A Survey on Computation-Reduction Techniques for LLM Serving

Yavuz Ferhatosmanoglu
yff23@cam.ac.uk

1 Introduction

Building on our previous survey on large language model (LLM) inference and state-of-the-art serving systems, this survey focuses specifically on recent computation-reduction techniques. We examine four representative categories: (i) cross-request caching, (ii) early-exit mechanisms, (iii) adaptive layer skipping, and (iv) vector search–based KV retrieval. Together, these techniques inform and motivate future directions on nearest-neighbour search-based, systems-aware layer skipping strategies for efficient LLM inference.

2 Cross-request Caching

LLM serving optimisations for cross-request caching have evolved from black-box techniques towards internal, systems-aware architectures. Early solutions such as GPTCache [1] operate at the application level by performing semantic caching over model prompts and returning previously generated outputs. While effective for simple question-answering tasks, this approach is limited in its ability to exploit partial overlaps across requests.

In contrast, Prompt Cache [9] precomputes attention states for prompts and performs caching at the sub-prompt level. Prompts are segmented into reusable components defined by a *schema*, enabling cross-request reuse. However, when prompt modules are combined (e.g., multiple documents included in context), cross-attention between modules is ignored, resulting in an approximation of the full computation. LMCache [6, 21] addresses this limitation, in addition to the constraints of prefix caching, by enabling the reuse of non-prefix text chunks. It employs a fusion mechanism that computes cross-attention for only a subset of tokens within each reused KV cache. This preserves accuracy due to the observation of *sparse attention*, where only a small fraction of tokens significantly influence the final attention output. LMCache identifies these critical tokens using early-layer attention computations.

More recently, RAGCache [13] has emerged to accelerate retrieval-augmented generation by overlapping document retrieval with LLM inference. It caches KV vectors of retrieved documents across requests, organising cached states using a knowledge tree and prioritising nodes based on access frequency, size and cost of recomputing prefixes. Speculative candidate documents from CPU-based vector search are used to initiate GPU-based LLM inference, hiding retrieval latency. The primary performance gains stem from the observation that RAG workloads exhibit highly skewed retrieval patterns.

3 Systems-aware Early Exit LLMs

The previously-discussed cross-request caching approaches can be considered as avoiding full model computation to reuse previously computed attention states. In contrast, early exit (EE) LLMs enable requests to leave the inference pipeline at intermediate Transformer layers, allowing the skipping of the remaining layers of the model. These models typically incorporate a binary classifier at designed layers (*checkpoints*) to determine whether the current hidden token state is sufficient to generate the output token. Many approaches (EE-LLM [4], Apparate [7], Miao et al. [19]) show how EE can provide significant throughput gains, albeit in non-batched inference settings, with diminishing returns as batch sizes increase.

Two main challenges in productionising EE systems are [17]: (1) synchronising batches when some tokens exit early while others do not, and (2) handling missing KV cache vectors for early-exited tokens, which may be needed in later iterations. For challenge 1, batch approaches include: *consensus* (EE if all tokens exit), *greedy* (EE if at least one exits), and *majority* (EE if most tokens exit). For challenge 2, *KV-recomputation* reintroduces EE tokens in the next pass, expanding the batch and computing necessary entries, while *state-copying* replicates the EE tokens' KV vectors across all layers, although increasing memory usage and redundancy in the cache.

DREX [17] addresses these bottlenecks by introducing dynamic rebatching, which allows requests to follow independent execution paths. Instead of forcing batch-wide decisions, requests that exit early are processed immediately, while remaining requests are regrouped in a logical buffer and served when the regrouped batch is sufficiently large. This is further optimised through *SLA-aware forced flushing*, which avoids holding requests if their SLA would not be met, and an *adaptive rebatching threshold*, which avoids EE if the rebatching operation itself is predicted to be more expensive than the gains from early exiting. Additionally, DREX resolves memory concerns by utilising virtual memory mappings to share physical memory blocks across skipped layers, eliminating physical duplication and reducing CUDA memory usage by up to 18.3%.

By treating EE decisions as a dynamic scheduling problem rather than a static batching constraint, DREX provides a template for integrating vertical model-depth optimisations with horizontal system efficiency. The convergence of zero-copy rebatching and predictive thresholding enables EE LLMs to reach production-level throughput without sacrificing output quality through involuntary exits. As model architectures continue to scale in depth, such systems-aware mechanisms will be essential for transforming theoretical compute savings into practical, scalable inference performance.

4 Adaptive Layer Skipping

Unlike EE approaches, which skip only a suffix of model layers, layer skipping approaches operate at a finer granularity, allowing the execution of non-contiguous layers. Adaptive skipping approaches determine these layers dynamically based on the input. HadSkip [20], a training-based approach, enables adaptive layer skipping under a predefined inference budget by introducing a learnable binary gate before each Transformer layer to decide whether the layer should be skipped. This allows the model to dynamically select different subsets of layers to execute for different input tokens. To train these discrete gates, HadSkip employs a three-stage optimisation strategy combined with a homotopic training scheme. Training starts from a relaxed budget that allows most layers to be executed and gradually tightens the budget, preventing optimisation collapse

when many layers must be skipped. ReinMax [16] is used to address the non-differentiability of binary gating. While empirical results show that HadSkip achieves a superior accuracy–efficiency trade-off compared to early-exit baselines under tight compute budgets, its reliance on fine-tuning and per-layer gating introduces additional training complexity and poses challenges for integration with decoder-only LLM serving systems and KV-cache–centric optimisations.

Similarly, FlexiDepth [18] proposes a dynamic layer skipping approach by augmenting each layer with a lightweight router which determines whether a layer should be skipped. Tokens that are skipped are processed by aligning their representations with fully processed tokens, preventing representation drift. Unlike HadSkip, FlexiDepth freezes the original Transformer layers, and only trains the router and adapter, with a layer-skipping loss jointly optimised with the language modelling loss to penalise excessive layer usage. FlexiDepth reveals that simpler or repetitive tokens (e.g., copying, predictable phrases) require fewer layers, while tokens involving reasoning, abstraction, or uncertainty activate deeper computation. Despite substantial FLOP reductions, FlexiDepth does not translate into wall-clock speedups due to control-flow divergence and GPU inefficiencies arising from heterogeneous execution paths within a batch.

Reinforcement learning can also enable adaptive layer skipping, as demonstrated in ASTER [15], which models transformer inference as a sequential Markov Decision Process guided by a specialised "cognitive token". At each decision point, a scoring module evaluates this token's representation to predict the optimal subsequent layer. When the policy selects non-consecutive layers, a lightweight adapter module transitions the hidden states between different representation spaces. To preserve accuracy under large skipping, ASTER uses a dual-matching knowledge distillation strategy that mimics a full-layer teacher model's internal attention distributions. The policy is refined by a Time-aware Importance-weighted Dynamic Reward (TIDR) mechanism, which balances computational gains against task performance by rewarding larger skips while penalising semantic drift.

In contrast to the previous approaches, AdaSkip [10] proposes a *training-free* adaptive layer skipping strategy, focusing on long-context workloads, where the prefill phase dominates inference cost. Rather than skipping entire Transformer layers, it independently considers attention and feedforward (FFN) sublayers. It employs *offline importance learning*, estimating sublayer importance through the average cosine similarity between input and output vectors, and skips the least important sublayers (i.e., those with high IO similarity) according to a user-defined *acceleration ratio*. This enables substantial speed-up of the prefill phase. AdaSkip further incorporates *online importance learning*, using IO similarity observed during decoding to identify additional FFN skipping opportunities, thereby improving efficiency in the decode phase as well. Beyond layer skipping, similar training-free methods for expert skipping have been explored in Mixture-of-Experts models [11, 12], highlighting the broader potential of adaptive skipping strategies in reducing inference costs.

5 ANNS-based KV Retrieval and Computation

Approximate nearest-neighbour search (ANNS) has emerged as a powerful technique for managing large KV caches required for long-context inference. While traditional approaches rely on static eviction or heuristic pruning, recent methods such as MagicPiG [5], Unlimiformer [2], and PQCache [22] utilise ANNS to dynamically retrieve relevant tokens. However, dynamic layer- and expert-skipping strategies utilising vector search remain an underexplored area. RetrievalAttention [14] represents a state-of-the-art implementation that highlights both the challenges and systems-level opportunities of integrating ANNS into the Transformer architecture.

RetrievalAttention [14] is a training-free approach that exploits the inherent dynamic sparsity of attention by offloading the majority of KV vectors to CPU memory and retrieving "critical" tokens through an attention-aware vector search. It specifically addresses the out-of-distribution (OOD) challenge, where query and key vectors inhabit different distribution spaces due to distinct projection weights, meaning ANNS indexes cannot be used directly. To address this, RetrievalAttention establishes a mapping from query vectors to their exact k-nearest neighbour key vectors during index construction. At search time, the decoding query vector is matched against nearby query vectors, which then retrieve key vectors via this precomputed mapping. To remove the need to store intermediate query vectors, RoarGraph [3], a cross-modal ANNS index, is used to directly link key vectors that share common queries.

The system employs a hybrid CPU–GPU execution strategy: predictable tokens (e.g., initial tokens, recent sliding windows) are retained on the GPU, while the remaining tokens are dynamically retrieved from the CPU. To minimise PCIe bottlenecking, partial attention results from both devices are computed in parallel and merged using a rescaling method inspired by FlashAttention [8]. Evaluations demonstrate that this approach recovers near-full attention accuracy while scanning only 1–3% of the total cache data, making it the first system capable of serving an 8B model with 128K context on a single 24GB commodity GPU with acceptable latency.

While RetrievalAttention and similar methods demonstrate the potential of ANNS in optimising memory usage and retrieval efficiency, the integration of ANNS with dynamic layer skipping and EE strategies remains an underexplored area. Exploring these approaches could further enhance inference efficiency, enabling more adaptive and resource-efficient models.

6 Future Directions

Several promising directions lie at the intersection of ANNS-based semantic caching, layer skipping and systems-level optimisations for scaling and improving the efficiency of these strategies in production environments.

Retrieval-informed EE and layer skipping. A key gap in current research lies in integrating ANNS-based semantic caching methods with EE and adaptive layer skipping. Rather than processing each request independently, systems could maintain per-layer vector indices of intermediate hidden states. If the hidden representation of a new request at an early layer closely matches a cached state, the system could reuse the corresponding output or skip multiple subsequent Transformer layers, significantly reducing redundant computation across requests.

Systems-aware layer skipping beyond EE. Existing adaptive skipping techniques focus on model-level decisions without fully accounting for systems-level metrics such as end-to-end latency, batching efficiency, and memory overhead. While DREX [17] introduces systems-aware optimisations for EE LLMs, extending these ideas to general adaptive layer skipping remains an open challenge. For example, adaptive rebatching thresholds could be used to analytically determine whether the overhead of skipping is outweighed by the cost of executing a subset of layers.

Productionising ANNS-based layer skipping. Building on techniques in DREX [17], integrating ANNS-driven semantic caching into EE and adaptive layer skipping frameworks opens new opportunities for production-level optimisations. Vector search can enable more precise token-level execution decisions, allowing for the reuse of previously computed hidden states and reducing redundant computation. Combined with techniques such as copy-free rebatching buffers and virtual memory mappings, this approach can better manage the fragmented attention and execution states that arise when tokens in a batch follow heterogeneous paths.

7 Conclusion

This survey reviewed recent computation-reduction techniques for LLM inference, focusing on cross-request caching, EE approaches, adaptive layer skipping, and ANNS-based KV cache retrieval. While each approach offers efficiency gains, our analysis highlights the potential of combining retrieval-informed strategies with systems-aware optimisations. In particular, integrating per-layer semantic caching with adaptive skipping and early-exit techniques appears to be a promising direction. Future research in this space can combine model-level innovations with production-level optimisations, enabling efficient and scalable LLM serving.

References

[1] Fu Bang. 2023. GPTCache: An Open-Source Semantic Cache for LLM Applications Enabling Faster Answers and Cost Savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, Liling Tan, Dmitrijs Milajevs, Geeticka Chauhan, Jeremy Gwinnup, and Elijah Rippeth (Eds.). Association for Computational Linguistics, Singapore, 212–218. doi:10.18653/v1/2023.nlposs-1.24

[2] Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. 2023. Unlimiformer: Long-Range Transformers with Unlimited Length Input. arXiv:2305.01625 [cs.CL] https://arxiv.org/abs/2305.01625

[3] Meng Chen, Kai Zhang, Zhenying He, Yinan Jing, and X. Sean Wang. 2024. RoarGraph: A Projected Bipartite Graph for Efficient Cross-Modal Approximate Nearest Neighbor Search. *Proceedings of the VLDB Endowment* 17, 11 (July 2024), 2735–2749. doi:10.14778/3681954.3681959

[4] Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. EE-LLM: large-scale training and inference of early-exit large language models with 3D parallelism. In *Proceedings of the 41st International Conference on Machine Learning* (Vienna, Austria) *(ICML'24)*. JMLR.org, Article 277, 27 pages.

[5] Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. 2024. MagicPIG: LSH Sampling for Efficient LLM Generation. arXiv:2410.16179 [cs.CL] https://arxiv.org/abs/2410.16179

[6] Yihua Cheng, Yuhan Liu, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Kuntai Du, and Junchen Jiang. 2025. LMCache: An Efficient KV Cache Layer for Enterprise-Scale LLM Inference. *arXiv preprint arXiv:2510.09665* (2025).

[7] Yinwei Dai, Rui Pan, Anand Iyer, Kai Li, and Ravi Netravali. 2024. Apparate: Rethinking Early Exits to Tame Latency-Throughput Tensions in ML Serving. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles* (Austin, TX, USA) *(SOSP '24)*. Association for Computing Machinery, New York, NY, USA, 607–623. doi:10.1145/3694715.3695963

[8] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. arXiv:2205.14135 [cs.LG] https://arxiv.org/abs/2205.14135

[9] In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. 2024. Prompt Cache: Modular Attention Reuse for Low-Latency Inference. doi:10.48550/arXiv.2311.04934 arXiv:2311.04934 [cs].

[10] Zhuomin He, Yizhen Yao, Pengfei Zuo, Bin Gao, Qinya Li, Zhenzhe Zheng, and Fan Wu. 2025. AdaSkip: adaptive sublayer skipping for accelerating long-context LLM inference. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence (AAAI'25/IAAI'25/EAAI'25, Vol. 39)*. AAAI Press, 24050–24058. doi:10.1609/aaai.v39i22.34579

[11] Wei Huang, Yue Liao, Jianhui Liu, Ruifei He, Haoru Tan, Shiming Zhang, Hongsheng Li, Si Liu, and Xiaojuan Qi. 2025. Mixture Compressor for Mixture-of-Experts LLMs Gains More. arXiv:2410.06270 [cs.LG] https://arxiv.org/abs/2410.06270

[12] Yushi Huang, Zining Wang, Zhihang Yuan, Yifu Ding, Ruihao Gong, Jinyang Guo, Xianglong Liu, and Jun Zhang. 2025. MoDES: Accelerating Mixture-of-Experts Multimodal Large Language Models via Dynamic Expert Skipping. doi:10.48550/arXiv.2511.15690 arXiv:2511.15690 [cs].

[13] Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Shufan Liu, Xuanzhe Liu, and Xin Jin. 2025. RAGCache: Efficient Knowledge Caching for Retrieval-Augmented Generation. *ACM Trans. Comput. Syst.* 44, 1 (Nov. 2025), 2:1–2:27. doi:10.1145/3768628

[14] Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. 2024. RetrievalAttention: Accelerating Long-Context LLM Inference via Vector Retrieval. doi:10.48550/arXiv.2409.10516 arXiv:2409.10516 [cs].

[15] Fangxin Liu, Junjie Wang, Ning Yang, Zongwu Wang, Junping Zhao, Li Jiang, and Haibing Guan. 2025. ASTER: Adaptive Dynamic Layer-Skipping for Efficient Transformer Inference via Markov Decision Process. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25)*. Association for Computing Machinery, New York, NY, USA, 11853–11861. doi:10.1145/3746027.3755526

[16] Liyuan Liu, Chengyu Dong, Xiaodong Liu, Bin Yu, and Jianfeng Gao. 2023. Bridging Discrete and Backpropagation: Straight-Through and Beyond. arXiv:2304.08612 [cs.LG] https://arxiv.org/abs/2304.08612

[17] Xuting Liu, Daniel Alexander, Siva Kesava Reddy Kakarla, Behnaz Arzani, and Vincent Liu. 2025. Dynamic Rebatching for Efficient Early-Exit Inference with DREX. doi:10.48550/arXiv.2512.15705 arXiv:2512.15705 [cs].

[18] Xuan Luo, Weizhi Wang, and Xifeng Yan. 2025. Adaptive Layer-skipping in Pre-trained LLMs. doi:10.48550/arXiv.2503.23798 arXiv:2503.23798 [cs].

[19] Ruijie Miao, Yihan Yan, Xinshuo Yao, and Tong Yang. 2024. An Efficient Inference Framework for Early-exit Large Language Models. arXiv:2407.20272 [cs.CL] https://arxiv.org/abs/2407.20272

[20] Haoyu Wang, Yaqing Wang, Tianci Liu, Tuo Zhao, and Jing Gao. 2023. HadSkip: Homotopic and Adaptive Layer Skipping of Pre-trained Language Models for Efficient Inference. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 4283–4294. doi:10.18653/v1/2023.findings-emnlp.283

[21] Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. 2025. CacheBlend: Fast Large Language Model Serving for RAG with Cached Knowledge Fusion. doi:10.48550/arXiv.2405.16444 arXiv:2405.16444 [cs].

[22] Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. 2025. PQCache: Product Quantization-based KVCache for Long Context LLM Inference. arXiv:2407.12820 [cs.CL] https://arxiv.org/abs/2407.12820