Proceedings of CONF-SEML 2026 Symposium: Multimodal Data Acquisition: Applications in Physiological and Behavioral Research
DOI: 10.54254/2755-2721/2026.AD31512

Survey: Training-Free Structured Compression of Large Language Models

Zixuan Shao

Glasgow College, University of Electronic Science and Technology of China, Chengdu, China
2022190505021@std.uestc.edu.cn

Abstract. The well-known Large Language Model (LLM) compression is essential for enhancing computational efficiency, yet a systematic summary of investigation into structured pruning and low-rank decomposition remains absent in current literature. This work addresses the gap by providing a comprehensive review specifically focused on these two methodologies. Representative approaches are categorized and evaluated, including LLM-Pruner and SlimGPT for structured pruning, and ASVD and SVD-LLM for decomposition. These methods are rigorously analyzed in terms of algorithmic design, accuracy retention, and hardware adaptability. Through unified evaluation and comparative analysis, DISP-LLM and MoDeGPT are identified as the current state-of-the-art within their respective fields. Consequently, a conceptual framework is established to provide practical guidance for future research into efficient, training-free, and scalable LLM compression.

Keywords: LLM, Structured Pruning, Low-Rank Decomposition, Parameter Reduction

1. Introduction

The development of large language models (LLMs) such as GPT-4, PaLM, and LLaMA has changed the artificial intelligence research including natural language understanding, dialogue systems, code generation, and multimodal reasoning [1-3]. Actually, the trade-off for the sophisticated reasoning that seen in today's models is their sheer size. That is to say, operating billions of parameters demands a level of power and memory that many real-world applications simply cannot sustain. In order to bridge the gap between high-end performance and edge-side deployment, prioritize compression techniques that trim the fat without losing the intelligence are called for.

Structurally, LLM lifecycles transition from broad-scale pre-training to more nuanced post-training refinements, for example, the alignment and instruction tuning [4,5]. Within this pipeline, compression has shifted from an elective optimization to a vital necessity. Indeed, unlike behavioral fine-tuning, which recalibrates how a model responds, compression intervenes at the representational level to curb computational overhead, making the deployment of these massive systems economically and technically viable. (see Figure 1)

Model compression for LLMs encompasses a wide spectrum of approaches that can be categorized along several dimensions. Training-based methods, such as knowledge distillation, rely on additional optimization stages, whereas training-free approaches, including pruning and decomposition, operate directly on pretrained weights without retraining.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.AD31512

Moreover, methods can be structured or unstructured, where structured compression (e.g., layer pruning, block removal) maintains hardware-friendly patterns that facilitate speedup, while unstructured approaches (e.g., sparse pruning) primarily save storage but yield limited runtime benefits. Among these, pruning-based and low-rank decomposition- based methods have shown particular promise due to their balance between interpretability, efficiency, and compatibility with existing architectures. Quantization, although widely studied, focuses more on numerical representation and thus falls slightly outside the scope of this survey.

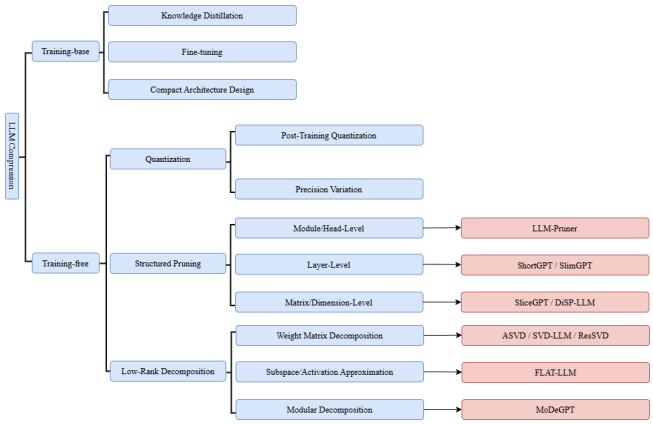

Figure 1. Overall taxonomy of Large Language Model (LLM) compression methods

In the literatures, it is seen that the current surveys [6-8] normally categorize LLM compression into quantization, pruning, and distillation. While they always [6] emphasizes inference efficiency and tuning-free methods that overlooks low-rank decomposition and offers only a high-level view of structured pruning. Similarly, Tang et al. [7] focuses on Transformer-specific pruning but lacks a unified framework to distinguish between varying granularities and design objectives. It is worth noting that Park et al. [8] include low-rank approximation, their analysis is too coarse-grained to differentiate modern fine-grained pruning schemes or incorporate the latest decomposition breakthroughs.

To address these present limitations in the above-mentioned reviews articles and books, this paper will provides a systematic and in-depth review of the most recent LLM compression methods, with a particular emphasis on structured pruning and low-rank decomposition, two complementary yet rapidly evolving families of techniques. Structured pruning removes redundant components such as layers, attention heads, or neuron groups, thereby achieving hardware-friendly acceleration, while low-rank decomposition approximates large weight matrices with compact factorized representations, reducing both memory footprint and computational cost. By bridging these two perspectives, this work not only synthesizes their theoretical principles and empirical outcomes but also highlights their interrelations, advantages, and practical trade-offs.

In summary, the contributions of this work are threefold: i) In this paper, a unified taxonomy and conduct a detailed examination of structured pruning methods for LLMs are established, including LLM-Pruner [9], ShortGPT [10], and SliceGPT [11], analyzing their structural granularity, pruning strategies, and acceleration effectiveness; ii) Then, a comprehensive review of recent low-rank decomposition methods such as ASVD [12], SVD- LLM [13], and MoDeGPT [14], elucidating their mathematical formulations, optimization strategies, and layer-specific adaptations to transformer architectures; iii) Last but not least it is provided unified evaluations and critical comparisons across existing methods, discussing their trade-offs among compression ratio, model fidelity, and

DOI: 10.54254/2755-2721/2026.AD31512

computational efficiency, while identifying promising research directions toward more principled and robust LLM compression frameworks.

2. Background

2.1. Foundations of model compression

It is well-known that the scaling of Transformer-based LLMs necessitates compression frameworks that balance efficiency with accuracy. These efforts rest on two theoretical pillars: the one is sparsity, and the other is low-dimensionality. For the sparsity, it posits that only a fraction of parameters are functionally essential. For the low-dimensionality, it suggests high-dimensional weights can be projected into compact subspaces. By framing redundancy through these lenses, the researcher can move beyond specific e algorithms to establish a generalized foundation for structural optimization.

2.2. Unstructured vs structured compression vs

Generally, a central distinction in sparsity-oriented compression lies in how sparsity is imposed on the parameter space. i) For the Un-structured compression, it treats each weight as an independent variable, but often producing irregular patterns that offer limited hardware acceleration; ii) For structured compression, it normally introduces sparsity at the level of coherent computational units —such as neurons, channels, attention heads, or entire blocks—resulting in regular sparsity patterns that align more effectively with modern hardware execution models. This distinction captures different assumptions about where redundancy resides within the architecture and forms a conceptual taxonomy for evaluating trade-offs between expressiveness, inference efficiency, and architectural coherence. By framing compression techniques within this structured–unstructured dichotomy, subsequent sections can situate specific methods within a well-defined conceptual space.

2.3. Mathematical formulation

Model compression is frequently formalized through constrained optimization or low-dimensional approximation. Sparsity-oriented approaches can be expressed as the following cardinality-constrained optimization problem:

$$min_{\theta'} L\left(\theta'\right) \quad s.t. \quad \| \theta' \|_0 \leq s, \tag{1}$$

where θ' denotes the compressed parameter vector and s defines the target number of nonzero parameters. This formulation is general and encompasses both structured and unstructured sparsity depending on how the constraint is applied to individual parameters or predefined groups.

In parallel, low-dimensional compression is commonly described through matrix approximation. Given a weight matrix $W \in R^{m \times n}$, its low-rank characterization takes the form

$$W \approx U_k \Sigma_k V_k^{\top}, \tag{2}$$

which represents the best rank-k approximation of W under the Frobenius norm. This formulation highlights the role of intrinsic correlations within weight matrices and provides a mathematical basis for replacing high-dimensional linear transformations with compact, low-rank representations.

DOI: 10.54254/2755-2721/2026.AD31512

In summary, these frameworks establish the theoretical basis for model compression by defining the structural and geometric redundancies inherent in LLMs. Also, the above-mentioned principled grounding allows the following sections to explore specific compression strategies through a rigorous, unified lens.

3. Structured pruning

3.1. Taxonomy of structured pruning

The term “structured pruning” in large language models (LLMs) refers to the removal of architecturally coherent computational units while retaining the dense tensor structures required for efficient execution on modern accelerators. Briefly, it eliminates whole functional blocks including neuron groups, attention heads, linear subspaces, and even entire transformer layers. Importantly, the elimination approach of entire transformer layers directly cuts inference latency and memory overhead. These operations target the Transformer Block as the primary computational unit (See Figure 2).Subsequently, the structured pruning strategies always organized into three major types according to their target granularity and underlying computational principles:

i) Module- and Head-Level Pruning focuses on the structured removal of fine-grained functional units embedded in transformer submodules, such as MLP neurons, attention heads, or channel groups. It is shown in the literatures that the LLM-Pruner [9] and similar methods utilize dependency-aware grouping to preserve inter-layer tensor alignment based on the model’s graph topology. Thus, by employing gradient-based saliency metrics on minimal calibration data, these approaches facilitate data-efficient pruning. However, by providing high flexibility in compression ratios, this category often yields modest FLOP reductions as the underlying macro-architecture persists.

ii)Layer-Level Structural Simplification treats transformer layers themselves as pruning units. Instead of focusing on internal submodules, these methods can assess the relative contribution of each layer to the model’s forward transformation. For instance, ShortGPT [10] exemplifies this approach by introducing a Block Influence metric that quantifies representational change via cosine similarity between layer inputs and outputs. Layers that contribute minimally to semantic progression can be removed entirely, yielding interpretable pruning decisions and substantial latency reduction. This category exposes the inherent depth redundancy found across large-scale transformer stacks and demonstrates that LLMs often operate far from minimal representational configurations.

iii) Matrix- and Dimension-Level Pruning category explores redundancy directly within the linear transformations, projection matrices, and feature dimensions constituting the computational core of transformers. It is reported that the SlimGPT [15] identifies redundant weight matrix dimensions via second-order sensitivity, however, the SliceGPT [11] or the DiSP-LLM [16] treat feature width as a direct optimization variable. By aligning pruning with the model's algebraic structure, these methods enable architectural reparameterization. Thus they maintain dense matrix operations, ensuring hardware efficiency while achieving global structural simplification.

As illustrated above, these categories changed the point of view of LLM redundancy from local dependencies to global architectural and subspace geometry. This taxonomy redefines structured pruning as a principled redesign grounded in computational invariance, representation sufficiency, and hardware constraints, rather than simple component removal. A visual synthesis of these strategies could be seen in Figure 2.

DOI: 10.54254/2755-2721/2026.AD31512

3.2. Representative methods

The representative methods could divided into the following four parts:

i) LLM-Pruner [9] adopts a dependency-aware approach to module-level pruning. It first identifies neuron groups via connectivity-aware structural discovery and then estimates group importance through first-order loss approximation using small calibration sets. By pruning dependency-preserving groups rather than isolated weights, it avoids misaligned feature maps and supports data-limited scenarios. A lightweight recovery stage using LoRA fine-tuning compensates for performance loss, preserving over 90% accuracy with approximately 20% parameter removal.

ii) ShortGPT [10] transitions pruning to the depth level of transformers. Its Block Influence metric evaluates how much each layer changes the hidden-state representation. Low-influence layers are removed directly, with experiments showing that up to 25% of layers can be discarded with minimal accuracy degradation across LLaMA and Mamba families. The method requires no gradient computation, scales efficiently, and integrates cleanly with quantization or other compression techniques.

iii) SlimGPT [15] extends the Optimal Brain Surgeon framework to structured units such as attention heads or matrix columns. Second-order error estimates are computed via approximate Hessian inverses, realized efficiently through batched Cholesky decomposition. A logarithmically increasing pruning schedule addresses cumulative sensitivity across depths. SlimGPT occupies an intermediate position—more principled than magnitude pruning but more flexible than full-layer removal—achieving improved latency–accuracy trade-offs.

iv) SliceGPT [11] exploits orthogonal invariance around RMSNorm to reduce dimensionality without retraining. PCA-based analysis identifies low-variance feature components, which are pruned while preserving dense computation.

v) DiSP-LLM [16], enables per-layer heterogeneous widths using binary dimension- selection matrices predicted by a hypernetwork. Its selection mechanism incurs no parameter overhead, supports efficient inference, and maintains high zero-shot accuracy even at aggressive pruning ratios.

3.3. Methodological evolution and analysis

The evolution of structured pruning reveals a shift from local dependency modeling toward global architectural restructuring and algebraic reparameterization. Early works such as LLM-Pruner [9] focused on preserving micro- level dependencies and ensuring data-efficient saliency evaluation. These methods offered fine control over pruning granularity but yielded modest acceleration due to limited architectural modification.

The emergence of layer-level approaches (e.g., ShortGPT [10]) marked a conceptual shift: pruning became a process of identifying structurally redundant depth rather than optimizing individual weights. This broadened perspective led to more interpretable pruning and enabled substantial real-world speedups.

Matrix-level and second-order methods such as SlimGPT [15] brought theoretical rigor by combining sensitivity analysis with hardware-efficient structured units. Their focus on linear transformations—the computational bottleneck of transformers—allowed pruning decisions to align more directly with FLOP reduction.

The latest generation introduces dimension-centric frameworks such as SliceGPT [11] and DiSP-LLM [16], which break the long-standing assumption that transformer layers share uniform width.

DOI: 10.54254/2755-2721/2026.AD31512

These models reduce redundancy by reshaping representational subspaces, offering a more fundamental rethinking of transformer architecture under compression constraints.

Collectively, these trajectories suggest that effective structured pruning is not achieved by maximizing sparsity, but by reorganizing model computation around functional invariants and algebraic constraints. This methodological progression points toward increasingly principled, hardware-aligned, and theoretically grounded pruning frameworks for next-generation LLM compression.

4. Low-rank decomposition

Low-rank decomposition has become a central paradigm for reducing the computational and parameter complexity of large language models (LLMs). By expressing high-dimensional weight matrices as products of lower-dimensional factors, these methods exploit the intrinsic spectral redundancy of transformer architectures. Figure 3 provides a structural schematic, showing how the principle of low-rank decomposition is applied to the key weight matrices within a Transformer Block. In contrast to pruning-based approaches, which remove structural components, low- rank methods operate from an algebraic perspective, viewing compression as a projection of learned representations into compact subspaces. Recent developments form two major research lines: (1) activation- and truncation-aware extensions of classical singular value decomposition, represented by the ASVD and SVD-LLM family [12,13,17,18]; and (2) modular and activation-space formulations represented by MoDeGPT and FLAT-LLM [14,19], which adapt decomposition to the structural heterogeneity of transformer modules.

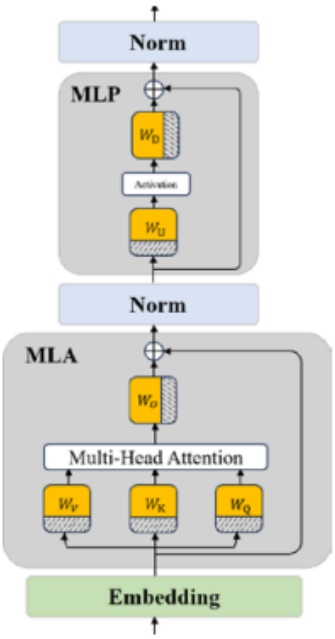

Figure 2. The foundational structure of a Transformer Block, illustrating the key weight matrices ($W_Q, W_K, W_V, W_O, W_U, W_D$) targeted by structured pruning in Large Language Models (LLMs)

4.1. Taxonomy of low-rank methods

Usually, Low-rank LLM compression can be categorized by decomposition targets, activation statistics, and architectural adaptivity.

DOI: 10.54254/2755-2721/2026.AD31512

Firstly, the Activation and Truncation-Aware Decomposition optimizes the factorization of W by incorporating activation covariance S, as demonstrated in ASVD and SVD-LLM. By targeting WS, these methods align truncation with information flow WX to preserve critical spectral components. Enhancements like adaptive rank allocation and residual compensation further address the heterogeneous redundancy across Transformer layers.

Secondly, the Modular Decomposition exploits the distinct functional roles of MLPs and attention mechanisms. Strategies such as MoDeGPT employ Nyström approximations for MLPs and coupled-rank decomposition for key–query projections. This structural alignment ensures that factorization respects cross-module dependencies, moving beyond uniform matrix operations to maintain global model integrity.

Thirdly, the Activation-Space and Subspace-Oriented Compression targets the decomposition of activation covariances instead of static weights. FLAT-LLM exemplifies this by utilizing per-head PCA to isolate intrinsic low-dimensional subspaces within multi-head attention. By reducing representation dimensionality and re-absorbing the basis into model parameters, these methods mitigate cross-head interference and align truncation with empirical covariance. This shift facilitates depth-wise adaptive capacity allocation based on semantic transformations.

As shown above, these strategies represent a transition from classical SVD toward activation-aware, module-aware, and subspace-aware formulations. It is very important that this evolution integrates algebraic decomposition with the specific structural geometry and information flow of Transformer architectures.

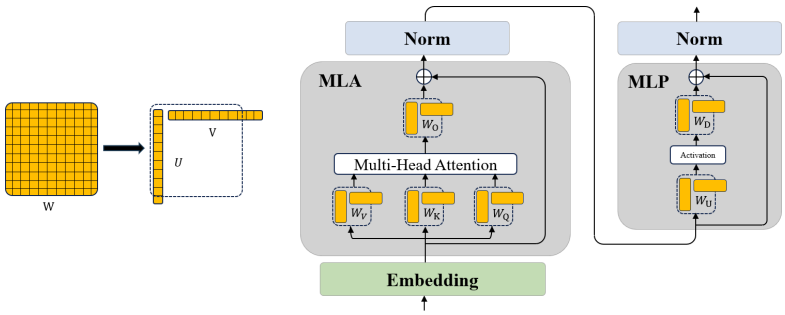

Figure 3. Conceptual diagram of low-rank decomposition application in Transformer architectures. A high-rank weight matrix$W$(left) is approximated by two low-rank factor matrices,$U$and$V$, such that $W \approx UV$

4.2. Representative methods

The development of low-rank decomposition for LLMs is illustrated through several representative methods, each contributing distinct algorithmic insights.

ASVD. Activation-aware SVD [12] introduces whitening-based decomposition that minimizes the reconstruction error of WX rather than W. By applying a diagonal or Cholesky-based whitening matrix S such that SX becomes normalized, SVD is performed on WS, yielding a decomposed form $\mathrm{W' = U_k \Sigma_k V_k^T S^{-1}}$. This activation sensitivity emphasizes high-impact spectral directions. An accompanying sensitivity-based truncation search (STRS) uses perplexity evaluation to automatically determine the optimal rank per layer, enabling adaptive compression without fine-tuning.

SVD-LLM. SVD-LLM [13] formalizes whitening through a theoretically grounded Cholesky factor of the activation covariance XX^T, ensuring that truncating a singular value σ_i incurs an exact reconstruction loss of σ_i. Its two- stage fine-tuning procedure updates low-rank factors

DOI: 10.54254/2755-2721/2026.AD31512

sequentially under LoRA-style reparameterization, avoiding interference between U_k and V_k^T. SVD-LLM v2 [17] enhances flexibility through heterogeneous rank allocation across layers and replaces the whitening step with a dual-SVD formulation to avoid numerical instability when XX^T is not positive-definite.

ResSVD. ResSVD [18] addresses the accumulation of truncation errors across depth by applying a second SVD to the residual matrix $R = W - W_{r1}$. The final approximation $W_r = W_{r1} + R_{r2}$ preserves residual spectral information ignored by standard SVD. Additionally, a partial-layer compression strategy restricts decomposition to later layers, redistributing the compression budget to mitigate error propagation in early transformations.

MoDeGPT. MoDeGPT [14] introduces a modular decomposition framework based on categorizing transformer components into three types: Type-I (MLP), Type-II (key–query), and Type-III (value–output). Nyström approximation is applied to Type-I layers to preserve activation diversity through ridge leverage scores. Type-II layers employ coupled-rank decomposition to align the subspaces of key and query projections, maintaining attention consistency. Classical SVD is used for Type-III layers. A global allocation mechanism based on entropic regularization distributes compression budgets according to influence scores, preventing over-compression of sensitive layers.

FLAT-LLM. FLAT-LLM [19] performs per-head PCA to extract intrinsic activation subspaces within multi-head attention. After truncation, the reduced bases are absorbed into the value and output projections, maintaining parameter efficiency. The importance-preserving rank selection (IPRS) algorithm assigns ranks adaptively according to input–output activation similarity, ensuring that deeper layers receive more capacity when their semantic transformation intensity is higher. The method requires no fine-tuning and is compatible with grouped-query attention architectures.

4.3. Mathematical and empirical analysis

The methodological evolution of low-rank approaches reflects a progression from classical spectral approximation toward statistically informed, modular, and activation-adaptive formulations.

Spectral Foundations and Activation Geometry. Classical SVD assumes that the principal singular directions of W correspond to high-impact subspaces. Activation-aware methods refine this assumption by instead decomposing WS, where S whitens or rescales activations such that $S^{-1}X$ becomes isotropic. This transformation better aligns singular directions with downstream information flow and ensures that truncation reflects true representational importance. The analytical insight that truncation loss equals discarded singular values under whitening [13] provides theoretical grounding for optimal rank selection.

Error Propagation in Deep Transformers. It is well-known that the problem with uniform low-rank truncation lies in its tendency to trigger cumulative error propagation, hitting the early Transformer blocks the hardest. Thus, ResSVD and SVD-LLM v2 [17,18] mitigate this by pivoting to heterogeneous rank allocation and residual-based fixes. Normally, this method aligns with the observation that representational demands vary by depth: i) early layers require high-capacity feature transformation; ii) however, deeper layers can be compressed more aggressively as they operate on lower-dimensional manifolds.

Modular Redundancy and Functional Constraints. These approaches recognize that attention and MLP modules exhibit distinct redundancy structures governed by their nonlinearities. For instance, the coupled structure of key–query projections imposes constraints on rank truncation, motivating CR decomposition [14]. So that such module-aware decomposition preserves cross-head attention consistency more effectively than uniform SVD.

DOI: 10.54254/2755-2721/2026.AD31512

Subspace-Based Adaptivity. It is always exemplified by FLAT-LLM [19]—leverages per-head PCA to uncover latent low-dimensional structures within activation spaces. Different form weight-centric SVD, this activation-aware strategy mitigates cross-head interference and maintains semantic consistency more effectively.

Overall, the combination of spectral theory, activation statistics, and modular architectural insights has reshaped low-rank decomposition from uniform matrix factorization into a family of structurally aligned, theoretically informed, and empirically robust compression frameworks. The shift from global SVD to activation- and module-aware techniques reflects a broader movement toward decompositions that respect the hierarchical and functional geometry of large-scale transformers.

5. Evaluations

Indeed, a consistent evaluation protocol is called-for to compare pruning-based and decomposition-based compression methods for large language models. Although individual works often differ in model size, calibration data, and downstream tasks, their overall characteristics can be qualitatively aligned through dimensions such as structural granularity, reliance on data, retraining requirements, approximation fidelity, hardware friendliness, and expected robustness under distribution shifts. In order to facilitate systematic comparison, Table 1 summarizes the qualitative advantages and limitations of representative pruning and low-rank methods.

To our best knowledge, the landscape of model pruning is defined by a tension between architectural interpretability and computational efficiency. That is to say, LLM-Pruner represents a class of gradient-based structural pruning that achieves fine-grained sparsity, but remains bottlenecked by gradient dependencies. Meanwhile, the investigations into layer-wise redundancy (e.g., ShortGPT and SlimGPT) demonstrate the feasibility of aggressive depth reduction, albeit at the risk of diminished stability under diverse data distributions. To overcome these drawbacks that metioned above, researchers developed the framework SliceGPT and DISP-LLM that focused on re-engineering internal dimensionality.

Table 1. Qualitative comparison of representative pruning- and decomposition-based LLM compression methods

Method	Granularity	Data Req.	Retrain	Fidelity	HW Friendly	Robust.
LLM-Pruner	Structural (token / head / block)	Medium (grad.)	Opt.	Medium	High	Medium
ShortGPT / SlimGPT	Layer-level	Low	No	Medium–Low	High	Low
SliceGPT	Token / channel slicing	Low	No	Medium	High	Medium
DISP-LLM	Embedding redistribution	Low	No	Medium	High	Med–High
ASVD/ SVD-LLM	Matrix low-rank	Low (activ.)	Opt./light	High	Medium	Med–High
ResSVD	Low-rank + residuals	Low	No	High	Medium	High
MoDeGPT	Module-level decomposition	Very low	No	High	Medium	Med–High
FLAT-LLM	Activation-space per head	Very low	No	High	Medium	High

DOI: 10.54254/2755-2721/2026.AD31512

In contrast, decomposition-based techniques leverage continuous approximations to maintain smoother optimization landscapes and often require no retraining. Activation-aware low-rank methods such as ASVD, SVD-LLM, and ResSVD reduce truncation errors by aligning decomposition with activation statistics or by compensating residual components. Modular approaches such as MoDeGPT decompose transformers by functional subsystems, while activation-space methods like FLAT-LLM identify low-dimensional subspaces directly from attention head activations. These methods deliver strong accuracy retention at high compression ratios with minimal calibration cost; however, their behavior under large-scale deployment or domain-shifted evaluation remains an open question.

As discussed above, pruning based approaches excel in architectural flexibility and align well with specialized hardware. And decomposition based alternatives typically offer superior weight fidelity with significantly lower training overhead. Therefore, it is suggested that there is no solution that could fit for all the problems exists, however, the choice of compression framework must be dictated by the specific bottlenecks of the target deployment environment.

6. Conclusion

In conclusion, structured pruning and low-rank decomposition represent two major directions in large language model (LLM) compression, each with distinct strengths. Structured pruning methods such as LLM-Pruner, SlimGPT, and DISP- LLM preserve architectural regularity and are well-suited for deployment where hardware efficiency is crucial. Among them, DISP-LLM offers the best adaptability through dimension-independent pruning. In contrast, decomposition-based methods like SVD-LLM, MoDeGPT, and FLAT-LLM achieve high compression without retraining, providing fast, training-free deployment with strong accuracy retention. Indeed, future optimization lies in the hybrid integration of pruning and decomposition to maximize interpretability. Indeed, this review article estabilised s unified benchmarks to ensure cross-method reproducibility. Therefore the key contributions include the synthesis of hardware-aligned, training-free compression strategies and the identification of performance-efficiency Pareto frontiers. Such insights catalyze the deployment of scalable LLMs while maintaining architectural integrity and minimizing computational overhead.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, et al. Gpt-4 technical report. arXiv preprint arXiv: 2303.08774, 2023.

[2] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, et al. Palm 2 technical report. arXiv preprint arXiv: 2305.10403, 2023.

[3] Meta GenAI. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv: 2307.09288, 2023.

[4] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. Advances in neural information processing systems, 35: 27730–27744, 2022.

[5] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. Advances in neural information processing systems, 36: 53728–53741, 2023.

[6] Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. Model compression and efficient inference for large language models: A survey. arXiv preprint arXiv: 2402.09748, 2024.

[7] Yehui Tang, Yunhe Wang, Jianyuan Guo, Zhijun Tu, Kai Han, Hailin Hu, and Dacheng Tao. A survey on transformer compression. arXiv preprint arXiv: 2402.05964, 2024.

[8] Seungcheol Park, Jaehyeon Choi, Sojin Lee, and U Kang. A comprehensive survey of compression algorithms for language models. arXiv preprint arXiv: 2401.15347, 2024.

DOI: 10.54254/2755-2721/2026.AD31512

[9] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. Advances in neural information processing systems, 36: 21702–21720, 2023.
[10] Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In Findings of the Association for Computational Linguistics: ACL 2025, pages 20192–20204, 2025.
[11] Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. arXiv preprint arXiv: 2401.15024, 2024.
[12] Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models. arXiv preprint arXiv: 2312.05821, 2023.
[13] Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. arXiv preprint arXiv: 2403.07378, 2024.
[14] Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Modegpt: Modular decomposition for large language model compression. arXiv preprint arXiv: 2408.09632, 2024.
[15] Gui Ling, Ziyang Wang, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. Advances in Neural Information Processing Systems, 37: 107112–107137, 2024.
[16] Shangqian Gao, Chi-Heng Lin, Ting Hua, Zheng Tang, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Disp-llm: Dimension-independent structural pruning for large language models. Advances in Neural Information Processing Systems, 37: 72219–72244, 2024.
[17] Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. Svd-llm v2: Optimizing singular value truncation for large language model compression. arXiv preprint arXiv: 2503.12340, 2025.
[18] Haolei Bai, Siyong Jian, Tuo Liang, Yu Yin, and Huan Wang. Ressvd: Residual compensated svd for large language model compression. arXiv preprint arXiv: 2505.20112, 2025.
[19] Jiayi Tian, Ryan Solgi, Jinming Lu, Yifan Yang, Hai Li, and Zheng Zhang. Flat-llm: Fine-grained low-rank activation space transformation for large language model compression. arXiv preprint arXiv: 2505.23966, 2025.