ZipCache: Accurate and Efficient KV Cache Quantization with Salient Token Identification

Yefei He[1] **Luoming Zhang**[1] **Weijia Wu**[2] **Jing Liu**[3]
Hong Zhou[1]* **Bohan Zhuang**[1,3]*

[1]Zhejiang University, China
[2]National University of Singapore, Singapore
[3]ZIP Lab, Monash University, Australia

Abstract

KV cache stores key and value states from previous tokens to avoid re-computation, yet it demands substantial storage space, especially for long sequences. Adaptive KV cache compression seeks to discern the saliency of tokens, preserving vital information while aggressively compressing those of less importance. However, previous methods of this approach exhibit significant performance degradation at high compression ratios due to inaccuracies in identifying salient tokens. Additionally, the compression process introduces excessive overhead, substantially increasing memory burdens and the generation latency. In this paper, we present ZipCache, an accurate and efficient KV cache quantization method for large language models (LLMs). First, we construct a strong baseline for quantizing KV cache. Through the proposed channel-separable tokenwise quantization scheme, the memory overhead of quantization parameters are substantially reduced compared to fine-grained groupwise quantization. To enhance the compression ratio, we propose normalized attention score as an effective metric for identifying salient tokens by considering the lower triangle characteristics of the attention matrix. The quantization bit-width for each token is then adaptively assigned based on their saliency. Moreover, we develop an efficient approximation method that decouples the saliency metric from full attention scores, enabling compatibility with fast attention implementations like FlashAttention. Extensive experiments demonstrate that ZipCache achieves superior compression ratios, fast generation speed and minimal performance losses compared with previous KV cache compression methods. For instance, when evaluating Mistral-7B model on GSM8k dataset, ZipCache is capable of compressing the KV cache by $4.98\times$, with only a 0.38% drop in accuracy. In terms of efficiency, ZipCache also showcases a 37.3% reduction in prefill-phase latency, a 56.9% reduction in decoding-phase latency, and a 19.8% reduction in GPU memory usage when evaluating LLaMA3-8B model with a input length of 4096. Code is available at https://github.com/ThisisBillhe/ZipCache/.

1 Introduction

LLMs with the next-token-prediction scheme have achieved remarkable advancements in various text-related tasks, such as language understanding [13, 34, 10], content creation [1, 5, 36], coding [3, 29, 42] and mathematics [33, 23, 35]. In this generation scheme, the forthcoming token interacts with all previous tokens via the attention mechanism [38], where the query, key and value states will be

*Corresponding author. Email: `zhouhong_zju@zju.edu.cn`, `bohan.zhuang@gmail.com`

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-2181

calculated for each token. As the past tokens will not be altered, previously computed key and value states can be stored as KV cache to prevent re-computations, significantly improving the generation speed. However, as the batch size and the input context length grows, the stored KV cache emerges as a new memory bottleneck for LLMs. For example, when serving a 175B-parameter LLM [1] with a batch size of 64 and a context length of 4096, the KV cache can occupy 1.2TB of memory space, while the model weights only require 350GB. Meanwhile, the size of KV cache will continue to increase as decoding progresses. Therefore, the compression of KV cache is crucial for the efficient deployment of LLMs.

Recent compression methods for KV cache can be broadly categorized into two types. The first type of methods compresses the KV cache uniformly, without considering the significance of individual tokens. To preserve performance, these methods often rely on either high-precision quantization [21] or maintaining recent tokens in full-precision [32], which undoubtedly compromise the compression ratio. Additionally, if salient tokens are not among the most recent ones, such as in information retrieval tasks, it may result in degraded performance. The other type of methods [46, 43, 16] compress KV cache adaptively by identifying salient tokens and compresses them separately. This approach aligns with the observation that a minority of tokens contribute the majority of attention scores [41], potentially achieving higher compression ratios than non-adaptive methods. However, current adaptive KV cache compression methods [46, 43] use accumulated attention scores as a metric of token saliency, which is insufficient in two aspects. First, accumulated attention scores is **inaccurate** in identifying important tokens. Due to the presence of attention masks, the attention matrix is a lower triangular matrix. Earlier tokens tend to have larger softmax attention values and more attention scores to be accumulated, as illustrated in Figure 3. Under this metric, the saliency of the most recent tokens can never surpass that of the first token, thereby introducing a bias in determining token saliency. Additionally, to obtain accumulated attention scores, full attention matrices must be explicitly computed and stored, which can be **inefficient** for serving LLMs. Given an input context length of l, fast attention implementations such as FlashAttention [8, 7] only require $O(l)$ memory by computing attention output in blocks without retaining complete attention matrices. By contrast, storing full attention matrices requires $O(l^2)$ memory, and the large number of memory accesses significantly slows down the inference speed, as depicted in Figure 4.

To address these challenges, we introduce ZipCache, an efficient KV cache compression method that attains exceptionally high compression ratios by accurate salient token identification. Figure 1 presents an overview of latency-accuracy comparisons among ZipCache and diverse KV cache compression methods. We start by designing an efficient quantization baseline for compressing the KV cache. To preserve performance, predecessor methods [32, 21] employ fine-grained groupwise quantization, which involves independent quantization for a small channel group within each token. However, this method necessitates storing extensive quantization parameters and results in significant memory overhead. By contrast, we introduce a channel-separable quantization scheme that decouples the quantization along channel and token dimensions. This method significantly reduces the quantization overhead without compromising performance. To accurately recognize salient tokens, we introduce a new token saliency metric based on normalized attention scores, which alleviates the bias towards earlier tokens that accumulate more values. All tokens, without exception, will be quantized to the target bit-width based on their estimated saliency, boosting the overall compression ratio. Moreover, to ease integration with fast attention implementations, we introduce an efficient approximation of the token saliency metric. This approximation only relies on computing and storing attention scores from a few number of tokens, which we refer to as probe tokens. An effective probe token selection strategy is then introduced to minimize performance loss. As a result, the majority of tokens can benefit from fast attention implementations, significantly enhancing the generation speed.

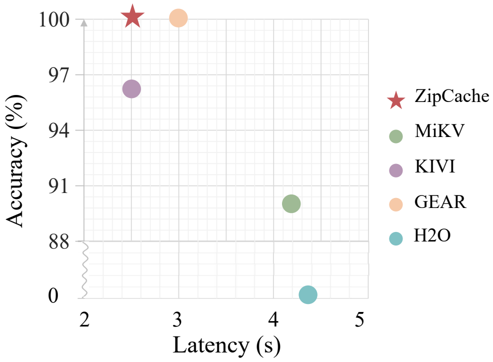


Figure 1: Accuracy and efficiency comparisons across various KV cache compression methods. Data is collected with LLaMA3-8B model on Line Retrieval dataset. Among these methods, ZipCache achieves the highest accuracy, generation speed and compression ratio. Details can be found in the supplementary material.

In summary, our contributions are as follows:

- We establish an efficient channel-separable quantization scheme for KV cache, which significantly reduces the overhead of quantization parameters without compromising performance compared to fine-grained groupwise quantization approach.
- We propose an accurate metric for assessing token saliency based on normalized attention scores. All tokens are adaptively quantized according to their assessed saliency, thereby improving the overall compression ratio.
- We further develop an efficient approximation method for the token saliency metric that integrates seamlessly with fast attention implementations, enhancing generation speed.
- By integrating these three techniques, we present ZipCache, an accurate and efficient framework for KV cache compression. Extensive experiments demonstrate that ZipCache reaches a new state-of-the-art performance for KV cache compression in terms of compression ratio, accuracy and generation efficiency.

2 Related Work

2.1 Model Quantization

Quantization is a prevalent technique for compressing deep neural networks by representing model weights and activations with lower numerical bit-widths. This technique can be categorized into two primary approaches based on the necessity of fine-tuning: post-training quantization (PTQ) [26, 17, 14] and quantization-aware training (QAT) [28, 31]. For large language models (LLMs), where fine-tuning can be data- and computation-intensive, PTQ is often the preferred method [40, 11, 45, 27]. In this paper, we also quantize KV cache in a post-training manner. For both approaches, quantization can be implemented at various levels of granularity, including channelwise, tokenwise, and groupwise approach. Typically, a finer quantization granularity involves the independent quantization of smaller parameter groups, which often results in improved performance albeit at the cost of more quantization parameters and increased memory overhead. In the context of LLMs, fine-grained quantization is frequently utilized due to the presence of outliers [22, 45]. However, for KV cache compression, this will greatly reduce the overall compression ratio.

Mixed precision quantization [39, 44, 12, 2] allocates varying bit-widths to distinct parts of a model or tensor, enabling a more compact compression. This approach originates from the observation that model components exhibit differing sensitivities to quantization. Consequently, components with low sensitivity can utilize reduced bit-widths without impairing performance. For LLMs, previous studies [46, 43, 30, 18] have shown significant disparities in the importance of tokens, indicating that heavy compression of non-critical tokens has minimal impact on overall performance. This insight highlights the applicability of mixed precision quantization for compressing the KV cache.

2.2 KV Cache Compression

While KV cache effectively prevents re-computation and significantly enhances generation speed, its memory footprint is notably substantial with long-context input. To alleviate this, many efforts have been made to reduce the KV cache size. Based on the compression method, these methods can be categorized into two groups: token dropping [46, 16, 30] and KV cache quantization [43, 21, 32]. The former identifies and drops unimportant tokens in the KV cache. For example, H2O [46] only maintain 20% heavy-hitted tokens and 20% recent tokens while evicting the rest. However, discarding tokens permanently erases their information, which proves to be suboptimal for tasks such as retrieval [43]. Conversely, the latter category employs quantization on the cached key and value states, and mixed precision quantization can further be applied once token importance is identified [43]. To tackle the outliers present in the KV cache, these methods extract the outlier as full precision [21] or use finer-grained quantization scheme [32], which increases the quantization overhead. In this study, we propose an efficient channel-separable quantization scheme with reduced quantization overhead and strong performance. Additionally, both categories of methods commonly adopt accumulated attention scores as the metric for token importance [46, 43]. However, we observe that this criterion is inaccurate and can result in significant performance deterioration at low bit-widths. In contrast, we achieve superior compression performance by utilizing a more accurate metric for identifying salient tokens.

https://doi.org/10.52202/079017-2181

3 Preliminary

3.1 Attention Block in LLMs

Given an input prompt, the generation process of LLMs can be broadly categorized into two distinct phases: the prefill phase, which computes and stores the KV cache for input tokens, and the decoding phase, where new tokens are generated through a next-token-prediction scheme. Given input data $\mathbf{X}$ and an attention block with its weight matrices $\mathbf{W}_{\mathrm{Q}}$, $\mathbf{W}_{\mathrm{K}}$ and $\mathbf{W}_{\mathrm{V}}$, the prefill phase can be formulated as:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_{\mathrm{Q}}, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_{\mathrm{K}}, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_{\mathrm{V}}, \tag{1}$$

$$\mathbf{A} = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right), \quad \mathbf{O} = \mathbf{A}\mathbf{V}. \tag{2}$$

Here, d_k is the dimension of the key, and $\mathbf{A}$ refers to the attention scores. $\mathbf{K}$ and $\mathbf{V}$ will be stored as KV cache. For clarity, we have omitted the output projection.

For the decoding phase, given $\mathbf{x}$ as the embedding vector of the current token, the query $\mathbf{q}$ becomes a vector and the KV cache matrices will be updated as follow:

$$\mathbf{q} = \mathbf{x}\mathbf{W}_{\mathrm{Q}}, \quad \mathbf{K} = \mathrm{Concat}(\mathbf{K}, \mathbf{x}\mathbf{W}_{\mathrm{K}}), \quad \mathbf{V} = \mathrm{Concat}(\mathbf{V}, \mathbf{x}\mathbf{W}_{\mathrm{V}}). \tag{3}$$

The attention output are then computed as follows:

$$\mathbf{a} = \mathrm{Softmax}\left(\frac{\mathbf{q}\mathbf{K}^T}{\sqrt{d_k}}\right), \quad \mathbf{o} = \mathbf{a}\mathbf{V}. \tag{4}$$

To ensure clarity and consistency, we introduce notation to define the hyper-parameters used in the paper. Specifically, we denote the batch size as b, the number of attention heads as h, the sequence length as l, and the head dimension as d.

3.2 Model Quantization

Uniform quantization is adopted in our study and all experiments. Given a floating-point vector $\mathbf{x}$, it can be uniformly quantized to k-bit as follows:

$$\hat{\mathbf{x}} = \mathcal{Q}_U(\mathbf{x}, k) = (\mathrm{clip}(\lfloor \frac{\mathbf{x}}{s} \rceil + z, 0, 2^k - 1) - z) \cdot s. \tag{5}$$

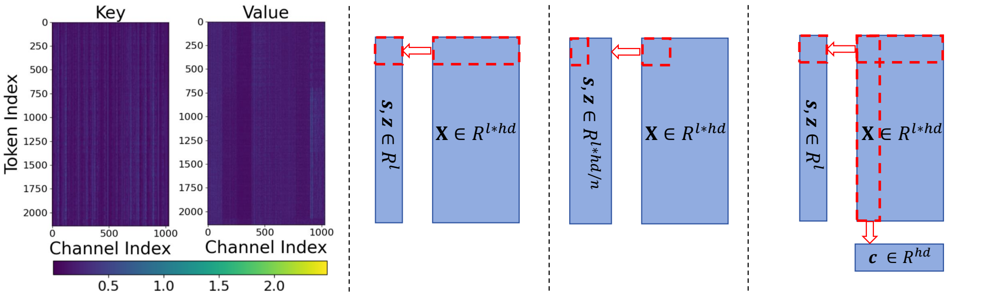
Here, $\lfloor \cdot \rceil$ denotes the round operation, $s = \frac{\max(\mathbf{x}) - \min(\mathbf{x})}{2^k - 1}$ and $z = -\lfloor \frac{\min(\mathbf{x})}{s} \rceil$ are quantization parameters. It should be noted that the quantization parameters are stored in full-precision, which can lead to significant overhead if the quantization is fine-grained.

4 Method

4.1 A Strong Baseline for KV Cache Quantization

Tokenwise quantization, as depicted in Figure 2(b) is prevalent in quantizing large language models (LLMs) due to the distinct representations of individual tokens. However, it has been widely observed, as illustrated in Figure 2(a), that outliers emerge within the channel dimensions of key and value matrices [43, 32], posing challenges for tokenwise quantization. To address this, recent work [32] resorts to groupwise quantization, where outlier channels are processed in distinct groups, as illustrated in Figure 2(c). However, this fine-grained quantization approach introduces excessive memory overhead, thereby significantly impacting the compression ratio. For instance, considering $\mathbf{X} \in \mathbb{R}^{b \times h \times l \times d}$ as the data to be quantized and a group size of n, tokenwise quantization only results in $2bl$ quantization parameters, while groupwise quantization would yield $\frac{2bhld}{n}$ quantization parameters. Since these parameters are usually stored in full precision, this overhead would constitute a substantial portion of the storage cost for quantized data.

Motivated by depthwise separable convolution [19], we introduce an efficient channel-separable tokenwise quantization scheme, which disentangles the channel and token dimensions. As shown in



(a) Visualization of Key and Value States

(b) Tokenwise Quantization

(c) Groupwise Quantization

(d) Channel-separable Tokenwise Quantization

Figure 2: Visualization and different quantization granularities for key and value states. Here, we omit the batch dimension for simplicity. For keys, channel outliers emerge, yet token representations exhibit minimal differences. For values, both channel outliers and distinct token representations exist.

Figure 2(d), our approach initiates by normalizing each channel of data $\mathbf{X}$ with a scaling factor $\mathbf{c}$. For the i-th channel in $\mathbf{X}$, the normalization process can be formulated as:

$$\mathbf{X}_i = \frac{\mathbf{X}_i}{\mathbf{c}_i}, \text{ where } \mathbf{c}_i = \sqrt{\max(|\mathbf{X}_i|)}. \tag{6}$$

After normalization, each channel is scaled to a closed magnitude, mitigating the influence of outliers during tokenwise quantization. Subsequently, tokenwise quantization can be reliably applied and the scales $\mathbf{c}$ are multiplied back to restore the magnitude of each channel. The process of channel-separable tokenwise quantization is summarized in the supplementary material. Within this quantization scheme, the total number of quantization parameters amounts to $hd + 2bl$, representing a notable reduction compared to groupwise quantization, while effectively balancing the outlier channels and the representation of each token.

Table 1: Performance comparisons of different quantization granularities for KV cache. The KV cache is quantized to 4-bit and the compression ratio is calculated with $b = 8$, $hd = l = 4096$ and $n = 32$. Data is collected with LLaMA3-8B model on GSM8k dataset.

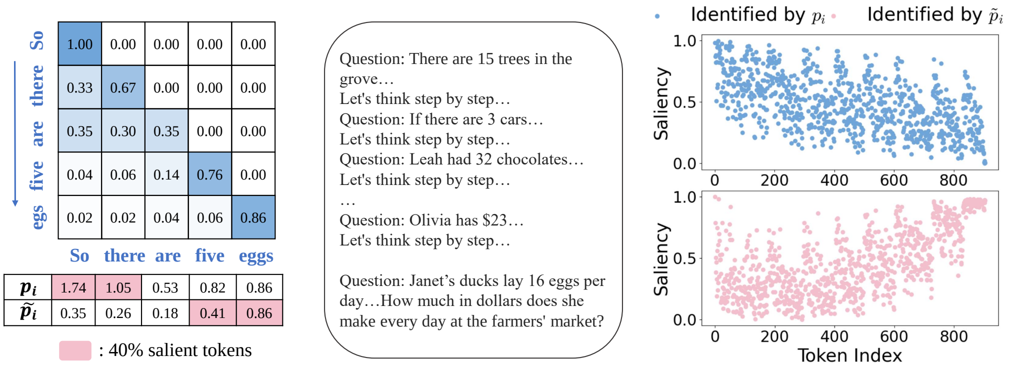
Key Cache Quantization Granularity	Value Cache Quantization Granularity	Quantization Parameters	Compression Ratio	Acc.(%)
/	/	0	1×	55.88
Groupwise	Groupwise	$4bhld/n$	3.2×	54.51
Tokenwise	Tokenwise	$4bl$	3.99×	49.81
Channelwise	Tokenwise	$2hd + 2bl$	4.00×	52.77
Channelwise	Channel-separable Tokenwise	$3hd + 2bl$	4.00×	**54.74**

As referred to Figure 2(a), since the differences in token representations are small in key cache, we employ channelwise quantization for the key cache to further reduce overhead and employ channel-separable tokenwise quantization for the value cache. As depicted in Table 1, this configuration yields superior performance with reduced quantization overhead compared with groupwise quantization, thereby establishing a robust baseline for KV cache quantization.

4.2 Accurate Salient Token Identification

Adaptive KV cache compression [46, 43, 16] aims to discern the saliency of each token, keeping the information of salient tokens while evicting or aggressively compressing the rest, to achieve a higher compression ratio. These salient tokens, also referred to as "Heavy Hitters" [46], are often identified based on accumulated attention scores. Given attention score matrix $\mathbf{A} \in \mathbb{R}^{l \times l}$, the saliency of token i is estimated by:

$$p_i = \sum_{k=1}^{l} \mathbf{A}_{k,i}. \tag{7}$$



	So	there	are	five	eggs
So	1.00	0.00	0.00	0.00	0.00
there	0.33	0.67	0.00	0.00	0.00
are	0.35	0.30	0.35	0.00	0.00
five	0.04	0.06	0.14	0.76	0.00
eggs	0.02	0.02	0.04	0.06	0.86

p_i	1.74	1.05	0.53	0.82	0.86
$\tilde{p}_i$	0.35	0.26	0.18	0.41	0.86

▨ : 40% salient tokens

(a)

Question: There are 15 trees in the grove…
Let's think step by step…
Question: If there are 3 cars…
Let's think step by step…
Question: Leah had 32 chocolates…
Let's think step by step…
…
Question: Olivia has $23…
Let's think step by step…

Question: Janet's ducks lay 16 eggs per day…How much in dollars does she make every day at the farmers' market?

(b)

(c)

Figure 3: (a) A toy example to illustrate accumulated attention scores and normalized attention scores. Initial tokens have larger attention scores and more values to be accumulated. (b) A sample from GSM8k dataset with chain-of-thoughts (CoT) prompting. (c) The probability of each token being selected as a salient token, measured by both accumulated and normalized attention scores. Tokens correspond to the final question are identified as low saliency by accumulated attention scores.

Tokens with large saliency values are then considered salient tokens. However, this approach has inherent limitations due to the lower triangular nature of the attention score matrix, as illustrated in Figure 3(a). There are two primary issues. **Firstly**, earlier tokens benefit from having more values accumulated since the elements above the diagonal are all zero. For instance, in a sequence of length l, the initial token accumulates l positive values, whereas the final token only accumulates one. **Secondly**, $\mathrm{Softmax}$ function converts real numbers into probabilities, so that the earlier rows of the attention matrix tending to have higher values, as fewer numbers are involved in the $\mathrm{Softmax}$ calculation. Consequently, the accumulated attention score of the final token will always be smaller than that of the first, which exceeds 1. To address this, previous works, such as H2O [46], always maintain recent caches in full precision. Nevertheless, this solution is suboptimal since recent tokens are not necessarily the most significant ones.

To enhance the evaluation of each token's saliency, we introduce an accurate token saliency metric based on normalized attention scores $\tilde{p}_i$:

$$\tilde{p}_i = \frac{\sum_{k=1}^{l} \mathbf{A}_{k,i}}{\mathrm{nnz}(\mathbf{A}_{:,i})} \tag{8}$$

Here, $\mathrm{nnz}(\mathbf{A}_{:,i})$ denotes the number of non-zero elements in the i-th column of $\mathbf{A}$. As evidenced in Figure 3(a), normalizing the accumulated attention scores mitigates the influence of excessively large values in the initial rows of the attention score matrix, thereby delivering a more precise assessment. To validate the efficacy of our new metric, we input a sample from GSM8k dataset with chain-of-thoughts (CoT) prompting to the LLaMA3-8B model and identify saliency of each token by Eq. 7 and Eq. 8, respectively. As depicted in Figure 3(b) and (c), the salient tokens are at the end of the prompt, which correspond to the question for LLM to answer. However, these tokens are identified as low saliency by accumulated attention scores. Under the KV cache compression framework, these tokens would either be discarded or quantized to extremely low bit-width, resulting in a significant performance deterioration. In contrast, our method accurately identifies the salient tokens. Additional experimental results regarding the accuracy of our method will be detailed in Section 5.2.

4.3 Efficient Approximation of Saliency Metric

As analyzed in Section 4.2, adaptive KV cache compression requires the explicit computation of full attention scores, as referred to Figure 4(b), which clashes with fast attention implementations like FlashAttention [8, 7, 9]. As shown in Figure 4(c), FlashAttention computes attention outputs in tiles without storing the intermediate attention scores. To reconcile the efficiency of FlashAttention with the substantial compression offered by adaptive KV caching, we devise an effective approximation for Eq. 8 as a measure of token saliency. Specifically, we sample a small group of tokens, designated

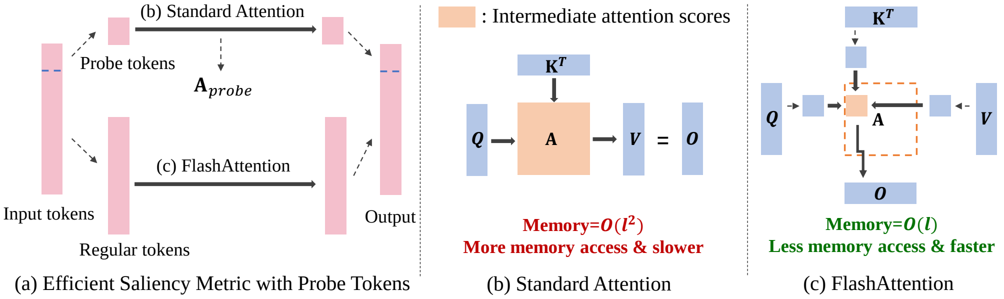


Figure 4: (a): Efficient saliency metric only requires attention scores of probe tokens through standard attention, enabling fast computation for the majority of tokens through FlashAttention. (b): In standard attention, full attention scores are computed before deriving the attention output. (c): FlashAttention avoids large attention matrix memory transfers by partitioning input matrices into blocks for incremental computation.

as **probe tokens**, and compute their attention scores $\mathbf{A}_{probe}$ as follows:

$$\mathbf{A}_{probe} = \text{Softmax}\left(\frac{\mathbf{Q}_{probe}\mathbf{K}^T}{\sqrt{d_k}}\right). \tag{9}$$

By substituting $\mathbf{A}_{probe}$ into Eq. 8, we can approximate the saliency of all tokens. For the remaining non-probe tokens, their attention scores do not have to be computed explicitly, enabling the integration of fast attention implementations to expedite the generation process, as illustrated in Figure 4(a).

However, the positions of the probe tokens will undoubtedly affects the accuracy of the approximated token saliency and the selection of probe tokens is under explored. In this study, we suggest four strategies for sampling probe tokens:

- **Random tokens**. The probe tokens are randomly sampled from all positions.
- **Special tokens**. The special tokens and punctuation tokens will be treated as probe tokens.
- **Recent tokens**. The most recent tokens are selected as probe tokens.
- **Random+recent tokens**. The probe tokens will be divided into two parts, one using recent tokens and the other randomly selecting from the remaining tokens.

Table 2: Performance comparisons of various probe strategies. Data is collected from LLaMA3-8B model on GSM8k dataset. We quantize 40% salient tokens to 4-bit and the remaining 60% tokens to 2-bit. The proportion of probe tokens is 10%.

Probe Strategy	Acc.(%)
All tokens	52.54
Random tokens	47.46
Special tokens	46.78
Recent tokens	51.10
Random+recent tokens	**52.08**

It should be emphasized that our study diverges from prior research [16] in that, rather than directly choosing special or recent tokens as salient tokens, we opt to sample a subset of tokens as "probes" to detect the salient ones. As depicted in Table 2, we present a comprehensive comparison of the performance among four distinct sampling strategies. Among the four strategies examined, a hybrid approach that combines recent tokens with randomly selected tokens emerges as the most effective. Unless otherwise specified, this hybrid strategy with 5% recent tokens and 5% random tokens will be employed in our method.

5 Experiment

5.1 Implementation Details

Models and datasets. To validate the efficacy of our proposed method, we conduct experiments with three open-source LLMs: Mistral [20], LLaMA2 [37] and LLaMA3. These models are evaluated on three challenging benchmarks: GSM8k [6] for math problem solving, HumanEval [4] for code generation, and Line Retrieval [25] for data retrieval. To ensure reproducibility, the reported results are obtained using the Language Model Evaluation Harness [15] and LongEval [24].

https://doi.org/10.52202/079017-2181

Quantization and generation settings. We employ mixed precision quantization for KV cache where salient tokens will be quantized to 4-bit while the remaining will be quantized to 2-bit. For both subsets, we apply channelwise quantization for the key cache and channel-separable tokenwise quantization for the value cache. The proportion of salient tokens will be denoted by "Saliency Ratio" in the experimental results. During the decoding process, ZipCache adopts a streaming strategy [21] and repeats the compression process for the KV cache whenever 100 new tokens are generated.

5.2 Comparison with SOTA methods

5.2.1 Evaluation on GSM8k

We begin our evaluation on GSM8k dataset with chain-of-thoughts (CoT) prompting, and the results are presented in Table 3. This task requires LLM to solve mathematical problems and return the final answer without multiple options. This task poses considerable challenges and previous KV cache compression methods manifest notable declines in accuracy. For instance, KIVI [32] shows an accuracy drop of 7.89% on LLaMA3-8B model, indicating the suboptimality of preserving recent tokens in full precision instead of identifying salient ones. Moreover, there is a substantial decrease in accuracy, amounting to 20.4%, for MiKV [43] under the high compression ratio. This suggests that accumulated attention scores mistakenly identify salient tokens, resulting in the loss of vital information during compression. By contrast, the proposed normalized attention scores can accurately measure token saliency, leading to a substantial enhancement in accuracy by 18.27% for LLaMA3-8B models in comparison to MiKV. In comparison to GEAR [21], which quantizes the entire KV cache to 4-bit, our approach additionally quantizes 40% tokens to 2-bit with enhanced performance on Mistral-7B model. This underscores the superiority of accurate adaptive compression of KV cache.

Table 3: Performance comparisons on GSM8k with CoT prompts. Here, "H/L" denotes the bit-width for salient tokens (high-precision) and regular tokens (low-precision), respectively. The compression ratio is calculated with an average input length of $l = 840$.

Model	Method	Bit-width (H/L)	Saliency Ratio	Compression Ratio	Acc.(%)
Mistral-7B	FP16	16/16	100%	1×	41.62
	H2O [46]	16/0	40.0%	2.50×	1.67
	GEAR [21]	4/4	100%	3.00×	39.42
	KIVI [32]	16/2	15.2%	3.46×	39.04
	MiKV [43]	4/2	60.0%	4.98×	36.32
	ZipCache	4/2	60.0%	**4.98**×	**41.24**
LLaMA2-7B	FP16	16/16	100%	1×	14.18
	H2O [46]	16/0	40.0%	2.50×	13.50
	GEAR [21]	4/4	100%	3.00×	12.96
	KIVI [32]	16/2	15.2%	3.46×	13.19
	MiKV [43]	4/2	60.0%	4.98×	9.02
	ZipCache	4/2	60.0%	**4.98**×	**13.50**
LLaMA2-13B	FP16	16/16	100%	1×	28.05
	H2O [46]	16/0	40.0%	2.50×	26.00
	GEAR [21]	4/4	100%	3.00×	25.40
	KIVI [32]	16/2	15.2%	3.46×	27.29
	MiKV [43]	4/2	60.0%	4.98×	23.65
	ZipCache	4/2	60.0%	**4.98**×	**27.85**
LLaMA3-8B	FP16	16/16	100%	1×	55.88
	H2O [46]	16/0	40.0%	2.50×	27.82
	GEAR [21]	4/4	100%	3.00×	49.43
	KIVI [32]	16/2	15.2%	3.46×	47.99
	MiKV [43]	4/2	70.0%	4.69×	35.48
	ZipCache	4/2	70.0%	**4.69**×	**53.75**

5.3 Evaluation on HumanEval

In this subsection, we assess the performance of code generation across various KV cache compression methods, as summarized in Table 4. Remarkably, ZipCache attains a compression ratio of 4.94× without sacrificing performance when tested with the Mistral-7B model, outperforming predecessor methods. Moreover, when evaluating on LLaMA3-8B model, our approach outperforms KIVI-2 [32] by 7.32% with a significantly higher compression ratio (4.39× vs. 2.55×). It should be noted that the

https://doi.org/10.52202/079017-2181

average input length for this task is only 119, while KIVI retains the recent 32 tokens in full-precision, thereby considerably diminishing its overall compression ratio. This underscores the advantage of ZipCache over methods that consistently retain information of recent tokens.

Table 4: Performance comparisons on HumanEval for code generation. Here, "H/L" denotes the bit-width for salient tokens (high-precision) and regular tokens (low-precision), respectively. 0-bit denotes the tokens are evicted. The compression ratio is calculated with an average input length of $l = 120$.

Model	Method	Bit-width (H/L)	Saliency Ratio	Compression Ratio	Acc.(%)
Mistral-7B	FP16	16/16	100%	1×	29.27
	H2O [46]	16/0	40.0%	2.50×	14.63
	GEAR [21]	4/4	100%	3.00×	28.05
	KIVI [32]	16/2	26.7%	2.55×	28.05
	MiKV [43]	4/2	60.0%	4.94×	27.44
	ZipCache	4/2	60.0%	**4.94**×	**29.27**
LLaMA2-7B	FP16	16/16	100%	1×	14.02
	H2O [46]	16/0	40.0%	2.50×	11.59
	GEAR [21]	4/4	100%	3.00×	**13.02**
	KIVI [32]	16/2	26.7%	2.55×	11.59
	MiKV [43]	4/2	80.0%	4.39×	10.37
	ZipCache	4/2	80.0%	**4.39**×	12.80
LLaMA3-8B	FP16	16/16	100%	1×	33.54
	H2O [46]	16/0	40.0%	2.50×	15.85
	GEAR [21]	4/4	100%	3.00×	28.66
	KIVI [32]	16/2	26.7%	2.55×	25.61
	MiKV [43]	4/2	80.0%	4.39×	29.88
	ZipCache	4/2	80.0%	**4.39**×	**32.93**

5.3.1 Evaluation on Line Retrival

We further evaluate the data retrieval performance of various KV cache compression methods on Line Retrieval [25] dataset, where LLMs are required to retrieve specific content from a record of lines using a corresponding line index. The accuracy results under various number of lines are depicted in Figure 5. Notably, all quantization-based compression methods exhibit superior performance compared to the eviction-based approach H2O [46]. For eviction-based methods, information is permanently discarded upon eviction, whereas quantization introduces only minor errors while preserving the integrity of the data. Additionally, in comparison to KIVI [32], which always maintains recent caches at full precision, our approach consistently achieves better retrieval accuracy. This can be attributed to the nature of retrieval tasks, where salient tokens may appear at any position within the context, rather than being confined to the most recent caches. Moreover, when compared to MiKV [43], which employs accumulated attention scores as a saliency metric, our method yields a remarkable 42% accuracy improvement when evaluated using 200 lines on the Mistral-7b model. This substantial enhancement once more highlights the effectiveness of normalized attention scores in identifying salient tokens.

Additional experimental results on HumanEval [4] can be found in the supplementary material.

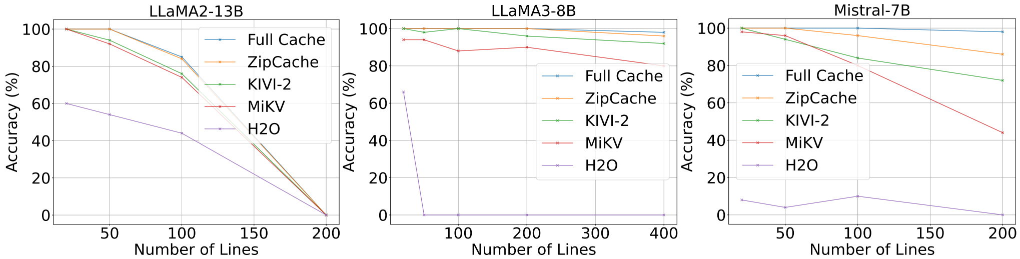
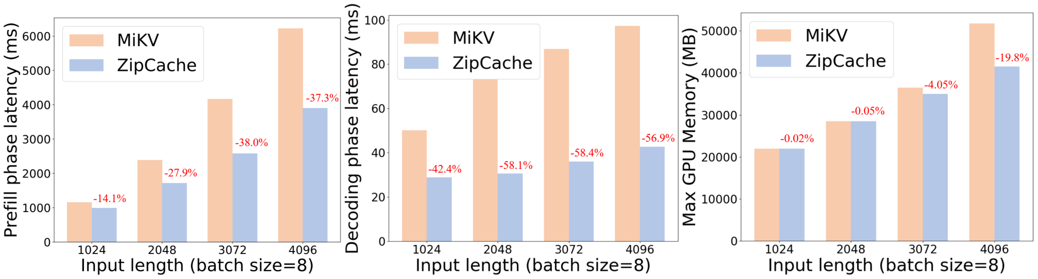


Figure 5: Performance comparisons of various KV cache compression methods on Line Retrieval.

https://doi.org/10.52202/079017-2181

5.4 Generation Efficiency

In this subsection, we compare the latency and memory consumption of ZipCache and MiKV [43] under various input lengths, as depicted in Figure 6. Data is collected by serving LLaMA3-8B model on a Nvidia A100 GPU. MiKV employs accumulated attention scores to estimate token saliency, necessitating the use of standard attention for both prefill and decoding phases. Conversely, through an efficient approximate saliency metric, ZipCache requires only the calculation of the attention matrix for 10% of the tokens, while the remaining 90% tokens can be computed using either FlashAttention [7] or FlashDecoding [9]. Consequently, ZipCache achieves faster inference speed and lower memory usage, boasting a 37.3% reduction in prefill-phase latency, a 56.9% reduction in decoding-phase latency, and a 19.8% reduction in GPU memory usage when the input length scales to 4096.



(a) Prefill phase latency (b) Decoding phase latency (c) GPU memory

Figure 6: Comparisons of prefill-phase, decoding-phase latency and memory consumption between MiKV and ZipCache.

6 Conclusion and Future Work

In this paper, we have proposed ZipCache, an accurate and efficient mixed-precision quantization framework for compressing KV cache. To commence, we introduce a channel-separable quantization scheme for KV cache, effectively reducing the overhead of storing quantization parameters compared to traditional fine-grained quantization schemes without performance degradation. Additionally, we present a novel metric for accurately assessing token saliency based on normalized attention scores. This metric enables adaptive quantization of all tokens according to their saliency, leading to improved compression ratios without sacrificing model performance. Moreover, we introduce an efficient approximation method for the token saliency metric, seamlessly integrating with fast attention implementations such as FlashAttention and FlashDecoding. This enhancement significantly boosts generation speed and reduces GPU memory requirements. Our extensive experiments have demonstrated that ZipCache achieves state-of-the-art compression performance in terms of compression ratio, accuracy and generation speed. We believe that ZipCache will pave the way for more practical and scalable deployment of LLMs in various real-world applications.

Limitations and Broader Impacts. While ZipCache presents promising advancements in KV cache mixed-quantization frameworks for LLMs, the saliency ratio is manually specified before evaluation and cannot be automatically adjusted based on task datasets. Moreover, similar to other generative models, ZipCache can potentially be used to generate malicious content.

Acknowledgement This work was supported by National Key Research and Development Program of China (2022YFC3602601).

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] A. Chauhan, U. Tiwari, et al. Post training mixed precision quantization of neural networks using first-order information. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1343–1352, 2023.

[3] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[4] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, march 2023. *URL https://lmsys. org/blog/2023-03-30-vicuna*, 3(5), 2023.

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

[7] T. Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. 2023.

[8] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

[9] T. Dao, D. Haziza, F. Massa, and G. Sizov. Flash-decoding for long-context inference, 2023.

[10] J. C. de Winter. Can chatgpt pass high school exams on english language comprehension? *International Journal of Artificial Intelligence in Education*, pages 1–16, 2023.

[11] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

[12] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019.

[13] M. Du, F. He, N. Zou, D. Tao, and X. Hu. Shortcut learning of large language models in natural language understanding. *Communications of the ACM*, 67(1):110–120, 2023.

[14] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[15] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, 12 2023.

[16] S. Ge, Y. Zhang, L. Liu, M. Zhang, J. Han, and J. Gao. Model tells you what to discard: Adaptive kv cache compression for llms. In *The Twelfth International Conference on Learning Representations*, 2024.

[17] Y. He, L. Liu, J. Liu, W. Wu, H. Zhou, and B. Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36, 2023.

[18] L. Hou, R. Y. Pang, T. Zhou, Y. Wu, X. Song, X. Song, and D. Zhou. Token dropping for efficient bert pretraining. *arXiv preprint arXiv:2203.13240*, 2022.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[20] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[21] H. Kang, Q. Zhang, S. Kundu, G. Jeong, Z. Liu, T. Krishna, and T. Zhao. Gear: An efficient kv cache compression recipefor near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527*, 2024.

[22] Y. J. Kim, R. Henry, R. Fahim, and H. H. Awadalla. Finequant: Unlocking efficiency with fine-grained weight-only quantization for llms. *arXiv preprint arXiv:2308.09723*, 2023.

[23] C. Li, W. Wang, J. Hu, Y. Wei, N. Zheng, H. Hu, Z. Zhang, and H. Peng. Common 7b language models already possess strong math capabilities. *arXiv preprint arXiv:2403.04706*, 2024.

[24] D. Li, R. Shao, A. Xie, Y. Sheng, L. Zheng, J. Gonzalez, I. Stoica, X. Ma, , and H. Zhang. How long can open-source llms truly promise on context length?, June 2023.

[25] D. Li, R. Shao, A. Xie, Y. Sheng, L. Zheng, J. Gonzalez, I. Stoica, X. Ma, and H. Zhang. How long can context length of open-source llms truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[26] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2020.

[27] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[28] J. Liu, R. Gong, X. Wei, Z. Dong, J. Cai, and B. Zhuang. Qllm: Accurate and efficient low-bitwidth quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[29] J. Liu, C. S. Xia, Y. Wang, and L. Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] Z. Liu, A. Desai, F. Liao, W. Wang, V. Xie, Z. Xu, A. Kyrillidis, and A. Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36, 2024.

[31] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

[32] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

[33] H. Luo, Q. Sun, C. Xu, P. Zhao, J. Lou, C. Tao, X. Geng, Q. Lin, S. Chen, and D. Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

[34] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[35] M. Tan, L. Wang, L. Jiang, and J. Jiang. Investigating math word problems using pretrained multilingual language models. *arXiv preprint arXiv:2105.08928*, 2021.

[36] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[37] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

[39] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8612–8620, 2019.

[40] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

[41] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2023.

[42] F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, pages 1–10, 2022.

[43] J. Y. Yang, B. Kim, J. Bae, B. Kwon, G. Park, E. Yang, S. J. Kwon, and D. Lee. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024.

[44] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021.

[45] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

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

Appendix

A Calculation of Overhead for Different Quantization Schemes

Assuming $b = 8$, $hd = l = 4096$, and that the KV cache is quantized to 4-bit, we proceed to calculate the actual compression ratio for different quantization granularities. For groupwise quantization with a group size of $n = 32$, the compression ratio R_{group} is given by:

$$R_{group} = \frac{2 \times bhld \times 16}{2 \times bhld \times 4 + \frac{4bhld}{n} \times 16} = 3.200 \tag{A}$$

For tokenwise quantization, the compression ratio R_{token} can be calculated as:

$$R_{token} = \frac{2 \times bhld \times 16}{2 \times bhld \times 4 + 4 \times bl \times 16} = 3.992 \tag{B}$$

For our proposed quantization baseline, the compression ratio $R_{baseline}$ is determined by:

$$R_{baseline} = \frac{2 \times bhld \times 16}{2 \times bhld \times 4 + 3 \times hd \times 16 + 2 \times bl \times 16} = 3.995 \tag{C}$$

B Implementation Details of ZipCache

In this section, we provide an overview of the channel-separable tokenwise quantization scheme in Algorithm 1. Additionally, we present the process of ZipCache's prefill phase as described in Algorithm 2, as well as its decoding phase detailed in Algorithm 3. It is worth mentioning that during both the prefill and decoding phases, rather than calculating attention outputs separately for probe tokens and regular tokens followed by merging, FlashAttention [7] is utilized to compute the attention output for all tokens simultaneously. Additionally, attention scores of probe tokens are calculated. By bypassing the substantial memory accesses associated with matrix splitting and merging, this strategy enhances generation speed.

Algorithm 1: Channel-separable Tokenwise Quantization (CSTQuant)

```
procedure CSTQuant:
    Input: data $\mathbf{X} \in \mathbb{R}^{l \times hd}$, target bit-width $k$
    for $i \leftarrow 0$ to $hd$ do
        $\mathbf{c}_i = \sqrt{\max(|\mathbf{X}_i|)}$
        $\mathbf{X}_i = \frac{\mathbf{X}_i}{\mathbf{c}_i}$ // Normalizing each channel of $\mathbf{X}$
    $\hat{\mathbf{X}}$ =TokenQuant$(\mathbf{X}, k)$ // Do tokenwise quantization
    for $i \leftarrow 0$ to $hd$ do
        $\hat{\mathbf{X}}_i = \hat{\mathbf{X}}_i \times \mathbf{c}_i$ // Rescale each channel of $\mathbf{X}$
    return $\hat{\mathbf{X}}$
```

C Additional Experimental Results

C.1 Effect of Saliency Metric

In this section, we conduct ablation studies on our saliency metric, as shown in Table A. It demonstrates the superiority of our saliency metric over using accumulated attention scores or consistently prioritizing the latest tokens.

C.2 Accuracy and Efficiency Comparisons of various KV cache compression methods

In this section, we present the accuracy and efficiency comparisons of various KV cache compression methods, as presented in Table B. Data is collected by evaluating LLaMA3-8B model on 200-line

Algorithm 2: ZipCache for Prefill Phase

```
procedure ZipCachePrefill:
    Input: Query states $\mathbf{Q}$, key states $\mathbf{K}$, value states $\mathbf{V}$, saliency ratio $r\%$, bit-width for salient
           tokens $k_h$, bit-width for regular tokens $k_l$
    // Salient Token Identification
    Select probe tokens and compute their attention scores $\mathbf{A}_{probe}$ by Eq. 9
    Measure the token saliency $\tilde{p}$ with $\mathbf{A}_{probe}$ by Eq. 8
    // Computing Attention Output with FlashAttention
    $\mathbf{O} = \text{FlashAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$
    // Compressing KV Cache
    Partition key states: $\mathbf{K}_{salient}, \mathbf{K}_{regular} = \text{Split}(\mathbf{K}, \tilde{p}, r\%)$
    Partition value states: $\mathbf{V}_{salient}, \mathbf{V}_{regular} = \text{Split}(\mathbf{V}, \tilde{p}, r\%)$
    $\mathbf{K}_{salient} = \text{ChannelQuant}(\mathbf{K}_{salient}, k_h)$, $\mathbf{V}_{salient} = \text{CSTQuant}(\mathbf{V}_{salient}, k_h)$
    $\mathbf{K}_{regular} = \text{ChannelQuant}(\mathbf{K}_{regular}, k_l)$, $\mathbf{V}_{regular} = \text{CSTQuant}(\mathbf{V}_{regular}, k_l)$
    $\hat{\mathbf{K}} = \text{Concat}(\mathbf{K}_{salient}, \mathbf{K}_{regular})$
    $\hat{\mathbf{V}} = \text{Concat}(\mathbf{V}_{salient}, \mathbf{V}_{regular})$
    // Return Attention Output and Compressed KV Cache
    return $\mathbf{O}$, $(\hat{\mathbf{K}}, \hat{\mathbf{V}})$
```

Algorithm 3: ZipCache for Decoding Phase

```
procedure ZipCacheDecoding:
    Input: Query vector $\mathbf{q}$, key vector $\mathbf{k}$, value vector $\mathbf{v}$, KV cache $(\hat{\mathbf{K}}, \hat{\mathbf{V}})$, saliency ratio $r\%$,
           bit-width for salient tokens $k_h$, bit-width for regular tokens $k_l$, decoding token index
           $i$, probe attention score $\mathbf{A}_{probe}$
    $\mathbf{K} = \text{Concat}(\mathbf{k}, \hat{\mathbf{K}})$ // Concatenate key cache
    $\mathbf{V} = \text{Concat}(\mathbf{v}, \hat{\mathbf{V}})$ // Concatenate value cache
    $\mathbf{o} = \text{FlashAttention}(\mathbf{q}, \mathbf{K}, \mathbf{V})$ // Compute attention output
    $i = i + 1$
    if $i == 100$ then
        // Re-compress every 100 tokens
        Extract $\mathbf{K}[: -100]$ and $\mathbf{V}[: -100]$ and adaptively compress them with $\mathbf{A}_{probe}$
        Reset $i = 0$, $\mathbf{A}_{probe} = \text{None}$
    else if $i > 95$ or $randint(0, 100) < 5$ then
        // probe tokens consists of 5% recent and 5% random tokens.
        Compute attention scores $\mathbf{a}$ of current token by Eq. 4
        $\mathbf{A}_{probe} = \text{Concat}(\mathbf{a}, \mathbf{A}_{probe})$
    // Return Attention Output, KV Cache and Attention Scores from Probe
       Tokens
    return $\mathbf{o}$, $(\mathbf{K}, \mathbf{V})$, $\mathbf{A}_{probe}$
```

retrieval task with a Nvidia A100 GPU. We use a batch size of 8 and an average input length of 3072. To ensure a fair comparison, we implement GEAR and KIVI with FlashAttention integration.

Notably, the latency of ZipCache is lower than that of GEAR, which can be attributed to our efficient quantization scheme, whereas GEAR has a high overhead due to outlier extraction. Compared to KIVI, ZipCache's latency is slightly higher (2584.01 ms vs. 2482.26 ms), but ZipCache achieves a higher compression ratio and better performance. This difference is due to KIVI's fixed compression strategy, while we adaptively compress the KV cache based on the saliency. In comparison to MiKV [43], which identifies salient tokens through accumulated attention scores, our method achieves a notable 10.0% accuracy improvement by accurately pinpointing salient tokens and a substantial 38.0% decrease in prefill latency by integrating FlashAttention [7].

Moreover, to demonstrate the efficacy of the proposed efficient quantization scheme, we also implement ZipCache with groupwise quantization. The results show that using groupwise quantization

Table A: The effect of various saliency metric on GSM8k with CoT prompts. Here, “H/L" denotes the bit-width for salient tokens (high-precision) and regular tokens (low-precision), respectively. “Locality" means the recent tokens are identified as salient tokens. The compression ratio is calculated with an average input length of $l = 840$.

Model	Metric	Bit-width (H/L)	Saliency Ratio	Compression Ratio	Acc.(%)
Mistral-7B	FP16	16/16	100%	1×	41.62
	Locality	4/2	60.0%	4.98×	25.40
	Accumulated Attention Scores	4/2	60.0%	4.98×	38.20
	Normalized Attention Scores	4/2	60.0%	4.98×	**41.24**

increases inference speed (2664.05 ms vs. 2584.01 ms) and reduces the compression rate ($3.81\times$ vs. $4.43\times$) due to massive quantization overhead.

Table B: Accuracy and efficiency comparisons over LLaMA3-8B on the 200-line retrieval task. Here, “H/L" denotes the bit-width for salient tokens (high-precision) and regular tokens (low-precision), respectively. 0-bit denotes the tokens are evicted. Saliency ratio denotes the proportion of salient tokens. The compression ratio is calculated with an average input length of $l = 3072$.

Method	Bit-width (H/L)	Saliency Ratio	Compression Ratio	Acc.(%)	Prefill-phase Latency (ms)
FP16	16/16	100%	1×	100	2340.11
H2O	16/0	40.0%	2.50×	0	4335.01
GEAR	4/4	100%	3.00×	100	2968.43
KIVI	16/2	8.33%	4.36×	96	2482.26
MiKV	4/2	80.0%	4.43×	90	4170.61
ZipCache	4/2	80.0%	4.43×	100	2584.01
ZipCache (Groupwise Quantization)	4/2	80.0%	3.81×	100	2664.05

C.3 Evaluation on LongBench

In this subsection, we evaluate the performance of ZipCache on LongBench using the longchat-7b-v1.5-32k model, as shown in Table C. The results show that ZipCache outperforms the previous state-of-the-art method, KIVI on long context scenario.

Table C: Performance comparisons on LongBench.

Model	Method	Qasper	QMSum	MultiNews	TREC	TriviaQA	SAMSum	LCC	RepoBench-P
Llama-2-7b-chat	FP16	21.92	21.01	26.09	64.00	83.51	41.5	58.39	52.26
	KIVI-2	14.31	**20.76**	25.75	64.00	**83.38**	39.14	56.17	50.12
	ZipCache	**20.93**	20.69	**26.12**	**64.50**	**83.38**	**40.11**	**56.60**	**52.00**

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 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: The authors have discussed the limitations of the work.

   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]

https://doi.org/10.52202/079017-2181

Justification: The paper does not include theoritical 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 authors have disclosed all the information needed to reproduce the main experimental results.

   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?

https://doi.org/10.52202/079017-2181

Answer: [Yes]

Justification: The code is open sourced.

Guidelines:

- 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 authors have included experimental details.

   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: Error bars are not reported because of limited computing resources.

   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.

https://doi.org/10.52202/079017-2181

- 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 authors have provided information on computing resources.

   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: The authos confirm that the paper conforms with the NeurIPS Code of Ethics.

   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: [Yes]

    Justification: The authors have discussed broader impacts of the paper.

    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

https://doi.org/10.52202/079017-2181

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: The paper has 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: The authors have cited the related papers.

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: The 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: The 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: The 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.

https://doi.org/10.52202/079017-2181