ARKVALE: Efficient Generative LLM Inference with Recallable Key-Value Eviction

Renze Chen
Peking University
crz@pku.edu.cn

Zhuofeng Wang
Peking University
2200012827@stu.pku.edu.cn

Beiquan Cao
Peking University
2200012988@stu.pku.edu.cn

Tong Wu
Peking University
2200013212@stu.pku.edu.cn

Size Zheng
Peking University
zhengsz@pku.edu.cn

Xiuhong Li
Peking University
lixiuhong@pku.edu.cn

Xuechao Wei
Peking University
xuechao.wei@pku.edu.cn

Shengen Yan
Infinigence-AI
yanshengen@gmail.com

Meng Li
Peking University
meng.li@pku.edu.cn

Yun Liang[*]
Peking University
ericlyun@pku.edu.cn

Abstract

Large Language Models (LLMs) are widely used in today's tasks of natural language processing. To support applications like multi-turn chats, document understanding, and content generation, models with long context lengths are growing in importance. However, managing long contexts brings substantial challenges due to the expansion of key-value cache (KV cache). Longer KV cache requires larger memory, limiting the batch-size and thus decreasing throughput. Also, computing attention over long KV cache incurs more memory access, hurting the end-to-end latency. Prior works find that it is sufficient to use only the recent and high-impact tokens for attention computation, allowing the eviction of less vital tokens to reduce memory footprint. Nonetheless, we observe a dynamic shift in token importance across different decoding steps. Tokens initially evicted might regain importance after certain decoding steps. To address this, we propose ARKVALE, a page-based KV cache manager that can recognize and recall important tokens evicted before. We asynchronously copy the filled page into external memory (e.g., CPU memory) as backup and summarize/compress it into a much smaller digest by constructing the bounding-volume of the keys in the KV-page. Before attention computation, we measure all pages' importance based on their digests, recall the important ones, evict the unimportant ones, and select the top-ranked pages for attention computation. Experiment results show that ARKVALE performs well on various long context tasks with negligible accuracy loss under 2k$\sim$4k cache budget and can improve decoding latency up to $2.2\times$ ($1.7\times$ in average) and batching throughput up to $4.6\times$ ($3.5\times$ in average). Our code is now available at `https://github.com/pku-liang/ArkVale`.

[*]Corresponding author.

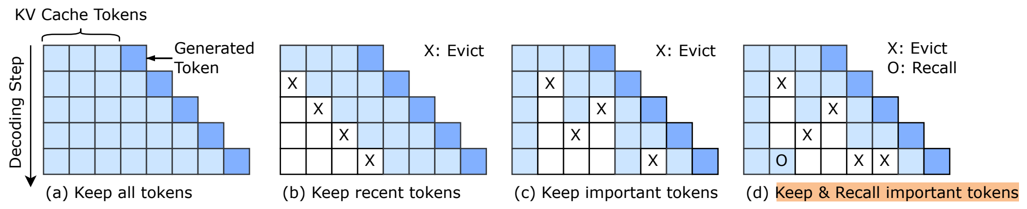


Figure 1: Comparison of different KV cache eviction works.

1 Introduction

Large Language Models (LLMs) are rapidly gaining universal presence, underpinning a myriad of natural language processing applications, including dialogue systems [2, 52, 16], document summarization [62, 24], code completion [13, 46], and question answering [31]. The context length supported by large models is also growing progressively to support more applications like multi-turn chat [2, 52, 16] and text summarization [28, 68, 22]. This shift has seen an expansion from an initial range of up to 16k, advancing steadily towards lengths of 32k [1], 128k [43], and even 2048k [21].

However, handling long contexts poses significant challenges. The key-value cache (KV cache) of LLMs scales with both the batch size and the length of historical context. For each token generated by an LLM, the query must attend to the entire set of context key-values. On one hand, longer KV caches occupy more memory spaces, limiting batch-size as well as throughput. On the other hand, computing attention with long KV cache incurs more memory accesses, hurting the inference latency.

Previous works [58, 64, 37, 23, 40, 6] reveal sparsity in KV cache, which means that only a subset of tokens in the KV cache significantly influence the accuracy of attention computation. They propose two characteristics of token importance: locality and persistence. Locality implies that recent tokens tend to be important for current attention, while persistence suggests that tokens previously deemed important are likely to retain their importance over time. They devise algorithms to retain recent tokens [58, 37, 64] (Figure 1 (b)) or vital tokens [58, 37, 64, 40, 23, 6] (Figure 1 (b)) while evicting older or less crucial ones to reduce memory usage of KV cache and memory accesses of attention.

In this work, we delve deeper into the dynamism of token importance, observing that tokens previously deemed unimportant may regain importance over time and vice versa. Previous works ignore this and may risk permanently discarding the tokens that are vital later on, inadvertently decreasing model accuracy. To address these challenges, we propose ARKVALE, inspired by vLLM [36] to organize tokens into pages for fine-grained management of KV cache. Upon filling a KV-page, we asynchronously copy it to CPU memory as a backup, and summarize/compress it into a much smaller digest by constructing a bounding-volume of the keys of the KV-page. Before each attention computation, we dynamically estimate the importance of each page based on their digests and current query, and selectively recall & evict some pages based on their importance scores (Figure 1 (d)).

Our contribution can be summarized as follows:

- We characterize the dynamism of the importance of tokens in the KV cache of LLM, and find that some unimportant tokens may regain importance over time.
- We propose ARKVALE, a KV cache manager which organize tokens into pages and dynamically evict & recall them based on their importance.
- We propose a method based on bounding-volume to summarize the pages and estimate the importance of them with a given query.

We evaluate ARKVALE against many state-of-the-art KV cache eviction works on various long-context benchmarks. Experimental results show ARKVALE performs well on all the long context tasks with negligible accuracy loss under a cache budget of 2k∼4k and speedup model decoding latency up to 2.2× (1.7× in average) and batching throughput up to 4.6× (3.5× in average) in long context scenarios. Our code is now available at `https://github.com/pku-liang/ArkVale`.

2 Related Work

Many works have been proposed to extend context window of pre-trained models. One prevalent method is the integration of Rotary Position Embeddings (RoPE) [49]. Fine-tuning RoPE's scaling enables the Llama-2 model [55], initially handling 4k tokens, to be expanded to support 32k tokens in LongChat [1] and 128k tokens in Yarn-Llama-2 [43]. Recently, LongRoPE [21] pushes this boundary even further to accommodate up to 2048k tokens. However, as models become more capable of dealing with long context, they also face new challenges in terms of memory usage and inference efficiency. ARKVALE is designed to address such issues.

Some training-aware methods have been proposed to handle these problems. Multi-Query Attention (MQA) [47] and Group-Query Attention (GQA) [7] aim to train LLMs with fewer attention heads in KV cache. Works like RWKV [42], RetNet [50], and Mamba [25] propose Linear Attention/RNN to limit KV cache size. Sparse attention architectures [61, 10, 56, 17, 45, 33, 53] design special sparse pattern of attention during training to control the KV cache size during inference. But these works require much training effort for pre-training or fine-tuning while ARKVALE is a training-free method.

There are also some post-training methods, which evict tokens in KV cache to reduce memory usage and accelerate attention computation. StreamingLLM [58] and LM-Infinite [26] only keep initial tokens and recent tokens to maintain a fixed-size KV cache. Methods like H2O [64], Scissorhands [37] and TOVA [40] keep important tokens in KV cache based on the historical or current attention scores. FastGen [23] categorizes tokens and employs a more nuanced approach for choosing which KV cache tokens to preserve. Keyformer [6] improves the eviction score function by leveraging the Gumbel distribution. Q-Hitter [63] combines KV-cache quantization with KV-cache eviction. These methods rely on historical data to dictate cache eviction and may risk discarding the tokens that are important in the future. Works like IceFormer [38], SparQ [5], ALISA [65], and Quest [51] use post-training sparse attention to alleviate such issue but they require all KV cache residing in memory and thus cannot control KV cache size to optimize memory usage. Many scheduling-based methods [19, 18, 60, 15, 66, 14, 36, 48, 32, 27, 67] are also employed to optimize memory efficiency for LLMs or other DNNs, aiming to enhance memory usage and execution latency through proper computation partitioning and scheduling.

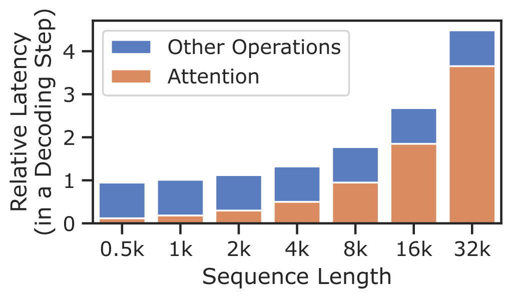
3 Background

3.1 Attention Computation and Generative Inference of LLM

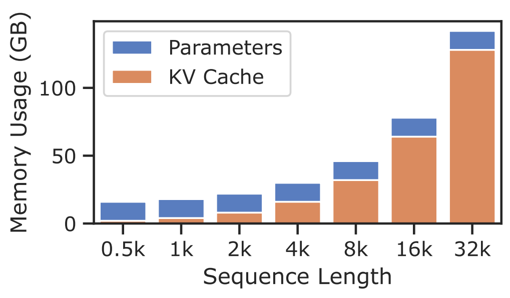
The attention computation in a typical LLM involves mixing token-level information of queries $\mathbf{Q} \in \mathbb{R}^{s_q \times d}$ and keys $\mathbf{K} \in \mathbb{R}^{s_{kv} \times d}$ with values $\mathbf{V} \in \mathbb{R}^{s_{kv} \times d}$, where d is the hidden-dimension and s_q (s_{kv}) means the number of query (key/value) tokens. The attention computation can be formulated as: $\mathbf{S} := (\mathbf{Q} \cdot \mathbf{K}^\top / \sqrt{d}) \in \mathbb{R}^{s_q \times s_{kv}}$, $\mathbf{P} := \texttt{softmax}(\mathbf{S}) \in \mathbb{R}^{s_q \times s_{kv}}$, $\mathbf{O} := (\mathbf{P} \cdot \mathbf{V}) \in \mathbb{R}^{s_q \times d}$, where $\mathbf{P}$ is often called "attention map/scores" and $\mathbf{P}_{i,j}$ reflects the importance of key/value token j to query token i. Then, the generative inference process of LLM mainly consists of two stages: the prefill/prompt stage and the decoding/generation stage. The prefill stage takes a prompt sequence with length s_{in} as input and caches the keys & values computed by each layer of the LLM. The decoding stage uses and updates the KV cache to generate tokens step-by-step, where the current token depends on past tokens' keys & values stored in the KV cache. For the attention in prefill stage, $s_{\text{in}} = s_q = s_{kv}$, while $s_q = 1$ and $s_{kv} =$ #past-tokens $+ 1$ in each decoding step.

3.2 Impact of Long Context

For a decoding step with batch-size as b, number of transformer layers as l, history sequence length as s, hidden-dimension as h, and data type as `float16`, the KV cache requires $4bsh$ bytes memory accesses for each attention computation and occupies $4blsh$ bytes of memory in total. The latency & memory breakdown of a decoding step of model LongChat-7b-v1.5-32k [1] ($l = 32, h = 4096$) with $b = 8$ and $s = 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}$ is shown in Figure 2. As the sequence length increases, accessing the KV cache incurs significant overhead, thereby impacting the end-to-end latency of inference. Besides, as the sequence length increases, the KV cache itself consumes more memory space; for instance, with just a context length of 16k, a batch size of 8 nearly saturates the



(a) Inference latency of attention and other operations.



(b) Memory usage of KV cache and parameters.

Figure 2: Latency & memory breakdown of a decoding step of LongChat-7b-v1.5-32k model with batch-size $b = 8$ and different history sequence-lengths $s = 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}$.

80GB memory of an A100 GPU. This not only hampers the use of larger batch sizes but also poses challenges for deploying models with more parameters.

4 Observation

4.1 Token-level Sparsity of KV Cache

To tackle the aforementioned issues, we aim to exploit the sparsity of the KV cache. We collect data by running LongChat-7b-v1.5-32k [1] on LongBench [9]. Figure 3 illustrates the sparsity of the KV caches of different layers. We organize the tokens in KV cache into pages (with page-size=32), and rank the pages of each layer based on the highest attention scores of the tokens within each page for each decoding step. We can observe that, in each layer except the initial one, less than 10 pages (320 tokens) contribute more than 99% scores in sum, achieving over 95% sparsity. Thus we can retain only a subset of KV tokens for attention.

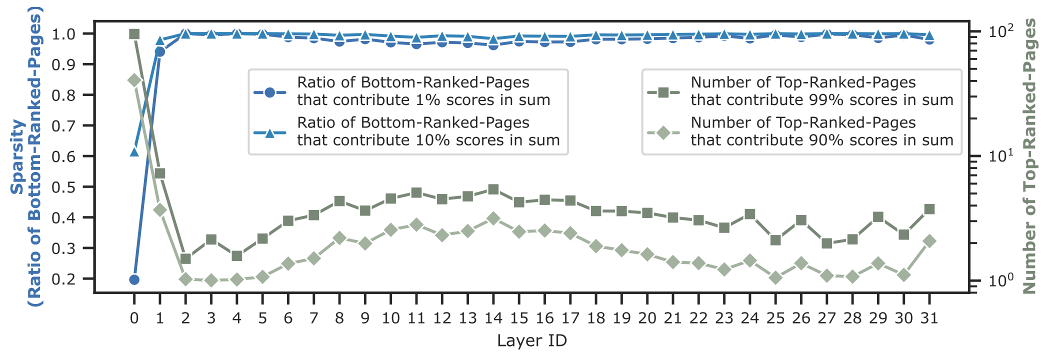
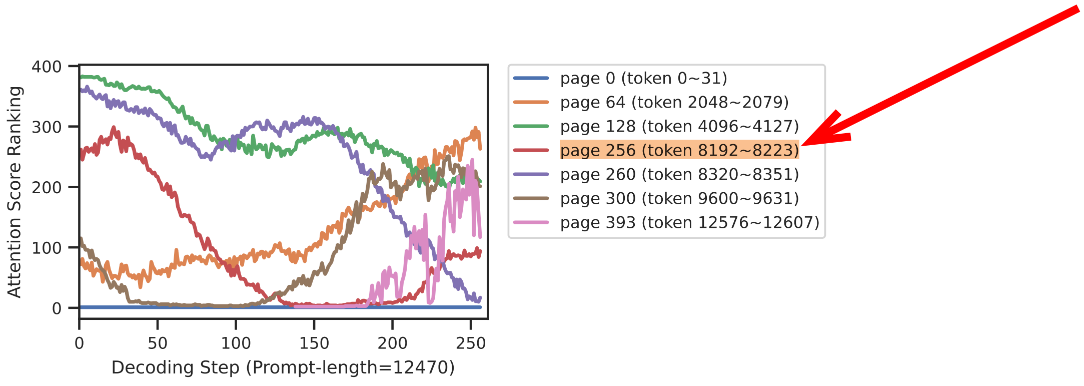


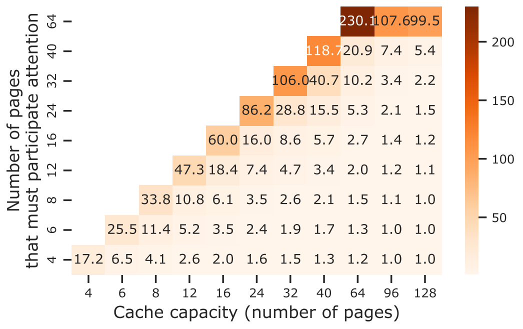
Figure 3: Token-level sparsity of KV cache. We organize the tokens in KV cache into pages (with page-size=32) and rank the pages of each layer based on the highest attention (softmax) scores of the tokens within each page.

4.2 Dynamism of Token Importance

Previous efforts leverage the sparsity of KV cache to control its size by retaining recent tokens while discarding older ones or selectively evicting less important tokens based on their historical attention scores, shown in Figure 1 (b) (c). However, we find that the importance of tokens in the KV cache can dynamically change over time, as shown in Figure 4a. For example, page 256 initially holds little importance but later becomes crucial, with its significant role (at position 12620) occurring nearly 4500 tokens after its sequence position (at position 8192). Previous methods risk prematurely discarding such pages to save KV cache space. To address this, we propose ARKVALE, a shown



(a)



(b)

Figure 4: (a) Dynamsim of token (page) importance (page-size=32). The sample is from GovReport [28]. (b) Number of page-recalls (in average) needed during a decoding step with page-size=32.

in Figure 1 (d), a method designed to recognize shifts in token importance and properly recall vital tokens as well as evict unimportant ones, thereby preventing substantial drops in accuracy.

5 Techniques of ARKVALE

5.1 System Design

Figure 5 shows the design of ARKVALE. Inspired by vLLM [36], ARKVALE arranges KV cache tokens into pages, enabling coarse-grained management of tokens. As shown in Figure 5 (a), once a page is filled, its keys & values are asynchronously moved from GPU to external memory (CPU memory typically) for backup, with keys summarized into a digest kept on the GPU (explained in § 5.2). Prior to attention computation, ARKVALE gauges the importance of each page (whether cached or evicted), using the query and page digests (Figure 5 (b), detailed in § 5.2), and subsequently ranks these pages (Figure 5 (c)) based on their importance scores. The top-k pages (governed by a hyper-parameter k) must engage in attention computation (Figure 5 (e)). If any top-k pages were evicted before, they will be recalled from backup, and bottom-ranked pages in GPU will be evicted for exchange (Figure 5 (d)).

Depending on the cache-size and the number of pages selected for attention, the overhead of page recall may be different. To validate the feasibility of our approach and ensure that recalls do not incur excessive overhead, we uniformly sample four instances from each dataset included in LongBench [9] and gather data from the inference process of LongChat-7b-v1.5-32k [1] to simulate eviction and recall scenarios. We configure the page-size $p = 32$ and vary both the cache capacity c (pages) and the attention size k (the number of top-ranked pages selected for attention). The results, depicted in Figure 4b, reveal that as long as $k < \min(40, c/2)$, we can limit the number of recalls per decoding step to under 10, amounting to a total of 5 MB of data transfer. This transfer takes merely hundreds of microseconds over contemporary CPU-GPU communication interfaces, which is one or two orders of magnitude faster than the end-to-end latency of each decoding step (usualy a few

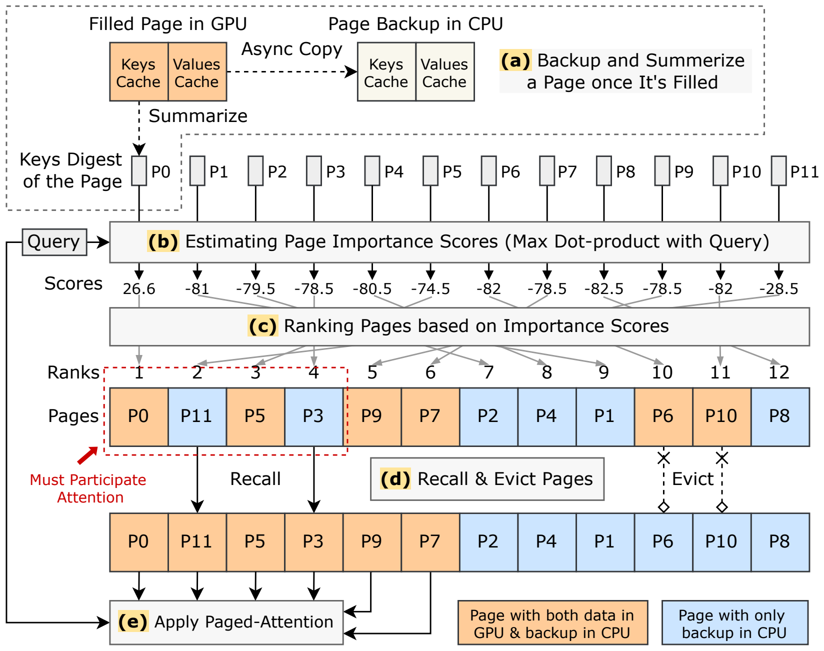
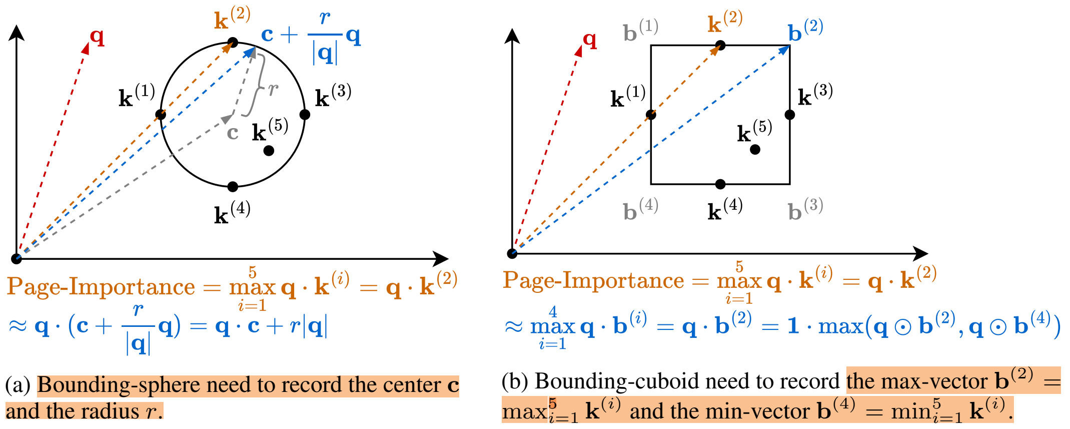


Figure 5: Design overview of ARKVALE

to tens of milliseconds in long context scenarios on NVIDIA A100 GPU). Thus, for page-size p and cache-capacity c (tokens), we set $k = \min(C, c/2)/p$, where C is a hyper-parameters (default $C = 40 * 32 = 1280$).

5.2 Page Summarization & Importance Estimation


$\mathbf{q}$
$\mathbf{k}^{(1)}$
$\mathbf{k}^{(2)}$
$\mathbf{k}^{(3)}$
$\mathbf{k}^{(4)}$
$\mathbf{k}^{(5)}$
$\mathbf{c} + \frac{r}{|\mathbf{q}|}\mathbf{q}$
r
$\mathbf{c}$
$\mathbf{b}^{(1)}$
$\mathbf{b}^{(2)}$
$\mathbf{b}^{(3)}$
$\mathbf{b}^{(4)}$
$\text{Page-Importance} = \max_{i=1}^{5} \mathbf{q} \cdot \mathbf{k}^{(i)} = \mathbf{q} \cdot \mathbf{k}^{(2)}$
$\approx \mathbf{q} \cdot (\mathbf{c} + \frac{r}{|\mathbf{q}|}\mathbf{q}) = \mathbf{q} \cdot \mathbf{c} + r|\mathbf{q}|$
$\text{Page-Importance} = \max_{i=1}^{5} \mathbf{q} \cdot \mathbf{k}^{(i)} = \mathbf{q} \cdot \mathbf{k}^{(2)}$
$\approx \max_{i=1}^{4} \mathbf{q} \cdot \mathbf{b}^{(i)} = \mathbf{q} \cdot \mathbf{b}^{(2)} = \mathbf{1} \cdot \max(\mathbf{q} \odot \mathbf{b}^{(2)}, \mathbf{q} \odot \mathbf{b}^{(4)})$

(a) Bounding-sphere need to record the center $\mathbf{c}$ and the radius r.

(b) Bounding-cuboid need to record the max-vector $\mathbf{b}^{(2)} = \max_{i=1}^{5} \mathbf{k}^{(i)}$ and the min-vector $\mathbf{b}^{(4)} = \min_{i=1}^{5} \mathbf{k}^{(i)}$.

Figure 6: Summarize page keys $\{\mathbf{k}^{(i)}\}_{i=1}^{5}$ into their bounding-volume (sphere/cuboid). We can estimate the max-dot-product between query $\mathbf{q}$ and keys $\{\mathbf{k}^{(i)}\}_{i=1}^{5}$ using the bounding-volume.

For every filled page, we maintain a digest as a compression of it. The digest is much smaller than the page itself, and can be used to estimate the importance of each page regardless of whether the page has been evicted before attention computation. This estimation enables us to rank pages to selectively recall or evict pages based on their importance.

Given a page with n keys $K = \{\mathbf{k}^{(i)}\}_{i=1}^{n}$ ($\mathbf{k}^{(i)} \in \mathbb{R}^d$), and a query $\mathbf{q} \in \mathbb{R}^d$, we measure its importance using the maximum dot product between the query and the keys: $I(\mathbf{q}, K) = \max_{\mathbf{k} \in K} \mathbf{q} \cdot \mathbf{k}$. For two sets of keys, K_1 and K_2, $I(\mathbf{q}, K_1) > I(\mathbf{q}, K_2)$ implies that there's at least one token in K_1 with higher attention (softmax) scores compared to every token in K_2.

It's evident that $\text{argmax}_{\mathbf{k}\in K}\mathbf{q}\cdot\mathbf{k}$ must lie on a vertex of the convex hull of the point set K. Hence, we can approximate $I(\mathbf{q}, K)$ by constructing a convex covering set of K. We leverage the concept of **bounding-volume** [3, 34] to summarize the keys K of a page. A bounding-volume for a set of points is a closed region that completely contains all of them, which is widely used in computer graphics [34, 29, 11, 12].

Bounding-sphere. We can cover the keys K with a sphere, needing to store only a center $\mathbf{c} \in \mathbb{R}^d$ and a radius $r \in \mathbb{R}$ as the digest. Given a query $\mathbf{q} \in \mathbb{R}^d$, we can estimate the importance $I(\mathbf{q}, K)$ leveraging $\mathbf{c}$ and r. Among the points $\mathbf{k}'$ on the surface of the sphere, those for which $\mathbf{k}' - \mathbf{c}$ is parallel to $\mathbf{q}$ yield the maximum dot product with $\mathbf{q}$. Thus we have $\mathbf{k}' = \frac{r}{|\mathbf{q}|}\mathbf{q}$, leading to:

$$I(\mathbf{q}, K) = \max_{\mathbf{k}\in K}\mathbf{q}\cdot\mathbf{k} \approx \mathbf{q}\cdot(\mathbf{c} + \frac{r}{|\mathbf{q}|}\mathbf{q}) = \mathbf{q}\cdot\mathbf{c} + r|\mathbf{q}| \tag{1}$$

An example is shown in Figure 6a. Finding the minimal bounding sphere is complicated; hence, we employ an efficient approximation method: we take the center of the Axis-Aligned Bounding Box (AABB) [12, 11] of K as our sphere center: $\mathbf{c} = \frac{1}{2}(\min_{\mathbf{k}\in K}\mathbf{k} + \max_{\mathbf{k}\in K}\mathbf{k})$ (min and max are element-wise operations in this paper). For the radius r, three alternatives are considered:

$$r_{\text{max}} = \max_{\mathbf{k}\in K}|\mathbf{c}-\mathbf{k}| \quad r_{\text{center}} = \frac{1}{2}(\min_{\mathbf{k}\in K}|\mathbf{c}-\mathbf{k}| + r_{\text{max}}) \quad r_{\text{mean}} = \frac{1}{|K|}\sum_{\mathbf{k}\in K}|\mathbf{c}-\mathbf{k}| \tag{2}$$

where $|...|$ means L2-norm. Strictly speaking, only r_{max} guarantees to enclose all points in K, but the latter two can avoid overestimating $I(\mathbf{q}, K)$ in practice, as shown in Figure 7 and discussed in §6.2.

Bounding-cuboid. We can also cover the keys K directly using an Axis-Aligned Bounding Box (AABB) [12, 11]. It needs to store the max-vector $\mathbf{b}^{\text{max}} = \max_{\mathbf{k}\in K}\mathbf{k}$ and min-vector $\mathbf{b}^{\text{min}} = \min_{\mathbf{k}\in K}\mathbf{k}$ as the digest. Given a query vector $\mathbf{q} \in \mathbb{R}^d$, we can estimate the importance $I(\mathbf{q}, K)$ of K with the boundaries of the box. This is achieved by summing the maximum product by the boundary values and $\mathbf{q}$ across all dimensions:

$$I(\mathbf{q}, K) = \max_{\mathbf{k}\in K}\mathbf{q}\cdot\mathbf{k} \approx \sum_{i=1}^{d}\max_{\mathbf{k}\in K}\mathbf{q}_i\mathbf{k}_i = \sum_{i=1}^{d}\max(\mathbf{q}_i\mathbf{b}_i^{\text{max}}, \mathbf{q}_i\mathbf{b}_i^{\text{min}}) = \mathbf{1}\cdot\max(\mathbf{q}\odot\mathbf{b}^{\text{max}}, \mathbf{q}\odot\mathbf{b}^{\text{min}}) \tag{3}$$

Here, $\mathbf{1} \in \mathbb{R}^d$ denotes a vector of ones, and $\odot$ means element-wise multiplication. An example is shown in Figure 6b. Similar to the bounding-sphere approach, we can employ different "radius vectors" to define cuboids with varying sizes:

$$\mathbf{r}^{\text{max}} = \max_{\mathbf{k}\in K}\texttt{abs}(\mathbf{c}-\mathbf{k}) \quad \mathbf{r}^{\text{center}} = \frac{1}{2}(\min_{\mathbf{k}\in K}\texttt{abs}(\mathbf{c}-\mathbf{k}) + \mathbf{r}^{\text{max}}) \quad \mathbf{r}^{\text{mean}} = \frac{1}{|K|}\sum_{\mathbf{k}\in K}\texttt{abs}(\mathbf{c}-\mathbf{k}) \tag{4}$$

where `abs(...)` computes element-wise absolute values. For each $\mathbf{r}$, we can derive $\mathbf{b}^{\text{max}} = \mathbf{c} + \mathbf{r}$ and $\mathbf{b}^{\text{min}} = \mathbf{c} - \mathbf{r}$, where $\mathbf{c} = \frac{1}{2}(\min_{\mathbf{k}\in K}\mathbf{k} + \max_{\mathbf{k}\in K}\mathbf{k})$ is the center of the AABB of K.

6 Evaluation

6.1 Experimental Setup

We apply our method to LongChat-7b-v1.5-32k [1] and use 6 datasets from LongBench [9] for benchmarking: HotpotQA [59], NarrativeQA [35], Qasper [20], GovReport [28], TriviaQA [30], and PassageRetrieval [9], along with the passkey-retrieval tasks. For comparison, we choose the state-of-the-art KV cache eviction methods including StreamingLLM [58], H2O [64], and TOVA [40] as baselines. As Figure 3 illustrates, the initial layers exhibit relatively low sparsity; hence, we do not apply ArkVale and baselines to the first two layers of model.

Our experiment platform comprises an Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz and an NVIDIA A100 80GB PCIe GPU. The software stack includes CUDA version 12.3, PyTorch [41, 8] version 2.3.0, and HuggingFace Transformers [57] version 4.40.0. We implement ArkVale on top of Huggingface Transformers, with CUTLASS [54], FlashInfer [60], and RAFT [44] for certain kernels.

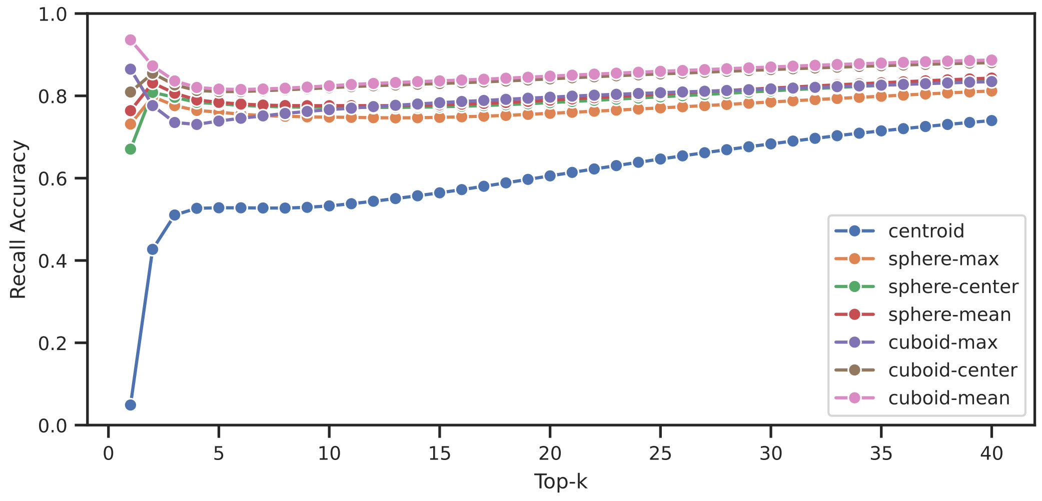


Figure 7: Recall accuracy (the proportion of pages predicted to be among the top-k that indeed belong to the top-k) of different estimation methods. "Centroid" is the baseline which just uses the centroid of keys to estimate the max-dot-product with given query.

6.2 Estimation Accuracy

We begin by evaluating the accuracy of the page summarization & importance estimation methods discussed in §5.2. Using data from the LongBench [9] datasets employed in our experiments, we simulate each decoding step to collect both the actual page rankings and the estimated rankings derived using the digests. For varying values of k, we define recall accuracy as the proportion of overlap between the estimated top-k page set E_k and the true top-k page set R_k, that is: $|E_k \cap R_k|/|R_k|$.

Figure 7 illustrates the recall accuracy of different estimation techniques across various k values. The "centroid" method, which uses the centroid [4] (element-wise average of keys) of page keys as the digest and estimates page importance by directly taking the dot product of the query and the centroid, serves as a straightforward baseline. Other methods are the six introduced in §5.2: spheres with r_{max}, r_{center}, and r_{mean} as well as cuboids with $\mathbf{r}^{\text{max}}$, $\mathbf{r}^{\text{center}}$, and $\mathbf{r}^{\text{mean}}$. As shown in Figure 7, the centroid method cannot recall the top-4 pages with even 50% accuracy and achieves less than 5% accuracy in recalling the top-1 page, substantially undermining inference accuracy. Conversely, our six proposed methods guarantee at least a 60% recall accuracy, with the standout cuboid-mean method ensuring a 95% accuracy for top-1 recall and consistently over 80% for other k values. The cuboid-based methods generally outperform their sphere-based counterparts due to more retained information (two vectors for cuboids versus one vector and a scalar for spheres). Furthermore, the "mean" variants tend to perform better than others, potentially because they provide a more balanced boundary estimation, avoiding overestimation of page importance.

6.3 LongBench Results

Next, we evaluate the accuracy of ARKVALE in general long-context tasks. We employ 6 datasets from LongBench [9] as benchmarks, covering tasks such as multi-document QA with HotpotQA [59], single-document QA with NarrativeQA [35], and Qasper [20], text summarization with GovReport [28], few-shot learning with TriviaQA [30], and synthetic tasks with PassageRetrieval [9]. The target model is LongChat-7b-v1.5-32k [1]. Our comparison baselines include the original model and the state-of-the-art KV cache eviction methods: StreamingLLM [58], H2O [64], and TOVA [40]. We configure four cache budget settings: 4096, 2048, 1024, and 512. Notably, H2O and TOVA's original implementations require the full attention scores matrix to maintain states, preventing the use of optimizations like Flash-Attention [19] in prefill stage and causing Out-of-Memory (OOM) for many benchmark samples. To address this, we split each input into context and question/instruction sections and adopt a two-phase prefill: the first phase uses Flash-Attention [19] to handle the lengthy context without OOM (while we also separately compute the last row of attention scores for H2O

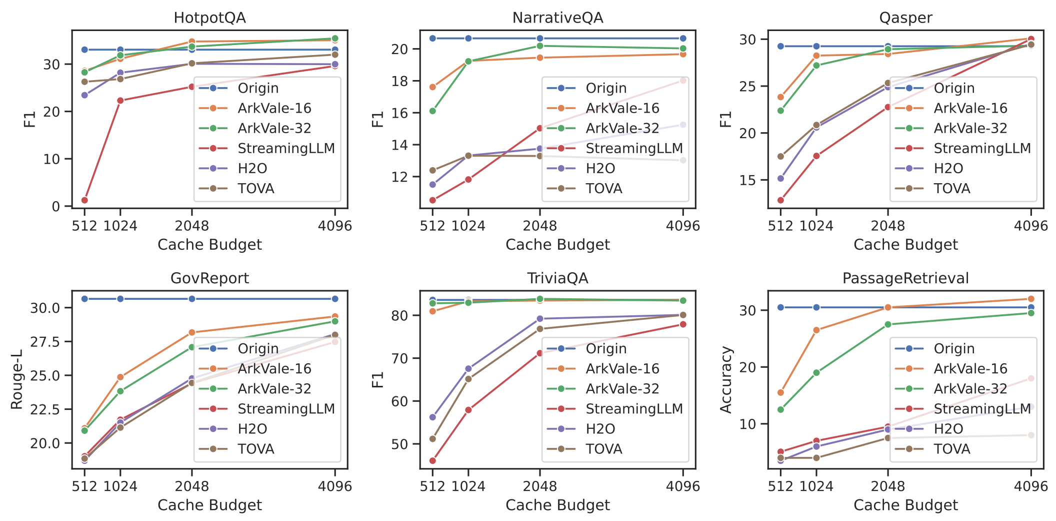


Figure 8: Evaluation on 6 long-context datasets in LongBench [9] with different cache budgets.

and TOVA's initial states update with little memory overhead), and the second phase uses brutal-force attention to process the shorter instruction, ensuring a complete score matrix for H2O and TOVA's states updates. This approach can also better demonstrate the long-range dependencies handling abilities of each method.

Figure 8 shows the results, where ArkVale-16 and ArkVale-32 represent ArkVale with page-sizes of 16 and 32, respectively. ArkVale persistently surpasses baselines across datasets and cache budgets. It nearly equals Origin's performance with budgets over 2048, while other baselines show noticeable disparities when the budget dips below 2048 and even 4096. Specifically, ArkVale achieves comparable results to Origin at budgets of 1024 for HotpotQA, 2048 for NarrativeQA and PassageRetrieval, 1024 for Qasper, and 512 for TriviaQA. Page-size differences have little impact in most test-cases; however, ArkVale-16 often outperforms ArkVale-32 when budgets are tight, as it selects more pages under the same limited budget, enhancing the chances of hitting vital tokens.

6.4 Passkey Retrieval Results

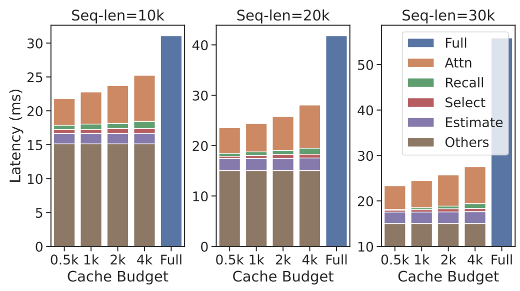
Table 1: Accuracy of passkey retrieval tasks

Context Length	10k				20k				30k			
Cache Budget	512	1024	2048	4096	512	1024	2048	4096	512	1024	2048	4096
StreamingLLM [58]	0%	5%	15%	40%	0%	0%	5%	20%	0%	0%	5%	10%
H2O [64]	5%	5%	15%	40%	0%	5%	5%	20%	5%	5%	5%	15%
TOVA [40]	5%	10%	20%	40%	5%	5%	10%	20%	5%	5%	5%	15%
ArkVale	**100%**	**95%**	**100%**	**95%**	**95%**	**100%**	**100%**	**100%**	**100%**	**95%**	**95%**	**100%**

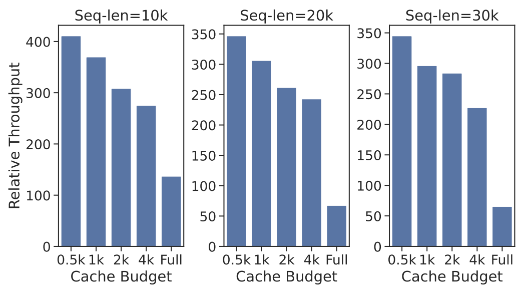
We further examine the accuracy of ArkVale under long-range dependency scenarios in detail. We employ the passkey retrieval task [39] as our benchmark. We establish three context lengths for our tests: 10k, 20k, and 30k. For each context length, we generate 20 unique test cases, each with the passkey inserted at depths corresponding to 0%, 5%, ..., up to 95% of the text's length. Following the setup in Section §6.3, we compare ArkVale against StreamingLLM [58], H2O [64], and TOVA [40]. Similarly, we adopt two-phase prefill for input texts to prevent OOM incidents of H2O and TOVA and more effectively evaluate each method's long-range dependencies handling.

The results are shown in Table 1. Baselines, which permanently evict some tokens, risk discarding passkey-related information during inference, leading to accuracy falling short of 50% and declining further as the context length increases or the cache budget decreases. In contrast, ArkVale examines the importance of evicted tokens (pages) and promptly recalls vital ones, thereby maintaining a stable passkey retrieval accuracy above 95% across all tested context lengths and cache budgets.

6.5 Performance Results



(a) Latency Breakdown (batch-size=4)



(b) Throughput Comparison

Figure 9: Decoding latency breakdown and achievable maximum throughput with page-size=32 and different seq-lens & cache budgets compared to model with full KV cache.

We set up test-cases with lengths close to 10k, 20k, and 30k from GovReport [28] in LongBench [9] and examined ArkVale's inference latency with settings of a batch-size=4, page-size=32, and KV cache budgets of 512, 1024, 2048, and 4096. Figure 9a shows the results. Here, "Full" refers to the baseline without KV cache eviction, whereas "Estimate," "Select," and "Recall" denote the overhead for estimating page importance, choosing pages for computations and evictions, and recalling pages from CPU memory, respectively. ArkVale outperforms the baseline across all the tested text lengths, reaching up to $2.2\times$ speedup ($1.7\times$ in average). The main bottleneck of the baseline is the full-attention, whose latency grows linearly with seq-len, whereas our attention's latency is mainly governed by the fixed cache budget. Our extra latency overhead mostly arises from estimating importance, which grows linearly with the $\#pages = \frac{\text{seq-len}}{\text{page-size}}$. Recalling, conversely, adds small overhead (especially when seq-len is large and cache-budget is small), consistent with the discussion in §5.1.

We further compare the achievable maximum throughput in serving scenarios of ArkVale versus the baseline under a 40 GB memory limit for KV cache. Figure 9b shows the results. Our throughput reaches up to $4.6\times$ ($3.5\times$ in average) to that of baseline. This achievement is not only due to the reduced inference latency but also to a decreased per sample memory usage of the KV cache, enabling larger batch-size to enhance weight sharing to lessen the average latency of Linear computations per sample.

7 Limitation

The limitation of ArkVale mainly lies on storing a backup for each KV cache page in external memory (typically the CPU memory). Although the latency of data transfer for backup during decoding phase can be hided by asynchronous copy, the backup latency during prefilling phase is hard to completely overlapped. Also, the backups may occupy much CPU memory. When the CPU's memory capacity is insufficient, these backups may need to be offloaded to disk storage.

8 Conclusion

In this paper, we propose ArkVale, a method designed for managing KV cache eviction and recall. It organizes KV cache tokens into pages, backs them up, and creates summaries, enabling the recognition of page importance to recall vital pages evicted before. Our method performs well on various long context tasks with few accuracy loss under a cache budget of 2k$\sim$4k and speeds up decoding latency up to $2.2\times$ and boosts decoding throughput up to $4.6\times$ in long-context scenarios.

9 Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. T2325001) and Beijing Municipal Science and Technology Program (No. Z241100004224015).

References

[1] How Long Can Open-Source LLMs Truly Promise on Context Length? | LMSYS Org.

[2] Introducing ChatGPT. https://openai.com/index/chatgpt.

[3] Bounding volume. *Wikipedia*, April 2024.

[4] Centroid. *Wikipedia*, May 2024.

[5] SparQ Attention: Bandwidth-Efficient LLM Inference. In *ICLR Workshop ME-FoMo*, April 2024.

[6] Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: KV Cache Reduction through Key Tokens Selection for Efficient Generative Inference. In *MLSys*. arXiv, April 2024.

[7] 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 *EMNLP*, December 2023.

[8] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, volume 2 of *ASPLOS '24*, pages 929–947, New York, NY, USA, April 2024. Association for Computing Machinery.

[9] 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, August 2023.

[10] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. (arXiv:2004.05150), December 2020.

[11] Gino Van Den Bergen. Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools*, 2(4):1–13, January 1997.

[12] Panpan Cai, Chandrasekaran Indhumathi, Yiyu Cai, Jianmin Zheng, Yi Gong, Teng Sam Lim, and Peng Wong. Collision Detection Using Axis Aligned Bounding Boxes. In Yiyu Cai and Sui Lin Goei, editors, *Simulations, Serious Games and Their Applications*, pages 1–14. Springer Singapore, Singapore, 2014.

[13] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code, July 2021.

[14] Renze Chen, Zijian Ding, Size Zheng, Meng Li, and Yun Liang. MoteNN: Memory Optimization via Fine-grained Scheduling for Deep Neural Networks on Tiny Devices. In *2024 61st Design Automation Conference (DAC)*, 2024.

[15] Renze Chen, Zijian Ding, Size Zheng, Chengrui Zhang, Jingwen Leng, Xuanzhe Liu, and Yun Liang. Magis: Memory optimization via coordinated graph transformation and scheduling for dnn. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 607–621, New York, NY, USA, 2024. Association for Computing Machinery.

[16] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, and Joseph E. Gonzalez. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

[17] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers, April 2019.

[18] Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning, July 2023.

[19] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *NIPS*. arXiv, June 2022.

[20] 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, May 2021.

[21] Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM Context Window Beyond 2 Million Tokens, February 2024.

[22] Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. Multi-News: A Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model, June 2019.

[23] 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 *ICLR*. arXiv, 2024.

[24] Tanya Goyal and Greg Durrett. Evaluating Factuality in Generation with Dependency-level Entailment, October 2020.

[25] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, December 2023.

[26] Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. LM-Infinite: Simple On-the-Fly Length Generalization for Large Language Models, November 2023.

[27] Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Hanyu Dong, and Yu Wang. FlashDecoding++: Faster Large Language Model Inference on GPUs. *arXiv preprint arXiv:2311.01282*, 2023.

[28] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient Attentions for Long Document Summarization, April 2021.

[29] Pablo Jiménez, Federico Thomas, and Carme Torras. 3D collision detection: A survey. *Computers & Graphics*, 25(2):269–285, 2001.

[30] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension, May 2017.

[31] Ehsan Kamalloo, Nouha Dziri, Charles L. A. Clarke, and Davood Rafiei. Evaluating Open-Domain Question Answering in the Era of Large Language Models, July 2023.

[32] Marisa Kirisame, Steven Lyubomirsky, Altan Haan, Jennifer Brennan, Mike He, Jared Roesch, Tianqi Chen, and Zachary Tatlock. Dynamic Tensor Rematerialization. 2021.

[33] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. In *ICLR*, September 2019.

[34] James T. Klosowski, Martin Held, Joseph SB Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[35] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA Reading Comprehension Challenge, December 2017.

[36] 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 *SOSP*, SOSP '23, pages 611–626, New York, NY, USA, October 2023. Association for Computing Machinery.

[37] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the Persistence of Importance Hypothesis for LLM KV Cache Compression at Test Time. In *NIPS*, August 2023.

[38] Yuzhen Mao, Martin Ester, and Ke Li. IceFormer: Accelerated Inference with Long-Sequence Transformers on CPUs. In *ICLR*, October 2023.

[39] Amirkeivan Mohtashami and Martin Jaggi. Landmark Attention: Random-Access Infinite Context Length for Transformers. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*, July 2023.

[40] Matanel Oren, Michael Hassid, Yossi Adi, and Roy Schwartz. Transformers are Multi-State RNNs, January 2024.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[42] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, Johan S. Wind, Stansilaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the Transformer Era, May 2023.

[43] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient Context Window Extension of Large Language Models. In *ICLR*, October 2023.

[44] Rapidsai. Rapidsai/raft: Raft contains fundamental widely-used algorithms and primitives for data science, graph and machine learning., 2022.

[45] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full Attention Transformer with Sparse Computation Cost. In *NIPS*, volume 34, pages 22470–22482. Curran Associates, Inc., 2021.

[46] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code Llama: Open Foundation Models for Code, January 2024.

[47] Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need, November 2019.

[48] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. *ICML*, 2023.

[49] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding, November 2023.

[50] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive Network: A Successor to Transformer for Large Language Models, July 2023.

[51] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference. In *ICML*, 2024-06-15.

[52] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

[53] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse Sinkhorn Attention. In *ICML*, pages 9438–9447. PMLR, November 2020.

[54] Vijay Thakkar, Pradeep Ramani, Cris Cecka, Aniket Shivam, Honghao Lu, Ethan Yan, Jack Kosaian, Mark Hoemmen, Haicheng Wu, Andrew Kerr, Matt Nicely, Duane Merrill, Dustyn Blasig, Fengqi Qiao, Piotr Majcher, Paul Springer, Markus Hohnerbach, Jin Wang, and Manish Gupta. CUTLASS, January 2023.

[55] 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 Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh 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, July 2023.

[56] Hanrui Wang, Zhekai Zhang, and Song Han. SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110, February 2021.

[57] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric 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, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[58] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *ICLR*. arXiv, September 2023.

[59] 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, September 2018.

[60] Zihao Ye, Lequn Chen, Ruihang Lai, Yilong Zhao, Size Zheng, Junru Shao, Bohan Hou, Hongyi Jin, Yifei Zuo, Liangsheng Yin, Tianqi Chen, and Luis Ceze. Accelerating self-attentions for llm serving with flashinfer, February 2024.

[61] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. In *NIPS*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020.

[62] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. Benchmarking Large Language Models for News Summarization, January 2023.

[63] Zhenyu Zhang, Shiwei Liu, Runjin Chen, Bhavya Kailkhura, Beidi Chen, and Atlas Wang. Q-Hitter: A Better Token Oracle for Efficient LLM Inference via Sparse-Quantized KV Cache. 6:381–394, 2024-05-29.

[64] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H$_2$O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *NIPS*, December 2023.

[65] Youpeng Zhao, Di Wu, and Jun Wang. ALISA: Accelerating Large Language Model Inference via Sparsity-Aware KV Caching. In *ISCA*. arXiv, 2024.

[66] Size Zheng, Renze Chen, Meng Li, Zihao Ye, Luis Ceze, and Yun Liang. vmcu: Coordinated memory management and kernel optimization for dnn inference on mcus. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 452–464, 2024.

[67] Size Zheng, Siyuan Chen, Peidi Song, Renze Chen, Xiuhong Li, Shengen Yan, Dahua Lin, Jingwen Leng, and Yun Liang. Chimera: An Analytical Optimizing Framework for Effective Compute-intensive Operators Fusion. In *HPCA*, pages 1113–1126, 2023.

[68] 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, April 2021.

A Appendix / supplemental material

Optionally include supplemental material (complete proofs, additional experiments and plots) in appendix. All such materials **SHOULD be included in the main submission.**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss it in Section 7.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon. "
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The design of our method is comprehensively discussed in §5 and the experiment setup is detailed in §6.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Due to some constraints, our code is currently not available for open access. Nonetheless, our design is thoroughly discussed in §5.

Guidelines: For some reasons, we cannot open-source our code now.

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experiment setup is outlined in §6.1. Additional details on specific settings are also provided for each experiment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: In this paper's experiments, metrics for model performance (F1, Rouge-L, Accuracy) do not require error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experiment environment is detailed in §6.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This paper does not entail ethical risks.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: Due to space constraints, we don't delve into the broader impacts within the main text. In a nutshell, our work positively contributes to the wider application of LLMs by reducing inference latency and memory usage while maintaining accuracy.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: In §6.1, we cite the methods we compare to and the datasets we use and specify versions of the software tools employed.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.