLU, TAQ improves over SliM-LLM on Qwen2.5-3B, Qwen2.5-7B, and Gemma-2-9B, and lands within $0.10\,$EM on Qwen2-7B. Overall, the best TAQ variant matches or exceeds SliM-LLM in 7 of the 8 MMLU-Pro/CodeMMLU model–task pairs while spending roughly 1–2 fewer effective bits per linear layer, indicating that the gains do not stem from greater bit budget or allocation freedom but from *where* precision is spent.

Label-free proxies vs. oracle (RQ3). Label-free TAQ-IS and TAQ-KL closely track the label-informed TAQ-O reference. On CodeMMLU/Qwen2.5-3B, TAQ-KL nearly matches TAQ-O while using less memory (46.83% vs. 47.41%; $2.25\,$GB vs. $3.20\,$GB), and on MMLU-Pro/Qwen2.5-7B TAQ-KL exceeds TAQ-O (35.35% vs. 32.13% at lower memory), consistent with TAQ-O being a marginal-sensitivity diagnostic rather than a true upper bound under joint quantization. Together, these results indicate that representation- and output-sensitivity signals are useful proxies for layer importance without requiring labeled validation data.

(1) Ablation studies. We test whether TAQ is robust to three practical choices: calibration size, bit budget, and scoring signal. First, we vary the calibration size, $|D_{\text{calib}}| \in \{64, 128, 256, 512, 1024\}$, and find that EM changes by only $\sim$1.3 points on Qwen2.5-7B/MMLU-Pro, showing that small calibration sets are sufficient (Table 7). Second, we vary the bit budget by promoting the top $K\% \in \{10, 25, 50\}\%$ layers to 8-bit. Performance stays in a narrow range: $K{=}25\%$ gives the best MMLU-Pro result, while $K{=}10\%$ is slightly better on CodeMMLU. Since increasing the budget to 50% does not improve performance, the benefit is not explained by simply protecting more layers; selecting the right layers matters, so we use $K{=}25\%$ as a strong default trade-off (Table 5). Some TAQ configurations exceed FP16 accuracy; we interpret these cases cautiously. A possible explanation is that task-conditioned allocation preserves precision in layers most relevant to the calibration task while compressing less relevant layers more aggressively, acting as an implicit regularizer. This may also explain why performance can drop when K is too high. Finally, we test which scoring signals best select the 8-bit layers. For TAQ-IS, we compare entropy-only, variance-only, and the default combined score with the model, calibration size, and $K{=}25\%$ fixed. Variance-only works best on MMLU-Pro, entropy-only works best on CodeMMLU, and the combined score is most stable overall. Across TAQ scoring families, different tasks favor different signals, with TAQ-KL leading on reasoning and TAQ-O leading on code (Table 6, Table 4). **(2) Additional baselines and controls.** We add two control baselines to separate the effect of task-aware layer selection from the effect of simply using more precision. First, **Last25** uses the same $4/8$-bit budget as TAQ-IS and TAQ-KL, but assigns 8-bit precision to the last 25% of layers. Second, **Uniform 8-bit** assigns 8-bit precision to every layer, providing a higher-memory reference for the effect of increasing precision globally. The results show that task-aware selection matters. On MMLU-Pro, at least one TAQ variant outperforms Last25 on every backbone, for example 35.35 vs. 31.35 on Qwen2.5-7B and 42.48 vs. 40.97 on Gemma-2-9B. Compared with Uniform 8-bit, TAQ is not always higher in raw EM, but it is often competitive while using substantially less memory, and on Qwen2.5-7B and Gemma-2-9B the best TAQ variant also exceeds the available Uniform 8-bit result. Together, these controls show that TAQ's improvements are not explained simply by

Table 1. **Main results. W** is the per-model quantized weight size (GB). **Bold** marks the best and underlined the second-best quantized result. TriviaQA results are deferred to subsection 6.4.

Method	Math Reasoning: MMLU-Pro (EM↑)								Code Understanding: CodeMMLU (EM↑)							
	Qwen2.5-3B		Qwen2.5-7B		Qwen2-7B		Gemma-2-9B		Qwen2.5-3B		Qwen2.5-7B		Qwen2-7B		Gemma-2-9B	
	EM↑	W↓	EM↑	W↓	EM↑	W↓	EM↑	W↓	EM↑	W↓	EM↑	W↓	EM↑	W↓	EM↑	W↓
W\O (FP16)	37.50	5.75	33.20	14.19	41.06	14.19	42.48	17.21	50.29	5.75	48.78	14.19	51.32	14.19	53.52	17.21
GPTQ	13.57	1.93	18.51	5.19	23.83	5.19	40.63	5.75	24.66	1.93	34.91	5.19	35.16	5.19	**51.27**	5.75
AWQ	09.86	2.50	23.54	5.19	29.15	5.19	41.60	5.74	20.61	2.50	40.28	5.19	40.09	5.19	50.68	5.74
SliM-LLM	30.86	2.77	31.01	5.83	38.48	5.83	39.89	5.83	47.04	2.77	47.17	5.83	**49.32**	5.83	50.44	8.31
TAQ-IS (Ours)	30.76	2.25	**33.35**	5.95	37.99	5.95	41.60	6.76	46.29	2.25	47.22	5.95	48.49	5.95	51.03	6.76
TAQ-O (Ours)	**33.01**	3.20	32.13	9.05	38.57	9.05	39.40	9.02	**47.41**	3.20	**49.51**	9.05	49.22	9.05	50.54	9.02
TAQ-KL (Ours)	28.03	2.25	**35.35**	5.95	**39.11**	5.95	**42.48**	6.76	46.83	2.25	47.02	5.95	47.12	5.95	50.83	6.76

assigning 8-bit precision to the last layers or by using uniformly higher precision; the task-aware choice of protected layers is important. Full results are reported in Table 8. **(3) Mixed-task calibration.** We next test whether TAQ remains effective on mixed-task calibration data, addressing **RQ4**. We consider two settings. First, instead of computing a separate TAQ-IS allocation for each benchmark, we compute one allocation on Qwen2.5-7B using a mixed calibration set of 512 prompts sampled from TriviaQA, CodeMMLU, and MMLU-Pro. We then evaluate this same allocation separately on each benchmark. The mixed allocation improves over the single-task allocation in our run: CodeMMLU increases from 47.22 to 49.61 EM, MMLU-Pro from 33.35 to 34.28 EM, and TriviaQA from 12.55 to 17.09 EM (Table 9). We hypothesize that using a broader calibration pool can reduce overfitting. **(4) Cross-task allocation stability.** Second, we check how much the selected layers actually change across tasks and models. Across the 5×3 model/dataset grid, 87.5% of layer assignments remain the same across calibration tasks, with cross-task Spearman correlation $\rho = 0.757$. The layers that do change are mostly near the first and last transformer blocks, consistent with the boundary-layer pattern in Figure 7. Together, these results suggest that many important layers are shared across tasks, while task-specific differences are concentrated in a small number of layers. Thus, TAQ can often produce a robust shared allocation. **(5) Calibration source vs. allocation policy.** We also test whether the gains come from task-aligned calibration data alone, or from the task-aware allocation policy itself. In subsection 8.3, we evaluate on a broader AWQ-supported model suite: Phi-4, Qwen3, Llama-3.1, Qwen2.5, and Mistral. We compare three settings: **AWQ-Mixed**, where AWQ is calibrated on a mixed pool from three tasks; **AWQ-OT**, where AWQ is calibrated only on the target task; and our **TAQ-IS** and **TAQ-O** task-aware allocation policies. If task-aligned calibration alone were sufficient, AWQ-OT should consistently improve over AWQ-Mixed. Instead, AWQ-OT is unstable: it improves Llama-3.1, but degrades Phi-4 and Qwen2.5, and changes Mistral only slightly. In contrast, the best quantized results come from TAQ-IS or TAQ-O. **(6) Hardware and deployment validation.** We validate TAQ with real quantized kernels on a single NVIDIA A40, comparing against FP16/FP32, uniform 4-bit, GPTQ-Int4, and AWQ-Int4 across three backbones and two batch sizes. On Qwen2.5–7B at batch size 1, TAQ-IS improves throughput from 27.63 to 37.29 tok/s relative to FP16, giving $+35\%$ throughput and -26% latency; the gain remains positive at batch size 8 ($+11.6\%$; section 9). TAQ-IS also outperforms uniform 4-bit (28.71 tok/s), GPTQ-Int4 (4.40 tok/s), and AWQ-Int4 (16.12 tok/s). Under the unified memory accounting in section 9, TAQ-IS/TAQ-KL use 5.95 GB on Qwen2.5–7B, a 0.76 GB premium over uniform 4-bit GPTQ/AWQ (5.19 GB), while preserving 8-bit precision on task-sensitive layers. With real NF4/Int8 kernels, Qwen2.5–3B realizes 2.20 GB weight size and 2.31 GB peak memory, corresponding to $2.6\times$ compression over FP16. The slight difference from the accounting-based W reported in Table 1 (2.25 GB for Qwen2.5–3B TAQ-IS/TAQ-KL) comes from backend-specific packing in the realized NF4/Int8 kernels. Full throughput, latency, and memory breakdowns are in section 9. **(6) Inter-layer score stability and residual-stream propagation.** We test whether layer scores from clean FP16 activations remain reliable after upstream 4-bit quantization perturbs the residual stream. Since TAQ uses these scores mainly to choose higher-precision layers, we evaluate rank preservation via Spearman's ρ between clean FP16 rankings and rankings recomputed after quantizing the first eight transformer blocks. Rankings remain highly stable on TriviaQA, with combined-score correlations of ρ=0.997/0.999/0.990 on Qwen2.5–7B and ρ=0.967/0.969/0.952 on Gemma-2–9B (TAQ-IS/TAQ-O/TAQ-KL); Qwen2.5–7B is also stable on CodeMMLU ($\rho\approx$0.75–0.80). The main weaker case is Gemma-2–9B on CodeMMLU and MMLU-Pro ($\rho\approx$0.38–0.59), indicating stronger task-dependent inter-layer interactions where early-layer quantization can alter later-layer rankings. Complementary cosine similarity is also positive: after an initial drop at the cutoff, it stabilizes around 0.93, so residual-stream perturbations do not grow with depth. Full results are in section 10.

Impact Statement

This work introduces a task-aware post-training quantization approach that allocates precision using signals derived from an LLM's hidden representations, rather than relying on task-agnostic reconstruction proxies or additional training. By turning quantization into a per-task allocation problem, our method can preserve task performance under tight memory and compute budgets, enabling more efficient deployment of specialized LLM applications.

The primary positive impact is improved accessibility: reducing inference cost and memory footprint can lower hardware barriers, decrease energy consumption per request, and make it feasible to run task-focused models in resource-constrained settings. Because our approach requires only a small set of unlabeled, task-specific prompts and performs no gradient updates, it can also simplify maintenance compared to repeated fine-tuning cycles, supporting faster iteration and easier auditing of task-specific behavior.

These efficiency gains can also strengthen responsible deployment by making it easier to run comprehensive evaluations under realistic serving constraints. Lower inference cost enables more extensive testing across prompt variants and edge cases, while task-specific calibration encourages practitioners to explicitly define the intended use and validate performance in that setting. We recommend pairing task-aware quantization, and verifying stability under representative prompt variability prior to deployment. We extend the discussion of broader impact, limitations, and future work in Appendix 5.1.

References

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332, 2022.

Dettmers, T., Svirschevski, R., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., and Alistarh, D. SpQR: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. HAWQ: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 293–302, 2019.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*, pp. 291–326. Chapman and Hall/CRC, 2022.

Guan, Z., Huang, H., Su, Y., Huang, H., Wong, N., and Yu, H. APTQ: Attention-aware post-training mixed-precision quantization for large language models. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pp. 1–6, 2024.

Heimersheim, S. and Nanda, N. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024.

Hendel, R., Geva, M., and Globerson, A. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, 2023.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Huang, W., Qin, H., Liu, Y., Li, Y., Liu, Q., Liu, X., Benini, L., Magno, M., Zhang, S., and Qi, X. SliM-LLM: Salience-driven mixed-precision quantization for large language models. *arXiv preprint arXiv:2405.14917*, 2024.

Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Lu, K., et al. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Jawahar, G., Sagot, B., and Seddah, D. What does bert learn about the structure of language? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 3651–3657, 2019.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. AWQ: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100, 2024.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in GPT. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Nguyen, D. M., Phan, T. C., Le Hai, N., Doan, T.-T., Nguyen, N. V., Pham, Q., and Bui, N. D. CodeMMLU: A multi-task benchmark for assessing code understanding & reasoning capabilities of codellms. In *The Thirteenth International Conference on Learning Representations*, 2025.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Rahimi, E., Hirshel, E., Himelstein, R., LeVi, A., Mendelson, A., and Baskin, C. Step-wise refusal dynamics in autoregressive and diffusion language models. *arXiv preprint arXiv:2602.02600*, 2026.

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Shao, W., Chen, M., Zhang, Z., Xu, P., Zhao, L., Li, Z., Zhang, K., Gao, P., Qiao, Y., and Luo, P. OmniQuant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

Sharkey, L., Chughtai, B., Batson, J., Lindsey, J., Wu, J., Bushnaq, L., Goldowsky-Dill, N., Heimersheim, S., Ortega, A., Bloom, J., et al. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*, 2025.

Shnaidman, A., Feiglin, E., Yaari, O., Mentel, E., Levi, A., and Lapid, R. Activation steering for masked diffusion language models. *arXiv preprint arXiv:2512.24143*, 2025.

Skean, O., Arefin, M. R., Zhao, D., Patel, N., Naghiyev, J., LeCun, Y., and Shwartz-Ziv, R. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Tenney, I., Das, D., and Pavlick, E. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4593–4601, 2019.

Turner, A. M., Thiergart, L., Leech, G., Udell, D., Vazquez, J. J., Mini, U., and MacDiarmid, M. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Yang, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Liu, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Liu, Y., Cui, Z., Zhang, Z., Guo, Z., and Fan, Z. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Zhang, F., Liu, Y., Li, W., Lv, J., Wang, X., and Bai, Q. Towards superior quantization accuracy: A layer-sensitive approach. *arXiv preprint arXiv:2503.06518*, 2025.

Zhao, J., Derakhshan, A., Hyman, J. K., Dong, J., Jyothi, S. A., and Harris, I. CoopQ: Cooperative game inspired

layerwise mixed precision quantization for llms. *arXiv preprint arXiv:2509.15455*, 2025.

Zheng, Z., Song, X., and Liu, C. Mixllm: Llm quantization with global mixed-precision between output-features and highly-efficient system design. *arXiv preprint arXiv:2412.14590*, 2024.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

5. Assets and compute

5.1. Discussion Extension

We present TAQ, a per-task PTQ framework that uses internal representations to identify task-critical layers and allocate bits accordingly, turning quantization into a *policy* problem rather than a uniform compression step. Across most models and tasks, TAQ yields strong accuracy–memory trade-offs, with especially large gains on specialized workloads where strong baselines can degrade sharply. In some cases, quantized TAQ models even outperform the FP16 model. We view this as a hypothesis that task-aware quantization can preserve task-relevant signal while suppressing irrelevant activation or weight noise, acting as a form of task-conditioned denoising rather than only compression. The label-free variants, TAQ-KL and TAQ-IS, often closely match, and occasionally exceed, the TAQ-O oracle reference, suggesting that intrinsic stability and output-sensitivity signals can recover useful task-aware layer rankings. TAQ-O is therefore best understood as a label-informed diagnostic reference under our evaluation protocol, not as a deployable method or theoretical upper bound.

Limitations. TAQ relies on calibration prompts that represent the target deployment distribution; performance may degrade under distribution shift. In addition, TAQ-IS, TAQ-KL, and TAQ-O use tractable local proxies rather than solving the full joint mixed-precision allocation problem, so higher-order interactions among quantized layers may not be fully captured.

Broader impacts. TAQ may have positive societal impacts by making LLM inference more efficient and accessible, reducing memory footprint, latency, and potentially energy costs for task-specific deployments. However, more efficient deployment can also lower barriers to using capable LLMs in harmful or sensitive applications, and task-conditioned quantization may degrade behavior under distribution shift or poorly specified calibration data; therefore, TAQ should be evaluated on target-domain safety, privacy, fairness, and robustness criteria before deployment, especially in high-stakes settings.

Future work. Future work should extend TAQ beyond layer-level allocation and model quantization error more directly. Our linearity analysis suggests that quantization error can appear as a persistent residual-stream direction, motivating interpretability-based corrections such as activation steering or task-direction estimation.

5.2. Motivation

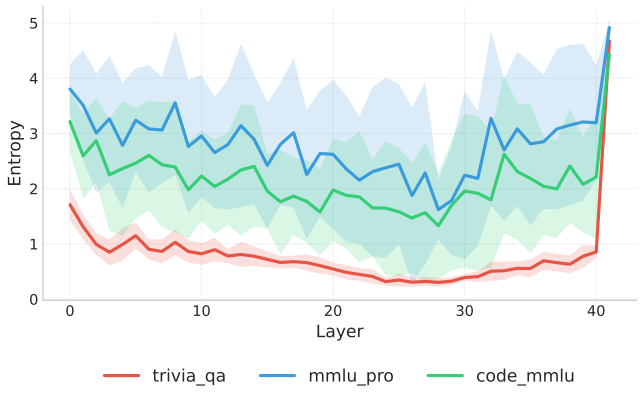

Figure 1. Information entropy across layers with error bars on three tasks(*Gemma2-9B*).

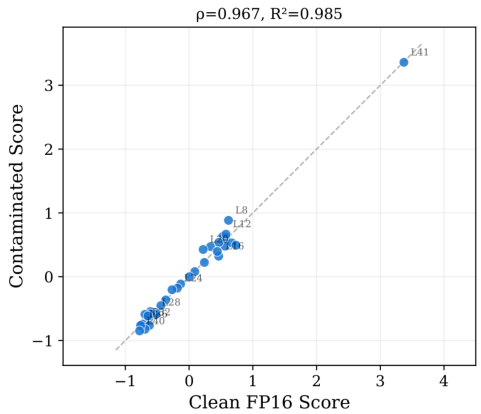

Figure 2. Task-relevant score after (contaminated) quantization.

5.3. Existing assets and licenses

Our experiments use only public third-party datasets, model checkpoints, and baseline methods, and we do not redistribute third-party data or model weights. The datasets are TriviaQA v1.0 validation data (https://nlp.cs.washington.edu/triviaqa/; Apache-2.0 code license, with original question/document copyrights retained by their owners), CodeMMLU (https://huggingface.co/datasets/

Fsoft-AIC/CodeMMLU; MIT), and MMLU-Pro (https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro; MIT dataset license; Apache-2.0 evaluation code). The model checkpoints are Qwen2.5-3B-Instruct (https://huggingface.co/Qwen/Qwen2.5-3B-Instruct; Qwen Research License), Qwen2.5-7B-Instruct (https://huggingface.co/Qwen/Qwen2.5-7B-Instruct; Apache-2.0), Qwen2-7B-Instruct (https://huggingface.co/Qwen/Qwen2-7B-Instruct; Apache-2.0), Gemma-2-9B-it (https://huggingface.co/google/gemma-2-9b-it; Gemma license / Google usage terms), Phi-4 (https://huggingface.co/microsoft/phi-4; MIT), Llama-3.1-8B-Instruct (https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct; Llama 3.1 Community License), and Mistral-7B-Instruct-v0.3 (https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3; Apache-2.0). The auxiliary component-analysis suite in subsection 8.3 also includes a Qwen3 backbone; the submitted experiment table reports this backbone only at the family level, so the exact Qwen3 checkpoint identifier and license should be inserted from the experiment manifest before final submission rather than inferred from the family name. Baselines are credited to GPTQ, AWQ, and SliM-LLM; we use released code/assets when licensed for research use or our own reimplementation of the published method. All assets are cited to their original creators and used only for research and evaluation in accordance with their licenses and terms.

5.4. Compute resources

All reported calibration, quantization, evaluation, ablation, robustness, and hardware-validation experiments were run on single-GPU NVIDIA A40 workers with 48 GB of GPU memory. Each model–task–method configuration used one A40 worker and no distributed training, since TAQ is training-free and performs calibration-time scoring, weight-only quantization, and evaluation rather than gradient-based training. A typical configuration required approximately 1.5–3 A40 GPU-hours for 3B models and 3–6 A40 GPU-hours for 7B/9B models, including calibration/scoring, quantization, and evaluation on $2{,}048$ examples. Hardware throughput and latency measurements required approximately 0.5 A40 GPU-hours per configuration. Across all reported experiments, including baselines, ablations, robustness analyses, and hardware validation, the total compute was approximately 450 A40 GPU-hours. Including preliminary, exploratory, and failed runs not reported in the paper, the full research project required approximately 750 A40 GPU-hours. Persistent storage per run was below 100 GB.

6. Detailed results

6.1. Memory–performance trade-off

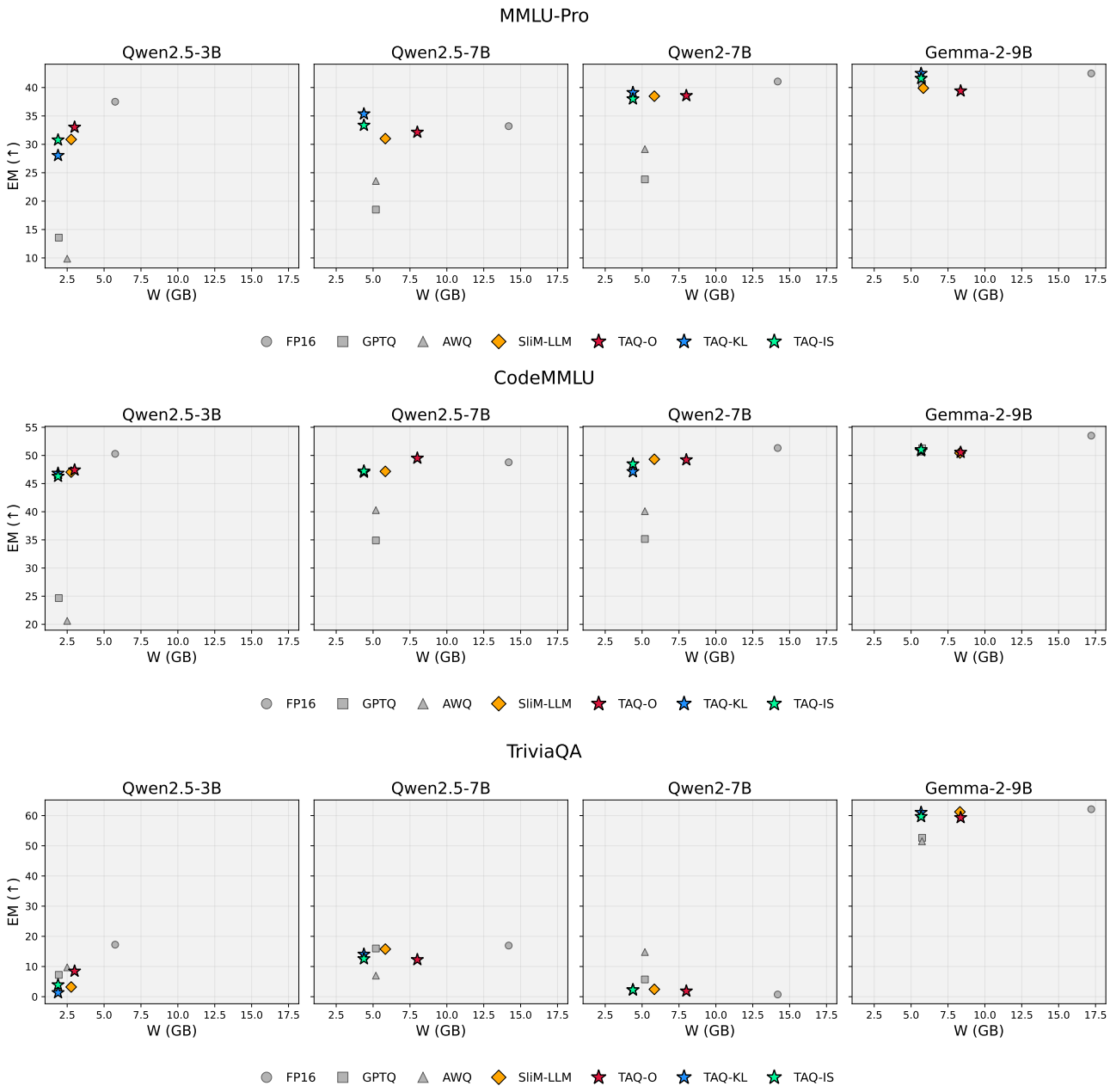

Figure 3. **Accuracy–memory trade-off across benchmarks and models.** Each panel corresponds to one benchmark dataset (MMLU-Pro, CodeMMLU, TriviaQA). Within each benchmark, we plot exact match accuracy (EM, ↑) against the realized weight footprint W in GB for Qwen2.5-3B, Qwen2.5-7B, Qwen2-7B, and Gemma-2-9B. Markers denote FP16, GPTQ, AWQ, SliM-LLM, TAQ-O, TAQ-KL, and TAQ-IS. Points closer to the upper-left indicate better accuracy–memory efficiency.

6.2. Layerwise precision allocation

For TAQ-IS and TAQ-KL, the top 25% of layers are assigned 8-bit precision while the remaining layers are assigned 4-bit precision. TAQ-O uses a different diagnostic allocation: the selected sensitive layers are retained in FP16 and the remaining layers are quantized to 4-bit. For visual comparability, the allocation plots show TAQ-O retained layers on the high-precision

row; semantically these layers are FP16-retained and are accounted as 16-bit in the memory tables.

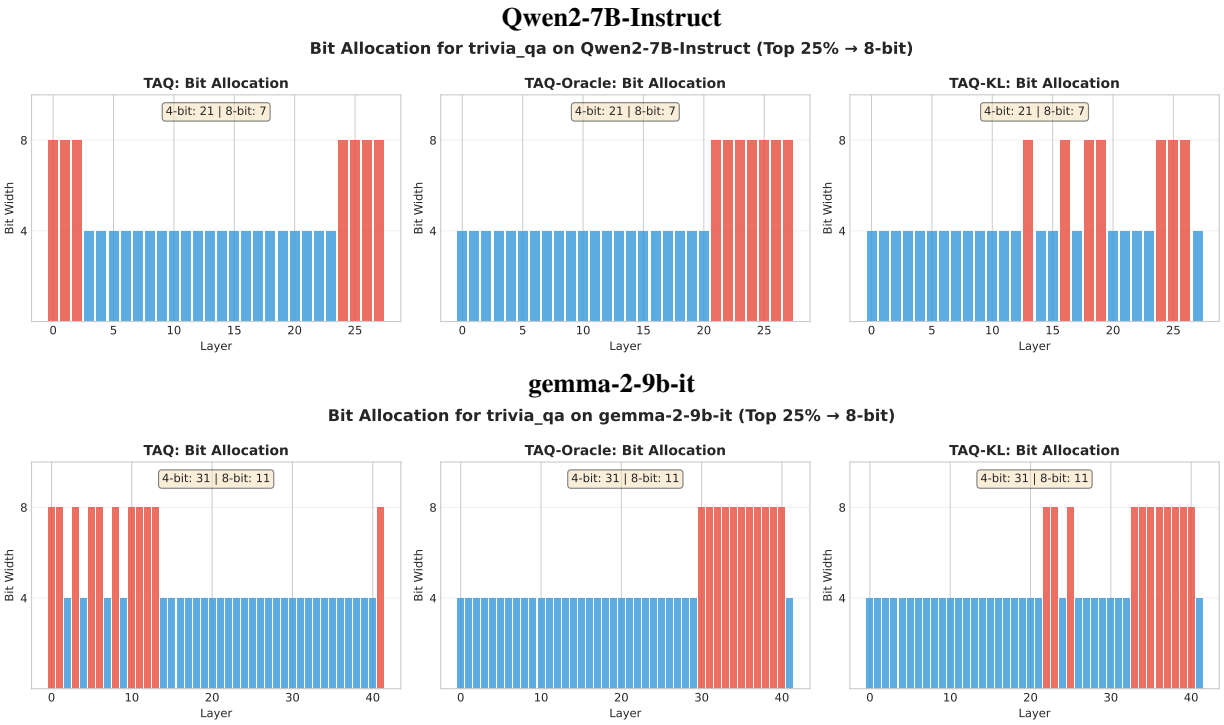

Figure 4. Layerwise high-precision allocation on TriviaQA.

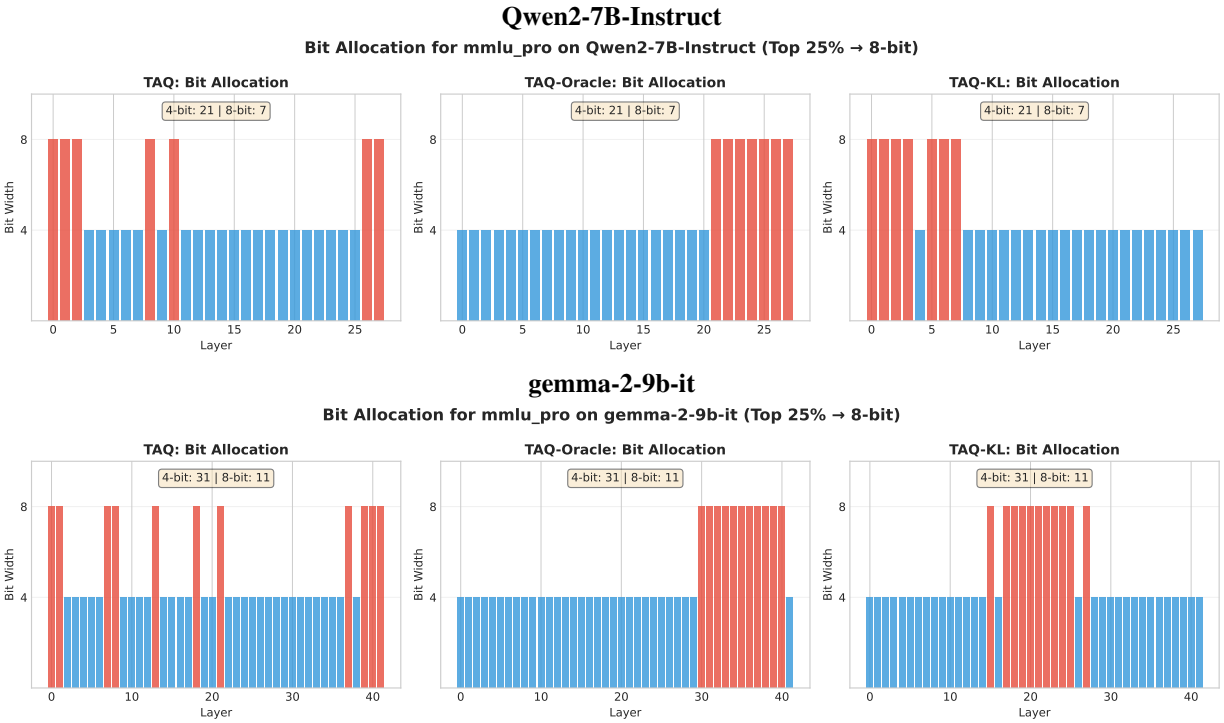

Figure 5. Layerwise high-precision allocation on MMLU-Pro.

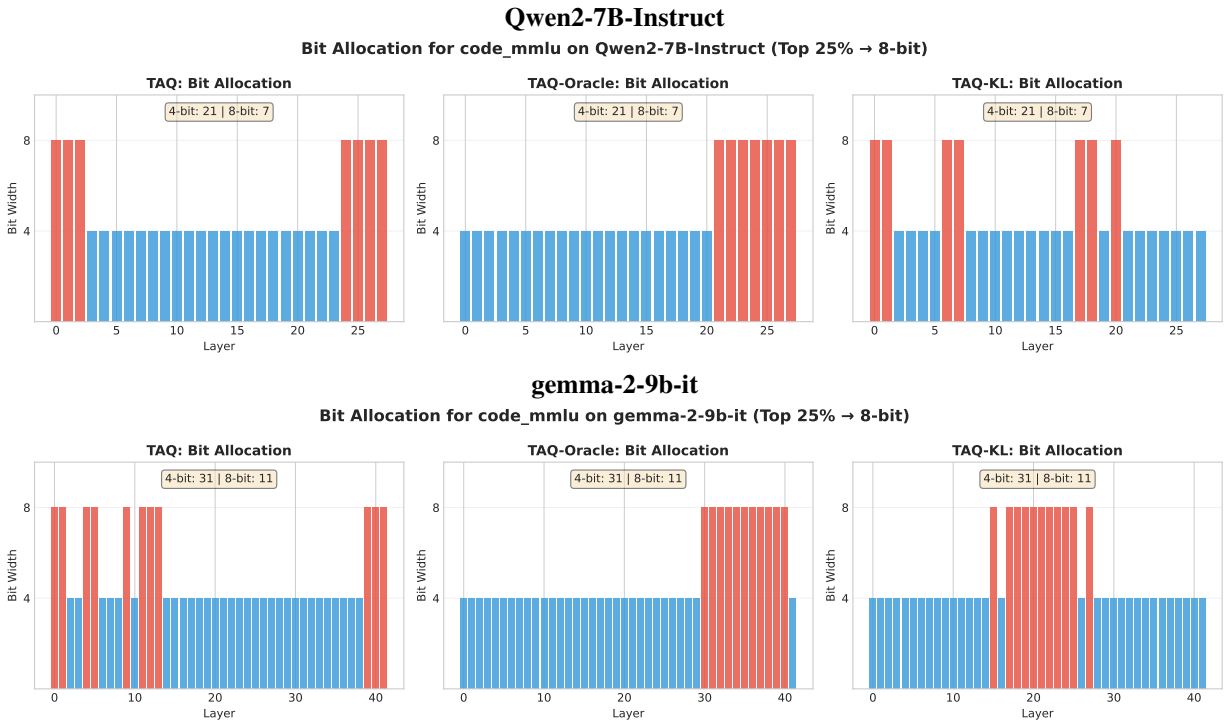

Figure 6. Layerwise high-precision allocation on CodeMMLU.

6.3. Precision allocation statistics

Table 2 reports the average bits per transformer-linear layer and the realized weight footprint (GB) under the unified accounting in section 9. Effective-bit numbers are computed on linear weights only (excluding embeddings, `lm_head`, and norms). TAQ-IS/TAQ-KL use a constrained $4/8$-bit policy, while SliM-LLM is a less-constrained mixed-precision baseline whose realized precision budget is substantially higher. This contextualizes the main-paper comparison: the best TAQ variant outperforms SliM-LLM on 7 of 8 MMLU-Pro/CodeMMLU model–task pairs, and is within 0.10 EM on the remaining CodeMMLU/Qwen2-7B pair, while spending roughly 1–2 fewer effective bits per linear layer.

Table 2. **Precision allocation statistics.** Effective bits per linear layer and realized quantized weight size (GB). TAQ-IS/TAQ-KL use a constrained $4/8$-bit policy; SliM-LLM uses a less-constrained mixed-precision allocation with a higher effective bit budget; TAQ-O retains FP16 on selected layers.

Method	Qwen2.5-3B	Qwen2.5-7B	Qwen2-7B	Gemma-2-9B	Llama-3.1-8B
Effective bits per linear layer					
SliM-LLM	7.17	6.13	6.14	7.39	–
TAQ-IS	4.87	4.60	4.61	5.05	5.00
TAQ-KL	4.87	4.60	4.61	5.05	5.00
TAQ-O	7.77	8.41	8.42	7.43	8.50
Realized weight footprint (GB)					
SliM-LLM	2.77	5.83	5.83	8.31	–
TAQ-IS	2.25	5.95	5.95	6.76	5.68
TAQ-KL	2.25	5.95	5.95	6.76	5.68
TAQ-O	3.20	9.05	9.05	9.02	8.36

6.4. Knowledge retrieval: TriviaQA full results

Table 3 reports the full TriviaQA results across the four main backbones.

Table 3. **Knowledge Retrieval — TriviaQA full results** (EM↑, F1↑; W is the realized weight footprint in GB). **Bold** marks the best and underlined the second-best quantized result per metric/backbone. Extended TriviaQA evaluation across additional backbones (Phi-4, Qwen3, Llama-3.1, Mistral) and additional baseline regimes (mixed-calibration AWQ, on-task AWQ-OT) is reported in subsection 8.3.

Method	Qwen2.5-3B			Qwen2.5-7B			Qwen2-7B			Gemma-2-9B		
	EM↑	F1↑	W↓	EM↑	F1↑	W↓	EM↑	F1↑	W↓	EM↑	F1↑	W↓
W\O (FP16)	17.24	24.69	5.75	16.94	25.62	14.19	00.73	12.94	14.19	62.11	68.92	17.21
GPTQ	07.23	15.54	1.93	**15.97**	**26.57**	5.19	05.71	17.10	5.19	52.64	61.04	5.75
AWQ	**09.72**	17.66	2.50	06.98	18.15	5.19	**14.79**	**25.16**	5.19	51.51	60.14	5.74
SliM-LLM	03.22	17.48	2.77	15.77	26.20	5.83	02.44	21.09	5.83	**61.23**	**68.87**	8.31
TAQ-IS (Ours)	03.91	17.14	2.25	12.55	23.95	5.95	02.25	20.79	5.95	59.67	68.08	6.76
TAQ-O (Ours)	08.45	**20.34**	3.20	12.26	23.23	9.05	01.86	20.58	9.05	59.33	67.67	9.02
TAQ-KL (Ours)	01.32	13.69	2.25	14.06	23.83	5.95	02.25	21.17	5.95	61.04	68.58	6.76

7. Sensitivity and ablation studies

This section reports the auxiliary sweeps referenced from Table 4. Table 4 compares the three TAQ scoring families head-to-head on Qwen2.5-7B; Table 5 reports the top-$K\%$ promotion sweep; Table 6 reports the α/β sweep at $K{=}25\%$; and Table 7 reports the calibration-size sensitivity.

Table 4. **Per-metric scoring head-to-head** on Qwen2.5-7B (EM↑). No single rule dominates across tasks.

Metric	MMLU-Pro	CodeMMLU
TAQ-IS	33.35	47.22
TAQ-O	32.13	**49.51**
TAQ-KL	**35.35**	47.02

Table 5. **Top-$K\%$ promotion sweep** for TAQ-IS on Qwen2.5-7B (calib=512, $G{=}128$). Performance varies within a narrow range; $K{=}25\%$ gives the best MMLU-Pro result and is within one EM point of the best CodeMMLU result in this sweep.

$K\%$	8-bit layers	MMLU-Pro EM↑	CodeMMLU EM↑
10%	3	33.25	**48.00**
25%	7	**34.62**	47.22
50%	14	32.52	46.88

Table 6. **α/β sweep for the TAQ-IS combined score** ($K{=}25\%$ fixed).

α	β	Signal	MMLU-Pro EM↑	CodeMMLU EM↑
0.0	1.0	Variance only	**35.11**	48.54
0.5	0.5	Combined (default)	34.62	47.22
1.0	0.0	Entropy only	30.32	**50.44**

Table 7. **Calibration-size sensitivity** for TAQ-IS on Qwen2.5-7B/MMLU-Pro (eval=2,048). EM moves within ∼1.3 EM points across a 16× range.

$\lvert D_{\text{calib}}\rvert$	64	128	256	512 (default)	1024
TAQ-IS EM↑	34.18	**34.62**	33.79	33.35	33.89

8. Additional baselines and robustness

8.1. Structural baselines: Last25 and Uniform 8-bit

Table 8 reports per-model EM on MMLU-Pro for the Last25 and Uniform 8-bit structural references introduced in Table 4.

Table 8. **Additional reference baselines on MMLU-Pro (EM↑).** Last25 matches the bit-allocation pattern of TAQ-IS/TAQ-KL by promoting the last 25% of layers; Uniform 8-bit is a higher-memory reference in which all transformer layers receive 8-bit precision. TAQ rows are reproduced from Table 1 for direct comparison.

Method	Qwen2.5-3B	Qwen2.5-7B	Qwen2-7B	Gemma-2-9B
Last25 (4/8, last-25%)	25.10	31.35	38.53	40.97
Uniform 8-bit	37.11	33.06	40.72	42.12
TAQ-O (Ours)	**33.01**	32.13	38.57	39.40
TAQ-KL (Ours)	28.03	**35.35**	**39.11**	**42.48**
TAQ-IS (Ours)	30.76	33.35	37.99	41.60

8.2. Mixed-task calibration and cross-task allocation stability

Table 9 reports the single-task vs. mixed-task EM comparison on Qwen2.5-7B; mixed-task calibration improves over single-task on every benchmark in this run. Figure 7 is the bit-allocation heatmap aggregated across the 5×3 model/dataset grid.

Table 9. **Mixed-task calibration** on Qwen2.5-7B (TAQ-IS, calib=512, eval=2,048). A single allocation derived from a uniform pool of all three tasks is evaluated per benchmark.

Eval Dataset	Single-task EM	Mixed-task EM	Δ
CodeMMLU	47.22	**49.61**	+2.39
MMLU-Pro	33.35	**34.28**	+0.93
TriviaQA	12.55	**17.09**	+4.54

Across the 5×3 model/dataset grid, 87.5% of layer assignments remain unchanged across calibration tasks, with cross-task Spearman correlation ρ=0.757. The layers that change are mostly near the first and last transformer blocks, indicating that many high-precision assignments are shared across tasks while task-specific differences concentrate in a smaller set of boundary layers.

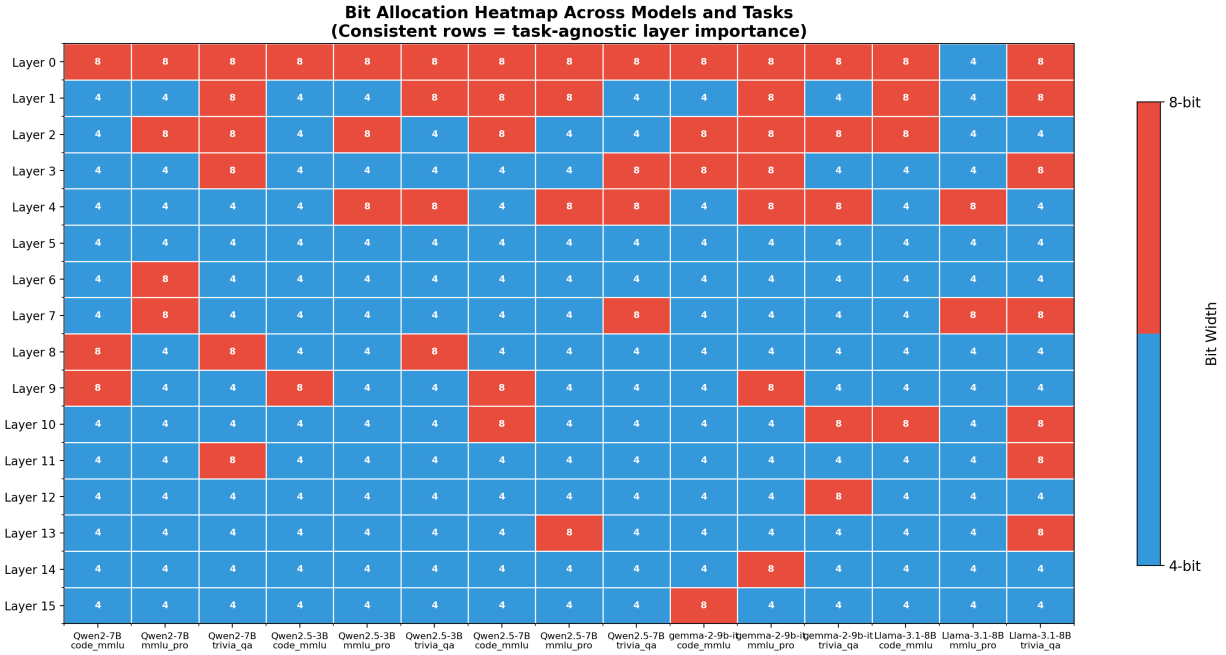

Figure 7. **Bit-allocation heatmap across** 5 **backbones** $\times$ 3 **datasets.** 87.5% of layers receive identical bit-widths regardless of calibration task; task-specific variation concentrates at boundary blocks.

8.3. Calibration source vs. allocation policy

This subsection consolidates the controlled ablation used to disentangle the contribution of *task-aligned calibration data* from the contribution of the *task-aware allocation policy*. We pose two sub-questions: (**AQ1**) If we perfectly align calibration data with the test task (on-task calibration), can a task-agnostic policy (AWQ) recover full-precision performance? (**AQ2**) Does a task-aware allocation policy provide robustness that data alignment alone cannot, especially in regimes where standard minimization is unstable?

We compare three setups on TriviaQA across an auxiliary AWQ-supported model suite: (i) **AWQ-Mixed**: balanced pool of math, code, and trivia (N=2048); (ii) **AWQ-OT**: 512 examples drawn exclusively from TriviaQA (perfect on-task alignment, tests AQ1); (iii) **TAQ-IS / TAQ-O**: our task-aware policies on TriviaQA. We use this auxiliary suite because GPTQ checkpoints were unavailable for some main-experiment models; running AWQ/AWQ-OT under the same PTQ codepath gives a consistent comparison. Since this ablation uses a different sampled setting from the main TriviaQA experiments (512 samples), absolute values may differ slightly from Table 3.

The experiment refutes **AQ1** and supports **AQ2**. Under **AQ1**, AWQ-OT should consistently improve over AWQ-Mixed, but its behavior is unstable: Llama-3.1 improves ($53.37 \rightarrow 55.66$ EM) while Phi-4 collapses ($2.25 \rightarrow 0.73$ EM) and Mistral changes minimally. Under **AQ2**, the best TAQ policy is more stable across this auxiliary suite, nearly matching FP16 on Phi-4 with TAQ-IS (42.33 vs. 43.16 EM) and exceeding AWQ-OT on Qwen3 with TAQ-IS (11.82 vs. 9.81 EM), while TAQ-O gives the highest Qwen3 EM (18.51).

9. Hardware and deployment

We benchmarked all methods on NVIDIA A40 hardware across three models, two batch sizes, and with GPTQ-Int4 and AWQ-Int4 baselines from HuggingFace, for a total of 45 configurations.

9.1. Throughput and latency

The headline results for Qwen2.5–7B are shown in Table 11. TAQ-IS achieves a 35% throughput improvement at batch size 1—a measured speedup on A40 hardware, not a theoretical estimate. At batch size 8, the gain persists at $+11.6\%$ in the full benchmark sweep. For Qwen2–7B against an FP32 baseline, TAQ-O delivers a $4.56\times$ throughput improvement,

increasing throughput from 8.0 to 36.3 tokens/s in the same sweep.

Table 11. Real hardware benchmark results for Qwen2.5–7B on NVIDIA A40 (batch size 1; throughput in tokens/s, latency in ms/token).

Method	Throughput↑	Latency↓	vs. FP16
FP16	27.63	36.25	baseline
TAQ-IS	37.29	26.82	$+35\%$ / -26%
TAQ-KL	32.83	30.48	$+19\%$ / -16%
TAQ-O	27.86	35.93	$+1\%$ / -1%
UNIFORM4	28.71	34.83	$+4\%$ / -4%
GPTQ-Int4	4.40	227.26	-84% / $+527\%$
AWQ-Int4	16.12	62.51	-42% / $+72\%$

9.2. Memory accounting

We compare TAQ's realized weight footprint against uniform 4-bit GPTQ and AWQ under a single convention: W counts the stored weight tensors needed for inference, including quantized linear weights, FP16 group-wise scales with group size G=128, per-group zero-points at the layer's bitwidth, FP16-retained linear blocks (TAQ-O only), input embeddings and `lm_head` in FP16 (tied where the model ties them), and final/RMS layer norms in FP16. Concretely, with $b_\ell \in \{4, 8, 16\}$, per-layer transformer-linear parameter count N_ℓ, vocabulary size V, hidden size h, and $\mathbf{1}_{\text{untied}}$ indicating untied embeddings, We report $W = W_{\text{bytes}}/2^{30}$ in GB. Under this convention, our GPTQ/AWQ numbers reproduce the realized HuggingFace 4-bit checkpoints to within $<2\%$; for example, Qwen2.5–7B gives 5.19 GB.

Table 12. Realized weight footprint for Qwen2.5–7B .

Method	Avg. bits/param	Realized W	Allocation strategy
FP16 baseline	16.0	14.18 GB	none
GPTQ / AWQ	4.0	5.19 GB	uniform 4-bit
TAQ-IS / TAQ-KL	5.0	5.95 GB	task-aware mixed 4/8-bit

The 0.76 GB premium over uniform 4-bit GPTQ/AWQ is consistent with the 5-vs-4 average bit budget; it buys 8-bit precision on the critical 25% of layers identified by our task-aware allocation. The component breakdown for TAQ-IS / TAQ-KL is 3.80 GB quantized linears + 0.10 GB scales + 0.03 GB zero-points + 2.03 GB FP16 embeddings/`lm_head` + $<$0.01 GB norms.

9.3. Real-kernel validation

To demonstrate that TAQ's per-layer bit allocation is deployable with real quantized kernels, we replaced each layer with hardware-accelerated equivalents following TAQ's `bit_map`: NF4 (4-bit) kernels for layers assigned 4-bit, and Int8 kernels for layers assigned 8-bit. Results for Qwen2.5–3B are shown in Table 13.

Table 13. Real-kernel memory validation on Qwen2.5–3B.

Method	Weight size	Peak memory	Compression vs. FP16
FP16 baseline	5.79 GB	5.79 GB	1.0×
GPTQ-Int4	1.93 GB	2.06 GB	3.0×
AWQ-Int4	1.92 GB	1.96 GB	3.0×
TAQ real kernels	2.20 GB	2.31 GB	2.6×

TAQ with real kernels achieves 2.6× memory compression with working inference. The additional 0.27–0.28 GB over GPTQ/AWQ corresponds to the 25% of layers kept at 8-bit precision—the layers identified by our analysis as critical for task performance.

10. Inter-layer analyses

This section consolidates the analyses referenced from Table 4. subsection 10.1 reports per-architecture aggregate statistics, per-layer Spearman correlations, scatter plots, and the cosine-similarity decay vs. distance from the cutoff layer for the inter-layer score-stability experiment.

10.1. Inter-layer score stability under upstream quantization

Table 14 reports per-architecture aggregate statistics for the inter-layer score-stability experiment after quantizing the first eight transformer blocks. The stability-signal Spearman column corresponds to the variance/stability signal, while the combined-score column corresponds to the full TAQ score. Figure 8 and Figure 9 report per-task Spearman correlations between clean FP16 layer-importance rankings and rankings recomputed after quantizing the first eight transformer blocks. Figure 10 and Figure 11 provide per-layer scatter plots, while Table 15 reports the cosine-similarity decay vs. distance from the cutoff layer for the canonical Qwen2.5-7B / TriviaQA / TAQ-IS run. Complementary cross-model summary, propagation, and per-layer activation MSE figures are provided as Figure 12, Figure 13, and Figure 14.

Table 14. **Per-architecture aggregate statistics** of the inter-layer score-stability experiment (first 8 blocks quantized, downstream layers re-scored against FP16). Values are mean ± standard deviation across datasets and scoring-family runs where applicable. The stability-signal column corresponds to the variance/stability signal; the combined-score column corresponds to the full TAQ score.

Architecture	Stability-score ρ	Combined-score ρ	FP16 cosine sim
Qwen2.5-7B	0.990 ± 0.005	0.804 ± 0.156	0.931 ± 0.003
Gemma-2-9B	1.000 ± 0.000	0.658 ± 0.237	0.970 ± 0.006

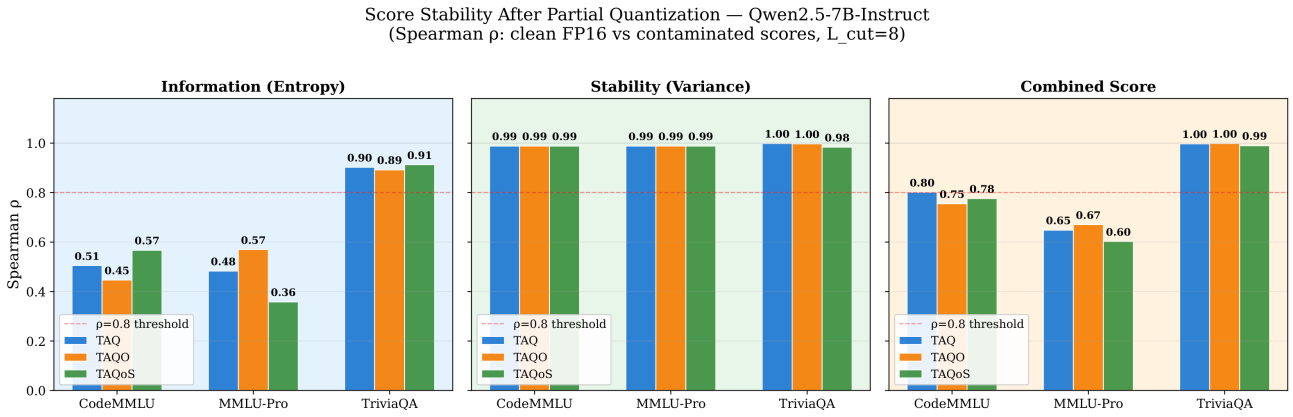

Figure 8. **Clean-vs-contaminated layer-score stability on Qwen2.5-7B.** Spearman correlations between clean FP16 layer-importance scores and scores recomputed after quantizing the first eight transformer blocks. The full combined score is highly stable on TriviaQA and moderately stable on CodeMMLU and MMLU-Pro, while the variance/stability signal is near-perfect across tasks.

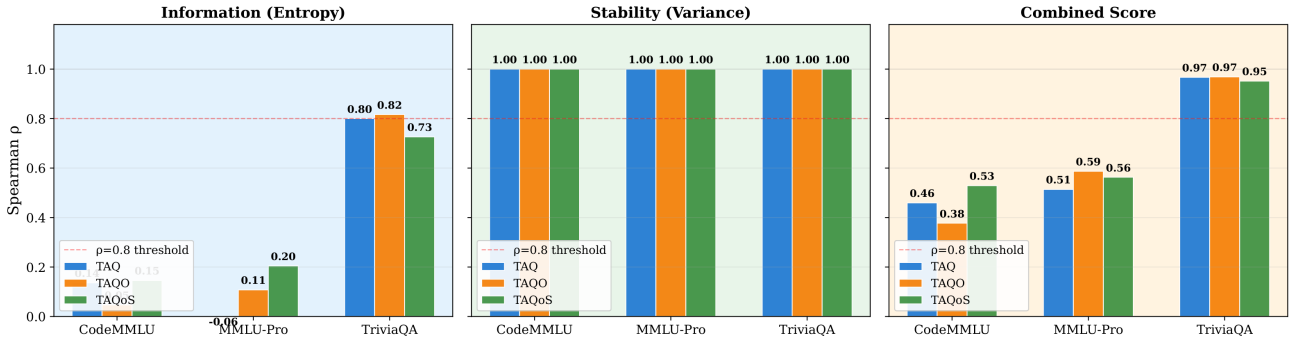

Figure 9. **Clean-vs-contaminated(quantized) first 8 layers-score stability on Gemma-2-9B.**

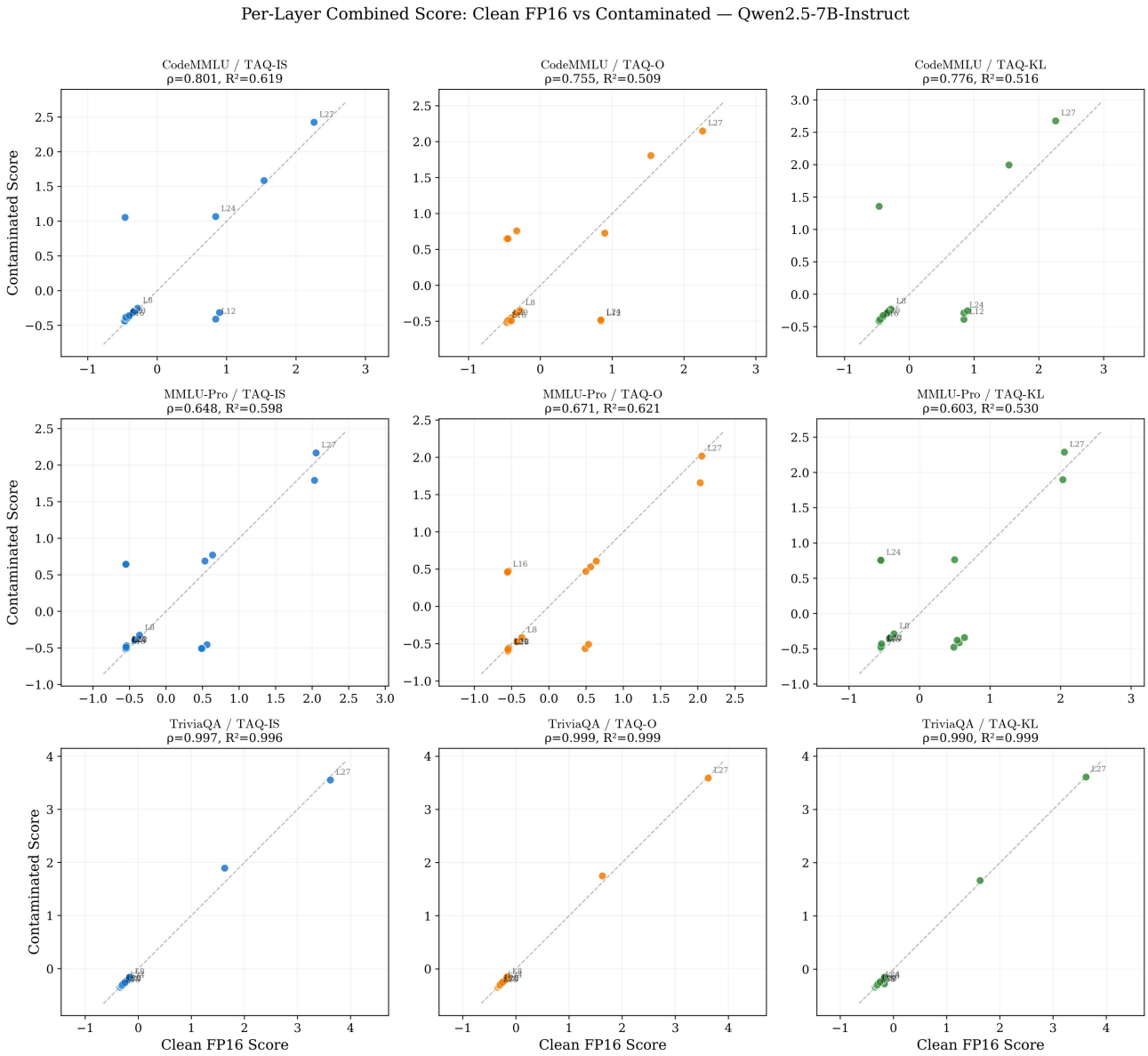

Figure 10. **Per-layer clean-vs-contaminated score agreement on Qwen2.5-7B.** Each panel compares clean FP16 layer-importance scores against scores obtained after upstream 4-bit quantization of the first eight transformer blocks. Alignment is strongest on TriviaQA, supporting clean FP16 scores as useful proxies rather than exact joint-quantization scores.

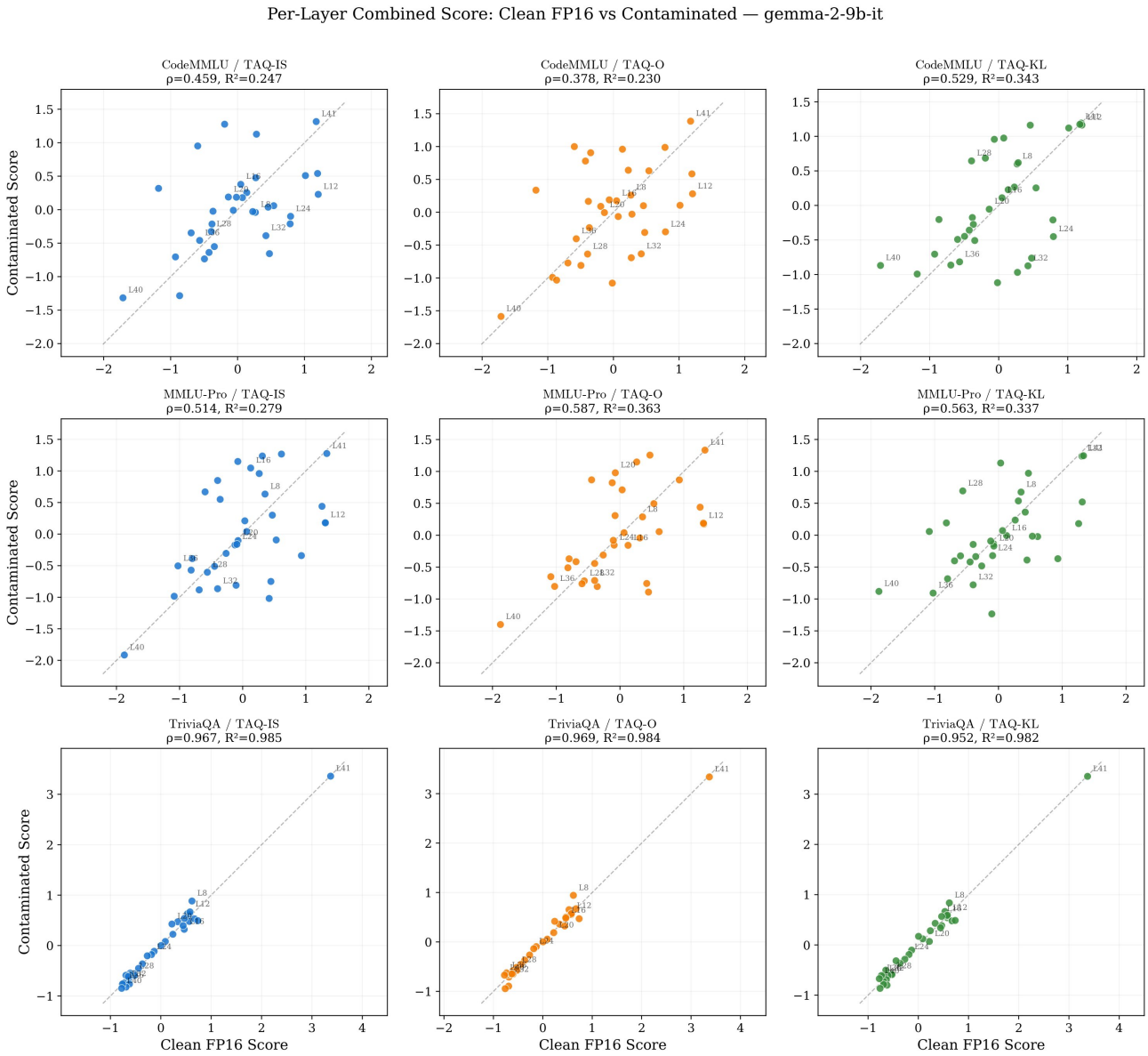

Figure 11. **Per-layer clean-vs-contaminated score agreement on Gemma-2-9B.** TriviaQA remains highly stable, while CodeMMLU and MMLU-Pro show larger deviations, suggesting stronger task-dependent inter-layer interactions.

Table 15. **Cosine-similarity decay vs. distance from cutoff** on Qwen2.5-7B / TriviaQA / TAQ-IS (first 8 blocks quantized). Cosine similarity drops sharply at distance $+1$ and then stabilizes in the 0.93–0.95 range, consistent with a stable residual-stream propagation plateau.

Distance	Cosine sim	Interpretation
0 (layer 8)	1.000	Cutoff layer (no error yet)
$+1$ (layer 9)	0.880	Sharp initial drop
$+7$ (layer 15)	0.932	Stabilization
$+19$ (layer 27)	0.933	Sustained plateau

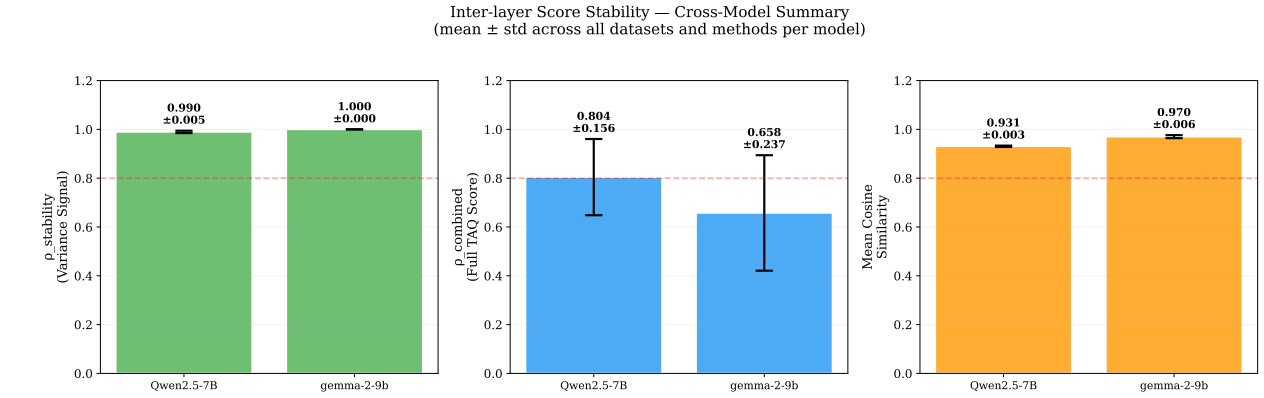

Figure 12. **Cross-model summary of the inter-layer contamination experiment.** Rank-preservation Spearman ρ, FP16 cosine similarity, and activation MSE aggregated over both architectures and the three TAQ families; the residual-stream propagation pattern is consistent across models.

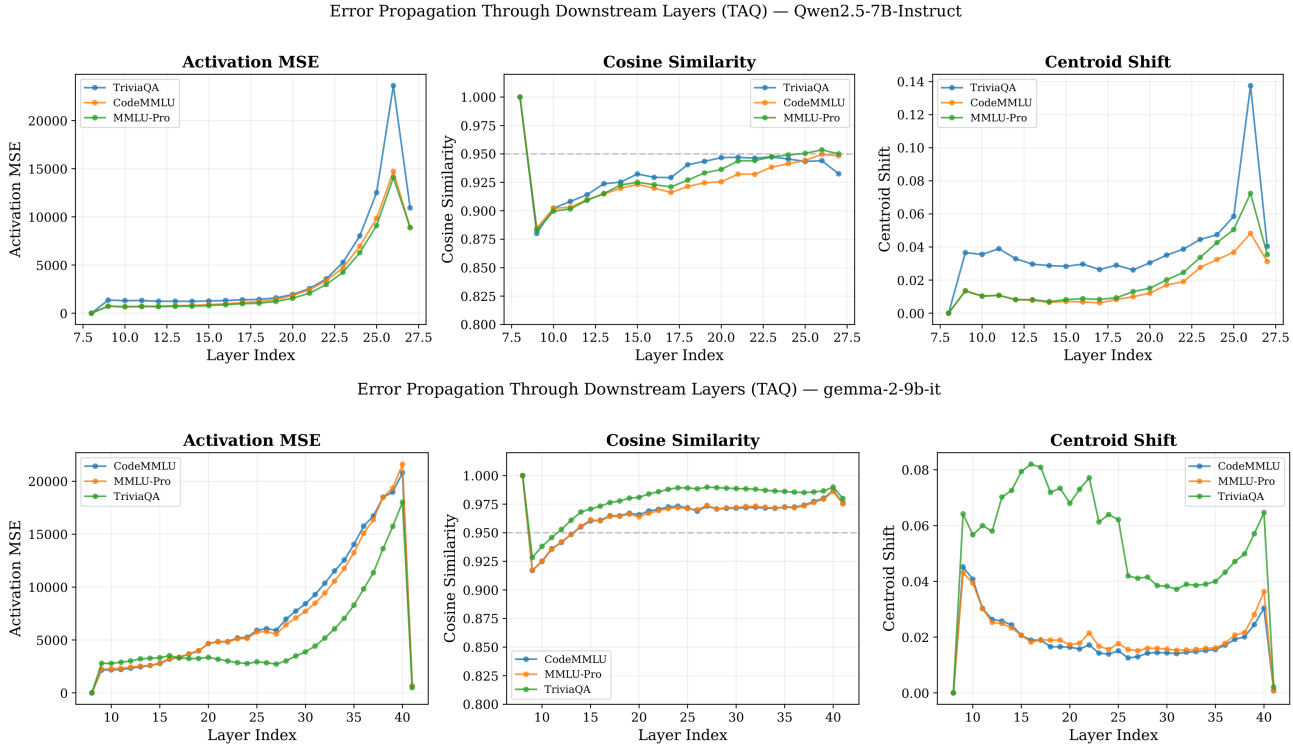

Figure 13. **Error-propagation curves.** Cosine similarity vs. distance from the cutoff, illustrating the residual-stream propagation pattern on Qwen2.5-7B and Gemma-2-9B for TAQ-IS.

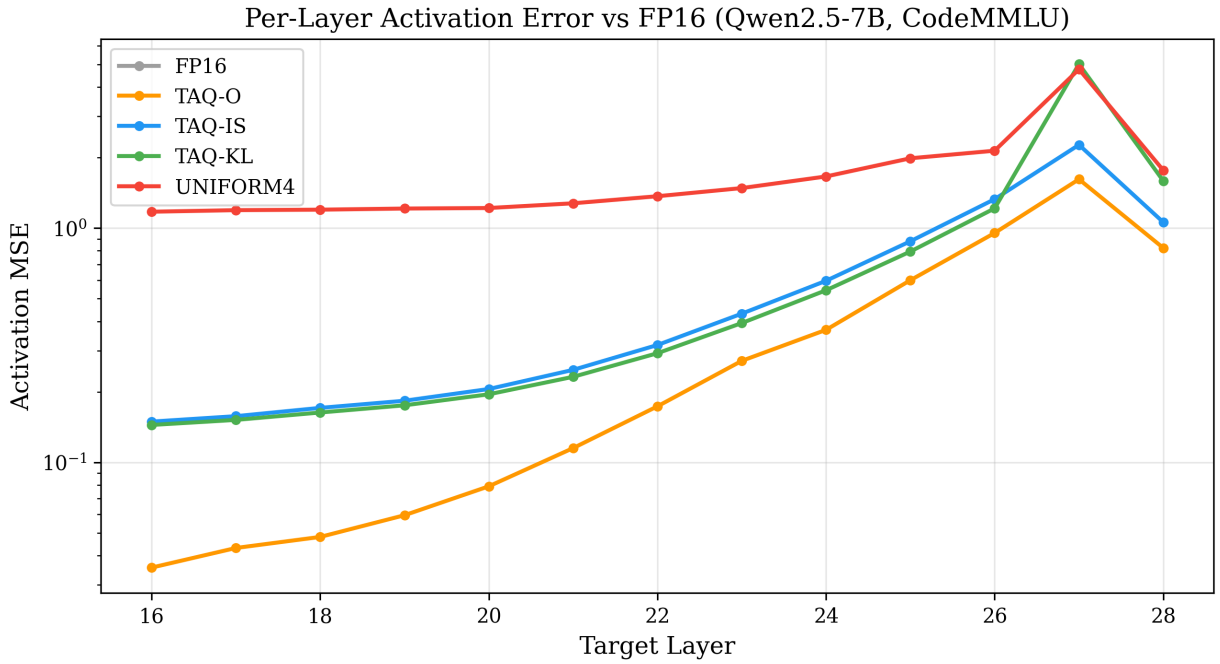

Figure 14. **Per-layer activation MSE** for the three TAQ variants and a uniform 4-bit reference; distortion concentrates near boundary blocks.