FAMPWQ: Fisher Information-based Adaptive Mixed Precision Weight Quantization for Effective LLM Inference

Gongwei Lee[1,3†], **Ji Liu**[3,2†*], **Juncheng Jia**[1], **Ji Wu**[2]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]Electronic Engineering, Tsinghua University, Beijing, China
[3]Hithink Research, Hangzhou, China

Abstract

Recent years have witnessed remarkable achievements of Large Language Models (LLMs) in multiple domains, while the excessive resource requirements of LLMs hinder the deployment on resource-constrained devices. Although model quantization stands out as an effective approach, conventional quantization approaches typically incur severe performance degradation due to uniform bit-width or simple heuristic sensitivity evaluation. In this paper, we propose a novel Fisher information-based Adaptive Mixed Precision Weight Quantization approach, i.e., FAMPWQ, which performs layer-adaptive weight quantization for effective LLM inference on commodity GPUs. First, we propose a system model with a novel Fisher information metric to measure the layer-wise sensitivity to quantization. Second, we propose a reinforcement learning-based bit-width allocator in FAMPWQ, which generates an adaptive bit-width allocation strategy based on the Fisher information sensitivity metric. Extensive experiments on 7 models and 5 benchmarks demonstrate that FAMPWQ significantly outperforms 7 baseline approaches in terms of PPL (up to 3.39 smaller), accuracy (up to 6.87% higher), and LLM-as-a-judge comparison (up to 76% win rate).

1 Introduction

Recent years have witnessed remarkable progress of Large Language Models (LLMs) across a wide range of applications (Radford et al., 2019; Petroni et al., 2019; Brown et al., 2020). Most state-of-the-art LLMs are built upon the Transformer architecture (Vaswani et al., 2017), achieving strong performance by scaling model size to hundreds billions (Agarwal et al., 2025) or trillions (DeepSeek-AI, 2026) of parameters.

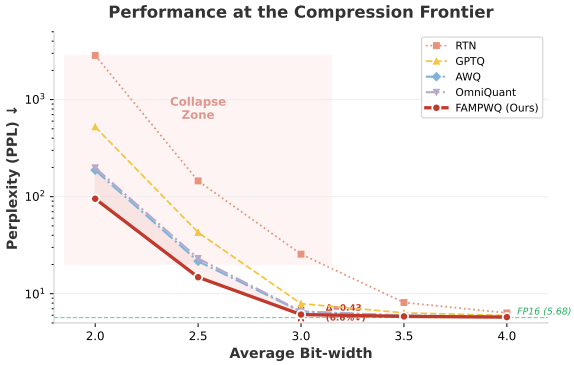


Figure 1: Compression-frontier behavior on LLaMA-7B with WikiText-2. FAMPWQ keeps PPL lower as the average bit-width approaches 3 bits, where uniform quantization methods degrade rapidly.

However, the prohibitive resource requirements of LLMs hinder their deployment on resource-constrained devices, such as edge devices, consumer GPUs, or even inference GPUs. For instance, LLaMA (Touvron et al., 2023a,b; Dubey et al., 2024) spans from 7B to 405B, which may consume from 26GB to 1500GB memory with FP32, and up to 750GB memory with FP16. Similarly, the scale of Qwen (Bai et al., 2023; Team, 2024; Yang et al., 2025) can reach up to 235B corresponding to 435GB memory with FP16. In addition, DeepSeek-V4 (DeepSeek-AI, 2026) goes even larger: its 1.6T-parameter MoE (49B activated) requires about 865GB in mixed FP4/FP8, which significantly exceeds the memory of GPUs. In addition, the memory requirement scales from linear to quadratic with the sequence length. To deploy LLMs on resource-constrained devices, model quantization stands out as an effective approach.

Post-Training Quantization (PTQ) approaches (Zhu et al., 2024) directly quantize pre-trained LLMs without architectural modifications or retraining, albeit typically incurring performance degradation. However, existing PTQ approaches generally recognize the heterogeneous importance distribution of model weights (Dong et al., 2019;

† Equal contribution.
* Corresponding author: Ji Liu (jiliuwork@gmail.com)

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 34781–34801
October 24-29, 2026 ©2026 Association for Computational Linguistics

Gong et al., 2024; Wei et al., 2022), with their key differentiation stemming from the statistical approaches exploited to identify and preserve critical weights. While some existing quantization approaches, e.g., GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024), successfully reduce memory consumption through fixed bit-widths and outlier optimization, they nevertheless suffer from two fundamental limitations. First, their uniform bit-width allocation overlooks crucial layer-wise sensitivity variations, particularly in attention layers. Second, their localized outlier handling fails to account for global importance patterns across the LLM. As a consequence, the existing PTQ approaches may bring unacceptably severe performance degradation in real-life scenarios.

While some mixed-precision approaches, e.g., OWQ (Lee et al., 2024) and AMQ (Lee et al., 2025), attempt to address layer heterogeneity, they rely on coarse heuristics such as weight magnitude or raw gradient norms that fail to faithfully reflect quantization-induced degradation.

A fundamental challenge lies in the *heterogeneous sensitivity* of LLM layers to quantization. Empirically, we find that certain layers (particularly **attention value projections** and **MLP down-projections**) are orders of magnitude more sensitive than others. This reveals even a small number of aggressively quantized sensitive layers can disproportionately degrade model quality, while many redundant layers can tolerate extreme compression with negligible impact. Accurately identifying which layers are critical therefore becomes the key to effective mixed-precision quantization.

Existing sensitivity metrics (Frantar and Alistarh, 2022; Lin et al., 2024), however, are ill-suited to this task. Weight magnitude and gradient norms capture only first-order statistics and do not reflect the geometry of the loss surface under quantization-specific perturbations. Second-order point estimates, including Hessian-based (Dong et al., 2019) and standard Fisher-based metrics, evaluate curvature only at the unperturbed weights and remain agnostic to the bit-width-specific noise that quantization actually injects. To bridge this gap, we propose a *perturbation-based Fisher Information* metric that directly injects quantization-simulating perturbations into layer weights and measures the resulting shift in the FIM. Different from these point-estimate metrics, our formulation captures how quantization noise, rather than arbitrary parameter variations, distorts the local loss geometry,

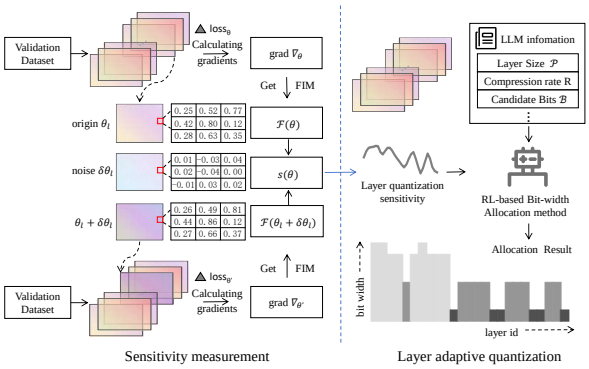


Figure 2: System model of FAMPWQ.

providing a principled and quantization-specific layer sensitivity measure.

In this paper, we propose a Fisher Information-based Adaptive Mixed Precision Weight Quantization (FAMPWQ) approach, i.e., a weight-only mixed-precision PTQ framework for fixed-memory LLM deployment. FAMPWQ introduces a quantization-perturbation Fisher sensitivity metric that estimates per-layer degradation than magnitude/gradient proxies, and exploits a low-cost proxy optimizer to allocate layer bitwidths under a storage budget. As shown in Figure 2, FAMPWQ consists of two stages: a perturbation-based Fisher sensitivity measurement stage and a Reinforcement Learning (RL)-based adaptive bit-width allocation stage. As shown in Figure 1, while uniform PTQ approaches are competitive around 4 bits, their PPL rises sharply below 3.5 average bits. By preserving sensitive layers and compressing tolerant layers more aggressively, FAMPWQ maintains significantly lower degradation below the 3-bit frontier.

The major contributions are as follows:

1. We propose a system model with a novel sensitivity measurement method based on a new Fisher Information metric for layer adaptive quantization. The Fisher Information metric explicitly injects quantization-simulating perturbations and measures the resulting Fisher shift to capture layer-wise sensitivity to quantization loss.

2. We propose an adaptive bit-width allocator in FAMPWQ to enable storage-constrained mixed-precision search guided by quantization-specific loss geometry. The allocator generates an adaptive bit-width allocation strategy based on Proximal Policy Optimization (PPO) and the quantization perturbation Fisher sensitivity of each layer, for layer-wise quantization of LLMs.

3. We implement FAMPWQ and maintain the compatibility with existing methods, e.g., GPTQ or AWQ. We carry out extensive experiments on 7 models and 5 benchmarks to demonstrate that FAMPWQ significantly outperforms 7 baseline approaches in terms of PPL (up to 3.39 smaller), accuracy (up to 6.87% higher), and LLM-as-a-judge comparison (up to 76% win rate).

2 Related Work

Recent LLM post-training quantization (PTQ) studies improve compression by reducing quantization error, correcting outliers, or calibrating quantized weights. SmoothQuant (Xiao et al., 2023) redistributes quantization difficulty between weights and activations, GPTAQ (Li et al., 2025) mitigates error accumulation through calibration, and OmniQuant (Shao et al., 2024) and ABQ-LLM (Zeng et al., 2025) further explore adaptive clipping and bit-balance strategies for low-bit settings. Other methods, such as SqueezeLLM (Kim et al., 2024) and OWQ (Lee et al., 2024), preserve salient weights or channels at higher precision. These works show the importance of protecting sensitive parameters, but their adaptation is generally local and does not directly optimize layer-wise precision under a global memory budget.

This limitation has motivated mixed-precision and search-based allocation. DeepSeek-V4 models (DeepSeek-AI, 2026) adopt mixed FP4/FP8 expert-aware quantization for MoE deployment at expert granularity. AMQ (Lee et al., 2025) exploits activation-guided mixed precision, HAQ (Wang et al., 2019) employs reinforcement learning for hardware-aware bit-width search, CO-PAL (Malla et al., 2024) formulates layer-wise allocation as combinatorial optimization, and BitWeaver (Gagnon et al., 2025) investigates hardware-efficient mixed-precision layouts. RL-PTQ (Wang et al., 2024b) applies reinforcement learning, but depends on repeated model-level evaluation. FAMPWQ focuses on layer-wise sensitivity differences under a global memory budget; it estimates layer sensitivity via Fisher Information and employs a proxy-guided PPO allocator to efficiently search storage-constrained bit-width configurations, achieving superb performance.

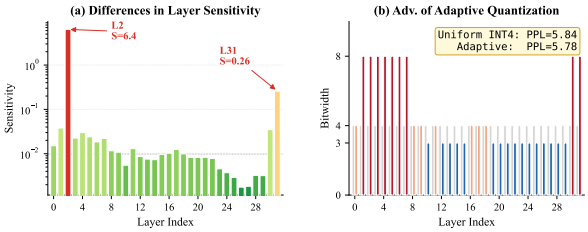


Figure 3: Motivation for adaptive bitwidth quantization.

3 System Model and Problem Formulation

In this section, we present the system model of FAMPWQ, and formulate the problem to address in LLM quantization.

3.1 System Model

While conventional quantization approaches employ identical bit-widths for all layers, LLM layers exhibit highly uneven tolerance to quantization. As shown in Figure 3(a), the measured layer sensitivity spans orders of magnitude within the same model, indicating that a few critical layers can dominate quantization-induced degradation. Figure 3(b) further shows that, under a comparable average precision budget, a mixed-precision allocation achieves lower WikiText-2 PPL than uniform INT4 by assigning higher bit-widths to sensitive layers and lower bit-widths to tolerant ones. These observations motivate a layer-adaptive quantization framework that explicitly measures sensitivity and allocates precision under a global storage constraint.

As shown in Figure 2, the system model of FAMPWQ consists of two stages: sensitivity measurement and layer-adaptive quantization. In the sensitivity measurement stage, we inject perturbation noise into layer weights and compute the Fisher information change to quantify each layer's sensitivity (Section 4.1). In the layer-adaptive quantization stage, an RL-based method allocates appropriate bit-widths to each layer based on the computed sensitivity (Section 4.2). The resulting allocation strategy is then applied via any compatible quantization method (e.g., AWQ).

3.2 Problem Formulation

Let us consider an LLM M composed of L layers. The sensitivity value of Layer l is denoted by $s_l, l \in [1, L]$. In order to quantize the LLM, we define a set of available bit-width options, $\mathcal{B}$, comprising one or more discrete choices (e.g., $\{4, 16\}$ or $\{2, 3, 4, 8\}$). We assume that the sensitivity of each layer is independent of its allocated bit-width q_l. Then, the accuracy degradation incurred by

quantizing an individual layer ΔAcc_l is proportional to its sensitivity and decreases exponentially with increasing bit-width (Zhou et al., 2018), as shown in Formula 1:

$$\Delta Acc_l(\mathcal{Q}) \propto s_l \cdot e^{-\alpha(q_l/B)}, \tag{1}$$

where B represents the original (full-precision) bit-width, $\mathcal{Q} = \{(1, q_1), ..., (L, q_L)\}$ refers to a quantization bit-width allocation strategy, α is a positive constant decay rate, and e is Euler's number. We define the accuracy degradation by normalizing the exponential decay fluctuation induced by bit-width q_l as Formula 2.

$$\Delta Acc_l(\mathcal{Q}) = \frac{Acc_o \cdot s_l}{\sum_{i=1}^{L} s_i} \cdot \frac{(e^{-\alpha(q_l/B)} - e^{-\alpha})}{(1 - e^{-\alpha})}, \tag{2}$$

where Acc_o represents the original accuracy of LLM M without quantization. Afterwards, we can calculate the total accuracy degradation brought by all layers as defined in Formula 3.

$$\Delta Acc(\mathcal{Q}) = \frac{Acc_o}{\sum_{i=1}^{L} s_i} \sum_{l=1}^{L} s_l \cdot \frac{(e^{-\alpha(q_l/B)} - e^{-\alpha})}{(1 - e^{-\alpha})}. \tag{3}$$

The problem we address in this work is how to find a bit-width allocation strategy $\mathcal{Q}^*$ so as to minimize the accuracy degradation while achieving the compression rate target as formulated in Formula 4.

$$\begin{aligned} &\mathcal{Q}^* = \underset{\mathcal{Q}}{\operatorname{argmin}}\, \Delta Acc(\mathcal{Q}), \\ &\text{s.t.} \begin{cases} \forall \quad (l, q_l) \in \mathcal{Q}^*, q_l \in \mathcal{B}, \\ \sum_{l=1}^{L} p_l q_l \leq R \cdot B \cdot \sum_{l=1}^{L} p_l, \end{cases} \end{aligned} \tag{4}$$

where p_l is the number of parameters in Layer l, and R is the target compression ratio. This problem definition bridges the accuracy and memory requirement by optimizing bit-width allocation strategy $\mathcal{Q}$, where the objective function $\Delta Acc(\mathcal{Q})$ explicitly represents the accuracy degradation, while the compression rate target guarantees hardware compatibility in terms of memory requirement. This problem is complicated due to severe combinatorial explosion. The search space grows exponentially as $\mathcal{O}(|\mathcal{B}|^L)$. For instance, the search space reaches $3^{224} \approx 10^{106}$ for LLaMA3-8B (224 layers from 32 blocks × 7 layers) with only 3 bit-width options for each layer, rendering exhaustive search computationally prohibitive even for offline quantization.

4 FAMPWQ Methodology

In this section, we detail the methodology of FAMPWQ. We first describe the Fisher information-based sensitivity measurement for each layer. Then, we present the adaptive bit-width allocation method that minimizes accuracy degradation while achieving the compression rate target.

4.1 Fisher Information-based Sensitivity

Fisher information quantifies the amount of information that observable data carries about unknown model parameters. We leverage this property to measure the sensitivity of each layer to quantization noise: a layer whose Fisher information changes substantially under perturbation is highly sensitive. Specifically, we compute the Fisher information of each layer with its original weights and with perturbed weights, and use the difference as the sensitivity measure.

In order to quantify layer sensitivity, we inject a perturbation $\delta\theta_l$ into the parameters of each Layer l and measure the resulting shift in the FIM. While a generic perturbation, e.g., uniform or magnitude-proportional noise, only reflects general parameter importance, we need to capture the *specific* noise incurred by b-bit quantization. We therefore exploit Formula 5 to generate the perturbation.

$$\delta\theta_l = Q_b(\theta_l) - \theta_l, \tag{5}$$

where $Q_b(\cdot)$ denotes the b-bit quantize, i.e., the dequantization operator. This definition ensures $\theta_l + \delta\theta_l = Q_b(\theta_l)$, so $\delta\theta_l$ is exactly the additive rounding perturbation introduced by b-bit quantization rather than an arbitrary direction.

To simplify sensitivity evaluation, we adopt a layer-independent strategy. We add the perturbation to only one target layer θ_l, while all other layers remain at their original parameters. Then, we can get the layer after adding perturbation noise as defined in Formula 6.

$$\theta_l^{\text{pert}} = \theta_l + \delta\theta_l, \tag{6}$$

where θ_l refers to the original parameters of Layer l and θ_l^{pert} is the parameters with perturbation.

We can calculate the gradients $\nabla_{\theta_l}\ell(x|\theta)$, which denotes the first-order derivative of the LLM. Then, we can derive the empirical Fisher Information Matrix (FIM) as defined in Formula 7.

$$\hat{F}(\theta_l) = \frac{1}{N} \sum_{n=1}^{N} (\nabla_{\theta_l}\ell(x_n|\theta) \nabla_{\theta_l}\ell(x_n|\theta)^\top). \tag{7}$$

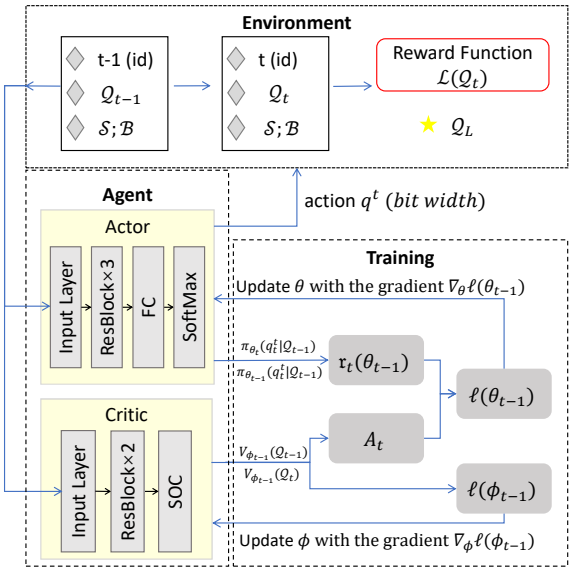


Figure 4: PPO-based adaptive bit-width allocation method. $\ell(\theta_{t-1}) = \mathbb{E}\left[\min\left(\mathfrak{r}_t(\theta_{t-1})A_t, \mathfrak{c}_t(\theta_{t-1})A_t\right)\right]$ and $\ell(\phi) = A_t^2$, which are exploited in Formulas 15 and 18. $\mathcal{S} = \{s_1, ..., s_L\}$ represents the sensitivity.

We use the diagonal vector of FIM denoted by $\mathcal{F}(\theta_l)$ to approximate the original FIM (Liu et al., 2024) as defined in Formula 8.

$$\mathcal{F}(\theta_l) = \text{diag}(I_{|\nabla_{\theta_l}\ell(x|\theta)|} \odot \hat{F}(\theta_l)). \tag{8}$$

Since the diagonal vector of FIM only depends on the diagonal elements of the original matrix, we can simplify the calculation of Formulas 7 and 8. We can calculate each element $f_i(\theta_l) \in \mathcal{F}(\theta_l)$ with i representing the i-th element in $\mathcal{F}(\theta_l)$ as defined in Formula 9.

$$f_i(\theta_l) = \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{J}(\nabla_{\theta_l}\ell(x_n|\theta))^2_{(i,j)}, \tag{9}$$

where $\nabla_{\theta_l}\ell(x_n|\theta)_{(i,j)}$ represents the element with the index $(i,\ j)$ in $\nabla_{\theta_l}\ell(x_n|\theta)$ and J represents the number of elements in i-th row of $\nabla_{\theta_l}\ell(x_n|\theta)$. Similarly, we can calculate the diagonal vector of FIM for the parameters with added perturbation noise denoted by $\mathcal{F}(\theta_l^{\text{pert}})$. See calculation details of FIM in Appendix A.2.2.

Finally, we take the FIM variation to indicate the sensitivity of a layer, which is defined in Formula 10.

$$s_l = \frac{\left\|\mathcal{F}(\theta_l^{\text{pert}}) - \mathcal{F}(\theta_l)\right\|_2}{\|\mathcal{F}(\theta_l)\|_2}, \tag{10}$$

where s_l represents the sensitivity of Layer l, $||\cdot||_2$ is the Euclidean norm over the diagonal FIM vector. The resulting scalar sensitivity value s_l is used in Formula 3.

4.2 Adaptive Bit-width Allocation

In this section, we present an RL-based adaptive bit-width allocation method. Since the combinatorial problem defined in Formula 4 is intractable, we transform it into a single loss function minimization problem as defined in Formula 11.

$$\min \mathcal{L}(\mathcal{Q}) = \Delta Acc(\mathcal{Q}) + P(\psi(\mathcal{Q})) \cdot ||\psi(\mathcal{Q})||^2, \tag{11}$$

where $P(\psi(\mathcal{Q}))$ is a penalty parameter and $\psi(\mathcal{Q})$ is the storage loss compared with the compression rate target R as defined in Formula 12.

$$\psi(\mathcal{Q}) = \sum_{i=1}^{L} p_i \cdot q_i - R \cdot B \cdot \sum_{i=1}^{L} p_i. \tag{12}$$

In addition, $P(\psi(\mathcal{Q}))$ depends on $\psi(\mathcal{Q})$ as defined in Formula 13.

$$P(\psi(\mathcal{Q})) = \begin{cases} P_{\text{penalty}} > 0, \text{if } \psi(\mathcal{Q}) > 0, \\ P_{\text{reward}} \leq 0, \text{otherwise}, \end{cases} \tag{13}$$

where P_{penalty} is the penalty when the quantization does not achieve the targeted compression rate and P_{reward} refers to the rewards brought by the extra quantization compression. Both P_{penalty} and P_{reward} are constant values.

While RL is an effective approach for complex combinatorial optimization problems (Cappart et al., 2021), we adopt Proximal Policy Optimization (PPO) (Schulman et al., 2017) for bit-width allocation. As shown in Figure 4, the architecture consists of an agent and the environment. The agent generates the bit-width allocation strategy while the environment provides feedback through a reward function. The agent consists of two modules: the actor generates bit-width allocation strategies and the critic guides policy optimization. Both the actor and critic modules are implemented as lightweight residual networks (see architecture details in Appendix A.6). During the quantization phase, both modules are first trained, after which the actor generates the final allocation strategy. The actor takes the layer ID, the current allocation $\mathcal{Q}$, per-layer parameter counts, per-layer sensitivity scores, and the candidate bit-widths $\mathcal{B}$ as input. It then outputs the bit-width for the corresponding layer. The critic receives the same inputs and produces a scalar value estimate to guide policy optimization.

4.2.1 Training Process

The training process contains multiple epochs, each of which consists of L steps. At the beginning

of the training, the bit-width allocation strategy $\mathcal{Q}$ is initialized to the highest selectable bit-width in each layer, i.e., $\forall(l, q_l) \in \mathcal{Q}_0, q_l \in \mathcal{B}$, which is exploited for the first epoch. For each epoch, at Step t, we denote the parameters of the actor network by θ_t and that of the critic network by ϕ_t. We denote the bit-width for Layer l at Step t by q_l^t. Then, the actor network generates the bit-width q_t^t for Layer t, and update $\mathcal{Q}_{t-1}$ to $\mathcal{Q}_t$ by replacing q_t^{t-1} by q_t^t. In addition, we denote the scalar value of the critic network by $V_{\phi_t}(\mathcal{Q}_t)$. Then, we compute the Temporal-Difference (TD) advantage (Rowland et al., 2024) A_t at Step t as defined in Formula 14.

$$A_t = \mathcal{L}(\mathcal{Q}_{t-1}) + \gamma V_{\phi_{t-1}}(\mathcal{Q}_t) - V_{\phi_{t-1}}(\mathcal{Q}_{t-1}), \tag{14}$$

where $\mathcal{L}(\mathcal{Q}_{t-1})$ is defined in Formula 11, $\gamma \in (0, 1)$ is a discount factor that controls the trade-off between immediate and future rewards. The actor network is updated by minimizing the clipped surrogate objective (Schulman et al., 2017) while ensuring stable policy improvements as defined in Formula 15.

$$\theta_t \leftarrow \theta_{t-1} - \eta_\theta \nabla_{\theta_{t-1}} \mathbb{E}\big[\min\big(\mathfrak{r}_t(\theta_{t-1})A_t, \mathfrak{c}_t(\theta_{t-1})A_t\big)\big] \tag{15}$$

where $\mathbb{E}[\cdot]$ corresponds to the empirical average, η_θ is a constant learning rate of the actor network, $\mathfrak{c}_t(\theta_{t-1})$ refers to a clip reward defined in Formula 16:

$$\mathfrak{c}_t(\theta_{t-1}) = \text{clip}(\mathfrak{r}_t(\theta_{t-1}), 1-\epsilon, 1+\epsilon), \tag{16}$$

where ϵ is a small constant controlling the policy update range, $\mathfrak{r}_t(\theta)$ represents the policy-dependent reward as defined in Formula 17.

$$\mathfrak{r}_t(\theta_{t-1}) = \frac{\pi_{\theta_t}(q_t^t|\mathcal{Q}_{t-1})}{\pi_{\theta_{t-1}}(q_t^t|\mathcal{Q}_{t-1})}, \tag{17}$$

where $\pi_{\theta_t}(q_t^t|\mathcal{Q}_{t-1})$ represents the probability to generate q_t^t with the actor network θ_t and the allocation strategy $\mathcal{Q}_{t-1}$.

Simultaneously, the critic network is updated to minimize the squared TD advantage:

$$\phi_t \leftarrow \phi_{t-1} - \eta_\phi \nabla_{\phi_{t-1}} A_t^2, \tag{18}$$

where η_ϕ is a constant learning rate of the critic network. See training details in Appendix A.6.

Table 1: PPL ↓ comparison on LLaMA-7B and LLaMA-13B with 4-bit and 3-bit average quantization. **Bold** indicates the lowest PPL and underlined indicates the second lowest.

Model		LLaMA-7B				LLaMA-13B			
Method	Avg bit	Wiki2	PTB	C4	Avg PPL	Wiki2	PTB	C4	Avg PPL
FP16	16	5.68	10.11	7.34	7.71	5.09	9.08	6.80	6.99
RTN	4	6.29	11.23	8.12	8.55	5.53	9.77	7.23	7.51
GPTQ	4	6.01	10.59	7.74	8.11	5.30	9.37	6.96	7.21
GPTQv2	4	5.89	10.46	7.58	7.98	5.24	9.31	6.93	7.16
OmniQuant	4	5.86	10.42	**7.53**	7.94	5.21	9.21	6.91	7.11
AMQ	4	5.88	10.43	7.62	7.98	5.25	9.40	6.97	7.20
OWQ	4	5.96	10.67	7.67	8.10	5.25	9.32	6.97	7.18
AWQ	4	5.83	10.42	**7.53**	7.93	5.20	9.20	6.90	7.10
FAMPWQ	4	**5.81**	**10.34**	7.54	**7.90**	**5.19**	**9.18**	**6.88**	**7.08**
RTN	3	25.58	89.45	30.81	48.61	11.40	26.36	14.38	17.38
GPTQ	3	7.90	14.72	10.23	10.95	5.82	8.62	6.78	7.07
GPTQv2	3	7.31	12.64	8.97	9.64	5.68	8.45	6.70	6.94
OmniQuant	3	6.49	11.43	8.19	8.70	5.48	8.21	6.35	6.68
AMQ	3	6.83	12.66	8.72	9.40	5.68	8.51	6.54	6.91
OWQ	3	6.65	12.47	8.62	9.25	5.66	10.02	7.43	7.70
AWQ	3	6.53	11.83	8.58	8.98	5.52	8.31	6.42	6.75
FAMPWQ	3	**6.35**	**11.33**	**8.07**	**8.58**	**5.40**	**8.20**	**6.25**	**6.62**

4.2.2 Inference Process

The inference process consists of L steps. Similar to the training process, the bit-width allocation strategy $\mathcal{Q}_0$ is initialized to the highest selectable bit-width. At each step t, the actor module generates a bit-width q_t^t for Layer t and updates $\mathcal{Q}_{t-1}$ by replacing q_t^{t-1} with q_t^t. After L steps, $\mathcal{Q}_L$ contains the generated bit-widths and is used as the adaptive allocation strategy to quantize the LLM.

5 Experiments

In this section, we present the experimental results. We first describe the experimental setup and then compare FAMPWQ with 7 baseline approaches across 7 models and 5 benchmarks. We implement FAMPWQ in Python while maintaining compatibility with existing quantization backends such as GPTQ, AWQ, and OmniQuant.

5.1 Experimental Setup

We take 6 state-of-the-art PTQ approaches, i.e., GPTQ (Frantar et al., 2023), GPTQv2 (Li et al., 2025), AMQ (Lee et al., 2025), OmniQuant (Shao et al., 2024), OWQ (Lee et al., 2024), and AWQ (Lin et al., 2024) as baseline approaches. We take a simple quantization approach by mapping floating-point values to their nearest discrete levels, which is denoted by Round-To-Nearest (RTN), as a baseline approach. We evaluate on 7 LLMs, i.e., LLaMA-7B, LLaMA-13B (Touvron et al., 2023a), LLaMA2-7B-chat, LLaMA2-13B-chat (Touvron et al., 2023b), Qwen2.5-7B, Qwen2.5-14B (Yang et al., 2024) and Mistral-7B-v0.1 (Jiang et al., 2023). In addition, we utilize 5 benchmarks: Wikitext-2 (Wiki2) (Merity et al., 2016), Penn Treebank (PTB) (Marcus et al., 1994), C4 (Raffel et al., 2020), lm-evaluation-harness (Gao et al., 2023), and Vicuna (Chiang et al., 2023), to evaluate the

Table 2: Zero-shot reasoning accuracy (↑) of quantized Qwen2.5-7B under 3-bit quantization.

Qwen2.5-7B							
Method	**Avg bit**	**BoolQ**	**ARC-E**	**ARC-C**	**HellaSwag**	**WinoGrande**	**Avg acc**
FP16 (Ref)	16	0.8471	0.8047	0.4778	0.6003	0.7301	0.6920
RTN	3	0.6425	0.5851	0.3846	0.5075	0.5983	0.5436
GPTQ	3	0.6845	0.5912	0.3756	0.4867	0.5891	0.5454
OWQ	3	0.7156	0.6083	0.3821	0.5074	0.6022	0.5631
GPTQv2	3	0.7324	0.6245	0.3878	0.5192	0.6114	0.5751
OmniQuant	3	0.7634	0.6572	0.3956	0.5348	0.6231	0.5948
AWQ	3	0.7612	0.6588	0.3941	0.5456	0.6245	0.5968
FAMPWQ (Ours)	3	**0.7854**	**0.6821**	**0.4032**	**0.5567**	**0.6341**	**0.6123**

PPL, the accuracy, and the LLM-as-a-judge comparison of diverse quantization approaches.

The hyperparameters used in our experiments are shown in Table A.3. All experiments are conducted on NVIDIA 4090 GPUs using PyTorch 2.0 (Paszke et al., 2019) with HuggingFace integration (Wolf et al., 2019), to ensure the consistent comparison with baseline approaches. For fair evaluation, we maintain identical experimental settings across all quantization approaches, including calibration data (128 randomly sampled sequences from C4).

5.2 Experimental Results

In this section, we present the experimental results in terms of the PPL with 3 benchmarks, the accuracy on zero-shot tasks, and the evaluation of FAMPWQ based on LLM-as-a-judge comparison.

5.2.1 Perplexity Evaluation

As shown in Table 1, FAMPWQ consistently achieves excellent performance in terms of PPL across 2 LLMs and 3 benchmarks when performing 4-bit and 3-bit quantization on average. To mitigate the influence of evaluation randomness, all PPL results reported in this section are averaged over 3 independent runs with different random seeds for calibration sampling, and we report the mean value across runs. With LLaMA-7B and 4 average bits, FAMPWQ attains the PPLs of 5.81 on WikiText-2 and 10.34 on PTB, outperforming the strongest baseline (AWQ) by 0.02 and 0.08, respectively. While FAMPWQ corresponds to slightly higher (0.01) PPL compared with AWQ and OmniQuant on C4, it still significantly outperforms other baseline approaches (from 0.13 to 3.39). We observe similar results with LLaMA-13B at 4 bits. Under 3-bit quantization, the advantage of FAMPWQ becomes substantially larger: FAMPWQ outperforms all baselines on both models, reducing average PPL by up to 2.37 over GPTQ and 0.40 over AWQ on LLaMA-7B. In addition, FAMPWQ outperforms baseline approaches (from 0.04 to 2.86 in average PPL) on Qwen2.5-7B, Qwen2.5-14B, and Mistral-7B-v0.1 (see Table A.9 in Appendix A.7.5).

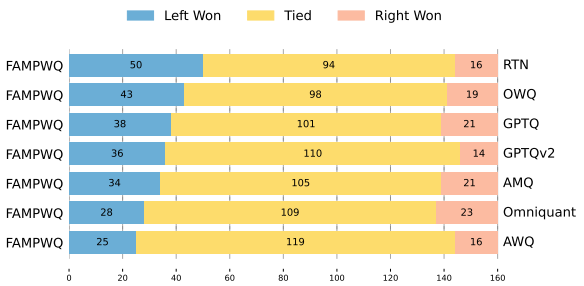


Figure 5: LLM-as-a-judge comparison based on GPT-3.5-turbo with 4-bit quantized LLaMA2-13B-chat.

5.2.2 Zero-shot Reasoning Evaluation

As shown in Table 2, we evaluate zero-shot reasoning using lm-evaluation-harness (see visualized Figure A.3 in Appendix for the 3-bit results of Qwen2.5-7B). Compared with RTN and GPTQ, which it outperforms by 6.87% and 6.69% in average accuracy respectively, FAMPWQ avoids the severe shrinkage of the radar profile, indicating better preservation of general reasoning ability under aggressive compression. Compared with stronger quantized baselines such as AMQ, AWQ and OmniQuant, FAMPWQ expands the outer boundary on most tasks and remains closer to the FP16 reference, especially on BoolQ, ARC-E, and WinoGrande, demonstrating that layer-adaptive bit-width allocation is critical for preserving model quality under aggressive compression.

5.2.3 LLM-as-a-judge Evaluation

To comprehensively evaluate the performance of FAMPWQ, we compare FAMPWQ with baseline approaches based on the quantized versions of the instruction-tuned LLaMA2-13B-chat model exploiting the Vicuna benchmark (Chiang et al., 2023). We use GPT-3.5-turbo (OpenAI, 2025) as a judge across 80 diverse questions. We mitigate position bias through bidirectional comparison, which results in 160 trials per comparison. As shown in Figure 5, FAMPWQ achieves substantially higher win rates than all baseline approaches (76% against RTN, 69% against OWQ, 64% against GPTQ, 72% against GPTQv2, 61% against AWQ and 54% against OmniQuant), where the win rate excludes tie samples. A two-sided binomial test on the head-to-head trials confirms that the comparison against the strong AWQ baseline is statistically significant ($p < 0.05$), reducing the risk that the observed judge preference is caused by evaluation noise. Additional 3-bit Vicuna-Bench

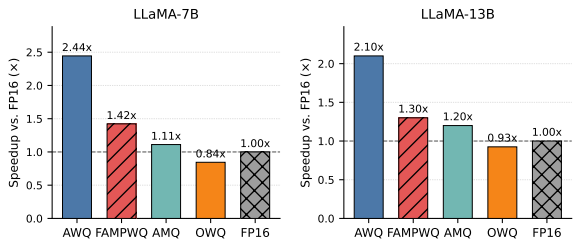


Figure 6: Inference speedup over FP16 on NVIDIA 4090 for 3-bit average quantization.

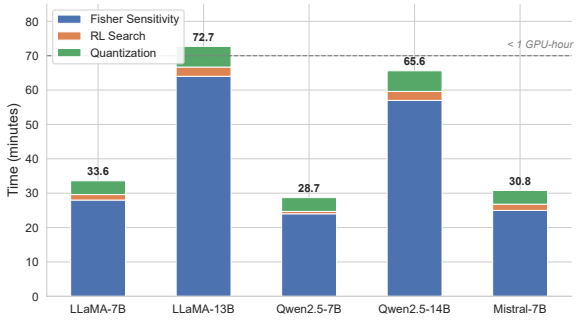


Figure 7: Preprocessing time breakdown of FAMPWQ across 5 models. Total cost remains below 1 GPU-hour even for 14B-scale models. Fisher sensitivity computation dominates, while RL search takes only 1–3 minutes.

results are reported in Appendix A.7.11.

5.2.4 Inference acceleration

As shown in Figure 6, FAMPWQ delivers a clear throughput advantage over FP16 (up to 42%) and both intra-layer (OWQ) (up to 69%) and activation-guided (AMQ) (up to 28%) mixed-precision baselines, while remaining slower than uniform low-bit AWQ due to heterogeneous kernel scheduling. AWQ retains the highest absolute throughput (2.44× on 7B, 2.10× on 13B) by exploiting uniform 4-bit kernels, while FAMPWQ delivers consistently higher accuracy at the same or lower average bit-width as shown in Table 1.

5.2.5 Computational Cost

The preprocessing overhead of FAMPWQ consists of three components: Fisher sensitivity computation, RL-based bit-width search, and the quantization itself. As shown in Figure 7, the total preprocessing time remains below 1 GPU-hour for all models tested, including 14B-scale models. Fisher sensitivity computation dominates the cost (24–64 minutes) and scales with model size. The RL search is lightweight (<5 minutes on a single GPU), as it operates on a proxy model rather than performing full quantization at each step. The entire preprocessing is a one-time offline cost, amortized across all subsequent inference.

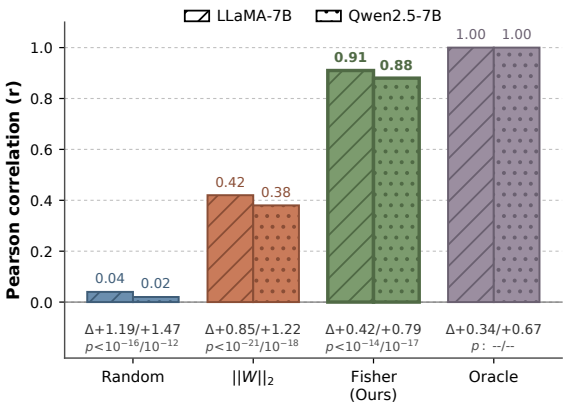


Figure 8: Pearson correlation (r) between sensitivity metrics and actual quantization degradation (Oracle).

Table 3: PPL ↓ of LLaMA-7B and Qwen2.5-7B on WikiText-2 with diverse perturbation strategies. **Bold** indicates the lowest PPL. $x\%\delta_1\theta$ refers to the δ_1 strategy with $\epsilon = x\%$. $x\%\delta_2\theta$ denotes the δ_2 strategy with $\beta = x\%$. $x$bit$\delta_3\theta$ represents the δ_3 strategy with $b = x$.

Perturbation Type	$1\%\delta_1\theta$	$10\%\delta_1\theta$	$20\%\delta_1\theta$	$1\%\delta_2\theta$	$10\%\delta_2\theta$	$20\%\delta_2\theta$	4bit$\delta_3\theta$	8bit$\delta_3\theta$
LLaMA-7B	6.61	6.57	6.68	6.62	6.61	6.72	**6.49**	6.53
Qwen2.5-7B	8.42	8.37	8.83	8.74	8.36	8.40	**8.27**	8.33

5.3 Ablation Study

In this section, we analyze the impact of diverse sensitivity measurement methods and the comparison of diverse bit-width allocation methods.

5.3.1 Artificial Perturbation

We compare $\delta\theta_l$ in Eq. 5 against two generic alternatives: magnitude-proportional uniform noise (δ_1) and Bernoulli-masked weight-proportional noise (δ_2) (see Appendix A.4 for details). As shown in Table 3, the quantization perturbation form at b=4 yields the lowest PPL on both models, beating δ_1 by up to 0.56 and δ_2 by up to 0.23. Only δ_3 matches the actual b-bit rounding perturbation in both direction and magnitude; δ_1 and δ_2 are agnostic to the target bit-width and therefore reflect only generic parameter importance.

5.3.2 Bit-width Allocation Strategy

We compare our PPO-based method with four alternative methods: greedy search, Bayesian optimization, simulated annealing, and a genetic algorithm. As shown in Table 4, the RL-based adaptive allocation strategy achieves substantially lower average PPL than these alternatives (up to 1.50 lower than Greedy, 1.33 lower than Bayesian optimization, 1.90 lower than simulated annealing, and 0.45 lower than the genetic algorithm), revealing the superb performance of our allocation method.

Table 4: PPL ↓ with different bit-width allocation methods. **Bold** indicates the lowest PPL.

Model	LLaMA-7B					Qwen2.5-7B			
Strategy	Avg bit	Wiki2	PTB	C4	Avg PPL	Wiki2	PTB	C4	Avg PPL
FP16	16	5.68	10.11	7.34	7.71	6.84	12.79	11.88	10.50
Greedy	3	6.90	12.12	9.22	9.41	8.81	16.05	14.16	13.67
Bayesian	3	7.29	12.57	9.86	9.91	8.37	15.24	13.78	12.46
Annealing	3	7.54	13.85	10.06	10.48	8.86	16.59	14.65	13.37
Genetic	3	6.59	11.71	8.79	9.03	8.27	15.26	13.64	12.39
RL (Ours)	3	**6.35**	**11.33**	**8.07**	**8.58**	**8.08**	**14.99**	**13.45**	**12.17**

5.3.3 Sensitivity Metric Comparison

We compare the FIM-based sensitivity metric against random allocation, weight magnitude ($\|W\|_2$), and Oracle sensitivity (actual per-layer PPL increase). As shown in Figure 8, our FIM-based sensitivity metric achieves a Pearson correlation of r=0.91, $p < 10^{-14}$ on LLaMA-7B and r=0.88, $p < 10^{-17}$ on Qwen2.5-7B with Oracle sensitivity.This significant correlation directly leads to better quantization performance: at 3.5-bit average, the FIM-based metric limits ΔPPL to +0.42, while weight magnitude yields +0.85 and random allocation yields +1.19 (see detals in Appendix Table A.14).

5.3.4 α Sensitivity Analysis

As shown in Figure 9, the decay rate parameter α exhibits a broad optimal region. For LLaMA-7B, the optimal α=18 yields a PPL of 6.35, while any $\alpha \in [15, 25]$ produces PPL within 0.09 of the optimum. For Qwen2.5-7B, α=20 is optimal (PPL 8.27), with <0.06 variation across the robust zone. This robustness to α simplifies hyperparameter selection and confirms that the exponential decay model in Formula 1 is a stable approximation.

6 Conclusion

In this work, we propose a novel Fisher information-based Adaptive Mixed Precision Weight Quantization approach, i.e., FAMPWQ. FAMPWQ introduces a novel perturbation-based Fisher Information metric to capture layer-wise quantization-specific sensitivity. In addition, FAMPWQ couples the metric with a new PPO-based allocation method to efficiently generate an adaptive bit-width allocation strategy with superb performance. Extensive experiments on 7 models and 5 benchmarks demonstrate that FAMPWQ outperforms 7 baselines in PPL (up to 3.39 smaller), accuracy (up to 6.87% higher), and LLM-as-a-judge comparison (up to 76% win rate), with particularly strong advantages at the 3-bit compression frontier.

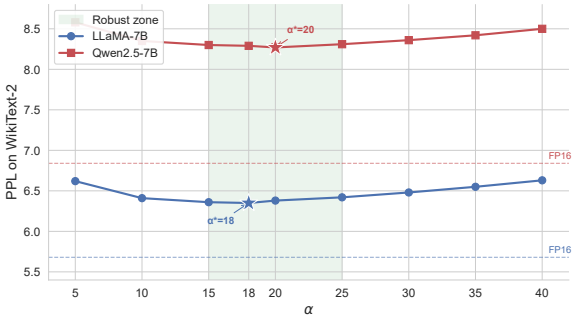


Figure 9: PPL on WikiText-2 as a function of decay rate α for LLaMA-7B and Qwen2.5-7B under 3-bit average quantization. Stars mark the optimal α for each model. The shaded green region indicates the robust zone ($\alpha \in [15, 25]$) where PPL variation is minimal (<0.3). Dashed lines show FP16 baselines.

Limitations

Several limitations remain for FAMPWQ. First, mixed-precision quantization can reduce inference throughput while heterogeneous bit-widths are less compatible with optimized uniform-precision kernels. Second, FAMPWQ currently targets weight-only quantization (WxA16), leaving joint weight-activation quantization to future work. Third, our experiments focus on dense Transformer models, and effectiveness on Mixture-of-Experts architectures remains untested. Finally, Fisher sensitivity estimation is a one-time offline cost but remains the dominant preprocessing component, motivating lighter sensitivity proxies.

Ethical Considerations

We focus on the quantization methods for LLMs in this paper. Our work does not involve the collection of new data, human subjects, or user-generated content. All experiments are conducted on publicly available datasets. We do not identify new ethical concerns arising from this work. The proposed method is model-agnostic and does not introduce new risks related to bias, privacy, or fairness beyond the underlying models and datasets. Nevertheless, we acknowledge that compressed LLMs may be subject to potential risk if deployed without appropriate safeguards. We exploited LLMs to polish the expression of this paper.

Acknowledgements

This work was partially (for Juncheng Jia) supported by the Priority Academic Program Development of Jiangsu Higher Education Institutions, Suzhou Frontier Science and Technology Program (Project SYG202310).

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Shun-ichi Amari. 1998. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. *Noise reduction in speech processing*, volume 2. Springer Science & Business Media.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, and Girish Sastry. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Quentin Cappart, Thierry Moisan, Louis-Martin Rousseau, Isabeau Prémont-Schwarz, and Andre A Cire. 2021. Combining reinforcement learning and constraint programming for combinatorial optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3677–3687.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

DeepSeek-AI. 2026. Deepseek-v4 technical report. Technical report.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. 2019. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 293–302, Seoul, Korea (South). IEEE.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Elias Frantar and Dan Alistarh. 2022. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers. *Preprint*, arXiv:2210.17323.

Garrett Gagnon, Srikanth Malla, Yangwook Kang, and Liu Liu. 2025. Bitweaver: Read-time truncation in memory. In *Proceedings of the 39th ACM International Conference on Supercomputing (ICS '25)*, pages 13–25. Association for Computing Machinery.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2023. A framework for few-shot language model evaluation.

Zhuocheng Gong, Jiahao Liu, Jingang Wang, Xunliang Cai, Dongyan Zhao, and Rui Yan. 2024. What makes quantization for large language model hard? an empirical study from the lens of perturbation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):18082–18089.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. 2024. Squeezellm: Dense-and-sparse quantization. *Preprint*, arXiv:2306.07629.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. 2019. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems (NeurIPS)*, 32.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2024. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(12):13355–13364.

Sangjun Lee, Seung-taek Woo, Jun-gyu Jin, Changhun Lee, and Eunhyeok Park. 2025. Amq: Enabling automl for mixed-precision weight-only quantization of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35520–35538.

Yuhang Li, Ruokai Yin, Donghyun Lee, Shiting Xiao, and Priyadarshini Panda. 2025. Gptaq: Efficient finetuning-free quantization for asymmetric calibration. *Preprint*, arXiv:2504.02692.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Ji Liu, Jiaxiang Ren, Ruoming Jin, Zijie Zhang, Yang Zhou, Patrick Valduriez, and Dejing Dou. 2024. Fisher information-based efficient curriculum federated learning with large language models. *Preprint*, arXiv:2410.00131.

Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. 2021. Group fisher pruning for practical network compression. In *Proceedings of the 38th International Conference on Machine Learning*, pages 7021–7032. PMLR.

Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul P. P. P. Grasman, and Eric-Jan Wagenmakers. 2017. A tutorial on fisher information. *Journal of Mathematical Psychology*, 80:40–55.

Srikanth Malla, Joon Hee Choi, and Chiho Choi. 2024. Copal: Continual pruning in large language generative models. In *Forty-first International Conference on Machine Learning*.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

OpenAI. 2025. GPT-3.5 Turbo. `https://platform.openai.com/docs/models/gpt-3.5-turbo`, accessed 2025-08.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *Preprint*, arXiv:1909.01066.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Mark Rowland, Rémi Munos, Mohammad Gheshlaghi Azar, Yunhao Tang, Georg Ostrovski, Anna Harutyunyan, Karl Tuyls, Marc G Bellemare, and Will Dabney. 2024. An analysis of quantile temporal-difference learning. *Journal of Machine Learning Research*, 25(163):1–47.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. Omniquant: Omnidirectionally calibrated quantization for large language models. In *ICLR*.

Sidak Pal Singh and Dan Alistarh. 2020. Woodfisher: Efficient second-order approximation for neural network compression. In *Advances in Neural Information Processing Systems*, volume 33, pages 18098–18109. Curran Associates, Inc.

Qwen Team. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Lacroix, and 1 others. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ming Tu, Visar Berisha, Martin Woolf, Jae-sun Seo, and Yu Cao. 2016. Ranking the parameters of deep neural networks using the fisher information. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2647–2651.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8604–8612.

Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. 2024a. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748*.

Zhaoyang Wang, Xingyu Liu, Haotong Zhang, and Wenqi Shao. 2024b. Rl-ptq: Reinforcement learning for post-training quantization. *arXiv preprint arXiv:2405.17508*.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 38087–38099. PMLR.

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Chao Zeng, Songwei Liu, Yusheng Xie, Hong Liu, Xiaojian Wang, Miao Wei, Shu Yang, Fangmin Chen, and Xing Mei. 2025. Abq-llm: Arbitrary-bit quantized inference acceleration for large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(21):22299–22307.

Zhen Zheng, Xiaonan Song, and Chuanjie Liu. 2024. Mixllm: Llm quantization with global mixed-precision between output-features and highly-efficient system design. *Preprint*, arXiv:2412.14590.

Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. 2018. Adaptive quantization for deep neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

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

A Appendix

A.1 Explanation of Notations

The meanings of the notations in this paper are summarized in Table A.1.

A.2 Preliminary

In this section, we present the quantization preliminary and introduce Fisher information calculation.

A.2.1 Quantization Preliminary

The quantization process can be classified into uniform and non-uniform (Wang et al., 2024a). The uniform quantization uses uniform and finite intervals (e.g., 2^b intervals for b-bit integer) to represent the original values. In contrast, non-uniform quantization utilizes non-uniformly spaced intervals, and the length of intervals can vary. Given a weight tensor $\mathbf{W}$ in a LLM, the quantization and de-quantization process can be defined as Formula A.1.

$$\mathbf{W}^Q = Q(\mathbf{W}), \quad \widetilde{\mathbf{W}} = Q^{-1}(\mathbf{W}^Q), \tag{A.1}$$

where $\mathbf{W}^Q$ is the quantized tensor, and $\widetilde{\mathbf{W}}$ is the recovered tensor. The quantization function $Q(\cdot)$ of a uniform quantization approach is defined as a rounding-to-nearest operation over the scaled input calculated in Formula A.2.

$$Q_{\text{uni}}(\mathbf{W}) = \text{clip}\left(\left\lfloor \frac{\mathbf{W}}{\alpha} \right\rceil + z; 0, 2^b - 1\right), \tag{A.2}$$

where $b \in \mathbb{N}$ is the bit-width, $\alpha \in \mathbb{R}$ is the scale factor, $z \in \mathbb{N}$ is zero-point or offset value, $\lfloor \cdot \rceil$ denotes the round-to-nearest-integer operator, and $clip(x, min_{value}, max_{value})$ represents a clip function of the input x with the minimum value and the maximum value. The corresponding de-quantization function is defined in Formula A.3.

$$Q_{\text{uni}}^{-1}(\mathbf{W}) = (\mathbf{W}^Q - z) \cdot \alpha. \tag{A.3}$$

Uniform quantization can be either symmetric or asymmetric according to the sign of the mapping space. In this paper, we use the symmetric uniform quantization approach. The symmetric quantization restricts the zero-point to 0 as defined in Formula A.4.

$$Q_{\text{uni_sym}}(\mathbf{W}) = \text{clip}\left(\left\lfloor \frac{\mathbf{W}}{\alpha} \right\rceil; -2^{b-1}, 2^{b-1} - 1\right) \tag{A.4}$$

Quantization approaches optimize the global loss as defined in Formula A.5.

$$\underset{\mathbf{W}^Q}{\text{argmin}}\, E = \underset{\mathbf{W}^Q}{\text{argmin}} \left\| \mathbf{W}X - \mathbf{W}^Q X \right\|_2^2, \tag{A.5}$$

where X is the input of the corresponding layer of the LLM.

A.2.2 Fisher Information

Fisher information quantifies the amount of information that observable data carries about the unknown parameters of a probabilistic model (Ly et al., 2017; Zheng et al., 2024). Fisher information can be used to evaluate parameter importance in neural networks by quantifying how sensitive the model output is to the noises of each parameter θ. We can denote the Fisher Information Matrix (FIM) by the expectation of the outer product of score vectors (Amari, 1998) as defined in Formula A.6.

$$F(\theta_l) := \mathbb{E}_{p_\theta(y|x)} \left[\nabla_{\theta_l} \log p_{\theta_l}(y|x) \nabla_{\theta_l} \log p_{\theta_l}(y|x)^\top \right], \tag{A.6}$$

where $p_{\theta_l}(y|x)$ represents the probability density function of the inference with LLM and parameters θ_l at Layer l, $\nabla_{\theta_l} \log p_{\theta_l}(y|x)$ denotes the first-order derivative of the LLM, which is calculated via the gradient. In practice, we use the empirical FIM to approximate the expected one (Kunstner et al., 2019) as shown in Formula A.7.

$$\hat{F}(\theta_l) = \frac{1}{N} \sum_{n=1}^{N} \nabla_{\theta_l} \log p_{\theta_l}(y_n|x_n) \nabla_{\theta_l} \log p_{\theta_l}(y_n|x_n)^\top, \tag{A.7}$$

where N represents the number of samples in the validation dataset $\mathcal{D}$.

FIM captures the essential influence of parameters on the likelihood function, where larger FIM values indicate more influential parameters should be preserved for inference. FIM can be used to evaluate the importance of layer-wise parameters (Tu et al., 2016), so as to preserve accuracy while compressing LLMs (Singh and Alistarh, 2020; Liu et al., 2021). While calculating FIM is computationally expensive with large gradient matrices, we use the diagonal vector of FIM to represent the FIM (Liu et al., 2024) as defined in Formula A.8.

$$\mathcal{F}(\theta_l) = \text{diag}(I_{|\nabla_{\theta_l} \log p_{\theta_l}(y|x)|} \odot \hat{F}(\theta_l)), \tag{A.8}$$

Table A.1: Summary of main notations

Symbols	Description
$\theta; \theta_l$	LLM Parameter; parameters in Layer l.
$F(\cdot); \hat{F}(\cdot); \mathcal{F}(\cdot)$	Fisher Information Matrix (FIM); empirical FIM; the diagonal vector of FIM.
$p_{\theta_l}(y\|x)$	The probability density function of the inference with θ_l.
$\nabla_{\theta_l} \log p_{\theta_l}(y\|x)$	The first-order derivative of θ_l, which is calculated by the gradient.
$\mathcal{D}; \mathcal{L}$	Validation dataset; quantization layer set.
$\mathcal{S}; s_l$	The set of sensitivity of all the layers; the sensitivity of Layer l.
$p_l; R$	The number of parameters in Layer l; target compression ratio.
$\mathcal{B}; B$	The set of candidate bit-widths; original (full-precision) bit-width.
$\mathcal{Q}; q$	The quantization bit-width allocation strategy; bit-width allocation action.
ΔAcc	The accuracy degradation incurred by quantizing an individual layer.
α	Positive constant decay rate.
$\delta\theta_l; \theta_l^{\text{pert}}$	Noise to θ_l; perturbed layer parameters.
$\nabla_{\theta_l} \ell(x\|\theta)$	The first-order derivative of the LLM.
$\mathcal{L}(\cdot); \psi(\cdot); P$	Loss function of the bit-width allocation problem; compression target loss; penalty or reward value.
$\theta_t; \phi_t$	The parameters of the actor network at Step t; the parameters of the critic network at Step t.
$V_{\phi_t}; A_t$	The scalar value of the critic network at Step t; Temporal-Difference (TD) advantage at Step t.
$\eta_\theta; \eta_\phi$	The learning rate of the actor network; the learning rate of the critic network.
$\mathfrak{r}_t(\theta); \mathfrak{c}_t(\theta)$	Policy-dependent reward for conservative strategy adaptation; the clip reward for limiting update range.

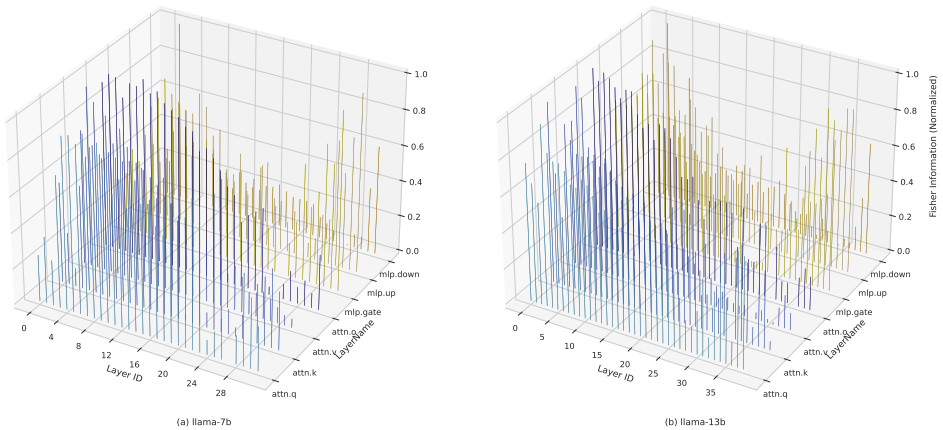


Figure A.1: Visualization of FIM for LLaMA-7B and LLaMA-13B.

where diag$(\cdot)$ represents the diagonal vector of a matrix, $I_{|\nabla_{\theta_l} \log p_{\theta_l}(y|x)|}$ is the identity matrix with the same size of the gradient matrix, and $\odot$ is element-wise multiplication.

A.3 Pearson Correlation Analysis

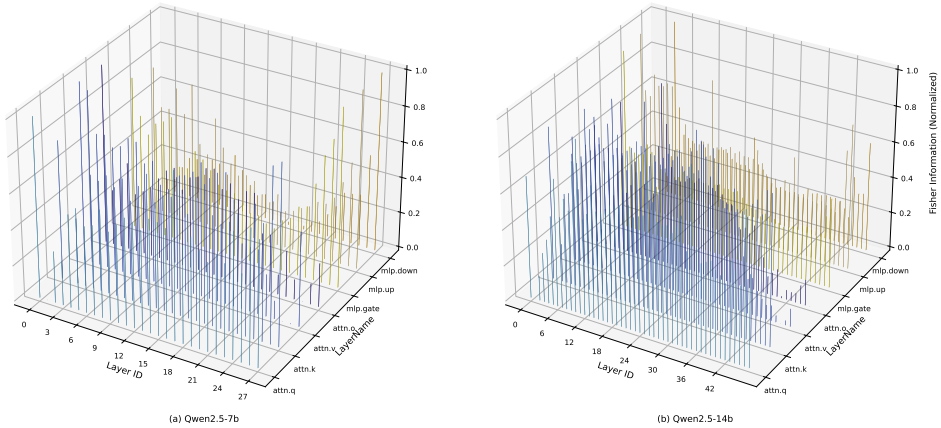
In order to verify the correlation between FIM and accuracy degradation, we calculate the Pearson correlation coefficients (Benesty et al., 2009) between the layer-wise quantization sensitivity and the increase in perplexity (PPL) with perturbations (δ_3 quantization-simulating perturbations at b=4) in each layer. As shown in Table A.2, the perturbations and the increase in PPL have a significant positive correlation with $R > 0.5$ and $P < 0.0001$, indicating that FIM is a reliable proxy for performance degradation.

Table A.2: Pearson correlation coefficients between FIM and the increase in PPL among different types of LLaMA-7B layers. R represents the related correlation coefficient and P refers to the Pearson value.

layer name	R	P
attn.q	0.884251	1.732190×10^{-43}
attn.k	0.917151	3.526569×10^{-52}
attn.v	0.761516	1.728182×10^{-25}
attn.o	0.811218	3.781759×10^{-31}
mlp.gate	0.717476	1.608838×10^{-21}
mlp.up	0.699258	4.318466×10^{-20}
mlp.down	0.612742	1.509229×10^{-14}

A.4 Perturbation Strategy Ablation

In Section 4.1 we adopt the quantization perturbation $\delta\theta_l = Q_b(\theta_l) - \theta_l$ as the canonical choice, since $\theta_l + \delta\theta_l = Q_b(\theta_l)$ and it therefore matches the exact perturbation that b-bit quantization injects. For completeness, here we provide the full mathematical definitions of two alternative perturbation forms used in the ablation in Section 5.4.1:



(a) Qwen2.5-7b (b) Qwen2.5-14b

Figure A.2: Visualization of FIM for Qwen2.5-7B and Qwen2.5-14B.

- Magnitude-proportional uniform noise: $\delta_1\theta_l = \epsilon \cdot \mu_l$, where $\mu_l = \mathbb{E}[\theta_l]$ and $\epsilon \sim \mathcal{U}(-1, 1)$. This form scales uniform noise by each layer's mean magnitude, preserving relative scale differences across layers but *not* the structure of the quantization perturbation.
- Bernoulli-masked weight-proportional noise: $\delta_2\theta_l = \beta \cdot \theta_l \odot \mathbf{m}$, with $\beta \in [0, 1]$ and $\mathbf{m} \sim \text{Bernoulli}(0.5)$. This form applies sparse, weight-proportional noise via a random mask. The fixed ratio β ensures consistent perturbation intensity but, like δ_1, is agnostic to the target bit-width b.

The corresponding empirical comparison is reported in the main text (Table 3); both alternatives are dominated by the quantization perturbation δ_3 at b=4.

A.5 Visualization of Sensitivity

As shown in Figures A.1 and A.2, the sensitivity (FIM) varies substantially across layers. Input and output-adjacent layers tend to be more sensitive because they shape low-level feature encoding and final predictions, while intermediate layers show more heterogeneous behavior. FAMPWQ exploits this diversity by preserving high-sensitivity layers and compressing low-sensitivity layers more aggressively.

A.6 Reinforcement Learning (RL)-based Network Training

As shown in Algorithm 1, the actor and critic networks are trained in multiple epochs. First, the actor and critic networks are randomly initialized (Line 1), and the initial bit-width allocation strategy $\mathcal{Q}_0$ is initialized to the maximum value for each layer (Line 2). Within each training epoch, the bit-width q_t^t is generated for each layer (Lines 4-5). The bit-width allocation strategy $\mathcal{Q}_t$ is updated with q_t^t (Line 6). Then, the loss function corresponding to $\mathcal{Q}_t$ is computed (Line 7). Afterwards, the Temporal-Difference (TD) advantage is calculated according to Formula 14 (Line 8). In addition, the clip reward and the policy-dependent reward are calculated based on Formulas 17 and 16 (Lines 9-10). Finally, the critic network ϕ_t and the actor network θ_t are updated based on Formulas 15 and 18 (Lines 11-12).

Algorithm 1 Reinforcement Learning (RL)-based Network Training

Input:
E: The number of training epochs
L: The number of layers in a LLM
Output:
θ; ϕ: Parameters of the pre-trained actor and critic network

1: θ_0;ϕ_0 ← Randomly initialize the actor and critic network
2: $\mathcal{Q}_0 \leftarrow [\max(\mathcal{B})]_L$
3: **for** Epoch $e = 1$ to E **do**
4: **for** Step $t = 1$ to L **do**
5: q_t^t ← Generate the bit-width for Layer t
6: $\mathcal{Q}_t$ ← Update $\mathcal{Q}_{t-1}$ by replacing q_t^{t-1} by q_t^t
7: $\mathcal{L}(\mathcal{Q}_{t-1})$ ← Calculate the loss according to Formula 11
8: A_t ← Calculate according to Formula 14
9: $\mathfrak{r}_t(\theta_{t-1})$ ← Calculate $\mathfrak{r}_t(\theta_{t-1})$ according to Formula 17
10: $\mathfrak{c}_t(\theta_{t-1})$ ← Calculate $\mathfrak{c}_t(\theta_{t-1})$ with $\mathfrak{r}_t(\theta_{t-1})$ and A_t according to Formula 16
11: θ_t ← Update θ_{t-1} with A_t, $\mathfrak{r}_t(\theta_{t-1})$ and $\mathfrak{c}_t(\theta_{t-1})$ according to Formula 15
12: ϕ_t ← Update ϕ_{t-1} with A_t according to Formula 18
13: **end for**
14: **end for**

Table A.3: Hyperparameter values. ϵ refers to the hyperparameter in Formula 16. For different models, recommendation α is given.

Hyperparameters	Value
$\eta_\theta; \eta_\phi$	0.0001; 0.0003
ϵ	0.2
γ	0.99
P_{penalty}	10000
P_{reward}	-1
α (LLaMA-7B)	18
α (LLaMA-13B)	15
α (LLaMA2-7B-chat)	20
α (LLaMA2-13B-chat)	20
α (Qwen2.5-7B)	20
α (Qwen2.5-14B)	20
α (Mistral-7B-v0.1)	30

Table A.4: PPL of Qwen2.5-7B and LLaMA-7B with quantization of 4-bit and 5-bit and diverse benchmarks (Wiki2, PTB, C4). "4-bit" represents that all the layers are quantized to 4-bit. Other rows report the PPL when the corresponding layers are quantized to 5-bit while other layers remain at 4-bit. **Bold** indicates the lowest PPL. underlined indicates the second-highest.

Model	**Qwen2.5-7B**			**LLaMA-7B**		
Layer name	Wiki2	PTB	C4	Wiki2	PTB	C4
4-bit	7.094	13.218	12.228	5.834	10.420	7.528
attn.q	7.094	13.205	12.222	5.831	10.414	7.520
attn.k	7.086	13.212	12.215	5.835	10.413	7.521
attn.v	7.072	13.170	12.193	5.783	10.367	7.487
attn.o	7.078	13.198	12.209	5.821	10.384	7.512
mlp.gate	7.058	13.170	12.182	5.822	10.391	7.495
mlp.up	7.047	13.125	12.167	**5.814**	**10.350**	7.493
mlp.down	**7.041**	**13.103**	**12.131**	5.816	10.383	**7.484**

Table A.5: PPL of FAMPWQ with varying average quantization bit-widths.

Model	**LLaMA-7B**			**Qwen2.5-7B**		
Avg bit	Wiki2	PTB	C4	Wiki2	PTB	C4
16	5.68	10.11	7.34	6.84	12.79	11.88
3.1	6.46	11.55	8.45	8.08	14.99	13.45
3.2	6.34	11.29	8.32	7.94	14.76	13.28
3.3	6.22	11.04	8.17	7.84	14.60	13.15
3.4	6.17	10.89	8.06	7.75	14.42	13.05
3.5	6.10	10.79	7.96	7.63	14.23	12.92
3.6	6.04	10.65	7.85	7.55	14.07	12.78
3.7	5.98	10.59	7.75	7.46	13.97	12.68
3.8	5.94	10.49	7.63	7.36	13.79	12.57
3.9	5.85	10.40	7.57	7.24	13.52	12.42
4.1	5.79	10.31	7.49	7.06	13.14	12.16
4.2	5.77	10.28	7.47	7.04	13.13	12.13
4.3	5.75	10.27	7.45	7.02	13.09	12.11
4.4	5.75	10.26	7.44	7.01	13.07	12.09
4.5	5.74	10.23	7.43	6.99	13.05	12.08

A.7 Experiment Details

In this section, we first present the hyperparameter values in experimental setup. Then, we present additional experiments, including the PPL with 5-bit quantization, varying average quantization bit-widths (from 3.1 to 4.5), the comparison of time consumption, and the quantization with 5-bits on average for LLaMA2-7B-chat, LLaMa2-13B-chat, and Mistral-7B-v0.1.

Table A.6: Quantization time with NVIDIA 4090 GPU.

Quant Method	RTN	GPTQ	GPTQv2	OWQ	OmniQuant	AWQ	FAMPWQ
LLaMA-7B	10s	369s	537s	349s	600s	129s	240s
LLaMA-13B	12s	619s	988s	598s	1125s	240s	388s

Table A.7: FAMPWQ preprocessing time on NVIDIA 4090 GPUs. Sensitivity is computed using the listed number of GPUs, while RL-based bit-width search is performed on a single GPU.

Model (#GPUs)	Sensitivity calculation (min)	Bit width optimization search (s)
LLaMA-7B (1)	28	97
LLaMA-13B (2)	64	159
LLaMA2-7B-chat (1)	28	95
LLaMA2-13B-chat (2)	62	148
Qwen2.5-7B (1)	24	43
Qwen2.5-14B (2)	57	156
Mistral-7B-v0.1 (1)	25	110

A.7.1 Calibration-Size Robustness

We examine whether the Fisher estimate and the resulting bit-width allocation are sensitive to the amount of calibration data. On LLaMA-7B, we construct nested C4 subsets containing 32, 64, 128, and 256 sequences for each of three independent seeds, and use the 256-sequence subset from the same seed as the reference. The analysis covers all 224 quantizable linear modules. To isolate calibration noise, every setting uses the same bit-conditioned proxy, candidate set $\{2, 3, 4\}$, parameter-weighted 3-bit budget, and deterministic same-budget allocator. Table A.8 reports the rank correlation, overlap among the top 10% most sensitive modules, and the fraction of module assignments that differ from the 256-sequence reference.

Even with only 32 calibration sequences, the Fisher ranking retains a Spearman correlation of 0.996 and a 97.1% top-10% overlap with the 256-sequence reference, while only 2.68% of module assignments change. At the 128-sequence setting used in the main experiments, the allocation difference decreases to 1.34%. These results show that the sensitivity ranking and budget-constrained allocation are stable with limited calibration data. This study uses 512-token calibration sequences and a deterministic proxy allocator, and does not rerun final-backend PPL for every calibration size; therefore, it establishes ranking and allocation stability rather than complete invariance of downstream quality.

A.7.2 PPL with Diverse Quantization Bit-width

As shown in Table A.4, the PPL corresponding to the quantization of 4-bit with one layer quantized to 5 bits can vary across different layers. MLP layers, especially the down-projection (mlp.down), correspond to significant PPL drop (up to 0.115 PPL drop). Among attention layers, the value projection

Table A.8: Robustness to calibration-set size on LLaMA-7B over three seeds. Each row is compared with the nested 256-sequence subset from the same seed.

C4 sequences	Spearman ↑	Top-10% overlap ↑	Allocation diff. ↓
32	0.996 ± 0.001	$97.1\% \pm 2.5\%$	$2.68\% \pm 0.45\%$
64	0.997 ± 0.001	$98.6\% \pm 2.5\%$	$2.08\% \pm 0.68\%$
128	0.998 ± 0.001	$98.6\% \pm 2.5\%$	$1.34\% \pm 1.34\%$
256	1.000 ± 0.000	$100.0\% \pm 0.0\%$	$0.00\% \pm 0.00\%$

(attn.v) corresponds to the highest sensitivity (up to 0.047). FAMPWQ aligns with this diversity and thus yields strong performance.

A.7.3 Diverse Average Quantization Bit-width

FAMPWQ can achieve varying average quantization bit-widths through adaptive layer-wise bit-width allocation strategies. As shown in Table A.5, FAMPWQ can achieve average bit-widths from 3.1 to 4.5, with correspondingly decreasing PPL.

A.7.4 Comparison of Time Consumption

As shown in Table A.6, the quantization time of FAMPWQ is comparable to several baselines and can be shorter than GPTQ (up to 37%), GPTQv2 (up to 61%), OWQ (up to 35%), and OmniQuant (up to 66%). Although RTN and AWQ can be faster than FAMPWQ by up to 97% and 46%, respectively, they may incur substantially larger performance degradation under aggressive compression.

As shown in Table A.7, the preprocessing stage of FAMPWQ, which consists of per-layer Fisher sensitivity computation and an RL-based bit-width optimization search, is conducted on NVIDIA RTX 4090 GPUs. The sensitivity computation accounts for the majority of the preprocessing cost and scales with model size, whereas the RL-based search is lightweight, requiring only tens of seconds on a single GPU. Overall, the total preprocessing time remains below 70 minutes even for 14B-scale models, demonstrating the practical efficiency and scalability of FAMPWQ.

A.7.5 Results of LLaMA2, Qwen2.5 and Mistral-7B-v0.1 models

As shown in Table A.11, FAMPWQ achieves the lowest average PPL under 4-bit quantization for both LLaMA2-7B-chat and LLaMA2-13B-chat (up to 1.09 lower than RTN, 0.28 lower than GPTQ, 0.05 lower than GPTQv2, 0.13 lower than OmniQuant, 0.16 lower than OWQ, and 0.05 lower than AWQ). The average reduction over all quantized baselines is larger on LLaMA2-7B-chat (0.28 PPL) than on LLaMA2-13B-chat (0.10 PPL), while

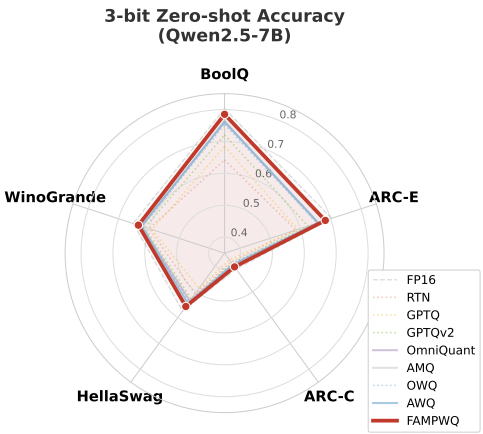


Figure A.3: Zero-shot accuracy comparison under 3-bit quantization on Qwen2.5-7B. FAMPWQ (red) consistently outperforms baselines across five reasoning tasks.

both model sizes show the best average PPL with FAMPWQ. Furthermore, FAMPWQ outperforms baseline approaches for the majority of the combinations of the benchmarks and models (up to 0.84 for Wiki2 and 1.29 for PTB in LLaMA2-7B-chat; up to 0.21 for Wiki2, 0.20 for PTB, and 0.39 for C4 in LLaMA2-13B-chat). While the PPL of FAMPWQ is slightly (0.07) higher than that of GPTQv2 with the combination of C4 and LLaMA2-7B-chat, FAMPWQ outperforms other baselines in this setting (1.12 lower than RTN, 0.30 lower than GPTQ, 0.09 lower than OmniQuant, 0.21 lower than OWQ, and 0.03 lower than AWQ).

As shown in Table A.9, FAMPWQ consistently achieves the best performance in terms of PPL across all 3 LLMs and 3 benchmarks under 4-bit quantization. With Qwen2.5-7B, FAMPWQ attains an average PPL of 10.81, which is up to 2.86 lower than that of RTN, and also lower than GPTQ, GPTQv2, OmniQuant, OWQ, and AWQ by 0.26, 0.07, 0.05, 0.19, and 0.04, respectively. On Qwen2.5-14B, FAMPWQ achieves an average PPL of 9.15, which is 1.41 lower than RTN, 0.19 lower than GPTQ, 0.06 lower than GPTQv2, and 0.02 lower than both OmniQuant and AWQ. For Mistral-7B-v0.1, FAMPWQ yields an average PPL of 8.04, representing reductions of 0.94, 0.12, 0.05, 0.07, and 0.01 compared to RTN, GPTQ, GPTQv2, OWQ, and AWQ, respectively.

As shown in Table A.10, FAMPWQ significantly outperforms baseline approaches (from 0.06% to 4.55%) in terms of average accuracy with Qwen2.5-7B and Qwen2.5-14B under 4-bit quantization. As shown in Table A.12, FAMPWQ achieves the highest average zero-shot accuracy under 4-bit quantization with both LLaMA2-7B-chat (up to 1.63% higher than RTN, 0.39% higher than GPTQ, and

Table A.9: PPL ↓ comparison on Qwen2.5, Qwen2.5-14B, and Mistral with 4-bit average quantization. "Avg PPL" denotes the average perplexity over Wiki2, PTB, and C4.

Model		**Qwen2.5-7B**				**Qwen2.5-14B**				**Mistral-7B-v0.1**			
Method	Avg bit	Wiki2	PTB	C4	Avg PPL	Wiki2	PTB	C4	Avg PPL	Wiki2	PTB	C4	Avg PPL
FP16	16	6.84	12.79	11.88	10.50	5.29	10.87	10.35	8.84	5.25	9.94	8.38	7.86
RTN	4	9.14	16.57	15.29	13.67	6.85	12.85	11.98	10.56	6.00	11.47	9.47	8.98
GPTQ	4	7.29	13.41	12.50	11.07	5.84	11.36	10.81	9.34	5.45	10.38	8.65	8.16
GPTQv2	4	7.20	13.24	12.21	10.88	5.82	11.19	10.63	9.21	5.43	10.25	8.60	8.09
OmniQuant	4	7.12	13.24	12.21	10.86	5.72	11.18	10.62	9.17	/	/	/	/
OWQ	4	7.26	13.32	12.43	11.00	5.78	11.20	10.66	9.21	5.44	10.28	8.62	8.11
AWQ	4	7.09	13.22	12.23	10.85	**5.70**	11.17	10.63	9.17	5.39	**10.19**	8.57	8.05
FAMPWQ	4	**7.07**	**13.17**	**12.20**	**10.81**	**5.70**	**11.15**	**10.61**	**9.15**	**5.37**	**10.19**	**8.55**	**8.04**

Table A.10: The accuracy ↑ of quantized Qwen2.5-7B and Qwen2.5-13B on zero-shot reasoning tasks. **Bold** indicates the highest accuracy and underlined indicates the second-highest. "Avg bit" represents the average width-bit. "Avg acc" represents the average accuracy of the 5 tasks.

Model		**Qwen2.5-7B**						**Qwen2.5-14B**					
Method	Avg bit	BoolQ	ARC-E	ARC-C	HellaSwag	WinoGrande	Avg acc	BoolQ	ARC-E	ARC-C	HellaSwag	WinoGrande	Avg acc
FP16	16	0.8471	0.8047	0.4778	0.6003	0.7301	0.6920	0.8522	0.8244	0.5597	0.6338	0.7537	0.7248
RTN	4	0.7883	0.7415	0.4377	0.5554	0.6645	0.63748	0.8144	0.7988	0.5042	0.6088	0.6921	0.6836
GPTQ	4	0.8394	0.7988	0.4692	0.5913	0.7111	0.68196	0.8404	0.8232	0.5546	0.6241	0.7334	0.7151
GPTQv2	4	0.8421	0.7974	0.4661	0.5923	0.7139	0.68236	0.8469	0.8167	0.5527	0.6244	0.7329	0.7147
OmniQuant	4	0.8132	0.7881	0.4679	0.5912	0.7104	0.67416	0.8454	0.8223	0.5475	0.6263	0.7568	0.7197
OWQ	4	0.8012	0.7832	0.4521	0.5723	0.6985	0.66146	0.8435	0.8123	0.5316	0.6183	0.7268	0.7065
AWQ	4	0.8143	0.7958	0.4650	**0.5926**	**0.7150**	0.67654	0.8391	**0.8274**	**0.5614**	0.6267	0.7537	0.7217
FAMPWQ (Ours)	4	**0.8495**	**0.7996**	**0.4812**	0.5853	0.6992	**0.68296**	**0.8496**	**0.8274**	0.5511	**0.6274**	**0.7576**	**0.7226**

Table A.11: Comparison of perplexity results ↓ of different 4-bit quantization approaches with the LLaMA2-7B-chat and LLaMA2-13B-chat model. "Avg PPL" represents the average PPL of the 3 benchmarks.

Model		**LLaMA2-7B-chat**				**LLaMA2-13B-chat**			
Method	Avg bit	Wiki2	PTB	C4	Avg PPL	Wiki2	PTB	C4	Avg PPL
FP16	16	6.94	12.07	9.51	9.51	5.09	9.08	6.79	6.99
RTN	4	7.96	13.70	10.94	10.87	6.42	11.05	9.02	8.83
GPTQ	4	7.29	12.76	10.12	10.06	6.29	10.93	8.78	8.67
GPTQv2	4	7.16	12.51	**9.75**	9.81	6.23	10.90	8.69	8.61
OmniQuant	4	7.15	12.45	9.91	9.84	6.27	10.99	8.82	8.69
OWQ	4	7.22	12.56	10.03	9.94	6.25	10.89	8.64	8.59
AWQ	4	7.15	12.48	9.85	9.83	**6.21**	10.87	8.65	8.58
FAMPWQ (Ours)	4	**7.12**	**12.41**	9.82	**9.78**	**6.21**	**10.85**	**8.63**	**8.56**

0.2% higher than AWQ) and LLaMA2-13B-chat (up to 2.87% higher than RTN), with FAMPWQ surpassing all or most baseline approaches for the majority of tasks.

A.7.6 Storage reduction

Comparable to single-precision quantization approaches, e.g., RTN, AWQ, GPTQ, GPTQv2, FAMPWQ incurs no extra storage overhead or no additional metadata, while OWQ corresponds to larger storage requirement with extra metadata. As shown in the Table A.13, FAMPWQ reduces storage overhead by 1%–3% of the original model size compared to OWQ. As the quantization bit width of the model weights decreases, the storage space required by each model is almost linearly reduced.

A.7.7 Sensitivity Metric Comparison

The sensitivity metric comparison is visualized in the main text (Figure 8). Table A.14 provides the same data in an extended format for reference.

A.7.8 Packed-Deployment Memory Accounting

Static model size alone does not capture the complete deployment footprint. We therefore perform analytical tensor accounting for packed Llama-2-7B inference with a 512-token prompt, 256 generated tokens, and an FP16 KV cache. The packed static weight footprints are 12.551 GiB for FP16, 2.862 GiB at a 3-bit average, and 3.622 GiB at a 4-bit average. Because FAMPWQ changes only the weight representation, its KV-cache footprint is identical to AWQ: 0.375 GiB at batch size 1 and 1.500 GiB at batch size 4. The largest per-layer FP16 materialization is at most 86 MiB (0.084 GiB), and this workspace is reused across sequential layer execution rather than allocated once per layer.

At the 3-bit average budget, packing reduces the weight footprint by 9.689 GiB, whereas the largest known temporary mixed-bit workspace is only 0.084 GiB. The resulting tensor-accounted saving is therefore at least 9.605 GiB, and the workspace is only 0.87% of the static weight saving. Under this accounting, temporary mixed-bit storage cannot offset the weight-memory reduction. This result is an analytical estimate rather than a measured runtime peak: activation and framework residuals, CUDA-reserved memory, allocator be-

Table A.12: The accuracy ↑ of quantized LLaMA2-7B-chat and LLaMA2-13B-chat on 5 zero-shot reasoning tasks under 4-bit quantization. **Bold** indicates the highest accuracy and underlined indicates the second-highest. "Avg bit" represents the average width-bit. "Avg acc" represents the average accuracy of the 5 tasks.

Model	**LLaMA2-7B-chat**							**LLaMA2-13B-chat**					
Method	Avg bit	BoolQ	ARC-E	ARC-C	HellaSwag	WinoGrande	Avg acc	BoolQ	ARC-E	ARC-C	HellaSwag	WinoGrande	Avg acc
FP16	16	0.8034	0.7028	0.4112	0.5740	0.6511	0.6285	0.8302	0.7571	0.447	0.6061	0.7103	0.6701
RTN	4	0.7425	0.6851	0.4018	0.5521	0.6551	0.6073	0.8125	0.7313	0.4085	0.5577	0.6787	0.6377
GPTQ	4	0.8017	0.6866	0.4052	0.5627	0.6421	0.6197	0.8220	0.7521	0.4436	0.5926	0.7127	0.6646
GPTQv2	4	0.8015	0.6883	0.4064	0.5636	**0.6477**	0.6215	0.8201	0.7530	0.4432	0.5993	0.7083	0.6648
OmniQuant	4	**0.8021**	**0.6975**	0.4071	0.5688	0.6418	0.6234	0.8204	0.7482	0.4324	0.5946	0.7034	0.6598
OWQ	4	0.7953	0.6898	0.4083	0.5642	0.6422	0.6199	0.8213	**0.7542**	0.4410	0.5892	0.7078	0.6627
AMQ	4	0.7963	0.6894	0.4067	0.5655	0.6423	0.6200	0.8208	0.7523	0.4392	0.5964	0.7008	0.6619
AWQ	4	0.7975	0.6948	0.4069	0.5672	0.6416	0.6216	0.8217	0.7478	0.4431	0.5937	**0.7166**	0.6645
FAMPWQ (Ours)	4	0.8012	0.6957	**0.4095**	**0.5692**	0.6424	**0.6236**	**0.8244**	0.7474	**0.4453**	**0.6023**	0.7127	**0.6664**

Table A.13: Storage requirements of various LLMs under different weight quantization precisions. Percentages in parentheses indicate the proportion relative to FP16 size.

Model		**RTN/AWQ/GPTQ&v2/FAMPWQ Storage (MB)**			**OWQ Storage (MB)**		
	FP16	**5-bit**	**4-bit**	**3-bit**	**5-bit**	**4-bit**	**3-bit**
LLaMA-7B	12,853	4,420 (34%)	3,589 (28%)	2,817 (22%)	4,505 (35%)	3,674 (29%)	2,899 (23%)
LLaMA-13B	24,826	8,550 (34%)	6,676 (27%)	5,164 (21%)	8,635 (35%)	6,761 (27%)	5,249 (21%)
LLaMA2-7B-chat	12,853	4,420 (34%)	3,589 (28%)	2,817 (22%)	4,505 (35%)	3,674 (29%)	2,899 (23%)
LLaMA2-13B-chat	24,826	8,550 (34%)	6,676 (27%)	5,164 (21%)	8,635 (35%)	6,761 (27%)	5,249 (21%)
Qwen2.5-7B	15,317	5,800 (38%)	5,197 (34%)	4,414 (29%)	5,883 (38%)	5,280 (34%)	4,502 (29%)
Qwen2.5-14B	28,172	10,600 (38%)	9,272 (33%)	7,697 (27%)	10,680 (38%)	9,365 (33%)	7,782 (28%)
Mistral-7B-v0.1	13,825	4,750 (34%)	3,841 (28%)	3,009 (22%)	4,826 (35%)	3,920 (28%)	3,075 (22%)

Table A.14: Sensitivity analysis of different metrics on LLaMA-7B and Qwen2.5-7B (extended from the main-text sensitivity comparison).

Metric	LLaMA-7B			Qwen2.5-7B		
	r ↑	Final PPL ↓	ΔPPL	r ↑	Final PPL ↓	ΔPPL
Random Allocation	0.04	6.87	+1.19	0.02	8.31	+1.47
Weight Magnitude ($\|W\|_2$)	0.42	6.53	+0.85	0.38	8.06	+1.22
FIM	**0.91**	**6.10**	**+0.42**	**0.88**	**7.63**	**+0.79**
Oracle (Ground-truth)	1.00	6.02	+0.34	1.00	7.51	+0.67

havior, and fragmentation still require measurement with an actual packed-kernel implementation.

A.7.9 Bit-width Allocation Visualization

Fig. A.4 visualizes the per-layer sensitivity scores and the corresponding quantization-optimized bit-width allocations. FIM captures gradient-level information that reveals additional critical layers overlooked by weight-only metrics. Thus, FAMPWQ can generate a structurally distinct allocation that assigns higher precision to the most loss-sensitive modules. In this way, FAMPWQ ultimately delivers excellent performance.

A.7.10 Validation of Modeling Assumptions

Our accuracy-degradation proxy (Formulas 1–3) rests on two assumptions: (1) exponential decay of degradation with bit-width, and (2) approximate layer independence.

Exponential Decay. Following Rate-Distortion Theory (Zhou et al., 2018), quantization error decreases exponentially with allocated bits. We verify this empirically by measuring per-layer PPL as a function of bit-width and fitting exponential curves ($R^2 > 0.95$ across all layer types for LLaMA-7B; see Figure 9).

Table A.15: Analytical packed-deployment memory accounting for Llama-2-7B. Values include packed weights, the final FP16 KV cache, and the known reusable workspace; they are not measured runtime peaks.

Avg. bit	**Batch**	**FP16**	**AWQ**	**FAMPWQ upper bound**	**Saving vs. FP16**
3	1	12.926 GiB	3.237 GiB	$\leq$ 3.321 GiB	$\geq$ 9.605 GiB
3	4	14.051 GiB	4.362 GiB	$\leq$ 4.446 GiB	$\geq$ 9.605 GiB
4	1	12.926 GiB	3.997 GiB	$\leq$ 4.081 GiB	$\geq$ 8.845 GiB
4	4	14.051 GiB	5.122 GiB	$\leq$ 5.206 GiB	$\geq$ 8.845 GiB

Table A.16: Controlled error additivity test on LLaMA-7B under 3-bit quantization. Relative error compares the additive prediction $\Delta(L_i) + \Delta(L_j)$ against the jointly measured degradation $\Delta(L_i, L_j)$.

Layers	$\Delta(L_i)$	$\Delta(L_j)$	**Sum**	**Actual**	**Rel. error**
L_{10}, L_{11} (Adjacent)	0.0084	0.0079	0.0163	0.016315	0.09%
L_5, L_{25} (Distant)	0.0062	0.0112	0.0174	0.017412	0.07%

Layer Independence. To validate the additivity assumption, we perform a controlled test on LLaMA-7B: we quantize layer pairs (L_{10}, L_{11}) and (L_5, L_{25}) individually, sum their PPL increases, and compare against the joint quantization. The relative error between the additive prediction and actual degradation is <0.1%, confirming that cross-layer interaction effects are negligible for the purpose of bit-width allocation.

α **Sensitivity.** The decay rate α in Formula 1 is robust across a wide range: as shown in Figure 9, PPL varies by less than 0.3 within $\alpha \in [15, 25]$ for both LLaMA-7B and Qwen2.5-7B.

Controlled Error Additivity Test. To further validate the layer-independence assumption used in the proxy model, we quantize two layers individually and jointly while keeping all other layers at FP16. If cross-layer interactions are negligible, the joint PPL increase should match the sum of the two individual increases.

As shown in Table A.16, the relative error is

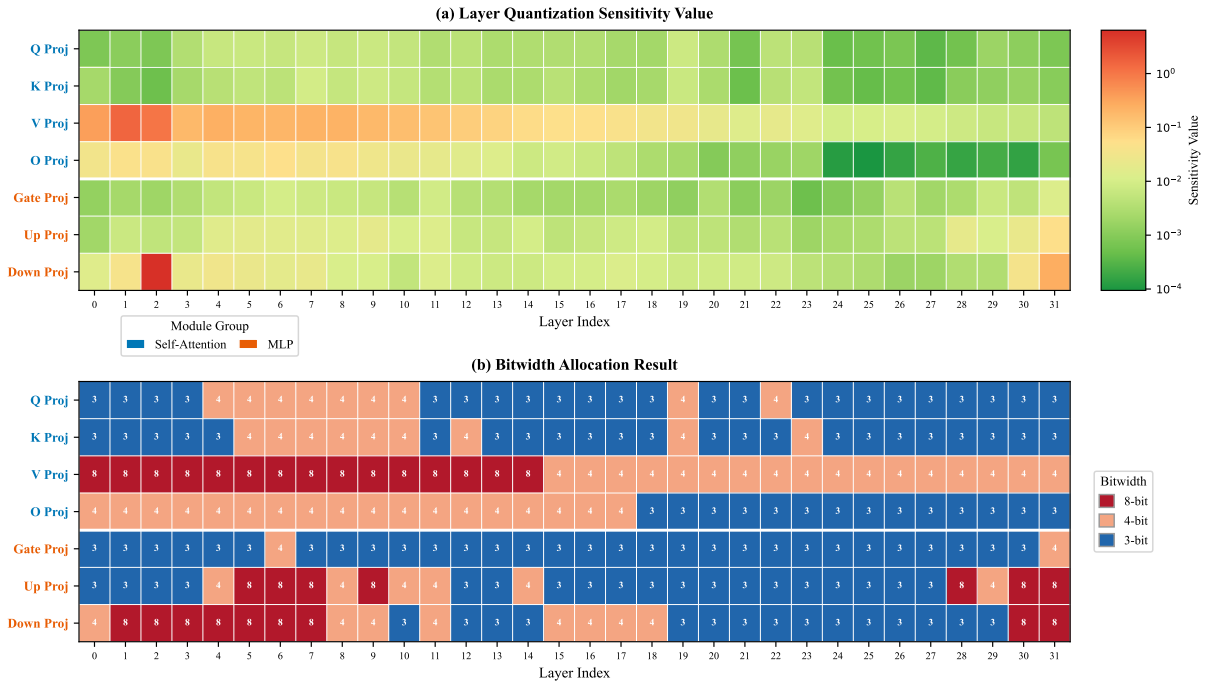


Figure A.4: Layer-wise sensitivity scores (top) and the resulting bit-width allocations (bottom) under the three metrics on LLaMA-7B ($b_t = 4$).

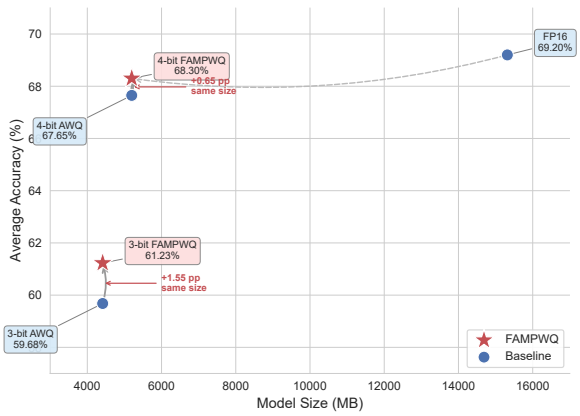


Figure A.5: Accuracy vs. model size trade-off for Qwen2.5-7B. At identical storage cost, FAMPWQ achieves +0.65 percentage points over AWQ at 4-bit and +1.55 points at 3-bit, demonstrating better accuracy-per-byte efficiency.

below 0.1% for both adjacent and distant layer pairs, supporting the approximation that cross-layer interaction effects are small for the purpose of bit-width allocation.

A.7.11 Vicuna-Bench Generation Quality at 3-bit

To bridge perplexity and real-world generation quality, we also evaluate 3-bit quantized models on Vicuna-Bench with GPT-4 as the judge. Table A.17 shows that FAMPWQ preserves generation quality better than the strongest uniform baseline under this aggressive compression setting.

Table A.17: Vicuna-Bench generation quality under 3-bit quantization. Win rate is measured against FP16 responses.

Method	Win rate	Avg. length
RTN	2.1%	164.2
GPTQ	13.3%	193.8
AWQ	18.5%	192.4
FAMPWQ (Ours)	**32.3%**	**208.0**

A.7.12 Accuracy-per-Byte Analysis

To evaluate the deployment efficiency of FAMPWQ, we analyze the accuracy-per-byte trade-off. As shown in Figure A.5, at identical storage costs (same average bit-width), FAMPWQ consistently achieves higher accuracy than the best uniform baseline (AWQ). The advantage grows as compression becomes more aggressive: +0.65 percentage points at 4-bit (5,197 MB) and +1.55 points at 3-bit (4,414 MB). This confirms that FAMPWQ extracts more quality from each byte of storage, making it particularly valuable for memory-constrained deployment scenarios where the goal is fitting the best possible model into a fixed VRAM budget.

A.7.13 Qualitative Generation Example at 3-bit

Aggregate win rates do not reveal which information is lost by a quantized model. We therefore com-

Table A.18: Representative Llama-2-Chat responses at a 3-bit average budget; light blue marks the factual error in 3-bit AWQ.

Model	Generated response
FP16	The cause of the delay is the introduction of a payment bug in the Tuesday update that can charge mobile orders twice. Seventeen beta users were affected, but desktop orders were not. The revised plan is to launch on Monday instead of Friday and to notify beta users and refund the affected users. The decision to launch on Monday will be made on Friday at 3 p.m.
3-bit AWQ	The cause of the delay was a payment bug that affected 17 of the 120 beta users but not the desktop orders. The customer impact was that the bug caused the mobile orders to be charged twice. The team was able to patch the issue by Tuesday and the patch was completed by Thursday. The revised plan was to move the launch to Monday and notify the affected beta users and refund them. The team will make the final go/no-go decision on Friday at 3 p.m.
3-bit FAMPWQ	The cause of the delay was a payment bug that charged mobile orders twice. This affected 17 beta users, but not the desktop users. The team was able to fix the issue by the end of the week, but the QA process will take two full days, so the launch will be on Monday. The decision to launch on Monday will be made on Friday at 3 p.m.

pare one representative response from FP16, 3-bit AWQ, and 3-bit FAMPWQ on Llama-2-Chat with `max_length=200`. The prompt requires a short summary of the cause, customer impact, and revised plan described in a multi-speaker dialogue:

> Briefly summarize the cause of the delay, the customer impact, and the revised plan described below. Write one short paragraph.
>
> **Maya:** Can we still launch on Friday?
> **Leo:** No. Tuesday's update introduced a payment bug that can charge mobile orders twice. Seventeen of our 120 beta users were affected; desktop orders were not affected. I can finish the patch by Thursday.
> **Nina:** QA needs two full days after the patch, so we should move the launch to Monday. I will notify the beta users and refund the 17 affected users today.
> **Maya:** Agreed. We will make the final go/no-go decision on Friday at 3 p.m.

Light-blue text marks AWQ's factual timeline error: it places the patch on Tuesday, the day the bug was introduced, and omits the two-day QA period. FAMPWQ preserves the cause, impact, QA delay, and Monday launch without introducing this contradiction, although it gives less precise patch timing and omits the notification and refund action. This example illustrates a specific low-bit failure mode rather than an aggregate claim.