Learnable Adaptive KV-cache Compression

Anonymous Author(s)
Affiliation
Address
email

Abstract

Efficient inference within Large Language Models (LLMs) commonly assumes the usage of a key-value (KV) cache. However, while it removes the quadratic bottleneck of vanilla attention, it trades it for a proportional—and often prohibitive—memory footprint that scales linearly with the sequence length. Modern approaches to reducing the KV cache memory either use token eviction or deterministic dimensionality reduction methods applied with a uniform budget to each layer. With this inflexibility in mind, we introduce a *learnable adaptive compression* method that dynamically retrofits existing KV cache for each layer with a trainable compression budget and encoding and decoding components. Experiments on LLAMA-3.1-8B across various benchmarks show that our method allows maintaining original model performance within 1% during $\times 2$ and $\times 3$ KV cache compression and $1\% - 2\%$ for $\times 4$ reduction. Our experiments also show that this trainable adaptive budgeting allows the model to devote more capacity to late layers, where semantic abstractions are denser, which offers layer-wise interpretability of attention sparsity, opening the door to principled analysis and hardware-aware scheduling during inference.

1 Introduction

Large Language Models (LLMs) exhibit state-of-the-art performance on a variety of language tasks [32, 3] ranging from generalistic question answering [7, 31, 38] to complex multi-hop reasoning in math [39, 16], coding [18] and STEM [36]. Such models usually impose substantial memory requirements to serve, one of which stems from the quadratic attention memory computation [20], imposed by the transformer architecture [44]. To circumvent this limitation, the *key* and *value* matrices comprising the attention can be saved in memory for each layer, which is known as *KV caching*. Although KV cache allows attention computation to become linear, it also introduces a memory footprint which scales linearly with the length of the input sequence. It is important to note that with the introduction of complex reasoning [15, 19] and Agentic [29, 42] LLM paradigms, such as Chain-Of-Thought (CoT) [48], Tool Integrated Reasoning [37, 40, 2, 35] and others [51, 50], along with the emergence of Large Reasoning Models (LRMs) [17, 13], the input and output sequence lengths within the model have grown substantially [11]. As the linearly scaling memory directly prohibits the generation of longer sequences, it becomes increasingly essential to find methods of KV cache compression that also preserve model performance.

Recent research has proposed several families of methods for reducing the memory of the KV cache. First, decreasing that memory can be achieved through *token-eviction* [55, 12, 9, 56], where the model removes key-value pairs of some tokens from the cache using heuristics or algorithms on the assumption that there exists a subset of tokens that are sufficient for solving the task. Instead of directly evicting, it is also possible to merge the cache components corresponding to several tokens [47, 54, 45, 34]. However, such methods can lead to hallucinations in tasks involving complex reasoning in a domain or during long context generation [54], as LLMs are sensitive to the criterion

Submitted to NeurIPS 2025 Workshop on Efficient Reasoning. Do not distribute.

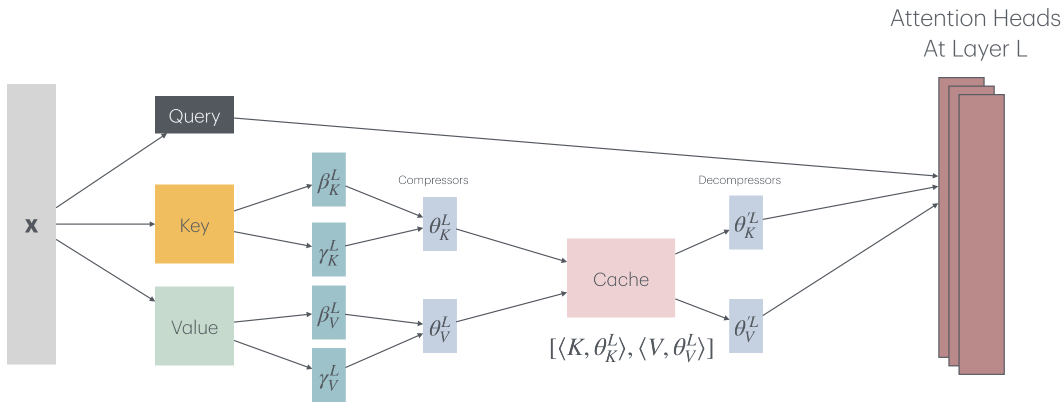

Figure 1: The figure depicts the unrolled compression process for any layer using our method. K and V represent the key-value matrices in the layers cache, while compression and decompression are completed using θ^L functions, which can be both linear projectors and parametrised neural networks. The budget of the compression per layer is learned using the β_V^L, β_K^L, while γ_L is a learnable binary mask representing which β_V^L, β_K^L columns to choose in the layer.

of token merging or eviction. Another set of compression techniques proposes novel architectural [43, 52] changes, such as Group-Query Attention [1] or Multi-head Latent Attention [5]. While these methods are effective, they require substantial resources and should be explicitly trained during pre- and/or post-training. Lastly, it is possible to reduce the dimensionality of the KV cache by applying low-rank decompositions [41, 53] or projections into constrained subspaces [22]. Both of these method families still enforce uniform per-layer budgets for compression. Towards this end, we propose a novel trainable method for KV cache compression that learns a compression budget and key-value encoding and decoding modules for each layer. Crucially, our framework allows to adaptively learn *where* and *how much* to compress. A depiction of our method per layer can be seen in fig. 1. Our method does not require (pre)training parallel to the LLM. The parameters can be efficiently learned by minimising an ℓ_2-like reconstruction loss on the sampled KV caches from several hundred articles of *wikitext-2* [30], meaning the method does not induce significant computational costs during calibration. Our results show that the proposed method allows for compressing the KV cache from $\times 2$ to $\times 4$ while maintaining the original model performance within $[1\%, 4\%]$. We further explore the learned per-layer budgets and observe that there is a strong positive nonlinear correlation between the depth of the layer and how much it can be compressed, which further shows the need for non-uniform per-layer compression. Our contributions are the following: (i) we propose a novel adaptive compression method that allows learning a per-layer compression budget, (ii) we show that this compression can reduce the KV cache memory from $\times 2$ to $\times 4$ and maintain the original performance within $[1\%, 2\%]$ while not requiring massive resources to train. (iii) Our ablations further show a strong non-linear trend that compression budgets converge to after training, showing the exact dynamic of allowed compression per layer.

2 Related Works

Formally, post-training KV cache compression methods can be divided into techniques that use token eviction, merging, and low-rank decompositions. An orthogonal line of research that can be used alongside these techniques is quantisation [26, 6, 23], which allows for further compression of the cache by reducing its biwise precision, incurring minor or no expense to overall model performance.

Token-eviction and merging Many heuristics have been used to find tokens closely correlated with attention without explicitly computing it [12, 25, 10]. Notably, even simply scoring tokens w.r.t. their L2 norms [8] is a reliable method for ranking tokens in terms of their importance. It is also possible, albeit more costly, to use the attention scores directly to retain relevant tokens [55].

Merging allows for combining KV cache entries instead of outright removing them [34, 54, 45]. These methods, however, incur attention inconsistency before and after merging, thus losing information within the sentence. It is also worth noting that both token eviction and merging methods permanently lose information, thereby degrading the overall LLM generation.

Low Rank Decompositions Most attempts at low-rank decomposition for KV cache involve applying tensor decomposition methods such as Singular Value Decomposition (SVD) [24] on the pre-trained weights of the model [21, 46, 53], sometimes followed by calibration of that LLM [33, 49]. Our work is also associated with Multihead Latent Attention [27, MLA], as we similarly attempt to compress the KV cache, yet do not use as much compute and data for optimisation. All of these methods also suffer from the fact that the compression rate within each layer is uniform, meaning that the dimensionality of the latent space is chosen statically or with minor adaptive heuristics. We address this by reframing the KV cache compression as a learnable allocation of per-layer compression budgets, subject to a total compression budget constraint.

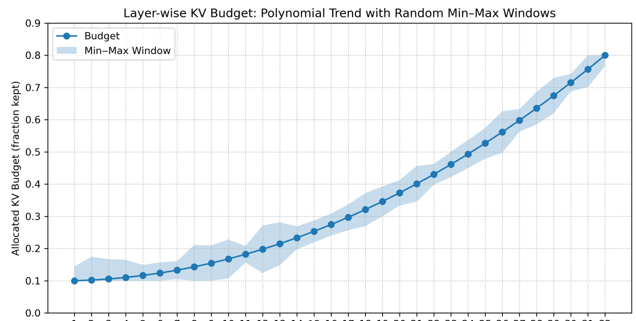

Figure 2: Compression Budget dynamics averaged across three trainings.

3 Methodolgy

4 Background: KV-Cache in Decoding

During autoregressive decoding in Transformer-based language models, the key–value (KV) cache is essential to avoid redundant computation. Let a Transformer have L layers, each with H attention heads and head dimension d. In step t, for each new token, the model produces query, key, and value projections: $Q_t^{(\ell)} = W_\ell^Q x_t, \quad K_t^{(\ell)} = W_\ell^K x_t, \quad V_t^{(\ell)} = W_\ell^V x_t$, for layer $\ell = 1, \ldots, L$. To compute self-attention at layer ℓ, one needs to attend over all past keys and values $\mathrm{Attention}(Q_t^{(\ell)},\ [K_1^{(\ell)}, \ldots, K_t^{(\ell)}],\ [V_1^{(\ell)}, \ldots, V_t^{(\ell)}])$ .. To avoid re-computing K, V for all past tokens at every step, the model caches these as $\mathbf{K}^{(\ell)} = [K_1^{(\ell)}, K_2^{(\ell)}, \ldots, K_{t-1}^{(\ell)}], \quad \mathbf{V}^{(\ell)} = [V_1^{(\ell)}, V_2^{(\ell)}, \ldots, V_{t-1}^{(\ell)}]$. Then at step t, only the new K_t, V_t are appended, and attention is computed efficiently over the cached sequence.

4.1 Adaptive Budget Learning with Global Constraint

To mitigate memory and bandwidth bottlenecks, we introduce an *adaptive KV-cache compressor* that learns a *layer-wise projection budget* under a global compression constraint. Unlike fixed uniform compression, our method allows each layer to allocate its own number of projection dimensions while ensuring that the *average* compression rate equals a pre-specified target ρ.

Linear Projections We construct a linear orthonormal cacher that compresses with an orthonormal projector and decodes with a linear pseudo-inverse. Per layer ℓ, a full orthonormal basis $W_\ell \in \mathbb{R}^{d \times d}$ (built via QR, subsampled HADAMARD, or random JL) is available, and decoding uses a numerically stable QR-based pseudo-inverse W_ℓ^+.

Layer-wise budget parameters. For each layer $\ell \in \{1, \ldots, L\}$ we maintain a learnable scalar $b_\ell \in \mathbb{R}$ and column scores $R_\ell \in \mathbb{R}^d$. The raw scalars $(b_1, \ldots, b_L)$ are transformed by a softmax and rescaled to enforce a global expected budget:

$$\tilde{b}_\ell = \frac{\exp(b_\ell)}{\sum_{j=1}^{L} \exp(b_j)}, \qquad k_\ell = d \cdot \rho \cdot L \cdot \tilde{b}_\ell, \qquad \frac{1}{L} \sum_{\ell=1}^{L} \frac{k_\ell}{d} = \rho, \tag{1}$$

i.e., the *average fraction of active columns* equals the target compression rate.

	Learnability			
Method	None	Decompressor	Comp+Decomp	Compression
LLama-3.1-8B (Orthonormal)	-55.2	-17.1	-12.1	
LLama-3.1-8B (MLP)	-	-16.9	-13.0	X2
LLama-3.1-8B (Adaptive)	-11.2	-	-0.9	
LLama-3.1-8B (Orthonormal)	-56.9	-18.3	-13.5	
LLama-3.1-8B (MLP)	-	-19.9	-16.1	X3
LLama-3.1-8B (Adaptive)	-14.4	-	-1.1	
LLama-3.1-8B (Orthonormal)	-58.1	-20.3	-14.7	
LLama-3.1-8B (MLP)	-	-21.4	-17.7	X4
LLama-3.1-8B (Adaptive)	-14.8	-	-1.9	
Baseline	84.5			

Table 1: Delta of different KV-cache compression methods compared to original LLaMA-3.1-8B performance. The table reports performance under three evaluation setups: using no training(None), training a decompressor only, and training both compressor and decompressor jointly. Results are shown for compression factors $\times 2$, $\times 3$, and $\times 4$, with the baseline (no compression) included for reference. We also use both orthonormal matrices for compression and decompression (Orthonormal) and MLPs. Adaptive refers to our approach.

Compression and Maksing At each layer, a nearly-binary column mask is drawn via the hard-concrete distribution [28]. Given R_ℓ and k_ℓ, the mask $m_\ell \in \{0,1\}^d$ is sampled as $m_\ell = \mathrm{HardConcrete}(R_\ell,\ k_\ell,\ T)$, with an annealed temperature T and a straight-through estimator so that $\mathbb{E}\|m_\ell\|_0 \approx k_\ell$. With the full basis $W_\ell \in \mathbb{R}^{d\times d}$ an input $x \in \mathbb{R}^d$ is compressed as $z = x\left(W_\ell \odot m_\ell\right)$, where $\odot$ denotes column-wise masking. This adaptively selects a subset of projection directions per layer while respecting the global budget. Training details and experimental setup can be seen in section B.

5 Results

Our results in table 1 show that using random projections without training for either compression or decompression consistently degrades model performance significantly. To mitigate this, we demonstrate that training the decompressor only while the projection matrix is initialised and fixed as orthonormal is also sufficient for adequate performance; however, it remains significantly inferior to jointly training both. In our experiments, we saw no massive difference between training only linear matrices vs shallow MLPs, thus signifying that using nonlinearity is not an inherently necessary component for KV cache compression. Our main results show that our adaptive approach allows us to achieve $\approx 1\%$ accuracy drop for $\times 2, 3, 4$ compression rates, significantly outperforming other benchmarks.

5.1 Budget Dynamics

To further understand how our method impacts the per-layer budgets, we complete three training runs from different seeds and record the per-layer compression budgets that the models have converged to. In fig. 2, we can see a clear, strong correlation between the depth of the layer and the allowed compression rate. This clear power law indicates that the model has learned to compress earlier layers significantly less than later layers, thereby avoiding the compounding of errors that would otherwise propagate towards later processing modules.

6 Conclusion

We introduced a learnable adaptive KV-cache compression method that allocates layer-wise budgets under a global constraint. Our results show that the approach achieves up to $\times 4$ compression with minimal accuracy loss, while revealing interpretable budget dynamics across layers. Beyond reducing memory and bandwidth demands, our framework opens the door to principled analysis of attention sparsity and more efficient inference scheduling in future LLM systems.

References

[1] J. Ainslie, J. Lee-Thorp, M. De Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[2] E. Arakelyan, P. Minervini, P. Verga, P. Lewis, and I. Augenstein. Flare: faithful logic-aided reasoning and exploration. *arXiv preprint arXiv:2410.11900*, 2024.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[4] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[5] A. L. DeepSeek-AI, B. Feng, B. Wang, B. Wang, B. Liu, C. Zhao, C. Dengr, C. Ruan, D. Dai, D. Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[6] T. Dettmers, R. Svirschevski, V. Egiazarian, D. Kuznedelev, E. Frantar, S. Ashkboos, A. Borzunov, T. Hoefler, and D. Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[8] A. Devoto, Y. Zhao, S. Scardapane, and P. Minervini. A simple and effective l_2 norm-based strategy for kv cache compression. *arXiv preprint arXiv:2406.11430*, 2024.

[9] A. Devoto, Y. Zhao, S. Scardapane, and P. Minervini. A simple and effective l_2 norm-based strategy for KV cache compression. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18476–18499, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics.

[10] H. Dong, X. Yang, Z. Zhang, Z. Wang, Y. Chi, and B. Chen. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. *arXiv preprint arXiv:2402.09398*, 2024.

[11] X. Feng, Z. Wan, M. Wen, S. M. McAleer, Y. Wen, W. Zhang, and J. Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

[12] S. Ge, Y. Zhang, L. Liu, M. Zhang, J. Han, and J. Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

[13] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[14] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[15] J. Huang and K. C.-C. Chang. Towards reasoning in large language models: A survey. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada, July 2023. Association for Computational Linguistics.

[16] S. Imani, L. Du, and H. Shrivastava. MathPrompter: Mathematical reasoning using large language models. In S. Sitaram, B. Beigman Klebanov, and J. D. Williams, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 37–42, Toronto, Canada, July 2023. Association for Computational Linguistics.

[17] A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[18] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[19] Z. Ke, F. Jiao, Y. Ming, X.-P. Nguyen, A. Xu, D. X. Long, M. Li, C. Qin, P. Wang, S. Savarese, et al. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*, 2025.

[20] F. D. Keles, P. M. Wijewardena, and C. Hegde. On the computational complexity of self-attention. In *International conference on algorithmic learning theory*, pages 597–619. PMLR, 2023.

[21] M. Khodak, N. Tenenholtz, L. Mackey, and N. Fusi. Initialization and regularization of factorized neural layers. *arXiv preprint arXiv:2105.01029*, 2021.

[22] J. Kim, J. Park, J. Cho, and D. Papailiopoulos. Lexico: Extreme kv cache compression via sparse coding over universal dictionaries. *arXiv preprint arXiv:2412.08890*, 2024.

[23] S. Kim, C. Hooper, A. Gholami, Z. Dong, X. Li, S. Shen, M. W. Mahoney, and K. Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

[24] V. Klema and A. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

[25] Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

[26] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100, 2024.

[27] A. Liu, B. Feng, B. Wang, B. Wang, B. Liu, C. Zhao, C. Dengr, C. Ruan, D. Dai, D. Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[28] C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[29] J. Luo, W. Zhang, Y. Yuan, Y. Zhao, J. Yang, Y. Gu, B. Wu, B. Chen, Z. Qiao, Q. Long, et al. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025.

[30] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[31] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.

[32] S. Minaee, T. Mikolov, N. Nikzad, M. A. Chenaghlu, R. Socher, X. Amatriain, and J. Gao. Large language models: A survey. *ArXiv*, abs/2402.06196, 2024.

[33] M. Moczulski, M. Denil, J. Appleyard, and N. de Freitas. Acdc: A structured efficient linear layer. *arXiv preprint arXiv:1511.05946*, 2015.

[34] P. Nawrot, A. Łańcucki, M. Chochowski, D. Tarjan, and E. M. Ponti. Dynamic memory compression: Retrofitting llms for accelerated inference. *arXiv preprint arXiv:2403.09636*, 2024.

[35] L. Pan, A. Albalak, X. Wang, and W. Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore, Dec. 2023. Association for Computational Linguistics.

[36] L. Phan, A. Gatti, Z. Han, N. Li, J. Hu, H. Zhang, C. B. C. Zhang, M. Shaaban, J. Ling, S. Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

[37] C. Qu, S. Dai, X. Wei, H. Cai, S. Wang, D. Yin, J. Xu, and J.-R. Wen. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343, 2025.

[38] A. Rogers, M. Gardner, and I. Augenstein. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *ACM Computing Surveys*, 55(10):1–45, 2023.

[39] B. Romera-Paredes, M. Barekatain, A. Novikov, M. Balog, M. P. Kumar, E. Dupont, F. J. Ruiz, J. S. Ellenberg, P. Wang, O. Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[40] Z. Shen. Llm with tools: A survey. *arXiv preprint arXiv:2409.18807*, 2024.

[41] P. Singhania, S. Singh, S. He, S. Feizi, and A. Bhatele. Loki: Low-rank keys for efficient sparse attention. *Advances in Neural Information Processing Systems*, 37:16692–16723, 2024.

[42] G. Sun, M. Jin, Z. Wang, C.-L. Wang, S. Ma, Q. Wang, T. Geng, Y. N. Wu, Y. Zhang, and D. Liu. Visual agents as fast and slow thinkers. *arXiv preprint arXiv:2408.08862*, 2024.

[43] Y. Sun, L. Dong, Y. Zhu, S. Huang, W. Wang, S. Ma, Q. Zhang, J. Wang, and F. Wei. You only cache once: Decoder-decoder architectures for language models. *Advances in Neural Information Processing Systems*, 37:7339–7361, 2024.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[45] Z. Wan, X. Wu, Y. Zhang, Y. Xin, C. Tao, Z. Zhu, X. Wang, S. Luo, J. Xiong, and M. Zhang. D2o: Dynamic discriminative operations for efficient generative inference of large language models. *arXiv preprint arXiv:2406.13035*, 2024.

[46] H. Wang, S. Agarwal, and D. Papailiopoulos. Pufferfish: Communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems*, 3:365–386, 2021.

[47] Z. Wang, B. Jin, Z. Yu, and M. Zhang. Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks. *arXiv preprint arXiv:2407.08454*, 2024.

[48] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, F. Xia, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

[49] W. Wen, C. Xu, C. Wu, Y. Wang, Y. Chen, and H. Li. Coordinating filters for faster deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 658–666, 2017.

[50] Y. Xie, A. Goyal, W. Zheng, M.-Y. Kan, T. P. Lillicrap, K. Kawaguchi, and M. Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

[51] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[52] L. Ye, Z. Tao, Y. Huang, and Y. Li. Chunkattention: Efficient self-attention with prefix-aware kv cache and two-phase partition. *arXiv preprint arXiv:2402.15220*, 2024.

[53] Z. Yuan, Y. Shang, Y. Song, Q. Wu, Y. Yan, and G. Sun. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

[54] Y. Zhang, Y. Du, G. Luo, Y. Zhong, Z. Zhang, S. Liu, and R. Ji. Cam: Cache merging for memory-efficient llms inference. In *Forty-first international conference on machine learning*, 2024.

[55] Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, R. Cai, Z. Song, Y. Tian, C. Ré, C. Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

[56] X. Zhou, W. Wang, M. Zeng, J. Guo, X. Liu, L. Shen, M. Zhang, and L. Ding. Dynamickv: Task-aware adaptive kv cache compression for long context llms. *arXiv preprint arXiv:2412.14838*, 2024.

A Memory and Bandwidth Bottlenecks

While caching avoids repeated projection cost, it introduces linearly growing memory overhead. The total size of the KV cache (in bytes) over a generation of length T, batch size B, and element precision e is:

$$\text{KV\_size} = 2 \cdot L \cdot H \cdot d \cdot B \cdot T \cdot e. \tag{2}$$

As T or B grows, the cache footprint may dominate the memory budget, even exceeding the model's own parameter size. Moreover, each decoding step requires reading from memory the cached $\mathbf{K}, \mathbf{V}$ to fuse with the new query; thus, memory bandwidth becomes a limiting factor. In practice, the decoding latency is often memory-bandwidth bound rather than FLOP bound.

B Training Details and Experimental setup

All parameters are calibrated *offline* on pre-existing KV-cache samples using only mean-squared reconstruction:

$$\mathcal{L} = \frac{1}{L}\sum_{\ell=1}^{L}\Big(\|K_\ell - \hat{K}_\ell\|_2^2 + \|V_\ell - \hat{V}_\ell\|_2^2\Big), \quad \hat{K}_\ell = \mathrm{Dec}_\ell(\mathrm{Enc}_\ell(K_\ell)),\ \hat{V}_\ell = \mathrm{Dec}_\ell(\mathrm{Enc}_\ell(V_\ell)). \tag{3}$$

No language-model loss is required; the softmax-coupled budgets enforce the exact average rate ρ during calibration.

In our setup, we use LLAMA-3.1-8B trained with KV cache samples from WIKITEXT-2-RAW-V1 and evaluate on GSM8K [4] and Math500 [14].