ALPHAQ: CALIBRATION-FREE BIT ALLOCATION FOR MIXTURE-OF-EXPERTS QUANTIZATION

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

ABSTRACT

Mixture-of-Experts (MoE) architectures scale computation via sparse expert activations, yet they remain memory-bound because all expert weights must reside in memory. Mixed-precision quantization can substantially reduce this footprint, but existing quantization methods estimate expert importance and assign bits based on calibration data. For frontier MoE LLMs, however, the original training data (and thus its true training distribution) is proprietary and inaccessible. Thus, any calibration set is at best a surrogate and may yield a biased or incomplete view of expert utilization, leading to suboptimal bit allocation. To address these problems, we propose **AlphaQ**, a *novel calibration-free bit-allocation method for MoE quantization*. AlphaQ is inspired by *Heavy-Tailed Self-Regularization (HT-SR) theory*; and it is based on a simple but effective principle: experts with more heavy-tailed weight spectra tend to be better trained, and therefore merit higher bits, and vice versa. We find that different MoE variants can exhibit substantial cross-expert quality variability, calling for a nuanced bit allocation, that is difficult to achieve with limited/biased calibration data. By leveraging HT-SR theory, AlphaQ incorporates expert-wise spectral heavy-tailedness and formulates mixed-precision quantization as a budget-constrained optimization problem that minimizes total quantization error under a global bit-budget constraint. Empirical results on DeepSeekV2-Lite, Qwen1.5-MoE, and Mixtral-8$\times$7B show that AlphaQ consistently outperforms calibration-based baselines under matched bit budgets. Notably, on Qwen1.5-MoE, AlphaQ achieves near full-precision accuracy, with an average expert precision of **3.5** bits, while delivering more than **4**$\times$ memory compression.

1 INTRODUCTION

Mixture-of-Experts (MoE) (Jacobs et al., 1991; Jordan & 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 & 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), 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 et al., 2025; Anonymous, 2025; Xie et al., 2025). While effective in practice, these approaches suffer from two fundamental limitations: local optimality and dependence on calibration data.

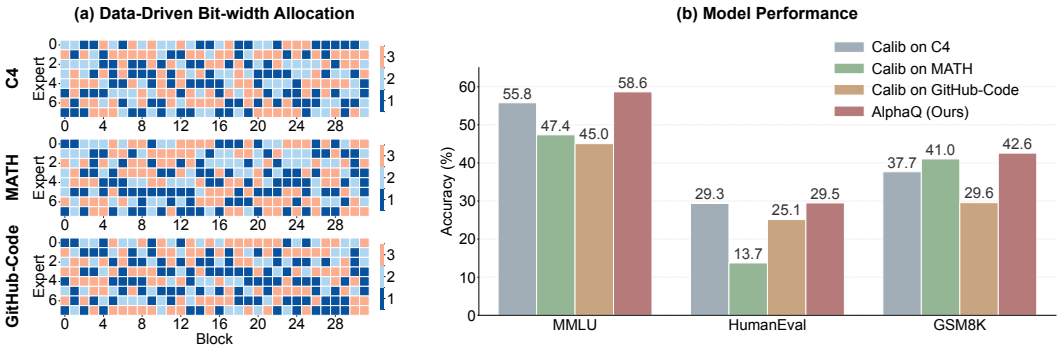

Figure 1: **Domain bias introduced by data-driven bit-width allocation in Mixtral-8×7B.** (a) Bit-width allocations calibrated on datasets across domains (C4 (Raffel et al., 2020), MATH (Hendrycks et al., 2021b), GitHub-Code (Team, 2024a)), illustrate calibration data-induced variations. (b) Mixtral-8×7B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

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 (Anonymous, 2025).

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 couple the performance of quantized models to the calibration distribution, ultimately leading to poor cross-domain generalization.

Figure 1 illustrates this effect on Mixtral-8×7B: models calibrated on different domain-specific datasets exhibit various data-driven bit-width allocation (expert activation patterns are provided in Appendix A.1), 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 & Mahoney, 2019), which substantiates the principle that experts with heavier-tailed weight spectra are empirically linked to more informative structure and stronger correlations (Martin & Mahoney, 2020; Martin et al., 2021; Martin & 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.

One primary contribution of AlphaQ is a unified, data-agnostic importance metric for quantifying expert disparity across MoE architectures and layers. Using this metric, we find importance diversity at three granularities: across MoE families, within a model and within a block. Notably, we also observed that, fine-grained, highly sparse MoEs (e.g., DeepSeekV2-Lite (Liu et al., 2024a),

[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.

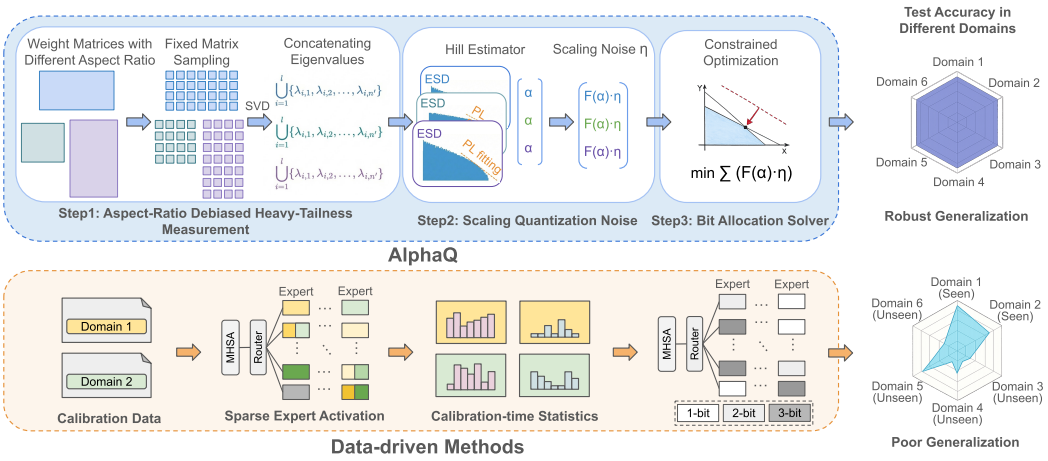

Figure 2: Comparison of the proposed AlphaQ framework and data-driven (or data-dependent) methods for MoE quantization.

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

Building on these observations, AlphaQ introduces a novel bit-allocation approach that formulates mixed-precision quantization as a budget-constrained optimization problem based on our importance metric. 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 substantially reduces memory usage, with minimal accuracy loss, outperforming previous mixed-precision PTQ approaches. For instance, quantizing Qwen1.5-MoE from 16-bit to an average expert bit-width of 2.5 reduces the model size from 28 GB to 6 GB, while the average accuracy on five zero-shot downstream tasks drops by only 2.5 percentage points. 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., 2024; 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; Anonymous, 2025; 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 (Anonymous, 2025) 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) proposed an expert-balanced sampling strategy for calibration, which improves traditional quantization methods in MoE but does not fix the limitation of calibration in bit allocation. Our proposed AlphaQ derives expert importance directly from model weights, enabling 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 & Mahoney, 2019; 2020; Martin et al., 2021; Martin & Mahoney, 2021). Rooted in Random Matrix Theory (Couillet & Liao, 2022) and statistical physics (Engel & 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 effective model diagnostics (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 present `PL_Alpha_Hill`, a calibration-free metric for measuring expert importance, together with an aspect-ratio-aware adaptation that mitigates shape-induced bias. We then describe our global, budget-constrained bit-allocation solver and detail how it computes mixed-precision assignments 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 A.2.

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 & 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:

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

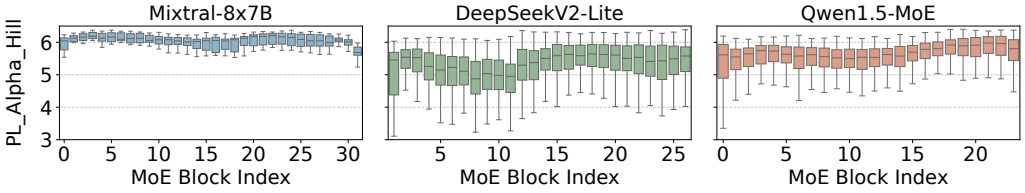

Figure 3: **Distribution of `PL_Alpha_Hill` across all up/gate/down projection in three representative MoE-LLMs.** The bottom and top of each boxplot indicate the minimum and maximum values of `PL_Alpha_Hill` across all up/gate/down projection within the block. The lower and upper edges of the box correspond 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 concatenated eigenvalues 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 A.3. We select k using the *Fix-finger* method (Yang et al., 2023), which aligns $\lambda_{\min}$ with the peak of the ESD. Notably, `PL_Alpha_Hill` is calibration-free and globally comparable across layers.

Mitigating Aspect-Ratio Bias. It is known that spectral shape metrics are systematically affected by the aspect ratio of the weight matrix (Hu et al., 2025b), often assigning larger scores to more "rectangular" matrices, even when the underlying training sufficiency is comparable. To make our estimate robust across varying matrix shapes, we compute the layer-wise tail exponent using the *Fixed-Aspect-Ratio Matrix Subsampling* (FARMS) heuristic (Hu et al., 2025b). Specifically, for each layer l, we partition its weight matrix into a set of submatrices with a fixed aspect ratio. For each submatrix, we form the associated Gram (correlation) matrix and compute its eigenvalues. Then we concatenate all eigenvalues, sorted in ascending order, and calculate `PL_Alpha_Hill` with Equation 3. We note that AlphaPruning (Lu et al., 2024) also proposes a bit-allocation strategy based on `PL_Alpha_Hill`, but it does not account for this aspect-ratio effect, which can lead to suboptimal allocations (please refer to Appendix A.4).

3.3 Bit Allocation Optimization

After calculating layer importance, we perform bit allocation to minimize accuracy degradation under a given bit budget.

We first analyze popular MoE models using `PL_Alpha_Hill`. Figure 3 shows the distribution across all layers within each MoE block for three models, with extended statistics in Appendix A.5. The alpha statistics reveal strong structured diversity that motivates 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. Expert-level analysis is provided in Appendix A.5.

ii) Cross-Block Diversity. Different MoE blocks exhibit distinct `PL_Alpha_Hill` distributions, with some consistently above or below the model average, requiring globally coordinated allocation across depth.

iii) Cross-Model Diversity. Within-block variance differs across architectures: relatively small in vanilla MoE (e.g., Mixtral-8×7B) but substantially larger in fine-grained MoE (e.g., DeepSeek-V2-Lite, Qwen-1.5-MoE), necessitating adaptive importance mapping.

These observations motivate us to allocate bits with fine granularity across diverse layers and blocks under a global budget.

However, importance alone is insufficient. Bit allocation requires quantifying both the benefit of higher precision and the cost of reduced precision elsewhere (an ablation study is provided in Appendix A.6). While `PL_Alpha_Hill` provides calibration-free importance estimates, it does not capture the magnitude of quantization noise at a given bit-width. Weights with similar importance may induce substantially different accuracy degradation due to differences in scale and distribution. We therefore explicitly incorporate quantization noise into our objective, optimizing importance-weighted quantization error under a global bit-budget constraint.

Modeling the Objective Function. The magnitude of weight quantization noise is determined by 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 A.7. To account for layer-wise differences in weight distribution, we modulate this term by the variance of the weight matrix $\mathrm{Var}(\mathbf{W}_l)$. We thus use $\mathrm{Var}(\mathbf{W}_l)2^{-2b}$ to quantify layer-wise quantization noise.

As discussed above, it is necessary to scale the quantization noise with importance. The more important a layer is, the more its quantization noise should be amplified in our optimization objective. Specifically, we scale layer-wise quantization noise with the factor $(\tilde{\alpha}/\alpha_l)^{\gamma}$, where we choose $\tilde{\alpha} = \mathrm{median}\{\alpha_l\}_{l=1}^{L}$ as the normalization factor and the tunable hyperparameter γ to control the curvature of the importance mapping (a sensitivity analysis and tuning strategy for γ are provided in Appendix A.8). Therefore, the scaled quantization noise $\eta_{l,b}$ of a b-bit quantized layer l is

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

And our optimization goal is to minimize the sum of scaled quantization noise across all layers.

Formulating the Constraints. 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}.

Solving Constrained Optimization. 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. equation 5 provides an optimal bit assignment that minimizes the total importance-weighted quantization noise, and thus the accuracy degradation. Eq. equation 5 is a standard instance of the Multiple-Choice Knapsack Problem (Pisinger & 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 three representative MoE LLMs: DeepSeekV2-Lite (Liu et al., 2024a), Qwen1.5-MoE (Team, 2024c), and Mixtral-8×7B (Jiang et al., 2024). For quantization, we follow

Table 1: **Results on DeepSeekV2-Lite, Qwen1.5-MoE and Mixtral-8×7B.** 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.
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**
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**

prior work (Huang et al., 2024) and perform weight-only quantization with group-wise, asymmetric GPTQ (Frantar et al., 2022) (group size 128). 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 the average bit-width across all layers in MoE blocks. We report results under two bits-per-layer settings: 2.5 and 3.5.

We compare AlphaQ against two baselines under matched budget setting: 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).

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

Table 2: **Ablation study for bit-budget allocation.** We compare perplexity (PPL) on WikiText2 and accuracy (Acc.) in six zero-shot benchmark under block-wise budget and global budget.

Bit	Budget	Mixtral-8×7B		DeepSeekV2-Lite	
		PPL (↓)	Acc. (↑)	PPL (↓)	Acc. (↑)
2	Block	**6.01**	**69.36**	7.83	60.96
	Global (Ours)	6.11	66.64	**7.30**	**63.05**
2.5	Block	5.41	72.28	6.99	64.61
	Global (Ours)	**5.17**	**72.39**	**6.64**	**65.69**

Table 3: **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	7.37	**7.30**	6.81	**6.69**

consistently stronger results across models and budgets, indicating that our calibration-free bit-allocation method achieves better performance. More detailed results are provided in Appendix A.9.

Notably, on Qwen1.5-MoE, AlphaQ with an average bit-width 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.

4.3 Ablation Study

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 2. In most 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. However, an exception appears for Mixtral-8×7B under a 2-bit budget, where block-wise budgeting achieves better performance than global budgeting. By analyzing the `PL_Alpha_Hill` distribution in Figure 3 and the mixed-precision configuration obtained under global budgeting, we find that most experts in Blocks 2–5 of Mixtral-8×7B have relatively large `PL_Alpha_Hill`, suggesting that these consecutive blocks are poorly trained. As a result, these blocks are assigned the lowest bit-width, creating a major bottleneck for internal signal propagation. In contrast, block-wise budgeting preserves higher precision for these blocks, leading to better overall performance. This result demonstrates the flexibility and interpretability of our alpha-based bit-allocation method across different MoE variants.

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 3, 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.

4.4 Efficiency

We evaluate the inference performance of AlphaQ on an NVIDIA A40 GPU (48GB) using the kernels developed by PMQ (Huang et al., 2024). In Appendix A.10, we report detailed results about trade-off between inference-speedup, memory footprint and accuracy . Our AlphaQ method substantially alleviates the MoE memory bottleneck across bit budgets. In particular, for Qwen1.5-MoE at 3.5 bit budget, it preserves BF16-level accuracy while reducing weight memory to roughly one quarter.

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.

REFERENCES

Anonymous. Towards global expert-level mixed-precision quantization for mixture-of-experts LLMs. In *Submitted to The Fourteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wAc718O8UM. under review.

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, pp. 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)*, pp. 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.

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*, pp. 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*, pp. 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.

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.

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, 2024.

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.

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.

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.

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*, pp. 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.

David Pisinger and Paolo Toth. Knapsack problems. In *Handbook of Combinatorial Optimization: Volume1–3*, pp. 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/.

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*, pp. 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*, pp. 27024–27037, 2025.

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*, pp. 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. Moqa: Rethinking moe quantization with multi-stage data-model distribution awareness. *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.

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

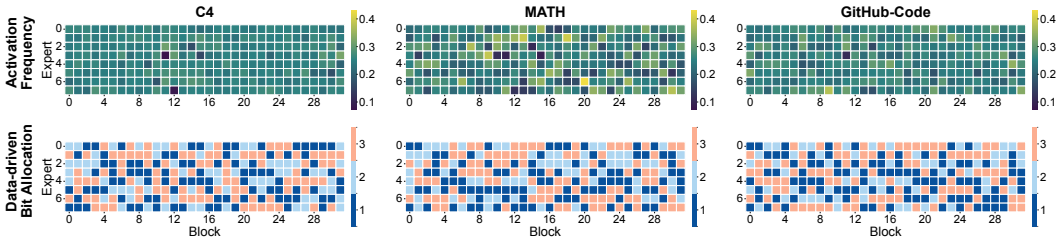

Figure 4: **Domain-dependent expert activation patterns and data-driven bit-width allocation in Mixtral-8×7B.** Activation frequencies (top) and corresponding data-driven bit-width allocations (bottom) across different domains (C4, MATH, GitHub-Code), illustrating substantial variations induced by calibration data from different domains.

A Appendix

A.1 Calibration-dependent Expert Activation and Bit-Width Allocation

To illustrate the influence of the calibration datasets for MoE bit-width allocation, we apply the data-driven method PMQ (Huang et al., 2024) to assign bits for Mixtral-8×7B, using datasets from different domains leads to substantially different activation statistics. As illustrated in Figure 4, the collected expert activation frequencies vary significantly as a function of the calibration data. As a result, different calibration-time statistics lead to various data-driven bit-width allocation.

A.2 Terminology Summary

To avoid ambiguity, we summarize the terminology used throughout this paper in Figure 5 and 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.

A.3 Derivation of the Hill Estimator

A.3.1 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.

A.3.2 From the Power-Law Density to a Pareto Form

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

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

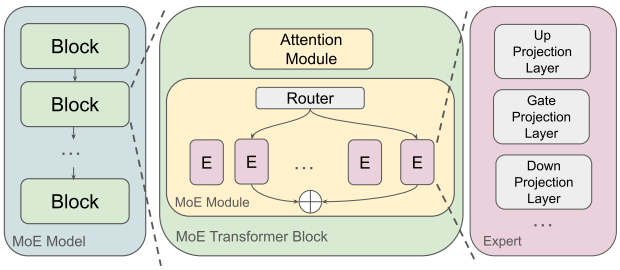

Figure 5: **Hierarchical relationship among blocks, experts, and layers in our paper.** A MoE model consists of multiple Transformer blocks; each block contains an attention module and an MoE module with multiple experts; each expert further comprises multiple layers (e.g. up, gate, and down projection).

where the normalization constant is determined by

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

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

A.3.3 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

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

A.3.4 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 equation 13 gives the Hill estimator used in Eq. 3:

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

Table 4: **Comparison between AlphaPruning (Lu et al., 2024) and AlphaQ under a 3-bit budget.** AlphaPruning directly applies a linear Alpha-based mapping without aspect-ratio–robust spectral estimation.

Model	Method	WikiText (PPL ↓)
Mixtral-8×7B	AlphaPruning	26.75
	AlphaQ	**4.37**
DeepSeekV2-Lite	AlphaPruning	23.15
	AlphaQ	**6.60**

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 $\hat{\alpha}$.

A.4 Comparison with AlphaPruning (Lu et al., 2024)

AlphaPruning uses PL_Alpha_Hill to allocate budget for sparsity and mixed-precision quantization. For quantization, they allocate a quantization bit-width for each Transformer block by using a linear mapping function $\phi : \mathbb{R}^L \to \mathbb{R}^L$ to map a sequence of PL_Alpha_Hill values $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_L)$ into the corresponding layer-wise quantization bit-widths $\phi(\boldsymbol{\alpha})$:

$$\phi(\boldsymbol{\alpha})_i = \mathrm{round}\left(\eta \left[\frac{\alpha_i - \alpha_{\min}}{\alpha_{\max} - \alpha_{\min}}(b_2 - b_1) + b_1\right]\right). \tag{16}$$

Here, $\phi(\boldsymbol{\alpha})_i$ denotes the i-th element of the resulting vector $\phi(\boldsymbol{\alpha})$, and α_i denotes the PL_Alpha_Hill value of the i-th layer. The terms $\alpha_{\min}$ and $\alpha_{\max}$ represent the minimum and maximum `PL_Alpha_Hill` values across all layers. The normalization factor η rescales the layer-wise bit-widths to satisfy a target global bit budget B. Each layer's bit-width is constrained within the interval $[\eta b_1,\ \eta b_2]$. We compute η by enforcing

$$\sum_{i=1}^{L} \phi(\boldsymbol{\alpha})_i d_i = B \cdot \sum_{i=1}^{L} d_i, \tag{17}$$

where d_i denotes the number of parameters in layer $\mathbf{W}_i$. Both sides of the equation represent the parameter-count-weighted global bit budget. The pair (b_1, b_2) are tunable hyperparameters that control the degree of non-uniformity in the layer-wise bit-width allocation.

We report the comparative results of **AlphaPruning** and **AlphaQ** in Table 4. The results suggest that AlphaPruning is less effective on MoE models, which we attribute to the following design choices:

1. Bit allocation is conducted only at the Transformer block level, without modeling finer-grained structures within MoE layers.
2. The method adopts a linear mapping function, which may limit its ability to capture non-linear quantization sensitivity.
3. Aspect-ratio differences are not accounted for, which introduces additional bias into `PL_Alpha_Hill` calculation.

A.5 More Detailed PL_Alpha_Hill Results

In Figure 7, we report the layer-wise `PL_Alpha_Hill` distribution within sampled blocks for four MoE models, including Mixtral-8×7B, Llama-MoE-3.5B (Zhu et al., 2024), DeepSeekV2-Lite, and Qwen1.5-MoE. Moreover, we report the expert-wise `PL_Alpha_Hill` values of these sampled blocks in Figure 6. For comparison, we also report the layer-wise `PL_Alpha_Hill` distribution within sampled blocks 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).

With the more detailed results, we find that:

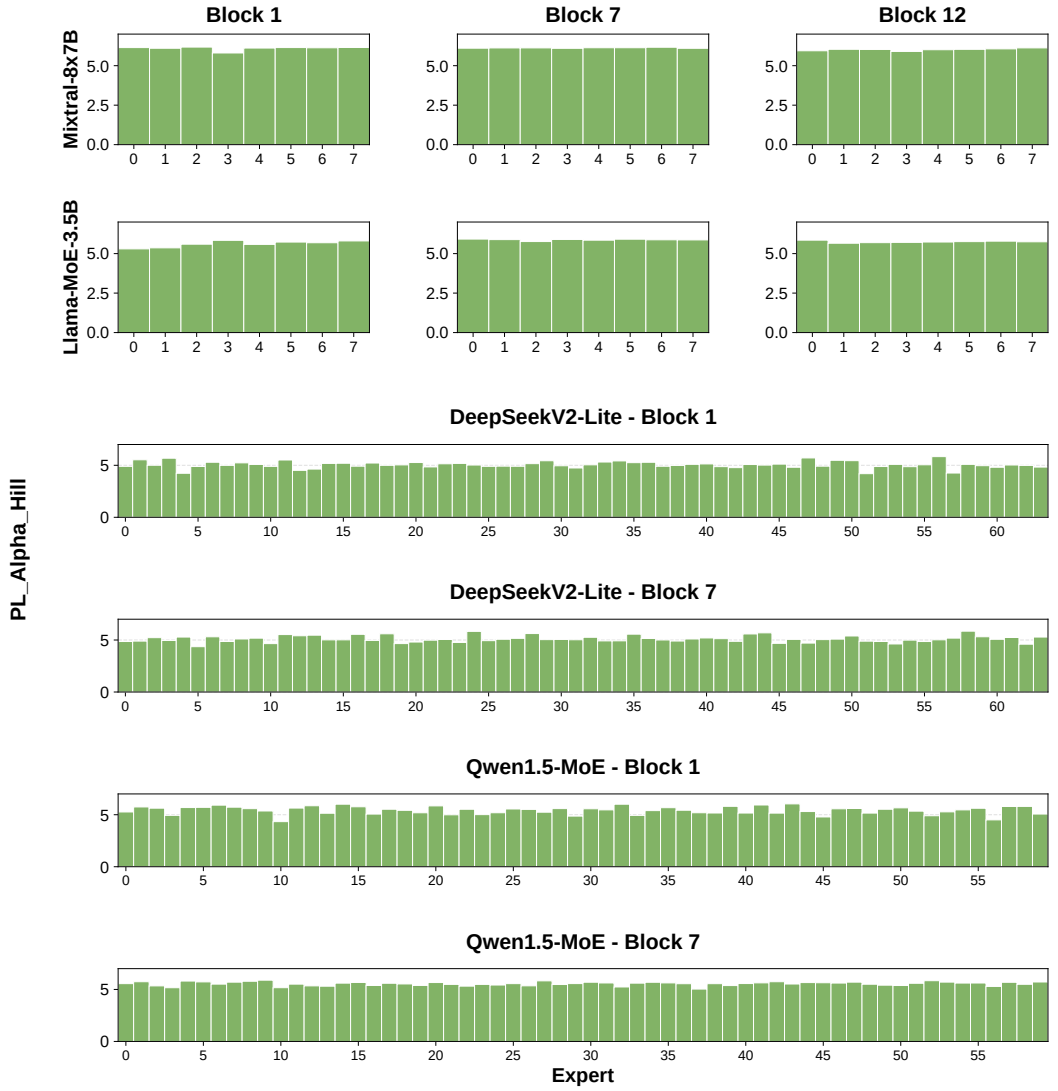

Figure 6: Expert-wise `PL_Alpha_Hill` distribution of sampled blocks in four MoE LLMs.

i) *Intra-Block Diversity.* Within the same MoE block, the up, gate, and down projection layers in the MoE structure 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) 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 layers are less fully trained and therefore have greater compression potential.

A.6 Ablation Study for Importance and Quantization Noise

We compare the experimental results of importance-only, quantization-noise-only, and our proposed AlphaQ. Specifically, in the importance-only setting, layers are sorted by importance, and we assign higher precision to the top-k most important layers. In the quantization-noise-only setting, we use uniform importance across layers and solve the constrained optimization problem for bit allocation.

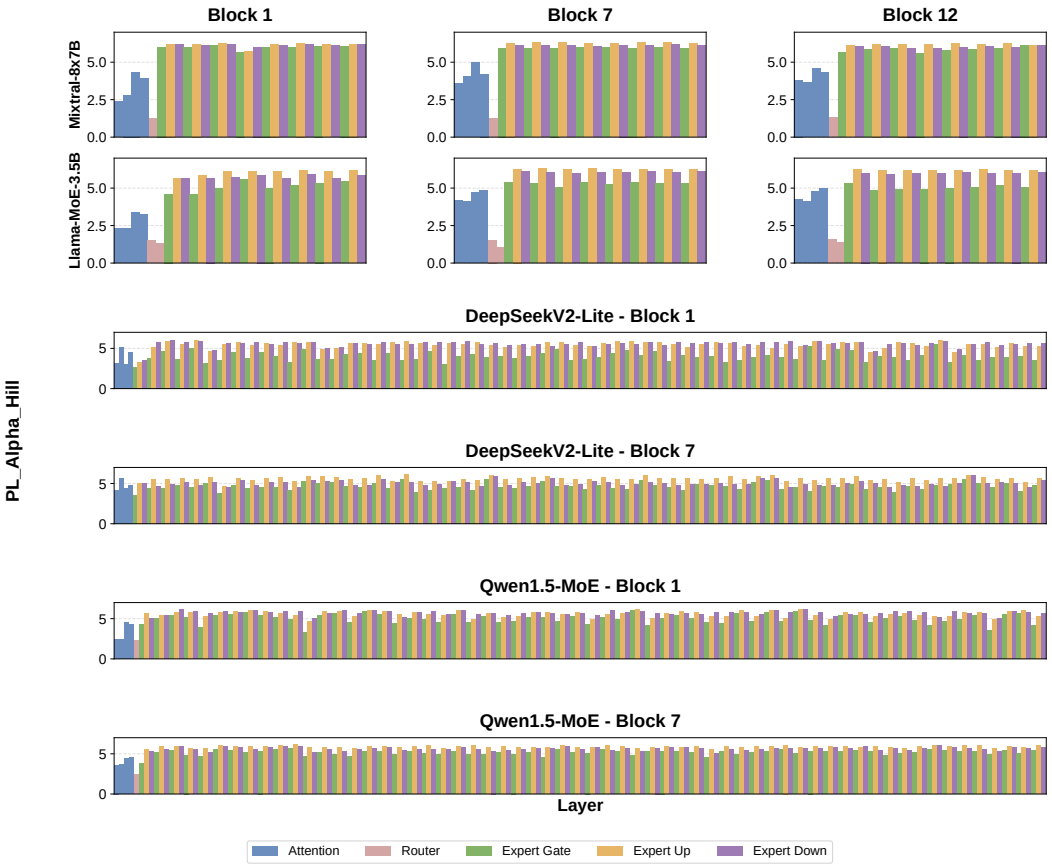

Figure 7: Layer-wise `PL_Alpha_Hill` distribution of sampled blocks in four MoE LLMs

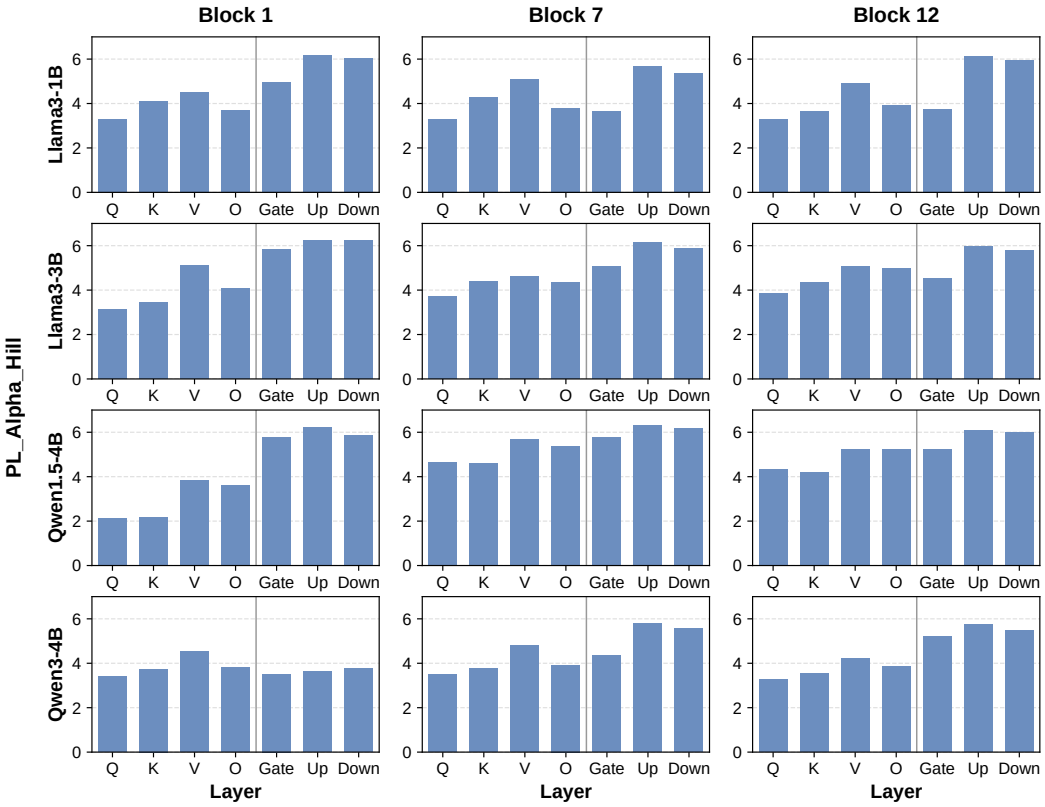

Figure 8: Layer-wise `PL_Alpha_Hill` distribution of sampled blocks in four non-MoE LLMs

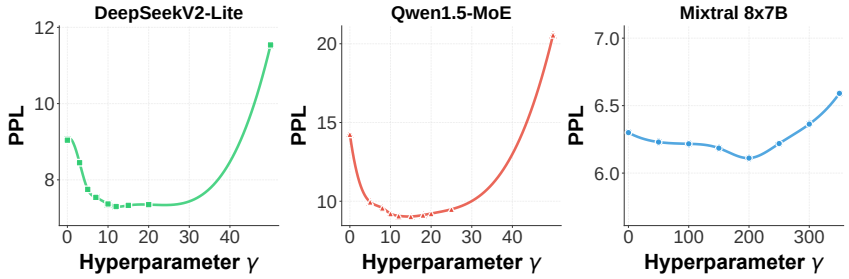

Figure 9: **Sensitivity analysis of hyperparameter** γ. Sensitivity varies across MoE models: DeepSeekV2-Lite and Qwen1.5-MoE are much more sensitive to γ than Mixtral-8×7B.

We consider two bit budgets: 2.5 and 3.5. For the 2.5-bit budget, each layer is assigned a bit-width from $\{2, 3\}$; for the 3.5-bit budget, the candidate bit-widths are $\{3, 4\}$. The results in Table 5 show that using only alpha-based importance or only quantization noise leads to larger accuracy degradation than our joint optimization, indicating that both signals are needed for effective bit allocation.

A.7 Justification of the Quantization Noise Model

Here, we provide a theoretical justification for modeling the layer-wise quantization error variance as $\eta_{l,b} \propto 2^{-2b}$. We consider a uniform quantizer applied to the weights of the l-th layer, denoted by $\mathbf{W}_l$. We assume the weight lies 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

$$\Delta_{l,b} = \frac{2R_l}{2^b}. \tag{18}$$

Under Bennett's high-resolution quantization hypothesis (Widrow & 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

$$\mathrm{Var}(\mathbf{E}_{l,b}) \approx \frac{\Delta_{l,b}^2}{12} = \frac{1}{12}\left(\frac{2R_l}{2^b}\right)^2 = \frac{R_l^2}{3} \cdot 2^{-2b}. \tag{19}$$

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,

$$\eta_{l,b} = c_l 2^{-2b}, \tag{20}$$

where c_l captures the layer-specific scale.

A.8 Ablation Study of Hyperparameter γ

We conduct an ablation study of the hyperparameter γ in Equation 4 and derive an empirical tuning strategy. We first examine sensitivity to γ across different MoE variants. As shown in

Table 5: **Ablation Study for Importance and Quantization Noise.**

Bits	Method	WikiText (PPL ↓)
2.5 bit	Importance-only	6.56
	Quantization-Noise-only	6.17
	AlphaQ	5.77
3.5 bit	Importance-only	5.06
	Quantization-Noise-only	4.31
	AlphaQ	4.24

Figure 9, fine-grained MoE models such as DeepSeekV2-Lite and Qwen1.5-MoE are much more sensitive to γ than the vanilla MoE model Mixtral-8×7B. By comparing these results with the `PL_Alpha_Hill` distributions in Figure 3, we observe a relationship between γ sensitivity and the variance of `PL_Alpha_Hill`: models with smaller variance exhibit lower sensitivity to γ, suggesting that a larger tuning interval can be used.

Based on these observations, we propose an empirical strategy for tuning γ. Specifically, we set the tuning interval to $(\mu_\alpha - 2)/\sigma_\alpha$, where μ_α and σ_α^2 denote the mean and variance of `PL_Alpha_Hill` values across all components, and 2 is a constant corresponding to good training sufficiency. We then optimize γ using binary search. Empirically, a value of γ that works well for one bit budget usually also performs well for other bit budgets on the same model.

A.9 Further Experiment Results

As shown in Table 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 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 10, Figure 11 and Figure 12, we report detailed bit allocation results from our AlphaQ method under 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**

A.10 Speedup and Memory Compression

Figure 13 illustrates the accuracy–speedup Pareto frontier on Mixtral 8×7B, comparing AlphaQ with uniform quantization and the data-driven baseline PMQ across different bit budgets. We observe that AlphaQ consistently dominates the frontier, providing a superior trade-off between model compression and inference speedup.

For memory compression, as shown in Table 9, AlphaQ effectively addresses the memory bottleneck of MoE models. For 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 only a 2.7% accuracy drop. Notably, under a 3.5-bit budget, Qwen1.5-MoE achieves accuracy on par with the BF16 model while reducing the weight memory footprint to one quarter.

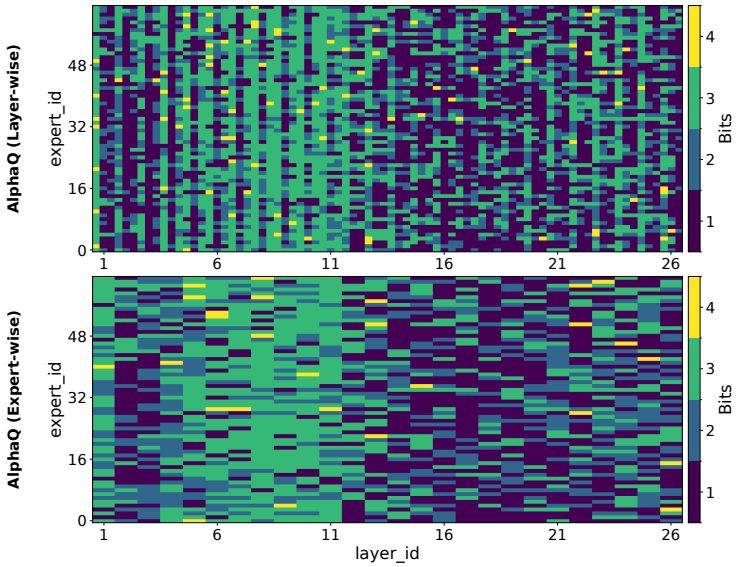

Figure 10: Bit allocation of DeepSeekV2-Lite under 2bit budget

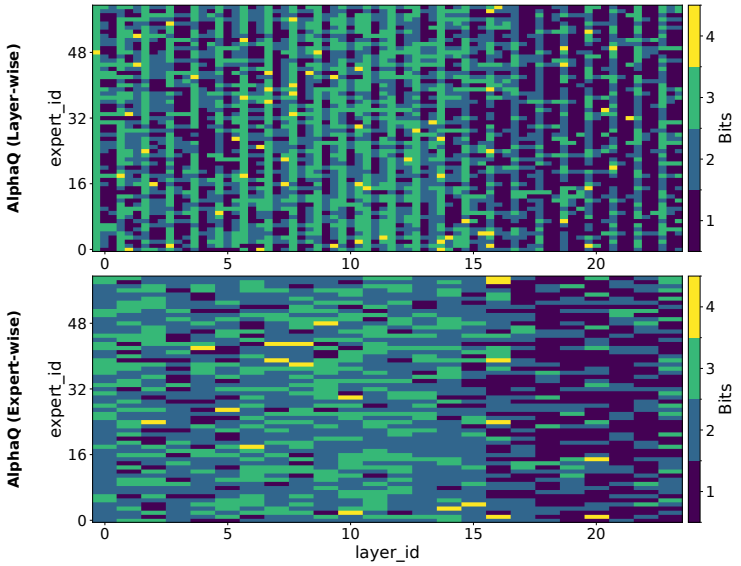

Figure 11: Bit allocation of Qwen1.5-MoE under 2bit budget

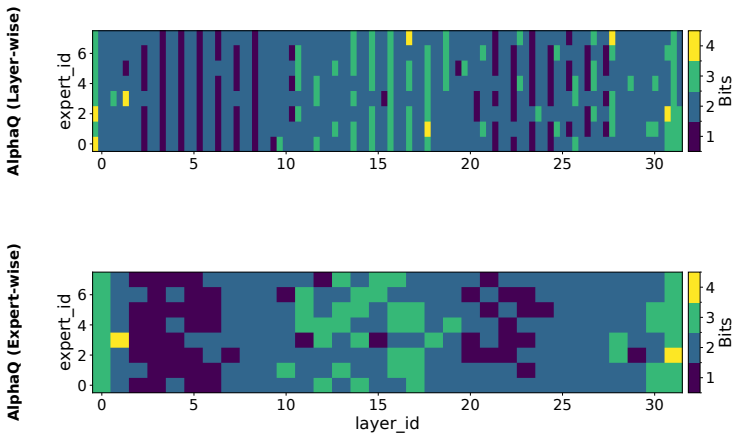

Figure 12: Bit allocation of Mixtral-8×7B under 2bit budget

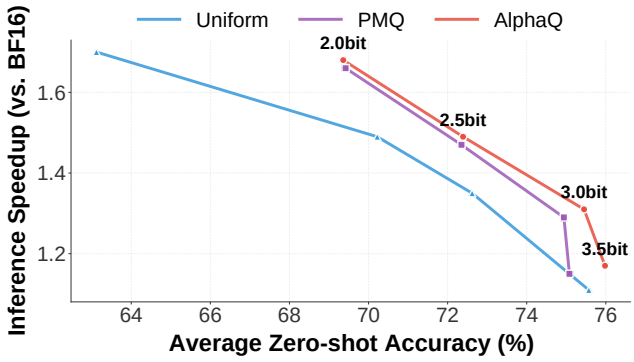

Figure 13: **Accuracy–speedup Pareto frontier on Mixtral-8×7B.** Average zero-shot accuracy versus inference speedup (relative to BF16) for varying bit budgets (2, 2.5, 3, 3.5 bits).

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**

A.11 LIMITATION

This work still has several limitations. First, our experiments are restricted to weight-only quantization and do not evaluate the applicability of AlphaQ in activation quantization settings. Second, AlphaQ relies on the degree of heavy-tailedness in the weight spectrum as a proxy for expert- and layer-level importance. While this assumption holds for the MoE language models evaluated in this work, its generality for non-language models or broader MoE architectures remains to be validated. Finally, 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 study.

Table 9: **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	Setting	Speedup ↑	Params (GB) ↓	Act. Params (GB) ↓
Mixtral-8×7B	BF16	1.00×	96.80	26.31
	2-bit	1.68×	13.41	3.75
	2.5-bit	1.47×	16.30	4.52
	3-bit	1.31×	19.12	5.27
	3.5-bit	1.17×	22.03	5.81
Qwen1.5-MoE	BF16	1.00×	28.94	5.74
	2.5-bit	1.22×	6.37	1.16
	3.5-bit	1.12×	7.60	1.45