From 2:4 to 8:16 sparsity patterns in LLMs for Outliers and Weights with Variance Correction

Egor Maximov[1,2,♣], Yulia Kuzkina[1,2,♣], Azamat Kanametov[1], Alexander Prutko[2], Maxim Zhelnin[3], Egor Shvetsov[4], Aleksei Goncharov[1,2]

[1]Moscow Independent Research Institute of Artificial Intelligence
[2]MIL Team [3]MWS AI [4]Applied AI
Correspondence: egor.maximov@mil-team.com
♣ indicates equal contribution.

Abstract

As large language models (LLMs) grow in size, efficient compression techniques like quantization and sparsification are critical. While quantization maintains performance with reduced precision, structured sparsity methods, such as N:M sparsification, often fall short due to limited flexibility and sensitivity to outlier weights. We explore 8:16 semi-structured sparsity, demonstrating its ability to surpass the Performance Threshold—where a compressed model matches the accuracy of its uncompressed or smaller counterpart under equivalent memory constraints. Compared to 2:4 sparsity, 8:16 offers greater flexibility with minimal storage overhead (0.875 vs. 0.75 bits/element). We also apply sparse structured patterns for salient weights, showing that structured sparsity for outliers is competitive with unstructured approaches, leading to equivalent or better results. Finally, we demonstrate that simple techniques such as variance correction and SmoothQuant-like weight equalization improve sparse models performance.

1 Introduction

Large language models (LLMs) are rapidly finding a wide range of applications (Veeramachaneni, 2025). However, as these models continue to scale in size and complexity, significant attention has been devoted to studying model compression techniques, particularly quantization and semi-structured sparsification (Frantar et al., 2025). The **Performance Threshold** is defined as the point at which a compressed model maintains comparable accuracy to its uncompressed or smaller counterpart, with equavalent memory usage (Frantar et al., 2025). Quantized models often surpass this threshold by effectively reducing the precision of weights while maintaining or even enhancing performance (Li et al., 2023; Frantar et al., 2025). In contrast, structured sparsity methods, such as N:M sparsification, typically struggle to meet this threshold for several reasons:

- **Limited Flexibility:** In the N:M scheme, N out of M elements in each block must be zeroed, which can restrict the model's adaptability (Zhang et al., 2022).
- **Lack of Hardware Support:** Only 2:4 structured sparsity is natively supported on mainstream GPUs, narrowing research to this specific pattern (Hu et al., 2024).
- **Potential Outlier Removal:** Removing or perturbing even a few outlier weights can disproportionately degrade model accuracy (Dettmers et al., 2023b).

Consequently, recent compression efforts increasingly favor quantization for its reliability, leaving sparsification methods underdeveloped (Harma et al., 2024; Frantar et al., 2025). This gap motivates our investigation into **8:16 semi-structured sparsity**. We aim to determine whether 8:16 sparsity can surpass the Performance Threshold and demonstrate its viability for real-world applications. While technical implementation details are beyond this work's scope, our analysis focuses on performance to pave the road for the future research.

Structured Sparse Outliers To mitigate sparsification-induced degradation, methods like SPQR restore salient weights via an auxiliary sparse matrix (e.g., CSR) (Dettmers et al., 2023b). While salient weights typically constitute only 1–5% of total weights, unstructured sparsity introduces inefficiencies: irregular memory access limits cache utilization, and metadata overhead grows linearly with non-zeros. In contrast, structured sparsity patterns (e.g., 4:256, 8:256, 16:256) enable predictable memory access and reduced metadata, improving hardware efficiency (Schulte et al., 2023). These patterns correspond to sparsity

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pages 957–965

levels of 1.5%, 3.1%, and 6.25%, respectively, balancing flexibility with practical constraints. In this work we aim to answer weather structured sparsity formats are enough to store outliers.

Our contributions:

(1) **We empirically show that the 8:16 sparsity pattern improves model performance.** Specifically, a sparse LLaMa-2-13B model using this pattern achieves the same performance as the dense LLaMa-2-7B model This suggests promising implications for future hardware development with 8:16 sparsity patterns support.

(2) **Structured Sparsity for Salient Weights (SSP FOR SW):** We propose and analyze SSP FOR SW, a structured sparsity patterns for critical weight subsets. This approach, in our experiments, not only enhances computational efficiency but also enables models with outliers stored in structured patterns (SSP) to outperform those using unstructured sparsity for salient weights.

(3) We propose two techniques for pre and post processing for weights sparsification. *SmoothQuant-Inspired Rebalancing:* We preprocess weights and activations before sparsification, adapting the SmoothQuant philosophy to balance distributions and improve robustness. *Variance Correction:* We introduce post-pruning variance adjustments to mitigate performance degradation caused by weight removal—a simple yet effective technique absent in existing literature[1].

2 Sparsity Pattern Selection and Hardware Implications

When designing sparsity patterns for practical deployment, we must balance performance with hardware implementation constraints. While numerous sparsity patterns exist theoretically, hardware-efficient patterns must satisfy several requirements:

- **Memory Alignment**: Patterns should align with hardware memory boundaries (typically powers of two)
- **Metadata Overhead**: Pattern selection bits must be efficiently packable
- **Circuit Complexity**: Decoding logic should be implementable with reasonable silicon area

Table 1 compares key characteristics of common N:M patterns. The 8:16 pattern offers an optimal balance between flexibility and hardware efficiency. While 16:32 achieves marginally better perplexity, its hardware implementation would require significantly more complex circuitry with diminishing performance returns.

Pattern	Configurations	Bits/Element	PPL RIA	PPL RIA+VC
2:4	6	0.75	22.526	16.66
4:8	70	0.81	12.80	11.58
8:16	**12,870**	**0.88**	**10.64**	**9.95**
16:32	601,080,390	1.00	9.98	9.51

Table 1: **Hardware considerations and performance.** Comparison of N:M sparsity patterns for LLaMA3-8B model. Perplexity (PPL) measured on WikiText-2 using RIA and RIA+VC.

From a hardware perspective, both 2:4 and 8:16 patterns provide a 50% theoretical FLOPs reduction and 2× reduction in memory bandwidth requirements. While empirical measurements for 8:16 acceleration aren't yet possible (as no hardware currently implements this pattern), we can project its performance based on the established relationship between memory bandwidth reduction and inference acceleration in sparse architectures. The existing 2:4 pattern achieves from ∼1.5–2× inference acceleration scaling with matrix size, and we expect similar scaling for 8:16 when implemented in silicon.

The selection of 8:16 over other patterns is deliberate. While 16:32 shows marginally better perplexity (Table 1), the main jump occurs from 4:8 to 8:16 yielding a significantly larger improvement. Moreover, hardware implementation of 16:32 would require substantially more complex circuitry for minimal performance gain, making 8:16 the pragmatic optimum for future hardware development.

3 Related work

Sparsity Patterns: (Hu et al., 2024) introduced a new 2:4 (50%) sparsity pattern for maintaining accuracy while achieving 2x math throughput for GEMM-like operations. (Zhao et al., 2024) presents a V:N:M sparsity pattern that achieves higher sparsity and faster computation than standard N:M patterns. Since widely available hardware supports only 2:4 patterns, most of recent research mainly studied this pattern.

Sparsification: Identifying weights importance[2] is a central challenge in weight pruning. Several

[1] To the best of our knowledge.

[2] In our work, we use the terms **salient weights** and weight **outliers** interchangeably for very important weights.

recent methods have been proposed for identifying such weights in LLMs, including SparseGPT (Frantar and Alistarh, 2023), Wanda (Sun et al., 2023), and OWL (Yin et al., 2023). Further RIA (Zhang et al., 2024) improves upon earlier approaches by evaluating the relative importance of each weight within its row and column before pruning.

Outliers: Dettmers et al. demonstrated that a small number of activation outliers can significantly impact LLM performance, highlighting the link between activation patterns and salient weights. Many subsequent post-training quantization methods adopted similar or identical metrics to identify these key weights (Dettmers et al., 2023b; Xiao et al., 2023; Lee et al., 2024). In addition, many approaches were developed to avoid distortions in these outliers during quantization or sparsification by isolating them. For example, Zhelnin et al. indentifies columns with most outliers and does not quantize these columns, Dettmers et al. stores identified salient weights in a separate sparse matrix, applies sparse matrix multiplication, and quantizes the remaining weights with minimal performance degradation. Following these studies we evaluate feasibility of storing outliers in semi-strucutred N:M patterns.

Post-Compression Fine-Tuning: Weight pruning or quantization typically leads to some model degradation (Kharinaev et al., 2025) even if salient weights were isolated (Zhelnin et al., 2024). Therefore, fine-tuning is often applied afterward. However, full-model fine-tuning is computationally prohibitive and efficient fine-tuning methods have emerged, these methods can be categorized as follows: (1) **Adapter-based** methods like QLoRA (Dettmers et al., 2023a), where low-rank adapters are trained. (2) **Selective-based** methods like GIFT-SW (Zhelnin et al., 2024), where only a small subset of weights is fine-tuned. (3) **Block-based** methods like EBFT (Guo et al., 2024), where the model is split into independent blocks, each fine-tuned separately.

4 Methods

In our preliminary experiments[3], we identified the most effective sparsification strategy through an ablation study. Our proposed **general pipeline** consists of four key stages:

1. **Weights Equalization**: Apply channel-wise scaling $S(W) \rightarrow W_{\text{ec}}$ using to equalize weight magnitudes, as described in Equation 1.

2. **Importance-Aware Pruning**: Employ the RIA method on scaled weights to identify and preserve salient weights W_{salient} (1.5%–6.2% of total weights) using high compression structured sparsity patterns $[4{:}256, 8{:}256, 16{:}256]$. Non-salient weights $W_{\neg\text{salient}}$ are pruned using 2:4 or 8:16 sparsity patterns.

3. **Variance Correction**: Compensate for distribution shifts in pruned weights through variance-preserving rescaling of $W_{\neg\text{salient}}$ as defined in Equation 2.

4. **Blockwise Fine-Tuning**: Perform error-bound aware tuning using EBFT (Guo et al., 2024), updating only $W_{\neg\text{salient}}$ and BatchNorm parameters while maintaining sparsity patterns through a fixed binary mask M.

4.1 SmoothQuant Adaptation

While SmoothQuant (Xiao et al., 2023) was designed for quantization, we adapt it for sparsification by solving the inverse problem: rather than normalizing weights for discretization, we redistribute importance scores between activations and weights.

Given the weight matrix $W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$ and the input $x \in \mathbb{R}^{C_{\text{in}}}$, we compute channel-wise scaling factors $s_j = \max(|x_j|)/\max(|W_{:,j}|)$ for $j = 1, ..., C_{\text{in}}$. The diagonal scaling matrix $S = \text{diag}(s_1, \ldots, s_{C_{\text{in}}})$ transforms:

$$W_{\text{ec}} = W \cdot S^{-1}, \quad x_{\text{scaled}} = x \cdot S \qquad (1)$$

This transformation preserves mathematical equivalence ($W_{\text{ec}} x_{\text{scaled}} = Wx$) while enabling clearer separation of salient vs. non-salient weights through magnitude redistribution.

Implementation Note: We only use W_{ec} to calculate a metric to evaluate the importance of weights. The actual weights and model activations do not change.

4.2 Variance Correction (VC)

Motivated by Nagel et al.'s bias correction for quantization, we propose variance correction to address distribution shifts from weight pruning:

[3]Preliminary experiments excluded due to space constraints; full ablation study available in supplementary materials.

$$W_{\neg\text{salient}}^{(\text{corrected})} = W_{\neg\text{salient}} \cdot \sqrt{\frac{\text{Var}(W_{\text{dense}})}{\text{Var}(W_{\neg\text{salient}}) + \epsilon}} \tag{2}$$

This rescaling preserves the original weight variance in pruned layers, maintaining stable activation statistics. Unlike bias correction, which requires learnable bias terms, our variance correction remains applicable to architectures without bias parameters.

5 Models and Datasets

Our study evaluates three language model families: LLaMA2 7B/13B (Touvron et al., 2023), LLaMA3 8B (Grattafiori et al., 2024), and Mistral 7B (Jiang et al., 2023), which incorporate distinct architectural variations. While all models share foundational transformer-based designs, their implementations differ in attention mechanisms and optimization strategies, enabling comparative analysis of performance characteristics across model scales and design paradigms.

For evaluation, we calculate perplexity scores on the validation splits of two standard text corpora: WikiText (Merity et al., 2016) and C4 (Raffel et al., 2020). To comprehensively assess reasoning capabilities, we employ five zero-shot benchmarks following established evaluation protocols: ARC Easy and Challenge (Clark et al., 2018) for structured knowledge reasoning, WinoGrande (Sakaguchi et al., 2021) for commonsense inference, HellaSwag (Zellers et al., 2019) for contextual understanding, and PIQA (Bisk et al., 2020) for physical interaction reasoning. The choice of baselines is similar to those in previous studies (Frantar and Alistarh, 2023; Lee et al., 2024; Frantar et al., 2022; Xiao et al., 2023; Zhelnin et al., 2024).

6 Results

This section presents the results of computational experiments. (1) We analyze the effectiveness of SmoothQuant (SQ) adaptation and Variance Correction (VC). (2) We evaluate a semi-structured 4:256 sparsity pattern for storing weight outliers. (3) We compare weight sparsification with 2:4 and 8:16 patterns, both with and without fine-tuning to demonstrate effectivness of 8:16 pattern. (4) Finally, we contrast structured and unstructured sparsity to validate semi-structured patterns for outlier storage.

① **Variance Correction and Weight Equalization with SmoothQuant.** To evaluate VC and SQ, we measured perplexity in the pruned LLaMA2-7B model (Table 4). Both methods reduced perplexity versus standard RIA pruning on C4 and Wikitext2 datasets, with VC being more efficeint. However, layer-wise fine-tuning via EBFT yielded the most significant improvement, outperforming both VC and SQ. Notably, all methods independently enhanced performance, and their combination achieved the lowest perplexity across datasets.

② **Structured Outlier Storage for RIA and Magnitude-Based Sparsification.** We recovered salient weights in semi-structured 2:4-pruned linear layers of LLaMA2-7B/13B models, storing outliers in 4:256 patterns. As Table 5 shows, preserving outliers—even under magnitude-based pruning—improves model performance. This section focuses on magnitude-based approaches to isolate implicit effects of other pruning techniques. It is evident that recovering even a small number of salient weights using the 4:256 format significantly improves perplexity for both models. Thus, structural recovery of salient weights can be integrated with various pruning methods to enhance the generative capabilities of pruned models.

③ **Comparison of 2:4 and 8:16 structured patterns.** Here, we compare perforomance for all models on reasoning tasks for two sparsity patterns 2:4 and 8:16.

We perform structural recovery of salient weights in the linear layers of language models subjected to semi-structured 2:4 and 8:16 pruning patterns. For each linear layer, salient weights are extracted in structural pruning patterns of 4:256, 8:256, and 16:256 and stored in a separate matrix.

LLaMA2-7B and LLaMA2-13B. Tables 2 and 3 present the mean accuracy across five zero-shot tasks for LLaMA2-7B and LLaMA2-13B. The results demonstrate that increasing the number of salient weights consistently improves model accuracy for both the RIA+SQ and RIA+SQ+VC+EBFT pruning methods when using the wikitext2 dataset. However, a slight decrease in accuracy is observed for RIA+SQ+VC+EBFT on the c4 dataset. The best performance for pruned models is achieved when isolating salient weights in the 16:256 pattern.

LLaMA3.1-8B and Mistral. Since wikitext2 yields better results with an increasing number of salient weights, we select this dataset for calibrating more modern models, such as LLaMA3.1-8B

Outliers:	4:256		8:256		16:256	
Sparsity:	**2:4**	**8:16**	**2:4**	**8:16**	**2:4**	**8:16**
C4						
RIA+SQ	55,97%	**61,04%**	57,48%	**61,18%**	59,16%	**61,81%**
RIA+SQ+VC+EBFT	57,87%	**61,01%**	58,72%	**60,95%**	59,74%	**61,91%**
WikiText2						
RIA+SQ	56,44%	**60,56%**	57,40%	**61,12%**	59,41%	**62,15%**
RIA+SQ+VC+EBFT	57,84%	**61,08%**	58,53%	**61,27%**	60,20%	**62,57%**

Table 2: Mean accuracy for LLaMA2-7B with calibrations sets as C4 and WikiText2 on zero-shot tasks: ARC-c, ARC-e, PIQA, Winogrande, Hellaswag. Mean performance for dense model is 64.79%.

Outliers:	4:256		8:256		16:256	
Sparsity:	**2:4**	**8:16**	**2:4**	**8:16**	**2:4**	**8:16**
C4						
RIA+SQ	60.18%	63.94%	61.19%	64.52%	62,96%	65,23%
RIA+SQ+VC+EBFT	61.50%	64.52%	62.18%	64.16%	63,34%	65,13%
WikiText2						
RIA+SQ	60.28%	63.70%	60.92%	64.03%	63,03%	65,03%
RIA+SQ+VC+EBFT	61.38%	63.63%	61.84%	63.76%	64,01%	65,40%

Table 3: Mean accuracy for LLaMA2-13B with calibration sets as C4 and WikiText2 on zero-shot tasks: ARC-c, ARC-e, PIQA, Winogrande, Hellaswag. Mean performance for the dense model is 67.77%.

Method	C4	WikiText2	Mean
Dense Model*	**5.47**	**5.47**	**5.47**
Magnitude*	37.77	37.96	37.87
RIA*	11.27	10.91	11.09
RIA +VC	9.21	8.92	9.07
RIA + SQ*	10.71	10.22	10.47
RIA + EBFT*	8.73	8.47	8.60
RIA + SQ + EBFT	8.67	8.40	8.54
RIA + SQ +VC+EBFT	7.96	7.95	7.96

Table 4: Perplexity results for LLaMA2-7B on C4 and WikiText2 datasets with 2:4 sparsity. The dense model achieves a perplexity of 5.47. Perplexity of RIA is improved by 28% with all of the modifications applied. Baseline methods are specified with asterisks.

Outliers	LLaMA2-7B	LLaMA-13B
0%	37.96	18.46
1.56% (4:256)	23.06	14.59

Table 5: Perplexity results for *magnitude based pruning* of LLaMA2-7B and LLaMA-13B models, with WikiText2 calibrration dataset, 2:4 sparsity pattrern is used.

and Mistral, during pruning. Table 6 shows the perplexity on wikitext2 for **LLaMA3.1-8B** and **Mistral** under 2:4 and 8:16 pruning patterns, depending on the number of structural salient weights. For Mistral, we omitted VC as it significantly degrades generative capabilities. In contrast, VC substantially improves perplexity for LLaMA3.1-8B. Mistral demonstrates superior robustness to pruning compared to LLaMA3. Under the 2:4 pruning pattern with the RIA+SQ method, LLaMA3's perplexity increases significantly, by up to 3.07× (from 6.24 to 19.13), whereas Mistral's perplexity rises by only 1.75× (from 5.32 to 9.34). For the 8:16 pruning pattern, the increase in perplexity is more moderate, with Mistral experiencing a 1.27× increase and LLaMA3 a 1.69× increase. Fine-tuning with EBFT enhances both models, regardless of the pruning pattern. Specifically, for LLaMA3, the increase in perplexity is limited to 1.68× (from 6.24 to 10.49), and for Mistral, it is reduced to 1.33× (from 5.32 to 7.10).

The incorporation of structural salient weights in the 2:4 and 8:16 pruning patterns significantly improves perplexity. We can observe this for both models. Increasing the number of recovered salient weights allows for a substantial reduction in perplexity following pruning. The models achieve optimal perplexity metrics when employing a 16:256 recovery scheme in conjunction with EBFT. In this scenario, LLaMA3's perplexity metrics are 8.24 and 7.40, while Mistral exhibits perplexity values of 6.27 and 5.82, depending on the pruning pattern used (2:4 or 8:16).

Outliers:	-		4:256		8:256		16:256	
Sparsity:	**2:4**	**8:16**	**2:4**	**8:16**	**2:4**	**8:16**	**2:4**	**8:16**
LLaMA3 (PPL=6.24)								
RIA+SQ	19.13	10.55	16.50	9.85	14.28	9.32	11.48	8.52
RIA+SQ+VC	16.19	10.09	14.47	9.52	12.73	9.11	10.68	8.42
RIA+SQ+VC+EBFT	10.49	8.08	9.18	7.88	9.53	7.69	8.24	7.40
Mistral (PPL=5.32)								
RIA+SQ	9.34	6.74	8.48	6.52	7.80	6.34	6.97	6.06
RIA+SQ+EBFT	7.10	6.14	6.86	6.06	6.59	5.96	6.27	5.82

Table 6: Perplexity for **LLaMA3-8B** and **Mistral** with WikiText2 as calibration set on different sparsity and outliers type.

These conclusions are further supported by the accuracy results of LLaMA3 and Mistral on zero-shot tasks. The table demonstrates that combining pruning with EBFT on Wikitext-2, and salient weight recovery enhances model performance. For the 16:256 recovery scheme, LLaMA3 shows a minimal accuracy drop of 7.48% and 4.34% (from 69.23 to 61.75 and from 69.23 to 64.89), while Mistral experiences drops of 6.43% and 3.28% (from 68.55 to 62.11 and from 68.55 to 65.27), depending on the pruning pattern used (2:4 or 8:16).

Structured VS unstructured salient weights. To emphasize the effectiveness of recovering salient weights in a structural format, we compare the performance of models subjected to pruning using RIA, SQ, VC, and EBFT methods, alongside both structural and unstructural salient weight recovery. For this comparison, the number of salient weights recovered in the unistructural format is kept comparable to that of the structural format, with EBFT conducted on Wikitext-2. The results are a presnted in Table 7. We see that both structural and unstructured weight recovery enhance model performance following pruning. However, the use of a structured format consistently yields better metrics compared to unstructured salient weights. Therefore, structural salient weights not only facilitate more efficient model inference but also contribute to improved model performance after pruning compared to their unstructured counterparts.

7 Conclusion

This work demonstrates that structured sparsity, when combined with targeted techniques for salient weight preservation, enables efficient LLM deployment without compromising performance. Our key findings are:

- **8:16 Sparsity Provides Optimal Efficiency:** The 8:16 pattern delivers superior performance-efficiency tradeoffs. Our sparse LLaMa-2-13B matches the dense LLaMa-2-7B baseline while reducing computational overhead, validating its practical value for resource-constrained deployment.

- **Structured Sparsity for Salient Weights (SSP FOR SW) Outperforms Unstructured Approaches:** Recovering critical weights in structured patterns (4:256/8:256/16:256) consistently improves perplexity and accuracy across models. This approach yields better performance than unstructured sparsity while maintaining hardware efficiency.

- **Pre/Post-Processing Techniques:** *SmoothQuant-Inspired Rebalancing* mitigates pruning-induced distribution shifts. Our novel *Variance Correction* technique compensates for activation distribution changes. Combined with layer-wise *EBFT* fine-tuning, these achieve the lowest perplexity (6.27 for Mistral with 8:16 sparsity).

8 Limitations

The 8:16 sparsity pattern lacks current hardware support, however this work advocates for the development of such hardware desing in the future. Our evaluation is restricted to only three models (LLaMA-2/3, Mistral) on zero-shot reasoning benchmarks, limiting generalizability to smaller models, or tasks like instruction following or code generation. Key ablation studies were omitted due to space constraints, obscuring the individual impact of components such as Variance Correction and SmoothQuant-inspired rebalancing.

9 Acknowledgement

This work was supported by a grant for research center in the field of artificial intelligence, provided by the Ministry of Economic Development of the Russian Federation (agreement No. 139-15-2025-013, dated June 20, 2025, subsidy identifier 000000C313925P4B0002).

References

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

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

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023a. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023b. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10323–10337.

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

Elias Frantar, Utku Evci, Wonpyo Park, Neil Houlsby, and Dan Alistarh. 2025. Compression scaling laws: Unifying sparsity and quantization. *arXiv preprint arXiv:2502.16440*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Song Guo, Fan Wu, Lei Zhang, Xiawu Zheng, Shengchuan Zhang, Fei Chao, Yiyu Shi, and Rongrong Ji. 2024. Ebft: Effective and block-wise fine-tuning for sparse llms. *arXiv preprint arXiv:2402.12419*.

Simla Burcu Harma, Ayan Chakraborty, Elizaveta Kostenok, Danila Mishin, Dongho Ha, Babak Falsafi, Martin Jaggi, Ming Liu, Yunho Oh, Suvinay Subramanian, and 1 others. 2024. Effective interplay between sparsity and quantization: From theory to practice. *arXiv preprint arXiv:2405.20935*.

Yuezhou Hu, Kang Zhao, Weiyu Huang, Jianfei Chen, and Jun Zhu. 2024. Accelerating transformer pre-training with 2: 4 sparsity. *arXiv preprint arXiv:2404.01847*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Artyom Kharinaev, Viktor Moskvoretskii, Egor Shvetsov, Kseniia Studenikina, Bykov Mikhail, and Evgeny Burnaev. 2025. Investigating the impact of quantization methods on the safety and reliability of large language models. *arXiv preprint arXiv:2502.15799*.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2024. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364.

Zhuo Li, Hengyi Li, and Lin Meng. 2023. Model compression for deep neural networks: A survey. *Computers*, 12(3):60.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. 2019. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1325–1334.

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

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

Christoph Schulte, Sven Wagner, Armin Runge, Dimitrios Bariamis, and Barbara Hammer. 2023. Best of both, structured and unstructured sparsity in neural networks. In *Proceedings of the 3rd Workshop on Machine Learning and Systems*, pages 104–108.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vinod Veeramachaneni. 2025. Large language models: A comprehensive survey on architectures, applications, and challenges. *Advanced Innovations in Computer Programming Languages*, 7(1):20–39.

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

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. 2023. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. In *Conference on Parsimony and Learning (Recent Spotlight Track)*.

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

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. 2024. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*.

Yuxin Zhang, Mingbao Lin, Zhihang Lin, Yiting Luo, Ke Li, Fei Chao, Yongjian Wu, and Rongrong Ji. 2022. Learning best combination for efficient n: M sparsity. *Advances in Neural Information Processing Systems*, 35:941–953.

Kang Zhao, Tao Yuan, Han Bao, Zhenfeng Su, Chang Gao, Zhaofeng Sun, Zichen Liang, Liping Jing, and Jianfei Chen. 2024. Beyond 2: 4: exploring v: N: M sparsity for efficient transformer inference on gpus. *arXiv preprint arXiv:2410.16135*.

Maxim Zhelnin, Viktor Moskvoretskii, Egor Shvetsov, Egor Venediktov, Mariya Krylova, Aleksandr Zuev, and Evgeny Burnaev. 2024. Gift-sw: Gaussian noise injected fine-tuning of salient weights for llms. *arXiv preprint arXiv:2408.15300*.

Appendix

This appendix presents additional experimental results comparing different sparsity patterns and quantization-aware fine-tuning strategies across several large language models.

Table 7 reports the zero-shot accuracy of LLaMA-2-7B and LLaMA-2-13B under varying levels of sparsity specifically, 4/256 (≈0.0156%), 8/256 (≈0.0313%), and 16/256 (≈0.0625%) using both unstructured and semi-structured pruning approaches. All models are calibrated on the WikiText dataset. Results indicate that semi-structured sparsity consistently matches or slightly outperforms unstructured sparsity across both model sizes and sparsity levels, suggesting that structured pruning can preserve task performance while offering hardware-friendly compression benefits.

Table 8 evaluates the impact of progressively incorporating advanced compression techniques including salient weight identification (RIA), smooth quantization (SQ), variance-aware calibration (VC), and efficient bias fine-tuning (EBFT) on zero-shot performance for LLaMA3-8B and Mistral. Accuracy is averaged across five standard benchmarks: ARC-c, ARC-e, PIQA, Winogrande, and Hellaswag, with WikiText2 used for calibration. The baseline accuracies (without compression) are 69.23% for LLaMA3-8B and 68.55% for Mistral. The results demonstrate that combining RIA+SQ with VC and EBFT yields consistent improvements over simpler pipelines, particularly at higher sparsity levels (e.g., 16:256 outliers with 2:4 or 8:16 block sparsity). Notably, Mistral shows strong resilience to compression, maintaining performance close to its uncompressed baseline even under aggressive sparsification.

Sparsity (%):	**4/256** (≈ 0.0156%)		**8/256** (≈ 0.0313%)		**16/256** (≈ 0.0625%)	
LLaMA-2-7B						
Unstructured	56.43%	60.08%	57.55%	60.55%	59.09%	61.88%
Semi-structured	**57,84%**	**61,08%**	**58,53%**	**61,27%**	**60,20%**	**62,57%**
LLaMA-2-13B						
Unstructured	61.37%	62.94%	**62.03%**	63.48%	62.796%	64.20%
Semi-structured	**61.38%**	**63.63%**	61.84%	**63.76%**	**64,01%**	**65,40%**

Table 7: Comparision between unstructured and semi structured sparsity patterns. Accuracy is reported for LLaMA-2 models under varying sparsity levels correspoining to following sparsity patterns: 4/256, 8/256/ and 16/266. WikiText is used for calibration.

Outliers:	**-**		**4:256**		**8:256**		**16:256**	
Sparsity:	**2:4**	**8:16**	**2:4**	**8:16**	**2:4**	**8:16**	**2:4**	**8:16**
LLaMA3 (acc=69.23%)								
RIA+SQ	51.16%	59.71%	52.85%	60.60%	54.75%	61.81%	58.01%	63.75%
RIA+SQ+VC	52.85%	60.19%	54.25%	60.98%	55.97%	62.29%	58.87%	63.98%
RIA+SQ+VC+EBFT	55.78%	62.01%	58.41%	62.71%	58.95%	63.20%	61.75%	64.89%
Mistral (acc=68.55%)								
RIA+SQ	56.72%	63.04%	58.30%	63.81%	59.12%	64.09%	61.38%	64.87%
RIA+SQ+EBFT	58.49%	63.63%	59.26%	64.07%	60.68%	64.48%	62.11%	65.27%

Table 8: Mean accuracy for **LLaMA3-8B** and **Mistral** with WikiText2 as calibration set on zero-shot tasks: ARC-c, ARC-e, PIQA, Winogrande, Hellaswag.