DELTAMOE: MEMORY-EFFICIENT INFERENCE FOR MERGED MIXTURE OF EXPERTS WITH DELTA COMPRESSION

Boyko Borisov
ETH Zurich
bborisov@student.ethz.ch

Xiaozhe Yao
ETH Zurich
xiayao@ethz.ch

Nezihe Merve Gürel
TU Delft
n.m.gurel@tudelft.nl

Ana Klimovic
ETH Zurich
akimovic@ethz.ch

ABSTRACT

Sparse Mixture of Experts (SMoEs) have emerged as an efficient architecture for large language models. While recent community efforts have focused on merging multiple models to create SMoEs, deploying these merged models remains challenging due to their substantial memory requirements. In this paper, we present DeltaMoE, a training-free delta compression pipeline that enables efficient deployment of SMoE models through structured sparsity and quantization. Our evaluation shows that DeltaMoE achieves up to a $2.34\times$ compression ratio and $2.57\times$ throughput improvement. DeltaMoE is also scalable with the number of experts, making it particularly suitable for large SMoE models.

1 INTRODUCTION

Recent advances in large language models (LLMs) have led to remarkable performance in various tasks. The performance of LLMs is largely correlated to their scale (i.e., number of parameters). However, training and serving these models in production is challenging due to the large memory and computational cost. Sparse Mixture of Experts (SMoEs), emerged as an efficient architecture for LLMs, addresses some of these challenges by activating only a subset of experts, improving training efficiency and inference speed (Fedus et al., 2022). This architecture has been adopted in several state-of-the-art models, such as Mixtral (Jiang et al., 2024), DeepSeek (DeepSeek-AI, 2024), and Switch Transformer (Fedus et al., 2022). However, training SMoEs still remains expensive Fedus et al. (2022) and serving them requires large memory capacity, as all experts need to be loaded into the GPU memory for fast inference.

To address the cost of "creating" SMoEs, recent efforts have been made to build SMoEs by fine-tuning dense models separately and merging them into an SMoE (Sukhbaatar et al., 2024; Goddard et al., 2025). These merged models often share the same model architecture and initialization. By analyzing the weight distribution in these particular models, we observe that **the difference between experts is often small.** This observation suggests that storing full expert weights independently may be inefficient. Instead, recent work on model delta compression (Yao et al., 2024; Liu et al., 2024b; Isik et al., 2023), has shown that small differences between weights often lead to an opportunity for more efficient model serving via delta compression. Based on this, we propose DeltaMoE, a **training-free** model compression and serving method that leverages quantization and structured sparsity for SMoEs, which improves serving efficiency without sacrificing model quality.

We show that our approach can achieve performance on par with the original SMoE model while reducing the memory footprint by up to $2.34\times$ and throughput by $2.57\times$ compared to serving SMoE models in FP16 precision. The improvement becomes more significant as the number of experts increases, making our approach particularly suitable for large SMoE models.

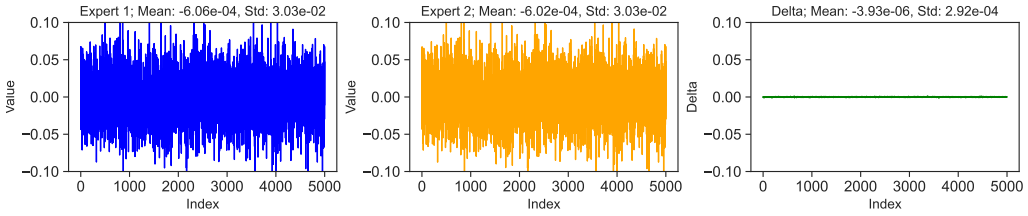

Figure 1: Expert weight values and delta of `phixtral-2x2_8`, an open-source merged SMoE (Labonne, 2024)

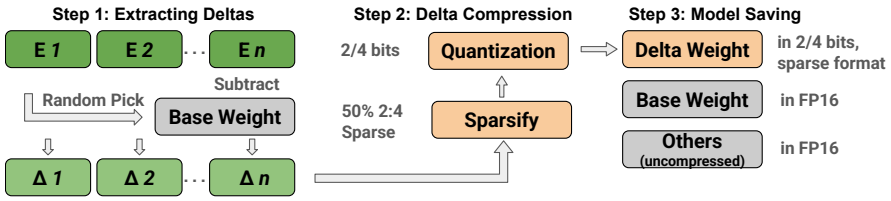

Figure 2: `DeltaMoE` compression algorithm

Related Work LLM compression has emerged as a crucial research direction for deploying these models. Most related to our work is DeltaZip Yao et al. (2024), which compresses multiple fine-tuned model variants. For SMoEs, there has been work on compressing the model as a whole, such as QMoE Frantar & Alistarh (2023) and Li et al. (2024). In contrast, we focus on a training-free compression method for merged MoEs (*MMoEs*) with both quantization and structured pruning.

2 Approach

2.1 Delta Compression for Mixture of Experts

Our compression approach is based on the observation that the weight differences, the *deltas*, between expert weights within the same SMoE block are often small for MMoEs, which allows for aggressive quantization and pruning with minimal impact on performance, due to the more compact quantization grid. We apply delta compression by defining a "base weight" representation W_b^l for each SMoE block l in the network and expressing the weights of each expert e in it as $W_b^l + \Delta_e^l$. The small magnitude of Δ_e allows for aggressive compression. Our compression pipeline includes:

1) Extracting Deltas. We randomly select one expert per SMoE block as the base weight, producing a base weight for each SMoE block. As a next step, we compute the difference between each expert and its correspondent base weight to obtain the delta. The base weights are then stored in FP16, while deltas undego our delta compression pipeline.

2) Expert Delta Compression. We apply a series of compression techniques optimized for GPU hardware features. In line with existing work in delta compression (Yao et al., 2024; Isik et al., 2023; Liu et al., 2024a), we employ both structured 2:4 pruning and quantization. Structured 2:4 pruning is setting at least two elements within each four contiguous elements in the delta matrix to 0, which allows us to take advantage of sparse tensor cores (Bai & Li, 2023) for efficient sparse matrix multiplication while quantization reduces the model's memory footprint. This way, we perform inference at a lower memory budget and reduce the memory bandwidth bottleneck.

3) Model Saving. Compressed delta weights are stored in a sparse, packed format. Specifically, for a weight matrix W, we store the indices of the non-zero elements as indices I and non-zero values V in 2/4 bits. Since we employ structured pruning, we can encode the indices in 2 bits only. The nonzero values are quantized to 2/4 bits, and packed into 32-bit integers. Base weights, attention layer weights, and other parameters (e.g., LM head) remain in FP16.

2.2 Compressed Mixture of Experts Serving Engine

Base and Delta Decoupling. At inference time, we separate base and delta computations using an algebraic manipulation. Let Y be the SMoE layer output, W_b the base weight, α_i the gating function output, and W_i the weights of the i-th expert. Then, the SMoE computation can be expanded as:

$$Y = \sum_i \alpha_i W_i X = \sum_i \alpha_i (W_b + \Delta_i) X = W_b X + \sum_i \alpha_i \Delta_i \tag{1}$$

Base weight computations use a standard FP16 GEMM, while delta multiplications remain in a sparse low-precision format. A key advantage of this decoupling is that the deltas remain compressed during inference, which significantly reduces the data movement and memory footprint during inference. This has two main benefits: it enables loading larger MoEs under the same memory budget and allows for employing larger batch sizes.

Delta Computation We integrate the *FP16×INT4-2:4* Sparse Marlin kernel (Frantar et al., 2024) for delta computations. This particular kernel is optimized for 4-bit precision and structured pruning.

3 Experiments

We conduct two sets of experiments to evaluate the performance of our approach. First, we evaluate the post-compression model accuracy on a set of benchmark datasets. Second, we evaluate the inference speed and memory footprint of the compressed model on a real-world serving system.

Our experiments focus on merged SMoE models, where multiple models, each fine-tuned on a different dataset and all originating from the same base model, are integrated into a single SMoE model. While our method is effective in this setting due to the small differences between experts, it is not inherently limited to merged models – any scenario where expert weight variations are small can benefit from our compression approach. We create an SMoE model where we replace the MLP blocks with SMoE blocks. We merge models fine-tuned from *Llama 3.1-8B-Instruct* (Meta, 2024), and use it as the base. We merge them by averaging their attention weights and incorporating the feedforward layers from the fine-tuned models as expert weights within the SMoE blocks. The routing function is trained on a combined subset of the expert models' fine-tuning datasets. The models are constructed using `mergekit` (Goddard et al., 2025).

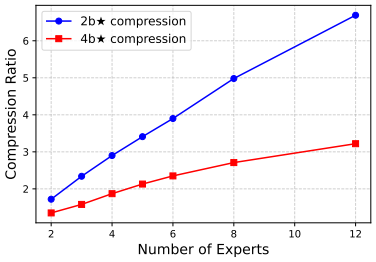

Figure 3: Compression ratio increases with the number of experts for a *Llama 3*-based SMoE.

3.1 Post-compression Model Quality and Model Size

#E	#Par	Size (GB)			BoolQ Accuracy			LogiQA Accuracy			MMLU Accuracy		
		FP16	4b⋆	2b⋆	FP16	4b⋆	2b⋆	FP16	4b⋆	2b⋆	FP16	4b⋆	2b⋆
2	14 B	27.3	20.3	18.8	84.4	83.6	83.5	31.3	31.8	31.0	68.5	64.5	64.4
3	19 B	38.6	24.5	16.5	83.8	83.7	83.6	31.5	31.2	31.5	68.4	68.5	68.4
4	25 B	49.9	24.5	17.2	84.4	84.4	84.7	31.6	31.8	32.4	68.6	68.6	68.5
5	31 B	61.2	28.7	17.9	84.6	84.5	84.6	32.1	31.2	32.0	68.7	68.7	68.8
6	36 B	72.4	30.9	18.9	84.5	84.4	84.5	32.7	32.3	32.0	68.7	65.1	68.6
8	47 B	95.0	35.1	20.9	85.1	85.0	84.9	31.0	31.2	30.6	68.6	68.7	68.5

Table 1: Model size and quality of `DeltaMoE` vs. uncompressed (FP16); ⋆ represents 50% structured sparsity.

Table 3.1 demonstrates the accuracy of the original model and post-compressed models. Albeit significant reduction in the size, we found the accuracies remain comparable after compression. Figure 3 shows the compression ratio as we add more experts. The compression ratio increases as we increase the number of experts, which allows serving significantly larger models within the same

memory budget. This is because the expert weights become the dominating factor in the total size of the model as we increase the number of experts, which is a key advantage of `DeltaMoE`.

3.2 SERVING PERFORMANCE

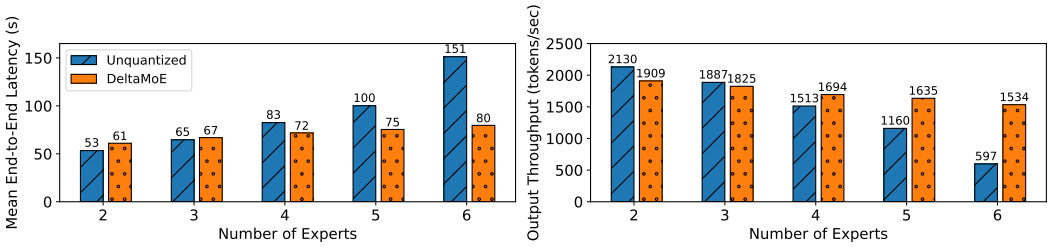

Figure 4: E2E latency and throughput for Poisson arrival rate $\lambda = 20$ and varying number of experts.

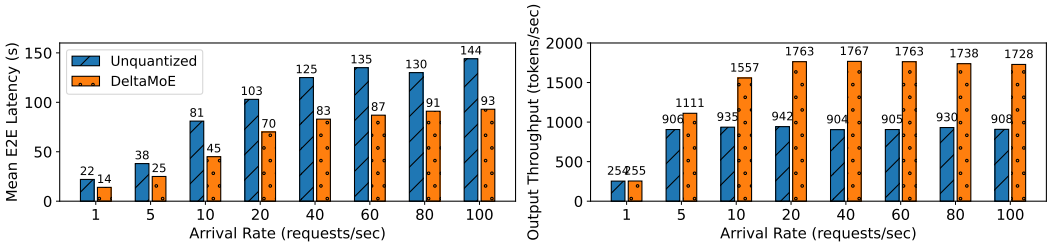

Figure 5: E2E latency and throughput for a 5 Expert SMoE with varying poisson arrival rate λ

We evaluate model performance using two key metrics: average end-to-end (E2E) latency and output throughput on an A100 SXM GPU. E2E latency is defined as the time elapsed from a client sending a prompt to receiving the last token of the response. Output throughput is defined as the average number of tokens generated per second. We use the SMoE model definition from HuggingFace Transformers (Wolf et al., 2020) for our baseline. We use a serving engine build on top of SGLang (Zheng et al., 2023) to serve both the baseline and the compressed models. To fit unquantized models on a single A100 GPU, we focus on evaluation of MoEs with up to six experts.

Figure 4 shows the relationship between the number of experts and performance. Increasing the number of experts leads to an increase in both speedup and throughput. For example, at 6 experts, we achieve a $1.91\times$ reduction in E2E latency and a $2.56\times$ increase in throughput. This gain largely stems from the lower memory footprint of `DeltaMoE`, enabling larger batch sizes. As the expert count rises, the batch size difference between `DeltaMoE` and the baseline widens further.

Another key factor is the decoupling of delta and base computations. On the one hand, it increases the number of FLOPs performed as we require to accumulate the results of the multiplication with the base weight and the delta weights. This overhead remains invariant to the number of experts. On the other hand, `DeltaMoE` replaces FP16 GEMM operations for the experts with more bandwidth-efficient and sparse *FP16-INT4-2:4* computations. As we increase the number of experts, the impact of the constant overhead decreases, reducing the E2E latency of `DeltaMoE`.

Figure 5 examines the performance of an SMoE with five experts across different request rates. `DeltaMoE` outperforms the baseline across all measured arrival rates. The largest speedups can be seen in the range between 5 and 20 requests per second, where the unquantized model has yet to reach its maximum throughput unlike the quantized one. We also observe a small latency reduction at an arrival rate of 1, when both models are operating at a small batch size. This suggests that the reduced data movement during matrix multiplication also contributes to the performance gains.

4 CONCLUSION

In this work, we introduce DeltaMoE, a **training-free** and efficient compression and serving technique for merged Sparse Mixture of Experts models, which significantly reduces the memory footprint without sacrificing model accuracy. Our evaluation showed that DeltaMoE achieved substantial improvement in both inference speed and memory efficiency, reaching a 1.88x speedup and 2.34x compression ratio for a 6 expert MoE. Our approach also scales when there are more experts in the network. One potential direction that we plan to explore in the future is extending and studying similar techniques on general SMoE models.

ACKNOWLEDGMENTS

This work was supported as part of the Swiss AI Initiative by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID a09 on Alps.

REFERENCES

Hongxiao Bai and Yun Li. Structured Sparsity in the NVIDIA Ampere Architecture and Applications in Search Engines, July 2023. URL https://developer.nvidia.com/blog/structured-sparsity-in-the-nvidia-ampere-architecture-and-applications-in-search-e

DeepSeek-AI. DeepSeek-V3 Technical Report, December 2024. URL http://arxiv.org/abs/2412.19437. arXiv:2412.19437 [cs] version: 1.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, June 2022. URL http://arxiv.org/abs/2101.03961. arXiv:2101.03961 [cs].

Elias Frantar and Dan Alistarh. Qmoe: Practical sub-1-bit compression of trillion-parameter models, 2023. URL https://arxiv.org/abs/2310.16795.

Elias Frantar, Roberto L. Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. Marlin: Mixed-precision auto-regressive parallel inference on large language models, 2024. URL https://arxiv.org/abs/2408.11743.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's MergeKit: A Toolkit for Merging Large Language Models, January 2025. URL http://arxiv.org/abs/2403.13257. arXiv:2403.13257 [cs].

Berivan Isik, Hermann Kumbong, Wanyi Ning, Xiaozhe Yao, Sanmi Koyejo, and Ce Zhang. Gpt-zip: Deep compression of finetuned large language models. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023.

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, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of Experts, January 2024. URL http://arxiv.org/abs/2401.04088. arXiv:2401.04088 [cs].

Maxime Labonne. Huggingface model page of phixtral-2x2_8, March 2024. URL https://huggingface.co/mlabonne/phixtral-2x2_8.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, Then Compress: Demystify Efficient SMoE with Hints from Its Routing Policy, March 2024. URL http://arxiv.org/abs/2310.01334. arXiv:2310.01334 [cs].

James Liu, Guangxuan Xiao, Kai Li, Jason D Lee, Song Han, Tri Dao, and Tianle Cai. Bitdelta: Your fine-tune may only be worth one bit. *arXiv preprint arXiv:2402.10193*, 2024a.

Jing Liu, Ruihao Gong, Mingyang Zhang, Yefei He, Jianfei Cai, and Bohan Zhuang. Me-switch: A memory-efficient expert switching framework for large language models, 2024b. URL https://arxiv.org/abs/2406.09041.

Meta. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen tau Yih, Jason Weston, and Xian Li. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm, March 2024. URL https://arxiv.org/abs/2403.07816. arXiv:2403.07816 [cs].

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771. arXiv:1910.03771 [cs].

Xiaozhe Yao, Qinghao Hu, and Ana Klimovic. Deltazip: Efficient serving of multiple full-model-tuned llms, November 2024. URL https://arxiv.org/abs/2312.05215. arXiv:2312.05215 [cs].

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody_Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Efficiently programming large language models using sglang. 2023.