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

Anonymous Authors[1]

Abstract

Many applications of large language models (LLMs) require only a narrow capability, yet common post-training quantization (PTQ) pipelines assign precision largely without regard to the target task. As a result, they may spend bits on layers that are less relevant to the task. We propose per-task mixed-precision PTQ guided by hidden representations. Given a small set of unlabeled calibration prompts from the target task, we estimate layer importance and allocate higher precision to task-relevant layers while lower to the rest, under a bits allocation budget. We introduce three task-aware allocation signals: **TAQ**, which scores layers using an information-stability criterion derived from activation geometry; **TAQO**, which ranks layers by direct sensitivity to single-layer quantization; and **TAQ-KL**, which measures output sensitivity via KL divergence under a noise proxy for quantization error. Together, these methods provide a simple, post-training framework that connects mechanistic signals to quantization decisions, enabling task-aligned compression without additional training. A reference implementation is available at .

1. Introduction

Large language models (LLMs) built on the Transformer architecture (Vaswani et al., 2017) continue to scale in size and capability, with empirical scaling laws showing systematic improvements as parameter counts grow (Kaplan et al., 2020). Yet this scaling also amplifies inference- and memory-side costs (Pope et al., 2023), making serving efficiency a bottleneck for deployed systems.

Alongside the objective of artificial general intelligence, many applications require only a narrow subset of LLM capabilities, such as code completion.(Chen et al., 2021; Rozière et al., 2023), mathematical reasoning (Cobbe et al., 2021; Hendrycks et al., 2021), or domain-specific question answering. This motivates fine-tuning and distillation to tailor models to specific tasks (Hu et al., 2021; Ouyang et al., 2022), typically at the cost of additional training cycles and ongoing maintenance.

Quantization is a complementary compression technique that represents model weights and activations with fewer bits, reducing memory footprint and speeding up inference with little or no additional training (Gholami et al., 2021). For LLMs, prior work covers post-training weight quantization with high numerical fidelity (Sharify et al., 2024; Frantar et al., 2023), techniques for mitigating activation outliers or redistributing precision (Dettmers et al., 2022; Xiao et al., 2022), and mixed-precision schemes operating under global bit-width constraints (Dong et al., 2019; Lin et al., 2024).

Mechanistic interpretability aims to uncover the mechanisms underlying model behavior through a three-stage loop: decomposition, function assignment, and validation via behavioral prediction and intervention (Sharkey et al., 2025). Applied to LLMs, this perspective suggests that tasks may be implemented by task-conditioned subsets of internal computation, although polysemanticity and superposition pose challenges for reliable decomposition and attribution (Scherlis et al., 2022). Feature-dictionary approaches and causal localization methods provide practical tools to identify task-relevant representations and to test their localization (Cunningham et al., 2023; Heimersheim & Nanda, 2024).

In contrast, LLM quantization methods are largely task-agnostic: common pipelines optimize layer-local proxies such as weight reconstruction error, activation-range calibration, or curvature-based criteria, rather than directly optimizing downstream task objectives (Nagel et al., 2020; Banner et al., 2019; Wang et al., 2020; Lin et al., 2024). As a result, precision is typically assigned using uniform bit-widths or heuristic mixed-precision rules that do not account for task-conditioned layer importance, despite substantial evidence that layers contribute heterogeneously to end-task behavior (Tenney et al., 2019; Meng et al., 2022a; Geva et al., 2021).

In this work, we explore per-task quantization, and inves-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

tigate whether task-salient signals can guide post-training mixed-precision weight quantization. Unlike prevailing PTQ pipelines, which are largely task-agnostic and typically optimize local reconstruction surrogates on multi-task calibration data, we introduce Task-Aware Quantization (**TAQ**) and propose three instantiations: **TAQ**, **TAQO**, and **TAQ-KL**. All three methods convert task-calibrated model behavior into per-layer precision decisions. TAQ ranks layers using a task-conditioned score that combines (i) a matrix-entropy measure of representational information and (ii) an activation-variance stability signal computed over calibration prompts, assigning higher precision to the most relevant layers while quantizing the remainder more aggressively. TAQO replaces surrogate scoring with an oracle sensitivity test: it quantizes one layer at a time and measures the resulting task performance drop, retaining FP16 only for the most sensitive layers (in addition to embedding and output layers). Finally, TAQ-KL estimates layer importance via output sensitivity by injecting layer-wise noise as a proxy for quantization error and scoring layers according to the induced KL divergence between baseline and perturbed output distributions, allocating higher precision to layers with larger KL. We make three contributions:

- **Task-aware mixed-precision PTQ.** We propose *Task-Aware Quantization* (TAQ), a post-training, weight-only quantization framework that uses a small set of task calibration prompts to derive *per-layer* precision allocations under a memory/compute budget, rather than optimizing task-agnostic reconstruction surrogates.
- **Three layer-importance signals.** We instantiate TAQ with three complementary layer-importance scoring rules: TAQ, TAQO, and TAQ-KL. These rules use task-specific hidden representations and a task-oriented calibration set to improve the sensitivity and localization of task features across layers within the quantization process.
- **Empirical evaluation across tasks and budgets.** On code, math, and trivia/knowledge benchmarks with open-weight LLMs, task-aware allocations consistently preserve accuracy under substantial compression and, in some cases, exceed the original model.

2. Background and Related Work

Task features and mechanistic interpretability. Mechanistic interpretability in LLMs aims to identify and causally validate internal components that implement specific behaviors (Wang et al., 2022; Sharkey et al., 2025), and has shown that for at least some behaviors, task-relevant computation is concentrated in a subset of layers and components. At the same time, features may be distributed or superposed, and it remains unclear whether task signals are best characterized as linearly decodable directions or as nonlinear, distributed representations (Sharkey et al., 2025). We do not assume that task information is localized; rather, we use component-level localization as an operational proxy for sensitivity to internal noise, which we seek to minimize under quantization. Accordingly, we treat hidden representations as an empirical signal of where task information flows through the network, and leverage this signal to guide per-task precision allocation.

LLM quantization and task-aware compression. PTQ is widely used to reduce memory footprint and inference latency. Modern methods primarily focus on mitigating quantization error induced by activation outliers and channel-wise sensitivity. For example, SmoothQuant reduces activation outliers through an equivalent rescaling that shifts quantization difficulty from activations to weights (Xiao et al., 2022), while Activation aware Weight Quantization (AWQ) targets low-bit weight-only quantization by identifying activation-salient channels and protecting a small subset of weights (Lin et al., 2024). However, these methods are task-agnostic, aiming to preserve the broad capabilities of the base model rather than optimize performance for specific downstream tasks. In contrast, we use task-conditioned hidden representations from a small calibration set to estimate layer importance and allocate precision accordingly, yielding stable task-specific sensitivity profiles and improved per-task PTQ performance.

2.1. Background

Quantization is the process of mapping floating-point tensors to lower-precision representations, thereby reducing memory footprint and improving throughput. A standard (asymmetric) uniform PTQ scheme applies

$$Q(x) \;=\; \mathrm{round}\left(\frac{x-z}{s}\right)\cdot s \;+\; z,$$

where s is a scale and z a zero-point offset (Nagel et al., 2021; Jacob et al., 2018; Krishnamoorthi, 2018). Beyond uniform bit-width quantization, recent progress in LLM compression can be broadly categorized along three axes: (i) *weight-only post-training quantization and rounding*, which minimize reconstruction error under fixed precision constraints (Frantar et al., 2023; Xiao et al., 2022; Shao et al., 2024); (ii) *sensitivity-based bit allocation* under a global precision budget, where layers or channels are assigned different bit-widths based on local importance estimates derived from curvature or second-order information (Dong et al., 2019); and (iii) *activation- and channel-aware calibration*, which leverages activation statistics to guide quantization decisions and has been increasingly extended to weight- and KV-cache quantization for long-context in-

ference (Lin et al., 2024). Activation-aware methods such as AWQ minimize an activation-weighted reconstruction objective of the form

$$\mathcal{L} = \|W - Q(W)\|_F^2 \odot S_{\text{act}},$$

implicitly treating activation magnitude as a universal proxy for parameter importance.

Per-layer activation analysis, representations, and task signals. Work on activation geometry shows that linear directions in hidden states encode semantic features and enable inference-time control, while localized edits can expose and manipulate factual computation (Turner et al., 2023; Zou et al., 2023; Meng et al., 2022b; Shnaidman et al., 2025). For a task dataset $D = \{q_j\}_{j=1}^n$ and layer-ℓ activations $x^{(\ell)}(p) \in \mathbb{R}^d$, a task-conditioned direction can be estimated, for example, by

$$d^{(\ell)} = \frac{1}{n} \sum_{q_j \in D} \Big(x^{(\ell)}(q_j) - x^{(\ell)}(\tilde{q}_j) \Big),$$

where $\tilde{q}_j$ are contrast prompts, and cross-task alignment can be measured via cosine similarity

$$s_i^{(k,\ell)} = \frac{\langle d_i^{(k)}, d_i^{(\ell)} \rangle}{\|d_i^{(k)}\| \, \|d_i^{(\ell)}\|}.$$

Per-layer activation analyses relate norms, sparsity, and directionality to downstream performance, motivating precision budgets that track where task-salient signal resides (Turner et al., 2023; Meng et al., 2022b; Arditi et al., 2024; Levi et al., 2025). To diagnose how information is distributed across layers, one can study the spectrum of the token–token Gram matrix: for hidden states $X_\ell \in \mathbb{R}^{n \times d}$, define $G_\ell = X_\ell X_\ell^\top$ with eigenvalues $\{\lambda_i\}_{i=1}^n$ and normalized $\tilde{\lambda}_i = \lambda_i / \sum_{j=1}^n \lambda_j$, then the matrix entropy

$$H(G_\ell) = -\sum_{i=1}^{n} \tilde{\lambda}_i \log \tilde{\lambda}_i \tag{1}$$

quantifies dispersion, with larger values indicating more diverse (isotropic) representations and smaller values indicating compression; empirically, intermediate layers often yield stronger task-useful features and exhibit a mid-depth peak (Skean et al., 2025). Finally, for benchmark tasks $\mathcal{D} = \{D_1, \ldots, D_M\}$ and a task-specific calibration set $C_k \subseteq D_k$, we compute per-layer relevance scores $r_k^{(\ell)}$ (e.g., direction norms or projection energies) and allocate per-layer bit widths $\mathbf{b} = (b_1, \ldots, b_N) \in \mathcal{B}$ by solving

$$\min_{\mathbf{b} \in \mathcal{B}} \mathcal{E}_k(\mathbf{b}) \quad \text{s.t.} \quad \text{Cost}(\mathbf{b}) \le \tau,$$

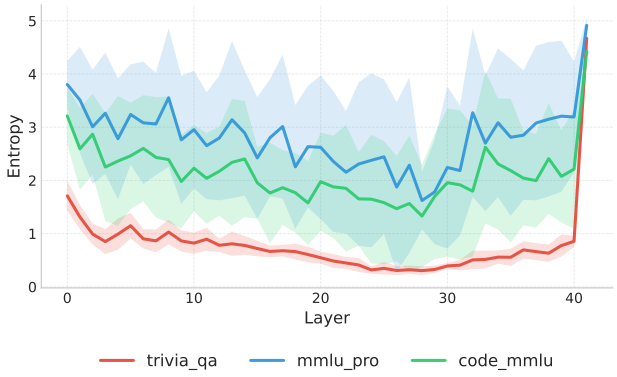

Figure 1. Information entropy across layers with error bars on three tasks, on *Gemma2-9B*.

where $\mathcal{E}_k$ is a task-conditioned error proxy and τ enforces a memory/latency budget, thereby assigning higher precision to layers with larger $r_k^{(\ell)}$.

We address the task-agnostic gap in PTQ by linking methods that optimize quantization objectives and allocate per-layer bit precision with advances in mechanistic interpretability and semantic analyses of hidden representations. Concretely, we instantiate PTQ pipelines with task-informed surrogates and per-layer allocation rules derived from representation diagnostics, yielding a high-performance, task-oriented quantization approach.

3. Per-Task Quantization

In this section, we first analyze how different downstream tasks exhibit distinct layer-wise sensitivities in LLMs. We then study how post-training quantization affects these layers unevenly, and leverage these observations to define task-aware quantization techniques: TAQ, TAQO, and TAQ-KL.

3.1. Task Representation Shifts During Quantization

Per layer task relevance scoring. Prior work (Skean et al., 2025) has shown that different tasks rely on different layers of an LLM during inference. This dependence is reflected in how hidden activations propagate through the model in response to a given task. To quantify this behavior, we measure the information entropy of layer-wise activations (Equation (1)) across three distinct tasks: trivia (TriviaQA (Joshi et al., 2017)), code (CodeMMLU (Nguyen et al., 2025)), and language understanding (MMLU-Pro (Wang et al., 2024)). As shown in Figure 1, each task exhibits a distinct entropy profile across layers. We report the mean entropy at each layer, with error bars indicating variability across disjoint evaluation subsets, revealing both systematic task-specific structure and layer-wise uncertainty. These results indicate that different layers contribute un-

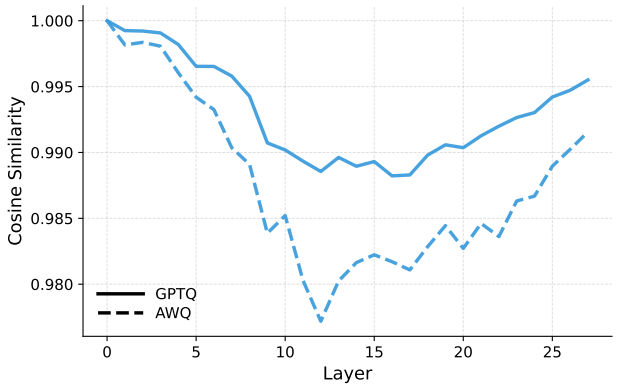

Figure 2. Task similarity across layers of quantized models relative to original model, on *Qwen2.5-7B* and MMLU-Pro.

evenly depending on the task.

Downstream tasks. These relevance scores quantifying information allow us to identify which layers are most heavily utilized by a given downstream task. Layers exhibiting high task-specific activation entropy are repeatedly engaged to transform and preserve task-relevant information, making them critical to task performance. Consequently, these layers are more sensitive to perturbations introduced by quantization.

Task activation aware quantization. In PTQ, layers that are heavily and consistently exercised by a task must retain sufficient representational capacity after quantization. We analyze widely used quantization methods, including *AWQ* (Lin et al., 2024) and *GPTQ* (Frantar et al., 2023), by measuring their representational similarity to the full-precision model on task activations. As shown in Figure 2, we report mean activation similarity to the baseline across layers on a coding task. The results show that schemes diverge from the original model to different extents across layers, indicating non-uniform sensitivity to precision reduction. This suggests indiscriminate quantization is suboptimal: excessive precision loss in task-critical layers can disproportionately harm performance, while less critical layers can tolerate more aggressive quantization. Together, these observations motivate task-aware, layer-sensitive strategies that preserve precision where it matters most.

3.2. Layer-Wise Task-Aware Quantization Framework

We study per-task, weight-only quantization for transformer-based language models. Given a small calibration set drawn from a downstream task, our goal is to allocate per-layer precision so that task performance is preserved while meeting a memory/compute budget.

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

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

where B is a budget. In practice, we approximate Eq. (2) by estimating a task-conditioned importance score s_ℓ from a small calibration set $\mathcal{D}_{\text{calib}}$, and allocating higher precision to layers with larger s_ℓ.

Quantization operator: weight-only, group-wise affine. For each linear weight matrix, we apply per-group affine quantization (group size G) with bitwidth $b \in \{4, 8\}$. For a weight group vector $w \in \mathbb{R}^G$, define $Q_b = 2^b - 1$ and

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

where Δ and z are group-wise scale and zero-point. Layers assigned $b_\ell = 16$ remain in FP16. All methods below share the same quantization operator in Eq. (3); they differ only in the computation of layer importance scores s_ℓ and in the resulting per-layer precision allocation $\mathbf{b}$.

3.3. TAQ: Task-Activation-aware Quantization

TAQ 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 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:

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

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

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

so larger Stab_ℓ corresponds to more stable (lower-variance) activations.

Combined score and allocation. We z-normalize both signals across layers, $z(\cdot)$, and combine them:

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

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:

$$b_\ell = \begin{cases} 8, & \ell \in \mathrm{TopK}(s; K\%), \\ 4, & \text{otherwise}, \end{cases} \tag{8}$$

Here, $\mathrm{TopK}(s; K\%)$ denotes the indices of the top $K\%$ layers ranked by s_ℓ. This yields a task-conditioned mixed-precision model without requiring task labels beyond the calibration prompts.

3.4. TAQO: Task-Aware Quantization via Oracle Layer Sensitivity

TAQO directly estimates *task impact* by measuring how much task performance drops when only one layer is quantized, keeping all others in FP16. This provides an oracle sensitivity signal that is directly aligned with the downstream metric.

Per-layer sensitivity. Let $\mathcal{M}(\cdot)$ be a task metric (e.g., averaged EM/F1) computed on a small held-out subset from $\mathcal{D}_{\mathrm{calib}}$, and let $\mathcal{S}_{\mathrm{base}} = \mathcal{M}(f_\theta)$ 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:

$$f^{(\ell)} = f_{\theta(\, b_\ell=4,\, b_{j\neq\ell}=16\,)}. \tag{9}$$

The oracle drop is

$$\Delta_\ell = \max\Big(0,\ \mathcal{S}_{\mathrm{base}} - \mathcal{M}\Big(f^{(\ell)}\Big)\Big)\,. \tag{10}$$

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:

$$b_\ell = \begin{cases} 16, & \ell \in \mathcal{E} \ \cup\ \mathrm{TopK}(\Delta; k), \\ 4, & \text{otherwise}. \end{cases} \tag{11}$$

This produces a sparse FP16 allocation concentrated on task-critical layers.

3.5. 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. Unlike TAQO, this does not require repeated metric evaluation; unlike TAQ, it focuses on task-conditioned output sensitivity.

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:

$$H'_\ell(x) = H_\ell(x) + \eta, \qquad \eta \sim \mathcal{U}\left(-\frac{\delta_\ell}{2}, \frac{\delta_\ell}{2}\right), \tag{12}$$

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

$$p(x) = \mathrm{softmax}\left(\frac{z(x)}{T}\right), \tag{13}$$

$$q_\ell(x) = \mathrm{softmax}\left(\frac{z'_\ell(x)}{T}\right). \tag{14}$$

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

$$s_\ell = \mathbb{E}_{x\sim\mathcal{D}_{\mathrm{calib}}}\left[\mathrm{KL}(p(x)\,\|\,q_\ell(x))\right]. \tag{15}$$

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.

Allocation. As in TAQ, we allocate 8-bit to the top $K\%$ layers by s_ℓ and 4-bit to the rest:

$$b_\ell = \begin{cases} 8, & \ell \in \mathrm{TopK}(s; K\%), \\ 4, & \text{otherwise}. \end{cases} \tag{16}$$

Summary. All three methods instantiate Eq. (2) by producing a task-conditioned mixed-precision vector $\mathbf{b}$. TAQ leverages representation information and activation stability, TAQO relies on oracle task sensitivity, and TAQ-KL measures output-distribution sensitivity using a noise-based proxy for quantization error.

4. Experiments

We empirically investigate the efficacy of *per-task* PTQ for LLMs. Our analysis is structured around three primary research questions:

- **RQ1 (Efficiency):** Does identifying and protecting task-relevant layers yield a superior accuracy-memory trade-off compared to task-agnostic mixed-precision allocation?
- **RQ2 (Proxy Effectiveness):** How closely can label-free signals, specifically representation- or output-based sensitivities, approximate oracle performance in guiding precision allocation?
- **RQ3 (Robustness):** Can task-aware allocation policies compensate for imperfect or mixed-task calibration data distributions?

4.1. Experimental Setting

Tasks and metrics. We evaluate performance across three diverse domains spanning knowledge retrieval, code generation, and mathematical reasoning: (i) **TriviaQA** (Knowledge Retrieval) (Joshi et al., 2017), reporting Exact Match (EM) and token-level F1; (ii) **CodeMMLU** (Code Generation) (Nguyen et al., 2025), reporting EM; and (iii) **MMLU-Pro** (Math/Reasoning) (Wang et al., 2024), reporting EM.

Models. Our primary results utilize five instruction-tuned open-weight LLMs from recent model families: Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Team, 2024), Qwen2-7B-Instruct (Team et al., 2024b), Gemma-2-9B-it (Team et al., 2024a),

Baselines. We benchmark against representative PTQ methods: **W\O (FP16)** (full precision), **GPTQ** (Frantar et al., 2023), and **AWQ** (Lin et al., 2024). To ensure fair comparison, all methods are evaluated using identical prompts and metric definitions per task, we also includes variants of this methods as ablation study and extend the analysis to separate the influence components of per task quantization.

Quantization protocol. We focus on **weight-only** quantization using group-wise affine quantization (Eq. (2)) 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 §3.2.

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, TAQ-KL), derive oracle sensitivities (TAQO), and fit quantizer parameters; no gradient-based updates are applied. For TriviaQA, we utilize indices 0–511 of the validation split for calibration. For MMLU-Pro and CodeMMLU, we employ stratified sampling from the official test splits to preserve subject distributions. All sampling uses a fixed seed of 42 to ensure zero overlap between splits.

Memory reporting conventions. We report **W** as the *estimated weight-only footprint* (GB). For a model with N parameters, the FP16 baseline is $W_{\text{FP16}} \approx 2N$ bytes. For mixed-precision policies with bitwidths $b_\ell \in \{4, 8, 16\}$, we calculate:

$$W \approx \sum_\ell \frac{b_\ell}{8} N_\ell + W_{\text{ovh}}, \tag{17}$$

where N_ℓ is the parameter count of layer ℓ and W_{ovh} accounts for quantization metadata (scales/zero-points).

TAQ implementation details. **TAQ** ranks layers via $R_\ell = \alpha\, \text{z}(H_\ell) + \beta\, \text{z}(S_\ell)$ (with α=β=0.5), utilizing a reservoir of r=256 token vectors to compute matrix-entropy H_ℓ and activation stability S_ℓ. It assigns 8-bit precision to the top K=25% of layers and 4-bit to the remainder. **TAQO** 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_{sens}=16 test prompts. **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 using up to 64 calibration prompts.

4.2. Experimental Results

Table 1 details the performance of our proposed methods (TAQ, TAQ-KL, TAQO) against standard task-agnostic baselines across three distinct domains. Exact bit allocations are provided in Section A.2. The results provide strong empirical evidence for our core hypotheses regarding allocation efficiency (**RQ1**) and the validity of label-free sensitivity proxies (**RQ2**).

Our primary hypothesis posits that protecting task-relevant layers enables superior performance under tighter memory constraints than task-agnostic approaches. For example, on the CodeMMLU benchmark using Qwen2.5-3B, TAQ-KL achieves an EM of 46.83%, outperforming GPTQ (24.66%)

Table 1. **Main results. W** is the per-model quantized weight size (GB); **Bold** indicates the best and underlined indicates the second-best quantized results.

Method	Math Reasoning: MMLU-Pro (EM↑)								Code Generation: 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	2.5	23.54	5.19	29.15	5.19	41.60	5.74	20.61	2.5	40.28	5.19	40.09	5.19	50.68	5.74
TAQ-Oracle (Ours)	**33.01**	3.00	32.13	8.00	38.57	8.00	39.40	8.36	**47.41**	3.00	**49.51**	8.00	**49.22**	8.00	50.54	8.36
TAQ-KL (Ours)	28.03	1.88	**35.35**	4.38	**39.11**	4.38	**42.48**	5.68	46.83	1.88	47.02	4.38	47.12	4.38	50.83	5.68
TAQ (Ours)	30.76	1.88	**33.35**	4.38	37.99	4.38	41.60	5.68	46.29	1.88	47.22	4.38	48.49	4.38	51.03	5.68

Method	Qwen2.5-3B			Qwen2.5-7B			Qwen2-7B			Gemma-2-9B		
	EM↑	F1↑	W↓	EM↑	F1↑	W↓	EM↑	F1↑	W↓	EM↑	F1↑	W↓
Knowledge Retrieval: TriviaQA												
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	2.5	06.98	18.15	5.19	**14.79**	**25.16**	5.19	51.51	60.14	5.74
TAQ-Oracle (Ours)	08.45	**20.34**	3.00	12.26	23.23	8.00	01.86	20.58	8.00	59.33	67.67	8.36
TAQ-KL (Ours)	01.32	13.69	1.88	14.06	23.83	4.38	02.25	21.17	4.38	**61.04**	**68.58**	5.68
TAQ (Ours)	03.91	17.14	1.88	12.55	23.95	4.38	02.25	20.79	4.38	59.67	68.08	5.68

and AWQ (20.61%) by over 20 percentage points. Crucially, this gain is achieved while utilizing *less* memory (W=1.88 GB) than the baselines (1.93 GB and 2.50 GB, respectively). Similarly, on the complex reasoning task MMLU-Pro with Qwen2.5-7B, TAQ-KL reaches 35.35% EM, surpassing the strongest baseline (AWQ: 23.54%) by nearly 12 points while reducing the memory footprint from 5.19 GB to 4.38 GB. These disparities highlight that uniform or weight-magnitude-based allocation strategies (GPTQ/AWQ) catastrophically degrade performance on specialized tasks at low bit-widths, whereas our task-aware policy effectively identifies and preserves the specific circuits required for code generation and reasoning.

We investigate whether label-free signals (representation entropy for TAQ and output KL-divergence for TAQ-KL) can approximate the sensitivity rankings derived from labeled data (TAQ-Oracle). The data indicates that these proxies are highly effective. On CodeMMLU (Qwen2.5-3B), the label-free TAQ-KL (46.83%) virtually matches the TAQ-Oracle (47.41%), despite the Oracle utilizing a significantly larger memory budget (3.00 GB vs 1.88 GB).

In some instances, label-free methods even exceed the Oracle baseline due to better generalization or regularization effects; for instance, on TriviaQA with Gemma-2-9B, TAQ-KL achieves 61.04% EM compared to the Oracle's 59.33%, and on MMLU-Pro (Qwen2.5-7B), TAQ-KL outperforms the Oracle (35.35% vs 32.13%) while using roughly half the memory (4.38 GB vs 8.00 GB). This confirms that intrinsic model signals, specifically the stability of the output distribution under perturbation, serve as robust, sufficient statistics for layer sensitivity, eliminating the need for labeled validation sets during the quantization process.Overall, Figure 3 shows the performance of each quantized model relative to its size on CodeMMLU, with our quantized models consistently outperforming the baselines. This continues on the other tasks, with results in Appendix A.1.

4.3. Ablation Study

While our results show per-task quantization is effective, it is unclear whether the gains come from the *task-aware policy* (identifying sensitive layers) or simply from *task-specific calibration data*. Standard PTQ methods like AWQ use a generic mixed-domain calibration set to reduce overfitting, but perfect alignment to the target task might remove degradation without a complex allocation strategy. To disentangle these effects, we run a controlled ablation on TriviaQA across the auxiliary model suite and pose two questions:

- **AQ1 (Alignment Hypothesis):** Is standard PTQ degradation primarily due to distribution shift? Specifically, if we perfectly align calibration data with the test task (on-task calibration), can a task-agnostic policy (AWQ) recover full-precision performance?
- **AQ2 (Policy Resilience):** Does a task-aware allocation

Figure 3. Model weights (GB) versus model performance (EM measure) across models, on CodeMMLU.

Table 2. **Component Analysis (Calibration Data vs. Policy).** Comparison of performance when varying the calibration source (Mixed vs. On-Task) and the allocation policy (Uniform vs. Task-Aware).

Method	Phi-4		Qwen3		Llama-3.1		Qwen2.5		Mistral	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
W/O (FP16)	43.16	51.24	11.23	26.33	57.86	66.00	11.43	21.90	18.90	35.18
AWQ (Mixed)	2.25	7.07	9.23	23.94	53.37	62.23	11.91	24.06	17.48	33.75
AWQ-OT (On-Task)	0.73	4.54	9.81	24.33	55.66	62.92	10.59	24.21	17.58	26.33
TAQ (Ours)	**42.33**	**50.81**	11.82	26.12	57.03	**65.35**	12.06	22.47	**18.75**	**34.74**
TAQO (Oracle)	0.73	4.57	**18.51**	**34.34**	**57.91**	65.14	**12.65**	**25.49**	18.51	34.34

policy (TAQ) provide robustness that data alignment alone cannot, especially in regimes where standard minimization is unstable?

Experimental configuration. We compare three distinct setups to isolate these variables of calibration sets: (i) AWQ (Mixed): balanced pool of math, code, and trivia (N=2048), representing the standard "generalist" calibration approach. (ii) AWQ-OT (On-Task): 512 examples drawn exclusively from the target task. This tests AQ1 by providing perfect data alignment. (iii) TAQ (Ours): Our original policy. We run this ablation on an auxiliary model suite because GPTQ checkpoints were unavailable for the main-experiment models; this enables consistent AWQ/OT baselines under the same PTQ codepath.

Experimental Results. Overall, the ablation refutes **AQ1** and supports **AQ2**. Under **AQ1** (alignment hypothesis), AWQ-OT should improve over AWQ (Mixed), but behavior is unstable: Llama-3.1 improves (53.37 $\rightarrow$ 55.66 EM) whereas Phi-4 collapses (2.25 $\rightarrow$ 0.73 EM) and Mistral changes minimally, suggesting a signal-to-noise hazard where AWQ overfits quantization parameters on a narrow on-task set. Under **AQ2** (policy resilience), TAQ consistently stabilizes performance by protecting critical layers, nearly matching FP16 on Phi-4 (42.33 vs. 43.16 EM) and improving over AWQ-OT on Qwen3 (11.82 vs. 9.81 EM).

5. Discussion

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 models and tasks, this yields strong accuracy, memory trade-offs, with especially pronounced gains on specialized workloads where standard baselines can collapse at low bit-widths; notably, label-free variants (TAQ-KL/TAQ) closely match, and occasionally exceed, the labeled Oracle, suggesting that intrinsic stability signals can serve as practical, task-aware sensitivity statistics. A remaining limitation is that TAQ still needs a small set of representative prompts to characterize the target task distribution; how to define such prompts robustly (e.g., under drift or fuzzy task boundaries) is an important deployment concern but not central to the core allocation mechanism studied here. Future work includes reducing this dependence toward fully data-free PTQ, enabling online adaptation or interpolation across tasks, and designing lightweight switching for multi-tenant serving. Overall, our results point to a novel perspective: *task information is depth-localized*, and preserving the right circuits, not simply increasing bit-width everywhere, is the key to making highly compressed LLMs reliable for task-centric applications.

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.

References

Arditi, A., Obeso, A., and Panickssery, N. Refusal in language models is mediated by a single direction. *arXiv preprint arXiv:2406.11717*, 2024. URL https://arxiv.org/abs/2406.11717.

Banner, R., Nahshan, Y., and Soudry, D. Post training 4-bit quantization of convolutional networks for rapid-deployment. In *Advances in Neural Information Processing Systems*, 2019.

Chen, M. et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., and Nakano, R. Training verifiers to solve math word problems. In *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

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., Shleifer, S., and Zettlemoyer, L. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022. URL https://arxiv.org/abs/2208.07339.

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 (ICCV)*, pp. 293–302, 2019.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.17323.

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. *arXiv preprint arXiv:2103.13630*, 2021. URL https://arxiv.org/abs/2103.13630.

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

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.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2704–2713, 2018.

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*, 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.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Levi, A., Zilka, A., Shimoni, Y., and Shenaidman, Y. Jailbreak attack initializations as extractors of compliance directions. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025. URL https://aclanthology.org/2025.findings-emnlp.179/.

Lin, J., Tang, J., Wang, H., Yang, S., and Han, S. Awq: Activation-aware weight quantization for llm compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024. URL https://arxiv.org/abs/2306.00978.

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, 2022a.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, 2022b. URL https://arxiv.org/abs/2202.05262.

Nagel, M., van Baalen, M., Blankevoort, T., and Welling, M. Up or down? adaptive rounding for post-training quantization. *arXiv preprint arXiv:2004.10568*, 2020. URL https://arxiv.org/abs/2004.10568.

Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., and Blankevoort, T. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. URL https://arxiv.org/abs/2106.08295.

Nguyen, D. M., Phan, T. C., Hai, N. L., Doan, T.-T., Nguyen, N. V., Pham, Q., and Bui, N. D. Q. CodeMMLU: A multi-task benchmark for assessing code understanding & reasoning capabilities of codeLLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CahIEKCu5Q.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

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.

Rozière, B. et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023. URL https://arxiv.org/abs/2308.12950.

Scherlis, A., Sachan, K., Jermyn, A. S., Benton, J., and Shlegeris, B. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

Shao, W., Huang, W., Zhang, Y., and Luo, P. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.13137.

Sharify, S. A., Damsgaard, D., Smit, A., et al. Post-training mixed-precision quantization for large language models. *arXiv preprint arXiv:2403.03280*, 2024. URL https://arxiv.org/abs/2403.03280.

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, Y., Yao, Y., Kim, G., et al. Activation steering for masked diffusion language models. *arXiv preprint arXiv:2512.24143*, 2025. URL https://arxiv.org/abs/2512.24143.

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*, 2024a.

Team, Q. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Team, Q. et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024b.

Tenney, I., Das, D., and Pavlick, E. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 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. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Wang, H. et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024. URL https://arxiv.org/abs/2406.01574.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Wang, Z., Qin, Z., Zhang, B., Ding, W., and Ding, J. Towards efficient post-training quantization of pre-trained language models. *arXiv preprint arXiv:2009.12772*, 2020. URL https://arxiv.org/abs/2009.12772.

Xiao, G., Lin, J., Seznec, M., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*, 2022. URL https://arxiv.org/abs/2211.10438.

Zou, A., Phan, L., Chen, S., Campbell, B., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., and Dombrowski, A.-K. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023. URL https://arxiv.org/abs/2310.01405.

A. Experiments

A.1. Weight V.S. Performance

Figure 4. Model weights (GB) versus model performance (EM measure) across models, on MMLU-Pro.

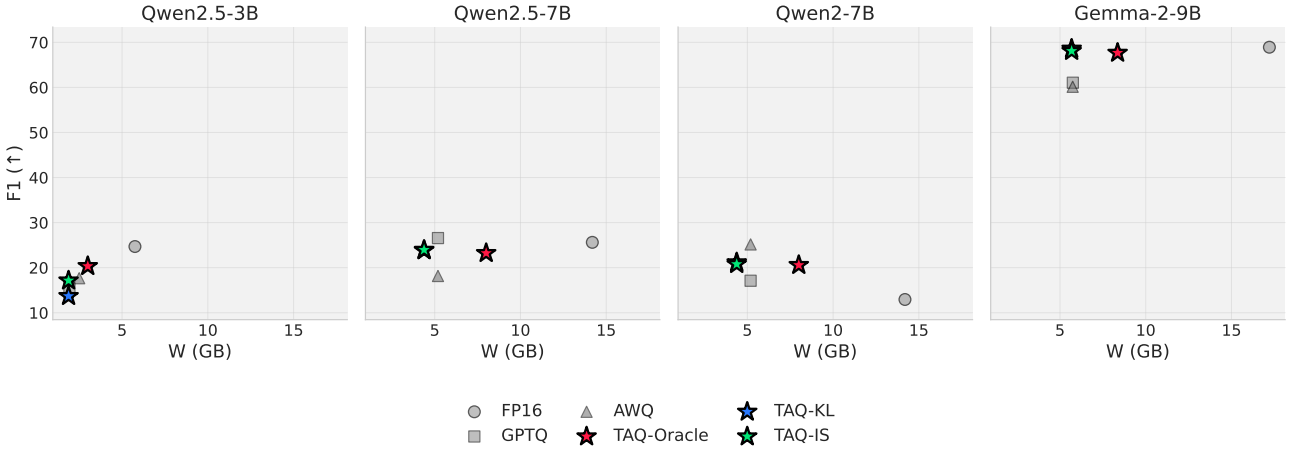

Figure 5. Model weights (GB) versus model performance (EM measure) across models, on TriviaQA.

A.2. Experiments Bit Allocations

This subsection presents the layerwise mixed-precision allocations produced by different heuristics across models and datasets. For each experiment, transformer layers are ranked by a given scoring method, and the top 25% of layers are assigned 8-bit precision, while the remaining layers are assigned 4-bit precision. We compare three strategies: TAQ, TAQ-Oracle, and TAQ-KL.

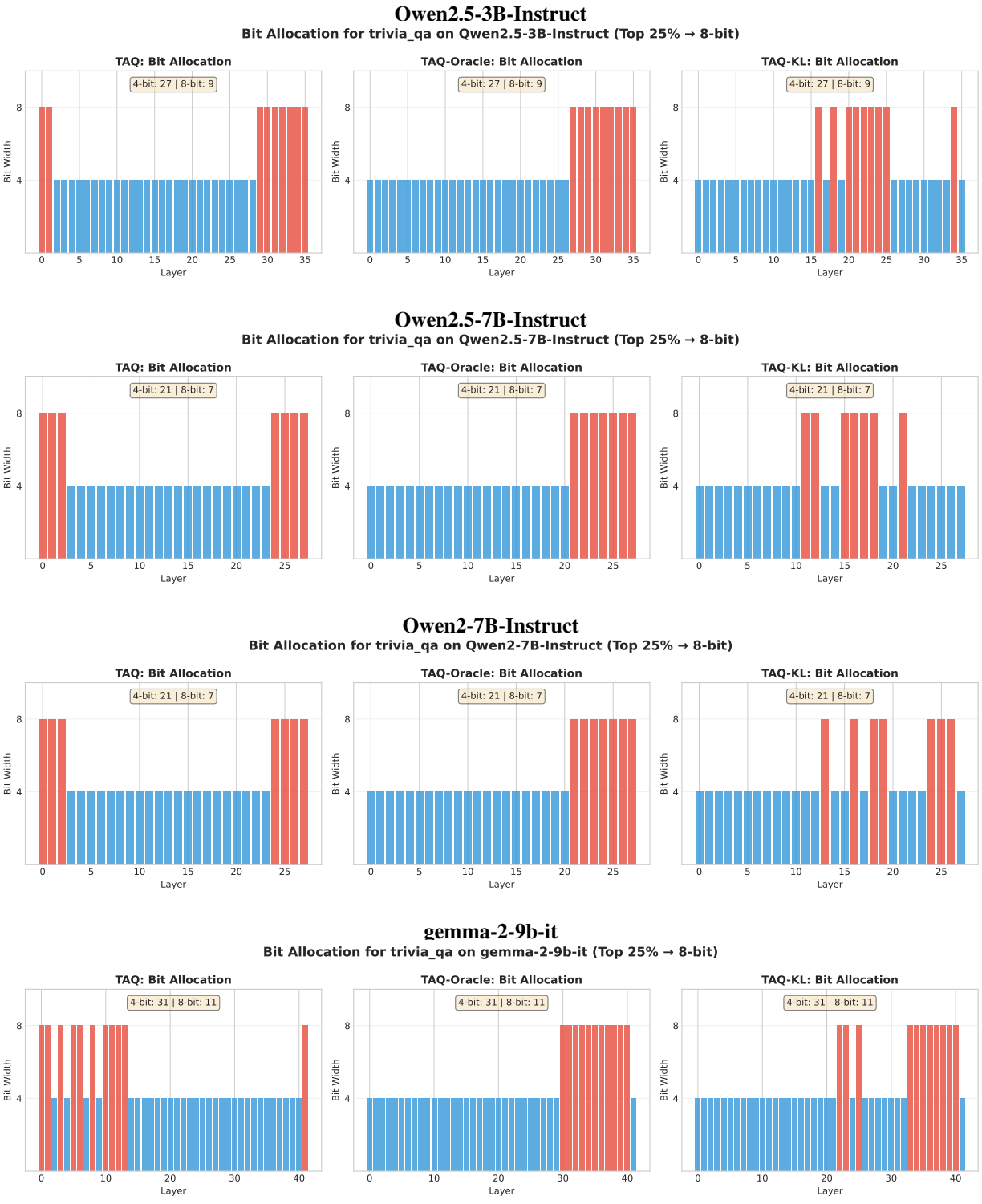

Figure 6. Layerwise mixed-precision allocations on TriviaQA across models. Each plot contains three panels (TAQ, TAQ-Oracle, TAQ-KL). Layers ranked in the top 25% by the respective method are assigned 8-bit precision; remaining layers use 4-bit.

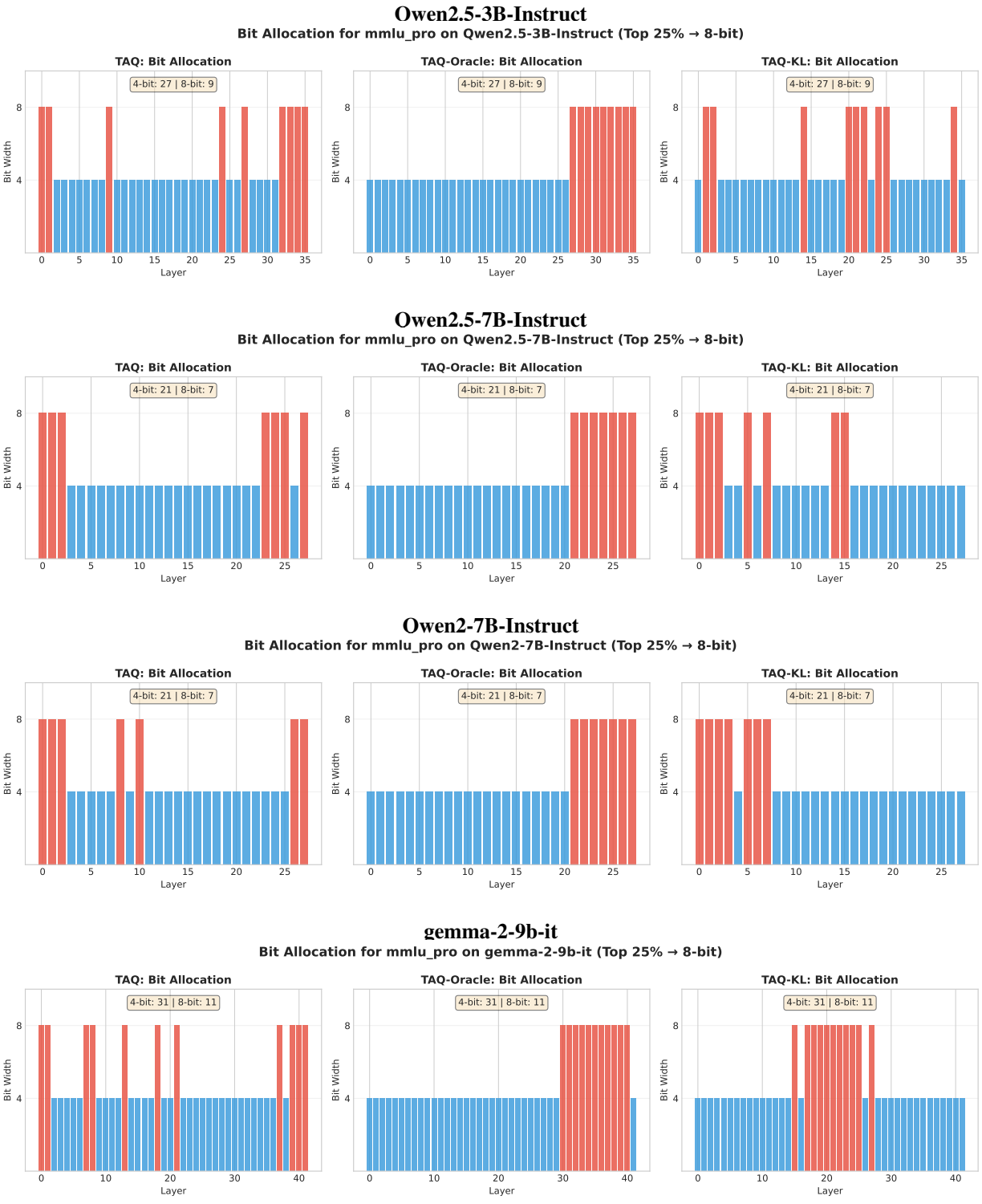

Figure 7. Layerwise mixed-precision allocations on MMLU-Pro across models. The top 25% of layers by each method's score are assigned 8-bit precision; remaining layers use 4-bit.

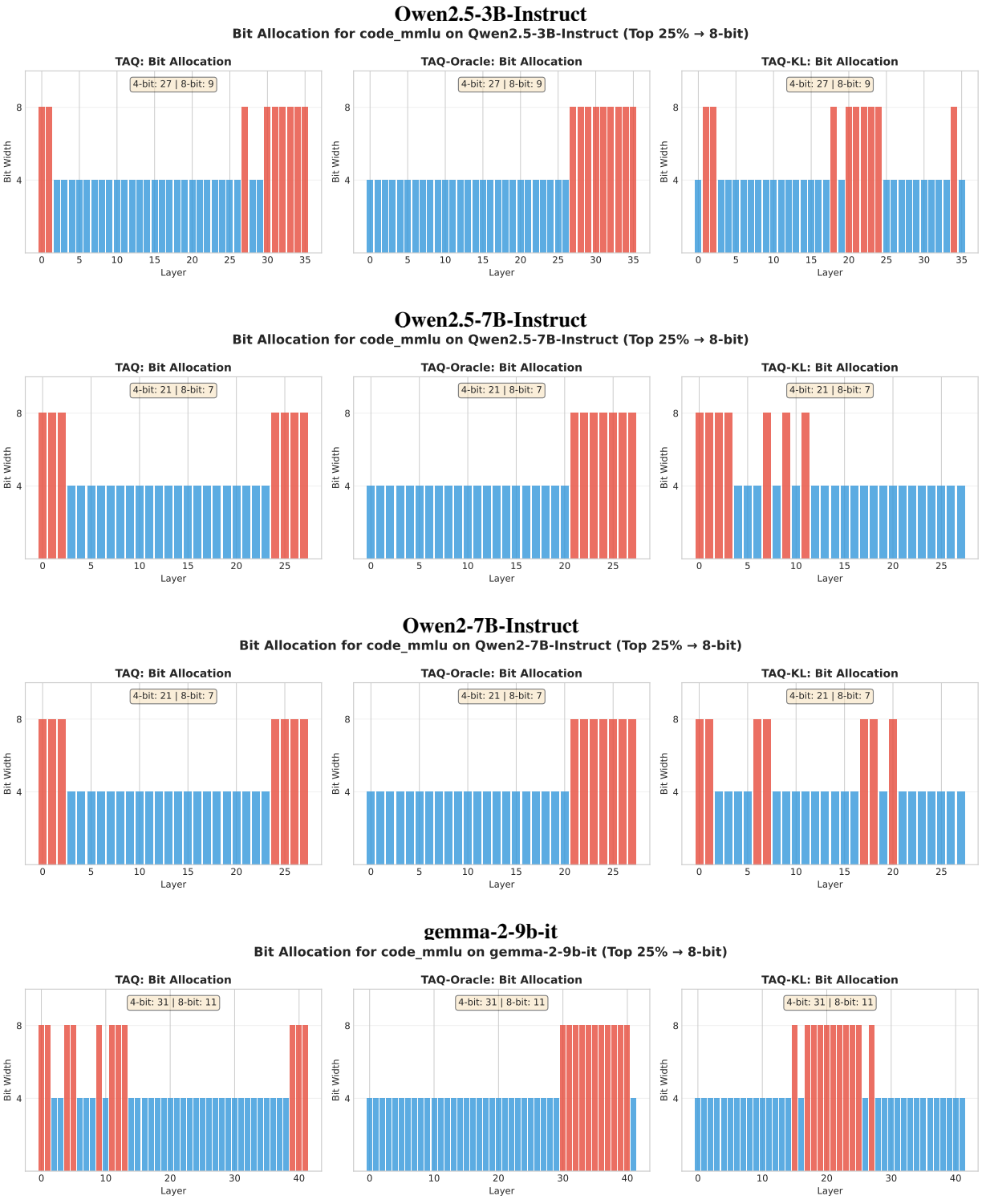

Figure 8. Layerwise mixed-precision allocations on CodeMMLU across models. Layers ranked in the top 25% by TAQ, TAQ-Oracle, or TAQ-KL are allocated 8-bit precision; remaining layers use 4-bit.

A.3. Average bits per layer for TAQ methods

Table 3. **Precision allocation statistics (ours).** Average bits per layer and estimated quantized weight sizes (GB) for each model.

Method	Qwen2.5-3B	Qwen2.5-7B	Qwen2-7B	Gemma-2-9B	Llama-3.1-8B
Avg bits per layer					
TAQ	5.00	5.00	5.00	5.05	5.00
TAQ-Oracle	8.00	9.14	9.14	7.43	8.50
TAQ-KL	5.00	5.00	5.00	5.05	5.00
Estimated quantized weight size (GB)					
TAQ	1.88	4.38	4.38	5.68	5.00
TAQ-Oracle	3.00	8.00	8.00	8.36	8.50
TAQ-KL	1.88	4.38	4.38	5.68	5.00