An Analysis of Expert Pruning and Quantization of Mixture of Experts Models

Ibrahim Musaddequr Rahman
University of Michigan
iamr@umich.edu

Michael Moffatt
University of Michigan
mmoffatt@umich.edu

Tongyuan Miao
University of Michigan
tymiao@umich.edu

***Abstract*—Mixture of Experts language models are among the most popular architectures, due to their ability to encode the information of a larger model without the computational downsides. For application-specific use cases, not all weights within the model are relevant. This paper analyzes the performance implications of pruning or quantizing the least relevant experts within a model, with both internal and external measures. Our results show that post-training quantization, even at aggressive bit-widths and across a large fraction of experts, has negligible impact on benchmark performance for tasks such as GSM8K and WMT16. In contrast, expert pruning exhibits highly task-dependent behavior: while some benchmarks degrade gradually, others, particularly reasoning heavy tasks, experience catastrophic performance collapse when even a small number of experts are removed. We further find that pruning strategies informed by expert usage can outperform random or highest-usage pruning. The code for this work is available at https://github.com/mmoffatt2/weight_weight_dont_tell_me.**

I. Introduction

A. Mixture of Experts Models

The Mixture of Experts (MoE) architecture, introduced by Jacobs et al. (1991), departs from monolithic neural networks by decomposing complex problems into simpler subtasks [1]. A gating network routes inputs to specialized expert networks, enabling computational efficiency through sparse activation. This modular design reduces task interference and allows targeted capacity scaling.

The original formulation defines the output as a stochastic mixture: $y = \sum_{i=1}^{n} p_i \cdot E_i(x)$, where E_i are expert networks and p_i are softmax-derived gating probabilities. Jacobs et al. trained this using a negative log-likelihood objective that encourages expert competition, driving specialization across distinct input regions.

B. Modern Implementations in Large Language Models

MoE architectures have become essential for scaling large language models efficiently. Modern implementations replace Transformer feed-forward networks with MoE layers that activate only a top-k subset of experts per token.

Key developments include: Switch Transformers [2], which routed to a single expert per token (k=1) and achieved 7.5× pretraining speedup over dense T5-XXL; GLaM [3], which scaled to 1.2 trillion parameters with 64 experts per layer while activating only two; and GShard [4], which enabled automatic sharding across TPU pods.

Recent open-source models have further advanced the field: Mixtral 8x7B [5] uses 8 experts with top-2 routing to match dense models twice its active size; OLMoE [6] demonstrates early emergence of expert specialization through transparent training pipelines; and DeepSeek-V3 and Qwen3-MoE improve efficiency via advanced load balancing and shared expert designs.

C. Pruning of Experts

Expert pruning reduces MoE parameter footprints under deployment constraints. Recent work shows pruning must account for both redundancy structure and routing dynamics.

C-PRUNE targets two empirical properties: (i) intra-layer expert homogeneity, where experts within a layer converge to similar functions, and (ii) inter-layer similarity patterns, where deeper layers show stronger redundancy [7]. It clusters similar experts within layers, then prunes redundant clusters globally using a unified scoring mechanism. Evaluations on DeepSeek-V2-Lite and Qwen1.5-MoE-A2.7B show this approach outperforms random pruning and local grouping baselines.

Complementing this, Cross-Layer Expert Flow (CLEF) captures global expert roles in cross-layer information flow [8]. CLEF estimates directed influence

from an expert in layer l to downstream routing in layer $l+1$. From this, the Expert Specialization Index (ESI) measures functional specialization through normalized entropy of downstream influence. Analysis reveals an architecture-strategy fit: highly specialized experts require router redirection after pruning, while interchangeable experts benefit from router deletion and probability renormalization. These findings provide a foundation for combining application profiling with structured pruning and subsequent compression.

D. Quantization of Experts

MoE quantization addresses the memory bottleneck of sparse activation without requiring expensive retraining. Standard post-training quantization methods like GPTQ and SmoothQuant apply uniform bitwidth across layers, which fails to exploit MoE sparsity and routing-specific activation patterns.

Recent work explores expert-usage aware quantization. QuantMoE-Bench demonstrates that mixed-precision quantization tailored to MoE structure outperforms fixed-bit approaches [9]. Using calibration data, it shows attention layers need higher precision than expert MLPs, shared experts need higher precision than token-conditioned experts, and earlier MoE layers need higher precision than later ones. However, these findings are limited to the WikiText dataset, leaving open questions about performance on reasoning, multiple-choice, or instruction-following tasks.

MxMoE treats MoE quantization as an algorithm-system co-design problem [10]. It focuses on efficient kernels for heterogeneous-precision experts, proposing a mixed-precision Group-GEMM execution strategy to reduce kernel-launch overhead when invoking experts sequentially.

MoQE takes a different approach, treating quantization variants as experts themselves [11]. It trains a lightweight router to select among multiple quantized model variants per sample, exploiting the fact that quantization methods fail non-uniformly across data subsets. This recovers performance comparable to the best single quantized model without significant latency increases.

While mixed-precision quantization shows promise for MoE compression, no comprehensive study has evaluated these techniques jointly with pruning across diverse, application-relevant datasets. This gap motivates our work.

II. Methodology

A. Model and Benchmark Selection

The core motivation for our work was to study how MoE routing behavior and expert importance vary across application domains. We evaluated our methods on a diverse suite of tasks: WIKITEXT for language modeling, GSM8K for mathematical reasoning, DS-1000 for code generation, HUMANEVAL for functional program synthesis, SWE-BENCH for software engineering tasks, and WMT for machine translation. After collecting routing traces across these tasks, we further analyzed the tasks that exhibited the most informative and distinctive routing behaviors.

After building a list of open-weight MoE models, we decided to start by focusing in on one specific model: `deepseek-moe-base-16B`, to learn the MoE expert patterns which could generalize to other models. DeepSeek provided a well-understood sparse MoE Transformer architecture with top-k routing and multiple experts per layer.

B. Tracing

To better understand the expert activation patterns, we developed a profiling infrastructure. While some models (specifically Mixtral-8x7B) expose their routing logits through the forward pass output, to support models without exposed router outputs, we used PyTorch hooks on the gating modules to intercept routing decisions. Furthermore, since there isn't a standard terminology for HuggingFace model configs, we created our own metadata file to keep track of the number of hidden layers, routed experts per layer, and top-k experts selected per token for each model architecture. We also made sure that our profiling system could detect quantized models and load them in their mixed-precision format using AutoGPTQ.

For each dataset and model, we kept track of the expert usage across all prompts in the dataset. This serves as the primary input for our pruning and quantization algorithms. We also collected detailed per-token routing to allow for more fine-grained analysis of routing patterns and variance.

C. Quantization

Given routing traces collected during profiling, we compute an expert usage statistic $u_{i,e}$, defined as the number of tokens routed to expert e in layer i. These statistics are aggregated across all prompts for a given task.

We then assign low precision to the globally least-used experts across all layers. Specifically, we select the K experts with the smallest usage counts and quantize them to a low bit-width:

$$b_{i,e} = \begin{cases} b_{\text{low}}, & (i,e) \in \text{BottomK}(u), \\ b_{\text{full precision}}, & \text{otherwise.} \end{cases}$$

Once bit-widths are assigned at the expert level, they must be mapped to concrete weight tensors. Because MoE architectures differ in how expert parameters are structured, we construct architecture-specific mappings.

For DeepSeek MoE models, each expert consists of a three-layer MLP with gate, up, and down projections. A single bit assignment for an expert is applied uniformly to all of its projections. Non-MoE layers and shared components are excluded from quantization, leaving attention layers and routing networks in full precision.

1) GPTQ: Post-Training Quantization Library: We apply post-training quantization using GPTQ [12], a Hessian-aware quantization method designed to accurately compress large Transformer models without retraining. GPTQ formulates quantization as a layer-wise reconstruction problem and leverages second-order information derived from a small calibration set to guide quantization decisions. This allows GPTQ to account for correlations between weights and to minimize the impact of quantization error on the model's outputs, rather than quantizing each weight independently.

We integrate GPTQ into our workflow in a mixed-precision setting by applying it selectively to expert parameters identified by our usage-aware bit assignment strategy. Experts assigned to low precision are quantized using GPTQ, while all remaining parameters are kept in full precision.

D. Expert Pruning

Based on tracing outputs, the least-used k experts per layer are identified. For layer ℓ with usage counts $\mathbf{c}^{(\ell)} = [c_0^{(\ell)}, \ldots, c_{n-1}^{(\ell)}]$, the removal set is:

$$\mathcal{R}^{(\ell)} = \{\text{indices of the } k \text{ smallest values in } \mathbf{c}^{(\ell)}\}$$

The retained set is $\mathcal{K}^{(\ell)} = \{0, \ldots, n-1\} \setminus \mathcal{R}^{(\ell)}$ with $|\mathcal{K}^{(\ell)}| = n - k$.

Experts are pruned by constructing a new module list:

$$\mathbf{E}_{\text{pruned}}^{(\ell)} = [E_i^{(\ell)} : i \in \mathcal{K}^{(\ell)}]$$

which removes the least-used expert modules while preserving ordering.

The pruning proceeds layer-by-layer for memory efficiency. After loading the model configuration and validating consistent final expert counts across layers, each MoE block is detected by searching for common architectural patterns. The expert module list is replaced with $\mathbf{E}_{\text{pruned}}^{(\ell)}$ based on either global or per-layer removal sets.

Concurrently, the router's linear layer is adjusted. Given original weights $\mathbf{W}_{\text{router}}^{(\ell)} \in \mathbb{R}^{n \times d}$, the pruned weights are:

$$\mathbf{W}_{\text{router, pruned}}^{(\ell)} = [\mathbf{W}_{\text{router}}^{(\ell)}[i,:] : i \in \mathcal{K}^{(\ell)}] \in \mathbb{R}^{(n-k) \times d}$$

The bias vector $\mathbf{b}_{\text{router}}^{(\ell)} \in \mathbb{R}^n$ is similarly reduced to $\mathbb{R}^{n-k}$ by selecting entries for retained experts. This row removal preserves routing for remaining experts while eliminating computations for pruned ones.

After modification, each layer is moved to CPU and device caches are cleared. The final model is saved with an updated configuration specifying the new expert count, enabling standard loading without custom logic.

E. Benchmark Evaluation

Lm-eval-harness is a widely used framework for standardized evaluation of language models across reasoning, multiple-choice, language modeling, and translation tasks [13]. The supported tasks can be found at: https://github.com/EleutherAI/lm-evaluation-harness/blob/main/lm_eval/tasks/README.md.

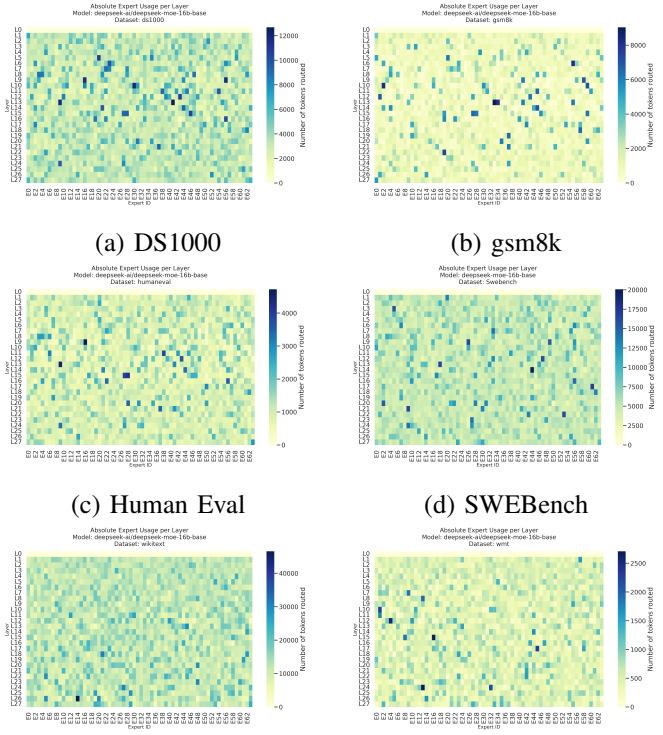
Our evaluation framework builds on the lm-eval-harness, extending it to support usage-aware, mixed-precision GPTQ-quantized MoE models. Quantized checkpoints are loaded via AutoGPTQ in mixed-precision mode. For each benchmark, we follow the default evaluation protocols defined by the harness, including task-specific metrics (e.g., accuracy for multiple-choice tasks and BLEU for translation). Each task is evaluated independently. We use the same task for tracing, quantization calibration, and evaluation to ensure alignment between observed expert usage patterns and downstream performance, thereby isolating the effects of expert pruning and quantization. Also, all evaluations are conducted in a zero-shot setting with a fixed batch size to ensure consistency across pruning and quantization.

III. Results

A. Initial Usage Profiling

To survey the usage of experts, the total usage of each expert in the original model was recorded when running all prompts for each benchmarks. It can be seen that gsm8k and WMT datasets exhibit the greatest expert sparsity, with particular experts used more than

others. This behavior manifests as ”hot spots” that can be optically seen within Figure 1. Datasets with more diverse tasks, such as SWEBench and WikiText, have expert usage more balanced.



(a) DS1000 (b) gsm8k

(c) Human Eval (d) SWEBench

(e) Wikitext (f) WMT

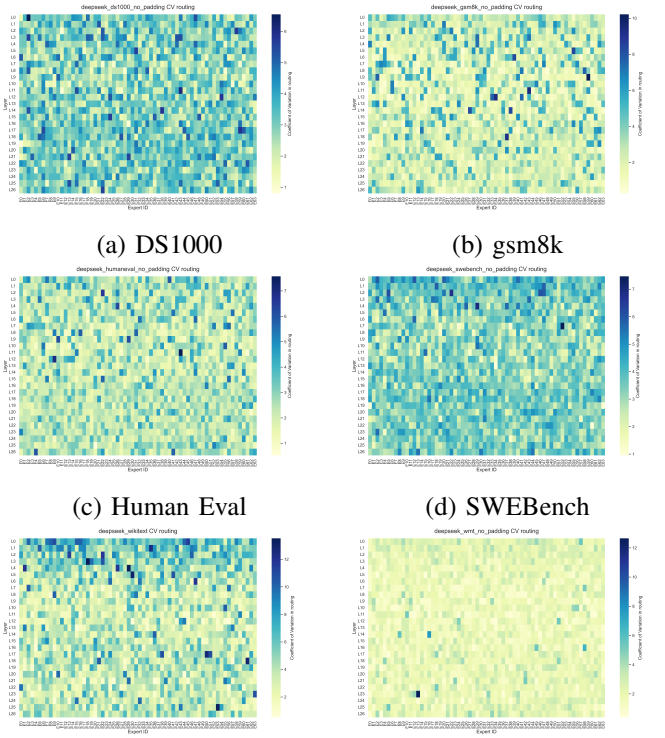
Fig. 1: Sparse Usage of Deepseek-MoE-16B

To get a better understanding of usage across prompts, the coefficient of variation of each expert's usage was also recorded in Figure 2. The variation is not strictly correlated with sparsity of usage, as can be seen with gsm8k. Variation patterns are also not uniform across all patterns, wikitext demonstrates a higher variance in earlier expert layers within the model.

B. Quantization

For the benchmarks chosen for this project, quantization had negligible effect on the benchmark scores. Various quantization schemes were experimented with, including per expert and global quantization, but all changes were within variance.

As shown with GSM8k 4bit and 8bit quantization, we observe that varying the number of quantized experts has minimal impact on task performance. GSM8K accuracy remains largely stable across a wide range of quantization levels, with differences on the order of a few percentage points and no consistent monotonic trend.



(a) DS1000 (b) gsm8k

(c) Human Eval (d) SWEBench

(e) Wikitext (f) WMT

Fig. 2: Per-Prompt Expert Variance, Deepseek-MoE-16B

# Quantized Experts	GSM8K Accuracy
70	0.15
100	0.17
1728	0.17

TABLE I: GSM8K accuracy for 4-bit quantized models with varying numbers of quantized experts.

Notably, even aggressive quantization, up to quantizing all experts, does not lead to catastrophic degradation in performance.

Similar results were found with WMT.

As a result, we instead focused our efforts on analysis of pruning strategies, where performance differences are more pronounced and interpretable.

C. Pruning

To the effects of pruning the lowest-k experts, comparisons against pruning the top-k and random-k experts are demonstrated. It can be seen that pruning top experts changes the activation pattern between different pruning criteria in Figure 5, while pruning of lowest experts keeps activations the same until most experts are removed as shown in Figure 6.

The effects of pruning do not effect the benchmark scores of each task equally. For some benchmarks, such

# Quantized Experts	GSM8K Accuracy
0	0.19
70	0.17
100	0.17
864	0.17
1728	0.16

TABLE II: GSM8K accuracy for 8-bit quantized models with varying numbers of quantized experts.

# Quantized Experts	WMT Score
0	42.55
50	42.79
70	43.17
100	43.66

TABLE III: WMT scores for 8-bit quantized models with varying numbers of quantized experts.

as HellaSwag, pruning results in a steady degradation of performance based on the number of experts. For others, such as gsm8k, even a single expert pruned causes performance to collapse. We compared removing lowest usage experts, random experts, and highest usage experts to confirm that expert tracing provided useful information for which experts were most useful for a given task. Figure 3 compares the HellaSwag performance across different levels of expert pruning. The largest gains in performance over random and highest usage removal seem to appear when over half the experts were removed.

For WMT16, similar results were found in 4. However, removing the lowest usage experts had a noticeably better performance than the other methods with a lower number of experts removed. Also, for some models, removing random experts did have lower performance than the top used experts, but the score differences were within the Standard Error.

For GSM8K, pruning had a significant impact on performance, bringing performance down to zero after the pruning of a single expert.

IV. Future Work

A. Investigation of Rapid Pruning Degradation

Certain difficult benchmarks, such as GSM8K, immediately went down to zero performance after the pruning of a single expert. Their activations patterns appear to follow similar trends as other pruning, yet the benchmark performance does not. More subjective analysis of output needs to be done in order to find the root cause of such performance losses, and what makes these datasets different than others.

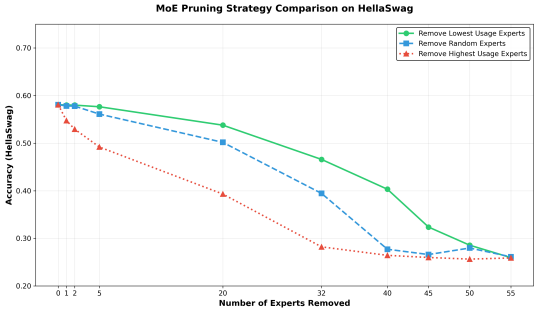


Fig. 3: HellaSwag accuracy under different levels of expert pruning.

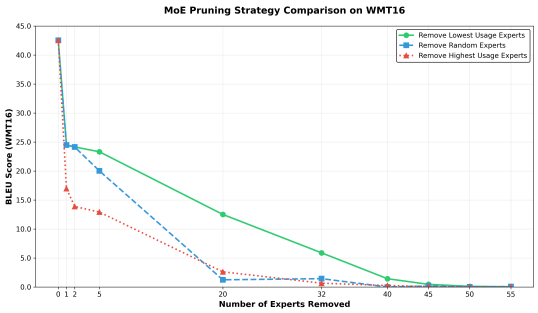
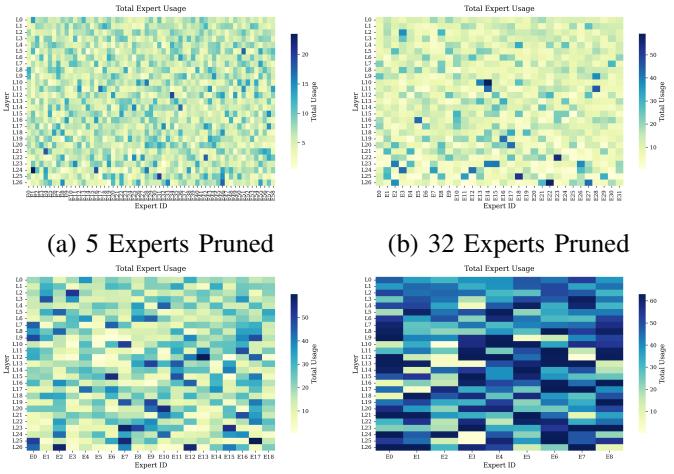


Fig. 4: WMT16 BLEU score under different levels of expert pruning.



(a) 5 Experts Pruned

(b) 32 Experts Pruned

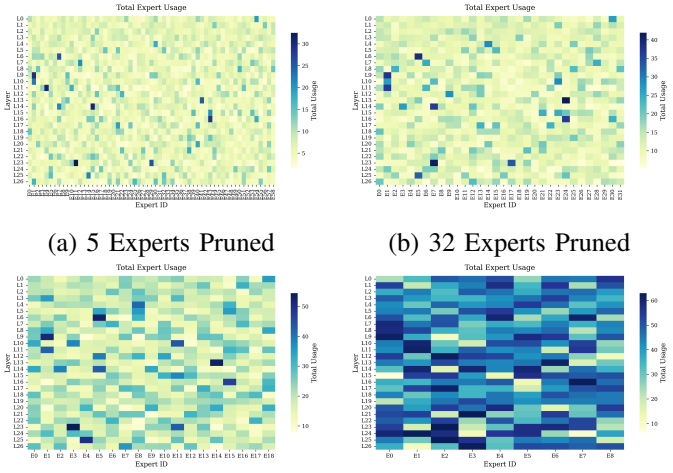
(c) 45 Experts Pruned

(d) 55 Experts Pruned

Fig. 5: WMT Top Expert Pruning

B. Usage-based Clustering of Inputs

Because expert usage is not uniform within across all prompts within a dataset, a subset of prompts could be selected that have more common expert usage patterns. Due to increased sparsity, these subsets may exhibit different behaviors than the aggregate benchmark to the removal of experts. Furthermore, there may be a ways



(a) 5 Experts Pruned (b) 32 Experts Pruned

(c) 45 Experts Pruned (d) 55 Experts Pruned

Fig. 6: WMT Least Expert Pruning

to predict or cluster these subsets ahead of time, and be able to reconfigure the model accordingly.

C. Verification of Trends Across Models

The trends observed within this paper have been observed with the Deepseek-MoE-16B model and common academic benchmarks. This methodology could be expanded to both modern models such as GPT-OSS-20B, but also to less academic benchmarks, such as leaked system prompts of popular AI applications. In particular, understanding which model-benchmark pairs are resilient to expert pruning, and which ones immediately collapse, would shed light on more fundamental properties of MoE models.

References

[1] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[2] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022.

[3] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, *et al.*, "GLaM: Efficient scaling of language models with mixture-of-experts," in *Proceedings of the 39th International Conference on Machine Learning*, pp. 5547–5569, PMLR, 2022.

[4] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "Gshard: Scaling giant models with conditional computation and automatic sharding," *arXiv preprint arXiv:2006.16668*, 2021.

[5] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[6] N. Muennighoff, A. Rush, N. Tazi, A. Chowdhery, T. Siro, D. Valter, F. Liu, F. Liu, B. Hui, O. Khattab, *et al.*, "OLMoE: Open mixture-of-experts language models," *arXiv preprint arXiv:2409.02060*, 2024.

[7] H. Guo, J. Yao, B. Wang, J. Du, S. Cao, D. Di, S. Zhang, and Z. Li, "Cluster-driven expert pruning for mixture-of-experts large language models," *arXiv preprint arXiv:2504.07807*, 2025.

[8] A. Authors, "Quantifying expert specialization for effective pruning in mixture-of-experts models." OpenReview submission, 2026. Under review at ICLR 2026.

[9] P. Li, X. Jin, Z. Tan, Y. Cheng, and T. Chen, "Quantmoe-bench: Examining post-training quantization for mixture-of-experts," 2025.

[10] H. Duanmu, X. Li, Z. Yuan, S. Zheng, J. Duan, X. Zhang, and D. Lin, "Mxmoe: Mixed-precision quantization for moe with accuracy and performance co-design," 2025.

[11] J. Zhang, Y. Zhang, B. Zhang, Z. Liu, and D. Cheng, "Moqe: Improve quantization model performance via mixture of quantization experts," 2025.

[12] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," 2023.

[13] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, and N. Thite, "The language model evaluation harness," *arXiv preprint arXiv:2109.00298*, 2021.