# Navigating the Design Space of MoE LLM Inference Optimization

Anonymous ACL submission

## Abstract

Mixture of Experts architectures have emerged as a powerful design for large language models, offering state-of-the-art performance through computational sparsity. Despite improved efficiency, their high memory demands hinder deployment on commonly available single-GPU systems. Existing approaches to mitigate this issue, including pruning, distillation, and quantization, often sacrifice model quality or increase inference latency. Recent MoE-specific strategies introduce dynamic expert offloading to DRAM, significantly reducing memory usage without degrading performance.

We evaluate and compare leading MoE optimization techniques, analyzing their memory, latency, and quality trade-offs. Building on these insights, we propose an automated MoE serving system that adaptively selects optimal configurations to meet diverse deployment constraints. This enables efficient, high-quality LLM inference on limited hardware resources.

## 1 Introduction

In recent years, we have observed the success of the Mix of Experts (MoE) design in state-of-the-art large language models (LLMs), such as DeepSeek-v3 (DeepSeek-AI et al., 2025), which demonstrate exceptional performance. The sparsity of MoE models enhances computational efficiency, although the memory requirements remain substantial. For instance, the medium-sized Mixtral 7x8B (Jiang et al., 2024) requires approximately 112GB of memory, exceeding the capacities of both the RTX 4090 (24GB) and the A6000 (80GB). To alleviate the substantial overhead resulting from the extensive memory requirements of MoE LLMs and to enable efficient deployment of these models on a single GPU, it is crucial to explore effective methods for reducing such costs.

There are various optimization methods that can reduce the memory consumption of MoE LLMs. Traditionally, techniques such as pruning (Ma et al., 2023), model distillation (TheBloke, 2023), and quantization (Dettmers et al., 2022; Wu et al., 2023) have been employed to remove redundant weights or decrease weight precision, thereby lowering memory usage. However, these approaches may result in increased inference time and decreased model quality (Liakopoulos et al., 2025).

On the other hand, several state-of-the-art methods specifically designed for MoE architectures have focused on offloading unused experts to DRAM. These methods can reduce the additional memory consumption caused by unused experts by dynamically swapping them based on demand, without negatively impacting the model's performance. We plan to evaluate and compare these designs to understand the advantages of each, including their offloading strategies and performance, and to explore possible trade-offs among these designs.

By identifying the trade-offs among latency, memory consumption, and output quality, we propose an automated system for MoE LLM serving that integrates comprehensive optimization techniques. This system automatically determines the optimal model-serving strategy based on selected parameters, effectively satisfying specific memory, latency, or output quality requirements.

## 2 Background

MoE models are well-known for utilizing different experts to specialize in various tasks, enabling a more efficient and scalable approach to complex problems. Recently, combining MoE and LLM has become popular, evidenced by the emergence of high-quality MoE-LLM models. DeepSeek-v3 (DeepSeek-AI et al., 2025) delivers outstanding performance as MoE architectures optimize data processing and decision-making in high-dimensional search spaces. However, only a few experts are invoked in each layer, while many experts occupy memory, leading to significant space wastage. For example, Mixtral 7x8B (Jiang et al., 2024) only uses 2 out of 8 experts, meaning about 75% of the memory is wasted when serving the entire model.

Current works are focusing on MoE LLM inference serving with expert offloading designs. These designs aim to load parts of the experts on the GPU while leaving others on the CPU to reduce the memory requirements of LLM serving. Unused experts can be swapped to DRAM to conserve memory and alleviate the memory bottleneck in MoE LLMs. These include (Eliseev and Mazur, 2023), which uses the next layer gating function to prefetch layers; MoE-infinity (Xue et al., 2024), which statistically prefetches experts across layers; and Fiddler (Kamahori et al., 2025), which utilizes the CPU for computation. AdapMoE (Zhong et al., 2025) skips unimportant experts, utilizes an expert cache, and employs a learnable prefetcher to reduce computational cost and expert miss-hit overhead. They also incorporate, or leave space for incorporating, traditional LLM memory optimization techniques such as int8 (Dettmers et al., 2022), int4 (Wu et al., 2023) quantization, and model distillation (TheBloke, 2023), creating an even larger exploration space.

# 3 Methodology

## 3.1 Techniques Under Evaluation

We focus on evaluating the **Mixtral-8x7B** (Jiang et al., 2024) model, a MoE LLM distinguished by its robust capabilities across a broad range of applications, including real-time language translation, advanced image recognition, and predictive analytics. These features make it well-suited for complex tasks across diverse industries. However, despite its versatility, the model's substantial size presents challenges, particularly its significant memory requirements, which often exceed the capacity of conventional hardware.

To address these limitations, it is critical to explore multiple optimization strategies. One approach is model distillation: by scaling down the Mixtral-8x7B model to a smaller Mixtral-7x4 variant, the memory footprint becomes more manageable. Another technique involves reducing numerical precision, such as converting model weights to INT8 format, thereby significantly decreasing memory consumption.

Quantization is a particularly effective method for managing model size and resource demands. However, it can affect output quality. For instance, quantization of a float16 tensor $\mathbf{X}_{f16}$ to int8 can be expressed as follows (Dettmers et al., 2022).

Beyond simple quantization, modular expert models—such as those available through GitHub projects like **Mixtral Offloading** (Eliseev and Mazur, 2023), **Fiddler** (Kamahori et al., 2025), and **MoE-Infinity** (Xue et al., 2024)—leverage Mixture-of-Experts techniques to balance performance and resource efficiency. We explore multiple expert offloading methods, including **Mixtral Offloading**, which combines quantization and the prefetching of a fixed number of experts. In particular, Mixtral offloading uses the Highly-Quantized Quantization method (Badri and Shaji, 2023).

**MoE-Infinity** further extends expert offloading by introducing statistical-based expert prefetching and caching techniques, offering an advanced optimization framework. Meanwhile, **Fiddler** adopts an alternative strategy: in addition to expert offloading, it dynamically estimates and compares the time to load an offloaded expert onto the GPU versus the computation time on the CPU, enabling more informed scheduling and improved performance.

## 3.2 End-to-end MoE LLM Serving Analysis

The end-to-end performance of LLM serving is critical; therefore, we systematically evaluate the inference performance and output quality of various MoE LLMs using a range of techniques, including quantization, distillation, general offloading, and expert offloading. To ensure comparability, we utilize publicly available methods described in related research papers and design a series of controlled experiments to evaluate these techniques and their respective implementations. In addition to directly applying these methods as-is, we also identify tunable parameters that can affect memory usage and latency for each technique, such as the number of offloaded experts in Mixtral Offloading and the total memory reserved for the model and expert cache in MoE-Infinity. We record key metrics such as GPU memory consumption, inference latency, and output quality.

**Inference latency.** For modern decoder-only LLMs (including all MoE models evaluated), inference latency is largely determined by the use of KV-cache, which avoids recomputation of previous tokens and ensures consistent latency for each newly generated token. Accordingly, LLM inference latency can be formally decomposed into the time to first token (TTFT), the time per output token (TPOT), and the total output length, as follows:

$$L = TTFT + output\_length \times TPOT \quad (1)$$

**Memory consumption.** The memory consumption of models incorporating offloading may include GPU memory usage, DRAM usage, and disk storage used for offloaded experts (Xue et al., 2025) or additional weights (Rajbhandari et al., 2020). All evaluations are conducted in a single-GPU environment. Given our focus on the high cost of GPU memory, we report the peak GPU memory usage observed during inference, which is measured using the `nvidia-smi` command.
**Output quality.** To assess output quality, we evaluate model perplexity. Specifically, we feed each model the first 262,144 tokens (split into sequences of length 1,024) from the WikiText-2 dataset and compute perplexity. Techniques such as quantization, distillation, and selective expert activation can affect the resulting output quality.

Using these three-dimensional metrics, we can illustrate the complex trade-off space introduced by various LLM MoE serving techniques.

## 4 Results

| GPU Model | Memory | RAM | Disk |
|---|---|---|---|
| RTX 4060 Mobile | 8GB | 32GB | SSD |
| RTX 4070 Mobile | 8GB | 32GB | SSD |
| RTX 4090 | 24GB | 64GB | SSD |
| RTX A6000 | 48GB | 1024GB | SSD |
| RTX A6000*3 | 144GB | 1024GB | SSD |

Table 1: GPU specifications and machine info.

### 4.1 Testbed

In this work, we implement and evaluate several techniques on Mixtral-8x7B, including quantization (Dettmers et al., 2022), model pruning (TheBloke, 2023), and Mixtral offloading on the machines listed in Table 1, using a variety of input prompts. We assess metrics such as memory consumption and execution time. Besides the specified mentions, the evaluation is conducted on a single RTX A6000.

### 4.2 MoE performance on various devices.

From Figure 1, we observe the inference latency of the MoE model with various input sizes across

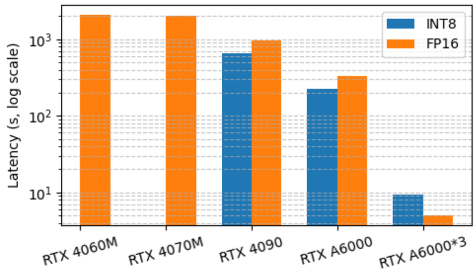


Figure 1: Latency by GPUs and Quantizations for generating 50 output tokens using Mixtral 7x8B.

different GPUs and quantization formats. Given that the Mixtral 7x8 model requires approximately 112GB of memory when loaded with Float16 (FP16), all GPUs—except for the configuration using 3×RTX A6000 with pipeline parallelism—are unable to fit the entire model in memory. As a result, these systems must offload some model weights to DRAM, or even to disk if DRAM capacity is insufficient. For example, our RTX 4060 machine has only 40GB of memory, making disk offloading unavoidable.

In our setup, we use PyTorch's default offloading policy, which is not optimized for MoE models. This leads to significant overhead from offloading, which diminishes the performance gap between different GPUs. For instance, the latency on a single RTX A6000 is approximately 44× higher than with 3×RTX A6000, and the RTX 4060 shows nearly the same latency as the RTX 4070. Interestingly, although INT8 quantization is generally slower for pure GPU inference, we observe up to a 1.48× latency improvement when offloading parameters to memory. This is because INT8 reduces the volume of data that needs to be swapped in and out of memory.

### 4.3 MoE performance on various techniques.

Table 2 summarizes the inference performance of Mixtral 8x7B model variants on an A6000 48GB GPU using quantization, distillation, and expert offloading techniques. Quantization to INT8 significantly reduces time-to-first-token (TTFT) but slightly increases per-token inference time (TPOT) and memory usage, maintaining high output quality. Distillation (4x7B) drastically reduces latency and memory usage but at the cost of significantly higher perplexity, indicating lower output quality. Mixtral-Offloading techniques effectively reduce both latency and GPU memory usage, demonstrating considerable memory efficiency with only mod-

Table 2: Inference Performance Comparison of Mixtral 8x7B Variants on the A6000 48GB GPU

| Method | TTFT (s) | TPOT (s) | Perplexity | GPU Memory (MiB) |
|---|---|---|---|---|
| Baseline | 4.54 | 3.41 | **4.06** | 44556 |
| Quantization INT8 | **1.13** | 4.01 | 4.07 | 45840 |
| Distillation 4x7B | 0.39 | 0.34 | 17.61 | 44374 |
| Mixtral-Offloading (expert = 2) | 0.65 | **0.26** | 4.82 | 15912 |
| Mixtral-Offloading (expert = 4) | 1.04 | 0.36 | 4.82 | 11964 |
| Mixtral-Offloading (expert = 6) | 1.39 | 0.48 | 4.82 | 8068 |
| Fiddler | 41.39 | 3.49 | - | **5198** |
| MoE-Infinity (mem = 0.95) | 8.14 | 0.90 | **4.06** | 47456 |
| MoE-Infinity (mem = 0.75) | 8.87 | 0.99 | **4.06** | 37888 |
| MoE-Infinity (mem = 0.50) | 8.88 | 1.20 | **4.06** | 25614 |
| MoE-Infinity (mem = 0.25) | 8.29 | 1.32 | **4.06** | 13704 |

erate degradation in perplexity. MoE-Infinity configurations show optimal perplexity, closely matching baseline output quality while allowing scalable memory utilization. These results highlight the trade-offs between latency, memory consumption, and output quality inherent in each optimization approach.

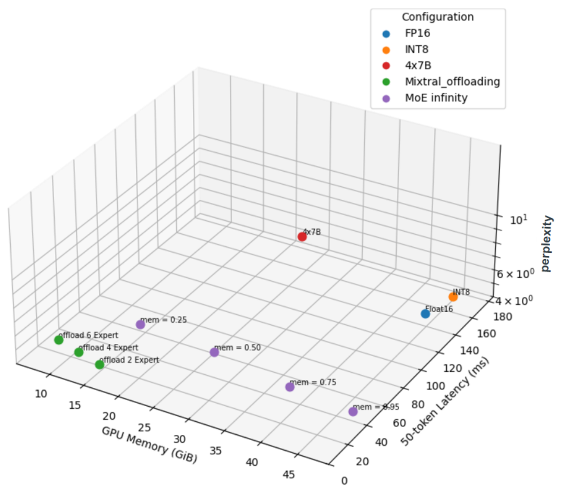


Figure 2: 3D trade-off between memory, latency, and perplexity.

Figure 2 visualizes the complex trade-off space between memory usage, latency, and perplexity. Each optimization technique has distinct advantages: MoE-Infinity excels at optimizing perplexity, while Mixtral-Offloading provides significant memory savings through quantization and offloading strategies. Furthermore, each technique contains hyperparameters that offer internal trade-offs. This complexity highlights that MoE LLM serving optimization entails navigating a multidimensional and intricate search space.

## 5 Automated MoE-LLM serving system

To navigate the complex trade-offs between latency, memory usage, and output quality, we propose an automated selection system that formulates MoE LLM serving as a multi-objective optimization problem. The system systematically profiles available serving techniques—including quantization, distillation, and expert offloading—under varying hyperparameter configurations (e.g., number of active experts, memory budgets, cache sizes). It records key performance metrics across these three dimensions to construct a comprehensive configuration-performance landscape.

Given user-defined deployment objectives (e.g., minimize latency under 16GB memory, or maximize quality within 2s/token), the system searches this landscape to select the optimal configuration. This enables automatic adaptation to diverse hardware constraints and application demands, reducing the need for manual tuning and trial-and-error experimentation.

## 6 Conclusion

In this work, we evaluate key MoE-LLM serving techniques—quantization, distillation, and expert offloading—highlighting their trade-offs across latency, memory usage, and output quality. To streamline deployment under diverse hardware and performance constraints, we propose an automated system that profiles these methods and selects optimal configurations based on user-defined objectives. Our findings enable efficient, high-quality inference on resource-limited devices and offer practical guidance for MoE-LLM deployment in real-world scenarios.

## Limitations

While our study provides a comprehensive evaluation of MoE-LLM serving techniques, several limitations remain. First, our experiments are conducted primarily on the Mixtral-8x7B model; although the findings may generalize, we do not validate them across other MoE architectures. More advanced MoE-LLMs and multimodal MoE models may exhibit different characteristics, which remain to be explored. Second, our use of perplexity as a measure of output quality is relatively limited; more comprehensive benchmarks should be employed to better evaluate the performance of MoE-LLMs. Finally, the proposed automated system may face challenges such as high profiling costs, dependency constraints, and potential errors, which warrant further investigation and validation in real-world deployment scenarios.

## References


H. Badri and A. Shaji. 2023. Half-quadratic quantization of large machine learning models. https://mobiusml.github.io/hqq_blog/. Accessed: 2025-04-07.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Artyom Eliseev and Denis Mazur. 2023. Fast inference of mixture-of-experts language models with offloading. *Preprint*, arXiv:2312.17238.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of Experts. *arXiv preprint arXiv:2401.04088*.

Keisuke Kamahori, Tian Tang, Yile Gu, Kan Zhu, and Baris Kasikci. 2025. Fiddler: Cpu-gpu orchestration for fast inference of mixture-of-experts models. *Preprint*, arXiv:2402.07033.

Dimitrios Liakopoulos, Tianrui Hu, Prasoon Sinha, and Neeraja J Yadwadkar. 2025. iserve: An intent-based serving system for llms. *arXiv preprint arXiv:2501.13111*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. *Preprint*, arXiv:1910.02054.

TheBloke. 2023. Mixtral-fusion-4x7b-instruct-v0.1-gguf. https://huggingface.co/TheBloke/Mixtral-Fusion-4x7B-Instruct-v0.1-GGUF. Accessed: 2025-03-30.

Xiaoxia Wu, Cheng Li, Reza Yazdani Aminabadi, Zhewei Yao, and Yuxiong He. 2023. Understanding INT4 quantization for language models: latency speedup, composability, and failure cases. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. 2024. Moe-infinity: Offloading-efficient moe model serving. *Preprint*, arXiv:2401.14361.

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. 2025. Moe-infinity: Efficient moe inference on personal machines with sparsity-aware expert cache. *Preprint*, arXiv:2401.14361.

Shuzhang Zhong, Ling Liang, Yuan Wang, Runsheng Wang, Ru Huang, and Meng Li. 2025. Adapmoe: Adaptive sensitivity-based expert gating and management for efficient moe inference. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '24, New York, NY, USA. Association for Computing Machinery.