AlphaQ: Calibration-Free Bit Allocation for Mixture-of-Experts Quantization

Wanqi Yang[1,2,3] **Yuexiao Ma**[4] **Alexander Conzelmann**[1,2,3] **Xiawu Zheng**[5]
Michael W. Mahoney[6,7,8] **T. Konstantin Rusch**[1,2,3,9] **Shiwei Liu**[1,2,3]
[1]Max Planck Institute for Intelligent Systems [2]ELLIS Institute Tübingen
[3]Tübingen AI Center [4]S-Lab, Nanyang Technological University
[5]Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Xiamen University
[6]International Computer Science Institute [7]Lawrence Berkeley National Laboratory
[8]University of California, Berkeley [9]Liquid AI

Abstract

Mixture-of-Experts (MoE) architectures scale model capacity through sparse expert activation, but their deployment remains memory-bound because all expert weights must reside in memory. Mixed-precision quantization can substantially reduce this footprint by assigning different bit-widths to different experts. Existing approaches, however, typically rely on calibration data to estimate expert importance and determine bit allocation. For frontier MoE LLMs, the original training data, and hence the true training distribution, is proprietary and inaccessible. As a result, calibration sets are inevitably imperfect surrogates, which can misestimate expert utilization and lead to suboptimal bit allocation. Motivated by the substantial cross-expert quality variability observed in modern MoE models, we propose **AlphaQ**, a *calibration-free bit-allocation method for MoE quantization*. AlphaQ draws on *Heavy-Tailed Self-Regularization (HT-SR) theory* and follows a simple principle: experts with more heavy-tailed weight spectra are typically better trained and should receive higher bit-widths, while experts with weaker heavy-tailed structure can be quantized more aggressively. AlphaQ operationalizes this principle by measuring expert-wise spectral heavy-tailedness and solving a budget-constrained optimization problem that minimizes total quantization error under a global bit-budget constraint. Across several MoE models, AlphaQ consistently outperforms calibration-based baselines under matched bit budgets. Notably, on Qwen1.5-MoE, AlphaQ achieves near full-precision accuracy with an average expert precision of only **3.5** bits, while delivering more than **4**$\times$ memory compression.

1 Introduction

Mixture-of-Experts (MoE) [Jacobs et al., 1991, Jordan and Jacobs, 1994, Shazeer et al., 2017, Lepikhin et al., 2020, Fedus et al., 2022, Zoph, 2022] have received widespread attention due to their computational efficiency: by routing each token to a small subset of experts, MoE achieves a strong trade-off between quality and efficiency at massive parameter counts. However, this sparsity often does *not* translate into memory reduction at deployment time. During inference, all expert weights must remain resident in GPU memory, making expert weights the primary memory bottleneck. Quantization is therefore an attractive path to make MoE deployable at scale [Frantar and Alistarh, 2023, Li et al., 2024, Hu et al., 2025a, Tao et al., 2025, Zheng et al., 2025, Chen et al., 2025]. However, since experts (and even layers within each expert) contribute differently to model performance [Sun et al., 2025, Hu et al., 2025a, Wang et al., 2026], precisely allocating bits across experts and layers remains challenging.

To address this challenge, most existing MoE quantization methods adopt a data-driven pipeline and derive mixed-precision configurations from input-dependent statistics [Huang et al., 2024, Duanmu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

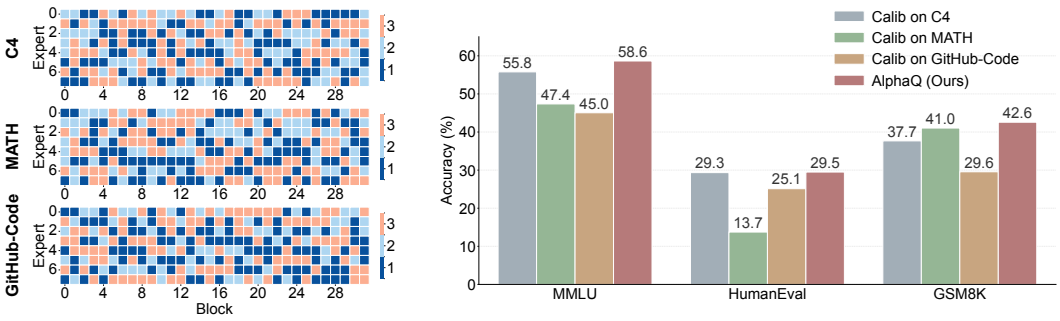


Figure 1: **Domain bias introduced by data-driven bit-width allocation in Mixtral-8×7B.** Left: bit-width allocations calibrated on datasets across domains (C4 [Raffel et al., 2020], MATH [Hendrycks et al., 2021b], GitHub-Code [Team, 2024a]) illustrate calibration-data-induced variations. Right: Mixtral-8×7B calibrated on these datasets with a 2.5-bit budget exhibits performance bias, overfitting to the calibration domain and degrading on unseen tasks, indicating that data-driven bit allocation reduces cross-domain generalization by coupling performance to the calibration distribution.

et al., 2025, Deng et al., 2026, Xie et al., 2025]. While effective in practice, these approaches suffer from two fundamental limitations: local optimality and dependence on calibration data.

First, some of these methods [Huang et al., 2024, Duanmu et al., 2025, Xie et al., 2025] allocate bits *locally* under a fixed budget, e.g., using the same bit budget for every block or expert, overlooking the fact that different Transformer blocks or experts contribute unequally to global performance. Such locally optimal choices can be globally suboptimal [Deng et al., 2026].

Second, since the original training data for frontier MoE LLMs is usually proprietary and inaccessible, existing quantization methods have to rely on biased and incomplete *calibration data* to estimate expert importance. Notably, an imperfect or non-representative calibration set may fail to activate certain experts, which yields biased importance estimates [Zheng et al., 2025, Hu et al., 2025a]. Consequently, calibration-based bit allocation directly couples the performance of quantized models to the calibration distribution, ultimately leading to poor cross-domain generalization.

Figure 1 illustrates this effect: Mixtral-8×7B [Jiang et al., 2024], quantized by the data-driven method PMQ [Huang et al., 2024] with different domain-specific datasets, exhibits substantially different mixed-precision allocations (expert activation patterns for different datasets are also provided in Appendix A), leading to highly skewed performance across common-sense tasks (MMLU [Hendrycks et al., 2021a]), math reasoning (GSM8K [Cobbe et al., 2021]), and coding (HumanEval [Chen et al., 2021]). For instance, a model calibrated on MATH excels at math reasoning but lags behind GitHub-Code-calibrated models on coding tasks, indicating overfitting to the calibration domain and degraded robustness on unseen domains. While an increasing number of approaches aim to improve the quality of calibration statistics [Zheng et al., 2025, Hu et al., 2025a], it remains fundamentally challenging for data-driven approaches to cover all potential domains.

In this paper, we tackle this challenge by proposing **AlphaQ**, a novel calibration-free[1] (and thus data-independent) bit-allocation method tailored for MoE quantization. AlphaQ is inspired by *Heavy-Tailed Self-Regularization (HT-SR) theory* [Martin and Mahoney, 2019], which substantiates the principle that experts with heavier-tailed weight spectra are empirically linked to more informative structure and stronger correlations [Martin and Mahoney, 2020, Martin et al., 2021, Martin and Mahoney, 2021]. Concretely, AlphaQ characterizes each expert's training sufficiency and importance from the shape of its weight-matrix spectrum, and allocates bit-widths under a global precision budget, i.e., experts with heavier-tailed spectra receive higher precision, whereas relatively under-trained or less sensitive experts with lighter-tailed spectra are quantized more aggressively.

At the core of AlphaQ is a unified, data-agnostic importance metric for quantifying expert disparity across MoE architectures and layers. Using this metric, we observe clear importance diversity across MoE families, within a model, and within a block. Notably, fine-grained, highly sparse MoEs (e.g.,

[1]We emphasize that the term *calibration-free* applies only to the bit-allocation stage. The subsequent quantization step can be implemented with any existing quantization method.

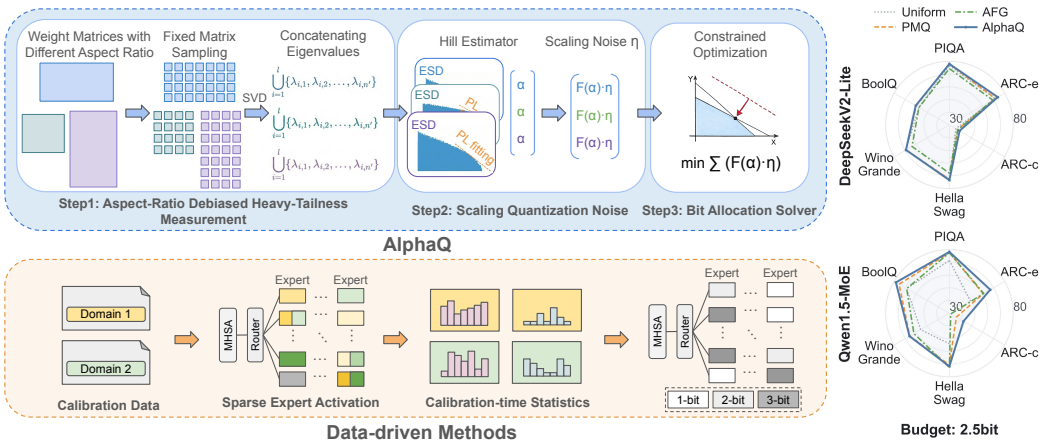


Figure 2: Comparison of the proposed AlphaQ framework and data-driven (or data-dependent) methods for MoE quantization.

DeepSeekV2-Lite [Liu et al., 2024a], Qwen1.5-MoE [Team, 2024c]) show larger variance, whereas vanilla MoEs with fewer experts (e.g., Mixtral-8×7B) show much smaller variance.

Based on this calibration-free importance signal, AlphaQ formulates mixed-precision quantization as a budget-constrained optimization problem. Our formulation is flexible enough to accommodate the large importance variance observed across MoE architectures, across models, depths, and widths, as well as across submodules within each expert. Figure 2 illustrates the comparison between AlphaQ and data-driven methods.

We demonstrate that, across a wide range of MoE-LLMs and benchmarks, AlphaQ reduces quantization-induced accuracy loss and outperforms existing calibration-based bit-allocation methods under matched bit budgets. With our quantized inference backend, the resulting low-bit MoE models further reduce weight memory footprint and improve end-to-end inference speed. Notably, with a 3.5-bit budget, Qwen1.5-MoE matches the BF16 model in accuracy while using just one-quarter of its weight memory footprint.

2 Background and Related Work

2.1 MoE Quantization

By representing weights and activations in low-precision formats, post-training quantization (PTQ) [Dettmers et al., 2022, Frantar et al., 2022, Xiao et al., 2023, Lin et al., 2024b, Liu et al., 2024b] substantially reduces the storage and computational cost of LLMs. Recently, PTQ for MoE-LLMs has received increasing attention. Prior works [Huang et al., 2024, Duanmu et al., 2025, Deng et al., 2026, Xie et al., 2025] primarily formulate optimization problems to assign different quantization precisions to sparsely activated experts. PMQ in MC-MoE [Huang et al., 2024] allocates precision by solving a transformer-block-wise linear programming problem using expert activation frequency and reconstruction error measured on calibration data. GEMQ [Deng et al., 2026] extends this approach by performing precision allocation at a global level. MxMoE [Duanmu et al., 2025] uses expert activation frequency and sensitivity from calibration to automatically generate customized kernels, enabling substantial speedups at comparable bit budgets. Furthermore, to account for the dependence of expert activation on calibration data, MoEQuant [Hu et al., 2025a] proposes an expert-balanced sampling strategy for calibration, which improves traditional quantization methods in MoE but does not remove the limitation of calibration in bit allocation. DynaMo [Zheng et al., 2025] emphasizes that MoE quantization should be aware of data-model distribution shifts by modeling expert dynamics across different input distributions. However, its experiments mainly consider transfer across general-purpose text distributions (e.g., WikiText2, C4), rather than explicitly exposing domain-dependent calibration bias in bit allocation. By contrast, AlphaQ directly targets this failure mode and avoids it through calibration-free global bit allocation.

2.2 HT-SR Theory and Metrics

Heavy-Tailed Self-Regularization (HT-SR) theory provides a framework to analyze neural network training sufficiency by examining the empirical spectral density (ESD) of layer-wise weight correlation matrices [Martin and Mahoney, 2019, 2020, Martin et al., 2021, Martin and Mahoney, 2021]. Rooted in Random Matrix Theory [Couillet and Liao, 2022] and statistical physics [Engel and den Broeck, 2001], HT-SR posits that the heavy-tailed structure of trained weight matrices' ESDs strongly predicts model performance [Martin et al., 2021, Yang et al., 2023]. ESDs are often modeled with "spikes" (ground-truth-aligned learned features) and a "bulk" (noise following the Marchenko-Pastur law [Wang et al., 2023]), but HT-SR identifies state-of-the-art networks as "strongly correlated systems," making heavy-tailed metrics more suitable. Informally, ESD heavy-tailed distributions stem from spike-bulk interaction, marking a critical "bulk-decay" phase for well-trained models [Kothapalli et al., 2024]. HT-SR metrics include "scale metrics" (correlating with generalization gaps [Yang et al., 2023]) and "shape metrics", e.g., fitted power-law exponents (α) [Martin et al., 2021] or the robust `PL_Alpha_Hill` [Zhou et al., 2023], which capture weight matrix structural quality for model diagnostics and pruning [Liu et al., 2024c, Lu et al., 2024, Hu et al., 2025b, He et al., 2025], even predicting state-of-the-art model quality without training/testing data [Martin et al., 2021, Yang et al., 2023].

3 AlphaQ

In this section, we introduce **AlphaQ**. We first define our notation and compute our calibration-free importance score from model weights using `PL_Alpha_Hill`. We then formulate a global, budget-constrained optimization to assign mixed precisions across experts and submodules.

3.1 Notation

In this work, we define a *block* as a Transformer block. In popular MoE models, each block comprises two modules: an attention module and an MoE module containing multiple experts and a routing layer. Each attention or MoE module consists of multiple *layers* (e.g., projection layers). A terminology summary is provided in Appendix B.

We consider a MoE model with L layers. Let $\mathbf{W}_i \in \mathbb{R}^{m\times n}$ denote the weight matrix of the i-th layer, and define the corresponding correlation matrix as $\mathbf{X}_i = \mathbf{W}_i^\top \mathbf{W}_i$. The empirical spectral density (ESD) of $\mathbf{X}_i$, viewed as a probability measure over the eigenvalue distribution of $\mathbf{X}_i$, is defined as

$$\mu_{\mathbf{X}_i} := \frac{1}{n}\sum_{j=1}^{n} \delta_{\lambda_j(\mathbf{X}_i)}, \tag{1}$$

where $\lambda_1(\mathbf{X}_i) \le \cdots \le \lambda_n(\mathbf{X}_i)$ are the eigenvalues of $\mathbf{X}_i$, and δ denotes the Dirac delta function [Hassani, 2009].

3.2 Estimating Layer Importance in MoE

When calibration-time activations are unreliable or biased, we derive the bit allocation signal directly from model parameters. We leverage the empirical observation that weight matrix spectral structure encodes learned correlations and noise sensitivity [Martin and Mahoney, 2019, 2020, Lu et al., 2024]. Specifically, HT-SR theory demonstrates that layers extracting more informative features develop heavy-tailed ESDs of their weight matrices, indicating better training and more effective inference signal extraction [Lu et al., 2024]. We therefore estimate layer importance based on ESD heavy-tailedness: weight matrices with strongly heavy-tailed ESDs encode richer correlation structures and receive higher bit-widths, while layers with lighter-tailed ESDs receive lower bit-widths.

PL_Alpha_Hill Metric. To quantify the heavy-tailed characteristics of weight matrix ESDs, we employ the `PL_Alpha_Hill` metric [Zhou et al., 2023]. This metric partitions the eigenvalue domain into equal-width bins and counts the eigenvalues within each. We define the ESD tail as eigenvalues from the bin with maximum count along with all eigenvalues exceeding this bin's upper bound, which is then modeled using a power-law (PL) density:

$$p(\lambda) \propto \lambda^{-\alpha}, \qquad \lambda_{\min} < \lambda < \lambda_{\max}, \tag{2}$$

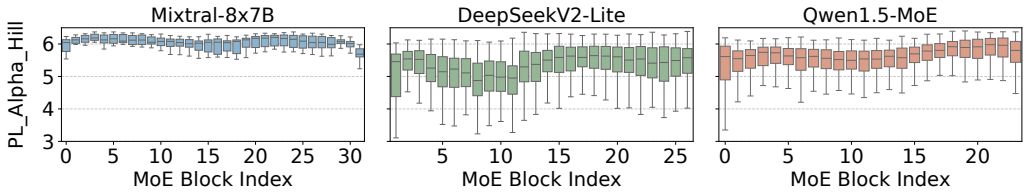


Figure 3: **Distribution of `PL_Alpha_Hill` across all up/gate/down projections in three representative MoE-LLMs.** The bottom and top of each boxplot indicate the minimum and maximum values of `PL_Alpha_Hill` across all up/gate/down projections within the block. The lower and upper edges of the box correspond to the first and third quartiles for that block, respectively, and the horizontal line inside the box denotes the median value.

where the exponent α characterizes the degree of heavy-tailedness, with smaller values indicating heavier tails. We then estimate the exponent α via the Hill estimator [Hill, 1975] on the ESD tail:

$$\alpha = 1 + \left(\frac{1}{k} \sum_{i=1}^{k} \ln \frac{\lambda_{n-i+1}}{\lambda_{n-k}} \right)^{-1}, \tag{3}$$

where $\{\lambda_i\}_{i=1}^{n}$ are the eigenvalues used for tail fitting, sorted in ascending order, and k determines the lower threshold $\lambda_{\min}$ for truncated power-law estimation. The derivation of Equation 3 is provided in Appendix C. We select k using the *Fix-finger* method [Yang et al., 2023], which aligns $\lambda_{\min}$ with the peak of the ESD. To overcome the effect of matrix aspect ratio on spectral shape measurements [Hu et al., 2025b], we use the *Fixed-Aspect-Ratio Matrix Subsampling* (FARMS) [Hu et al., 2025b]. Detailed computation is provided in Appendix C. Following HT-SR theory, we interpret smaller `PL_Alpha_Hill` values as stronger spectral heavy-tailedness and higher relative importance for bit allocation. Notably, `PL_Alpha_Hill` is a calibration-free and globally comparable importance metric.

Alpha-based Importance Analysis. As shown in Figure 3, across popular MoE models, `PL_Alpha_Hill` exhibits clear variation across layers, blocks, and architectures. Since smaller `PL_Alpha_Hill` indicates heavier-tailed spectra and higher relative importance, this variation motivates globally fine-grained bit allocation. Detailed statistics are deferred to Appendix D.

3.3 Bit Allocation Optimization

After establishing a globally comparable importance signal, we next allocate bit-widths to minimize accuracy degradation under a fixed global bit budget.

Importance-Scaled Quantization Noise. However, importance alone is insufficient to determine the bit-widths: bit allocation requires quantifying both the benefit of assigning higher precision and the cost of reducing precision elsewhere in the budget. While `PL_Alpha_Hill` provides a calibration-free importance estimate, it does not capture the numerical perturbation induced by low-bit quantization. Conversely, quantization-induced numerical perturbation (i.e. quantization noise) alone does not indicate whether the affected module is structurally important. Appendix E provides a detailed analysis of this interaction.

We model the magnitude of quantization noise as a function of both the quantization bit-width and the weight distribution. Under the standard uniform-noise approximation, the quantization noise is modeled as zero-mean noise whose variance is proportional to the squared quantization step size. Since the step size decays exponentially with the bit-width b, i.e., $\Delta_{l,b} \propto 2^{-b}$, the corresponding noise variance scales as 2^{-2b}. The derivation is provided in Appendix F. To account for layer-wise differences in weight distribution, we modulate this term by the variance of the weight matrix $\mathrm{Var}(\mathbf{W}_l)$ and use $\mathrm{Var}(\mathbf{W}_l)2^{-2b}$ to quantify layer-wise quantization noise.

We then scale this quantization noise by alpha-based importance, penalizing perturbations more strongly on important layers. Specifically, we scale layer-wise quantization noise by $(\tilde{\alpha}/\alpha_l)^{\gamma}$, where $\tilde{\alpha} = \mathrm{median}\{\alpha_l\}_{l=1}^{L}$ and γ is a data-free curvature parameter. Let $\alpha_{\min}$, $\alpha_{\max}$, and v_α denote the minimum, maximum, and variance of `PL_Alpha_Hill` over the target modules. We use the

default $\gamma_{\text{default}} = \alpha_{\min}(\alpha_{\max} - \alpha_{\min})/v_\alpha$, derived in Appendix G. We therefore define the scaled quantization noise of layer l at bit-width b as

$$\eta_{l,b} = \left(\frac{\tilde{\alpha}}{\alpha_l}\right)^{\gamma} \cdot \mathrm{Var}(\mathbf{W}_l) 2^{-2b} \tag{4}$$

Budget-Constrained Formulation. Using $\eta_{l,b}$ as the layer-wise cost, we minimize the total scaled quantization noise across all layers under a target bit budget. Let $\mathcal{B}$ be the set of candidate bit-widths (e.g., $\mathcal{B} = \{1, 2, 3, 4\}$). To formalize the allocation decision, we introduce a binary indicator variable $x_{l,b} \in \{0, 1\}$, where $x_{l,b} = 1$ if layer l is assigned bit-width b, and 0 otherwise. To ensure a valid configuration, we impose two constraints: i) each layer must be assigned exactly one bit; ii) for the layer l with N_l parameters, the cost of assigning bit b is $N_l \cdot b$. The total size of the quantized model must not exceed a target budget B_{tot}.

Combining the objective and constraints yields the following Integer Linear Programming (ILP) formulation:

$$\begin{aligned} \min_{\{x_{l,b}\}} \quad & \sum_{l=1}^{L} \sum_{b \in \mathcal{B}} x_{l,b} \cdot \eta_{l,b} \\ \text{s.t.} \quad & \sum_{l=1}^{L} \sum_{b \in \mathcal{B}} x_{l,b} \cdot N_l \cdot b \le B_{\text{tot}}, \\ & \sum_{b \in \mathcal{B}} x_{l,b} = 1, \quad \forall l \in \{1, \ldots, L\}, \quad x_{l,b} \in \{0, 1\}. \end{aligned} \tag{5}$$

Solving Eq. 5 minimizes the modeled importance-scaled quantization noise. Appendix O evaluates how this surrogate relates to measured errors across quantizers. Eq. 5 is a standard instance of the Multiple-Choice Knapsack Problem [Pisinger and Toth, 1998]. Modern ILP solvers (e.g., PuLP [Mitchell et al., 2011]) can solve it to global optimality within seconds.

4 Empirical Results

4.1 Experimental Setting

We evaluate AlphaQ on four representative MoE LLMs: DeepSeekV2-Lite [Liu et al., 2024a], Qwen1.5-MoE [Team, 2024c], Mixtral-8×7B [Jiang et al., 2024], and Qwen3-30B-A3B [Yang et al., 2025]. For `PL_Alpha_Hill` calculation, we use FARMS submatrices of size 256×256 with sampling stride 256 along each dimension. For quantization, we follow prior work [Huang et al., 2024] and perform weight-only quantization with group-wise, asymmetric GPTQ [Frantar et al., 2022] (group size 128, calibrated with 128 samples from WikiText2). We use uniform 4-bit quantization for all non-expert layers, including attention and the router. For experts, we apply mixed-precision quantization at layer granularity: each layer is assigned a bit-width from $\{1, 2, 3, 4\}$, and we control the compression level using the average bits per layer, defined as $B_{\text{avg}} = \sum_l N_l b_l / \sum_l N_l$ over allocated expert projections. This excludes quantization metadata and non-expert layers. Appendix N gives the accounting details. We report results under two bits-per-layer settings: 2.5 and 3.5.

We compare AlphaQ against two baselines under matched budget settings: Uniform, which assigns the same bit to all experts, and PMQ, a calibration-based mixed-precision MoE quantization method [Huang et al., 2024]. Specifically, for Uniform, when the budget is $x.5$, we follow prior work [Huang et al., 2024] by setting the first half of the MoE blocks to $x+1$ bits and the second half to x bits. Moreover, we compare AlphaQ with additional bit-allocation methods in certain settings, including block score predictor (BSP) [Li et al., 2024], a Hessian-based method (Hessian) [Dong et al., 2020], and Automated Fine-Grained MoE Quantization (AFG) [Xie et al., 2025]. For evaluation, we report perplexity (PPL ↓) on WikiText2 and average zero-shot accuracy (Avg. ↑) over five benchmarks: PIQA [Bisk et al., 2020], ARC-Easy (ARC-e), ARC-Challenge (ARC-c) [Clark et al., 2018], HellaSwag [Zellers et al., 2019], and WinoGrande (WinoG.) [Sakaguchi et al., 2021], using the EleutherAI LM Harness [Gao et al., 2024].

Table 1: **Results on DeepSeekV2-Lite, Qwen1.5-MoE, Mixtral-8×7B and Qwen3-30B-A3B.** Perplexity↓ on WikiText2 and accuracy↑ on five zero-shot tasks. Avg. denotes the average accuracy across the five tasks. The best results under each bit budget are highlighted in bold.

Model	Bits	Method	WikiText2↓	Accuracy↑					
				PIQA	ARC-e	ARC-c	HellaSwag	WinoG.	Avg.
Mixtral-8×7B	16	Original	3.84	83.57	83.67	59.30	84.03	76.24	77.36
	2.5	Uniform	6.16	79.38	77.40	50.26	58.84	73.40	67.86
		BSP	13.61	68.23	54.97	28.38	55.61	62.19	53.88
		Hessian	5.41	80.21	76.38	51.20	67.05	72.97	69.56
		AFG	–	74.32	69.20	37.57	67.88	63.80	62.55
		PMQ	5.32	80.36	77.10	51.28	69.23	**73.95**	70.38
		AlphaQ	**5.17**	**80.84**	**78.87**	**51.59**	**69.24**	73.24	**70.76**
	3.5	Uniform	4.29	81.28	83.04	54.95	73.58	**76.19**	73.81
		PMQ	4.25	81.21	82.49	53.92	72.90	76.09	73.32
		AlphaQ	**4.23**	**81.51**	**83.38**	**55.80**	**74.67**	76.10	**74.29**
DeepSeekV2-Lite	16	Original	6.31	80.08	76.47	48.98	78.01	71.51	71.01
	2.5	Uniform	8.14	76.66	71.34	35.84	72.44	69.46	65.15
		AFG	–	74.32	69.20	37.57	67.88	63.80	62.55
		PMQ	6.95	77.86	71.68	37.37	72.76	69.06	65.75
		AlphaQ	**6.64**	**78.02**	**73.77**	**39.22**	**73.38**	**69.30**	**66.74**
	3.5	Uniform	6.83	79.21	74.56	44.57	76.27	70.83	69.09
		PMQ	6.57	79.63	75.26	45.83	76.97	71.04	69.75
		AlphaQ	**6.44**	**79.89**	**75.51**	**46.51**	**77.21**	**71.21**	**70.07**
Qwen1.5-MoE	16	Original	7.22	80.52	69.23	44.11	77.21	68.59	67.93
	2.5	Uniform	10.36	71.49	49.07	29.44	53.98	55.96	51.99
		AFG	–	77.53	61.07	31.20	**72.77**	62.12	60.94
		PMQ	9.01	77.58	60.56	36.26	70.69	65.82	62.18
		AlphaQ	**8.14**	**78.21**	**66.97**	**42.72**	71.81	**66.12**	**65.17**
	3.5	Uniform	8.13	79.31	66.81	41.51	74.52	67.54	65.94
		PMQ	7.42	80.31	69.02	43.81	77.13	68.50	67.75
		AlphaQ	**7.38**	**80.46**	**69.15**	**44.33**	**77.70**	**68.55**	**68.04**
Qwen3-30B-A3B	16	Original	8.71	80.25	79.29	55.80	77.75	71.11	72.84
	2.5	Uniform	10.29	78.07	**75.93**	50.07	71.72	68.01	68.76
		PMQ	10.94	76.61	68.43	45.99	72.13	66.22	65.88
		AlphaQ	**9.27**	**78.34**	75.87	**50.35**	**74.49**	**68.71**	**69.56**

4.2 Main Results

As shown in Table 1, mixed-precision quantization generally yields higher quality than uniform bit-width assignment, especially under low-bit budgets, reflecting the heterogeneous importance of layers in MoE models. Moreover, compared with the calibration-based method PMQ, AlphaQ achieves consistently stronger results across models and budgets, indicating that our calibration-free bit-allocation method yields better performance. More detailed results are provided in Appendix H.

Notably, on Qwen1.5-MoE, AlphaQ with a bit budget of 3.5 performs competitively with the BF16 model in average accuracy over five zero-shot tasks. In the more aggressive 2.5-bit setting, AlphaQ also maintains relatively strong performance across all three models. For example, its average accuracy on zero-shot tasks drops by only 4.3% on DeepSeekV2-Lite, 2.8% on Qwen1.5-MoE, and 6.6% on Mixtral-8×7B, demonstrating that our bit allocation remains beneficial even when the bit budget is extremely tight. To evaluate AlphaQ on larger-scale MoE models, we include results for Qwen3-30B-A3B. Under a 2.5-bit budget, Qwen3-30B-A3B also surpasses both uniform and PMQ in overall performance.

4.3 AlphaQ vs. Multi-Domain Calibration Baselines

To further compare AlphaQ to baselines with broader calibration coverage, we use PMQ [Huang et al., 2024] on OLMoE-1B-7B [Muennighoff et al., 2024] under a 3-bit budget with various single-domain and mixed-domain calibration sets, and evaluate the quantized models on the common-sense

Table 2: **Comparison between AlphaQ and PMQ with multi-domain calibration on OLMoE-1B-7B under a 3-bit budget.** The best results in each benchmark are highlighted in bold.

Method	MMLU	MATH	CEval	Avg.
BF16	53.40	21.40	31.06	35.29
Uniform 3-bit	43.56	11.45	27.19	27.40
PMQ (C4)	43.80	12.07	25.11	26.99
PMQ (CEval-dev)	44.04	11.15	22.66	25.95
PMQ (MATH)	44.43	12.82	22.96	26.74
PMQ (C4+CEval-dev)	**44.79**	11.71	23.03	26.51
PMQ (C4+MATH)	44.38	12.91	25.78	27.69
PMQ (MATH+CEval-dev)	44.33	12.37	21.99	26.23
AlphaQ	44.75	**13.11**	**28.72**	**28.86**

Table 3: **Component ablation on OLMoE-1B-7B.** We report perplexity on WikiText2 and average accuracy over six zero-shot tasks. Alpha denotes `PL_Alpha_Hill`, and **Direct** means multiplying Alpha and Quantization Noise directly.

Method	PPL ↓	Avg. Acc. ↑
BF16	7.95	70.45
Uniform 3-bit	11.78	64.73
Noise-only	11.22	63.74
Alpha-only	10.03	66.23
Alpha+Noise (Direct)	9.56	66.81
Alpha+Noise (Ours)	**9.19**	**67.11**

Table 4: **Ablation study for bit-budget allocation.** We compare perplexity (PPL) on WikiText2 and accuracy (Acc.) on six zero-shot benchmarks under block-wise and global budgets.

Bit	Budget	Mixtral-8×7B		DeepSeekV2-Lite	
		PPL (↓)	Acc. (↑)	PPL (↓)	Acc. (↑)
2.5	Block	5.81	71.68	6.99	64.61
	Global (Ours)	**5.17**	**72.39**	**6.64**	**65.69**
3.5	Block	4.71	72.56	6.59	68.72
	Global (Ours)	**4.23**	**75.98**	**6.44**	**69.03**

Table 5: **Ablation study for bit-allocation granularity.** We compare perplexity on WikiText2 in expert-wise and layer-wise bit allocation under different bit budgets.

Model	2-bit PPL ↓		3-bit PPL ↓	
	Expert	Layer	Expert	Layer
Mixtral 8×7B	6.28	**6.11**	4.72	**4.37**
DeepSeekV2-Lite	7.47	**7.30**	6.81	**6.69**

benchmark MMLU, the math reasoning benchmark MATH, and the Chinese language benchmark CEval [Huang et al., 2023].

Table 2 shows that PMQ remains sensitive to the calibration domains even when multi-domain mixtures are allowed. Among the calibration setting variants, even the multi-domain PMQ configuration does not consistently match Uniform on every axis, whereas AlphaQ improves the overall average by avoiding calibration-domain bias. We also note that AlphaQ performs better than the recent cross-dataset method DynaMo on OLMoE benchmarks, with detailed results deferred to Appendix I.

These results show that enlarging calibration coverage shifts the failure mode across tasks, rather than removing sensitivity to the calibration datasets.

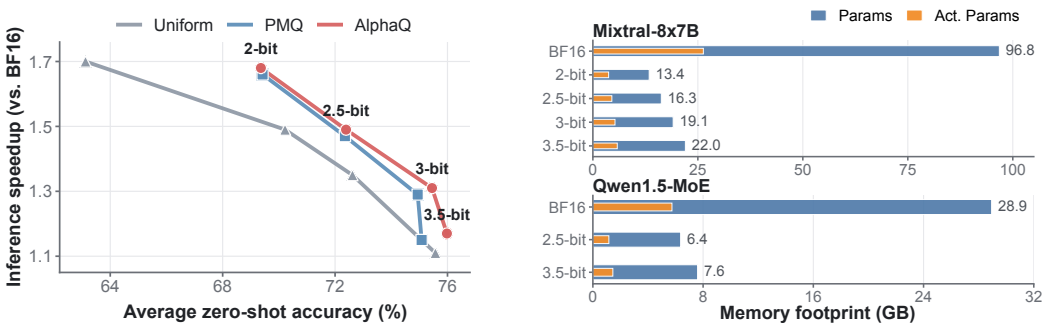
4.4 Ablation Study

Which Components Matter in AlphaQ? To isolate the contribution of each component, we perform an ablation study on OLMoE-1B-7B under a 3-bit budget. Our goal is to quantify the gains from `PL_Alpha_Hill`, quantization noise, and their combination.

Table 3 shows that `PL_Alpha_Hill` already provides a much stronger signal than quantization noise for bit allocation, indicating that `PL_Alpha_Hill` captures meaningful layer importance. However, neither `PL_Alpha_Hill` nor noise-only allocation is sufficient, as both remain clearly behind the final AlphaQ solution.

Incorporating quantization noise improves performance further. The gap between the Direct variant and the final variant shows that the way `PL_Alpha_Hill` and quantization noise are combined also matters. The best results are obtained only when the alpha-based importance metric and the quantization-noise term are properly used together.

How to Allocate Bit-Width Across Blocks? We conduct an ablation study on two budget-allocation strategies: i) fixing the global average bit-width for the entire model; and ii) fixing the average bit-width for each block. The results are reported in Table 4. In all settings, global budgeting outperforms block-wise budgeting. This indicates that different blocks in MoE models have unequal importance, and HT-SR theory provides a unified view for comparing layer importance across blocks.



(a) Accuracy–speedup on Mixtral-8×7B.

(b) Memory footprint of AlphaQ.

Figure 4: **End-to-end efficiency of AlphaQ.** Left: average zero-shot accuracy versus inference speedup relative to BF16 for varying bit budgets on Mixtral-8×7B. Right: parameter memory footprint of Mixtral-8×7B and Qwen1.5-MoE.

How to Allocate Bit-Width within Blocks? We perform an ablation study on bit-allocation granularity, comparing layer-wise and expert-wise allocation. As shown in Table 5, the finer-grained, layer-wise strategy consistently outperforms expert-wise allocation across all settings. These findings indicate that AlphaQ can effectively capture the relative importance of layers within each expert and allocate bit-widths accordingly, enabling finer-grained mixed-precision quantization.

4.5 Efficiency

PL_Alpha_Hill overhead. The offline preprocessing cost of computing `PL_Alpha_Hill` is modest. Even for Qwen3-30B-A3B, computing `PL_Alpha_Hill` takes about nine minutes, which we consider acceptable for practical deployment. Detailed timings are reported in Appendix J.

Quantized inference backend. We evaluate AlphaQ's inference performance on an NVIDIA A40 GPU (48 GB) using a quantized inference backend that combines low-bit dequantization with Tensor Core GEMM. Relative to the original BF16 model, this backend improves both prefill and decode. We reuse the PMQ-style HQQ [Badri and Shaji, 2023] backend for dequantization in the prefill stage of all layers and the decode stage of non-expert layers, and implement fused Triton kernels to reduce memory traffic in the decode stage of MoE expert layers. Detailed runtime implementation and profiling are provided in Appendix K.

End-to-end efficiency. Figure 4 reports the trade-offs among inference speedup, memory footprint, and accuracy. With our optimized quantized backend, AlphaQ consistently dominates the frontier for Mixtral-8×7B, providing a superior trade-off between accuracy and inference speedup. Furthermore, AlphaQ substantially alleviates the MoE memory bottleneck across bit budgets. For Mixtral-8×7B, AlphaQ reaches 1.68× speedup at 2-bit while reducing parameter memory from 96.8 GB to 13.4 GB. Even at a 3.5-bit budget, it reduces Mixtral-8×7B parameter memory to 22.0 GB, corresponding to 4.4× compression. For Qwen1.5-MoE at a 3.5-bit budget, AlphaQ preserves BF16-level accuracy with only 7.6 GB of weights, about one quarter of BF16 memory. More detailed efficiency results are provided in Appendix L.

5 Conclusion

In this work, we propose **AlphaQ**, a calibration-free bit-allocation method for MoE quantization. Motivated by HT-SR theory, AlphaQ derives an importance signal directly from model weights by measuring spectral heavy-tailedness. Using this metric, we systematically analyze MoE models from an HT-SR perspective and reveal importance diversity across multiple levels. Based on these observations, AlphaQ allocates bit-widths via a global constrained optimization under a fixed bit budget, enabling calibration-free bit allocation for MoE models. Extensive experiments show that weight-based bit allocation is an effective and promising approach for improving the generalization of mixed-precision MoE PTQ.

Acknowledgments and Disclosure of Funding

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 448588364. Alexander Conzelmann is a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645.

References

Hicham Badri and Appu Shaji. Half-quadratic quantization of large machine learning models, November 2023. URL `https://dropbox.github.io/hqq_blog/`.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Yuanteng Chen, Yuantian Shao, Peisong Wang, and Jian Cheng. Eac-moe: Expert-selection aware compressor for mixture-of-experts large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12942–12963, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Romain Couillet and Zhenyu Liao. *Random matrix methods for machine learning*. Cambridge University Press, 2022.

Jianing Deng, Song Wang, Dongwei Wang, Zijie Liu, Tianlong Chen, Huanrui Yang, and Jingtong Hu. Towards global expert-level mixed-precision quantization for mixture-of-experts LLMs, 2026. URL `https://openreview.net/forum?id=wAc718O8UM`.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529, 2020.

Haojie Duanmu, Xiuhong Li, Zhihang Yuan, Size Zheng, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. Mxmoe: Mixed-precision quantization for moe with accuracy and performance co-design. *arXiv preprint arXiv:2505.05799*, 2025.

A. Engel and C. P. L. Van den Broeck. *Statistical mechanics of learning*. Cambridge University Press, New York, NY, USA, 2001.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Elias Frantar and Dan Alistarh. Qmoe: Practical sub-1-bit compression of trillion-parameter models. *arXiv preprint arXiv:2310.16795*, 2023.

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

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, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

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

Sadri Hassani. Dirac delta function. In *Mathematical Methods: For Students of Physics and Related Fields*, pages 289–319. Springer, 2009.

Di He, Ajay Jaiswal, Songjun Tu, Li Shen, Ganzhao Yuan, Shiwei Liu, and Lu Yin. Alphadecay: Module-wise weight decay for heavy-tailed balancing in llms, 2025. URL `https://arxiv.org/abs/2506.14562`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021b.

Bruce M Hill. A simple general approach to inference about the tail of a distribution. *The annals of statistics*, pages 1163–1174, 1975.

Xing Hu, Zhixuan Chen, Dawei Yang, Zukang Xu, Chen Xu, Zhihang Yuan, Sifan Zhou, and Jiangyong Yu. Moequant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. *arXiv preprint arXiv:2505.03804*, 2025a.

Yuanzhe Hu, Kinshuk Goel, Vlad Killiakov, and Yaoqing Yang. Eigenspectrum analysis of neural networks without aspect ratio bias. *arXiv preprint arXiv:2506.06280*, 2025b.

Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. Mixture compressor for mixture-of-experts llms gains more. *arXiv preprint arXiv:2410.06270*, 2024.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36: 62991–63010, 2023.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Vignesh Kothapalli, Tianyu Pang, Shenyang Deng, Zongmin Liu, and Yaoqing Yang. Crafting heavy-tails in weight matrix spectrum without gradient noise. *arXiv preprint arXiv:2406.04657*, 2024.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Pingzhi Li, Xiaolong Jin, Yu Cheng, and Tianlong Chen. Examining post-training quantization for mixture-of-experts: A benchmark. 2024.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs. *Advances in Neural Information Processing Systems*, 37:87766–87800, 2024a.

Haokun Lin, Xinle Jia, Haobo Xu, Bingchen Yao, Xianglong Guo, Yichen Wu, Zhichao Lu, Ying Wei, Qingfu Zhang, and Zhenan Sun. DuQuant++: Fine-grained rotation enhances microscaling FP4 quantization. *arXiv preprint arXiv:2604.17789*, 2026.

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

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Kui Liu, Bart Goossens, Tom De Schepper, and Wilfried Philips. Improving post-training quantization via probabilistic programming. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025. doi: 10.1109/TCSVT.2025.3588737.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024b.

Zihang Liu, Yuanzhe Hu, Tianyu Pang, Yefan Zhou, Pu Ren, and Yaoqing Yang. Model balancing helps low-data training and fine-tuning. *arXiv preprint arXiv:2410.12178*, 2024c.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 32332–32344. PMLR, 2024d. URL `https://proceedings.mlr.press/v235/liu24bz.html`.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W Mahoney, and Yaoqing Yang. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *Advances in neural information processing systems*, 37:9117–9152, 2024.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Yuexiao Ma, Taisong Jin, Xiawu Zheng, Yan Wang, Huixia Li, Yongjian Wu, Guannan Jiang, Wei Zhang, and Rongrong Ji. OMPQ: Orthogonal mixed precision quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9029–9037, 2023a. doi: 10.1609/aaai.v37i7.26084.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Xuefeng Xiao, Rui Wang, Shilei Wen, Xin Pan, Fei Chao, and Rongrong Ji. Solving oscillation problem in post-training quantization through a theoretical perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7950–7959, 2023b. doi: 10.1109/CVPR52729.2023.00768.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. AffineQuant: Affine transformation quantization for large language models. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/ddfb58b61e7213dfb9e06c695475b2bd-Abstract-Conference.html.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. Outlier-aware slicing for post-training quantization in vision transformer. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 33811–33825. PMLR, 2024b. URL https://proceedings.mlr.press/v235/ma24f.html.

C. H. Martin and M. W. Mahoney. Heavy-tailed Universality predicts trends in test accuracies for very large pre-trained deep neural networks. In *Proceedings of the 20th SIAM International Conference on Data Mining*, 2020.

C. H. Martin, T. S. Peng, and M. W. Mahoney. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(4122): 1–13, 2021.

Charles H Martin and Michael W Mahoney. Traditional and heavy tailed self regularization in neural network models. In *International Conference on Machine Learning*, pages 4284–4293. PMLR, 2019.

Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165):1–73, 2021.

Stuart Mitchell, Michael OSullivan, and Iain Dunning. Pulp: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, 65:25, 2011.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

Lorenz K. Müller, Philippe Bich, Jiawei Zhuang, Ahmet Çelik, Luca Benfenati, and Lukas Cavigelli. SINQ: Sinkhorn-normalized quantization for calibration-free low-precision LLM weights. *arXiv preprint arXiv:2509.22944*, 2025. URL https://arxiv.org/abs/2509.22944.

David Pisinger and Paolo Toth. Knapsack problems. In *Handbook of Combinatorial Optimization: Volume1–3*, pages 299–428. Springer, 1998.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. *arXiv preprint arXiv:2502.05795*, 2025.

Wei Tao, Haocheng Lu, Xiaoyang Qu, Bin Zhang, Kai Lu, Jiguang Wan, and Jianzong Wang. Moqae: Mixed-precision quantization for long-context llm inference via mixture of quantization-aware experts. *arXiv preprint arXiv:2506.07533*, 2025.

ModelScope-Swift Team. Github-code dataset, August 2024a. URL https://modelscope.cn/datasets/swift/github-code.

Qwen Team. Introducing qwen1.5, February 2024b. URL https://qwenlm.github.io/blog/qwen1.5/.

Qwen Team. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters, February 2024c. URL https://qwenlm.github.io/blog/qwen-moe/.

Keyu Wang, Yangyi Huang, Jiale Kang, David González-Martínez, Weiyang Liu, and Shiwei Liu. DepthBench: Measuring how residual connections enable more computational depth, 2026. URL https://arxiv.org/abs/2609.32534.

Zhichao Wang, Andrew Engel, Anand D Sarwate, Ioana Dumitriu, and Tony Chiang. Spectral evolution and invariance in linear-width neural networks. *Advances in neural information processing systems*, 36:20695–20728, 2023.

Bernard Widrow and István Kollár. *Quantization noise: roundoff error in digital computation, signal processing, control, and communications*. Cambridge University Press, 2008.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR, 2023.

Zhanhao Xie, Yuexiao Ma, Xiawu Zheng, Fei Chao, Wanchen Sui, Yong Li, Shen Li, and Rongrong Ji. Automated fine-grained mixture-of-experts quantization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 27024–27037, 2025.

Shoukai Xu, Shuhai Zhang, Jing Liu, Bohan Zhuang, Yaowei Wang, and Mingkui Tan. Generative data free model quantization with knowledge matching for classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(12):7296–7309, 2023. doi: 10.1109/TCSVT.2023.3279281.

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

Yaoqing Yang, Ryan Theisen, Liam Hodgkinson, Joseph E Gonzalez, Kannan Ramchandran, Charles H Martin, and Michael W Mahoney. Test accuracy vs. generalization gap: Model selection in nlp without accessing training or testing data. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3011–3021, 2023.

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

Zihao Zheng, Xiuping Cui, Size Zheng, Maoliang Li, Jiayu Chen, Yun Liang, and Xiang Chen. Dynamo: Runtime switchable quantization for moe with cross-dataset adaptation. *arXiv preprint arXiv:2503.21135*, 2025.

Yefan Zhou, Tianyu Pang, Keqin Liu, Michael W Mahoney, Yaoqing Yang, et al. Temperature balancing, layer-wise weight analysis, and neural network training. *Advances in Neural Information Processing Systems*, 36:63542–63572, 2023.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. Llama-moe: Building mixture-of-experts from llama with continual pre-training. *arXiv preprint arXiv:2406.16554*, 2024.

Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.nips.cc/paper_files/paper/2018/hash/55a7cf9c71f1c9c495413f934dd1a158-Abstract.html.

Barret Zoph. Designing effective sparse expert models. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1044–1044. IEEE, 2022.

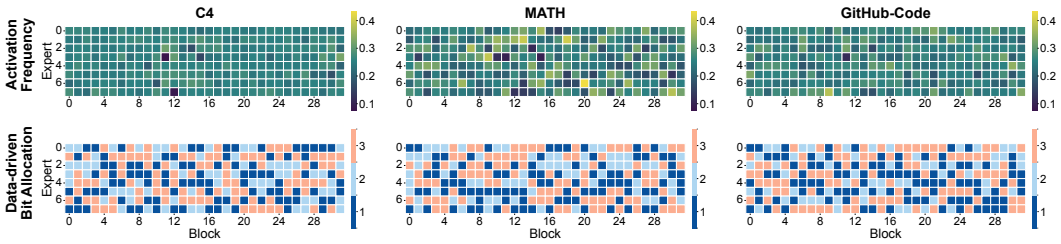


Figure 5: **Domain-dependent expert activation patterns and data-driven bit-width allocation in Mixtral-8×7B.** Activation frequencies (top) and corresponding bit-width allocations (bottom) across different domains (C4, MATH, GitHub-Code), illustrating substantial variations induced by calibration data from different domains.

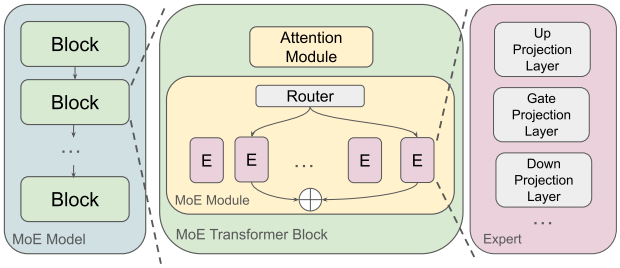


Figure 6: **Hierarchical relationship among blocks, experts, and layers in our paper.** A MoE model consists of multiple Transformer blocks; each block contains an attention module and an MoE module with multiple experts; each expert further comprises multiple layers (e.g. up, gate, and down projection).

A Calibration-dependent Expert Activation and Bit-Width Allocation

To illustrate the influence of calibration datasets on MoE bit-width allocation, we apply PMQ [Huang et al., 2024] to Mixtral-8×7B using datasets from different domains. As illustrated in Figure 5, the collected expert activation frequencies vary significantly as a function of the calibration data. As a result, different calibration-time statistics lead to substantially different data-driven allocations.

B Terminology Summary

To avoid ambiguity, we summarize the terminology used throughout this paper in Figure 6; the descriptions are as follows:

- **Global:** Global refers to the entire model. A global budget means that a single bit budget is defined for the whole model and shared across all blocks, experts, and layers.
- **Block:** A block refers to a Transformer block. In popular MoE models, each block consists of three main modules: an attention module, an MoE module containing multiple experts, and a routing module. A block-wise budget assigns the same bit budget independently to each Transformer block.
- **Expert:** An expert is a feed-forward network within the MoE module. Each expert typically contains multiple layers (e.g., up/gate/down projections). An expert-wise bit allocation means that all submodules within an expert share a single bit setting.
- **Layer:** A layer denotes an individual submodule within a module, such as projection layers (e.g., up, gate, and down projections in experts). A layer-wise bit allocation means assigning independent bit-widths to each submodule.

C Derivation of the Hill Estimator

C.1 Tail Model Assumption

Following Section 3.2, we model the ESD tail by a truncated power-law *density*:

$$p(\lambda) \propto \lambda^{-\alpha}, \qquad \lambda_{\min} < \lambda < \lambda_{\max}, \tag{6}$$

where α is the power-law exponent. Smaller α indicates heavier tails.

C.2 From the Power-Law Density to a Pareto Form

To derive the estimator used in Eq. 3, we rewrite Eq. 6 as a Pareto distribution. Let $\beta = \alpha - 1$. Then the density can be expressed as

$$p(\lambda) = \beta\, \lambda_{\min}^{\beta}\, \lambda^{-(\beta+1)}, \qquad \lambda \geq \lambda_{\min}, \tag{7}$$

where the normalization constant is determined by

$$1 = \int_{\lambda_{\min}}^{\infty} \beta\, \lambda_{\min}^{\beta}\, \lambda^{-(\beta+1)}\, d\lambda. \tag{8}$$

Eq. 7 is the standard Pareto form, and estimating β yields $\alpha = \beta + 1$.

C.3 Likelihood and Log-Likelihood

Let $\lambda_1, \ldots, \lambda_k$ be k independent tail samples with $\lambda_i \geq \lambda_{\min}$. Under equation 7, the likelihood of β is

$$L(\beta) = \prod_{i=1}^{k} \beta\, \lambda_{\min}^{\beta}\, \lambda_i^{-(\beta+1)} = \beta^k \lambda_{\min}^{k\beta} \prod_{i=1}^{k} \lambda_i^{-(\beta+1)}. \tag{9}$$

Taking logs gives

$$\ln L(\beta) = k \ln \beta + k\beta \ln \lambda_{\min} - (\beta + 1) \sum_{i=1}^{k} \ln \lambda_i. \tag{10}$$

C.4 Maximum Likelihood Estimation

Differentiating and setting the derivative to zero,

$$\frac{\partial \ln L}{\partial \beta} = \frac{k}{\beta} + k \ln \lambda_{\min} - \sum_{i=1}^{k} \ln \lambda_i = 0, \tag{11}$$

which yields

$$\frac{1}{\beta} = \frac{1}{k} \sum_{i=1}^{k} \ln\!\left(\frac{\lambda_i}{\lambda_{\min}} \right). \tag{12}$$

Therefore,

$$\hat{\beta} = \left(\frac{1}{k} \sum_{i=1}^{k} \ln\!\left(\frac{\lambda_i}{\lambda_{\min}} \right) \right)^{-1}, \qquad \hat{\alpha} = 1 + \hat{\beta}. \tag{13}$$

In our implementation, we concatenate eigenvalues across submatrices and sort them in ascending order:

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n. \tag{14}$$

We take the top-k eigenvalues $\{\lambda_{n-k+1}, \ldots, \lambda_n\}$ as tail samples and set the lower cutoff as the $(n-k)$-th order statistic, i.e., $\lambda_{\min} = \lambda_{n-k}$. Substituting $\lambda_i \leftarrow \lambda_{n-i+1}$ and $\lambda_{\min} = \lambda_{n-k}$ into Eq. 13 gives the Hill estimator used in Eq. 3:

$$\hat{\alpha} = 1 + \left(\frac{1}{k} \sum_{i=1}^{k} \ln \frac{\lambda_{n-i+1}}{\lambda_{n-k}} \right)^{-1}. \tag{15}$$

Intuition. The estimator is the reciprocal of the average log-excess above the threshold. Heavier tails produce larger log-excess values and smaller $\hat{\alpha}$. Lighter tails produce smaller log-excess values and larger $\hat{\alpha}$.

FARMS-based score computation. To make the spectral estimate comparable across heterogeneous matrix shapes, we compute `PL_Alpha_Hill` with the *Fixed-Aspect-Ratio Matrix Subsampling* (FARMS) heuristic [Hu et al., 2025b]. For each layer, we partition its weight matrix into submatrices with a fixed aspect ratio, form the corresponding Gram matrices, and collect their eigenvalues. We then concatenate these eigenvalues, sort them in ascending order, and apply the Hill estimator in Eq. 15. This yields a more shape-robust estimate of spectral heavy-tailedness.

D More Detailed `PL_Alpha_Hill` Results

Figure 3 summarizes the distribution of `PL_Alpha_Hill` across all up/gate/down projection layers in three representative MoE-LLMs. Based on this overview, we first summarize the observations that motivate global fine-grained bit allocation.

i) Cross-Layer Diversity. Within the same MoE block, layers exhibit distinct `PL_Alpha_Hill` values, demonstrating the necessity of layer-wise mixed precision.

ii) Cross-Block Diversity. Different MoE blocks exhibit distinct `PL_Alpha_Hill` distributions, with some blocks consistently above or below the model average. This indicates that importance varies across depth, so a globally coordinated allocation across blocks is required to use the bit budget effectively.

iii) Cross-Model Diversity. The within-block variance of `PL_Alpha_Hill` differs across architectures: it is relatively small in vanilla MoE models such as Mixtral-8×7B, but substantially larger in fine-grained MoE models such as DeepSeekV2-Lite and Qwen1.5-MoE. This suggests that the mapping from `PL_Alpha_Hill` values to importance should adapt to the model-level `PL_Alpha_Hill` distribution.

We further report the layer-wise `PL_Alpha_Hill` distribution within sampled blocks for four MoE models in Figure 7, including Mixtral-8×7B, Llama-MoE-3.5B [Zhu et al., 2024], DeepSeekV2-Lite, and Qwen1.5-MoE. We also report expert-wise `PL_Alpha_Hill` values for these sampled blocks in Figure 8. For comparison, Figure 9 reports layer-wise `PL_Alpha_Hill` distributions for four non-MoE models, including Llama3-1B [Grattafiori et al., 2024], Llama3-3B, Qwen1.5-4B [Team, 2024b], and Qwen3-4B [Yang et al., 2025]. These more detailed figures reveal three structural patterns.

i) Intra-Block Diversity. Within the same MoE block, the up, gate, and down projection layers exhibit different `PL_Alpha_Hill` values. This suggests that assigning a single bit-width per expert is suboptimal and that finer granularity (e.g., per-layer) is necessary.

ii) Inter-Expert Diversity. Experts within the same MoE layer show clear differences in `PL_Alpha_Hill`, implying that experts contribute unequally to model quality. Therefore, uniform expert precision wastes budget on less sensitive experts and under-protects critical ones.

iii) Module-Type Difference. In MoE models, the up, gate, and down projection layers generally have higher `PL_Alpha_Hill` values than the attention and router modules. This suggests that these MoE projection layers have greater compression potential.

E Importance and Quantization Noise Analysis

To construct the module-level analysis, we sample projection modules and quantize one sampled module at a time to 2-bit while keeping the remaining weights full-precision. We then measure the resulting perplexity increase as the module-level degradation. The quantization-noise axis uses our quantization noise model in Section 3.3. Since smaller `PL_Alpha_Hill` indicates heavier-tailed spectra and higher structural importance, AlphaQ predicts that large degradation should occur when high quantization noise overlaps with low `PL_Alpha_Hill`.

Figure 10 shows the joint view of `PL_Alpha_Hill` and quantization noise at the module level. Panel (a) visualizes this interaction directly, while Panel (b) compares individual and combined

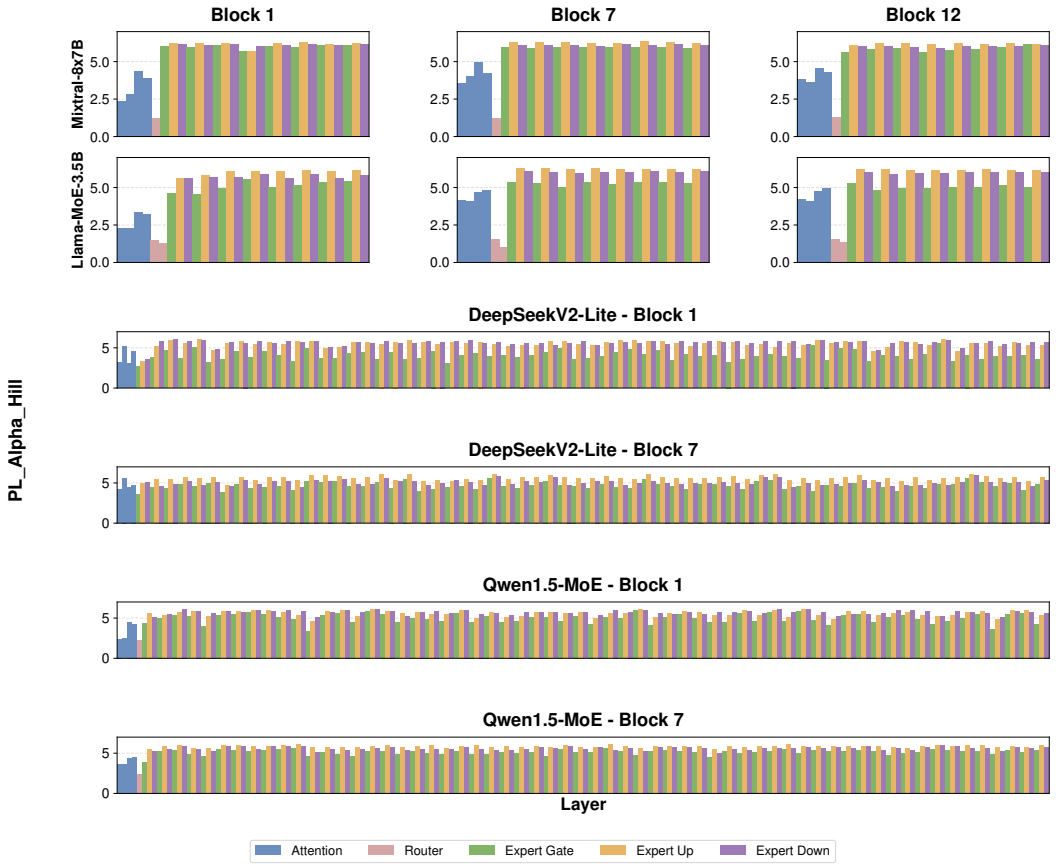


Figure 7: Layer-wise `PL_Alpha_Hill` distribution of sampled blocks in four MoE LLMs

metrics. Large degradation tends to concentrate in the regime of high quantization noise and low `PL_Alpha_Hill`, but the spread of points also indicates that neither signal alone is sufficient. The clearer monotonic trend of Alpha-scaled Quantization Noise supports the form of our objective: quantization noise should be weighted by structural importance rather than used alone. This is consistent with our component ablation in Table 3.

F Justification of the Quantization Noise Model

Here, we provide a theoretical justification for modeling the layer-wise quantization error variance as $\nu_{l,b} \propto 2^{-2b}$. We consider a uniform quantizer applied to the weights of the l-th layer, denoted by $\mathbf{W}_l$. We assume the weights lie within the interval $[-R_l, R_l]$, where R_l is a clipping value determined by the distribution of $\mathbf{W}_l$. For b-bit quantization, the step size $\Delta_{l,b}$ is defined as

$$\Delta_{l,b} = \frac{2R_l}{2^b}. \tag{16}$$

Under Bennett's high-resolution quantization hypothesis [Widrow and Kollár, 2008], when the step size is small enough relative to the signal variation, the quantization error $E = Q(w) - w$ can be approximated as a random variable uniformly distributed over $\left[-\frac{\Delta_{l,b}}{2}, \frac{\Delta_{l,b}}{2}\right]$. For a continuous uniform distribution with width $\Delta_{l,b}$, the variance is $\Delta_{l,b}^2/12$. Consequently, the variance of the quantization error in layer l can be approximated as

$$\mathrm{Var}(\mathbf{E}_{l,b}) \approx \frac{\Delta_{l,b}^2}{12} = \frac{1}{12}\left(\frac{2R_l}{2^b}\right)^2 = \frac{R_l^2}{3} \cdot 2^{-2b}. \tag{17}$$

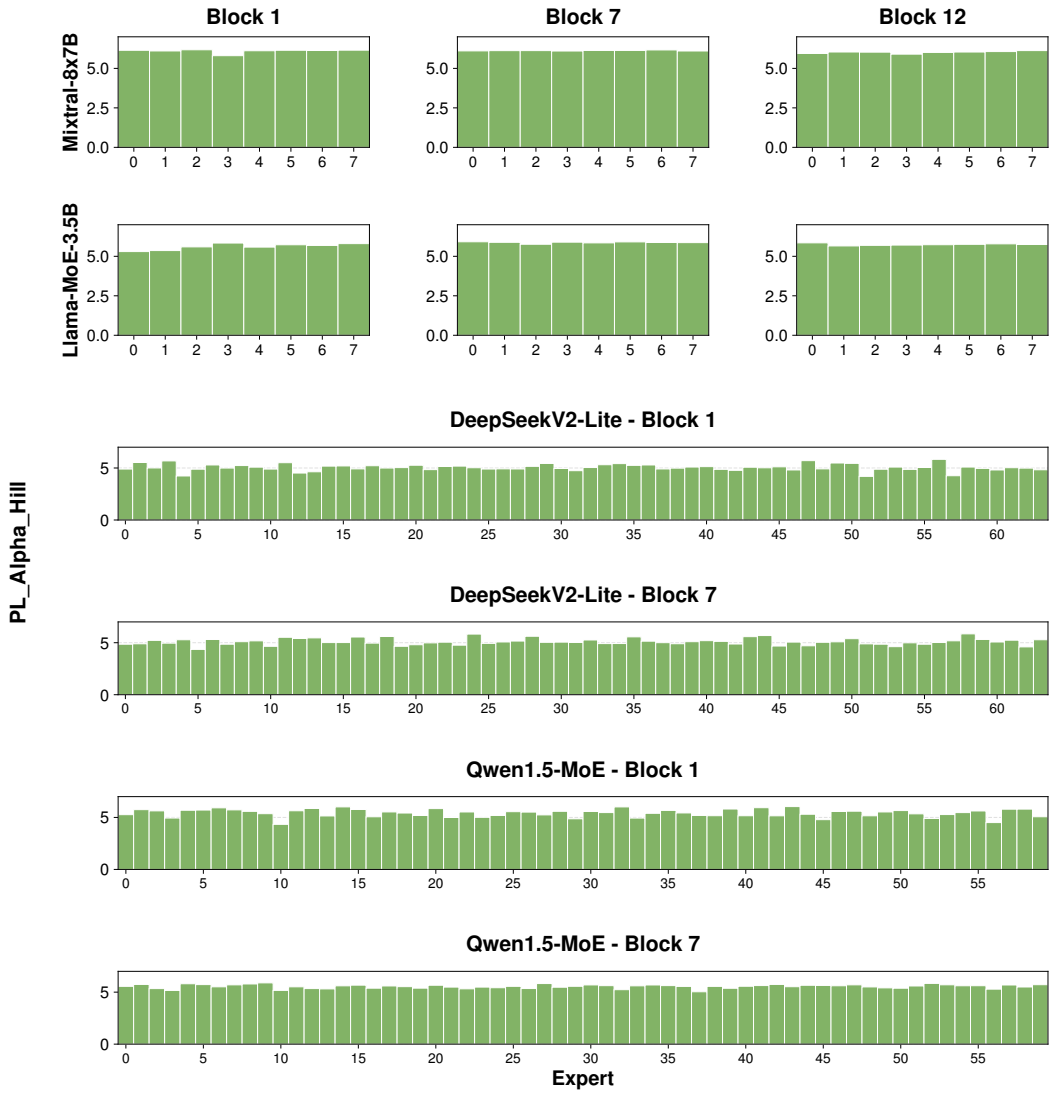


Figure 8: Expert-wise `PL_Alpha_Hill` distribution of sampled blocks in four MoE LLMs.

We therefore define $c_l = R_l^2/3$. Since the clipping range R_l is typically proportional to the standard deviation of the weights, it follows that c_l scales with $\mathrm{Var}(\mathbf{W}_l)$. This leads to the exponential decay model used in the main text,

$$\nu_{l,b} = c_l 2^{-2b}, \tag{18}$$

where c_l captures the layer-specific scale.

G Data-Free Default and Sensitivity of γ

We set γ from model-weight statistics. Let $\alpha_{\min}$, $\alpha_{\max}$, and v_α denote the minimum, maximum, and variance of `PL_Alpha_Hill` over the target modules. Smaller α indicates stronger heavy-tailedness, so $\alpha_{\min}$ and $\alpha_{\max}$ define model-internal high- and low-importance endpoints.

To set the scale of the power-law mapping $w_\gamma(\alpha) = (\tilde{\alpha}/\alpha)^\gamma$, where $\tilde{\alpha}$ is the median of $\{\alpha_l\}_l$, we construct a local discriminative proxy between the two endpoints. Specifically, we introduce two local density surrogates

$$p_{\text{high}}(\alpha) \propto \exp\left(-\frac{(\alpha-\alpha_{\min})^2}{2v_\alpha}\right), \quad p_{\text{low}}(\alpha) \propto \exp\left(-\frac{(\alpha-\alpha_{\max})^2}{2v_\alpha}\right), \tag{19}$$

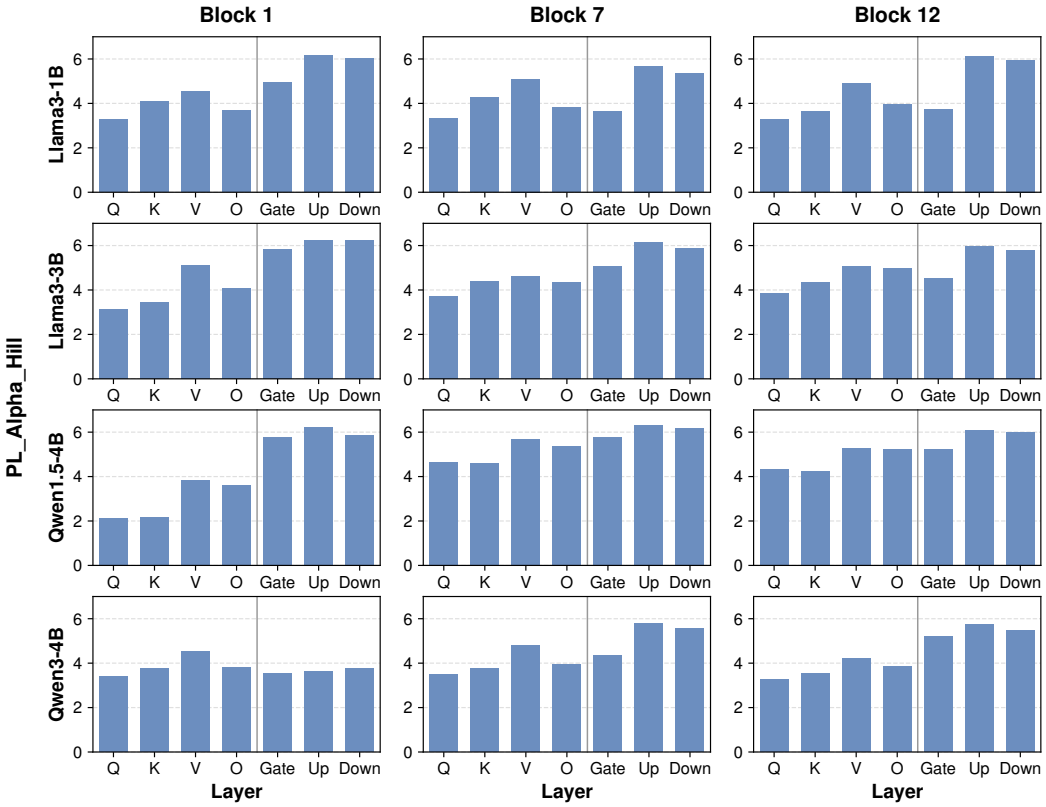


Figure 9: Layer-wise `PL_Alpha_Hill` distribution of sampled blocks in four non-MoE LLMs

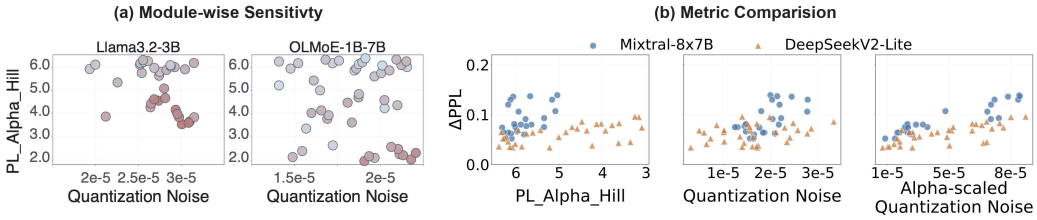


Figure 10: **Module-level relationship between PL_Alpha_Hill, quantization noise, and quantization degradation.** (a) Each point denotes a sampled module from Llama 3.2-3B or OLMoE-1B-7B. The horizontal axis is 2-bit quantization noise, the vertical axis is `PL_Alpha_Hill`, and darker points indicate larger PPL increase after 2-bit quantization. Severe degradation concentrates in the region with high quantization noise and low `PL_Alpha_Hill`. (b) Relationship between PPL degradation and three metrics on sampled modules in Mixtral-8×7B and DeepSeek-V2-Lite: `PL_Alpha_Hill`, Quantization Noise, and Alpha-scaled Quantization Noise. Among the three, Alpha-scaled Quantization Noise shows the clearest monotonic trend with degradation.

which share a common variance scale v_α and are centered at the high- and low-importance endpoints. This construction serves as a local proxy and does not assume a global parametric form of the α distribution.

We define the log-discriminant score

$$s(\alpha) = \log \frac{p_{\text{high}}(\alpha)}{p_{\text{low}}(\alpha)}. \tag{20}$$

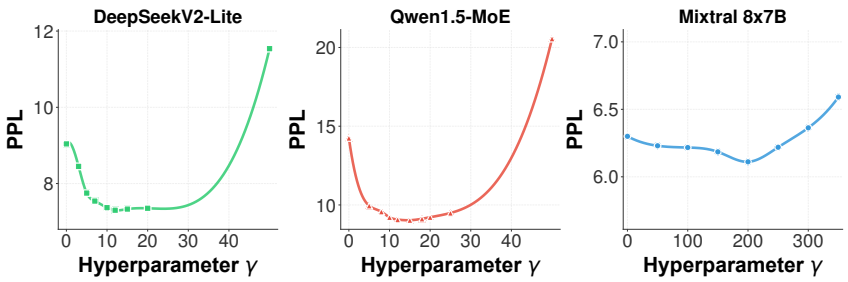


Figure 11: **Sensitivity analysis of** γ. Sensitivity varies across MoE models: DeepSeekV2-Lite and Qwen1.5-MoE are more sensitive to γ than Mixtral-8×7B.

Substituting the surrogate forms gives

$$s(\alpha) = -\frac{(\alpha - \alpha_{\min})^2}{2v_\alpha} + \frac{(\alpha - \alpha_{\max})^2}{2v_\alpha}. \tag{21}$$

Expanding the squares and simplifying yields

$$s(\alpha) = -\frac{\alpha_{\max} - \alpha_{\min}}{v_\alpha} \cdot \alpha + C, \tag{22}$$

where C is a constant independent of α. Therefore,

$$\frac{d}{d\alpha} s(\alpha) = -\frac{\alpha_{\max} - \alpha_{\min}}{v_\alpha}. \tag{23}$$

The discriminant score $s(\alpha)$ defines a data-free reference scale describing how rapidly importance changes between the two endpoints. We use this reference to calibrate the curvature of the power-law mapping $w_\gamma(\alpha)$.

The log-slope of the power-law mapping is

$$\frac{d}{d\alpha} \log w_\gamma(\alpha) = -\frac{\gamma}{\alpha}. \tag{24}$$

We match the local slope of $\log w_\gamma(\alpha)$ to the slope of $s(\alpha)$ at the high-importance endpoint $\alpha_{\min}$, where the mapping is most sensitive:

$$\frac{d}{d\alpha} \log w_\gamma(\alpha)\Big|_{\alpha=\alpha_{\min}} = \frac{d}{d\alpha} s(\alpha). \tag{25}$$

Substituting the two slopes gives

$$-\frac{\gamma}{\alpha_{\min}} = -\frac{\alpha_{\max} - \alpha_{\min}}{v_\alpha}, \tag{26}$$

which yields

$$\gamma_{\text{default}} = \frac{\alpha_{\min}(\alpha_{\max} - \alpha_{\min})}{v_\alpha}. \tag{27}$$

With this default γ, our bit allocation method adapts to different `PL_Alpha_Hill` distributions. Models with concentrated `PL_Alpha_Hill` distributions require larger curvature to expose importance differences, while broader distributions require weaker amplification. Figure 11 reports the sensitivity of AlphaQ to γ across representative MoE models.

H Further Experiment Results

As shown in Tables 6, 7, and 8, we provide more detailed results across models and bit budgets. We compare AlphaQ against PMQ [Huang et al., 2024] and Uniform under four bits-per-layer budget settings: 2.0 / 2.5 / 3.0 / 3.5. For evaluation, we report perplexity (PPL↓) on WikiText2

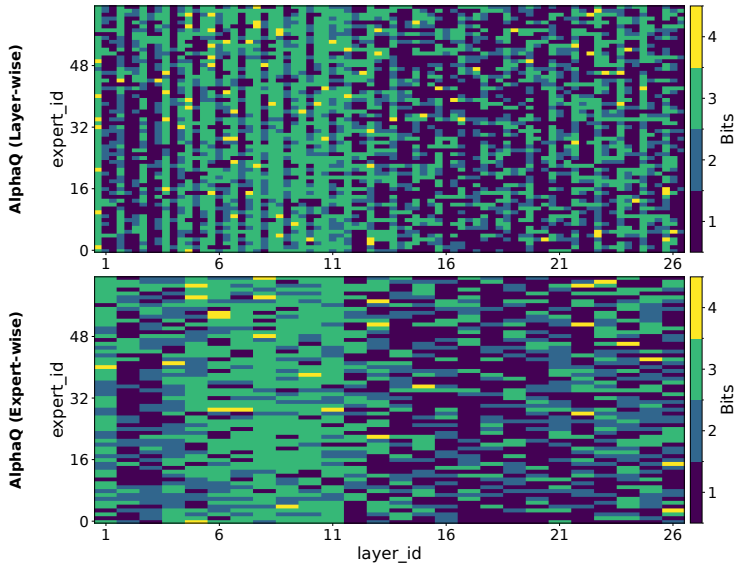


Figure 12: Bit allocation of DeepSeekV2-Lite under a 2-bit budget.

and average zero-shot accuracy (Avg.↑) over six benchmarks: PIQA [Bisk et al., 2020], ARC-Easy, ARC-Challenge [Clark et al., 2018], HellaSwag [Zellers et al., 2019], WinoGrande [Sakaguchi et al., 2021], and BoolQ [Clark et al., 2019] using the EleutherAI LM Harness [Gao et al., 2024].

In Figure 12, Figure 13, and Figure 14, we report detailed bit allocation results from AlphaQ under a 2-bit budget, with both layer-wise and expert-wise settings.

Table 6: Results on DeepSeekV2-Lite. Perplexity↓ on WikiText2 and accuracy↑ on six zero-shot tasks. The best results in each bit-width are highlighted in bold.

Bits	Method	WikiText2↓	PIQA↑	ARC-e↑	ARC-c↑	HellaSwag↑	WinoGrande↑	BoolQ↑	Avg.↑
16	Original	6.31	80.08	76.47	48.98	78.01	71.51	65.00	70.01
2	Uniform	9.57	73.07	60.61	35.84	62.87	62.04	55.03	58.24
	PMQ	7.99	75.14	61.68	37.37	66.20	**68.11**	57.66	61.03
	AlphaQ	**7.30**	**76.03**	**67.20**	**40.33**	**68.70**	67.92	**58.10**	**63.05**
2.5	Uniform	8.14	76.66	71.34	35.84	72.44	69.46	57.58	63.89
	PMQ	6.95	77.86	71.68	37.37	72.76	69.06	60.03	64.79
	AlphaQ	**6.64**	**78.02**	**73.77**	**39.22**	**73.38**	**69.30**	**60.45**	**65.69**
3	Uniform	7.50	78.50	72.51	40.83	74.49	70.11	60.66	66.18
	PMQ	6.70	79.16	73.15	41.44	75.04	70.23	61.53	66.76
	AlphaQ	**6.60**	**79.32**	**73.41**	**41.56**	**75.44**	**70.86**	**61.98**	**67.10**
3.5	Uniform	6.83	79.21	74.56	44.57	76.27	70.83	62.50	67.99
	PMQ	6.57	79.63	75.26	45.83	76.97	71.04	63.21	68.66
	AlphaQ	**6.44**	**79.89**	**75.51**	**46.51**	**77.21**	**71.21**	**63.85**	**69.03**

I Comparison with DynaMo

For reference, we also compare AlphaQ with the recent cross-dataset MoE quantization method DynaMo [Zheng et al., 2025]. Table 9 shows that AlphaQ performs better than DynaMo on the overlapping OLMoE benchmarks. A possible reason is that DynaMo relies on a heuristic token-utilization signal to define channel and expert importance and evaluates cross-dataset adaptation only

Table 7: Results on Qwen1.5-MoE-A2.7B. Perplexity↓ on WikiText2 and accuracy↑ on six zero-shot tasks. The best results in each bit-width are highlighted in bold.

Bits	Method	WikiText2↓	PIQA↑	ARC-e↑	ARC-c↑	HellaSwag↑	WinoGrande↑	BoolQ↑	Avg.↑
16	Original	7.22	80.52	69.23	44.11	77.21	68.59	79.57	69.87
2	Uniform	14.25	66.38	42.89	30.29	49.06	53.12	72.14	52.31
	PMQ	10.47	74.76	59.05	34.56	64.34	64.72	74.52	61.99
	AlphaQ	**9.03**	**76.12**	**61.04**	**36.89**	**66.91**	**67.76**	**76.84**	**64.26**
2.5	Uniform	10.36	71.49	49.07	29.44	53.98	55.96	68.69	54.77
	PMQ	9.01	77.58	60.56	36.26	70.69	65.82	75.54	64.41
	AlphaQ	**8.14**	**78.21**	**66.97**	**42.72**	**71.81**	**66.12**	**78.41**	**67.37**
3	Uniform	9.22	76.59	62.53	39.81	68.50	65.27	74.51	64.53
	PMQ	7.85	79.82	68.19	42.55	76.15	68.22	**77.89**	68.80
	AlphaQ	**7.60**	**80.11**	**68.81**	**43.51**	**76.94**	**68.45**	77.81	**69.27**
3.5	Uniform	8.13	79.31	66.81	41.51	74.52	67.54	76.81	67.75
	PMQ	7.42	80.31	69.02	43.81	77.13	68.50	79.22	69.67
	AlphaQ	**7.38**	**80.46**	**69.15**	**44.33**	**77.70**	**68.55**	**79.45**	**69.86**

Table 8: Results on Mixtral-8x7B. Perplexity↓ on WikiText2 and accuracy↑ on six zero-shot tasks. The best results in each bit-width are highlighted in bold.

Bits	Method	WikiText2↓	PIQA↑	ARC-e↑	ARC-c↑	HellaSwag↑	WinoGrande↑	BoolQ↑	Avg.↑
16	Original	3.84	83.57	83.67	59.30	84.03	76.24	85.35	78.69
2	Uniform	6.30	74.81	71.09	40.44	54.04	65.75	72.57	63.12
	PMQ	6.10	78.29	73.06	48.38	**64.95**	71.27	**80.58**	**69.42**
	AlphaQ	**6.01**	**78.52**	**74.62**	**46.59**	64.76	**71.31**	80.39	69.36
2.5	Uniform	6.16	79.38	77.40	50.26	58.84	73.40	82.01	70.22
	PMQ	5.32	80.36	78.45	51.58	69.23	**73.95**	82.54	72.35
	AlphaQ	**5.17**	**80.84**	**78.87**	**51.59**	**69.24**	73.24	**82.57**	**72.39**
3	Uniform	4.91	80.20	79.55	49.57	70.23	74.19	82.01	72.62
	PMQ	4.41	81.39	**82.53**	53.75	73.03	75.77	83.18	74.94
	AlphaQ	**4.37**	**81.56**	82.32	**54.44**	**73.47**	**77.27**	**83.64**	**75.45**
3.5	Uniform	4.29	81.28	83.04	54.95	73.58	**76.19**	84.36	75.57
	PMQ	4.25	81.21	82.49	53.92	72.90	76.09	83.84	75.08
	AlphaQ	**4.23**	**81.51**	**83.38**	**55.80**	**74.67**	76.10	**84.45**	**75.98**

on two general-purpose corpora (WikiText and C4). By contrast, AlphaQ avoids calibration-driven importance estimation altogether and still transfers better under the same OLMoE 3-bit setting.

J Computation Cost of `PL_Alpha_Hill`

Computing `PL_Alpha_Hill` is an offline preprocessing step, performed once per model before solving the ILP. To quantify this overhead, Table 10 reports the total time and average per-module time for one full alpha-computation pass on OLMoE-1B-7B and Qwen3-30B-A3B. We tested this on a dual-socket AMD EPYC 7713 system (64 cores per socket, SMT enabled, 128 physical cores / 256 logical CPUs, NUMA=2). For both models, computing `PL_Alpha_Hill` takes only a few minutes, which we consider practical for deployment.

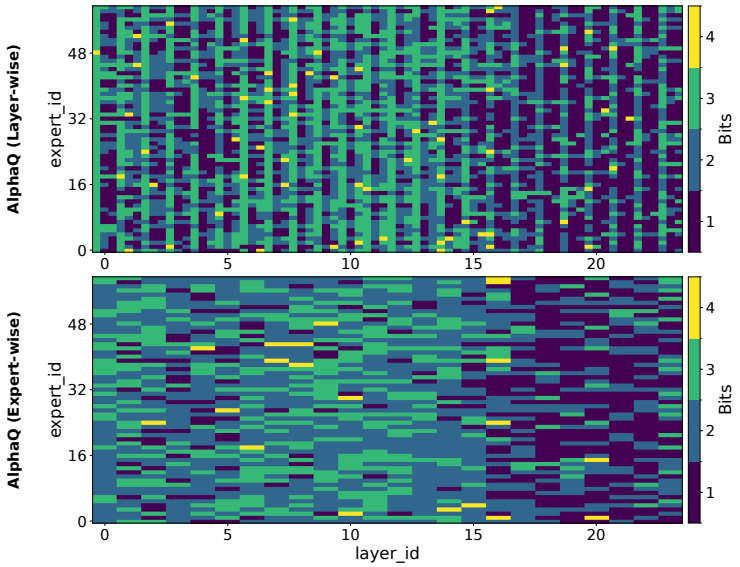


Figure 13: Bit allocation of Qwen1.5-MoE under a 2-bit budget.

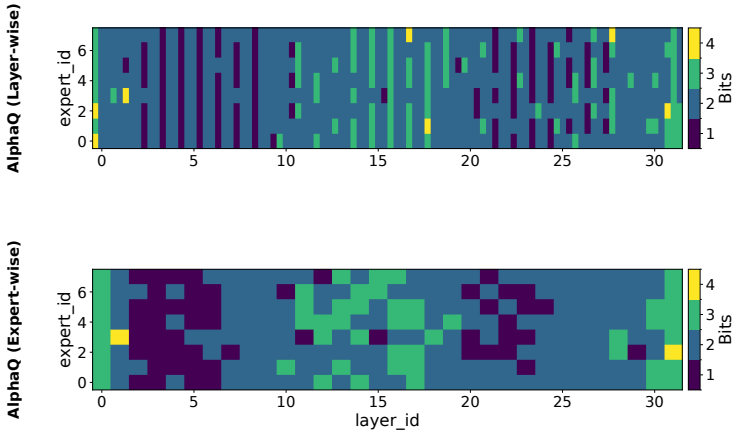


Figure 14: Bit allocation of Mixtral-8×7B under a 2-bit budget.

K Runtime Implementation and Decode Optimization

We use an optimized runtime backend for MoE expert layers. During prefill, we reuse the PMQ-style HQQ [Badri and Shaji, 2023] backend: a CUDA kernel performs bit unpacking and dequantization, followed by Tensor Core GEMM. Profiling shows this decomposition is highly effective in prefill, which is compute-bound, but yields limited speedups in memory-bound decode due to data movement between the dequantization and GEMM; concretely, `Memcpy HtoD` and `aten::copy_` become dominant in decode. We therefore implement decode-specific fused Triton kernels that fuse unpacking, dequantization, and GEMM into a single kernel. Profiling confirms that this change substantially reduces `Memcpy HtoD` and `aten::copy_` overhead during decode.

Optionally, on top of this backend, we cache per-layer quantization metadata on the GPU during token generation, avoiding repeated host-to-device transfers across decode steps. This caching introduces a modest peak-memory increase: under the same WikiText2 benchmark on Mixtral-8×7B, peak GPU memory rises from 13418MiB to 14259MiB (+841MiB, ∼6.1%) for the 2-bit quantized model, which remains acceptable for most GPUs.

Table 9: **Comparison with DynaMo on OLMoE-1B-7B at 3-bit.** DynaMo numbers are taken from the reported results in original paper [Zheng et al., 2025]. Lower is better for WikiText/C4 PPL, and higher is better for downstream accuracy.

Method	WikiText	C4	ARC-Challenge	ARC-Easy	PIQA
DynaMo	9.64	15.31	25.85	43.52	58.87
AlphaQ	**9.19**	**14.64**	**26.91**	**46.11**	**60.08**

Table 10: **Offline computation cost of `PL_Alpha_Hill`.** We report the total time and the average per-module time for one full alpha-computation pass.

Model	# Modules	Total Time (s)	Per-Module (ms)
OLMoE-1B-7B	3152	180.28	57.19
Qwen3-30B-A3B	12448	549.26	44.12

Table 11 reports end-to-end speedups on WikiText alongside decode token throughput. Relative to the PMQ-style backend, the fused kernel improves the decode token rate by nearly 1.5×; adding metadata caching further improves throughput to 10.3 tokens/s.

L Speedup and Memory Compression

For memory compression, as shown in Table 12, AlphaQ effectively addresses the memory bottleneck of MoE models. On Mixtral 8×7B, the parameter memory footprint is reduced from 96.8 GB (BF16) to approximately 13.4 GB under a 2-bit budget (7.2× compression ratio). Under a 3.5-bit budget, AlphaQ achieves a $4.4\times$ compression ratio. With our optimized kernel, AlphaQ keeps a similar memory footprint to PMQ under comparable low-bit budgets, while the measured speedup is slightly higher at both 2-bit and 2.5-bit. Notably, under a 3.5-bit budget, Qwen1.5-MoE reduces the weight memory footprint to approximately one quarter of BF16.

M Discussion

Why expert weights? We target MoEs because they have become a dominant architecture among state-of-the-art LLMs, making their efficient deployment increasingly important. Although sparse routing activates only a few experts per token, all expert weights must remain memory-resident in the deployment setting studied here. In Mixtral-8×7B, for example, expert weights account for over 96% of all model parameters. Quantizing experts therefore addresses the primary memory bottleneck of MoE models.

Moreover, dynamic routing and expert heterogeneity make mixed-precision allocation particularly sensitive to the calibration domain. AlphaQ recognizes this MoE-specific challenge and addresses it through calibration-free expert bit allocation.

Structural importance and quantization sensitivity. The relationship between spectral heavy-tailedness and training sufficiency has been extensively studied in prior HT-SR work [Lu et al., 2024, Martin and Mahoney, 2021]. We evaluate the quantization-specific relationship. In the mixed-precision allocation problem studied here, expert importance is operationalized through relative quantization sensitivity. Under matched quantization noise, an expert layer whose quantization causes greater performance degradation is more important to preserve.

We directly evaluate the quantization-specific relationship. In Appendix E, we quantize one projection to 2 bits while keeping all other modules in full precision. At comparable noise levels, modules with lower `PL_Alpha_Hill` exhibit greater PPL degradation. Under the same 3-bit budget, Table 3 shows that Alpha-only improves over Uniform from 11.78 to 10.03 PPL and from 64.73% to 66.23% average accuracy. Combining alpha with quantization noise further improves the results to 9.19 PPL and 67.11% accuracy, compared with 9.56 PPL and 66.81% accuracy from direct multiplication. These controlled and end-to-end results provide direct empirical support for `PL_Alpha_Hill` as an

Table 11: **WikiText2 decode performance with different backends on Mixtral-8×7B.** The reported gain is measured on the same 2-bit-quantized model.

Decode Backend	Token Rate (token/sec)↑
`Unpack + Dequant + GEMM`	6.43
`Fused Kernel`	9.84
`Fused Kernel + metadata caching`	10.36

Table 12: **Efficiency analysis on Mixtral-8×7B and Qwen1.5-MoE.** We report speedup relative to the BF16 baseline, total model size (Params), and activated parameters during inference.

Model	Method	Budget	Speedup ↑	Params (GB) ↓	Act. Params (GB) ↓
Mixtral-8×7B	BF16	16-bit	1.00×	96.80	26.31
	Uniform	2-bit	1.70×	13.61	3.70
	PMQ	2-bit	1.66×	13.41	3.73
	PMQ	2.5-bit	1.47×	16.24	4.53
	AlphaQ	2-bit	1.69×	13.41	3.75
	AlphaQ	2.5-bit	1.50×	16.30	4.52
	AlphaQ	3-bit	1.31×	19.12	5.27
	AlphaQ	3.5-bit	1.17×	22.03	5.81
Qwen1.5-MoE	BF16	16-bit	1.00×	28.94	5.74
	AlphaQ	2.5-bit	1.22×	6.37	1.16
	AlphaQ	3.5-bit	1.12×	7.60	1.45

importance signal for quantization, without assuming that training sufficiency itself determines expert importance.

AlphaQ explicitly accounts for this sensitivity. In Eq. 4, the first factor, $(\tilde{\alpha}/\alpha_l)^\gamma$, provides a data-free estimate of the relative quantization sensitivity of layer l from its weight structure, while the second, $\mathrm{Var}(W_l)2^{-2b}$, models the quantization noise at bit-width b. Their product is the sensitivity-weighted quantization cost minimized by the ILP. However, AlphaQ does not determine bit-width from heavy-tailedness alone: the ILP jointly optimizes all module–bit choices under a global budget. Therefore, a more heavy-tailed expert is not guaranteed to receive a higher bit-width; the final allocation depends jointly on its heavy-tailedness and quantization noise.

Relation to prior work. HT-SR itself is established, and prior work [Lu et al., 2024] applies `PL_Alpha_Hill` to coarse-grained layer pruning and a simple ranking-based mixed-precision extension for dense LLMs. These findings motivate our work, but applying this theory to quantization is not trivial. **(i)** We identify a MoE-specific problem: input-dependent routing makes calibration-based importance estimates vulnerable to calibration-domain bias. **(ii)** Expert projections have heterogeneous matrix shapes, making their raw spectral scores not directly comparable; AlphaQ therefore uses FARMS to control aspect-ratio effects. **(iii)** Alpha alone does not capture the numerical perturbation associated with each layer–bit choice. AlphaQ combines the structural signal with the layer- and bit-dependent noise term $\mathrm{Var}(W_l)2^{-2b}$ and globally optimizes all projection–bit choices under a fixed budget.

At the bit-allocation level, OMPQ [Ma et al., 2023a] uses a network-orthogonality proxy and linear programming to select mixed-precision configurations. AlphaQ instead combines an HT-SR-based structural signal with layer–bit quantization-noise costs and optimizes expert precision globally to address MoE heterogeneity and calibration-domain bias.

Quantizer design offers another complementary direction. AffineQuant [Ma et al., 2024a] reduces quantization error through equivalent affine transformations, while DuQuant [Lin et al., 2024a] redistributes activation outliers using rotations and permutations. DuQuant++ [Lin et al., 2026] extends fine-grained rotation to microscaling FP4 quantization. Outlier-aware Slicing [Ma et al., 2024b] studies reconstruction granularity in vision transformers, and the analysis of PTQ oscillation by Ma et al. [2023b] connects reconstruction behavior to module capacity. These methods improve the representation or reconstruction used by a quantizer, whereas AlphaQ selects expert projection

bit-widths. Combining transformation-based quantizers with AlphaQ would require rechecking the spectral statistics and noise costs used for allocation.

Beyond MoE quantization, broader model-compression research has studied both importance-aware component selection and reducing dependence on real calibration data. Discrimination-aware channel pruning [Zhuang et al., 2018] ranks channels using discriminative loss and reconstruction error. Probabilistic-programming-based PTQ [Liu et al., 2025] estimates quantization parameters from a small calibration set, whereas KMDFQ [Xu et al., 2023] generates proxy data through knowledge matching for data-free quantization. These methods address pruning or quantizer construction. AlphaQ instead derives importance directly from pretrained expert weights and allocates bit-widths across heterogeneous MoE layers without real or generated calibration data. AlphaQ is therefore complementary to these broader compression approaches.

Deployment scope. If proprietary data accurately represents a known and stable deployment domain, calibration-based allocation can specialize the model toward that domain and may achieve better matched-domain performance. However, this is a narrower objective than the general-purpose MoE deployment targeted by AlphaQ. AlphaQ instead avoids domain-specific routing statistics and preserves a domain-agnostic allocation. We do not claim that AlphaQ must outperform deliberate specialization to a perfectly known domain; it targets the broader setting where inputs are heterogeneous, unknown, or subject to distribution shift. Appendix Q examines the distinction between static weight structure and domain-dependent routing.

N Average-Bit Accounting

Consistent with Eq. 5, the reported average-bit budget is the parameter-count-weighted mean over the L expert linear layers participating in mixed-precision allocation:

$$B_{\text{avg}} = \frac{\sum_{l=1}^{L} N_l b_l}{\sum_{l=1}^{L} N_l}, \tag{28}$$

where N_l and b_l denote the parameter count and assigned bit-width of layer l, respectively. In our experiments, all expert linear layers participating in mixed-precision allocation have the same parameter count, so this parameter-weighted mean is equivalent to the arithmetic mean of their bit-widths.

Quantization metadata, including scales and zero-points, is excluded. Non-expert layers, including attention and router layers, are uniformly quantized to 4 bits and are also excluded from this average. Thus, an average expert bit-width is not the same as a whole-model storage cost.

This accounting is identical for all compared methods in the main experiments. Specifically, all methods use the same group-wise asymmetric GPTQ configuration with group size 128 and the same 4-bit setting for non-expert layers. Therefore, the reported budgets provide fair comparisons of mixed-precision allocation over expert linear layers. Additional experiments below state their different backends and candidate sets explicitly.

O Validation Across Quantization Backends

Noise-model fidelity. Grouped asymmetric GPTQ does not strictly follow our quantization noise model, and AlphaQ is formally optimal for the objective with our quantization noise model, rather than with the exact GPTQ error. The ILP primarily depends on the *relative costs* among layer–bit candidates. Our model provides an efficient, data-free approximation of this relative structure without running GPTQ during allocation.

We verified this directly on OLMoE-1B-7B using three distinct backends: grouped asymmetric RTN, GPTQ [Frantar et al., 2022], and HQQ [Badri and Shaji, 2023]. We sampled 192 expert projections from eight fixed blocks and quantized each projection to 2, 3, and 4 bits with group size 128. To exclude correlations dominated by the shared 2^{-2b} factor, Table 13 reports the correlation between our noise model and measured weight MSE separately at each bit-width.

For all three quantizers, the measured error decreased monotonically with bit-width for all 192 projections. The fitted $\log_2$-error slopes are close to the surrogate's theoretical slope of -2, and the

Table 13: **Noise-surrogate validation on OLMoE-1B-7B.** Correlations ρ are computed within each bit-width. Agreement compares module bit assignments obtained with analytical versus measured noise costs under the same 3-bit budget.

Quantizer	ρ_{2b}	ρ_{3b}	ρ_{4b}	$\log_2$-error slope	ILP agreement
RTN	0.995	0.997	0.996	-2.00	99.5%
GPTQ	0.886	0.932	0.944	-2.12	97.4%
HQQ	0.991	0.988	0.988	-2.08	99.0%

high within-bit correlations show that the surrogate largely preserves the layer-wise error ranking despite backend-specific differences in absolute error.

More directly, under the same AlphaQ pipeline and 3-bit budget, allocations obtained using our analytical noise model agreed with those obtained using measured RTN, GPTQ, and HQQ errors on at least **97.4%** of module assignments. For the full OLMoE-1B-7B model, computing all module–bit noise costs with our model took less than 1 second, whereas constructing the corresponding candidate errors with GPTQ by quantizing every module–bit candidate took 32 minutes. Thus, our model avoids exhaustive candidate quantization while producing nearly identical allocations at substantially lower cost.

End-to-end accuracy and calibration-free weights. AlphaQ uses $\mathrm{Var}(W_l)2^{-2b}$ as a calibration-free quantization noise model for the relative costs of module–bit candidates, rather than to reproduce each quantizer's absolute error. We directly evaluated its cross-quantizer generalization by applying the same AlphaQ allocation to RTN, GPTQ, and HQQ on OLMoE-1B-7B under a 3-bit budget (Table 14).

Table 14: **MMLU accuracy (%) across quantizers on OLMoE-1B-7B at 3 bits.** PMQ uses C4 for allocation. AlphaQ+HQQ requires no offline calibration data for allocation or weight quantization.

Quantizer	Uniform	PMQ (C4)	AlphaQ
RTN	31.35	32.03	**33.91**
GPTQ	43.56	43.80	**44.75**
HQQ	42.95	43.11	**43.81**

AlphaQ consistently improves MMLU accuracy over both Uniform and PMQ (C4), despite the different absolute errors produced by these quantizers. These results demonstrate that the simplified noise model provides a transferable allocation surrogate across RTN, GPTQ, and HQQ in this setting.

AlphaQ can be directly combined with calibration-free quantizers to make the entire weight-quantization pipeline calibration-free. Specifically, AlphaQ determines the bit-width allocation without data, after which methods such as HQQ and SINQ [Müller et al., 2025] construct quantized weights using only the weights. GPTQ is used in our main experiments to ensure fair comparison with prior allocation methods, but it is not required by AlphaQ. We validate this combination with HQQ: AlphaQ+HQQ achieves **43.81%** MMLU accuracy without any calibration data, outperforming Uniform+HQQ (**42.95%**) and PMQ+HQQ (**43.11%**), while remaining only 0.94 percentage points below AlphaQ+GPTQ.

P Composition with Activation and KV-Cache Quantization

AlphaQ allocates bit-widths, HQQ quantizes the static expert weights, and activations are quantized dynamically, while KIVI [Liu et al., 2024d] quantizes KV-cache activations online. None of these stages requires offline calibration data.

We evaluated this combination on OLMoE-1B-7B. AlphaQ selected from $\{2, 3, 4\}$ bits under an average 3-bit budget, and HQQ produced the corresponding quantized weights. Inputs to all linear layers were dynamically quantized using per-token symmetric INT8 quantization (A8). During inference, 2-bit KIVI quantized keys per channel and values per token. We evaluated the resulting $2 \times 2 \times 2$ configurations on 128 WikiText-2 sequences (Table 15).

Table 15: **Weight, activation, and KV-cache composition on OLMoE-1B-7B.** WikiText-2 PPL (↓) on 128 sequences. A8 denotes dynamic per-token symmetric INT8 quantization; cache quantization uses 2-bit KIVI.

Weights	Activations	FP16 cache	2-bit cache
FP16	FP16	14.62	15.42
FP16	A8	15.35	15.50
AlphaQ+HQQ	FP16	18.37	19.60
AlphaQ+HQQ	A8	18.33	18.87

Adding A8 to AlphaQ+HQQ changed PPL only from 18.37 to 18.33, introducing no additional quality loss in this experiment. These results demonstrate that AlphaQ can be directly composed with dynamic activation quantization and activation quantization does not significantly amplify the weight-quantization error in our experiments. Meanwhile, expert-weight and final KV-cache memory were reduced by $4.87\times$ and $6.40\times$, respectively.

HQQ, A8, and KIVI can therefore be directly composed as calibration-free weight and activation quantization stages. The dominant quality loss in this setting came from weight quantization, which is our target task.

Q Spectral Structure, Routing, and Calibration-Dependent Allocations

Heavy-tailedness and activation frequency. We examined whether the relationship between spectral heavy-tailedness and training sufficiency is reflected in router behavior. For each of the 1,024 block–expert pairs in OLMoE-1B-7B, we averaged `PL_Alpha_Hill` over its gate, up, and down projections, and compared the 50 lowest- and 50 highest-scoring experts. Using an equal number of sequences from WikiText-2, MATH, and GitHub-Code, the low-alpha group, corresponding to greater HT-SR-inferred training sufficiency, was selected for **16.78%** of tokens per expert on average, compared with **8.76%** for the high-alpha group (Table 16). Thus, experts with greater HT-SR-inferred training sufficiency are selected more frequently on average. Together with the quantization-sensitivity results in Appendix E, this provides quantitative support for the relationship between training sufficiency and expert importance.

Table 16: **Spectral groups and routing in OLMoE-1B-7B.** Expert scores average gate/up/down projection scores. Mean per-expert activation frequency is measured using equal numbers of sequences from three domains.

Group	Number of block–expert pairs	Mean activation frequency
Lowest α (heavier-tailed)	50	16.78%
Highest α (lighter-tailed)	50	8.76%

However, this relationship is not strictly monotonic. Some structurally significant experts may still be rarely activated under this input distribution. For example, Block 7 / Expert 30 had one of the lowest α values in the entire model, yet was selected by only **4.69%** of tokens.

`PL_Alpha_Hill` is a static metric of an expert's weights, whereas activation frequency depends on the input domain. For example, Block 10 / Expert 54 has the smallest `PL_Alpha_Hill` in its block, but is activated by only 0.27% of GitHub-Code tokens, compared with 26.35% on WikiText-2. Moreover, Block 0 / Expert 57 is activated by only 1.63% of MATH tokens, yet has the highest average routing weight when selected among all experts in its block. Thus, an expert with a pronounced heavy-tailed structure can be rarely activated in a particular domain, and activation frequency alone does not fully characterize its importance.

Is AlphaQ a median of calibration-based allocations? The allocation results show that AlphaQ does not behave entirely like a median-based solution among calibration-based allocations. On OLMoE-1B-7B, we compare AlphaQ allocations with PMQ allocations using WikiText-2, MATH, and GitHub-Code. For each expert, we defined the calibration-derived interval using the minimum

and maximum bit-widths assigned across the three datasets, and then checked whether AlphaQ's allocation fell within this interval (Table 17).

Table 17: **AlphaQ assignments relative to PMQ calibration-derived intervals.** OLMoE-1B-7B, using three calibration domains.

Average expert bits	Within interval	Within (%)	Outside (%)
2.5	761 / 1,024	74.3	25.7
3.0	743 / 1,024	72.6	27.4

At 2.5- and 3-bit budgets, respectively, 25.7% and 27.4% of AlphaQ's assignments fell outside the ranges produced by all three calibration datasets. Most assignments lie within the calibration-derived interval, but interval membership does not imply equality to the median. These results therefore do not establish that AlphaQ is a median aggregation of calibration-based allocations. The counts here are expert-level counts, distinct from the projection-level ILP agreement in Table 13.

Stable and domain-dependent assignments. We analyze PMQ allocations for 1,024 block–expert pairs in OLMoE-1B-7B under a 2.5-bit budget, using WikiText-2, MATH, and GitHub-Code as calibration datasets. Stable assignments are uncommon, whereas strong dataset dependence is widespread: 255 experts (24.9%) receive identical bit-widths across all three datasets, including 110 consistently high-bit and 145 consistently low-bit experts. In contrast, 715 experts (69.8%) differ by at least 2 bits; 639 experts (62.4%) switch between 1 and 4 bits. The remaining 54 experts (5.3%) vary by 1 bit. Representative dataset-dependent experts closely follow domain-specific routing (Table 18).

Table 18: **Examples of domain-dependent PMQ allocations on OLMoE-1B-7B at 2.5 bits.** Indices are zero-based. Entries follow WikiText-2 / MATH / GitHub-Code order.

Expert	PMQ bits	Routing frequency (%)
Block 10 / Expert 54	4 / 1 / 1	26.35 / 0.47 / 0.27
Block 13 / Expert 20	1 / 4 / 1	10.04 / 52.62 / 1.75
Block 4 / Expert 55	1 / 1 / 4	1.03 / 2.24 / 63.92

The 110 consistently high-bit experts are potential domain-general candidates, while the dataset-dependent experts are potential domain-specific candidates. In each representative case, the expert receives its highest precision on the dataset where it is routed most frequently. These results show that calibration-based allocations largely reflect domain-dependent routing, whereas AlphaQ provides an independent structural allocation derived from static weights.

R Vision and Multimodal Evaluation

We evaluated `SigLino-MoE` on DTD and `DeepSeek-VL2-Tiny` on ScienceQA-IMG [Lu et al., 2022] under a 3-bit expert budget. Both Uniform and AlphaQ used asymmetric group-wise RTN with group size 128. AlphaQ allocated bit-widths to expert linear layers, while all other weights remained at 4 bits. Table 19 reports the results.

Table 19: **Initial cross-modal results at a 3-bit expert budget.** Accuracy (%); Δ denotes percentage-point improvement over Uniform.

Model	Task	Uniform	AlphaQ	Δ
SigLino-MoE	DTD	67.95	**69.81**	+1.86
DeepSeek-VL2-Tiny	ScienceQA-IMG	81.24	**82.56**	+1.32

The results show clear cross-modal generalization in the tested settings: AlphaQ improves over uniform quantization on both the vision MoE and the multimodal MoE, by 1.86 and 1.32 percentage points, respectively. These results provide initial evidence that AlphaQ can transfer to vision MoE and multimodal MoE.

S Limitation

This work still has several limitations.

First, we focus solely on reducing domain bias in bit allocation. In quantization, data-dependent procedures that are orthogonal to allocation, such as error compensation in GPTQ, can still inject calibration-induced domain bias; we leave that to future work.

Second, we do not provide direct comparisons with all existing MoE quantization and compression methods, which may limit a comprehensive assessment of the method's relative advantages. We leave these directions for future work and plan to further extend AlphaQ to broader MoE architectures and more comprehensive empirical studies.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the paper's scope and main contributions, including calibration-free bit allocation for MoE quantization, global allocation, and an optimized quantized inference backend. The claims are supported by the empirical results in Section 4 and scoped by the limitations section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The paper includes a dedicated Limitations section (Appendix S) discussing the scope of weight bit allocation, activation/cache composition, the HT-SR importance proxy, cross-modal evidence, calibration effects, and incomplete comparison coverage.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes].

Justification: We state the assumptions behind the analytical components and provide derivations for the Hill estimator, the quantization-noise model, and the default γ choice in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: Section 4 specifies the evaluated models, quantization setup, bit budgets, baselines, and benchmarks, while the appendix provides additional ablations, detailed results, efficiency measurements, and derivations needed to reproduce the main claims.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: Public code and usage instructions for the main bit-allocation and quantization pipeline are available at `https://github.com/Superone77/AlphaQ`. The additional analyses and their reported settings are described in Appendices M–R; this revision does not claim a complete executable reproduction package for every additional experiment.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The experimental setting section specifies the models, quantization method, calibration sample count for GPTQ, group size, bit-width candidates, average bit budgets, baselines, and evaluation benchmarks. Additional details and expanded tables are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No].

Justification: We do not report error bars because the main experiments are deterministic post-training quantization and benchmark evaluations under fixed settings. The paper instead reports exact outcomes for various fixed configurations.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The paper reports the NVIDIA A40 GPU used for inference-efficiency measurements and the dual-socket AMD EPYC 7713 CPU system used for `PL_Alpha_Hill` computation. The appendix reports alpha-computation time and per-module cost, and the efficiency section reports runtime and memory measurements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The research uses public models, public benchmarks, and standard post-training quantization/evaluation procedures. It does not involve human subjects, private user data, or unsafe data collection.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No].

    Justification: The paper does not include a standalone broader-impact discussion. The expected positive impact is reducing inference memory and compute cost for MoE models, while generic risks of deploying compressed LLMs remain similar to those of the underlying models.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release a new pretrained language model, image generator, scraped dataset, or other high-risk asset. The planned release is code for quantization, bit allocation, and evaluation.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: Existing models, datasets, evaluation tools, and quantization methods used in the paper are cited in the relevant sections. The released code package will document external dependencies and asset licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The new asset introduced by this work is the code and scripts for AlphaQ bit allocation, quantization, evaluation, and plotting. These assets will be released with documentation and reproduction instructions in the supplementary material and public repository once this paper is released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: LLMs are the target models evaluated in this work, not a non-standard component used to develop the core method. Any use of LLMs for writing, editing, or formatting does not affect the methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.