Q-PAT: QUANTIZATION-AWARE PRUNING-AWARE TUNING FOR LOW-BIT LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Structured pruning and low-bit quantization are two key techniques for reducing the deployment cost of large language models. However, structural decisions made during pruning may misalign with the quantization errors introduced later, potentially degrading model performance under low-bit precision. In this work, we propose Q-PAT, a quantization-aware pruning-aware tuning method tailored for low-bit deployment. Building on the Pruning-Aware Tuning (PAT) framework, Q-PAT constructs a channel-wise quantization-sensitivity proxy based on the INT4 fake-quantization error of low-rank weights. The resulting sensitivity signal is incorporated into a Mask–Quant regularization term through a stop-gradient mechanism, thereby constraining the optimization of trainable structural masks at the loss level without directly injecting quantization perturbations into the main forward pass. This design enables mask optimization to account for heterogeneous low-bit quantization sensitivity across channels during task adaptation, better coordinating task adaptation, structural compression, and quantization error. Experimental results show that Q-PAT preserves strong downstream task performance under low-bit quantization and achieves improved INT4 performance on representative Gemma-7B settings. In particular, Q-PAT improves the average score across 14 downstream tasks from 56.69% to 57.38% and increases MMLU accuracy from 24.04% to 33.64%.

1 INTRODUCTION

Deploying large language models (LLMs) under limited memory and computational budgets requires compression methods that preserve task performance while reducing inference costs. Structured pruning and low-bit quantization address complementary aspects of this challenge: structured pruning removes model dimensions, whereas quantization reduces the numerical precision used to represent the remaining parameters. Methods such as SliceGPT (Ashkboos et al., 2024a) demonstrate the potential of reducing network dimensions while retaining dense computation, and post-training quantization methods such as GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024) enable accurate low-bit weight representations. Together, these approaches motivate compact models whose deployment efficiency depends on both their retained structure and their numerical precision.

However, complementary compression mechanisms do not necessarily yield compatible optimization decisions. When pruning is performed without accounting for subsequent quantization, the retained structure is selected according to an objective that does not explicitly reflect the precision used at deployment. A channel that contributes favorably during higher-precision adaptation may incur substantial approximation error after low-bit quantization. Conversely, two structures with similar pre-quantization performance may respond differently to the same quantization procedure. This possibility motivates the central question of this work: how can structural decisions made during task adaptation account for the quantization sensitivity of the resulting model?

Pruning-Aware Tuning (PAT) (Liu et al., 2025) provides a foundation for studying this question by integrating structural pruning into fine-tuning. Through Hybrid Sparsification Modules and trainable structural masks, PAT lets model adaptation and structure selection proceed together, reducing the gap between learning task-relevant representations and identifying removable dimensions. However, its pruning-aware formulation does not explicitly incorporate a low-bit quantization-sensitivity signal into mask optimization. Consequently, extending PAT to low-bit deployment requires consid-

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

ering not only which channels support adaptation, but also how structural choices interact with the numerical errors introduced by quantization.

The broader value of coordinating pruning and quantization has already motivated joint compression methods. For example, GETA (Qu et al., 2025) combines structured pruning with quantization-aware training. In this paper, we focus on incorporating quantization sensitivity into trainable structural masks within pruning-aware task adaptation. We also investigate whether a weight-based sensitivity proxy can guide these masks through the training objective while preserving the main adaptation forward pass. This formulation places the interaction between structure selection and quantization at the loss level, providing a distinct mechanism for introducing deployment precision into pruning-aware tuning.

We propose Q-PAT (Quantization-Aware Pruning-Aware Tuning), a quantization-sensitivity-guided extension of PAT for low-bit LLM deployment. Q-PAT constructs a channel-wise sensitivity proxy from the INT4 fake-quantization error of low-rank weights. This proxy captures variation in local quantization error across channels and supplies an additional signal for structural optimization during task adaptation. We use it as a surrogate for quantization sensitivity; it does not directly measure the end-to-end task degradation caused by quantizing the deployed model. The motivation is that exposing mask optimization to this channel-level information can help it select structures that better accommodate low-bit approximation.
The main contributions are summarized as follows:

- Formulating an extension of pruning-aware tuning that explicitly accounts for channel-wise quantization sensitivity during structural mask optimization.
- Introducing a sensitivity proxy based on INT4 fake-quantization error and a stop-gradient Mask–Quant regularizer that incorporates this signal at the loss level.
- Demonstrating improved INT4 downstream performance in representative Gemma-7B settings, providing empirical support for coordinating task adaptation, structural compression, and deployment precision through quantization-sensitive mask learning.

2 Related Work

2.1 Structured Pruning of Large Language Models

LLM pruning broadly falls into unstructured and structured approaches. SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2024) estimate weight importance to induce high levels of sparsity, although realizing consistent end-to-end speedups from unstructured sparsity typically requires specialized kernels or hardware. Structured methods instead remove complete channels, hidden dimensions, or Transformer blocks, directly reducing dense computation. LLM-Pruner (Ma et al., 2023) leverages structural dependencies to prune coupled components, whereas SliceGPT (Ashkboos et al., 2024a) exploits computational invariance to remove hidden dimensions. More recent literature explores flexible structural decisions: FLAP (An et al., 2024) uses feature fluctuations for retraining-free pruning, and SLEB (Song et al., 2024) identifies and removes redundant Transformer blocks. Yang et al. (2025) allocate pruning ratios according to interactions among modules; DRPruning (Deng et al., 2025) uses distributionally robust optimization to address uneven degradation across data domains; and Gao et al. (2025) optimize structural masks with policy gradients. PAT brings structural selection into parameter-efficient fine-tuning. Building on this formulation, Q-PAT studies how channel selection can account for the low-bit quantization applied after tuning.

2.2 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning adapts a pretrained model by updating a small set of additional parameters or low-dimensional reparameterizations while keeping most pretrained weights fixed. Low-Rank Adaptation (LoRA) is a widely used parameter-efficient fine-tuning approach based on low-rank parameterizations, with recent work further investigating its initialization and training dynamics (Li et al., 2025). DoRA (Liu et al., 2024) extends this approach by decomposing weights into magnitude and direction. IGU-LoRA (Cui et al., 2026), FlexLoRA (Liu et al., 2026), and RaLoRA (Ye et al., 2026) investigate adaptive rank allocation using integrated gradients and uncertainty, spectral entropy, and gradient intrinsic dimensionality, respectively. These approaches reflect the varying

adaptation capacity required across layers. PAT combines low-rank adaptation with structured pruning in a single training framework. Q-PAT retains this parameter-efficient formulation and augments structural selection with a channel-level signal relevant to low-bit deployment.

2.3 Low-Bit Quantization and Quantization-Aware Adaptation

Low-bit quantization compresses models by reducing the precision of weights or activations. GPTQ (Frantar et al., 2023) uses approximate second-order information for weight quantization, while AWQ (Lin et al., 2024) protects salient weights using activation statistics. OmniQuant (Shao et al., 2024) learns calibration parameters for low-bit mappings; QuaRot (Ashkboos et al., 2024b) applies rotations to mitigate activation outliers; and AQLM (Egiazarian et al., 2024) uses additive quantization for compression at very low bit widths. LoftQ (Li et al., 2024) jointly designs quantization and LoRA initialization, demonstrating how quantization error can be considered during parameter-efficient adaptation. Recent studies also examine the roles of model scale and training completeness (Ouyang et al., 2025), as well as the limits of extremely low-bit post-training quantization (Zhao et al., 2025). These methods primarily optimize numerical representation, whereas structured pruning determines which structural units remain. Q-PAT connects the two by converting quantization errors in low-rank factors into a channel-level regularization signal. Low-bit sensitivity thus influences mask optimization through the loss, while the task forward pass retains floating-point weights.

3 Method

3.1 Overview

Let d denote the model's hidden dimension. Q-PAT extends PAT with a channel-level quantization-sensitivity proxy and a Mask–Quant regularizer in its low-rank adaptation modules. Within each trainable module, task features pass through a floating-point residual transformation. In parallel, the INT4 fake-quantization errors of the two low-rank factors are aggregated into sensitivity estimates aligned with the hidden channels. These estimates are detached during backpropagation and weight the structural regularizer; they neither alter the forward features nor directly overwrite the mask scores. Representation learning and sensitivity-guided mask optimization follow separate gradient paths, while the shared structural mask receives an auxiliary signal reflecting low-bit error. Figure 1 illustrates the framework.

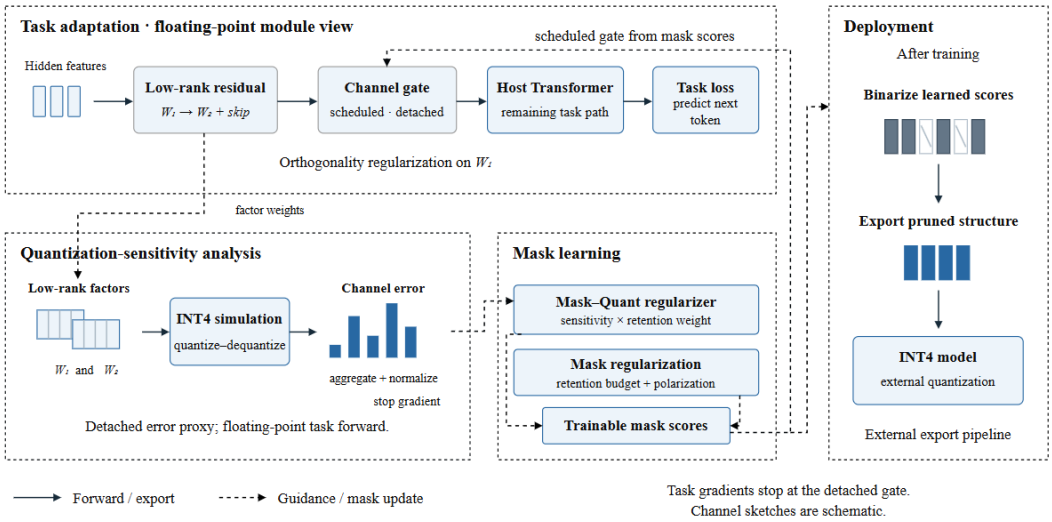


Figure 1: The Q-PAT framework.

3.2 Structural Pruning and Representation Preservation

Q-PAT adopts PAT's low-rank residual transformation and trainable structural mask. For an input hidden state h, let W_1 and W_2 denote the first and second low-rank weight factors. Setting rank r = 200. Equation (1) defines the floating-point residual representation h_{fp}.

$$h_{\mathrm{fp}} = W_2\,(W_1 h) + h \tag{1}$$

The second factor W_2 is initialized to zero, so the residual transformation satisfies $h_{\mathrm{fp}} = h$ exactly at initialization. The complete module also includes channel-wise scaling c and a scheduled gate $a_t(P)$. When the identity-preserving mechanism is enabled, c is a learnable vector initialized to one; otherwise, $c = 1$. The main forward pass uses the detached gate $\mathrm{sg}(a_t)$, and the mask scores P are initialized to zero. For a gate schedule of length T, the temperature and bias can be defined in Equations (2) and (3), respectively.

$$\tau_t = \begin{cases} \left(1 - \frac{\ln(\max(1,t))}{\ln(\max(2,T))} + \varepsilon\right)^{-1}, & t < T, \\ 10^6, & t \geq T, \end{cases} \tag{2}$$

$$\beta_t = \max\left(0.5 - \frac{t}{T},\, 0\right) \tag{3}$$

Set $t_f = \lfloor 0.8T \rfloor$. The temperature changes with the training step for $t < t_f$ and remains at its value at t_f thereafter; the bias continues to follow the current step. We use $\varepsilon = 10^{-6}$ in Equation (2). For numerical stability, the gate uses the exponent-capped function $g(z) = [1 + \exp(\min(-z, 10))]^{-1}$. Because the forward gate is detached, the task loss does not directly update P through this path at any stage of training. Fixing the temperature during the final 20% of the schedule still permits mask updates from the auxiliary losses.

$$a_t = \begin{cases} g\,(\tau_t P) + \beta_t, & t < t_f, \\ g\left(\tau_{t_f}\ \mathrm{sg}(P)\right) + \beta_t, & t \geq t_f, \end{cases} \tag{4}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator. The nonnegative bias in the scheduled gate means that $a_t(P)$ need not lie strictly in [0, 1]. At inference, a deterministic gate with high temperature and zero bias is rounded to obtain a binary mask. For a target retention ratio ρ, the target number of active channels is $d_{\mathrm{tar}} = \mathrm{round}(\rho d)$; we use $\rho = 0.75$ by default. The structural sparsity objective combines a channel-count constraint with a mask-polarization term, as specified in Equations (5)-(7).

$$\mathcal{L}_{count} = 10\,\frac{\left|\sum_{j=1}^{d} \mathcal{B}_{train}\,(P_j) - d_{tar}\right|}{\max\,(d_{tar}, 1)} \tag{5}$$

$$\mathcal{L}_{polar} = \mathrm{Mean}_{P_j > 0}\,(P_j - 0.5)^2 + \mathrm{Mean}_{P_j \leq 0}\,(P_j + 0.5)^2 \tag{6}$$

$$\mathcal{L}_{prune} = \mathcal{L}_{count} + \mathcal{L}_{polar} \tag{7}$$

During training, the binarization function $\mathcal{B}_{\mathrm{train}}(P)$ returns one for $P_j \geq 0$ and uses a straight-through estimator in the backward pass. This counting rule differs from the deterministic inference gate. The polarization term computes separate means over positive and nonpositive mask scores, assigning a mean of zero to an empty group. Together, these terms encourage the target channel count and polarized scores, without imposing a hard constraint on the final number of retained channels.

$$\mathcal{B}_{train}(P_j) = \mathbf{1}[P_j \geq 0]$$
$$m_j = \mathrm{round}(g(10^6 P_j))$$

The retained structure is determined by the actual inference mask m. Although this gate approximates the condition $P_j > 0$ in exact arithmetic, boundary behavior matters in implementation. At $P_j = 0$, g (0)= 0.5 and PyTorch's round-to-even rule returns zero; finite precision can also map very small positive scores to zero. Structure export therefore uses the inference gate itself. Q-PAT introduces no additional top-k selection or sensitivity-based sorting. Physical structure export and final INT4 conversion are performed in the subsequent deployment pipeline. When identity preservation is enabled, Q-PAT regularizes the rows of W_1 toward orthogonality to stabilize the low-rank representation, as in Equation (8). The term $\mathrm{MSE}(W_1 W_1^{\top}, I_r)$, averaged over r^2 elements, constrains correlations among the low-rank basis vectors. It acts on W_1 rather than the structural mask and measures basis orthogonality rather than reconstruction error between the complete module's input and output.

$$\mathcal{L}_{id} = \mathrm{MSE}\left(W_1 W_1^{\top}, I_r\right) \tag{8}$$

3.3 Quantization Sensitivity and Mask Optimization

Q-PAT constructs its structural signal by applying symmetric b-bit fake quantization separately to each output row of W_1 and W_2, with $b = 4$ by default. The operation is applied to the two factors individually, rather than to their product W_2W_1 or the full model. For output row o of either factor W, the scale is computed in FP32 according to Equation (9). The row-wise maximum absolute weight is clamped to a minimum of 10^{-8} before division by $(2^{b-1} - 1)$ to handle all-zero rows.

$$s_o = \frac{\max(\max_i |W_{o,i}|, 10^{-8})}{2^{b-1} - 1} \tag{9}$$

$$Q_b\left(W_{o,i}\right) = s_o \ \text{clip}\left(\text{round}\left(\frac{W_{o,i}}{s_o}\right), -2^{b-1}, 2^{b-1} - 1\right) \tag{10}$$

The quantize–dequantize result is cast back to the original weight dtype. Although the fake-quantization operator can support a straight-through estimator, the error-statistics branch runs without gradient tracking, and the resulting sensitivity is detached. Consequently, the Mask–Quant term does not update W_1 or W_2 through this branch. Fake quantization supplies the error proxy, while the task forward pass continues to use floating-point factors.

$$\widetilde{W} = W + \text{sg}\left(Q_b(W) - W\right) \tag{11}$$

Columns of W_1 correspond to input hidden channels, and rows of W_2 correspond to output hidden channels. We aggregate their fake-quantization errors along the respective dimensions to obtain a two-sided proxy aligned with the same hidden-channel index. Differences between the original and dequantized weights are computed in FP32, as specified in Equations (12)–(14).

$$e_j^{(1)} = \text{Mean}_k \left(W_1[k, j] - \widetilde{W}_1[k, j]\right)^2 \tag{12}$$

$$e_j^{(2)} = \text{Mean}_k \left(W_2[j, k] - \widetilde{W}_2[j, k]\right)^2 \tag{13}$$

$$e_j = e_j^{(1)} + e_j^{(2)} \tag{14}$$

The sum of these two error terms is min–max normalized within each module to produce the channel sensitivity S_j in Equation (15). Setting $\varepsilon = 10^{-8}$ to stabilize the denominator. This normalization measures relative differences among channels within a module; the resulting values are not absolute sensitivity estimates that can be compared directly across modules or quantization protocols.

$$S_j = \frac{e_j - \min(e)}{\max(e) - \min(e) + \varepsilon} \tag{15}$$

A larger $S_j \in [0, 1]$ indicates greater relative fake-quantization error in the low-rank factors associated with channel j. The quantity serves as a local error proxy, rather than a direct measurement of task loss after deployment. We recompute it at every forward pass without an exponential moving average and treat it as constant during backpropagation through the Mask–Quant branch. The Mask–Quant objective couples these sensitivity estimates to the structural mask. We define $q_j = \sigma(P_j)$ as the continuous retention weight used by the regularizer. This quantity is distinct from the scheduled forward gate $a_t(P)$ and is not a stochastic sampling probability. Equation (16) couples q_j with $\text{sg}(S_j)$, assigning a larger penalty to channels that combine high sensitivity with high retention weight.

$$\mathcal{L}_{MQ} = \frac{1}{d} \sum_{j=1}^{d} \sigma\left(P_j\right) \ \text{sg}\left(S_j\right) \tag{16}$$

With S_j detached, the regularizer acts directly on the mask scores P. For a fixed P_j, a larger S_j produces a larger positive gradient, so gradient descent pushes that score downward and reduces its retention weight. Differentiating Equation (16) is defined as follows:

$$\frac{\partial \mathcal{L}_{MQ}}{\partial P_j} = \frac{1}{d} \text{sg}\left(S_j\right) \sigma\left(P_j\right) \left[1 - \sigma\left(P_j\right)\right]$$

The factor $\sigma(P_j)(1 - \sigma(P_j))$ is largest near $P_j = 0$, so a higher retention weight does not necessarily imply a larger gradient. The regularizer establishes a structural preference against locally

sensitive channels. Its effect on the learned model arises in the combined training objective, and we evaluate downstream performance empirically. Equation (17) combines the pruning constraint, identity-preserving term, and Mask–Quant regularizer into the auxiliary objective for a trainable mask module. The identity term is active when identity preservation is enabled; the Mask–Quant term is evaluated when the module's internal mask is trainable.

$$\mathcal{L}_{aux} = \mathcal{L}_{prune} + \mathcal{L}_{id} + \lambda_{MQ}\mathcal{L}_{MQ} \tag{17}$$

The default Mask-Quant coefficient is λ_{MQ} = 0.02 and can be adjusted in the experimental configuration. The task loss updates the trainable low-rank factors and channel-wise scaling, while the detached forward gate blocks direct task gradients to *P*. Mask scores receive gradients primarily from the pruning constraint and Mask–Quant term; the identity-preserving term acts on W_1. Let *M* denote the number of modules contributing auxiliary losses. When the outer training loop sums these losses with unit weights, the complete objective takes the form of Equation (18).

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \sum_{\ell=1}^{M} \mathcal{L}_{aux}^{(\ell)} \tag{18}$$

4 Experiments

We evaluate Q-PAT from five perspectives: overall downstream performance, performance retention after low-bit quantization, component contributions, sensitivity to quantization group size, and the relationship between sensitivity and channel selection.

4.1 Experimental Setup

Table 1 summarizes the experimental setup using Gemma-2B, Gemma-7B, and Llama2-7B with 25% structural pruning, evaluated in BF16 and group-wise INT4 . PAT and Q-PAT are tested across all settings, while LLM-Pruner and SliceGPT are additionally evaluated on Gemma-7B with 1.5M examples; 14-task Avg denotes the mean of 13 zero-shot benchmarks and MMLU.

Table 1: Experimental configuration.

Item	**Configuration**
Backbones	Gemma-2B, Gemma-7B, Llama2-7B
Training data	LaMini; 200,000 or 1.5 million examples for each backbone
Pruning budget	25% structural pruning target (75% retention)
Training schedule	Sequence length 512; 800 training steps
Evaluation precision	BF16 and group-wise INT4 (W4A16); default group size 128; LoRA parameters and lm_head remain in BF16
Group-size sweep	$g \in \{32, 64, 128, 256\}$
Benchmarks	13 standard zero-shot benchmarks and MMLU (14 benchmarks in total)
Metrics	14-task Avg and MMLU. MMLU uses the official lm-eval group aggregate and receives the same weight as each of the other 13 benchmarks.

4.2 Baselines

We compare Q-PAT with three baselines: PAT (Liu et al., 2025), LLM-Pruner (Ma et al., 2023), and SliceGPT (Ashkboos et al., 2024a). PAT integrates structural pruning into parameter-efficient fine-tuning through low-rank adaptation and trainable channel masks, and serves as the primary comparator across Gemma-2B, Gemma-7B, and Llama2-7B at both training-data sizes. LLM-Pruner removes coupled components using structural dependencies, while SliceGPT exploits computational invariance to remove hidden dimensions. These two methods provide additional structured-pruning comparisons on Gemma-7B with 1.5 million training examples.

4.3 Overall Downstream Performance

Table 2 reports 14-task Avg for PAT and Q-PAT across three backbones, two training-data sizes, and two numerical precisions. The paired BF16 and INT4 results characterize performance before and

after quantization. We further examine MMLU in Section 4.4 to distinguish changes in the overall average from those on a knowledge-intensive benchmark.

Table 2: Overall downstream performance of PAT and Q-PAT.

Backbone	Data	PAT BF16 ↑	Q-PAT BF16 ↑	PAT INT4 ↑	Q-PAT INT4 ↑
Gemma-2B	200k	0.579144	0.619242	0.571821	0.579468
Gemma-2B	1.5M	0.591224	0.606599	0.568814	0.578945
Gemma-7B	200k	0.668742	0.667575	0.563141	0.567844
Gemma-7B	1.5M	0.666692	0.671831	0.566879	0.573781
Llama2-7B	200k	0.632733	0.632487	0.630745	0.631172
Llama2-7B	1.5M	0.629534	0.631066	0.625409	0.631095

Q-PAT achieves a higher INT4 14-task Avg than PAT in all six configurations in Table 2. On Gemma-2B, the improvements are 0.76 and 1.01 percentage points with 200,000 and 1.5 million training examples, respectively. The corresponding gains on Gemma-7B are 0.47 and 0.69 points. On Llama2-7B, the methods are nearly tied at 200,000 examples, with Q-PAT ahead by 0.04 points, while the gain increases to 0.57 points at 1.5 million examples. The BF16 comparison varies across backbones: Q-PAT improves Gemma-2B by 4.01 and 1.54 points, whereas the differences on Gemma-7B and Llama2-7B range from -0.12 to +0.51 points. These results establish consistent INT4 gains across the tested settings, with the magnitude of both floating-point and low-bit improvements depending on the backbone and data scale.

4.4 Performance Retention after Low-Bit Quantization

To separate pre-quantization performance from robustness to the subsequent numerical conversion, we examine Gemma-7B trained on 1.5 million examples with a 25% structural pruning target. Table 3 pairs BF16 and group-wise INT4 results at group size 128 for PAT, Q-PAT, LLM-Pruner, and SliceGPT. Alongside absolute INT4 scores, we report each method's BF16-to-INT4 drop to quantify the additional degradation associated with quantization.

Table 3: Performance before and after quantization on Gemma-7B with 1.5 million training examples and a 25% pruning target. INT4 uses group size 128. Scores and drops are on a 0–1 scale; drop = BF16 - INT4.

Method	MMLU BF16 ↑	MMLU INT4 ↑	MMLU Drop ↓	Avg BF16 ↑	Avg INT4 ↑	Avg Drop ↓
PAT	0.603475	0.240350	0.363125	0.666692	0.566879	0.099813
Q-PAT	**0.612876**	**0.336419**	**0.276457**	**0.671831**	**0.573781**	**0.098050**
LLM-Pruner	0.517538	0.201573	0.315965	0.537259	0.427458	0.109801
SliceGPT	0.507346	0.197461	0.309885	0.548225	0.403645	0.144580

Q-PAT obtains the highest INT4 MMLU accuracy (33.64%) and 14-task Avg (57.38%) among the methods in Table 3. Relative to PAT, it reduces the MMLU quantization drop from 36.31 to 27.65 percentage points, an improvement of 8.67 points computed from the unrounded scores. The corresponding drops in 14-task Avg are close: 9.81 points for Q-PAT and 9.98 for PAT. Q-PAT's INT4 MMLU accuracy exceeds those of LLM-Pruner and SliceGPT by 13.48 and 13.90 points, respectively, while its average score is higher by 14.63 and 17.01 points. The paired results reveal two sources of the final INT4 advantage: higher pre-quantization scores and reduced quantization degradation. In the comparison with PAT, the reduction in degradation is substantial on MMLU and smaller on the aggregate metric.

4.5 Ablation Study

We ablate Q-PAT on Gemma-7B using 1.5 million LaMini examples, a 25% structural pruning target, and group-wise INT4 quantization (W4A16, group size 128). Starting from the method, we separately remove quantization sensitivity modeling (w/o QSM), mask-aware coupling (w/o Mask-aware), and the identity-preserving constraint (w/o Identity).

Table 4: Component ablations on Gemma-7B with 1.5 million training examples and INT4 group size 128. Q-PAT denotes the independently trained reference run for this ablation study.

Task	Q-PAT ↑	w/o QSM ↑	w/o Identity ↑	w/o Mask-aware ↑
ARC-Challenge	0.404437	**0.412969**	0.403584	0.401877
ARC-Easy	0.642677	0.643939	0.635101	**0.647727**
BoolQ	**0.663303**	0.661468	0.654740	0.661162
COPA	0.820000	**0.830000**	0.780000	0.710000
HellaSwag	**0.543119**	0.540430	0.514439	0.542621
MultiRC	0.548061	0.547855	0.543729	**0.549917**
OpenBookQA	**0.408000**	0.406000	0.396000	0.402000
PIQA	0.729597	0.722524	0.717084	**0.732318**
RTE	**0.653430**	0.624549	0.638989	0.649819
Social-IQA	**0.331627**	**0.331627**	0.331116	0.331116
WiC	0.498433	0.496865	0.493730	**0.501567**
WinoGrande	0.656669	0.647987	**0.666930**	0.662194
WSC273	0.706960	0.732601	0.739927	**0.747253**
MMLU	**0.336419**	0.335351	0.303091	0.322615
14-task Avg	**0.567338**	0.566726	0.558461	0.561584

Q-PAT achieves the highest 14-task Avg in Table 4 (0.567338). Removing QSM, mask-aware coupling, or identity preservation lowers the average to 0.566726, 0.561584, and 0.558461, corresponding to reductions of 0.06, 0.58, and 0.89 percentage points. Identity preservation has the largest effect in this comparison, followed by mask-aware coupling; removing QSM yields a smaller change in the same direction. These component-wise results support the combined design. Table 4 uses an independently trained reference run, so its absolute scores differ from the main results in Table 3; the observed differences are descriptive and do not establish statistical significance.

4.6 Sensitivity to Quantization Group Size

We test whether Q-PAT's relative gains depend on the default quantization group size. For each method, we hold the Gemma-7B checkpoint trained on 1.5 million examples and all other quantization settings fixed, vary the INT4 group size over $g \in \{32, 64, 128, 256\}$, and measure the 14-task Avg.

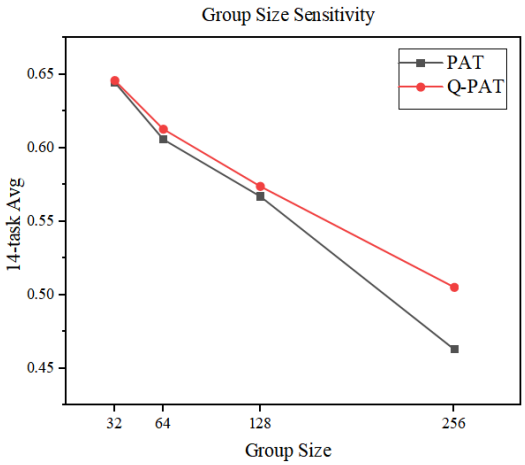


Figure 2: INT4 performance of PAT and Q-PAT on Gemma-7B across quantization group sizes.

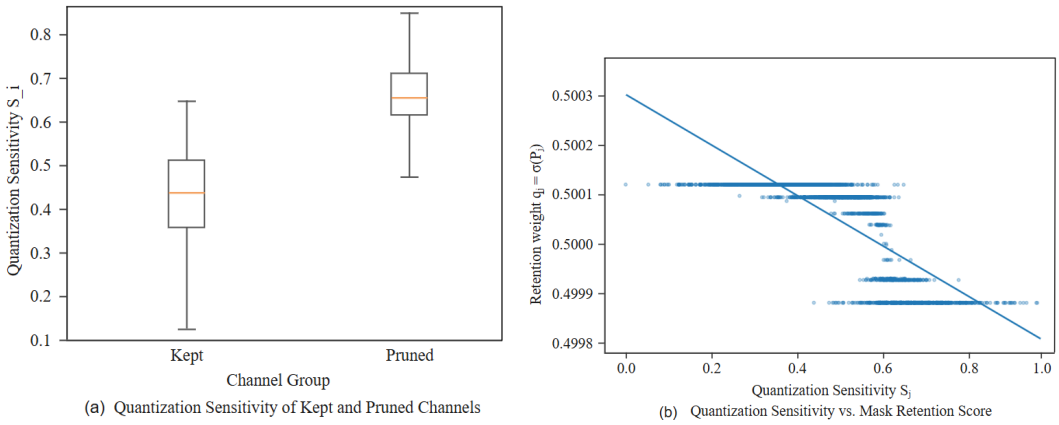
As group size increases from 32 to 256, both algorithms show a monotonic decline in 14-task Avg (Figure 2), consistent with greater degradation under coarser quantization. Q-PAT retains a higher

average score at every tested group size, and the gap is largest at g = 256. Therefore, its relative advantage extends beyond the default g = 128 configuration.

4.7 Quantization Sensitivity and Channel Selection

To measure how the Mask–Quant objective relates to the learned structure, we analyze the shared structural mask from Full Q-PAT on Gemma-7B trained with 1.5 million examples. Figure 3(a) compares sensitivity distributions for retained and pruned channels, and Figure 3(b) relates sensitivity to the continuous retention weight. We normalize sensitivity within each module using Equation (15). The retention weight is the sigmoid of the mask score and differs from the binary inference mask.

Pruned channels exhibit a higher sensitivity distribution than retained channels in the analyzed branch (Figure 3(a)). The Spearman coefficient of -0.9761 in Figure 3(b) indicates a strong negative rank association between sensitivity and continuous retention weight, consistent with the direction of the Mask–Quant penalty.This coefficient describes channel ordering, rather than the absolute size of changes in retention weight or their stability across training runs. Because sensitivity is normalized within modules, this branch-level association should not be interpreted as a uniform relationship across the entire model. The analysis illustrates how sensitivity and selection align in the learned mask; Tables 2–4 provide the separate evidence for downstream performance, without attributing a causal performance effect to the correlation alone.



(a) Quantization Sensitivity of Kept and Pruned Channels

(b) Quantization Sensitivity vs. Mask Retention Score

Figure 3: Quantization sensitivity and channel selection in the analyzed branch.

5 Conclusion and Future Directions

We introduced Q-PAT, a quantization-aware extension of Pruning-Aware Tuning for low-bit large language models. Q-PAT addresses the potential mismatch between structural decisions made during task adaptation and quantization errors introduced at deployment. Specifically, it derives a channel-wise quantization-sensitivity proxy from the INT4 fake-quantization error of low-rank weights and incorporates this signal into structural mask optimization through a Mask–Quant regularizer with a stop-gradient mechanism. This design enables quantization sensitivity to guide structure selection at the loss level without directly introducing quantization perturbations into the main forward pass. Experiments on Gemma-7B show improvements in both the average score across 14 downstream tasks and MMLU accuracy under INT4 quantization. These results support the use of quantization sensitivity to guide pruning-aware adaptation in the evaluated settings. Future work will examine the consistency of these gains across model architectures, pruning ratios, and quantization configurations, and investigate how well the low-rank sensitivity proxy captures quantization-induced degradation in the final compressed model.

AI USE STATEMENT

Generative AI tools were used solely for language polishing after the manuscript had been completed by the authors.

REFERENCES

Y. An, X. Zhao, T. Yu, M. Tang, and J. Wang. Fluctuation-Based Adaptive Structured Pruning for Large Language Models. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 10, pp. 10865–10873, 2024.

S. Ashkboos, M. L. Croci, M. G. do Nascimento, T. Hoefler, and J. Hensman. SliceGPT: Compress Large Language Models by Deleting Rows and Columns. International Conference on Learning Representations (ICLR), 2024a.

S. Ashkboos, A. Mohtashami, M. L. Croci, B. Li, P. Cameron, M. Jaggi, D. Alistarh, T. Hoefler, and J. Hensman. QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs. Advances in Neural Information Processing Systems, vol. 37, 2024b.

X. Cui, H. Li, R. Zeng, Y. Zhao, J. Qian, W. Duan, B. Liu, and Z. Zhou. IGU-LoRA: Adaptive Rank Allocation via Integrated Gradients and Uncertainty-Aware Scoring. International Conference on Learning Representations (ICLR), 2026.

H. Deng, W. Jiao, X. Liu, J. Li, M. Zhang, and Z. Tu. DRPruning: Efficient Large Language Model Pruning through Distributionally Robust Optimization. Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 29152–29173, 2025.

V. Egiazarian, A. Panferov, D. Kuznedelev, E. Frantar, A. Babenko, and D. Alistarh. Extreme Compression of Large Language Models via Additive Quantization. Proceedings of the 41st International Conference on Machine Learning, PMLR 235, pp. 12284–12303, 2024.

E. Frantar and D. Alistarh. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. Proceedings of the 40th International Conference on Machine Learning, PMLR 202, pp. 10323–10337, 2023.

E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. International Conference on Learning Representations (ICLR), 2023.

Y. Gao, Z. Liu, W. Zhang, B. Du, and G.-S. Xia. Bypass Back-propagation: Optimization-based Structural Pruning for Large Language Models via Policy Gradient. Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 29356–29377, 2025.

S. Li, X. Luo, X. Tang, H. Wang, H. Chen, W. Luo, Y. Li, X. He, and R. Li. Beyond Zero Initialization: Investigating the Impact of Non-Zero Initialization on LoRA Fine-Tuning Dynamics. Proceedings of the 42nd International Conference on Machine Learning, PMLR 267, pp. 35519–35535, 2025.

Y. Li, Y. Yu, C. Liang, N. Karampatziakis, P. He, W. Chen, and T. Zhao. LoftQ: LoRA-Fine-Tuning-aware Quantization for Large Language Models. International Conference on Learning Representations (ICLR), 2024.

J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. Proceedings of Machine Learning and Systems, vol. 6, 2024.

M. Liu, C. Si, and Y. Jia. FlexLoRA: Entropy-Guided Flexible Low-Rank Adaptation. International Conference on Learning Representations (ICLR), 2026.

S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. DoRA: Weight-Decomposed Low-Rank Adaptation. Proceedings of the 41st International Conference on Machine Learning, PMLR 235, pp. 32100–32121, 2024.

Y. Liu, H. Yang, Y. Chen, R. Zhang, M. Wang, Y. Du, and L. Du. PAT: Pruning-Aware Tuning for Large Language Models. Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, no. 23, pp. 24686–24695, 2025.

X. Ma, G. Fang, and X. Wang. LLM-Pruner: On the Structural Pruning of Large Language Models. Advances in Neural Information Processing Systems, vol. 36, pp. 21702–21720, 2023.

X. Ouyang, T. Ge, T. Hartvigsen, Z. Zhang, H. Mi, and D. Yu. Low-Bit Quantization Favors Undertrained LLMs. Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 32338–32348, 2025.

X. Qu, D. Aponte, C. Banbury, D. P. Robinson, T. Ding, K. Koishida, I. Zharkov, and T. Chen. Automatic Joint Structured Pruning and Quantization for Efficient Neural Network Training and Compression. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 15234–15244, 2025.

W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. Qiao, and P. Luo. OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models. International Conference on Learning Representations (ICLR), 2024.

J. Song, K. Oh, T. Kim, H. Kim, Y. Kim, and J.-J. Kim. SLEB: Streamlining LLMs through Redundancy Verification and Elimination of Transformer Blocks. Proceedings of the 41st International Conference on Machine Learning, PMLR 235, pp. 46136–46155, 2024.

M. Sun, Z. Liu, A. Bair, and J. Z. Kolter. A Simple and Effective Pruning Approach for Large Language Models. International Conference on Learning Representations (ICLR), 2024.

M. Yang, S. Lin, C. Li, and X. Chang. Let LLM Tell What to Prune and How Much to Prune. Proceedings of the 42nd International Conference on Machine Learning, PMLR 267, pp. 70833–70849, 2025.

J. Ye, H. He, M. Li, F. Han, T. Chen, and P. Ye. Gradient Intrinsic Dimensionality Alignment: Narrowing the Gap Between Low-Rank Adaptation and Full Fine-Tuning. International Conference on Learning Representations (ICLR), 2026.

J. Zhao, M. Zhang, M. Wang, Y. Shang, K. Zhang, W. Guan, Y. Wang, and M. Zhang. PTQ1.61: Push the Real Limit of Extremely Low-Bit Post-Training Quantization Methods for Large Language Models. Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 4483–4502, 2025.