KEYFORMER: KV CACHE REDUCTION THROUGH KEY TOKENS SELECTION FOR EFFICIENT GENERATIVE INFERENCE

Muhammad Adnan [*1] **Akhil Arunkumar** [2] **Gaurav Jain** [2]
Prashant J. Nair [1] **Ilya Soloveychik** [2] **Purushotham Kamath** [2]

ABSTRACT

Transformers have emerged as the underpinning architecture for Large Language Models (LLMs). In generative language models, the inference process involves two primary phases: prompt processing and token generation. Token generation, which constitutes the majority of the computational workload, primarily entails vector-matrix multiplications and interactions with the Key-Value (KV) Cache. This phase is constrained by memory bandwidth due to the overhead of transferring weights and KV cache values from the memory system to the computing units. This memory bottleneck becomes particularly pronounced in applications that require long-context and extensive text generation, both of which are increasingly crucial for LLMs.

This paper introduces "Keyformer", an innovative inference-time approach, to mitigate the challenges associated with KV cache size. Keyformer leverages the observation that approximately 90% of the attention weight in generative inference focuses on a specific subset of tokens, referred to as "key" tokens. Keyformer retains only the key tokens in the KV cache by identifying these crucial tokens using a novel score function. This approach reduces both the KV cache size and memory bandwidth usage without compromising model accuracy. We evaluate Keyformer's performance across three foundational models: GPT-J, Cerebras-GPT, and MPT, which employ various positional embedding algorithms. Our assessment uses a variety of tasks, with an emphasis on summarization and conversation tasks involving extended contexts. We show that Keyformer reduces inference latency by 2.1× and improves token generation throughput by 2.4×, while preserving the model's accuracy.

1 INTRODUCTION

Transformers have proven to be particularly successful in tasks such as language modeling (Lewis et al., 2019; Brown et al., 2020; Raffel et al., 2020), image recognition (Dosovitskiy et al., 2020), recommendations (Sun et al., 2019; de Souza Pereira Moreira et al., 2021; Adnan et al., 2023; Zhao et al., 2023), and text generation with the advent of Large Language Models (LLMs). Unfortunately, LLM deployment presents critical *inference latency and throughput* concerns. This is primarily attributed to the sequential autoregressive nature of generative inference, particularly when handling inputs with larger contexts. Despite advancements, modern LLMs face challenges in efficiently processing longer input sequences, as evidenced by recent studies (Bai et al., 2023; Li et al., 2023; Chen et al., 2023; Huang et al., 2021a). Unfortunately, the increased memory and compute requirements associated with longer sequences exacerbate LLM inference latency and reduce throughput. This paper proposes inference-time strategies for accuracy-preserving, low-latency, high-throughput LLM systems.

LLMs employ Transformers and rely on the 'attention mechanism' to understand the relationships between words within a given input sequence (Vaswani et al., 2017). As the attention mechanism scales quadratically with the size of the input sequence, it tends to present the largest latency overhead during inference (Sukhbaatar et al., 2019; Dao et al., 2022; Choromanski et al., 2020). Additionally, due to the autoregressive nature of token generation in LLMs, there is a need to *recompute* key and value vectors for all previous tokens. To mitigate this, LLMs utilize a storage structure known as a Key-Value Cache (KV cache) (Ott et al., 2019). KV cache retains previously computed key-value pairs, eliminating the need for costly re-computation of these vectors.

However, KV cache presents scalability challenges. Accessing the KV cache from off-chip memory during token generation introduces additional memory latencies and is constrained by memory bandwidth limitations. For instance, in the MPT-7B model illustrated in Figure 1(a), increasing the sequence length by 16× (from 512 to 8K) results in a

[*]Work done when the author was an intern at d-Matrix. [1]Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada [2]d-Matrix, Santa Clara, California, USA. Correspondence to: Muhammad Adnan <adnan@ece.ubc.ca>.

Proceedings of the 7^{th} MLSys Conference, Santa Clara, CA, USA, 2024.

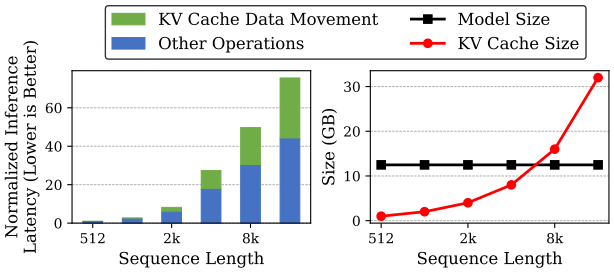


Figure 1. (a) Inference latency normalized to sequence length of 512. We measure the KV cache data movement for MPT-7B (MosaicML, 2023) model with varying sequence length (50% context + 50% text generation). (b) The KV cache size and model size as sequence length varies. The studies are performed on an NVIDIA A100 GPU with a batch size of 1 and beam size of 4.

more than *50× increase* in inference latency. Moreover, approximately 40% of the total inference time (highlighted in green) is consumed by KV cache data movement. Importantly, a larger context not only increases the size of the KV cache but also prolongs the time required for other operations (depicted in blue). Similarly, as shown in Figure 1(b) for the MPT-7B model, the KV cache size surpasses the model size when the sequence length exceeds 8K. Thus, KV cache sizes present a roadblock to enabling low-latency, high-throughput inference for large sequences.

Previous studies have explored mitigating attention mechanisms' memory and computation requirements when dealing with longer sequences (Zaheer et al., 2020; Kitaev, 2020; Wang et al., 2020; Beltagy et al., 2020). While system-level optimizations like FlexGen (Sheng et al., 2023), Flash Attention (Dao et al., 2022), Paged Attention (Kwon et al., 2023), and multi-dimensional partitioning (Pope et al., 2023) aim to improve the scalability of generative AI, they often overlook the fundamental challenge of expanding KV cache size. Techniques like multi-query (Shazeer, 2019) and group-query attention (Ainslie et al., 2023) propose reducing KV cache size by eliminating specific attention heads from writing to the KV cache, but these methods typically require *resource-intensive model retraining or fine-tuning*. This becomes complex as various accelerators are already deployed in the field. Thus, there is a pressing need for inference-time techniques for KV cache reduction. This is even more challenging as any proposed technique must conform to the strict constraints for model accuracy. For instance, *MLPerf (Reddi et al., 2020) mandates that any optimization applied to LLMs maintain a model accuracy between 99% to 99.9% of the baseline*.

To address these concerns, we introduce Keyformer [1], a novel method for dynamically reducing the KV cache size during inference. Keyformer does this by intelligently discarding unnecessary tokens without losing accuracy. The critical insights of Keyformer are demonstrated in Figure 2,

[1] https://github.com/d-matrix-ai/keyformer-llm.

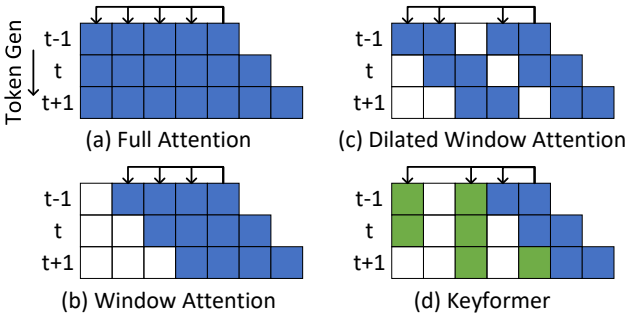
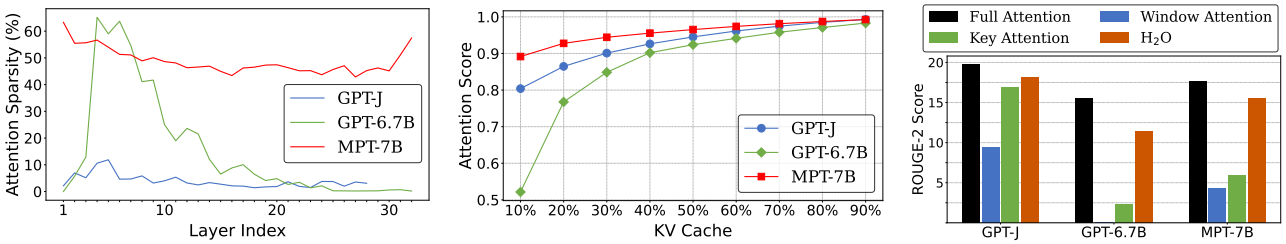


Figure 2. Attention block for generative inference. (a) Full attention with current token attending all previous tokens. (b) Window attention ($w = 4$): Focusing on the most recent 4 tokens. (c) Dilated window attention ($w = 4$, dilation = 1). (d) Keyformer ($w = 2, k = 2$): A mix of recent window (w) and key tokens (k). White color indicates no attention, while blue color indicates attention. The green color identifies the key tokens and their respective attention. The values of the three consecutive token generation iterations are $t-1, t, t+1$.

where we also compare with existing state-of-the-art attention techniques for inference optimizations. Figure 2(a) illustrates the traditional 'Full Attention' (Brown et al., 2020) mechanism, where each newly generated token attends to *all* preceding tokens in the sequence. Figure 2(b) depicts 'Window Attention,' (Child et al., 2019) which maintains a fixed-size *sliding window* of recent tokens, thereby reducing the size of KV cache. However, this method restricts the model's capacity to capture comprehensive semantic information from the past, leading to lower-quality text generation and decreased accuracy. Figure 2(c) presents a variant called 'Dilated Window Attention,' with similar accuracy limitations to windowed attention.

To address this, Keyformer leverages the insight that certain tokens carry more significance than others. Specifically, it observes that *nearly 90% of the attention weight focuses on a small subset known as* key tokens. These tokens are crucial for LLMs to grasp context but may fall outside the sliding window of window attention. Keyformer introduces a mixed attention approach, depicted in Figure 2(d), which combines recent tokens with the preceding key tokens when generating the next token. Our experiments show that Keyformer demonstrates significant improvements over state-of-the-art methods such as H_2O (Zhang et al., 2023). This is because, unlike H_2O, which identifies "heavy hitters" solely based on attention scores, Keyformer considers the importance of discarded tokens in identifying key tokens. *We have open-sourced* Keyformer *code.*

We evaluate Keyformer on multiple models, including GPT-J (Wang & Komatsuzaki, 2021), Cerebras-GPT (Dey et al., 2023), and MPT (MosaicML, 2023), across various tasks like summarization and conversation for long sequences. Even with a 50% reduction in KV cache, Keyformer preserves accuracy while reducing inference latency by 2.1× and boosting token generation throughput by 2.4×.



(a) Attention Sparsity (b) Average Attention Score (c) Accuracy

Figure 3. (a) Default attention sparsity across different models. (b) Average attention score of three different models with 90% of attention score dedicated to 40% of the tokens called key tokens. (c) Accuracy comparison of three models with different attention schemes. 'Full Attention' uses the full KV cache size, while 'Window Attention' and 'H_2O' use 50% of the KV cache size. All models use the CNN/DailyMail (See et al., 2017) dataset for the summarization task.

2 Background and Motivation

2.1 Inference Process in Large Language Models

In language modeling, the task involves estimating the probability of the next token based on preceding tokens $x_1, x_2, \ldots, x_n$. For generative Large Language Models (LLMs), the inference process unfolds in two phases:

1. **Prompt Processing Phase:** This phase helps input context undergo causal processing, enabling the model to generate keys and values for all tokens within the context. These key-value pairs are then stored in the KV cache.
2. **Token Generation Phase:** This phase sequentially and auto-regressively generates text. Each token is produced by passing through all layers of the generative model. Notably, the generation of the next token relies on the previously generated tokens and their order.

To enhance inference efficiency, repeated and complex computations of Key (K) and Value (V) tensors across all layers are avoided by caching these tensors. This is referred to as the KV cache. The KV cache is sequentially populated during each token generation step (Strati et al., 2024), until the text generation process is completed.

2.2 Reducing KV Cache Size by Exploiting Sparsity

To address the challenge posed by the expanding KV cache, let us examine a sequence, denoted as S_n, comprising n tokens and its KV cache contents for a single attention head and layer. In full attention, the KV cache components involve n keys and values. These grow proportionally with S_n. To mitigate this, we can shrink the KV cache size to accommodate shorter sequences, designated as S_k. This involves using a reduced number of tokens, transitioning from n to k, where S_k is a subset of S_n, and k is less than n. This reduction can be achieved by leveraging the inherent sparsity within the attention mechanism of LLMs.

Despite the substantial computational demands during the training of transformers, there exists inherent sparsity within the attention mechanism. However, the extent of sparsity may vary depending on the particular downstream task. Figure 3a illustrates the diverse levels of attention sparsity among different models utilized for summarization tasks with the CNN/DailyMail dataset. This variability manifests across various levels of the model, including the overall model, individual layers, and distinct sections of the model.

2.3 Improving Performance by Using Key Tokens

In Figure 3b, the Cumulative Distribution Function (CDF) depicts the relationship between attention score and the fraction of the total context. Notably, a small subset of tokens receives the most attention during text generation. This underscores the significance of specific key tokens and their pivotal role in comprehending context and facilitating text generation. However, dynamically determining which tokens serve as key tokens, especially in cases where the input sequence contains unknown or unseen tokens during inference, presents a considerable challenge.

2.3.1 *Leveraging Score Function to Identify Key Tokens*

We introduce a score function f_θ for each token to identify the k key tokens out of a total of n tokens. In the multi-head attention mechanism, attention scores determine the degree of connection between a single token and all other tokens. This is described by Equation 1.

$$\text{Attention Score} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{1}$$

The natural choice is to utilize the attention score as the score function, denoted as $f_\theta(\text{acc}; \text{attn})$. This method is commonly observed in previous state-of-the-art work, such as H_2O (Zhang et al., 2023). It identifies tokens that consistently receive higher attention scores during the prompt and token generation phases as the most critical or key tokens.

We can choose and retain these k tokens based on their accumulated attention scores, creating what we refer to as

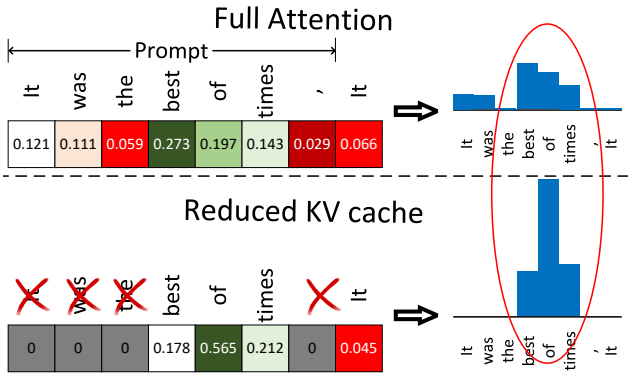


Figure 4. Reducing the KV cache introduces a change in the distribution of attention scores. As tokens are removed, their distribution becomes uneven among the remaining cached tokens. Thus, it affects the identification of key tokens according to the score function $f_\theta(acc\ attn)$. The figure shows this effect for the attention scores for the MPT-7B (MosaicML, 2023) model with a 50% reduction in KV cache.

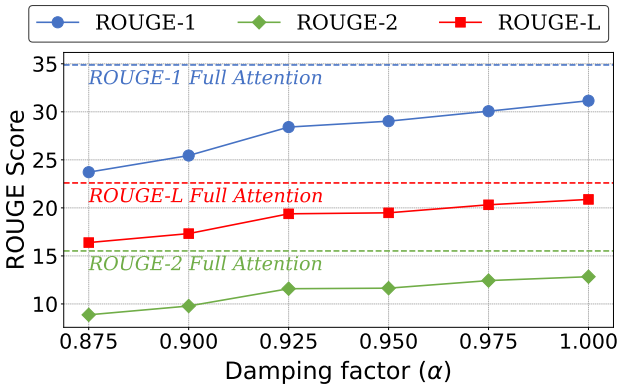


Figure 5. The effect of damping on the model quality for Cerebras-GPT-6.7B model with 50% KV cache reduction. Even after damping the score function to counteract the excess attention score, one does not achieve the model quality of the full attention model.

"Key Attention". However, relying solely on these k tokens during attention does not provide the necessary accuracy and yields poor performance. This is shown in Figure 3c.

In this comparison, both 'Window Attention' and 'Key Attention' demonstrate inferior performance compared to full attention, even when the window and key tokens parameters are reduced by $^n/_2$. While reducing the sizes of the window and key tokens relative to the total tokens (n) is crucial for minimizing the size of the KV cache, it also leads to a significant decrease in accuracy. This decline primarily stems from the loss of recent context in key-token attention and crucial context in window attention. Building on this observation, we propose a mixed approach that combines selected key tokens with recent tokens to reduce the KV cache size while also preserving accuracy.

2.3.2 *Problem: Uneven Score Distribution*

Figure 4 shows the distribution of attention scores (f_θ) for full attention, as described in Equation 2. When KV cache is reduced, tokens with lower scores are discarded. This alters the score function, shown in Equation 3, as the term $\sum_{m=n-k}^{n} e^{x_m}$ becomes zero.

$$f_\theta(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \quad i = 1, 2, \ldots, n \tag{2}$$

$$f_\theta(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j} + \cancel{\sum_{m=n-k}^{n} e^{x_m}}} \tag{3}$$

This removal of tokens disrupts the distribution of the score function. This is because the attention weight of the discarded tokens is unevenly distributed among the tokens within the reduced KV cache. This uneven distribution arises due to the nature of the inherent *softmax* function.

Figure 4 illustrates this phenomenon by comparing the distribution of the score function for full attention with that after KV cache reduction. When the score distribution is uneven, the model may not attend to the most relevant tokens in the sequence. Consequently, this can result in a loss of contextual information, reduced accuracy, and potentially lower-quality text generation.

2.3.3 *Motivation: Damping the Score Function*

A straightforward approach involves damping the score function using a damping factor to counteract the excess attention score resulting from discarded tokens. Assume α is the damping factor. It modifies the score function as $\bar{f}_\theta = \alpha f_\theta$. Ideally, we aim to dampen the score function by a factor equivalent to $\sum_{m=n-k}^{n} e^{x_m}$, where $n - k$ represents the tokens that have been discarded.

Figure 3b shows that, even with a 50% reduction in the KV cache size, the average accumulated attention score of key tokens remains consistently high, ranging from approximately 90% to 95%. We conduct a sweep across a range of values to explore the impact of different damping factors (α) on overall model quality. This analysis is performed with a KV cache size set at 50% and a recent ratio of 20% (representing the percentage of recently generated tokens) for the Cerebras-GPT-6.7B (Dey et al., 2023) model.

However, as depicted in Figure 5, even after the application of a damping factor, it is not possible to achieve the same quality as the full attention model. This discrepancy stems from a significant change in the score distribution of the remaining tokens within the reduced KV cache. These findings underscore the inadequacy of relying solely on the accumulated attention score-based score function $f_\theta(\text{acc}; \text{attn})$ for identifying key tokens. Hence, addressing the impact of discarded tokens within the score function is crucial to achieve higher model accuracy or meet the accuracy requirements of benchmarks like *MLPerf*.

3 KEYFORMER: INTUITION AND DESIGN

Keyformer leverages the inherent sparsity within decoder layers by identifying key tokens using a *mixture of recent tokens*. It adjusts the changes in the score function resulting from discarded tokens by applying regularization to the unnormalized logits for the identification of key tokens.

3.1 Logits Regularization

We strategically remove $n - k$ tokens from the context in the prompt processing phase. This helps us maintain a constant KV cache size with k tokens during generation and prevents unwarranted memory expansion. Thereafter, Keyformer uses logits regularization technique. The introduction of added distribution (ζ) to regularize the reduced logits enables our model to remain robust and adaptive. It helps identify the key tokens even in the presence of unknown contexts during inference-time. Keyformer adds this noise to the unnormalized logits derived from QK^T, as illustrated in Equation 4. Also, the type of distribution added significantly impacts the resulting probability distribution.

$$y_i = x_i + \zeta_i, \quad where\ x_i = \frac{Q[i,:]K[:,i]^T}{\sqrt{d_k}} \tag{4}$$

Here, y_i are the adjusted logits, x_i are the unnormalized logits, and ζ_i is the added distribution for regularization.

3.2 Choice of Distribution for Regularization

The regularization distribution added to unnormalized logits impacts key tokens identification and model quality. Thus, we aim to draw intuition using the semantics of LLMs.

3.2.1 Intuition: Bias Towards Initial Tokens

Previous research, such as streaming LLMs (Xiao et al., 2023) and the H_2O model (Zhang et al., 2023), has shown a bias towards initial tokens. This bias stems from accumulated attention scores favoring initial tokens due to cumulative effects during decoding iterations. We propose using a skewed distribution to leverage this bias and effectively model the distribution of maximum values (key tokens). This distribution favors initial tokens while maintaining an asymmetric profile, enhancing the representation of tokens drawn from the recent context window.

Gumbel Logit Adjustment: Our choice of distribution is inspired by the *Gumbel distribution* (Cooray, 2010). The Gumbel distribution is particularly well-suited for our key tokens identification task, as it characterizes the distribution of maximum values within a set of samples and is skewed towards initial tokens. This makes it an ideal candidate for modeling key tokens for long sequences.

$$f_{Gumbel}(\zeta_i) = e^{-\zeta_i - e^{-\zeta_i}} \tag{5}$$

$$f_{Gumbel}(y_i) = e^{-(y_i - x_i) - e^{-(y_i - x_i)}} \tag{6}$$

Equation 5 presents the standard Gumbel pdf applied to unnormalized logits, while Equation 6 displays the pdf of logits adjusted with Gumbel addition. Additionally, it is noteworthy that the Gumbel distribution holds significance in statistical theory. It captures the essence of the Gumbel limit theorem, which asserts that common probability distributions (such as normal, exponential, uniform, etc.) converge to the Gumbel distribution. This underscores its appropriateness for modeling the identification of key tokens.

In theory, selecting a regularization distribution that promotes uniformity after normalization aids in key tokens identification. This is crucial during inference when information about discarded tokens is unavailable. To quantify the spread of probability distributions post-normalization, we employ entropy, defined as $H(p) = -\sum p_i \log(p_i)$. Our analysis indicates that Gumbel-based logit adjustment fosters a more uniform distribution, suggesting its effectiveness as a regularization technique for key tokens identification, as demonstrated in Equation 7 and Equation 8.

$$\mathbf{z} = \text{softmax}(\mathbf{y}) \tag{7}$$

$$H(\mathbb{E}[\mathbf{z}_{Gumbel}]) > H(\mathbb{E}[\mathbf{z}]) \tag{8}$$

3.3 Keyformer Score Function

We propose a novel score function for Keyformer, denoted as $f_\theta(\text{Keyformer})$, to address the limitations of the accumulated attention-based score function ($f_\theta(\text{acc attn})$). This new score function integrates the Gumbel noise distribution into the unnormalized logits. However, it fails to account for the discarded tokens in forming the underlying probability distribution. To rectify this, we introduce a temperature parameter, denoted as τ, as shown in Equation 9.

$$f_\theta(x_i) = \frac{e^{(x_i + \zeta_i)/\tau}}{\sum_{j=1}^{k} e^{(x_j + \zeta_j)/\tau}}, \quad i = 1, 2, \ldots, k \tag{9}$$

The probabilistic score functions described above are akin to the concept of *Gumbel Softmax* (Jang et al., 2016). This score function offers a continuous relaxation of discrete random variables (Maddison et al., 2016). This alignment corresponds with our *primary objective of identifying a subset of past tokens* $S_k \subset S_n$ that conveys the same semantic information as the original complete set of tokens.

3.3.1 Significance of the Temperature Parameter (τ)

The 'temperature' parameter (τ) is pivotal in regulating the smoothness of the probabilistic distribution. Higher values of τ ($\tau \to \infty$) yield uniform probabilities, assigning equal

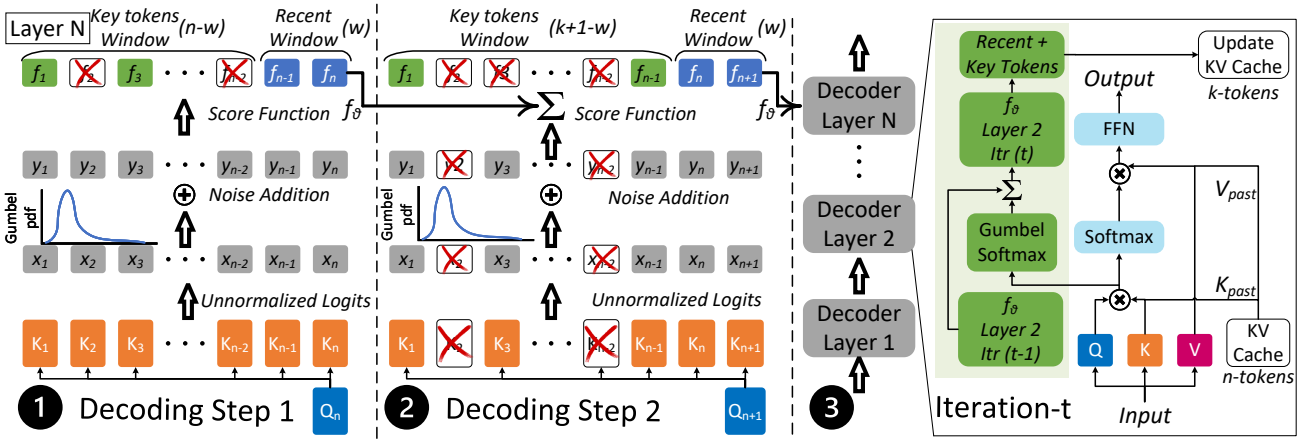


Figure 6. The overview of Keyformer: ❶ Initial decoding step involves token generation with n tokens in KV cache. Keyformer induces noise for key tokens identification, selecting w tokens from the recent window and $(k-w)$ tokens from the remaining $(n-w)$ to maintain (k) tokens in KV cache. ❷ Subsequent decoding step uses the reduced KV cache from the previous iteration. ❸ The design of Keyformer from the perspective of a single decoder layer involves taking unnormalized logits from QK^T and introducing 'Gumbel' noise. This is done using a Gumbel-based probability distribution and helps address the issue of key tokens being discarded. The score function (f_θ) accumulates over decoding steps for each layer and head.

scores to all tokens. Conversely, lower values of τ $(\tau \to 0)$ produce a sharper distribution, prioritizing specific tokens based on their unnormalized logits. This parameter governs the degree of randomness in probabilities. It is crucial when tokens are removed from the KV cache, as they cannot be reintroduced without recomputing their keys and values.

In Equation 10, we illustrate the dynamic nature of τ at each decoding iteration t. To achieve this, we define a range for τ spanning from τ_{init} to τ_{end}. In each decoding step, we increment τ by $\Delta\tau$, a value determined by the range of τ and the length of the text being generated, denoted as T.

$$\tau = \tau_{init} + t\Delta\tau, \quad \Delta\tau = \frac{\tau_{end} - \tau_{init}}{T} \tag{10}$$

This strategy is based on the premise that we need a more uniform or randomized probability distribution as more tokens are discarded. Through empirical analysis, we discovered that setting $\tau_{init} = 1$ and $\tau_{end} = 2$ produces optimal outcomes (refer to Appendix A.8). This decision aligns with our objective of maintaining a non-random score function during the prompt phase, where all tokens are available. When τ is set to one, the Gumbel softmax approach is nearly equivalent to a standard *softmax*. As we advance through decoding iterations and discard more tokens to maintain a static KV cache size, we systematically increase the randomness in our score function f_θ. This is achieved by incrementally raising τ with $\Delta\tau$.

3.3.2 *Leveraging Score Function Accumulation*

The accumulation of the score function is essential for identifying key tokens based on their consistent behavior throughout decoding steps. Without accumulation, token discarding would rely solely on the current token's correlation with previous tokens. Although the correlation of the current token is significant in identifying key tokens, their behavior should remain consistent across most generated tokens. To discern key tokens based on this consistent behavior, we accumulate the score function (f_θ) across both the prompt and token generation phases, as depicted in Figure 6.

3.4 Keyformer Algorithm

Figure 6 presents an overview of Keyformer. We highlight its key functionalities in discarding tokens based on sparsification, using a mixture of recent and key tokens, and introducing a novel Gumbel softmax-based score function for key tokens identification. During the prompt processing phase, Keyformer calculates keys and values for all n tokens within the prompt length S_n to predict the first token. Given the KV cache budget, Keyformer retains a recent window of w recent tokens while discarding $n-k$ tokens from the $n-w$ tokens window, thereby identifying $k-w$ tokens. The top-$(k-w)$ tokens from the $n-w$ window are selected based on the Keyformer score function. The combination of key tokens $(k-w)$ and recent tokens (w) forms the reduced KV cache. As there are no discarded tokens during the prompt processing phase, Keyformer uses a temperature parameter, $\tau_{init} = 1$, to approximate the *softmax* probability distribution. This is illustrated in decoding step 1.
In the token generation phase, Keyformer operates with a reduced KV cache. The first generated token attends solely to the k tokens within the KV cache, as depicted in decoding step 2. The recent window w shifts right by a single token, while the score function f_θ accumulates with the score function from the previous decoding step. During each decoding step of the token generation phase,

Algorithm 1 Keyformer

Input: KV cache size: k
Input: Recent Window: w
Input: Text Generation Length: T
Input: Temperature Parameter: τ_{init}, τ_{end}, $\Delta\tau$
Input: Prompt Sequence Length: S_n
Output: Reduced Sequence Length: S_k
Initialize $f_\theta \leftarrow \phi$, $\tau \leftarrow \tau_{init}$, $\Delta\tau \leftarrow \frac{\tau_{end}-\tau_{init}}{T}$
Initialize $\zeta_i \leftarrow$ *Gumbel Distribution*
for $t = 0$ **to** T **do**
  $\tau \leftarrow \tau_{init} + t\Delta\tau$
  if phase $\leftarrow$ prompt **then**
    $x_i \leftarrow \frac{Q_i K^T_{S_n}}{\sqrt{d}}, m \leftarrow n$
  else
    $x_i \leftarrow \frac{Q_i K^T_{S_k}}{\sqrt{d}}, m \leftarrow k$
  end if
  $f_\theta(i) \leftarrow f_\theta(i) \; + \; \frac{e^{(x_i+\zeta_i)/\tau}}{\sum_{j=1}^{m} e^{(x_j+\zeta_j)/\tau}}$
  $S_w \leftarrow$ *Recent w tokens*
  $S_{key} \leftarrow \arg\max_{(k-w)} f_\theta[:-w]$
  $S_k \leftarrow S_{key} \cup S_w$
end for

$k - w$ key tokens are identified from a window of size $k + 1 - w$. Consequently, one token has been added, and another has been removed from the recent window. Since we add and remove tokens from the 'key tokens' window, we can improve accuracy while maintaining a static KV cache size equal to S_k. Moreover, the temperature parameter τ increases by $\Delta\tau$ to adjust for the number of removed tokens in the probability distribution of the score function. The detailed algorithm for Keyformer is provided in Algorithm 1.

4 Evaluation

We evaluate Keyformer across three significant model families: GPT-J (Wang & Komatsuzaki, 2021), Cerebras-GPT (Dey et al., 2023), and MPT (MosaicML, 2023), each using distinct position encoding techniques. GPT-J incorporates RoPE (Su et al., 2022), Cerebras-GPT employs learnable position embeddings, and MPT utilizes ALiBi (Press et al., 2021). By including models with varied position encoding methods, we ensure the robustness and generalizability of our findings across representative model families. We employed a fixed beam size of 4 for all evaluations.

Setup: We conducted evaluations on two representative text generation tasks: summarization, utilizing the CNN/DailyMail (See et al., 2017) and GovReport (Huang et al., 2021b) datasets, and conversation, employing the SODA dataset (Kim et al., 2022). The GPT-J model was fine-tuned specifically for summarization, while Cerebras-GPT and MPT are pre-trained models. We utilized the MPT-chat version of the MPT model for conversation tasks, which was fine-tuned for dialogue generation. All models were pre-trained with a sequence length of 2k.

To address long document summarization, we utilized the MPT-storywriter version of the MPT model, fine-tuned for writing fictional stories. This model accommodates a context length of 65k and can generate content up to 84k tokens long. Additionally, we evaluated four tasks from the lm-eval-harness (Gao et al., 2021) framework: PIQA (Bisk et al., 2020), Winogrande (Sakaguchi et al., 2021), OpenBookQA (Mihaylov et al., 2018), and COPA (Roemmele et al., 2011). These tasks involve few-shot evaluation of autoregressive language models and were executed using the NVIDIA A100 (80GB) GPUs.

Baselines: To evaluate the accuracy of Keyformer, we compared it against Full Attention. Full Attention acts as our benchmark and represents the gold standard for accuracy. We aim to achieve an accuracy target within the range of 99% to 99.9% of Full Attention. This goal aligns with the high-quality standards set by industry benchmarking entities like MLPerf (Reddi et al., 2020). Additionally, we performed comparisons with Window Attention and the recent H_2O model (Zhang et al., 2023), adjusting the KV cache size from 20% to 90% of the prompt length.

4.1 Accuracy Results

To assess the impact of KV cache reduction on text generation quality, we relied on the ROUGE score (Lin, 2004), a widely-used metric for evaluating fluency and coherence. ROUGE measures the overlap of n-grams between generated and reference text, providing a standardized measure for text quality. According to MLPerf, ROUGE scores, including *ROUGE-1, ROUGE-2, and ROUGE-L*, should reach 99% to 99.9% of their original values for summarization tasks. Thus, even with reduced KV cache, our model should maintain the desired ROUGE scores. Figure 7 depicts accuracy comparisons between Keyformer and other methods (Full Attention, Window Attention, and H_2O) across different KV cache sizes. The illustration focuses on the ROUGE-2 score, which measures bi-gram overlap. Trends for ROUGE-1 and ROUGE-L are detailed in Appendix A.5.

The results highlight the importance of the previous context for model performance. For instance, Window Attention relies solely on recent tokens, leading to a significant loss of accuracy. Thus, identifying key tokens is crucial for achieving desired model accuracy. Across various KV cache budgets, Keyformer consistently outperforms the state-of-the-art H_2O. It shows that the key tokens it identifies are more important than the heavy hitters identified by H_2O. For instance, Keyformer attains the target ROUGE score

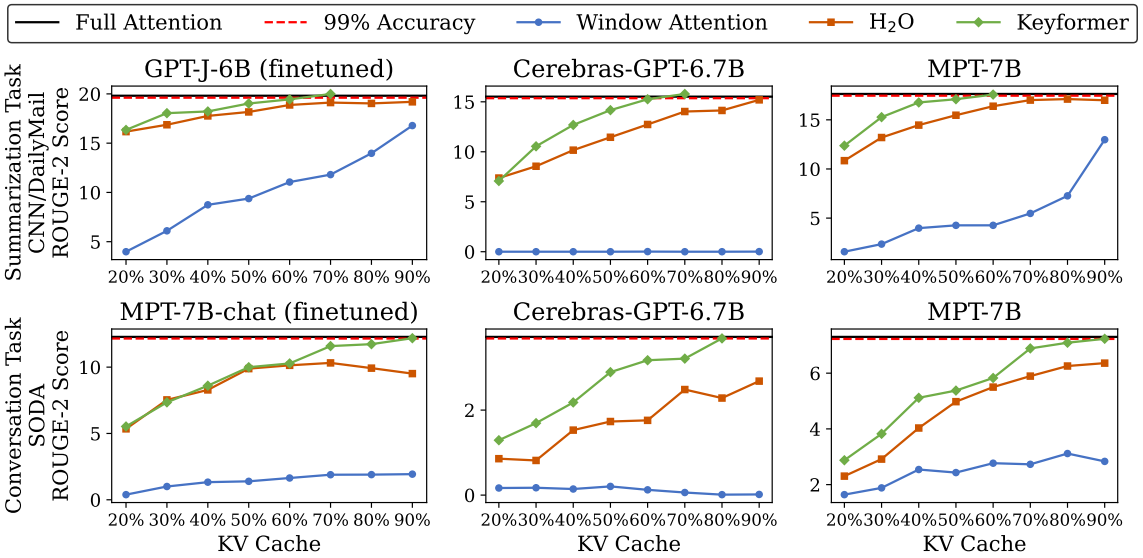


Figure 7. The accuracy comparison involves Full Attention, Window Attention, H_2O, and Keyformer with different KV cache sizes. The solid black line represents Full Attention without discarding tokens and with a full KV cache. The red dotted line denotes the 99% accuracy threshold, aligning with the MLPerf guidelines (Reddi et al., 2020). Despite using 90% KV cache, both Window Attention and H_2O fall short of the desired accuracy. In contrast, Keyformer achieves baseline accuracy with only 70% of the KV cache size.

with just 70% of the KV cache, whereas H_2O fails to reach this goal even with a larger KV cache budget. Furthermore, Keyformer surpasses the baseline accuracy by up to 1.73% (0.9% for GPT-J-6B and 1.73% for Cerebras-GPT-6.7B for summarization task) achieved with full attention. This demonstrates the regularization effect of the introduced Gumbel noise in the score function of Keyformer and its positive impact on key tokens identification.

Long Context Summarization: We assessed the effectiveness of KV cache reduction in Keyformer while maintaining accuracy for handling long contexts. This evaluation was conducted on the MPT-7B-story writer model, pre-trained with a context length of 65k. We utilized the Government report (Huang et al., 2021b) dataset, which contains reports authored by government research agencies and features longer summaries and documents. This dataset requires a deep understanding of context to extract crucial information for summarization. Figure 8 shows the accuracy comparison among Keyformer, H_2O, and Full Attention. Notably, even with a 50% KV cache size, Keyformer maintains the desired 99% accuracy threshold, while H_2O shows significantly lower accuracy at the same KV cache size.

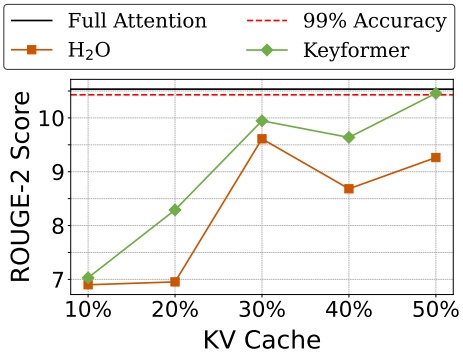


Figure 8. Evaluating long context summarization using MPT-7B-storywriter for GovReport dataset with 8k sequence length.

4.2 Performance Results

To assess the performance advantages of Keyformer with reduced KV cache, we considered two critical inference metrics: inference latency and generation throughput for the target models. Our Keyformer implementation is seamlessly integrated with Huggingface (Wolf et al., 2019) model cards, ensuring ease of adoption. We disabled CPU offloading in cases where the model and KV cache exceeded GPU HBM memory capacity, ensuring consistent evaluation. We generated a synthetic dataset to maintain evaluation consistency, where all prompts were padded with synthetic text. We employed the MPT-7B-storywriter model to generate an equal number of tokens for each prompt. We tested various combinations of prompt and generation lengths.

Figure 9 presents inference latency speedup while Table 1 shows improvement in generation throughput in comparison to a Full Attention-based method. With a 50% KV cache reduction, Keyformer significantly reduces inference latency, achieving 2.1× with the same batch size. Moreover, the reduced KV cache size allows Keyformer to handle twice the batch size compared to full attention, increasing token generation throughput by 2× with the same batch size and 2.4× with a bigger batch size.

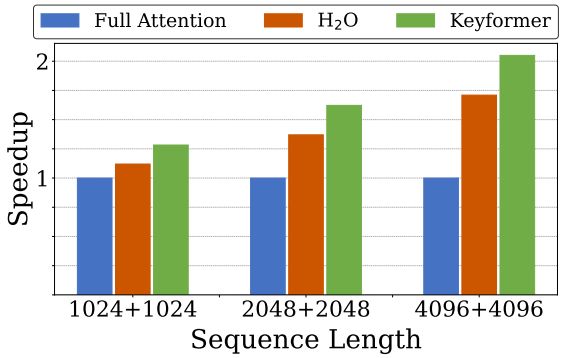


Figure 9. Speedup of inference in an iso-accuracy setting for the MPT-7B model. Here, Keyformer reduces KV cache by 50% and H_2O reduces KV cache by 10%. H_2O falls short of baseline accuracy with a 50% KV cache.

Table 1. The generation throughput (tokens/sec) for the MPT-7B model across different sequence lengths. "1024 + 1024" shows the sum of the prompt length and the token generation length. "OOM" stands for out-of-memory and "BS" for batch size.

Sequence Length	**Full Attention** Original cache	**H_2O** 90% KV cache	Keyformer 50% KV cache
1024 + 1024	24.9	27.8	32.0
2048 + 2048	15.0	20.5	24.3
4096 + 4096 (BS=1)	8.3	14.1	17.0
4096 + 4096 (BS=2)	OOM	OOM	19.85

Performance Improvement Breakdown: To understand the sources of performance benefits with Keyformer-based KV cache reduction, we primarily consider two factors:

1. **Reduced KV cache**: A smaller KV cache significantly reduces the data movement from off-chip GPU HBM.
2. **Scaled Dot Product Optimization**: The number of tokens in the KV cache is reduced from n to k.

The above two factors reduce the overall smaller size of matrices. Thus, they enable an optimized scaled dot product within the multi-head attention block $(QK^T)V$.

It is worth noting that in LLMs, which are memory-bound, the main performance boost comes from reducing KV cache data movement rather than matrix multiplication. However, Keyformer's key tokens identification process introduces some overhead due to Gumbel softmax. Figure 10 illustrates the normalized performance improvement for Keyformer, considering both reduced KV cache data movement and optimized scaled dot product. These enhancements are demonstrated for the MPT-7B-storywriter model with a 50% KV cache reduction, including the additional overhead from Keyformer's Gumbel softmax. The results indicate that Keyformer-based KV cache reduction decreases KV cache data movement by 2.9× and improves computational efficiency in the attention module's scaled dot product by 1.3×, particularly for sequences of length 4k.

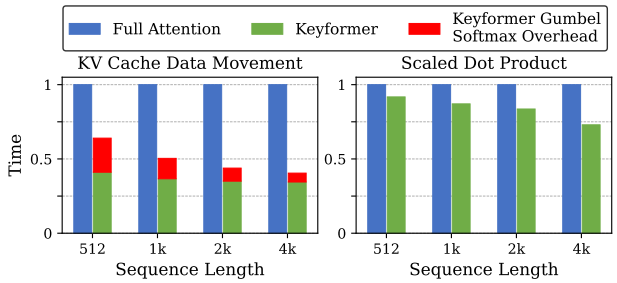


Figure 10. Normalized time for KV cache data movement and Scaled Dot Product $(QK^T)V$ with Keyformer 50% KV cache reduction along with its score function overhead.

Table 2. Few shot results for different tasks. H_2O and Keyformer are with 50% KV cache.

Task	**Attention Method**	**Cerebras-GPT-6.7B 0-Shots**	**Cerebras-GPT-6.7B 5-Shots**	**MPT-7B 0-Shots**	**MPT-7B 5-Shots**
COPA	Full	73.0	73.0	80.0	83.0
	H_2O	68.0	74.0	75.0	82.0
	Keyformer	70.0	74.0	76.0	84.0
OpenBookQA	Full	34.8	36.8	41.8	43.4
	H_2O	32.2	38.0	37.6	44.0
	Keyformer	32.6	38.6	40.6	43.2
Winogrande	Full	60.2	58.4	68.7	71.8
	H_2O	56.7	59.5	63.2	72.1
	Keyformer	57.9	59.7	64.1	72.3
PIQA	Full	74.2-	74.4	79.9	80.7
	H_2O	72.9	73.9	79.4	79.8
	Keyformer	73.0	73.6	79.2	80.1

4.3 Few-Shot Evaluation

We performed few-shot experiments using four tasks from the `lm-eval-harness` framework and pre-trained models to evaluate Keyformer's performance under varying numbers of shots during inference. Table 2 presents the results for 0 and 5 shots, demonstrating that Keyformer consistently surpasses previous approaches across all tasks and shot settings. Even with a 50% reduction in KV cache size, it achieves accuracy close to the full attention baseline.

4.4 Ablation Studies

4.4.1 Shared versus Per-Layer Score Function

The score function f_θ defines what constitutes a key tokens in the context. In generative LLMs with stacked decoder layers, the score function (f_θ) can either be shared across all layers (*Shared*) or dedicated to each layer (*Per-Layer*). In the *Per-Layer* approach, f_θ(Per-Layer) assigns a dedicated score function to each decoder layer, with accumulation occurring at each decoding stage. Conversely, f_θ(Shared) uses a single global score function for all decoder layers, with accumulation across decoder layers and decoding stages.

Table 3 illustrates the accuracy comparison between Per-Layer and Shared score functions, maintaining the original positional information and KV cache size constant. Notably, using the Per-Layer score function yields better accuracy

Table 3. ROUGE Score comparison with different methods and score functions for summarization task using the CNN/DailyMail dataset.

Model	Attention Method	Score fn f_θ	KV Cache Size	ROUGE-1	ROUGE-2	ROUGE-L
MPT-7B	Full	-	Original	38.6373	17.6329	24.506
	Full (99% Accuracy)	-	Original	38.2509	17.4565	24.2609
	Window	-	60%	18.1296	4.2655	11.5288
	H_2O	Per-Layer	60%	36.9616	16.3865	24.2301
	StreamingLLM	-	60%	1.3572	0.0179	1.0281
	Keyformer *(New Pos)*	Per-Layer	60%	36.9152	16.9092	23.7218
	Keyformer ***(Org Pos)***	Per-Layer	**60%**	**38.7134**	**17.5976**	**24.5724**
	Keyformer *(Org Pos)*	Shared	60%	38.2537	17.3732	24.2579

than the shared score function. This aligns with the intuition that transformers learn hierarchical text representations across layers, with lower layers capturing local syntactic and semantic features and higher layers capturing more abstract and complex patterns (Geng et al., 2023). Therefore, having a Per-Layer score function for each layer aids in key tokens identification specific to that layer.

4.4.2 *New vs. Original Positional Information*

We investigated how reducing the KV cache size affects the positional information used for the keys in Keyformer. We explored two approaches: Keyformer *(Org Pos)* and Keyformer *(New Pos)*. In Keyformer *(Org Pos)*, the position information reflects the original positions of tokens within the text. Conversely, Keyformer *(New Pos)* uses positions based on the new arrangement of tokens within KV cache.

Table 3 presents the accuracy comparison with consistent KV cache size and score function. Notably, when using the original positional information, Keyformer excels in accuracy. However, incorporating new positional information during inference leads to a slight drop in accuracy. Nevertheless, even with new positional information, Keyformer outperforms the state-of-the-art H_2O.

4.4.3 *Comparison with Alternative Distributions*

We conducted an ablation study to assess how different logit adjustment distributions affect model accuracy or key tokens identification. We evaluated three regularization strategies and compared them with Gumbel-based logit adjustment. These are no logit adjustment, constant logit adjustment, and Gaussian distribution-based logit adjustment.

No Logit Adjustment: To examine the effect of omitting regularization on unnormalized logits, we experimented without logit adjustment, where $y_i = x_i + \zcancel{\zeta_i}$, mirroring the approach used in H_2O (Zhang et al., 2023).

Constant Logit Adjustment: To study the impact of constant regularization on all unnormalized logits, we experimented with constant logit adjustment, setting $y_i = x_i + c$, where c is the constant that is being added to every unnormalized logit.

Gaussian Logit Adjustment: We also used a symmetric Gaussian distribution for logit adjustment.

$$f_{Gaussian}(\zeta_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\zeta_i - \mu))^2}{2\sigma^2}\right) \quad (11)$$

$$f_{Gaussian}(y_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - (x_i + \mu))^2}{2\sigma^2}\right) \quad (12)$$

Equation 11 presents the Gaussian probability density function (pdf) with mean μ and variance σ^2 applied to unnormalized logits, while Equation 12 displays the pdf of logits adjusted with Gaussian addition.

We established the baseline accuracy lower bound for Gumbel-based adjustments after analyzing the accuracy of the summarization task on the CNN/Daily Mail dataset. We evaluated this task with a 60% reduction in KV cache. Table 4 provides a comparison of different approaches. We utilized a standard Gumbel pdf with $\mu = 0.5772$ and $\sigma = 1.2825$. For comparison, the Gaussian pdf had an identical mean and variance, and the constant logit adjustment employed a constant value of $c = 0.5772$. The "No logit adjustment" approach uses the method in prior work, H_2O.

Thus, empirical evidence shows that the Gumbel distribution, known for its skewness to initial tokens, is an effective regularization mechanism for key token identification.

Table 4. Empirical evaluation with different logit adjustments. Summarization task with 60% KV cache.

Model	ROUGE-2 Score			
	Gumbel	Gaussian	Constant	None
GPT-J-6B	**19.44**	14.53	12.49	18.87
Cerebras-GPT-6.7B	**15.25**	9.54	8.98	12.73
MPT-7B	**17.57**	10.17	7.56	16.38

4.4.4 *Recent Window versus Key token Window Ratio*

We conducted a sensitivity study to examine the impact of varying the ratio of recent tokens w on the size of the KV cache. This resulted in changes in the number of key tokens $(k - w)$. Results in Appendix A.4 indicate that the models perform better when the recent tokens ratio w falls within the range of 20% to 30%. This observation aligns with our hypothesis that recent and key tokens are critically important for LLM inference.

4.4.5 *Comparison with Attention Sinks*

Recent research introduced StreamingLLM (Xiao et al., 2023), which introduced the concept of "attention sinks." StreamingLLM enables Language Models (LLMs) trained with a finite-length attention window to handle infinite sequence lengths without fine-tuning. This is achieved by retaining the first four tokens (known as "attention sinks") and a moving window of recent tokens, where $w = k-4$. To compare StreamingLLM with Keyformer, we maintained a KV cache size of 60% for both techniques. Table 3 displays the accuracy comparison, showing that StreamingLLM struggles in summarizing text by relying on only the first four tokens as attention sinks and the remaining tokens from a recent window (Appendix A.7).

5 Related Work

Attention Speedup: Prior work focus on improving inference speed for transformer (Vaswani et al., 2017) based models. PoWER-BERT (Goyal et al., 2020) utilizes word-vector elimination by exploiting redundancy for encoder-based models. Linformer (Wang et al., 2020) tries to reduce the attention mechanism from quadratic to linear. Reformer (Kitaev, 2020) reduces attention complexity by locality-sensitive hash (LSH). Linear transformers (Katharopoulos et al., 2020) store accumulated states rather than preserving every representation. FLAT (Kao et al., 2023) suggests optimized dataflow, while other research (Wang et al., 2022) overlaps communication with dependent computation to enhance attention execution. In contrast, Keyformer aims to reduce KV cache, speeds up attention by reducing the tokens.

Sparse Attention: One line of work sparsifies the attention mechanism to reduce the computational and memory capacity of the attention block. BigBird (Zaheer et al., 2020) combines random, windowed, and global attention to maintain the accuracy for transformers while sparsifying the attention block. LongFormer (Beltagy et al., 2020) also utilizes windowed attention with task-based local attention to achieve sparse attention. Spatten (Wang et al., 2021) introduces sparsity at both the head and token levels. However, it needs a dedicated architecture to exploit sparsity. Furthermore, these works do not address inference optimizations.

KV Cache Reduction: El-Attention (Yan et al., 2021) modifies the multi-head attention module to reduce the KV cache size, leveraging key and value stability during incremental decoding for reuse across layers. In contrast, H_2O (Zhang et al., 2023) identifies heavy-hitters and keeps them in the KV cache to reduce its size, neglecting the attention score distribution shift that occurs post elimination of previous tokens from the KV cache, leading to accuracy trade-offs. Other approaches (Liu et al., 2023; Anagnostidis et al., 2023) introduce sparsity at both coarse and fine-grained levels, targeting the elimination of specific heads and tokens during inference. However, these methods require task-specific predictors and fine-tuning of pre-trained models. Another method (Mu et al., 2023) compresses prompts into gist tokens to reduce KV cache. Landmark Attention (Mohtashami & Jaggi, 2023) represents token blocks with an additional landmark token in the vocabulary, necessitating computationally intensive retraining or fine-tuning for gist or landmark token integration.

6 Future Work

Recent techniques like Multi-Query Attention (MQA) (Shazeer, 2019) and Group-Query Attention (GQA) (Ainslie et al., 2023) aim to train foundation models with fewer attention heads. However, such models are typically used after fine-tuning for specific tasks. While the detailed evaluation of Keyformer with these models is deferred to future work, it is worth noting that Keyformer can still be applied on top of MQA or GQA-based models. This is because it discards redundant tokens regardless of the number of heads. Additionally, we plan to integrate Keyformer into the LLM's attention block by replacing the standard softmax with a Keyformer-based softmax. This introduces sparsity during training, addressing the quadratic computational and memory complexities of transformers. This direction aims to enhance scalability to longer contexts without sacrificing accuracy.

7 Conclusion

Advancements in large language models (LLMs) are pushing for longer contexts and extensive text generation, with models trained on sequences of millions of tokens. However, this trend strains system memory bandwidth, leading to execution costs. In longer contexts, the KV cache size, primarily responsible for memory bandwidth consumption and inference latency, exceeds the model parameters' size. To address this, we proposed Keyformer, which effectively reduces the KV cache size upto 50% without sacrificing accuracy by discarding tokens across heads, layers, and beams, identifying essential tokens (key tokens) based on a novel score function. Keyformer can be applied to LLMs at inference time, without requiring fine-tuning, while also improving latency and token generation throughput.

Acknowledgements

We thank the anonymous reviewers for their feedback. Muhammad Adnan's Ph.D. is supported by the Intel TSA and Natural Sciences and Engineering Research Council of Canada (NSERC) [RGPIN-2019-05059] Grants.

REFERENCES

Adnan, M., Maboud, Y. E., Mahajan, D., and Nair, P. J. Accelerating recommendation system training by leveraging popular choices. *Proc. VLDB Endow.*, 15(1):127–140, sep 2021. ISSN 2150-8097.

Adnan, M., Maboud, Y. E., Mahajan, D., and Nair, P. J. Adrec: Advanced feature interactions to address covariate-shifts in recommendation networks. *arXiv preprint arXiv:2308.14902*, 2023.

Adnan, M., Maboud, Y. E., Mahajan, D., and Nair, P. J. Heterogeneous acceleration pipeline for recommendation system training. In *Proceedings of the 51st International Symposium on Computer Architecture (ISCA)*. ACM, 2024.

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

Anagnostidis, S., Pavllo, D., Biggio, L., Noci, L., Lucchi, A., and Hoffmann, T. Dynamic context pruning for efficient and interpretable autoregressive transformers. *arXiv preprint arXiv:2305.15805*, 2023.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

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

Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. Longlora: Efficient fine-tuning of long-context large language models. *arXiv:2309.12307*, 2023.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Cooray, K. Generalized gumbel distribution. *Journal of Applied Statistics*, 37(1):171–179, 2010.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

de Souza Pereira Moreira, G., Rabhi, S., Lee, J. M., Ak, R., and Oldridge, E. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pp. 143–153, 2021.

Dey, N., Gosal, G., Zhiming, Chen, Khachane, H., Marshall, W., Pathria, R., Tom, M., and Hestness, J. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster, 2023.

Dong, S., Cheng, W., Qin, J., and Wang, W. Qaq: Quality adaptive quantization for llm kv cache. 2024.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Gao, L., Tow, J., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., McDonell, K., Muennighoff, N., et al. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 2021.

Ge, S., Zhang, Y., Liu, L., Zhang, M., Han, J., and Gao, J. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2024.

Geng, S., Yuan, J., Tian, Y., Chen, Y., and Zhang, Y. Hiclip: Contrastive language-image pretraining with hierarchy-aware attention. *arXiv preprint arXiv:2303.02995*, 2023.

Goyal, S., Choudhury, A. R., Raje, S., Chakaravarthy, V., Sabharwal, Y., and Verma, A. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pp. 3690–3699. PMLR, 2020.

Huang, L., Cao, S., Parulian, N., Ji, H., and Wang, L. Efficient attentions for long document summarization. *arXiv preprint arXiv:2104.02112*, 2021a.

Huang, L., Cao, S., Parulian, N., Ji, H., and Wang, L. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1419–1436, Online, June 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.112.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

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

Kao, S.-C., Subramanian, S., Agrawal, G., Yazdanbakhsh, A., and Krishna, T. Flat: An optimized dataflow for mitigating attention bottlenecks. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 295–310, 2023.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Kim, H., Hessel, J., Jiang, L., West, P., Lu, X., Yu, Y., Zhou, P., Bras, R. L., Alikhani, M., Kim, G., Sap, M., and Choi, Y. Soda: Million-scale dialogue distillation with social commonsense contextualization. *ArXiv*, abs/2212.10465, 2022.

Kitaev, N. Kaiser, ł., levskaya, a.: Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*, 2023.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

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

Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Liu, Z., Wang, J., Dao, T., Zhou, T., Yuan, B., Song, Z., Shrivastava, A., Zhang, C., Tian, Y., Re, C., et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

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

MosaicML. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. *URL www. mosaicml. com/blog/mpt-7b. Accessed*, pp. 05–05, 2023.

Mu, J., Li, X. L., and Goodman, N. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*, 2023.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In Ammar, W., Louis, A., and Mostafazadeh, N. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Reddi, V. J., Cheng, C., Kanter, D., Mattson, P., Schmuelling, G., Wu, C.-J., Anderson, B., Breughe, M., Charlebois, M., Chou, W., Chukka, R., Coleman, C., Davis, S., Deng, P., Diamos, G., Duke, J., Fick, D., Gardner, J. S., Hubara, I., Idgunji, S., Jablin, T. B., Jiao, J., John, T. S., Kanwar, P., Lee, D., Liao, J., Lokhmotov, A., Massa, F., Meng, P., Micikevicius, P., Osborne, C., Pekhimenko, G., Rajan, A. T. R., Sequeira, D., Sirasao, A., Sun, F., Tang, H., Thomson, M., Wei, F., Wu, E., Xu, L., Yamada, K., Yu, B., Yuan, G., Zhong, A., Zhang, P., and Zhou, Y. Mlperf inference benchmark, 2020.

Roemmele, M., Bejan, C. A., and Gordon, A. S. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*, 2011.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

See, A., Liu, P. J., and Manning, C. D. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099.

Shazeer, N. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Chen, B., Liang, P., Ré, C., Stoica, I., and Zhang, C. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pp. 31094–31116. PMLR, 2023.

Strati, F., Mcallister, S., Phanishayee, A., Tarnawski, J., and Klimovic, A. Déjàvu: Kv-cache streaming for fast, fault-tolerant generative llm serving. *arXiv preprint arXiv:2403.01876*, 2024.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2022.

Sukhbaatar, S., Grave, E., Bojanowski, P., and Joulin, A. Adaptive attention span in transformers. *arXiv preprint arXiv:1905.07799*, 2019.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1441–1450, 2019.

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

Wang, B. and Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Wang, H., Zhang, Z., and Han, S. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 97–110. IEEE, 2021.

Wang, I., Nair, P. J., and Mahajan, D. FLuID: Mitigating stragglers in federated learning using invariant dropout. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wang, S., Wei, J., Sabne, A., Davis, A., Ilbeyi, B., Hechtman, B., Chen, D., Murthy, K. S., Maggioni, M., Zhang, Q., et al. Overlap communication with dependent computation via decomposition in large deep learning models. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 93–106, 2022.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Yan, Y., Chen, J., Qi, W., Bhendawade, N., Gong, Y., Duan, N., and Zhang, R. El-attention: Memory efficient lossless attention for generation. In *International Conference on Machine Learning*, pp. 11648–11658. PMLR, 2021.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al. H _2 o: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv preprint arXiv:2306.14048*, 2023.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Zhao, Y., Wu, D., and Wang, J. Alisa: Accelerating large language model inference via sparsity-aware kv caching. *arXiv preprint arXiv:2403.17312*, 2024.