

PALU: KV-CACHE COMPRESSION WITH LOW-RANK PROJECTION

Chi-Chih Chang^{1,3*} Wei-Cheng Lin^{1*} Chien-Yu Lin^{2*}
 Chong-Yan Chen¹ Yu-Fang Hu¹ Pei-Shuo Wang¹ Ning-Chi Huang¹
 Luis Ceze² Mohamed S. Abdelfattah³ Kai-Chiang Wu¹

¹National Yang Ming Chiao Tung University ²University of Washington ³Cornell University

ABSTRACT

Post-training KV-Cache compression methods typically either sample a subset of effectual tokens or quantize the data into lower numerical bit width. However, these methods cannot exploit redundancy in the hidden dimension of the KV tensors. This paper presents a hidden dimension compression approach called *Palu*, a KV-Cache compression framework that utilizes low-rank projection to reduce inference-time LLM memory usage. *Palu* decomposes the linear layers into low-rank matrices, caches compressed intermediate states, and reconstructs the full keys and values on the fly. To improve accuracy, compression rate, and efficiency, *Palu* further encompasses (1) a medium-grained low-rank decomposition scheme, (2) an efficient rank search algorithm, (3) low-rank-aware quantization compatibility enhancements, and (4) optimized GPU kernels with operators fusion. Extensive experiments with popular LLMs show that *Palu* compresses KV-Cache by 50%, while maintaining strong accuracy and delivering up to **1.89× speedup** on the RoPE-based attention module. When combined with quantization, *Palu*'s inherent quantization-friendly design yields small to negligible extra accuracy degradation, while saving additional memory than quantization-only methods and achieving up to **2.91× speedup** for the RoPE-based attention. Moreover, it maintains comparable or even **better accuracy (up to 1.19 lower perplexity)** compared to quantization-only methods. These results demonstrate *Palu*'s superior capability to effectively address the efficiency and memory challenges of LLM inference posed by KV-Cache. Our code is publicly available at: <https://github.com/shadowpa0327/Palu>

1 INTRODUCTION

Large language models (LLMs) have propelled AI into new applications and capabilities, providing a high-level intelligence that previous machine learning (ML) models could not achieve. To speed up inference, caching the Key-Value states (KV-Cache) in memory is a simple yet effective technique. However, the size of the KV-Cache can grow rapidly, straining memory capacity and bandwidth especially with long context lengths (Fu, 2024); further, the memory-bounded nature of the decoding stage limits inference speed when loading KV-Cache data (Gholami et al., 2024). Therefore, KV-Cache compression has become a central research topic for running LLMs efficiently.

Although emerging attention mechanisms such as Multi-Query Attention (MQA) (Shazeer, 2019), Group-Query Attention (GQA) (Ainslie et al., 2023) and Multi-head Latent Attention (MLA) (DeepSeek-AI et al., 2024) can reduce KV-Cache size, it either requires model pre-training or has a significant impact on model's accuracy when converting from traditional Multi-Head Attention (MHA) (Chen et al., 2024). In contrast, post-training KV-Cache compression techniques offer an alternative approach to advance efficiency for existing models. Among various KV-Cache compression methods, quantization (Liu et al., 2024b; Hooper et al., 2024) and token eviction (Zhang et al., 2024; Xiao et al., 2024) stand out as effective strategies to reduce the memory footprint of KV-Cache.

*Equal contribution

[†]Correspondence to: Chi-Chih Chang, cc2869@cornell.edu

Quantization methods aim to reduce the bit-width used to represent each piece of data, while token eviction techniques focus on retaining a partial set of KV-Cache. However, both methods neglect the hidden dimensions of the KV-Cache, where substantial redundancy often resides. To capitalize on this untapped potential, we introduce *Palu*, a post-training KV-Cache compression framework that leverages low-rank projection to reduce the hidden dimension of KV tensors, offering an additional and orthogonal compression dimension to existing quantization and token eviction methods.

A naive way to utilize low-rank projection for compressing the KV-Cache is by directly mapping cached matrices into low-rank space (Jolliffe & Cadima, 2016; Zhao et al., 2024). However, this approach imposes an unacceptably heavy overhead of computing the decomposition matrices during runtime. To avoid this, *Palu* statically decomposes the Key and Value-projection weight matrices and caches the latent representations of the low-rank decomposition (see Fig. 1). This innovative design enables *Palu* to reduce memory while mitigating the runtime overhead of KV-Cache low-rank decomposition.

In designing an effective decomposition strategy for attention modules with multiple attention heads, we observed a clear trade-off between accuracy and reconstruction overhead. Decomposing the projection matrices across all attention heads together improves accuracy by preserving global information, but this approach significantly increases reconstruction costs. On the other hand, decomposing each head separately reduces reconstruction overhead but leads to a higher loss in accuracy. To address this, *Palu* introduces a medium-grained, group-head low-rank decomposition that strikes a balance between accuracy and reconstruction efficiency.

For LLMs, each linear projection module has a different sensitivity to compression (Sharma et al., 2023; Yuan et al., 2023). To exploit the sensitivity and improve accuracy, we design an efficient *rank search algorithm* based on Fisher information (Ly et al., 2017; Liu et al., 2021). Our algorithm automatically assigns a higher rank for important matrices and lower ranks for less critical ones, boosting accuracy at the same overall KV-Cache compression rate.

In addition to its low-rank decomposition, *Palu* is compatible with quantization techniques. We found that low-rank decomposition can introduce severe outliers in the latent representation, which significantly hinders accurate low-bit quantization. Although the Hadamard transformation has been shown to be effective for outlier elimination in recent studies (Tseng et al., 2024; Ashkboos et al., 2024b; Liu et al., 2024a; Chiang et al., 2024), its integration often introduces computational overhead during runtime. However, *Palu*'s inherent matrix pair structure makes it highly compatible with this technique, allowing the transformation matrices to be seamlessly fused into the forward and backward matrices, effectively mitigating outliers without impacting runtime efficiency.

We evaluate *Palu* on widely used LLMs and benchmarks. Our experiments demonstrate that *Palu* maintains strong zero-shot accuracy and perplexity with up to 50% low-rank compression. Moreover, when combining low-rank compression with quantization, *Palu* achieves an impressive **over 91.25% compression (11.4 \times reduction)** and yields a *significantly lower perplexity of 1.19* than KVQuant (Hooper et al., 2024), a state-of-the-art KV-Cache quantization method, which only achieves an 87.5% compression rate.

For latency evaluation, under a 50% KV-Cache compression rate without quantization, *Palu* demonstrates up to **$1.89\times$ and $2.2\times$ speedup** for RoPE-based and non-RoPE attention modules. When integrated with quantization, *Palu* achieves up to **$2.91\times$ and $6.17\times$ acceleration** on RoPE-based and non-RoPE attention, respectively. These results underscore *Palu*'s ability to significantly reduce KV-Cache memory footprint while boosting inference efficiency for LLMs.

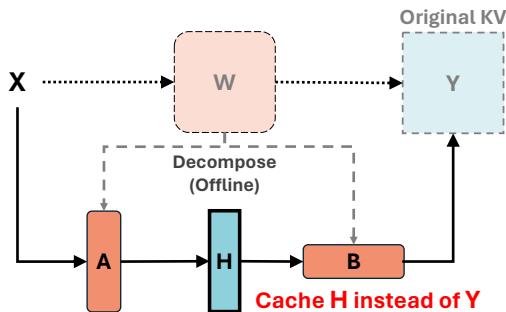


Figure 1: *Palu*'s low-rank projection method for KV-Cache reduction. A weight matrix W of linear projection is decomposed into two low-rank matrices. Input X is down-projected to latent representation H , which is cached. Y can be reconstructed from H using the up-projection matrix B .

non-RoPE attention modules, *Palu* achieves greater speedup compared to RoPE-based attention, as their reconstruction can be avoided with matrix fusion. A detailed workflow of *Palu* in the context of Llama’s RoPE-based attention is illustrated in Fig. 2.

3.2 DECOMPOSITION GRANULARITY

3.2.1 MULTI-HEAD LOW-RANK DECOMPOSITION

We name the per-head decomposition scheme in Sec. 3.1.1 as *multi-head low-rank decomposition (M-LRD)*. We found M-LRD often causes a non-negligible accuracy degradation (discussed further in Sec. 4.2), possibly because SVD fails to capture the common information shared across heads. Therefore, alternative approaches are needed to preserve model accuracy.

3.2.2 JOINT-HEAD LOW-RANK DECOMPOSITION

An alternative approach is to jointly decompose weight matrices for all heads. By considering the combined weight matrix $\mathbf{W}_{\text{joint}} = [\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_n] \in \mathbb{R}^{d \times (d_h \cdot n_h)}$, we can perform a single low-rank decomposition $\mathbf{W}_{\text{joint}} \approx \mathbf{A}_{\text{joint}} \mathbf{B}_{\text{joint}}$, where $\mathbf{A}_{\text{joint}} \in \mathbb{R}^{d \times r_{\text{joint}}}$ and $\mathbf{B}_{\text{joint}} \in \mathbb{R}^{r_{\text{joint}} \times (d_h \cdot n_h)}$. We call this scheme *joint-head low-rank decomposition (J-LRD)*.

J-LRD has the advantage of preserving the common principal components shared among different heads. This occurs because SVD is particularly effective at capturing the dominant components when applied to a larger, combined matrix, resulting in a more accurate approximation.

For J-LRD, the joint latent representation shared among all heads can be computed with $\mathbf{h}_{\text{joint}} = \mathbf{x} \mathbf{A}_{\text{joint}}$. During decoding, the original states for each head can be reconstructed via

$$[\mathbf{y}_1, \dots, \mathbf{y}_n] = \mathbf{h}_{\text{joint}} \mathbf{B}_{\text{joint}}.$$

High Inference Overhead. Despite better preserving model accuracy, J-LRD introduces *significant computational and memory overhead* during decoding. Specifically, the total number of floating point operations (FLOPs) to reconstruct the Key or Value state of one head now becomes $r_{\text{joint}} \cdot d_h \cdot n$. Assuming the same size as the total low-rank latent representations (*i.e.*, $r_{\text{joint}} = \sum_{i=1}^n r_i$), the total reconstruction cost is n times higher than M-LRD, whose total FLOPs is $r_i \cdot d_h \cdot n$. When considering the matrix fusion in Sec. 3.1.1, the fused matrix of J-LRD has a size of $r_{\text{joint}} \cdot d \cdot n$, which is also n times larger than M-LRD, leading to substantial higher memory consumption.

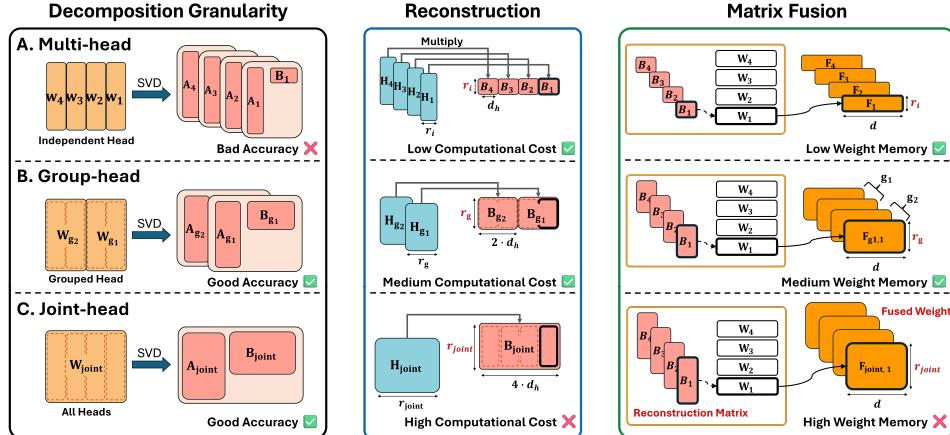


Figure 3: Performing decomposition at different granularities. Jointly decomposing multiple heads can achieve higher accuracy. Assuming the same total size of the latent representations (*i.e.*, $4 \cdot r_i = 2 \cdot r_g = r_{\text{joint}}$), the FLOPs for reconstruction overhead in joint-head decomposition schemes are 4 times larger than those in multi-head ones.

3.2.3 GROUP-HEAD LOW-RANK DECOMPOSITION

To balance the trade-off between accuracy and reconstruction cost, we propose *group-head low-rank decomposition* (G-LRD). G-LRD decomposes the matrices for a group of heads together. With combined weight matrices, it captures shared information within each group while limiting computational overhead and preserving accuracy.

To illustrate the G-LRD process, consider the weight matrices for a group of s heads, $\mathbf{W}_{g_j} = [\mathbf{W}_{j,1} \dots \mathbf{W}_{j,s}]$, where $\mathbf{W}_{g_j} \in \mathbb{R}^{d \times (d_h \cdot s)}$. We low-rank decompose $\mathbf{W}_{g_j} \approx \mathbf{A}_{g_j} \mathbf{B}_{g_j}$, where $\mathbf{A}_{g_j} \in \mathbb{R}^{d \times r_g}$ and $\mathbf{B}_{g_j} \in \mathbb{R}^{r_g \times (d_h \cdot s)}$. The latent representation shared among attention heads in the same group can be computed as $\mathbf{h}_{g_j} = \mathbf{x} \mathbf{U}_{g_j}$. During decoding, the original key or value for each head can be reconstructed via

$$[\mathbf{y}_{j,1} \dots \mathbf{y}_{j,s}] = \mathbf{h}_{g_j} \mathbf{B}_{g_j}.$$

The FLOPs for reconstructing the keys and values for each head in G-LRD is $r_g \cdot d_h \cdot n_g$, where $n_g = \frac{n}{s}$ is the number of groups. Comparing the cost to J-LRD and assuming the same total rank size ($r_g \cdot n_g = r_{\text{joint}}$), G-LRD reduces the reconstruction cost by n_g . Similarly, G-LRD also reduces the fused matrix size by n_g . To sum up, G-LRD offers a middle ground between computation overhead and approximation accuracy. We illustrate M-LRD, J-LRD and G-LRD in Fig. 3. Please refer to Appendix C for further discussions on the costs of different decomposition granularities.

3.3 AUTOMATIC RANK ALLOCATION

To allocate an ideal rank size to the decomposition target, it is crucial to accurately estimate the importance of the target matrix (*e.g.*, grouped weights). In *Palu*, we identify **Fisher information** (Ly et al., 2017; Liu et al., 2021) as an accurate approximator since it can quantify the amount of information for each parameter. We then employ the sum of Fisher information to estimate the importance of the weight matrix of each linear layer (Abdelfattah et al., 2021).

Assuming that the compression sensitivity is proportional to Fisher information, we determine the rank for each weight matrix by computing the ratio of its Fisher information to the total Fisher information across all decomposition targets. We use this ratio to allocate the compression rate (*i.e.*, rank level r), ensuring that more important layers retain higher rank levels. For a detailed ablation study on our Automatic Rank Allocation, please refer to Appendix G.3.

3.4 QUANTIZATION COMPATIBILITY

We integrate quantization into *Palu* to compress the KV-Cache further. We observe that low-rank compressed latent representations have severe outliers, which limit quantization applicability in *Palu*. Unlike natural outliers described in previous KV-Cache quantization literature (Liu et al., 2024b; Hooper et al., 2024), these outliers are induced by SVD-based low-rank factorization.

Fig. 4 (a) shows the distribution of low-rank compressed key states from a layer of Llama-2 with G-LRD. Repeating outlier patterns appear at the beginning of each decomposed group because SVD arranges larger eigenvalues in the initial rows or columns, resulting in rapidly descending values in the latent representation. This pattern stretches the data distribution and hurts quantization accuracy.

Inspired by recent LLM quantization literature (Ashkboos et al., 2024b; Tseng et al., 2024), we apply the Walsh-Hadamard transform (WHT, Fino & Algazi) to eliminate outliers (Fig. 4 (b)), enabling a high quantization accuracy. However, this transformation introduces an extra matrix multiplication with associated runtime overhead. Unlike earlier methods (Ashkboos et al., 2024b) that must apply online WHT when quantizing KV-Cache, we optimize this process by integrating the Hadamard matrix into low-rank decomposed weights with no additional compute overhead, as described by

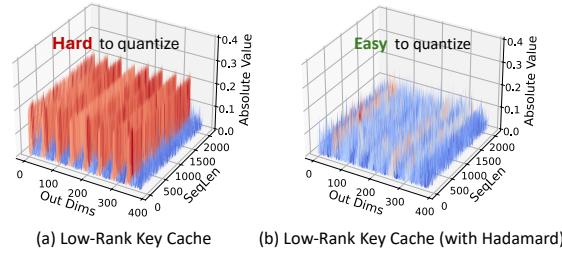


Figure 4: Activation distribution of the low-rank key caches at the 4th Llama-2 attention layer.

results demonstrate that combining low-rank compression and quantization significantly enhances inference efficiency, particularly in long-context scenarios.

End-to-end speedup. We present the end-to-end speedups in Fig. 5 (c) and (d), measuring the latency of generating the next token at various input lengths and comparing the results to the FP16 baseline. Similar to the attention performance results, *Palu* shows minimal or no speedup for short sequences but delivers significant acceleration for longer sequences. Without quantization, *Palu* achieves up to **1.71x and 2.05x speedups** for RoPE-based and non-RoPE models, respectively. With a 50% compression rate, *Palu* runs up to 32K input length on an RTX 4090 GPU. By incorporating 4-bit quantization, *Palu* handles even longer sequences and delivers **2.59x and 5.53x end-to-end speedups** at a 64K sequence length. *Palu* integrated with quantization provides a substantial speed advantage over KIVI-4-bit, which only reaches 1.78x and 1.81x speedups at 32K sequence length for RoPE and non-RoPE scenarios, respectively, and is out-of-memory for longer sequences.

4.5.2 KERNEL FOR ROPE-BASED ATTENTION SCORE

In this section, we evaluate the performance of our online reconstruction kernel for RoPE-based attention scores. We measure latency from the **pre-RoPE query vector to post-GEMV attention score**, and compare it with PyTorch’s GEMV, which is used in the baseline attention (see Fig. 2).

We present speedups for group size 1, 4, and 32 at different sequence lengths in Fig. 6. For gs-32 (J-LRD), the highest accuracy decomposition, the high reconstruction cost causes a significant slowdown across all sequence lengths. For gs-1 (M-LRD), our kernel achieves up to a 3.56x speedup at sequence length 16K, showing strong performance when moderate accuracy loss is acceptable. For gs-4 (G-LRD), our kernel reaches up to 1.95x speedup. These results emphasize the need to explore various decomposition granularities for better accuracy and speed tradeoffs.

We also observe that speedup decreases for sequence lengths beyond 16K due to rising reconstruction costs, shifting the online reconstruction from memory- to compute-bound. A potential optimization is to quantize the decomposed weight matrices further and leverage high-throughput, low-precision hardware (*e.g.*, INT4 Tensor Cores) for online reconstruction, which we leave for future work. Despite the speedup drop at longer lengths, *Palu*’s overall attention speedup increases with longer input, thanks to matrix fusion on the Value state and the reduced memory footprint.

5 RELATED WORK

SVD for LLM Compression. Several works have explored using SVD to compress LLMs. An early approach (Noach & Goldberg, 2020) applied standard SVD to weight matrices, resulting in significant compression errors. FWSVD (Hsu et al., 2022) addressed this by using Fisher information to prioritize parameters, while ASVD (Yuan et al., 2023) considered activation outliers. SVD-LLM (Wang et al., 2024) further minimized compression loss for each singular value. Unlike these methods, which compress model weights, *Palu* focuses on reducing KV-Cache size. A concurrent work, (Yu et al., 2024), also explores KV-Cache compression using low-rank projection, deriving low-rank matrices from KV-Cache with calibration data and grouping attention heads similarly to *Palu*’s G-LRD. However, it requires LoRA finetuning after decomposition. In contrast, *Palu* directly decomposes the weight matrix, preserving accuracy without finetuning, and introduces additional innovations, including rank search, quantization integration, and optimized GPU kernels.

KV-Cache Compression. Quantization is a widely used technique for compressing KV-Cache. Atom (Zhao et al., 2023) applies simple per-token quantization, while WKVQuant (Yue et al., 2024) introduces a two-level scheme to enhance accuracy. KIVI (Liu et al., 2024b) uses per-channel and per-token quantization for Keys and Values, combined with fine-grained group quantization at group size 32. KVQuant (Hooper et al., 2024) employs a similar setup but incorporates non-uniform quantization and sparse matrices to handle outliers. On top of these approaches, GEAR adds a

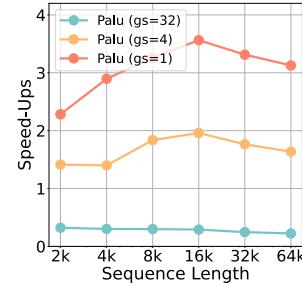


Figure 6: Speedup of *Palu*’s attention score kernel with online reconstruction.

low-rank matrix to compensate for quantization errors. In *Palu*, we leverage low-rank techniques to exploit hidden dimension redundancy and, with simple per-token quantization, achieve outstanding compression results.

MLA. The recently released DeepSeek-V2 model (DeepSeek-AI et al., 2024) introduces the MLA mechanism, which reduces KV-Cache size by down-projecting Key and Value to a low-rank space and reconstructing them to full rank at runtime. Although MLA may seem similar to *Palu* at a high level, particularly with J-LRD, our design and derivation processes are fundamentally different. Unlike MLA, a new attention mechanism requiring pre-training, *Palu* is specifically designed for post-training integration. *Palu* focuses on converting existing models with MHA or GQA to support low-rank compressed KV-Cache, preserving high accuracy while enhancing inference efficiency.

6 CONCLUSION

We introduce *Palu*, a novel KV-Cache compression framework that decomposes linear projection weight matrices and caches the compressed latent representations. We propose various optimizations, including group-head low-rank decomposition, automatic rank allocation algorithm, quantization compatibility enhancement, and customized kernels with operator fusion. With these optimizations, *Palu* can maintain accuracy while achieving significant memory reduction and high inference speedup. Experiments show that, with 50% low-rank compression and 4-bit quantization, *Palu* accelerates RoPE-based attention module by up to $2.91\times$ and delivers up to $2.2\times$ end-to-end speedup for the same RoPE-based model, while preserving strong accuracy on various benchmarks.

REFERENCES

- Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight NAS. 2021. URL <https://openreview.net/forum?id=0cmMMY8J5q>.
- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Shanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.
- Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefer, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024a. URL <https://openreview.net/forum?id=vXxardq6db>.
- Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefer, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024b.
- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.
- Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.
- Yuang Chen, Cheng Zhang, Xitong Gao, Robert D. Mullins, George A. Constantinides, and Yiren Zhao. Optimised grouped-query attention mechanism for transformers, 2024. URL <https://arxiv.org/abs/2406.14963>.
- Hung-Yueh Chiang, Chi-Chih Chang, Natalia Frumkin, Kai-Chiang Wu, and Diana Marculescu. Quamba: A post-training quantization recipe for selective state space models, 2024. URL <https://arxiv.org/abs/2410.13229>.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.365. URL <https://aclanthology.org/2021.naacl-main.365>.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jia Shi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhua Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyu Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahaedokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur

Celebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparth, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaee, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhota, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keaneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvaraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghatham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondi, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Tim-

othy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vitor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1074–1084, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1102. URL <https://aclanthology.org/P19-1102>.

Fino and Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, C-25(11):1142–1146, 1976. doi: 10.1109/TC.1976.1674569.

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

Yao Fu. Challenges in deploying long-context transformers: A theoretical peak performance analysis, 2024. URL <https://arxiv.org/abs/2405.08944>.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muenninghoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lin-tang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL <https://zenodo.org/records/10256836>.

Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. Ai and memory wall, 2024. URL <https://arxiv.org/abs/2403.14123>.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *CoRR*, abs/1911.12237, 2019. URL <https://arxiv.org/abs/1911.12237>.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian J. McAuley. Longcoder: A long-range pre-trained language model for code completion. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 12098–12107. PMLR, 2023. URL <https://proceedings.mlr.press/v202/guo23j.html>.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

Yen-Chang Hsu, Ting Hua, Sungjen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=uPv9Y3gmAI5>.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZeVKeefYf9>.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

- Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL <https://aclanthology.org/P17-1147>.
- Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can context length of open-source llms truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.
- Xin Li and Dan Roth. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*, 2002. URL <https://aclanthology.org/C02-1150/>.
- Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7021–7032. PMLR, 2021. URL <http://proceedings.mlr.press/v139/liu21ab.html>.
- Tianyang Liu, Canwen Xu, and Julian McAuley. Repobench: Benchmarking repository-level code auto-completion systems, 2023.
- Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant–llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024a.
- Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024b.
- Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul Grasman, and Eric-Jan Wagenmakers. A tutorial on fisher information, 2017.
- Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models, 2023.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.
- Matan Ben Noach and Yoav Goldberg. Compressing pre-trained language models by matrix decomposition. In Kam-Fai Wong, Kevin Knight, and Hua Wu (eds.), *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, AACL/IJCNLP 2020, Suzhou, China, December 4-7, 2020*, pp. 884–889. Association for Computational Linguistics, 2020. URL <https://aclanthology.org/2020.aacl-main.88/>.
- Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=R8sQPPGCV0>.

- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
- Pratyusha Sharma, Jordan T. Ash, and Dipendra Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction, 2023.
- Noam Shazeer. Fast transformer decoding: One write-head is all you need, 2019. URL <https://arxiv.org/abs/1911.02150>.
- Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864, 2021. URL <https://arxiv.org/abs/2104.09864>.
- Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poult, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Bin Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.
- Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fdbd053c1c4a845aa-Abstract.html>.
- 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.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL <https://aclanthology.org/2020.emnlp-demos.6>.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=NG7ssS51zVF>.

Hao Yu, Zelan Yang, Shen Li, Yong Li, and Jianxin Wu. Effectively compress kv heads for llm, 2024. URL <https://arxiv.org/abs/2406.07056>.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models, 2023.

Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. Wkvquant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*, 2024.

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

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuan-dong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. *CoRR*, abs/2403.03507, 2024. doi: 10.48550/ARXIV.2403.03507. URL <https://doi.org/10.48550/arXiv.2403.03507>.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zhenga, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *arXiv preprint arXiv:2310.19102*, 2023.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. QMSum: A new benchmark for query-based multi-domain meeting summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5905–5921, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nacl-main.472. URL <https://aclanthology.org/2021.nacl-main.472>.

APPENDIX

A QUANTIZATION BASICS

Quantization techniques use discrete low-bit values to approximate high-precision floating points. The general asymmetric uniform quantization function is defined as:

$$\bar{\mathbf{X}} = \text{clamp}\left(\left\lfloor \frac{\mathbf{X}}{s} \right\rfloor + z, 0, 2^B - 1\right), \quad (8)$$

where $\bar{\mathbf{X}}$ denotes the approximated tensor with low-bit representations (i.e., 4-bit integers), \mathbf{X} is the floating-point tensor, $s = \frac{\mathbf{X}_{\max} - \mathbf{X}_{\min}}{2^B - 1}$ is the scaling factor, and $z = \left\lfloor \frac{-\mathbf{X}_{\min}}{s} \right\rfloor$ is a zero-point. The $\lfloor \cdot \rfloor$ is the rounding operation.

B KERNEL IMPLEMENTATION DETAILS

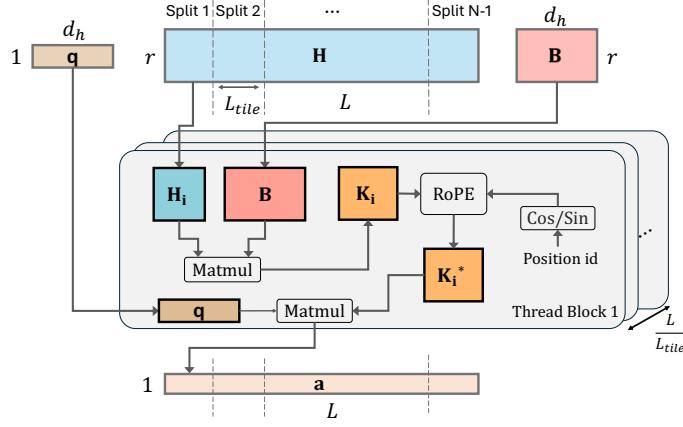


Figure 7: Illustration of our fused GPU kernel for computing attention scores with online reconstruction. In this figure, \mathbf{q} represents the query vector, \mathbf{H} denotes the low-rank compressed key states, and \mathbf{B} stands for the reconstruction matrices.

Kernel for attention score calculation with reconstruction. The central idea of *Palu* is to leverage low-rank latent representations to accelerate the attention mechanism by reducing data transfer overhead. Instead of working directly with the full-sized key matrix, we store and transfer a compressed low-rank latent representation, denoted as $\mathbf{H} \in \mathbb{R}^{L \times r}$. During computation, our custom GPU kernel performs an on-the-fly reconstruction using a reconstruction matrix $\mathbf{B} \in \mathbb{R}^{r \times d_h}$, producing a restored key matrix $\mathbf{K} \in \mathbb{R}^{L \times d_h}$, where L is the sequence length, d_h is the hidden dimension, and r denote the remaining rank after performing low-rank projection. The query vector, represented as $\mathbf{q} \in \mathbb{R}^{1 \times d_h}$, then multiplies with the reconstructed keys to obtain the attention scores.

To efficiently leverage parallelism, we perform tiling along the sequence length dimension L . Specifically, we split the sequence into smaller tiles of size L_{tile} , assigning each tile to a dedicated thread block. Each thread block independently reconstructs a submatrix $\mathbf{H}_i \in \mathbb{R}^{L_{\text{tile}} \times d_h}$ from the low-rank latent representation \mathbf{H} , then applies the positional embedding using RoPE, and finally performs the matrix-vector multiplication between \mathbf{q} and \mathbf{H}_i to produce partial attention scores. This design ensures that all intermediate computations, from reconstruction to embedding and final multiplication, remain entirely in on-chip memory (i.e., share memory), thus minimizing high-latency memory access and taking full advantage of the GPU’s parallel processing capabilities to achieve significant speedups.

C DISCUSSION REGARDING MEMORY USAGE

In this work, the experimental results focus on the compression rate of the KV-Cache as a key metric. However, it is crucial to consider overall memory savings as a more significant factor. For instance, as demonstrated in Sec. 2.2, a typical compression rate of 30% can lead to an increase in weight size by approximately 40%. This increase is calculated under the assumption that $m = n$ and $r = 0.7n$, resulting in the equation $\frac{mr+nr}{mn} = 1.4$. Such an increase indicates substantial extra memory usage.

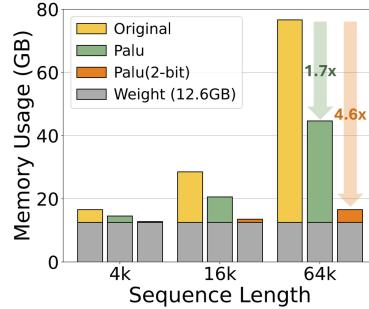
This issue primarily arises in J-LRD decomposition schemes, where the projections of all heads are decomposed jointly. In contrast, our M-LRD decomposition schemes and optimized G-LRD schemes involve non-square target matrices. For example, in the G-LRD scheme with a group size of 4, the target matrix is formed by concatenating the original projection matrices of each attention head in the group. In the Llama-2-7b model, with an embedding dimension of 4096 and head dimensions of 128, each projection matrix is 4096×128 , resulting in a concatenated matrix of size 4096×512 . In this case, the dimensions should be considered as $m = 8n$. Applying the referenced equation $\frac{mr+nr}{mn}$ with $r = 0.7n$, we find that $\frac{mr+nr}{mn} = 0.7875$, indicating no additional storage cost and, in fact, achieving an additional 21.25% memory savings.

Furthermore, it is important to highlight that the weights associated with the K and V projections account for only 2 out of 7 linear layers within transformer blocks, comprising merely 16% of the parameters in Llama-2-7b models. This limits the overall impact on memory usage. Thus, while J-LRD may incur overhead, the M-LRD and G-LRD schemes provide efficient alternatives that do not lead to increased memory usage, making them viable options for practical applications.

D OVERALL MEMORY REDUCTION

In Fig. 8, we show the total memory usage, including model weights and KV-Cache under different sequence lengths. We observe that KV-Cache quickly becomes the memory bottleneck as sequence length grows. At a sequence length of 64k, the KV-Cache constitutes 78% of the total memory consumption. With 50% low-rank compression, *Palu* effectively reduces memory usage by $1.7\times$, and when combined with 2-bit quantization, it further decreases total memory usage by $4.6\times$.

Figure 8: Total memory usage for various sequence lengths in Llama-2-7B. low-rank compression is at 50% for *Palu*.



E INTEGRATING *Palu* WITH LoRA FINETUNE

LoRA (Hu et al., 2022) has become one of the most widely used efficient fine-tuning techniques for adapting models to particular tasks or domains with limited data. It has also been applied with LLM compression approaches (Wang et al., 2024; Ma et al., 2023) as a post-compression recovery technique to recover information loss after compression. In *Palu*, LoRA is also applicable to boost the accuracy further.

Table 6: Ablation study of low-rank decomposition group size on perplexity for the Llama-2-7B model at a 50% compression rate using Wikitext-2.

Method	Group Size	Perplexity
Baseline	-	5.47
J-LRD	32	5.62
	16	5.74
	8	5.88
	4	6.01
	2	6.42
M-LRD	1	6.81

G.2 INFLUENCE OF WALSH-HADAMARD TRANSFORM

We conduct the ablation study to profile the benefits of applying the Walsh-Hadamard Transform (WHT). Experiment results are reported at Table 7. On the 3-bit quantization level, we observe that the Hadamard Transform only brings a slight amount of perplexity. However, when we quantize the low-rank representation more extremely (*i.e.*, 2-bit), we can observe a notable 4.17 perplexity enhancements. It’s worth re-emphasizing that Hadamard Transform will not bring extra overhead during inference, as *Palu* optimizes the WHT process via offline preprocessing. The reader may refer to Sec. 3.4 for more details.

G.3 AUTOMATIC RANK ALLOCATION VS. UNIFORM RANK ALLOCATION

Table 8 presents the ablation study on the impact of different rank allocation schemes on the model’s accuracy. Applying rank searching results in a notable performance improvement. For instance, at a compression rate of 50%, there is a significant reduction in perplexity by 2.18. Fig. 9 visualizes the rank allocation across different transformer blocks for key and value projection layers. The results clearly demonstrate a non-uniform allocation result. Specifically, we observe that the value is generally allocated a higher rank than the key. Additionally, the first half of the layers are assigned higher ranks, indicating their greater importance in preserving model performance. This visualization underscores the effectiveness of our rank search algorithm in identifying and allocating appropriate ranks to different components, thereby optimizing the balance between compression and accuracy.

Table 7: Ablation Study on different quantization settings for quantizing low-rank latent representations. Same as Sec. 4.3, we use the WikiText-2 with sequence length set to 4096 as the evaluation benchmark.

Method	Wikitext-2 PPL ↓
Llama2-7B	5.12
Palu-30% (FP16)	5.25
+ 3-bits w/o Hadamard	5.52
+ 3-bits w Hadamard	5.33 (0.19↓)
+ 2-bits w/o Hadamard	9.48
+ 2-bits w Hadamard	5.77 (3.71↓)
Palu-50% (FP16)	5.63
+ 3-bits w/o Hadamard	5.99
+ 3-bits w Hadamard	5.77 (0.22↓)
+ 2-bits w/o Hadamard	10.58
+ 2-bits w Hadamard	6.41 (4.17↓)

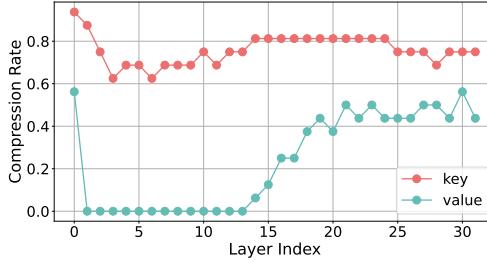


Figure 9: Visualization of layer-wise low-rank compression rate on Llama-2-7B with 50% of overall compression rate. Here, compression rates (*i.e.*, rank) are allocated using the proposed Fisher Information-based automated rank allocation algorithm.

Table 8: Ablation study on w/ and w/o rank search. We use Llama-2-7b and Wikitext-2 with sequence length 2048 as the benchmark.

	Rate=30%	Rate=50%	Rate=70%
Uniform	6.34	7.36	10.77
Automatic (ours)	5.62 (0.72↓)	6.02 (1.36↓)	8.59 (2.18↓)

H EXPERIMENT DETAILS

H.1 ZERO-SHOT EVALUATION DETAILS

We selected six zero-shot tasks from the LM-eval benchmark to evaluate *Palu*:

- OpenBookQA (accuracy, Mihaylov et al.)
- HellaSwag (acc_norm, Zellers et al.)
- PIQA (accuracy, Bisk et al.)
- ARC-Easy (accuracy, Clark et al.)
- ARC-Challenge (acc_norm, Clark et al.)
- WinoGrande (accuracy, Sakaguchi et al.)

We report accuracy for WinoGrande, PIQA, and ARC-Easy, and accuracy normalized by sequence length (acc_norm) for HellaSwag and ARC-Challenge.

H.2 LONGBENCH EVALUATION DETAILS

For the LongBench evaluation in the manuscript, we selected eight tasks from four subgroups, ensuring a comprehensive evaluation of *Palu*. The tasks and their corresponding metrics are detailed below:

- Single-Document QA:
 - Qasper (F1 score, Dasigi et al.)
- Summarization:
 - QMSum (ROUGE score, Zhong et al.)
 - MultiNews (ROUGE score, Fabbri et al.)
- Few-shot Learning:
 - TREC (classification score, Li & Roth)
 - TriviaQA (F1 score, Joshi et al.)
 - SAMSum (ROUGE score, Gliwa et al.)
- Code Completion:

- LCC (similarity score, Guo et al.)
- RepoBench-P (similarity score, Liu et al.)

During the evaluation, we set the maximum sequence length to 31500 for both the Mistral and LongChat model.