KVmix: Gradient-Based Layer Importance-Aware Mixed-Precision Quantization for KV Cache

Fei Li, Song Liu*, Weiguo Wu, Shiqiang Nie, Jinyu Wang

School of Computer Science and Technology, Xi'an Jiaotong University
lifei@stu.xjtu.edu.cn, {liusong, wgwu, shiqiang.nie, jinyu.wang}@xjtu.edu.cn

Abstract

The high memory demands of the Key-Value (KV) Cache during the inference of Large Language Models (LLMs) severely restrict their deployment in resource-constrained platforms. Quantization can effectively alleviate the memory pressure caused by KV Cache. However, existing methods either rely on static one-size-fits-all precision allocation or fail to dynamically prioritize critical KV in long-context tasks, forcing memory-accuracy-throughput tradeoffs. In this work, we propose a novel mixed-precision quantization method for KV Cache named KVmix. KVmix leverages gradient-based importance analysis to evaluate how individual Key and Value projection matrices affect the model loss, enabling layer-specific bit-width allocation for mix-precision quantization. It dynamically prioritizes higher precision for important layers while aggressively quantizing less influential ones, achieving a tunable balance between accuracy and efficiency. KVmix introduces a dynamic long-context optimization strategy that adaptively keeps full-precision KV pairs for recent pivotal tokens and compresses older ones, achieving high-quality sequence generation with low memory usage. Additionally, KVmix provides efficient low-bit quantization and CUDA kernels to optimize computational overhead. On LLMs such as Llama and Mistral, KVmix achieves near-lossless inference performance with extremely low quantization configuration (Key 2.19bit Value 2.38bit), while delivering a remarkable 4.9× memory compression and a 5.3× speedup in inference throughput.

Code — https://github.com/LfLab-AI/KVmix

Introduction

Large Language Models (LLMs) (Vaswani et al. 2017), such as GPT (Radford et al. 2019), Llama (Touvron et al. 2023a), and their derivatives, have significantly advanced the field of Natural Language Processing (NLP). These models exhibit outstanding performance (Hadi et al. 2023; Chang et al. 2024) across a diverse array of tasks, including text generation, question answering, and machine translation. The Key-Value (KV) Cache plays an essential role in the autoregressive decoding process of LLMs. The KV Cache substantially reduces redundant computations in the attention mechanism by storing KV states from preceding time steps for subsequent token generation (Xiao et al. 2023). Nevertheless, as sequence lengths grow, the memory footprint of the KV Cache increases linearly, presenting a formidable challenge to hardware resources. For instance, a 70B-parameters model may require over 50GB of memory to maintain the KV Cache for a 20k-token sequence, exceeding typical GPU capacity. In scenarios involving multiple concurrent requests, the KV Cache for each request cannot be shared due to its dependence on unique preceding prompts. Although model parameters can be reused, memory quickly becomes saturated due to the KV Cache demands. Once memory is depleted, data is offloaded to system memory or even disks, resulting in frequent High Bandwidth Memory (HBM) exchanges with system memory. This process causes latency to surge exponentially, leading to catastrophic performance degradation.

The KV Cache's characteristics outlined above severely restrict LLM deployment and inference efficiency in resource-constrained environments, underscoring the pressing need for efficient memory optimization (Liu et al. 2024b). Recent research tackling this issue has predominantly focused on reducing the memory overhead of the KV Cache through quantization and sparsification techniques (Shi et al. 2024; Adnan et al. 2024). Quantization methods, in particular, have gained widespread adoption in industry, significantly contributing to the scalability and accessibility of large-scale models (Kumar 2024). Quantizing the KV Cache can markedly reduce memory usage. Existing quantization methods have demonstrated impressive model performance even at very low bit-widths. However, these methods either rely on static one-size-fits-all precision allocation schemes (Liu et al. 2024c,a), lacking flexibility and performance-aware adaptation capabilities, or incur high computational costs of dynamic quantization while failing to adaptively prioritize critical KVs in long-context tasks (Dong et al. 2024; Duanmu et al. 2024). Therefore, they are forced to make suboptimal trade-offs among memory usage, model accuracy, and computational throughput.

To address these problems, this paper proposes KVmix, a novel mixed-precision quantization method for KV Cache. Compared to existing mixed quantization methods (Dong et al. 2024; Li et al. 2025), KVmix analyzes the importance differences of different model layers at a very low cost, thereby allowing for flexible modification of the quantization configuration based on the model's performance requirements. This flexibility enables KVmix to maximize the

*Corresponding author.

compression rate of the KV Cache and the throughput of the model while maintaining controllable precision. Our specific contributions are as follows:

- We propose a novel layer importance-aware mixed-precision quantization method. This method assesses the importance of KVs at each layer by computing the $L2$ gradient norms of Key and Value projection weights with respect to the model's loss function. Based thereon, it independently applies mixed-precision quantization to different layers, allocating higher bit-widths to critical layers and lower to less influential ones. Therefore, it provides the flexibility to balance between accuracy and resource efficiency across diverse inference scenarios.
- We propose a dynamic pivotal context selection strategy to optimize long-context tasks. According to the KV importance analysis, it adaptively keeps full-precision KV pairs for recent pivotal tokens while aggressively compressing older pairs. This strategy ensures high-quality sequence generation in long-context inference scenarios while dynamically reducing the number of full-precision KV pairs for better memory optimization.
- We design efficient CUDA implementations and a high-compression 3-bit quantization method for KVmix. Extensive experimental results show that KVmix achieves nearly lossless model accuracy across multiple LLMs and datasets, with a 4.9× memory usage reduction and a 5.3× speedup in inference efficiency, outperforming prior state-of-the-art (SOTA) quantization methods for KV Cache.

Related Work and Motivation

Related Work

To mitigate KV Cache memory challenges, researchers have developed many optimization methods, primarily centered on compression techniques and dynamic memory management (Kwon et al. 2023; Lee et al. 2024). We mainly discuss the KV Cache compression techniques related to this work. Existing compression approaches encompass quantization, sparsification (Zhang et al. 2023; Li et al. 2024), and KV Cache sharing (Sun et al. 2024; Wu and Tu 2024). Our method is orthogonal to existing weight quantization (Frantar et al. 2022; Lin et al. 2024) and sparsification methods, and it can also be used as a guideline for the importance of different layers during KV sparsification to achieve more accurate KV eviction.

Extensive research has focused on reducing the memory overhead of KV Cache through quantization. For example, KIVI (Liu et al. 2024c) introduced a 2-bit asymmetric quantization technique, employing per-channel quantization for Keys and per-token quantization for Values. KVQuant (Hooper et al. 2024) proposed a non-uniform quantization strategy, integrating pre-RoPE per-channel Key quantization with per-token Value quantization. It employs offline calibration to manage outliers, achieving robust performance in long-context inference scenarios. QAQ (Dong et al. 2024) developed a dynamic mixed-precision quantization method that calculates quantization bits for Keys and Values online and optimizes the trade-off between accuracy and compression ratio by predicting attention scores. Atom (Zhao et al. 2024) investigated a mixed-precision scheme involving 4-bit and 8-bit activations, dynamically quantizing activations to adapt to input distributions. QJL (Zandieh, Daliri, and Han 2025) introduced a 1-bit quantization technique for the Key, leveraging the Johnson-Lindenstrauss transform followed by sign-bit quantization. KVTuner (Li et al. 2025) frames the mixed quantization of KV caches as a search optimization problem, aiming to find the optimal KV quantization configuration within a vast search space. Numerous other studies have also made significant contributions to KV Cache quantization (Yue et al. 2024; Yang et al. 2024b; Liu et al. 2024a). These efforts generally aim to quantize as many KVs as possible to the lowest feasible bit-width. While methods such as KVQuant, QAQ, and KVTuner have introduced mixed-precision quantization for KV, they often entail significant computational or search overhead. In contrast, this work introduces a lightweight and flexible framework that allows users to dynamically balance quantization bit-width and model accuracy based on specific deployment requirements.

Moreover, several works have identified the attention sink phenomenon, where attention scores excessively favor initial or recent tokens, and newly generated tokens emphasize recent contexts. StreamingLLM (Xiao et al. 2023) leverages this characteristic to achieve efficient infinite-length streaming inference by retaining attention sinks (e.g., initial tokens) alongside recent tokens. PyramidInfer (Yang et al. 2024a) applies layer-wise KV cache compression, selectively keeping key contexts based on attention patterns. Inspired by these works, we propose a dynamic pivotal context selection strategy. Unlike prior approaches, ours determines pivotal context size based on layer-specific KV importance analysis and dynamically updates full-precision KV pairs during decoding, better balancing memory efficiency and generation quality.

Motivation

Current KV Cache quantization methods rely on fixed quantization strategies, neglecting the varying contributions of KV across layers to the final output. To substantiate that quantizing Keys or Values from different layers impacts the model differently, we selectively applied 2-bit quantization to the Keys or Values of distinct layers and evaluated the effects on the model's accuracy, with results presented in Fig. 1. The findings demonstrate that quantizing Keys or Values from different layers has varying impacts on the model's generation quality. However, *efficiently analyzing the contribution disparities across layers of the model to allocate different quantization bits to Keys or Values remains a critical challenge that needs to be addressed.*

In each layer of the KV Cache, the computation process works as follows: at time step t, the i-th layer receives hidden states $H_{i-1,t}$ from the preceding layer's output. These hidden states are used to compute the current token's K and V as: $K_{i,t} = W_{k_i} \cdot H_{i-1,t}$ and $V_{i,t} = W_{v_i} \cdot H_{i-1,t}$, where W_{k_i} and W_{v_i} are the projection weights for K and V at the i-th layer. After computation, the computed $K_{i,t}$ and $V_{i,t}$ are concatenated with the previously stored KV, yielding the complete K and V sequences up to time step t: $K_{i,1:t} = [K_{i,1}, K_{i,2}, \ldots, K_{i,t}]$ and $V_{i,1:t} = [V_{i,1}, V_{i,2}, \ldots, V_{i,t}]$. This computation process indicates that the W_{k_i} and W_{v_i} deter-

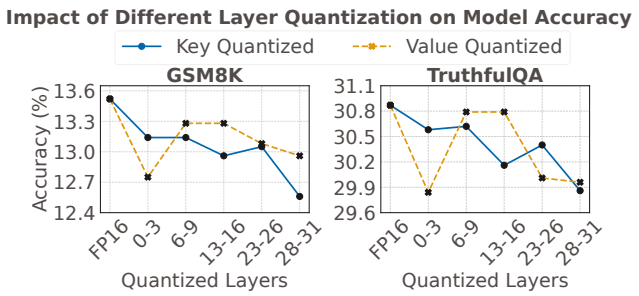

Figure 1: Accuracy of the Llama 2-7B model (Touvron et al. 2023b) on the GSM8K (Cobbe et al. 2021) and TruthfulQA (Lin, Hilton, and Evans 2022) datasets using lm_eval (Gao et al. 2024) (FP16 represents no quantization; 0-3 indicates 2-bit quantization applied individually to the Key or Value of layers 0 through 3, respectively. And so on).

mine how Key and Value are extracted from the hidden states, directly affecting the quality of the KV pairs generated by the attention mechanism and the layer's contribution to the model's output. Fig. 2 provides heatmaps of the W_k and W_v for the Llama 2-7B model. The heatmaps highlight two key insights: ① *Significant variations in KV weight values across different layers.* ② *Distinct distribution patterns of KV weights within the same layer.* For KVs, their values dynamically adapt to changes in the input; however, W_k and W_v are learned during the model's training phase and remain static during inference. As a result, these weights can be leveraged to assess the importance differences of Keys and Values across layers.

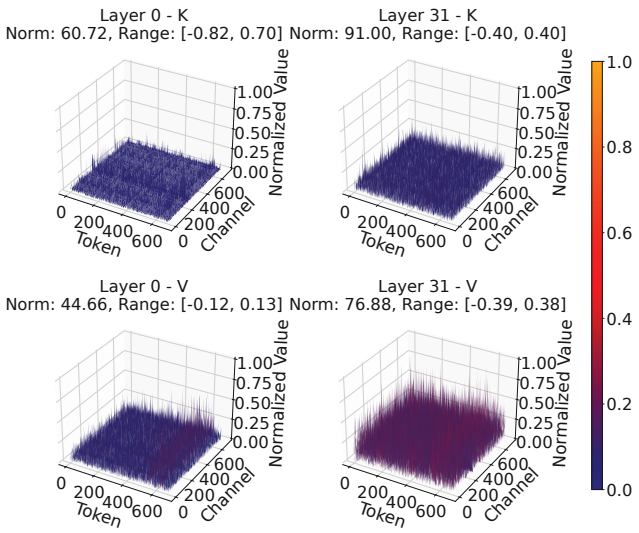

Figure 2: Projection matrix weights of K and V across different layers for the Llama 2-7B model. "Norm" represents the L2 norm of weight matrix for each layer, while "Range" indicates the range of values within each layer's weight matrix.

Methodology

KV Importance Analysis

In the attention mechanism, KV is computed by applying the KV projection weight matrices to the hidden state of the previous layer for linear transformation, then combining the result with the Query vector to calculate the attention output. Therefore, the magnitude of the KV projection weights alone is insufficient to measure the importance of KV across layers, necessitating a more precise evaluation metric. This metric should quantify the sensitivity of K and V at each layer to the model's loss function L. Based on the Motivation and by using the chain rule, we can obtain:

$$K_{i,t} = W_{k_i} \cdot H_{i-1,t} \tag{1}$$

$$\longrightarrow \nabla_{W_{k_i}} L = \frac{\partial L}{\partial K_i}\frac{\partial K_i}{\partial W_{k_i}} = \frac{\partial L}{\partial K_i} H_{i-1}^T \tag{2}$$

$$\longrightarrow ||\frac{\partial L}{\partial K_i}||_2 = \frac{||\nabla_{W_{k_i}} L||_2}{||H_{i-1}||_2} \tag{3}$$

$$V_{i,t} = W_{v_i} \cdot H_{i-1,t} \tag{4}$$

$$\longrightarrow \nabla_{W_{v_i}} L = \frac{\partial L}{\partial V_i}\frac{\partial V_i}{\partial W_{v_i}} = \frac{\partial L}{\partial V_i} H_{i-1}^T \tag{5}$$

$$\longrightarrow ||\frac{\partial L}{\partial V_i}||_2 = \frac{||\nabla_{W_{v_i}} L||_2}{||H_{i-1}||_2} \tag{6}$$

Here, $\nabla_{W_{k_i}} L$ and $\nabla_{W_{v_i}} L$ denote the gradients of L with respect to W_{k_i} and W_{v_i}, respectively. $\frac{\partial L}{\partial K_i}$ and $\frac{\partial L}{\partial V_i}$ represent the partial derivatives of L with respect to the Key and Value, respectively. To quantify the perturbation, assume that the quantization operation introduces small perturbations to the Key and Value matrices, i.e., $K_i^q = K_i + \Delta K$ and $V_i^q = V_i + \Delta V$, where ΔK and ΔV are the quantization errors. The change in L due to quantization is $\Delta L = L(K^q, V^q) - L(K, V)$, which is the difference between the original loss and the quantized loss. To approximate ΔL, perform a first-order Taylor expansion around Key:

$$L(K^q, V^q) \approx L(K, V) + \frac{\partial L}{\partial K} \cdot \Delta K + \frac{\partial L}{\partial V} \cdot \Delta V \tag{7}$$

Thus, the loss change due to quantization is:

$$\Delta L \approx \frac{\partial L}{\partial K} \cdot \Delta K + \frac{\partial L}{\partial V} \cdot \Delta V. \tag{8}$$

$\frac{\partial L}{\partial K}$ and $\frac{\partial L}{\partial V}$ are the gradients $\frac{\partial L}{\partial K_i}$ and $\frac{\partial L}{\partial V_i}$ computed earlier, so:

$$\Delta L \approx \langle \frac{\partial L}{\partial K_i}, \Delta K \rangle + \langle \frac{\partial L}{\partial V_i}, \Delta V \rangle \tag{9}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. For a fixed ΔK, a larger $\left\| \frac{\partial L}{\partial K_i} \right\|_2$ (take L2 norm) amplifies ΔL, indicating greater sensitivity. The weight gradient norm thus reflects Key's impact on L, as $\|\nabla_{W_{k_i}} L\|_2$ proxies $\left\| \frac{\partial L}{\partial K_i} \right\|_2$ (modulo input scaling). The Value is the same.

Based on the above analysis, we propose the KVmix profiler, a gradient-based method that quantifies the contribution of each layer's Key and Value to the model's output,

enabling a judicious mixed-precision quantization strategy. Specifically, we compute the $L2$ norm of the gradients of the model's loss function L with respect to the Key and Value projection weight matrices for each model layer ($\|\nabla_{W_{k_i}} L\|_2$ and $\|\nabla_{W_{v_i}} L\|_2$), and evaluate the importance of the Key and Value components based on these $L2$ norm values. KVmix profiler captures the dynamic sensitivity of these parameters during the model inference process, and provides a layer-specific importance metric to support efficient mixed-precision quantization in subsequent inference stages.

The implementation of KVmix profiler consists of the following three key steps: ① **Data preparation and forward propagation**. A full-precision model is loaded, and multiple prompts are randomly sampled from a target dataset to serve as input data. These prompts are tokenized into input tensors using the tokenizer. Leveraging the autoregressive property of LLMs, each input tensor is shifted left by one position to be used as the corresponding label tensor for computing the model's loss function. Subsequently, the loss value for each input is determined through forward propagation. ② **Gradient calculation and importance assessment**. For the i-layer of the model, the gradients of the losses with respect to the projection weights of the Key (W_{k_i}) and Value (W_{v_i}) are computed independently. This process begins with backpropagation to calculate the gradients, i.e., $\nabla_{W_{k_i}} L$ and $\nabla_{W_{v_i}} L$. The magnitude of these gradients is then evaluated using the $L2$ norm, i.e., $\|\nabla_{W_{k_i}} L\|_2$ and $\|\nabla_{W_{v_i}} L\|_2$. The importance scores of the Keys and Values for each layer can be expressed as:

$$s_{k_i} = \|\nabla_{W_{k_i}} L\|_2, s_{v_i} = \|\nabla_{W_{v_i}} L\|_2 \tag{10}$$

A larger s_{k_i} or s_{v_i} signifies a greater impact of that i-th layer's Key or Value on model's output. To enhance assessment reliability, the gradient norms can be averaged across multiple prompts (p), yielding an average importance score for each layer's Key and Value (P is the number of prompts):

$$\bar{s}_{k_i} = \frac{1}{P} \sum_{p=1}^{P} s_{k_i}^{(p)}, \bar{s}_{v_i} = \frac{1}{P} \sum_{p=1}^{P} s_{v_i}^{(p)} \tag{11}$$

We classify the importance of the Key and Value components across all model layers using the importance scores. The top 20% of the layers of $\bar{s}_{k_i}$ and $\bar{s}_{v_i}$ are quantized with high-bit representations (e.g., 3-bit or 4-bit), while the remaining 80% of layers adopt more aggressive low-bit quantization (e.g., 2-bit). This 20%-80% split is not fixed and can be dynamically adjusted according to the requirements to balance the trade-off between model accuracy and memory usage. Increasing the proportion of low-bit quantization layers can further reduce the memory usage of the KV Cache, but may sacrifice some accuracy. ③**Model Configuration and Inference**. The KV quantization configuration results derived from the above steps are incorporated into the model configuration, enabling the quantized model to be used for inference. The detailed workflow is depicted in Fig. 3. The profiling is performed offline and therefore does not affect inference efficiency. Moreover, the profiling is performed once, allowing the model to reuse the results for subsequent inference tasks.

Asymmetric Low-Bit Quantization

Asymmetric Quantization Strategy We use per-channel and per-token grouping quantization methods for Key and Value, respectively. The KV Cache has the shape $[B, nh, T, D]$, where B is the batch size, nh is the number of attention heads, T is the token sequence length, and D is the head dimension. When the Key is quantized per channel (D), the tensor is reshaped to $[B \times nh \times D, T]$, with each group comprising all tokens of a single channel. This approach is inspired by the distributional properties of the Key Cache that exhibit significant outliers in the channel dimension, i.e., certain channels exhibit significantly large magnitude values. Per-channel quantization isolates errors within each channel and prevents outliers from affecting other channels. When Value is quantized per token, the shape of the tensor is preserved, and each group contains all channels of a single token. Unlike the Key Cache, the Value Cache has no pronounced outliers, but plays a critical role in computing the attention output. Per-token quantization confines errors to individual tokens, preserving the integrity of other important tokens. This asymmetric quantization strategy effectively reduces errors introduced during KV Cache quantization.

Group-Wise Low-Bit Quantization We use group-wise low-bit quantization to minimize KV Cache memory usage. The process includes: ① Calculation of scaling factor s. For each group (per-channel for Key, per-token for Value), compute $s = \frac{max\_val - min\_val}{q_{max}}$ using group min/-max values, where q_{max} denotes the maximum quantized value. ② Element quantization. Quantize elements with $q = round\left(\frac{x - min\_val}{s}\right)$, where x represents the original element value, and q is the quantized value. ③ Clipping. Limit q to $\max(0, \min(q, q_{max}))$. ④ Storage and dequantization. The quantized values are stored using bit operations within a 32-bit integer ($int32$). For 4, 2, and 1 bit, the number of elements per $int32$ is: $feat\_per\_int = 32/bit$. Dequantization is performed by $x = q \cdot s + min\_val$. For **3-bit quantization**, we introduce a new packing strategy to maximize memory efficiency. We organize the quantized elements into blocks of 11, each stored in a 32-bit integer, with the first 10 elements quantized to 3 bits and the 11th element to 2 bits. The clipping range is adjusted based on the element index:

$$q_{max} = \begin{cases} 7, & i = 0, 1, ..., 9 \\ 3, & i = 10 \end{cases} \tag{12}$$

where i is the element index within a block. This strategy increases packing density by 10% over uniform 3-bit quantization that can only hold 10 elements per $int32$.

Dynamic Pivotal Context Selection

In the KV Cache, not all Keys and Values are equally important for generating future tokens. Recent tokens provide the most relevant contextual information for the generation of subsequent tokens and typically have more impact on the tokens to be generated. We define the KVs corresponding to these pivotal recent tokens as the Recent Pivotal Context (RPC). To optimize model performance while minimizing memory usage, we propose a dynamic RPC selection strategy

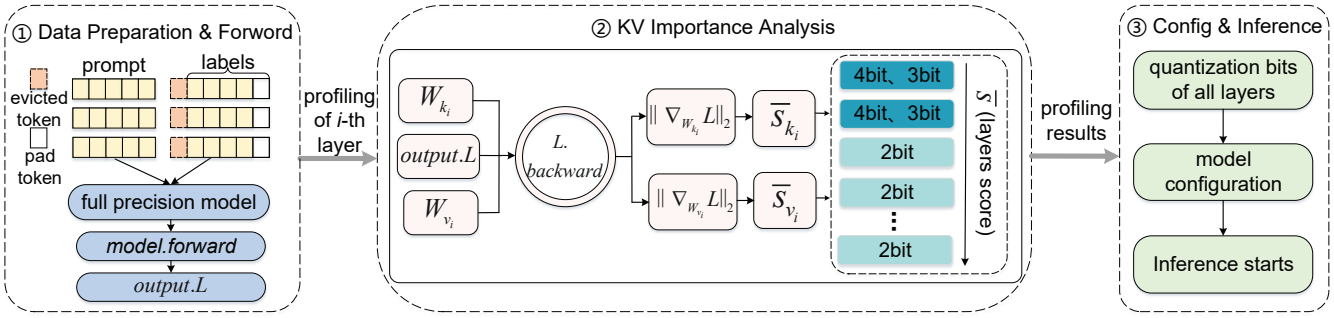

Figure 3: The overview of KVmix profiler.

based on the importance analysis provided by KVmix profiler. Specifically, for the i-layer, we assign it an RPC selection ratio r based on the $\bar{s}_{k_i}$ and $\bar{s}_{v_i}$ scores, with higher $\bar{s}_{k_i}$ and $\bar{s}_{v_i}$ resulting in larger r. The number of RPCs is computed by $num\_RPC = \lfloor r \times current\_RPC \rfloor$. $current\_RPC$ is the sum of the number of new KV states at the current time step and the number of historical RPCs. The corresponding number of KV pairs is selected as RPCs based on num_RPC. We keep full precision for RPCs while performing mixed quantization for less critical and older KV pairs, illustrated in Fig. 4. This strategy ensures that the number of full-precision RPCs is dynamically reduced in runtime during long context inference, thus avoiding excessive memory pressure caused by preserving a large number of full-precision KV pairs, while maintaining high-quality sequence generation. Additionally, since the importance of Key and Value may differ within the same layer, the RPC selection ratio for Key and Value varies accordingly within that layer. The RPC selection ratio can be adjusted to balance accuracy and memory: increasing it improves accuracy but requires more memory.

CUDA Implementation

During model inference, the quantization of the KV Cache introduces additional overhead due to quantization and dequantization operations. To improve inference efficiency, we design efficient CUDA kernels for quantization, dequantization, and matrix-vector multiplication. ① **Fusion of quantization and concatenation.** In the decoding phase, the KV states of current layer are concatenated with the historical KV Cache. Quantizing the current states before concatenation causes extra memory access overhead. We fuse quantization and concatenation into a single CUDA kernel, processing each element in a streaming manner. KV elements are quantized and appended directly to the historical KV Cache, thereby reducing memory access. CUDA thread blocks process tokens in parallel, and shared memory caches intermediate results to enhance data locality. ② **Fusion of dequantization and matrix-vector multiplication.** In attention computation, the quantized KV requires dequantization before matrix-vector multiplication. Dequantizing the full KV beforehand increases memory usage. We fuse dequantization with multiplication, dequantizing each element on-the-fly and immediately multiplying and accumulating it with its corresponding element, minimizing memory overhead. ③ **Efficient kernels for multi-bit quantization configurations.** To support KVmix's various quantization bit-widths, we develop CUDA kernels for 1-, 2-, 3-, and 4-bit quantization, along with tailored matrix-vector multiplication kernels for each configuration, ensuring compatibility across bit-widths.

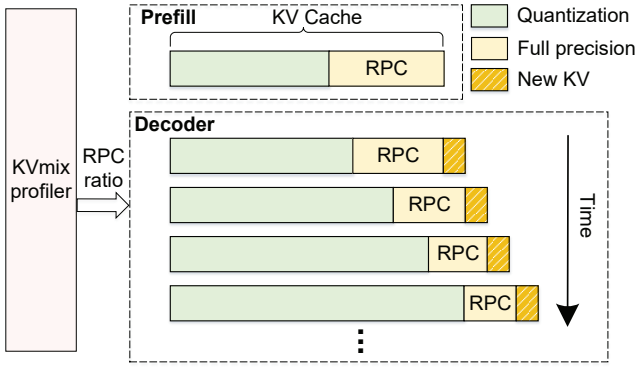

Figure 4: Dynamic adjustment of quantized KV Cache based on RPC during prefill and decoding phases.

Experimental Results

Experimental Setup

We evaluated the proposed method using Llama 2-7B-hf, Llama 3-8B-Instruct, Llama 3.1-8B (Grattafiori et al. 2024), Mistral-7B-Instruct-v0.3 (Jiang et al. 2023), and Falcon-7B (Almazrouei et al. 2023) models. The datasets were selected based on three distinct evaluation schemes: ① Long Context Evaluation: We used the LongBench (Bai et al. 2024) benchmark to assess performance on long-context tasks. It encompasses multiple key long-text application scenarios. Due to the limited GPU memory, the maximum sequence length was set to 4096. ② Language Modeling: We measured the perplexity on the Wikitext-2 (Merity et al. 2016) dataset to evaluate its language modeling capabilities. ③ Mathematical Reasoning: We employed the GSM8K (Cobbe et al. 2021) dataset to assess the model's performance on mathematical reasoning tasks. We used the NVIDIA RTX 4090 GPU (24GB) to evaluate the model's inference efficiency and the KV cache's compression rate.

Profiling Results

We used 3-bit and 2-bit mixed quantization for Key, and 4-bit and 2-bit mixed quantization for Value. When the KV is quantized to 3 bits or 4 bits, the RPC proportion is set to 20%; for 2-bit quantization, the RPC proportion is set to 10%. When the RPC proportion exceeds 20%, its contribution to accuracy improvement is marginal; thus, we selected 20% as the high-bit configuration for KVmix. The group size for quantization is 32. We selected 30 prompts from the LongBench for KV importance analysis. By the KVmix profiler, we can obtain the KV bit configurations and RPC proportions for each layer. When utilizing the KVmix profiler, randomly selecting 20 to 30 prompts is sufficient to yield reliable importance analysis results; additional prompts do not significantly alter the outcomes. For the models and experimental environment used in this work, this process requires only 10 to 15 minutes, highlighting the efficiency of the KVmix profiler. Users can flexibly customize the quantization configuration by adjusting the proportion of layers with different bit-widths in the KVmix profiler to meet varying accuracy or memory requirements. Fig. 5 illustrates the trends in accuracy, KV memory usage, and throughput as we varied the proportion of model layers quantized to 3 and 4 bits. When the proportion of model layers quantized to 3 and 4 bits is set to 20%, the optimal tradeoff among these three factors is achieved. Under this configuration, the average quantization bit-width for Key is 2.19 (exact value: 2.1875), and for Value is 2.38 (exact value: 2.375). The detailed configuration obtained using the KVmix profiler is shown in Fig. 6. Unless otherwise specified, the configuration is applied to the k-2.19v2.38 quantization in subsequent experiments.

Performance Evaluation

Long Context We evaluated the performance of various quantization configurations across 8 distinct datasets from the LongBench benchmark, using the FP16 model as the baseline. The detailed experimental results are presented in the Table

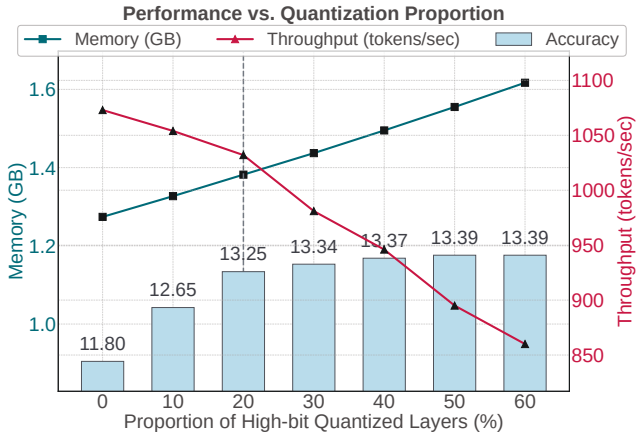

Figure 5: Performance variation of Llama 2-7B with different quantization configurations ("10%" indicates the top 10% important layers are quantized to 4 and 3 bits, and the remaining layers are quantized to 2 bits. The dataset is GSM8K).

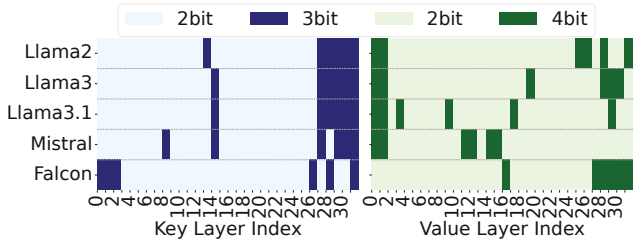

Figure 6: Detailed quantization configurations of KVmix-k2.19v2.38 for different models.

1. The findings indicate that KVmix-k2.19v2.38 achieves an average accuracy loss (average values on 4 different models) of 1.67% compared to the FP16 baseline. In contrast, the average accuracy loss of KVmix-2bit reached 4.53% relative to the baseline. In addition, random selection of high-bit quantization layers (random-k2.19v2.38) leads to an average accuracy loss of 4.06%. These accuracy losses are significantly higher than KVmix-k2.19v2.38, demonstrating the advantage of KV importance-aware mixed quantization. The average accuracy of KVmix-k2.19v2.38w/oRPC decreases by 3.28% compared with KVmix-k2.19v2.38, proving the effectiveness of using RPC for improving model accuracy.

We compared KVmix against prior SOTA methods for KV Cache, specifically Key per-channel and Value per-token methods, i.e., KIVI (Liu et al. 2024c) and KVQuant (Hooper et al. 2024), since they are known to minimize KV quantization errors. Additionally, we evaluated KVmix against the latest SOTA method, QJL (Zandieh, Daliri, and Han 2025). Table 2 presents the accuracy results. The results show that KVmix-k2.19v2.38 surpasses the performance of KIVI-2bit-r64 and QJL-3bit, reducing the average accuracy loss by 1.50% and 0.68%, respectively. While KVQuant-3bit-1% achieves accuracy comparable to KVmix-k2.19v2.38, its memory compression ratio and inference efficiency fall short of those delivered by KVmix-k2.19v2.38 (Fig. 7, Fig. 8). By increasing the quantization bit-width, KVmix-k2.28v2.56 demonstrates a more significant advantage over KVQuant-3bit-1% in accuracy, while maintaining a comparable memory compression ratio (4.8×) and superior inference acceleration (5.23×). This flexibility in balancing accuracy and quantization bit-width represents a critical strength of KVmix.

GSM8K and Wikitext-2 We evaluated the capabilities of the quantized model in language modeling and mathematical reasoning using the FP16 model as the baseline. Atom (Zhao et al. 2024) exhibits very poor performance in long contexts, and thus, we only compare it in this section. The evaluation was conducted using the lm_eval (Gao et al. 2024) framework, where the quantized model replaced the Hugging Face model. Specifically, we measured the accuracy on the GSM8K dataset and the perplexity on the Wikitext-2 dataset. The experimental results are detailed in Table 3. The results show that 2bit (k-T, v-T) suffers a catastrophic performance loss on GSM8K and Wikitext-2, and the model almost loses its reasoning ability. For 4bit (k-T, v-T), the performance loss of the model on GSM8K and Wikitext-2 also reached 9.17%

Models	Methods	Datasets								Average
		TriviaQA	Qasper	MF-en	QMSum	2WikiMQA	Rbench-P	TREC	PsgRetr-en	
Llama-2-7B	FP16	78.89	9.55	22.86	21.19	9.94	55.64	66.00	6.64	33.839
	KVmix-2bit	77.57	9.58	22.47	20.45	9.15	56.34	66.00	5.29	33.356
	random-k2.19v2.38	78.30	9.39	22.54	20.41	9.46	56.36	66.00	5.49	33.494
	KVmix-k2.19v2.38w/oRPC	77.95	9.19	21.03	19.98	9.05	56.13	65.50	5.61	33.055
	KVmix-k2.19v2.38	**78.78**	**9.59**	**22.82**	**20.49**	**9.77**	**56.54**	**66.00**	**5.72**	**33.714**
Llama-3-8B	FP16	78.35	40.75	46.80	21.69	32.39	49.77	70.50	37.00	47.156
	KVmix-2bit	76.13	39.18	45.70	21.20	32.19	44.56	71.00	36.30	45.783
	random-k2.19v2.38	78.01	39.17	45.90	21.22	32.02	45.36	71.00	36.50	46.148
	KVmix-k2.19v2.38w/oRPC	77.12	39.04	45.18	21.03	32.05	45.20	71.00	36.00	45.828
	KVmix-k2.19v2.38	**78.13**	**39.15**	**46.31**	**21.26**	**32.20**	**47.56**	**71.00**	**36.50**	**46.514**
Llama-3.1-8B	FP16	83.67	11.53	31.13	22.88	13.92	61.84	67.50	19.50	38.996
	KVmix-2bit	83.10	10.90	30.76	22.11	13.08	58.92	67.00	19.00	38.109
	random-k2.19v2.38	83.25	10.90	31.05	22.34	13.05	59.26	67.00	19.00	38.231
	KVmix-k2.19v2.38w/oRPC	82.18	11.05	30.86	22.20	13.27	58.51	67.00	19.00	38.009
	KVmix-k2.19v2.38	**83.28**	**11.40**	**31.49**	**22.90**	**12.92**	**59.96**	**67.50**	**19.50**	**38.619**
Mistral-7Bv0.3	FP16	84.29	36.19	54.70	21.79	35.08	53.06	73.50	32.50	48.889
	KVmix-2bit	84.08	34.29	53.87	21.37	33.39	50.99	73.50	32.00	47.936
	random-k2.19v2.38	84.01	34.35	53.61	21.45	33.40	50.59	73.50	32.50	47.926
	KVmix-k2.19v2.38w/oRPC	83.07	34.18	52.65	21.10	32.32	51.30	73.50	32.50	47.578
	KVmix-k2.19v2.38	**84.03**	**35.67**	**53.68**	**21.84**	**33.81**	**51.98**	**73.50**	**32.75**	**48.408**
Falcon-7B	FP16	6.94	3.87	7.47	3.96	4.87	12.92	14.00	3.95	7.248
	KVmix-2bit	5.96	3.15	6.19	3.28	4.22	11.40	13.50	3.21	6.364
	random-k2.19v2.38	6.11	3.10	6.36	3.26	4.24	11.45	13.50	3.26	6.410
	KVmix-k2.19v2.38w/oRPC	6.05	3.01	6.54	3.22	4.13	11.41	13.00	3.22	6.323
	KVmix-k2.19v2.38	**6.64**	**3.28**	**7.06**	**3.52**	**4.60**	**12.71**	**14.00**	**3.62**	**6.929**

Table 1: Model accuracy of 4 LLMs on LongBench with different quantization configurations. KVmix-k2.19v2.38 uses the configurations of Fig. 6. KVmix-2bit uses the asymmetric 2-bit (Key per-channel and Value per-token) quantization for all model layers (RPC ratio is set to 10%). random-k2.19v2.38 randomly selects 20% of the model layers to perform asymmetric 3-bit and 4-bit quantization for Key and Value (RPC ratio is set to 20%), and the remaining layers are 2-bit quantization (RPC ratio is set to 10%). KVmix-k2.19v2.38w/oRPC is KVmix-k2.19v2.38 without RPC (RPC ratio is set to 0%).

and 5.28%, respectively. In contrast, on the Wikitext-2, the perplexity score of KVmix-k2.19v2.38 is almost comparable to the baseline, while on the more challenging GSM8K mathematical reasoning task, KVmix-k2.19v2.38 has an accuracy loss of 2.00%, which significantly outperforms the 2bit (k-T, v-T) and the 4bit (k-T, v-T). Moreover, on GSM8K, KVmix-k2.19v2.38 shows a significant accuracy improvement compared to KVmix-2bit and random-k2.19v2.38, which do not leverage the KV importance analysis for more accurate quantization. Compared to Atom-4bit and other SOTA methods, KVmix-k2.19v2.38 also has an accuracy advantage. Notably, the Atom-4bit performs 4-bit quantization on both the model weights and activations, which results in greater accuracy loss. These results demonstrate the superior performance of KVmix-k2.19v2.38 in complex task reasoning.

Inference Efficiency and Memory Usage Evaluation

We evaluated the inference throughput and memory usage of KVmix during inference. To ensure fairness, we applied identical input data across all evaluated methods. The number of input tokens is 688, the maximum number of new tokens is set to 1024, and the model is Llama 2-7B-hf. We compared KVmix against the KIVI-2bit-r64, KVQuant-3bit-1%, QJL-3bit, and Atom-4bit. Memory usage results are illustrated in Fig. 7, with a batch size fixed at 4. The reported memory usage represents the peak memory usage during inference minus the memory occupied by the model before inference. To fully utilize the GPU memory, we incrementally increased the batch size to explore KVmix's maximum throughput. The throughput results are shown in Fig. 8. The baseline (FP16), Aotm-4bit, and KIVI-2bit-r64 reach out of memory at batch sizes of 4, 18, and 28, respectively, while the KVmix-k2.19v2.38 can reach a maximum batch size of 30 with an inference throughput of 1032 tokens per second.

The results reveal that KVmix-k2.19v2.38 achieves a 4.9× reduction in memory usage and up to a 5.3× increase in throughput compared to the baseline. This efficient memory compression stems from KVmix's extremely low bit quantization and dynamic RPC strategy, which progressively reduces the full-precision KV as inference progresses. In contrast, KIVI employs a fixed full-precision residual strategy, unable to dynamically reduce the number of full-precision KVs.

Methods	TriviaQA	Qasper	MF-en	QMSum	2WikiMQA	Repobench-P	TREC	PsgRetr-en	Average
FP16	78.89	9.55	22.86	21.19	9.94	55.64	66.00	6.64	33.839
KIVI-2bit-r64	77.08	9.16	22.55	20.12	9.05	56.15	66.00	5.62	33.216
QJL-3bit	78.25	9.10	22.60	20.45	9.68	56.09	66.00	5.72	33.486
KVQuant-3bit-1%	78.79	10.51	22.61	20.58	9.75	55.62	66.00	5.76	33.703
KVmix-k2.19v2.38	78.78	9.59	22.82	20.49	9.77	56.54	66.00	5.72	**33.714**
KVmix-k2.28v2.56	78.05	10.21	23.21	20.63	9.72	56.61	66.00	6.08	**33.814**

Table 2: Accuracy comparison of different quantization methods on LongBench using the Llama 2-7B-hf model. KIVI-2bit-r64 uses 2-bit quantization with a full-precision residual of 64. KVQuant-3bit-1% uses 3-bit quantization and 1% outlier handling. QJL-3bit uses 3-bit quantization. KVmix-k2.28v2.56 increases the proportion of high-bit quantization layers in KVmix-k2.19v2.38 to 30%.

Methods	GSM8K (acc↑)	Wikitext-2 (ppl↓)
FP16	13.52	8.71
2bit (k-T, v-T)	0.83	11089
4bit (k-T, v-T)	12.28	9.17
KVmix-2bit	11.80	8.73
random-k2.19v2.38	11.97	8.73
Atom-4bit	12.30	9.32
KIVI-2bit-r64	12.75	8.80
QJL-3bit	13.11	8.75
KVQuant-3bit-1%	13.23	8.71
KVmix-k2.19v2.38	**13.25**	**8.71**

Table 3: Model accuracy (acc) on GSM8K and perplexity (ppl) on Wikitext-2 using Llama 2-7B-hf. 2bit (k-T, v-T) uses the symmetric 2-bit (Key per-token and Value per-token) quantization for all model layers, and 4bit (k-T, v-T) uses the symmetric 4-bit quantization; their RPC ratio is set to 0.

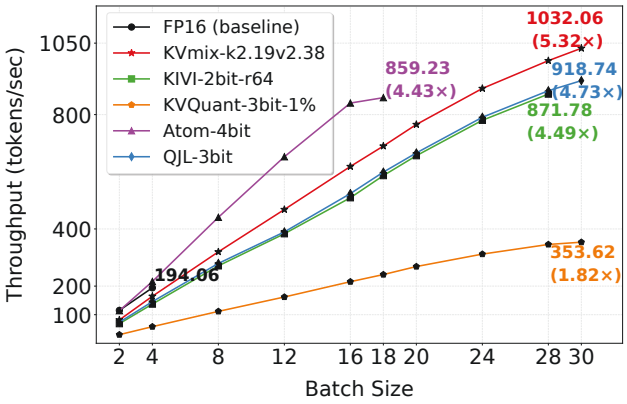

Figure 8: Inference throughput of different quantization methods with different batch sizes on the Llama 2-7B-hf model.

Thus, KVmix saves more memory than KIVI-2bit despite using Key-2.19 and Value-2.38 bit quantization. Meanwhile, Atom quantizes both model weights and activations while utilizing tensor cores for optimized kernel, achieving a higher throughput at the same batch size but incurring greater model accuracy degradation (Table 3). While KVQuant achieves significant memory compression, its inference efficiency is hampered by substantial preprocessing requirements. QJL implements "zero-overhead" quantization by eliminating the need to store extra constants like zero-points and scaling factors. This allows it to achieve a slightly better memory compression compared to KVmix, but its inference efficiency and accuracy are lower than those of KVmix.

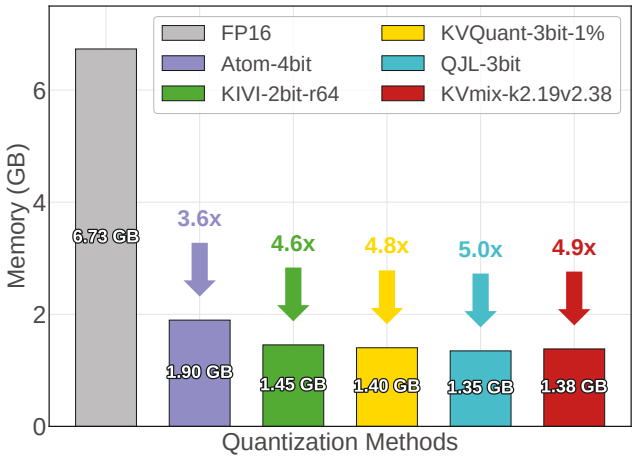

Figure 7: Dynamic peak memory usage of different methods during inference on the Llama 2-7B-hf model.

Conclusion

This paper proposes KVmix, a novel mixed quantization method tackling the KV Cache memory bottleneck in LLM inference. KVmix creatively integrates layer importance analysis based on KV weight gradients into KV quantization and integrates dynamic long-context optimization to cut memory usage while maintaining generation quality. It achieves significant memory and efficiency gains with minimal loss in accuracy, offering flexibility to adapt quantization strategies to diverse scenarios. Future work will explore integrating lightweight mechanisms for real-time KV bit adjustments into KVmix to enhance adaptability.

Acknowledgments

This research was funded by National Key R&D Program of China (2022YFB4501604).

References

Adnan, M.; Arunkumar, A.; Jain, G.; Nair, P. J.; Soloveychik, I.; and Kamath, P. 2024. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6: 114–127.

Almazrouei, E.; Alobeidli, H.; Alshamsi, A.; Cappelli, A.; Cojocaru, R.; Debbah, M.; Goffinet, É.; Hesslow, D.; Launay, J.; Malartic, Q.; et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Bai, Y.; Tu, S.; Zhang, J.; Peng, H.; Wang, X.; Lv, X.; Cao, S.; Xu, J.; Hou, L.; Dong, Y.; et al. 2024. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*.

Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3): 1–45.

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

Dong, S.; Cheng, W.; Qin, J.; and Wang, W. 2024. Qaq: Quality adaptive quantization for llm kv cache. *arXiv preprint arXiv:2403.04643*.

Duanmu, H.; Yuan, Z.; Li, X.; Duan, J.; Zhang, X.; and Lin, D. 2024. Skvq: Sliding-window key and value cache quantization for large language models. *arXiv preprint arXiv:2405.06219*.

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

Gao, L.; Tow, J.; Abbasi, B.; Biderman, S.; Black, S.; DiPofi, A.; Foster, C.; Golding, L.; Hsu, J.; Le Noac'h, A.; Li, H.; McDonell, K.; Muennighoff, N.; Ociepa, C.; Phang, J.; Reynolds, L.; Schoelkopf, H.; Skowron, A.; Sutawika, L.; Tang, E.; Thite, A.; Wang, B.; Wang, K.; and Zou, A. 2024. A framework for few-shot language model evaluation.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Hadi, M. U.; Qureshi, R.; Shah, A.; Irfan, M.; Zafar, A.; Shaikh, M. B.; Akhtar, N.; Wu, J.; Mirjalili, S.; et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 3.

Hooper, C.; Kim, S.; Mohammadzadeh, H.; Mahoney, M. W.; Shao, S.; Keutzer, K.; and Gholami, A. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37: 1270–1303.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

Kumar, A. 2024. Residual vector quantization for KV cache compression in large language model. *arXiv preprint arXiv:2410.15704*.

Kwon, W.; Li, Z.; Zhuang, S.; Sheng, Y.; Zheng, L.; Yu, C. H.; Gonzalez, J.; Zhang, H.; and Stoica, I. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 611–626.

Lee, W.; Lee, J.; Seo, J.; and Sim, J. 2024. {InfiniGen}: Efficient generative inference of large language models with dynamic {KV} cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 155–172.

Li, X.; Xing, Z.; Li, Y.; Qu, L.; Zhen, H.-L.; Liu, W.; Yao, Y.; Pan, S. J.; and Yuan, M. 2025. KVTuner: Sensitivity-Aware Layer-Wise Mixed-Precision KV Cache Quantization for Efficient and Nearly Lossless LLM Inference. *arXiv preprint arXiv:2502.04420*.

Li, Y.; Huang, Y.; Yang, B.; Venkitesh, B.; Locatelli, A.; Ye, H.; Cai, T.; Lewis, P.; and Chen, D. 2024. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37: 22947–22970.

Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.-M.; Wang, W.-C.; Xiao, G.; Dang, X.; Gan, C.; and Han, S. 2024. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. In *MLSys*.

Lin, S.; Hilton, J.; and Evans, O. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3214–3252. Dublin, Ireland: Association for Computational Linguistics.

Liu, R.; Bai, H.; Lin, H.; Li, Y.; Gao, H.; Xu, Z.; Hou, L.; Yao, J.; and Yuan, C. 2024a. Intactkv: Improving large language model quantization by keeping pivot tokens intact. *arXiv preprint arXiv:2403.01241*.

Liu, Y.; Li, H.; Cheng, Y.; Ray, S.; Huang, Y.; Zhang, Q.; Du, K.; Yao, J.; Lu, S.; Ananthanarayanan, G.; et al. 2024b. Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, 38–56.

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

Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Shi, L.; Zhang, H.; Yao, Y.; Li, Z.; and Zhao, H. 2024. Keep the cost down: A review on methods to optimize llm's kv-cache consumption. *arXiv preprint arXiv:2407.18003*.

Sun, Y.; Dong, L.; Zhu, Y.; Huang, S.; Wang, W.; Ma, S.; Zhang, Q.; Wang, J.; and Wei, F. 2024. You only cache once:

Decoder-decoder architectures for language models. *Advances in Neural Information Processing Systems*, 37: 7339–7361.

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

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

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

Wu, H.; and Tu, K. 2024. Layer-condensed kv cache for efficient inference of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 11175–11188.

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

Yang, D.; Han, X.; Gao, Y.; Hu, Y.; Zhang, S.; and Zhao, H. 2024a. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. *arXiv preprint arXiv:2405.12532*.

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

Yue, Y.; Yuan, Z.; Duanmu, H.; Zhou, S.; Wu, J.; and Nie, L. 2024. Wkvquant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*.

Zandieh, A.; Daliri, M.; and Han, I. 2025. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 25805–25813.

Zhang, Z.; Sheng, Y.; Zhou, T.; Chen, T.; Zheng, L.; Cai, R.; Song, Z.; Tian, Y.; Ré, C.; Barrett, C.; et al. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36: 34661–34710.

Zhao, Y.; Lin, C.-Y.; Zhu, K.; Ye, Z.; Chen, L.; Zheng, S.; Ceze, L.; Krishnamurthy, A.; Chen, T.; and Kasikci, B. 2024. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6: 196–209.