A Simple and Effective L_2 Norm-Based Strategy for KV Cache Compression

Alessio Devoto[‡*] **Yu Zhao**[†*] **Simone Scardapane**[‡] **Pasquale Minervini**[†]
[‡]Sapienza University of Rome [†]The University of Edinburgh
{yu.zhao, p.minervini}@ed.ac.uk
{alessio.devoto, simone.scardapane}@uniroma1.it

Abstract

The deployment of large language models (LLMs) is often hindered by the extensive memory requirements of the Key-Value (KV) cache, especially as context lengths increase. Existing approaches to reduce the KV Cache size involve either fine-tuning the model to learn a compression strategy or leveraging attention scores to reduce the sequence length. We analyse the attention distributions in decoder-only Transformers-based models and observe that attention allocation patterns stay consistent across most layers. Surprisingly, we find a clear correlation between the L_2 norm and the attention scores over cached KV pairs, where a low L_2 norm of a key embedding usually leads to a high attention score during decoding. This finding indicates that *the influence of a KV pair is potentially determined by the key embedding itself before being queried*. Based on this observation, we compress the KV Cache based on the L_2 norm of key embeddings. Our experimental results show that this simple strategy can reduce the KV Cache size by 50% on language modelling and needle-in-a-haystack tasks and 90% on passkey retrieval tasks without losing accuracy. Moreover, without relying on the attention scores, this approach remains compatible with FlashAttention, enabling broader applicability.

1 Introduction

Handling long contexts is desirable for large language models (LLMs), as it allows them to perform tasks that require understanding long-term dependencies Liu et al. [2024], Fu et al. [2024], Chen et al. [2023], Staniszewski et al. [2023], Zhao et al. [2024], Tworkowski et al. [2024]. A key component for modelling long context is the KV Cache, which stores the keys and values of past tokens in memory to avoid recomputing them during generation. However, processing long-context inputs often results in a high decoding latency since it requires repeatedly reading a potentially large KV Cache from high-bandwidth memory (HBM) to the streaming multiprocessor (SM) during decoding [Fu, 2024]. Consequently, the practical deployment of LLMs is frequently hindered by hardware limitations. To address the issue of KV Cache growth, various KV Cache compression methods have been proposed. These methods can be broadly categorised into trainable approaches, which involve modifications to the model architecture Ainslie et al. [2023], or fine-tuning regime to inherently manage KV Cache size Nawrot et al. [2024], and non-trainable approaches, which apply post-hoc compression techniques to reduce the cache footprint without altering the underlying model Li et al. [2024], Zhang et al. [2024b], Ge et al. [2023b]. While these methods have shown promise, they often involve complex algorithms or significant computational overhead, limiting their practicality; for example, post-hoc compression algorithms usually evict KV pairs based on attention scores, which is not compatible with FlashAttention [Dao et al., 2022] and thus prevents their applications in modern LLMs inference systems.

4th NeurIPS Efficient Natural Language and Speech Processing Workshop (ENLSP-IV 2024).

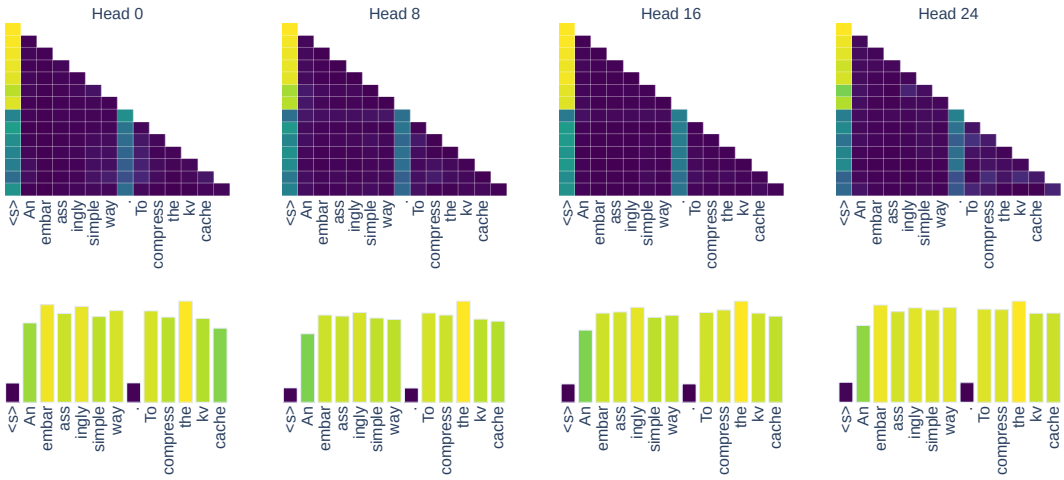


Figure 1: Five heads at layer 9 of Llama2-7b. Attention score (top) and L_2 norm (bottom) are highly correlated. We observe similar patterns across most layers and for a wide range of inputs. More examples provided in Appendix C

We show that, surprisingly, the L_2 norm of cached keys has a high correlation with attention scores. More specifically, we observe that a low L_2 norm of a key embedding usually leads to a high attention score during decoding. Based on this observation, we propose a simple and highly effective strategy for KV Cache compression: *keeping in memory only the keys with lowest L_2 norm, and the corresponding values*. Unlike many existing methods, our heuristic can be applied off-the-shelf to any transformer-based decoder-only LLM without the need for additional training or significant modifications. More importantly, our method estimates the influence of cached key-value pairs without the need to compute the attention scores. Therefore, unlike other compression methods [Holmes et al., 2024, Li et al., 2024], it can be easily integrated with the popular FlashAttention [Dao et al., 2022].

Our experimental results demonstrate that this heuristic allows maintaining model performance in language modelling tasks and in tasks that require the model to store and retrieve the most critical information, such as passkey retrieval [Mohtashami and Jaggi, 2023] and needle-in-a-haystack tasks [Kamradt, 2023].

2 Background on LLM Inference

In transformer-based LLMs, the input sequence is represented as a tensor $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the token embedding dimension. Each x_i corresponds to an embedding of a token in the sequence. The tensor $\mathbf{X}$ is processed by a series of transformer blocks, each composed of a multi-head self-attention and a feed-forward layer.

Given an input $\mathbf{X} \in \mathbb{R}^{n \times d}$, the multi-head attention mechanism performs multiple attention operations in parallel, allowing the model to attend to information from different representation subspaces. It does so by first computing three projections: the query, key, and value matrices, denoted as $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively. These are obtained by linear transformations of the input $\mathbf{X}$:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V, \tag{1}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$ are learned projection matrices, and d_k is the dimensionality of the queries and keys. Next, the output is computed using the scaled dot-product attention. The attention output is calculated as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{2}$$

In the multi-head attention mechanism, this process is repeated h times, each with different learned projections $\mathbf{W}_Q^{(i)}, \mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)}$ for each head h, resulting in H separate attention outputs. These

outputs are concatenated and projected back to the original dimension d using a final learned matrix $\mathbf{W}_O \in \mathbb{R}^{hd_k \times d}$:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\mathbf{W}_O \tag{3}$$

where each attention head head_h is defined as $\text{head}_h = \text{Attention}(\mathbf{Q}^{(h)}, \mathbf{K}^{(h)}, \mathbf{V}^{(h)})$.

KV Cache During autoregressive inference, where tokens are generated sequentially, the model has to compute the attention distributions over all previously generated tokens at each step. Without optimisations, this would involve recalculating the key ($\mathbf{K}$) and value ($\mathbf{V}$) projections for every past token at each new step. The KV Cache addresses this inefficiency by storing the key and value projections for each token after they are first computed. Instead of recalculating these projections for past tokens, the model retrieves the cached $\mathbf{K}$ and $\mathbf{V}$ values during subsequent inference steps.

When generating a new token at time step t, the attention computation is performed as:

$$\text{Attention}(\mathbf{Q}_t, [\mathbf{K}_{1:t-1}; \mathbf{K}_t], [\mathbf{V}_{1:t-1}; \mathbf{V}_t]) \tag{4}$$

where $[;]$ denotes concatenation along the sequence dimension, and $\mathbf{K}_{1:t-1}$ and $\mathbf{V}_{1:t-1}$ are retrieved from memory. The key $\mathbf{K}_t$ and value $\mathbf{V}_t$ for the current token are computed normally.

The KV Cache can significantly reduce computational costs by avoiding redundant calculations. However, storing the cached key and value matrices for every token in the sequence incurs substantial memory usage, which grows linearly with the sequence length. For a model with L layers, H attention heads, and a sequence length of n, the total memory required is $L \times H \times n \times d_k \times 2\times$, where the factor of 2 accounts for both the key and value matrices and *precision* represents the number of bytes used to store each value in the memory, typically corresponding to the bit-width of the data type (e.g., 16 bits for half-precision or 32 bits for single-precision floating point).

Though the KV Cache improves the computational efficiency, it requires repeatedly reading potentially large KV Cache from high-bandwidth memory to the streaming multiprocessor during decoding. To address this, recent works [Zhang et al., 2024b, Ge et al., 2023a, Li et al., 2024, Luohe et al., 2024] have proposed compressing the KV Cache to reduce memory usage.

3 Analysis of the Attention Distributions

We first examine the attention scores on the language modelling task for a range of popular LLMs. By analysing the key embeddings and the attention distribution, we observe that key embeddings with low L_2 norm are often associated with higher attention scores. In Figure 1, we provide an example using Llama-2-7b [Touvron et al., 2023], where the columns represent different heads, the first row presents the attention distribution over the KV pairs, and the second row presents the L_2 norm of each key embedding. We observe that the tokens with high attention scores, such as `"<s>"` and `"."`, have significantly lower L_2 norm values than others. While Xiao et al. [2024] already observed peaked attention distributions for specific tokens, and Darcet et al. [2024] pointed out the influence of high L_2 norm hidden states on attention maps, we are the first, to the best of our knowledge, to point out the correlation between the L_2 norm of the *key embeddings* and attention score. Based on our observation, we consider the following research question: can we compress the KV Cache based on the L_2 norm of the key embeddings?

An intuitive way to estimate the influence of compressing the KV Cache is by examining the attention scores that are dropped due to the compression. In the following, we formally define this influence.

Given a prompt consisting of n tokens $(x_1, x_2, ..., x_n)$, the LLM first encodes them into a KV Cache—this step is referred to as the *pre-filling phase*. Then, the model autoregressively generates the next token x_{n+1}. When performing KV Cache compression, some key-value pairs may be dropped and thus cannot be attended to. We define the attention loss caused by the compression as the sum of the attention scores associated with the dropped KV pairs:

$$\mathcal{L}_{l,h}^{m} = \sum_{p \in D_{l,h}} a_{l,h,p}, \tag{5}$$

where $a_{l,h,p}$ is the attention score of the p-th token in the layer l, head h. In Equation (5), $D_{l,h}$ denotes the positions of m pairs of dropped KV, $|D_{l,h}| = m$, which depends on the compression

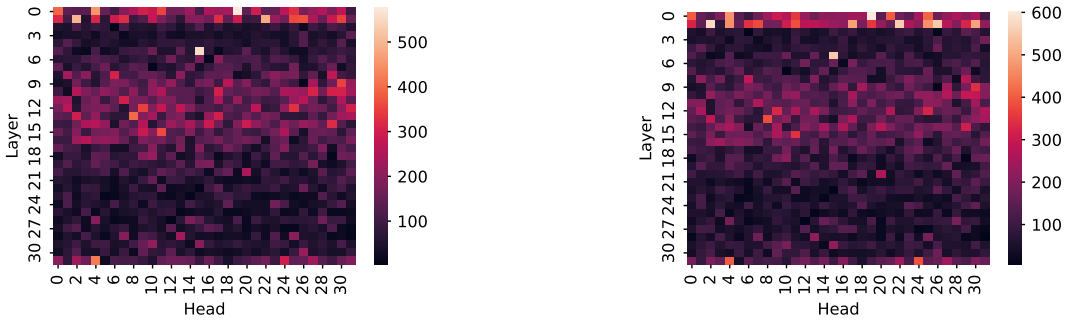


Figure 2: ALR, as defined in Equation (7), for each head and layer in Llama2-7b (left) and Llama2-7b-32k long context model (right). A lower value means a higher correlation between L_2 norm and attention score.

method. An ideal compression algorithm aims to drop the KV pairs with the lowest attention scores, which will have less impact on the output. However, such attention scores are unavailable for a compression algorithm since it needs x_{n+1} to query the full KV Cache in advance. Instead, we drop KV pairs with the highest L_2 norm in key embeddings and use attention loss caused by ideal compression as the reference:

$$\mathcal{Y}^m_{l,h} = \mathcal{L}^m_{l,h} - \mathcal{L}^{m,ref}_{l,h}, \tag{6}$$

where $\mathcal{L}^{m,ref}_{l,h}$ is the reference attention loss, and $\mathcal{Y}^m_{l,h}$ is a non-negative value. A lower $\mathcal{Y}^m_{l,h}$ indicates a lower difference and thus a higher correlation between the attention score and the L_2 norm. To measure the overall difference between ideal attention score-based compression and L_2 norm-based compression, we sum up the $\mathcal{Y}^m_{l,h}$ over different numbers of compressed KV pairs:

$$\mathcal{Y}_{l,h} = \sum_{m=1}^{n} \mathcal{Y}^m_{l,h}. \tag{7}$$

We name the $\mathcal{Y}_{l,h}$ as ALR, which denotes the attention loss (Equation (5)) for a compression method using the ideal attention loss as reference. In Figure 2, we plot the $\mathcal{Y}$ across layers and heads. We observe that heads in the first two layers and some middle layers around the 12th layer have relatively high $\mathcal{Y}$ values. The heads in other layers have lower $\mathcal{Y}$ values, indicating a high correlation between L_2 norm and attention score.

By leveraging this correlation, we can compress the KV Cache based on the L_2 norm of key embeddings. Optionally, we can skip the compression at the layers with low correlation. We show ablation experiments skipping layers in Appendix A.

4 Experiments

We evaluate our method on language modelling and two long-context modelling tasks, i.e., needle-in-a-haystack and passkey retrieval. In addition, we test on tasks from LongBench [Zhang et al., 2024a], specifically devised to evaluate the model's long context abilities. Based on the observation supported by Figure 2, the heads in the first two layers usually have a low correlation between L_2 norm and attention score, so we do not perform compression on these layers as default. We conduct experiments to investigate the impact of compression on different layers in Appendix A.

Language Modelling Tasks For language modelling, we let the KV Cache grow until a specific pre-defined length and subsequently start to discard the tokens with the highest L_2 norm. We show in Figure 3 that evicting even up to the 50% of KV Cache does not impact perplexity. Perplexity increases, as expected, once we exceed the pre-training context length. We show more results, including next token accuracy in Appendix A. To further verify that keys with low L_2 norm capture significant information, we test other eviction strategies, i.e. keeping tokens with highest L_2 norm and keeping random tokens. It is clear from Figure 3 that discarding tokens with low L_2 impairs performance, even more so than random discarding, thus highlighting the importance of these low L_2 norm keys.

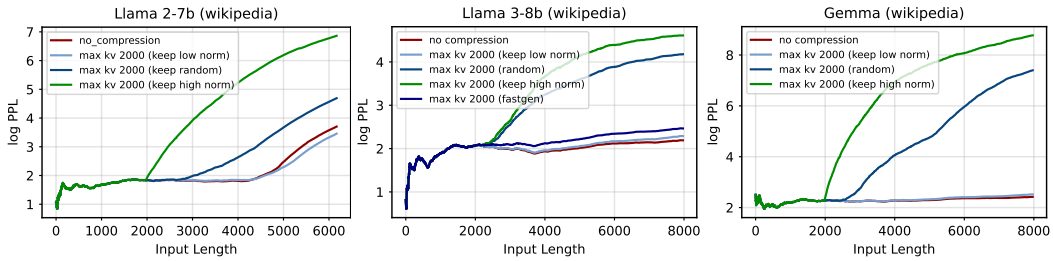


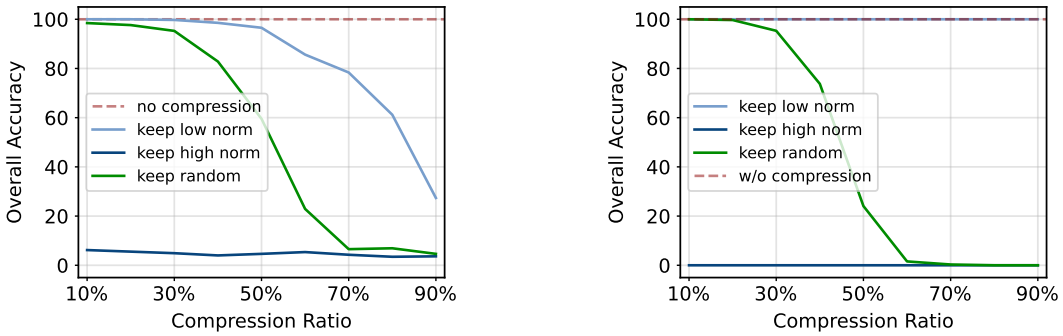
Figure 3: Perplexity for Llama 2-7b, Llama 3-8b and Gemma on language modelling task on wikipedia dataset.Additional results on coding dataset are available in Appendix A

Pressure Test on Long-Context Tasks The needle-in-a-haystack task [Kamradt, 2023] and passkey retrieval task [Mohtashami and Jaggi, 2023] are two synthetic tasks that are widely used to pressure test the long-context modelling capability of LLMs. In both tasks, the model needs to identify and retrieve the important information from a long context to generate correct answers. Thus, these tasks test the compression method's ability to keep important KV pairs and drop redundant ones.

In Figure 4a and Figure 4b, we present the experimental results of Llama-2-7b-80k [Fu et al., 2024]. We analyse additional models in Appendix B. As shown in Figure 4a, the model can preserve its performance on the needle-in-a-haystack task while compressing 30% of the KV Cache, and maintain 99% accuracy when compressing 50% of the KV Cache. Additionally, the model can achieve 100% accuracy on the passkey retrieval task even when compressing 90% of the KV Cache, as shown in Figure 4b.

Moreover, we compare other eviction strategies, like keeping KV pairs with low L_2 norm, keeping KV pairs with high L_2 norm, and keeping random KV pairs. In Figure 4a and Figure 4b, we observe that the model cannot answer correctly when keeping only high L_2 norm KV pairs, obtaining near zero and zero accuracy on the needle-in-a-haystack and passkey retrieval tasks, respectively. When we randomly compress the KV Cache, the performance decreases significantly faster than keeping low L_2 norm KV pairs. The above analysis indicates that KV pairs with low L_2 norm are critical to generating the correct answer and thus contain important information.

Experiments on LongBench Additionally, we evaluate on LongBench [Zhang et al., 2024a]. We test on several subsets, including NarrativeQA [Kociský et al., 2018], Qasper [Dasigi et al., 2021], HotpotQA [Yang et al., 2018], 2WikiMQA [Ho et al., 2020], and QMSum [Zhong et al., 2021]. We report the results for the recently released long context Llama3.1 and Llama 2-7b 80k in Figure 5. In addition, we show the complete per-subset results in Appendix B. The experimental results show that compressing the KV Cache with low L_2 norm only introduces a small accuracy decrease even when compressing 50% KV Cache, while compressing KV Cache with high L_2 norm results in almost zero accuracy.



(a) Accuracy on the needle-in-a-haystack task.

(b) Accuracy on the passkey retrieval task.

Figure 4: Overall accuracy of llama-2-7b-80k on the needle-in-a-haystack task passkey retrieval task.

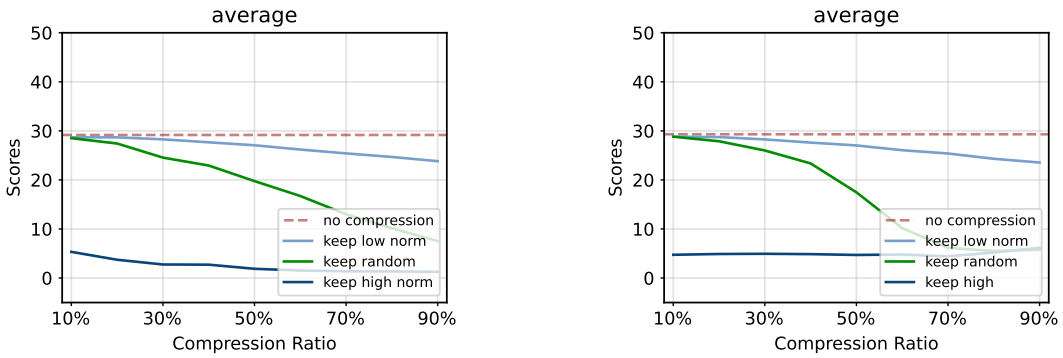


Figure 5: Overall scores on LongBench [Zhang et al., 2024a] of Llama3.1-8b (left) and llama-2-7b-80k (right) for different compression ratios ranging from 0% to 90%.

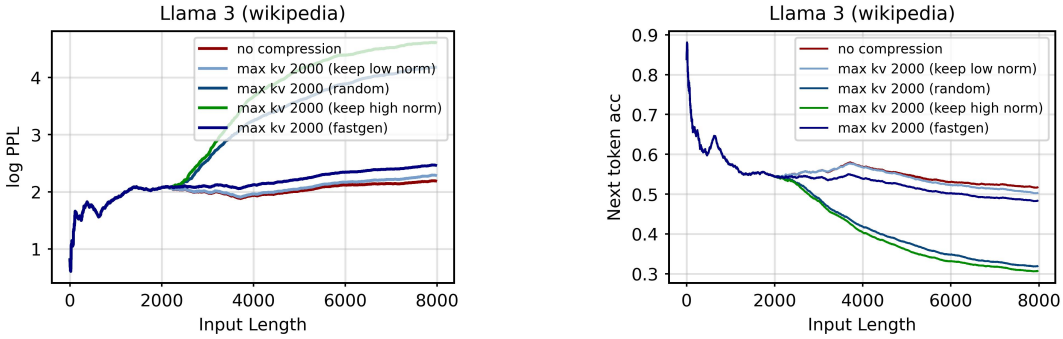


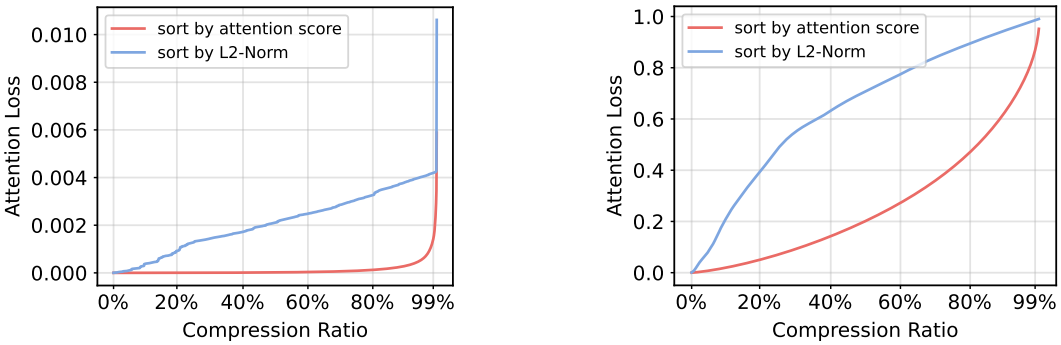
Figure 6: Perplexity and next token accuracy of Llama3-8b on the wikipedia dataset when compared to FastGen [Ge et al., 2023a] (only local, special and punctuation tokens).

Comparison with FastGen We use FastGen [Ge et al., 2023a], a popular method for KV Cache compression, as a baseline for assessing the effectiveness of our method. It is important to note that, like the majority of methods in the literature, FastGen utilises attention scores, which makes it incompatible with the popular FlashAttention [Dao et al., 2022], thereby limiting its efficiency and usability. For a fair comparison, we implement FastGen without using the attention scores, i.e., we only consider local, punctuation and special tokens. We perform experiments on language modelling with the Llama3 model [Dubey et al., 2024]. Our method still outperforms FastGen with up to 50% KV Cache eviction. We show the results in Figure 6.

5 Analysis

Attention score loss when using L_2 norm We discuss further the correlation between L_2 norm and attention scores. We already displayed in Figure 2 the L_2 norm and attention correlation across heads and layers using the original Llama2-7b and the long context Llama2-7b-32k and Llama2-7b-80k. We can see that patterns are quite consistent across all the models. To better visualise how correlation varies across different heads, in Figure 7, we only consider two heads from layer 10 and layer 0 and show the ALR from Equation (5). As expected, we see that in layer 0, the difference is larger due to a lower correlation.

Relationship between embedding and L_2 norm So far, we have identified a correlation between the L_2 norm of token key embeddings and the corresponding attention scores. This observation, while primarily empirical, it offers a direction for further explorations. Our investigation into the distribution of key embeddings revealed that tokens with lower L_2 norm tend to exhibit *sparse activations* with only a few dimensions showing significantly high values, while the majority of the dimensions remain near zero. This pattern suggests that the embeddings of these tokens are not fully utilising the available vector space, focusing their activations on a narrow subset of dimensions.



(a) Layer-7 Head-10, high correlation between attention score and L_2-Norm.

(b) Layer-0 Head-0, low correlation between attention score and L_2-Norm.

Figure 7: Attention loss of ideal compression and L_2 norm-based compression in Llama-2-7b-80k. The x-axis represents the compression ratio; the y-axis represents the attention loss (defined by Equation (5)) The results average over 1024 chunks on Wikipedia, with a length of 1024.

Figure 8 illustrates several examples of such tokens, highlighting the difference between tokens with high and low L_2 norm.

Interestingly, this sparsity aligns with the concept of "sink" tokens, as identified in previous studies [Xiao et al., 2024]. These tokens capture a direction in the embedding space such that many queries align closely with it, leading to increased attention scores for these tokens. In simpler terms, when the key embeddings of certain tokens are dominated by a small number of dimensions, it appears to create a situation where a large number of queries - regardless of their specific content - are naturally drawn to these tokens, increasing their attention weight. We hypothesise that the lower L_2 norm reflects a partial use of the available embedding space, leading to increased attention for these tokens. To test this, we zeroed out the dimensions responsible for the peaked activations in low-norm key embeddings and observed significant changes in attention maps (Figure 9). Randomly altering dimensions did not produce the same effect, suggesting that the specific active dimensions play a critical role in attention distribution. This finding suggests that the L_2 norm may serve as a proxy for the extent to which an embedding utilises the available vector space and, consequently, the degree to which it influences attention. Lower L_2 norm appears to correspond to embeddings that drive disproportionately high attention values due to their alignment with a common "sink" direction.

6 Related Work

Recently, various long-context LLMs, such as Gemini-Pro-1.5 [Reid et al., 2024], Claude-3 [Anthropic, 2024], and GPT4 [Achiam et al., 2023], have shown the promising capability to process

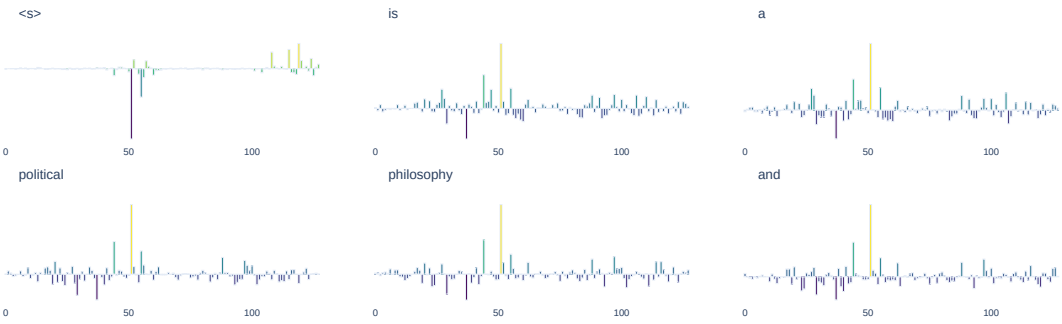


Figure 8: Key projections of the bos token $< s >$ vs other tokens. Each value represents the activation in a specific dimension for the embedding of the key projection. We found similar patterns across almost all heads and layers and in multiple texts. Only a few peaked activations (~ 50, ~ 56 and ~ 120) control the attention mechanism (see Figure 9). More plots like this in Appendix D

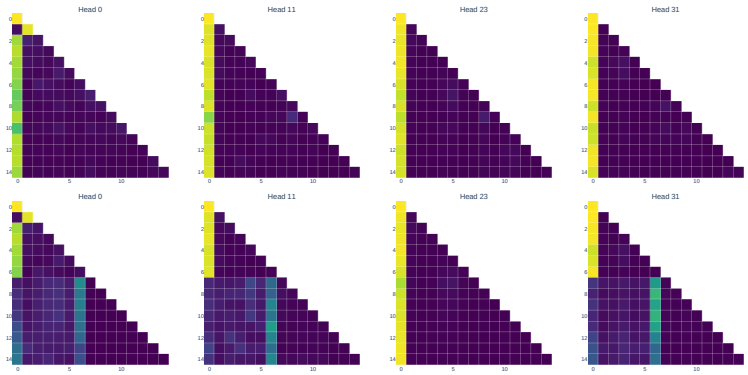


Figure 9: How the attention maps change if we set to zero a random activation (top) vs the specific peaked activations in the keys (bottom). We are setting the values at iteration 5.

hundred thousands of tokens in the context. The increased number of input lengths results in a high decoding latency; thus, there has been a growing interest in speeding up the decoding with long contexts. Some works propose efficient memory management strategies to reduce the IO time overheads, e.g., PageAttention [Kwon et al., 2023], Infinite-LLM [Lin et al., 2024] and vAttention [Prabhu et al., 2024]. Another line of research focuses on compressing the KV Cache to improve efficiency. DMC [Nawrot et al., 2024] compresses KV Cache by dynamically merging tokens while requiring expensive continual pre-training. For fine-tuning free compression strategy, H2O [Zhang et al., 2024b] identifies important KV pairs by leveraging the attention scores from all queries, FastGen [Ge et al., 2023a] leverages the different attention patterns in different heads for compression, and SnapKV [Li et al., 2024] selects KV pairs based on attention scores from user's query. Unlike these works, our method only utilises the L_2 norm of embedding for compression *without leveraging the attention information*, and to the best of our knowledge, we are the first to find that the influence of a KV pair can be determined by L_2 norm. Previous work [Darcet et al., 2024] finds the hidden states with high L_2 norm usually aggregate more important and global information. On the other hand, our findings indicate that a low L_2 norm of key embedding generally results in a high attention score. Concurrently to this work, Guo et al. [2024] uses the L_1 norm of values in the KV Cache and attention scores for compression.

7 Conclusions

In this paper, we introduced a simple yet highly effective strategy for KV Cache compression in LLMs based on the L_2 norm of key embeddings. We show that there is a significant correlation between the L_2 norm of a key embedding and its attention score. Leveraging this observation, we compress the KV Cache by retaining only those keys with the lowest L_2 norm. Our experimental results on various tasks show that our compression strategy maintains the predictive accuracy of the model while significantly reducing the memory footprint. Our approach is straightforward and can be applied directly to any transformer-based, decoder-only LLM.

8 Limitations

While our research offers valuable insights, we tested only on relatively small models (Llama family and Gemma up to 8 billion parameters). In future work, we will assess our method on larger-scale models to ensure our findings generalize Additionally, while we show that the L_2 norm played a significant role in our experiments, we do not have a comprehensive theoretical explanation for why this is the case. Understanding the underlying reasons behind the importance of the L_2 norm would require further theoretical exploration and empirical validation. Finally, we observed (Figure 2) that compressing based on L_2 norm can be less effective depending on the layer and head considered, and we intend to investigate per-head compression ratios to leverage this observation.

References

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

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=hmOwOZWzYE.

AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 2024.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=2dnO3LLiJ1.

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-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4599–4610. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.365. URL https://doi.org/10.18653/v1/2021.naacl-main.365.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yao Fu. Challenges in deploying long-context transformers: A theoretical peak performance analysis. *arXiv preprint arXiv:2405.08944*, 2024.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*, 2024.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023a.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@NeurIPS 2023)*, 2023b. URL https://openreview.net/forum?id=e9D2STGwLJ.

Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. Attention score is not all you need for token importance indicator in kv cache reduction: Value also matters, 2024. URL https://arxiv.org/abs/2406.12335.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/V1/2020.COLING-MAIN.580. URL https://doi.org/10.18653/v1/2020.coling-main.580.

Connor Holmes, Masahiro Tanaka, Michael Wyatt, Ammar Ahmad Awan, Jeff Rasley, Samyam Rajbhandari, Reza Yazdani Aminabadi, Heyang Qin, Arash Bakhtiari, Lev Kurilenko, et al. Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference. *arXiv preprint arXiv:2401.08671*, 2024.

Greg Kamradt. Needle in a haystack - pressure testing llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2023.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328, 2018. doi: 10.1162/TACL_A_00023. URL https://doi.org/10.1162/tacl_a_00023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

Bin Lin, Tao Peng, Chen Zhang, Minmin Sun, Lanbo Li, Hanyu Zhao, Wencong Xiao, Qi Xu, Xiafei Qiu, Shen Li, et al. Infinite-llm: Efficient llm service for long context with distattention and distributed kvcache. *arXiv preprint arXiv:2401.02669*, 2024.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 11:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9.

Shi Luohe, Hongyi Zhang, Yao Yao, Zuchao Li, and hai zhao. Keep the cost down: A review on methods to optimize LLM's KV-cache consumption. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=8tKjqqMM5z.

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=tDRYrAkOB7.

Ramya Prabhu, Ajay Nayak, Jayashree Mohan, Ramachandran Ramjee, and Ashish Panwar. vattention: Dynamic memory management for serving llms without pagedattention. *arXiv preprint arXiv:2405.04437*, 2024.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Konrad Staniszewski, Szymon Tworkowski, Yu Zhao, Sebastian Jaszczur, Henryk Michalewski, Lukasz Kuci'nski, and Piotr Milo's. Structured packing in llm training improves long context utilization. *ArXiv*, abs/2312.17296, 2023. URL https://api.semanticscholar.org/CorpusID:266690935.

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

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36, 2024.

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=NG7sS51zVF.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1259. URL https://doi.org/10.18653/v1/d18-1259.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.814.

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, 2024b.

Yu Zhao, Yuanbin Qu, Konrad Staniszewski, Szymon Tworkowski, Wei Liu, Piotr Miłoś, Yuxiang Wu, and Pasquale Minervini. Analysing the impact of sequence composition on language model pre-training. *arXiv preprint arXiv:2402.13991*, 2024.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir R. Radev. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5905–5921. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.472. URL https://doi.org/10.18653/v1/2021.naacl-main.472.

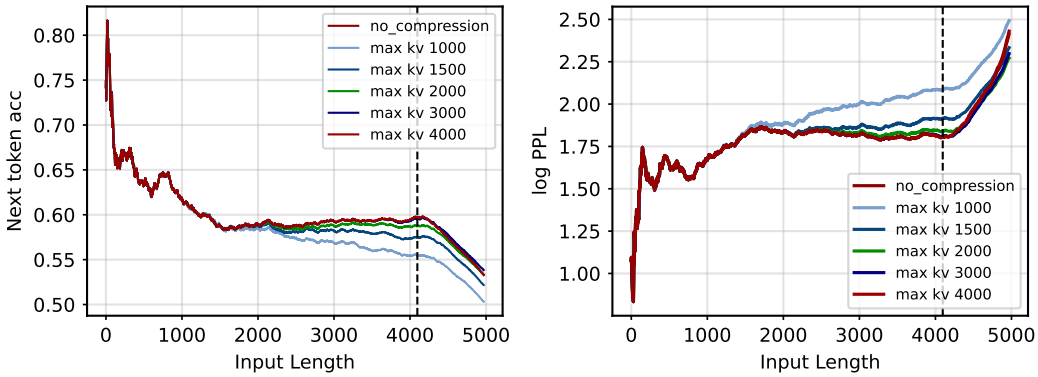


Figure 10: Results on language modelling task when skipping the first layer.

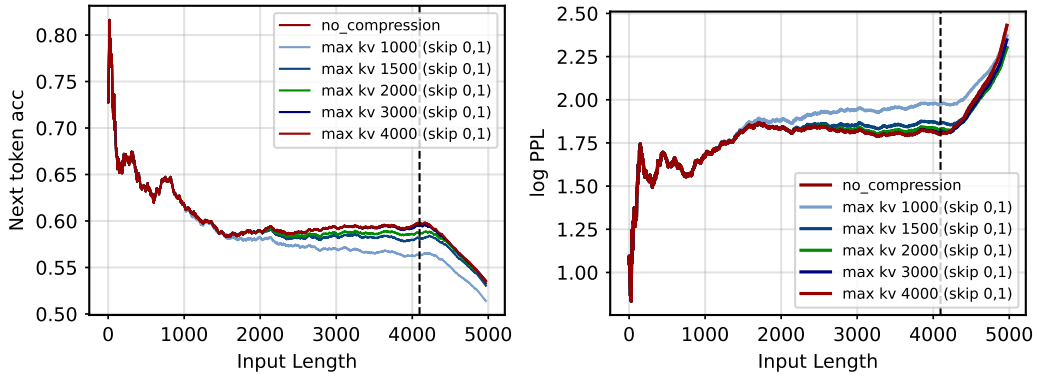


Figure 11: Results on language modelling task when skipping the first two layers.

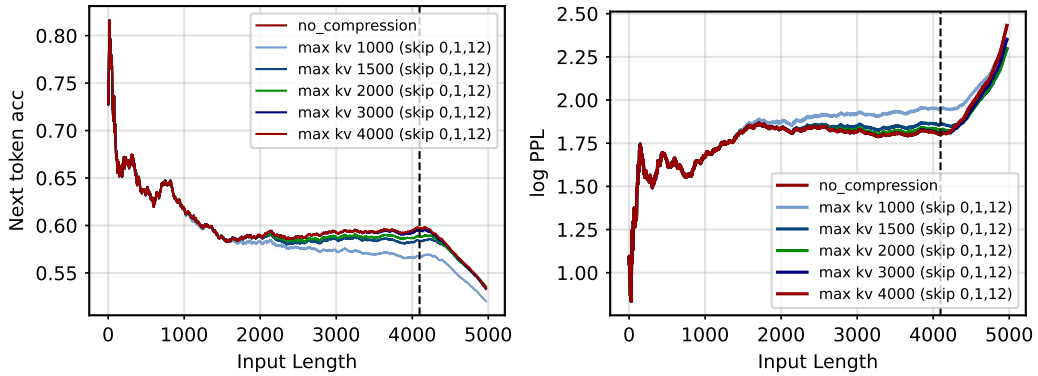
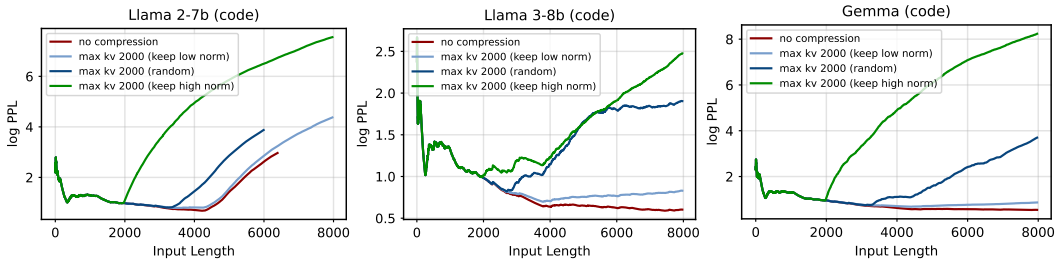


Figure 12: Results on language modelling task when skipping layers 0,1 and 12.

Figure 13: Skipping compression at different layers with Llama2-7b

A More results on Language modelling task

In the following, we show results when performing compression only on layers that show a lower correlation between L_2 norm and attention score. We show in Fig. 13 that for language modelling tasks, the different layer drop has little impact on final accuracy and perplexity. The difference becomes significant only when the KV Cache is pruned to retain only one thousand pairs. All experiments are averaged over 50 chunks from English Wikipedia.

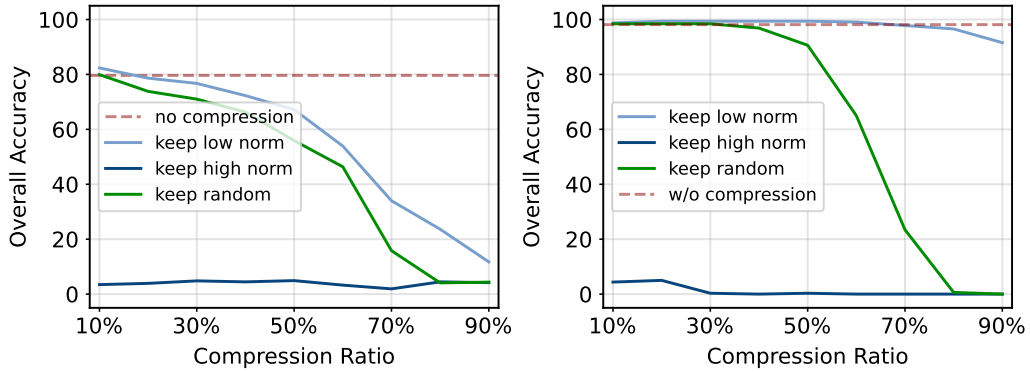


B More Results on Long-Context Modelling Tasks

In addition to llama-2-7b-80k [Fu et al., 2024], we test the compression method using llama-2-7b-longlora-32k-ft [Chen et al., 2023] on the needle-in-a-haystack and passkey retrieval tasks. As shown in Fig. 15a, we can see that compressing 30% of KV Cache only results in a slight performance degradation on the needle-in-a-haystack task. We also observe that the performance even increases slightly when we compress 10% of KV Cache. In figure Fig. 15b, we observe that the llama-2-7b-longlora-32k-ft maintains 100% performance when compressing 80% of KV Cache and only as a slight decrease when compressing 90% of KV Cache. Furthermore, the model fails to generate correct answers if we compress KV pairs with low L_2 norm and keep high L_2 norm ones. The evaluation results of llama-2-7b-longlora-32k-ft are consistent with the llama-2-7b-80k, which further indicates the effectiveness of compressing KV Cache using L_2 norm.

B.1 Analysis of Skipped Layers

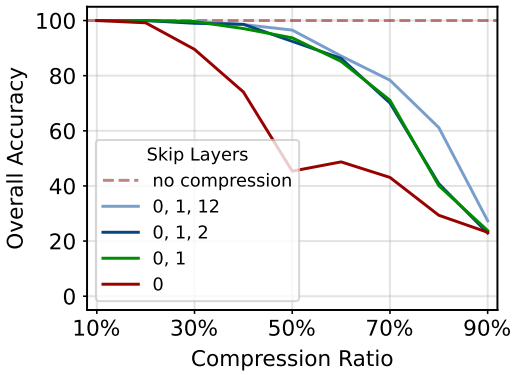
As shown in Fig. 2, we find heads in the first two layers and the middle layers have a relatively low correlation between attention scores and L_2 norm. Thus, we conduct experiments to analyse the impact of skipping layers that have a low correlation for compression. As shown in Fig. 16a and Fig. 16c, we observe that only skipping the first layer (layer-0) decreases the performance on the needle-in-a-haystack task significantly. We can see that skipping the first two layers (layer-0,1) has a similar performance compared to skipping the first three layers (layer-0,1,2). Furthermore, as shown in Fig. 16b and Fig. 16d, only skipping the first layer can result in significant performance degradation. We also find that the compression ratio is not proportional to the overall accuracy of models in the passkey retrieval task when we compress the first layer, where the accuracy shows a U-shape curve regarding the compression ratio.



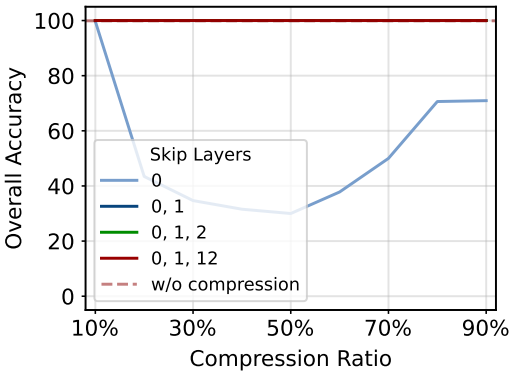
(a) Overall accuracy of Llama-2-7b-longlora-32k-ft on the needle-in-a-haystack task.

(b) Overall accuracy of Llama-2-7b-longlora-32k-ft on the passkey retrieval task.

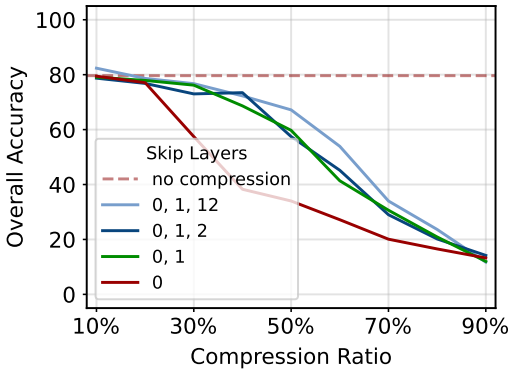
Figure 15: Evaluation results of Llama-2-7b-longlora-32k-ft on the needle-in-a-haystack and passkey retrieval tasks.



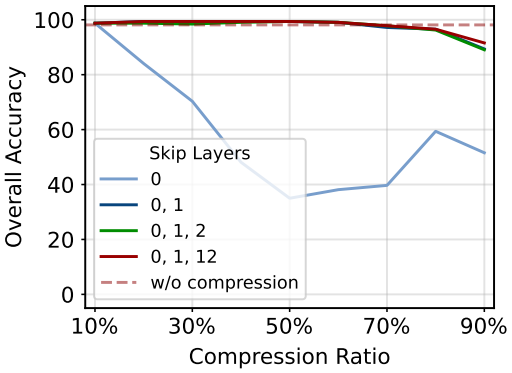
(a) Overall accuracy of Llama-2-7b-80k on the needle-in-a-haystack task.



(b) Overall accuracy of Llama-2-7b-80k on the passkey retrieval task.

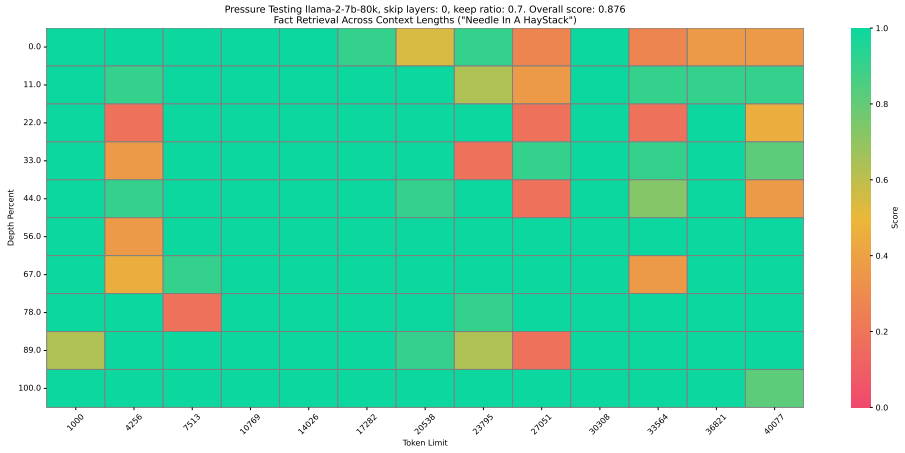


(c) Overall accuracy of Llama-2-7b-longlora-32k-ft on the needle-in-a-haystack task.

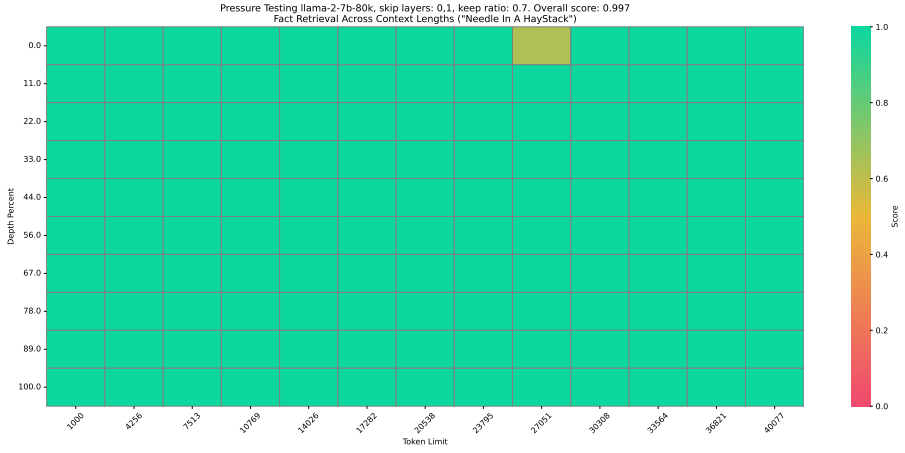


(d) Overall accuracy of Llama-2-7b-longlora-32k-ft on the passkey retrieval task.

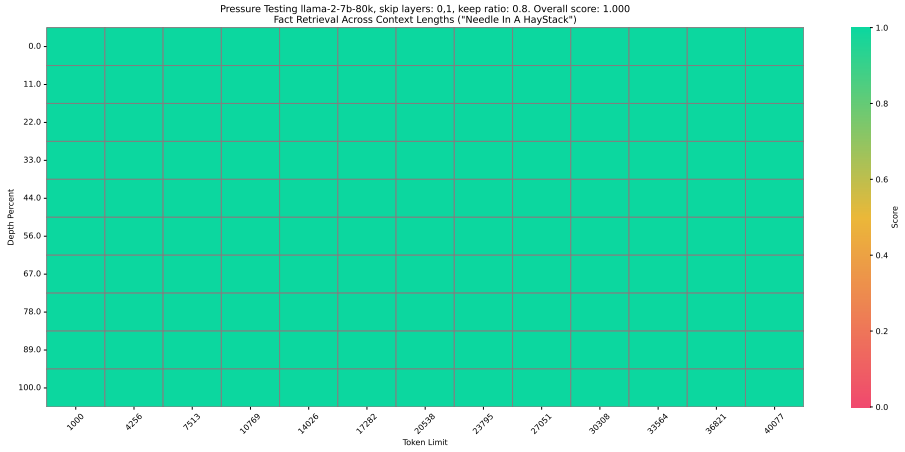
Figure 16: Analysing of skipping different layers for compression.



(a) Llama-2-7b-80k, skip layer-0, compression ratio 30%

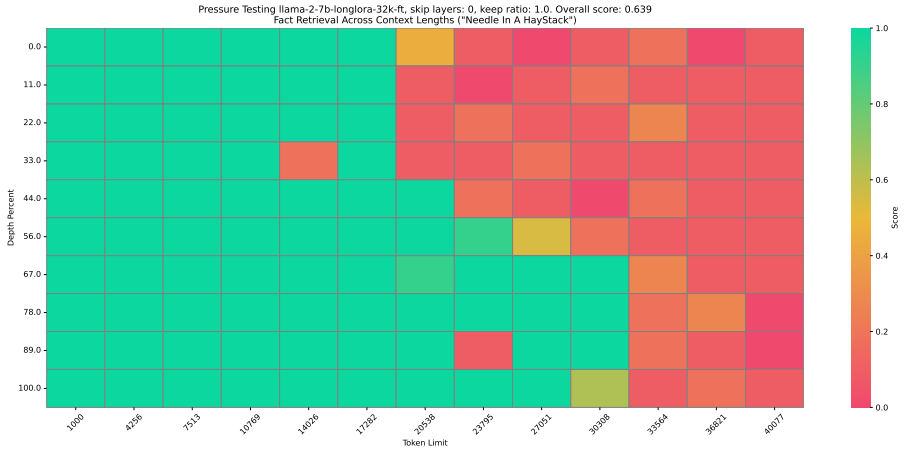


(b) Llama-2-7b-80k, skip layer-0 and layer-1, compression ratio 30%

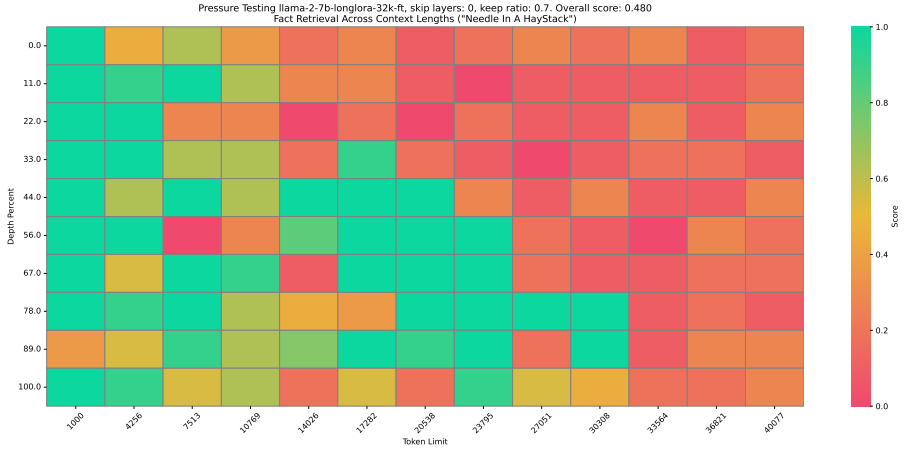


(c) Llama-2-7b-80k, skip layer-0 and layer-1, compression ratio 20%

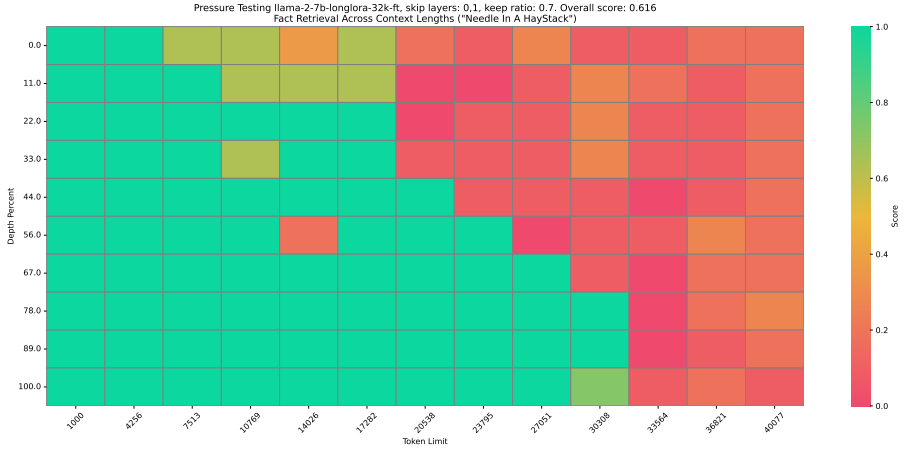
Figure 17: Detailed results of Llama-2-7b-80k on the needle-in-a-haystack task.



(a) Llama-2-7b-longlora-32k-ft, without compression

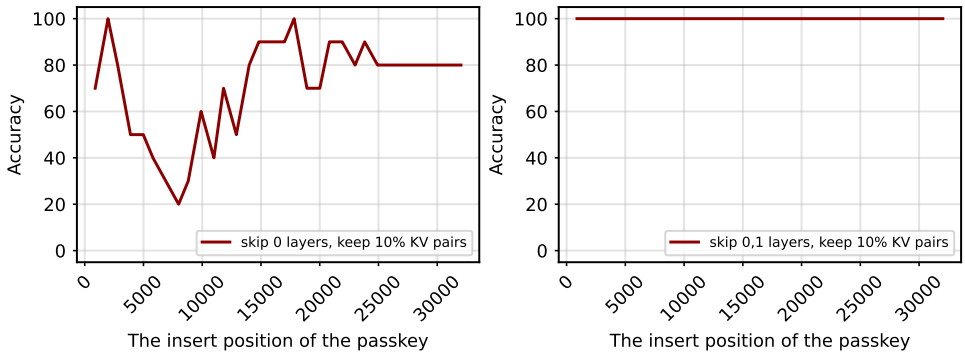


(b) Llama-2-7b-longlora-32k-ft, skip layer-0, compression ratio 30%



(c) Llama-2-7b-longlora-32k-ft, skip layer-0 and layer-1, compression ratio 30%

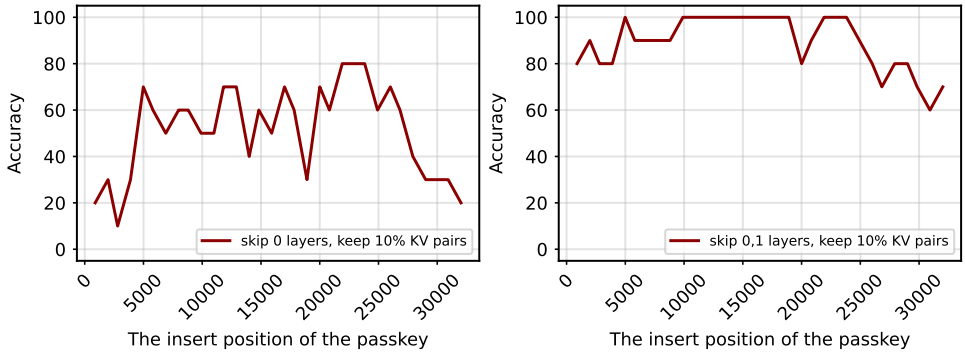
Figure 18: Detailed results of Llama-2-7b-longlora-32k-ft on the needle-in-a-haystack task.



(a) Llama-2-7b-80k, skip layer-0, compression ratio 90%

(b) Llama-2-7b-80k, skip layer-0 and layer-1, compression ratio 90%

Figure 19: Accuracy on the passkey retrieval. The x-axis presents the position of the passkey, and the y-axis presents the accuracy.



(a) Llama-2-7b-longlora-32k-ft, skip layer-0, compression ratio 90%

(b) Llama-2-7b-longlora-32k-ft, skip layer-0 and layer-1, compression ratio 90%

Figure 20: Accuracy on the passkey retrieval. The x-axis presents the position of the passkey, and the y-axis presents the accuracy.

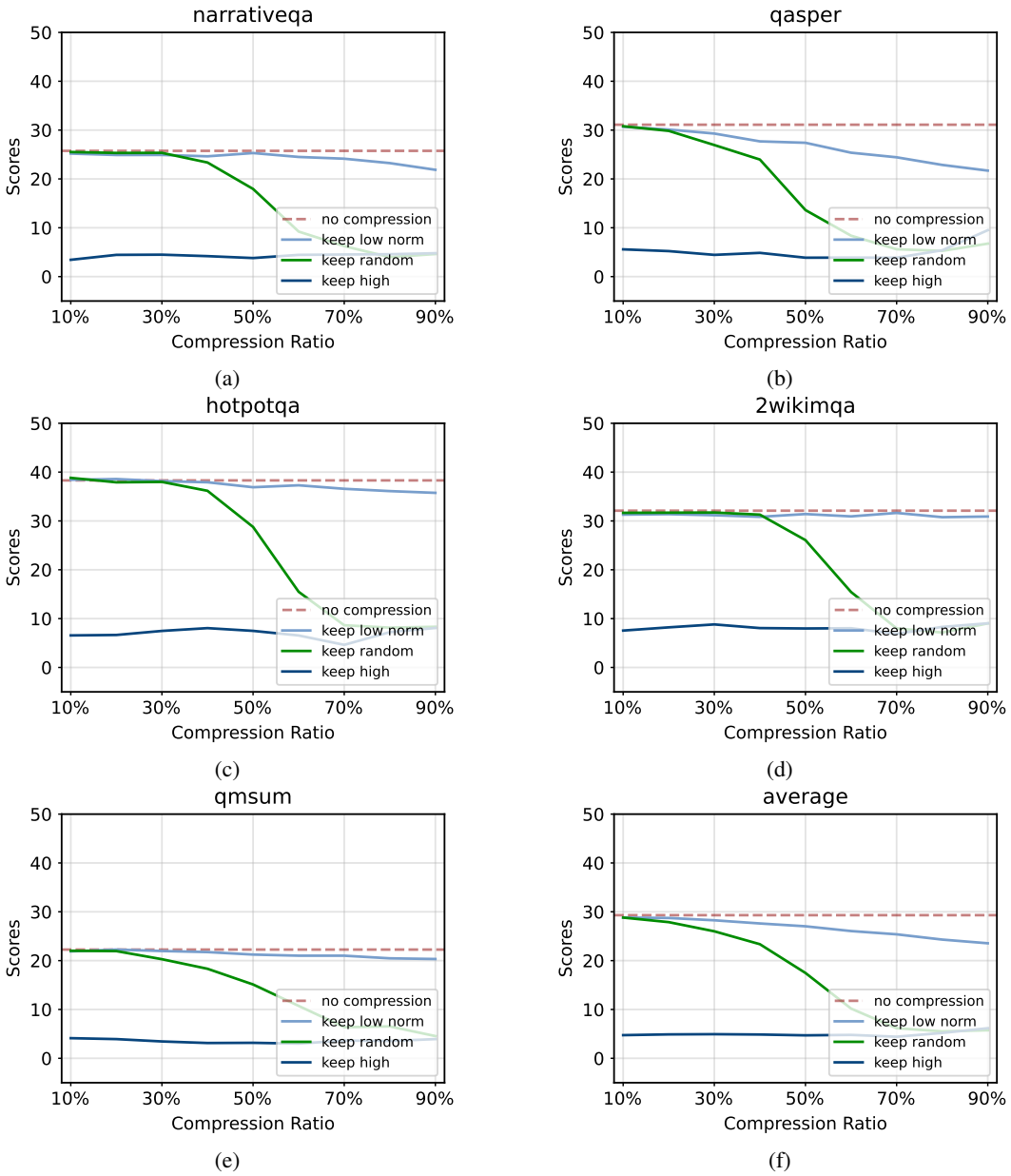


Figure 21: Evaluation results of Llama-2-7b-80k on long context tasks from Longbench, including narrativeqa and qasper, hotpotqa, 2wikimqa, and qmsum.

B.2 Longbench Evaluation

In this section we show detailed results from the LongBench dataset [Zhang et al., 2024a]. In Figure 21 we show results for Llama2-80k, while in Figure 22 we show results for the long context model Llama3.1-8b.

C More Visualizations

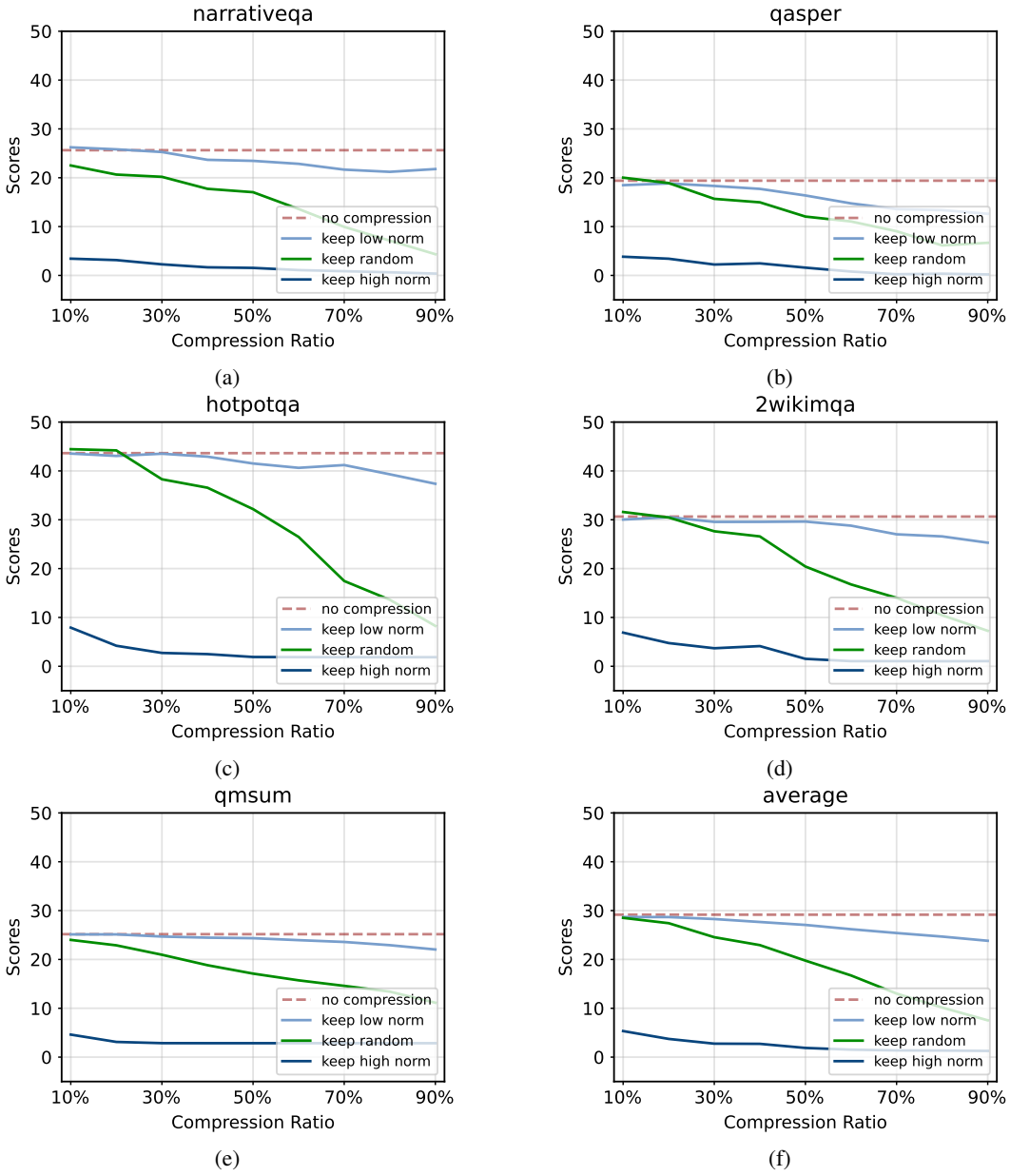


Figure 22: Evaluation results of Llama-3.1-8B on long context tasks from Longbench, including narrativeqa and qasper, hotpotqa, 2wikimqa, and qmsum.

Figure 23: Attention maps in Llama2-7B

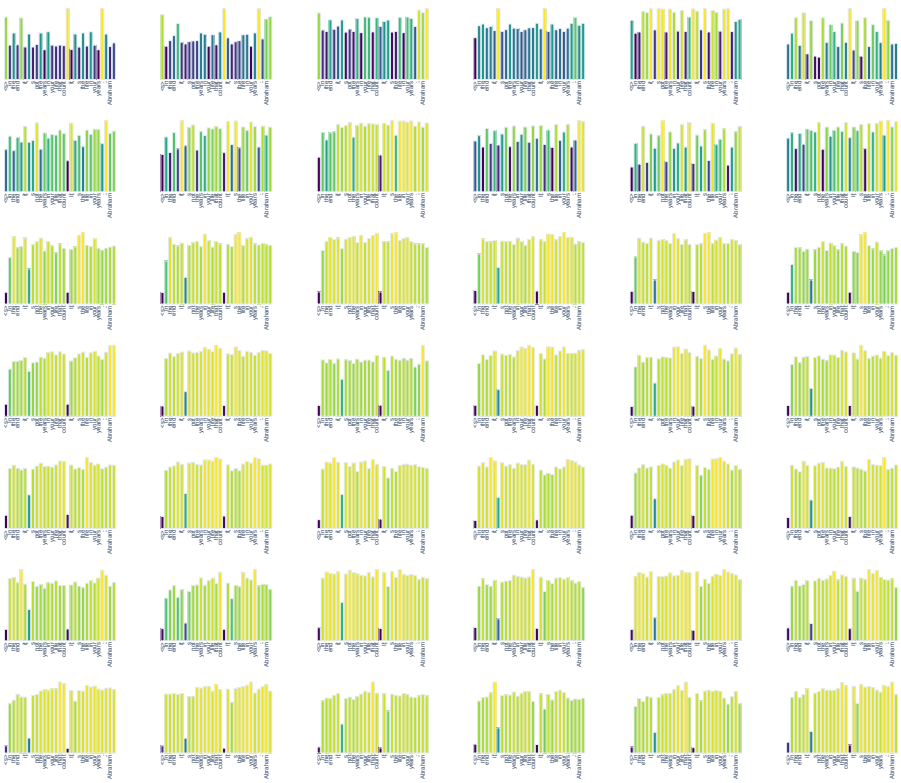


Figure 24: Norms of KV cache tokens in Llama2-7B

Figure 25: Attention maps in Llama2-7B

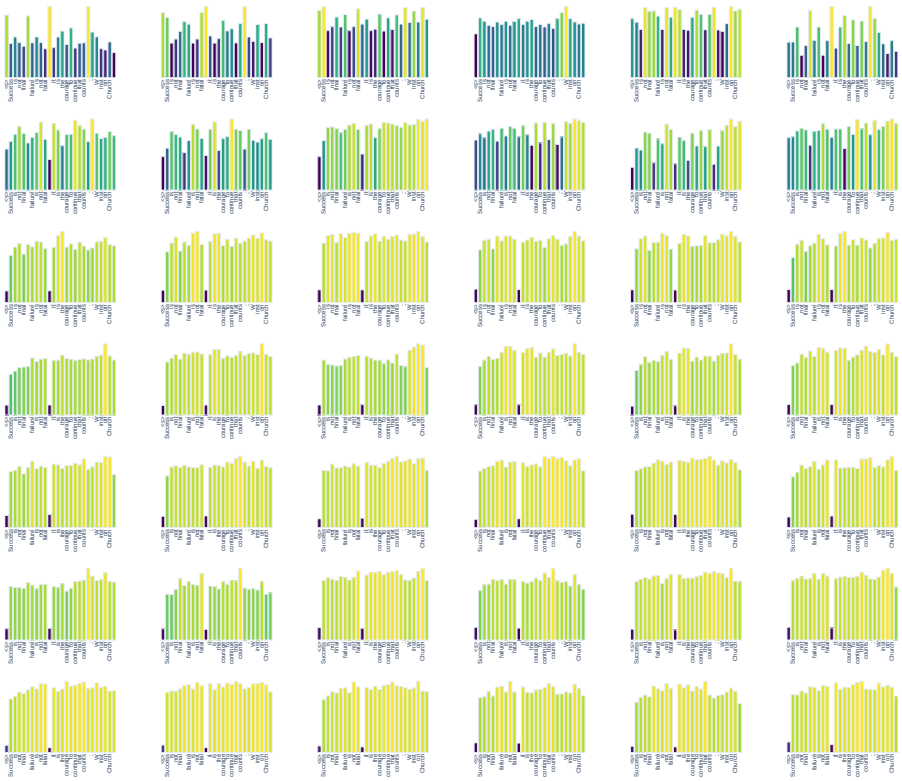


Figure 26: Norms of KV cache tokens in Llama2-7B

Figure 27: Attention maps in Llama2-7B

Figure 28: Norms of KV cache tokens in Llama2-7B

D Additional token embeddings plots

We show in Figure 29 some additional figure that represent Llama3-8b token embeddings sparsity.

E Experimental setup

In all experiments, we used the HuggingFace library and did not change the model's default hyperparameters. For language modelling, results are averaged across 50 samples. The Fig. 7 and Fig. 2 are the average results of 1024 examples with a chunk size of 1024 using Wikipedia.

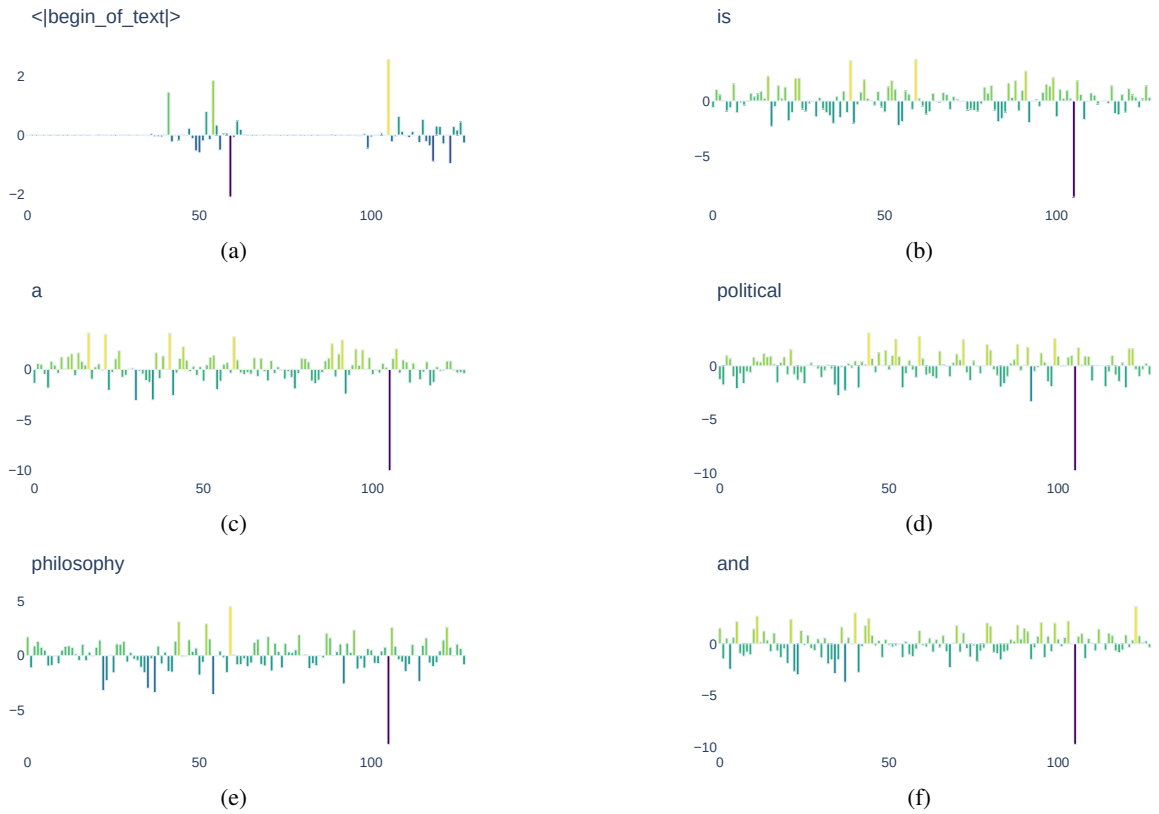


Figure 29: Key projections of Llama3-8b of the bos $|beginoftext|$ token vs other tokens. Each value represents the activation in a specific dimension for the embedding of the key projection. We found similar patterns across almost all heads and layers and in multiple texts.