# A Survey on Large Language Model Acceleration based on KV Cache Management

**Anonymous authors**
**Paper under double-blind review**

## Abstract

Large Language Models (LLMs) have revolutionized a wide range of domains such as natural language processing, computer vision, and multi-modal tasks due to their ability to comprehend context and perform logical reasoning. However, the computational and memory demands of LLMs, particularly during inference, pose significant challenges when scaling them to real-world, long-context, and real-time applications. Key-Value (KV) cache management has emerged as a critical optimization technique for accelerating LLM inference by reducing redundant computations and improving memory utilization. This survey provides a comprehensive overview of KV cache management strategies for LLM acceleration, categorizing them into token-level, model-level, and system-level optimizations. Token-level strategies include KV cache selection, budget allocation, merging, quantization, and low-rank decomposition, while model-level optimizations focus on architectural innovations and attention mechanisms to enhance KV reuse. System-level approaches address memory management, scheduling, and hardware-aware designs to improve efficiency across diverse computing environments. Additionally, the survey provides an overview of both text and multimodal datasets and benchmarks used to evaluate these strategies. By presenting detailed taxonomies and comparative analyses, this work aims to offer useful insights for researchers and practitioners to support the development of efficient and scalable KV cache management techniques, contributing to the practical deployment of LLMs in real-world applications.

## 1 Introduction

Large Language Models (LLMs) Hadi et al. (2023); Zhu et al. (2023), trained on massive corpora, have revolutionized various domains such as natural language processing Naveed et al. (2023); Min et al. (2024); Xu et al. (2024a), computer vision Liu et al. (2023a); Zhang et al. (2024c); Berrios et al. (2023), and multi-modal Zhang et al. (2024a); Cui et al. (2024); Wu et al. (2023b) tasks. Their ability to understand context and perform logical reasoning has enabled remarkable success in various fields, such as time series analysis Jin et al. (2023); Ma et al. (2024a), recommendation Tan & Jiang (2023); Wu et al. (2024c), autonomous driving Yang et al. (2023); Chen et al. (2024b); Fu et al. (2024b), and healthcare Qiu et al. (2023); Zhou et al. (2023b). These breakthroughs are powered by state-of-the-art architectures and training paradigms, enabling models to achieve unparalleled performance across diverse tasks. Prominent LLMs, such as GPT Brown et al. (2020); Radford et al. (2018; 2019), LLaMA Touvron et al. (2023); Dubey et al. (2024), DeepSeek Dai et al. (2024); DeepSeek-AI et al. (2024); Lu et al. (2024), Mistral Jiang et al. (2024a; 2023), and GLM Zeng et al. (2023); Du et al. (2022), are built on the foundational transformer architecture Vaswani et al. (2017), which excels at capturing long-range dependencies in sequential data. However, despite their powerful capabilities, the computational and memory demands of LLMs, particularly during inference, present significant challenges when scaling them to real-world, long-context, and real-time applications.

A critical bottleneck in LLM inference lies in the efficient management of Key-Value (KV) pairs. Recently, caching techniques Gracioli et al. (2015); Podlipnig & Böszörmenyi (2003) have been extensively employed to store previously computed intermediate results, allowing their reuse in subsequent inference steps to accelerate the model, such as graph neural networks Li & Chen (2021); Li et al. (2023c); Lin et al. (2020). Fortunately, the auto-regressive generation mechanism inherent to LLMs presents an opportunity to leverage

KV caching for efficient text generation. Specifically, auto-regressive generation enables LLMs to produce text token by token, with each token conditioned on all previously generated ones. While this approach is highly effective for generating coherent and contextually relevant outputs, it suffers from poor scalability with long input sequences, as the computational and memory requirements grow quadratically with sequence length. The KV cache addresses this issue by storing key and value matrices from previous decoding steps, enabling their reuse and significantly reducing redundant computations.

Several recent surveys Zhu et al. (2023); Zhuang et al. (2023); Park et al. (2024); Wang et al. (2024b); Ding et al. (2023); Miao et al. (2023); Wan et al. (2023); Zhou et al. (2024c); Tang et al. (2024c); Kachris (2024); Xu et al. (2023); Albalak et al. (2024); Zefan-Cai (2024) have explored the domain of efficient LLMs. These surveys primarily examine various aspects of LLM efficiency, presenting valuable insights while leaving room for further refinement and innovation. In particular, many of these works primarily focus on holistic approaches to improving LLM efficiency, examining a wide range of techniques across multiple dimensions, such as span data-level optimizations (e.g., prompt engineering), model architecture-level optimizations (e.g., efficient transformer designs), and system-level optimizations (e.g., task scheduling). For instance, Ding et al.Ding et al. (2023) explore efficiency techniques that integrate data-level and model architecture perspectives, while Miao et al.Miao et al. (2023) examine efficient LLM inference from a comprehensive system-level perspective. Similarly, Tang et al.Tang et al. (2024c), Wan et al.Wan et al. (2023), and Xu et al. Xu et al. (2023) provide analyses that encompass data, model, and system-level optimizations, reflecting holistic approaches to LLM acceleration.

On the other hand, some surveys focus on more specialized aspects for LLM acceleration. For example, Zhu et al.Zhu et al. (2023), Park et al.Park et al. (2024), Wang et al.Wang et al. (2024b), and Tang et al.Tang et al. (2024c) focus on model compression as a key aspect of model-level optimization. Similarly, Kachris et al.Kachris (2024) examine hardware acceleration strategies tailored for LLMs, while Xu et al.Xu et al. (2023) investigate parameter-efficient tuning approaches. Albalak et al.Albalak et al. (2024) discuss data selection strategies to enhance the efficiency of LLM training, and Xia et al.Xia et al. (2024) highlight collaborative techniques, such as speculative decoding Leviathan et al. (2023); Kim et al. (2024b), to accelerate model inference. Li et al. Li et al. (2024c) focus on prompt compression. Similar to our work, Shi et al.Shi et al. (2024), Li et al.Li et al. (2024a), and Yuan et al. Yuan et al. (2024) also explore the use of KV caches to accelerate LLMs. However, our survey is both complementary and more comprehensive, offering a detailed taxonomy of KV cache management for text-based and multi-modal LLMs. We categorize techniques across token-level, model-level, and system-level perspectives and include benchmarks for both text and multi-modal scenarios. In particular, complementing existing KV cache surveys, we provide a detailed comparison of the differences and advantages of existing models at the token-level, model-level, and system-level.

Specifically, this survey provides a comprehensive overview of the current state of KV cache management and its role in accelerating LLM inference. We begin by introducing the transformer architecture and the role of the KV cache in enabling efficient auto-regressive text generation. We then analyze the challenges associated with KV cache management, including its impact on computational complexity, memory usage, and real-time performance. Following this, we present a taxonomy of existing optimization techniques, categorizing them into token-level, model-level, and system-level optimization approaches. Additionally, we discuss datasets and evaluation metrics used to benchmark these techniques and provide insights into their effectiveness across various tasks and applications.

## 2 Preliminary

Large language models (LLMs), pretrained on vast corpora, have demonstrated superior capabilities in context understanding and logical reasoning. These models have achieved remarkable success across a wide range of tasks in various domains, including natural language processing Naveed et al. (2023); Min et al. (2024); Xu et al. (2024a) and computer vision Liu et al. (2023a); Zhang et al. (2024c); Berrios et al. (2023). Mainstream LLMs, such as GPT Bubeck et al. (2023), LLaMA Touvron et al. (2023), and DeepSeek Dai et al. (2024), are primarily built on the transformer architecture Vaswani et al. (2017). To explore the role of Key-Value (KV) cache management in accelerating LLM computations, we first outline the core components

Table 1: Notation Summary

| **Symbol** | **Definition** |
|---|---|
| $X$ | Input sequence of tokens |
| $\mathbf{X}$ | Dense representations of $X$ |
| $d_x$ | Dimensionality of the input embeddings. |
| $\mathbf{E}$ | Embedding matrix $\mathbf{E} \in \mathbb{R}^{d_{\text{vocab}} \times d_x}$. |
| $PE(X)$ | Positional encoding |
| $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i$ | Query, Key, and Value matrices |
| $d_k, d_v$ | Query/Key and Value dimension |
| $\mathbf{W}_{Q_i}, \mathbf{W}_{K_i}, \mathbf{W}_{V_i}$ | Weight matrices for computing $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i$. |
| $\mathbf{Z}_i$ | Self-attention Output |
| $\mathbf{W}_O$ | Weight matrix |
| $\mathbf{W}_1, \mathbf{W}_2$ | Weight matrices |
| $\mathbf{b}_1, \mathbf{b}_2$ | Bias vectors |
| $t$ | Sequence length index |
| $t_c$ | Number of tokens stored in the KV cache. |
| $\mathbf{K}_i^t, \mathbf{V}_i^t$ | Key and Value at step $t$ |
| $\hat{\mathbf{K}}_i^{t-1}, \hat{\mathbf{V}}_i^{t-1}$ | Cached Key and Value |
| $h$ | Number of attention heads per layer |
| $L$ | Number of transformer layers |
| $P(x_{t+1}\|x_1, \cdots, x_t)$ | Conditional probability |

of the transformer model and then introduce the mechanisms for managing the KV cache to accelerate the LLMs. Important notations in this survey are summarized in Tab. 1.

## 2.1 Transformer Architecture

Transformers Vaswani et al. (2017) have become the backbone of LLMs due to their ability to efficiently capture long-range dependencies sequential data, such as text. This capability makes them particularly well-suited for tasks like machine translation, text generation, and image captioning. The transformer architecture follows an encoder-decoder structure, where most LLMs utilize only the decoder component. We first introduce the core components of the Transformer decoder and then describe the critical auto-regressive generation mechanism. Particularly, we do not describe certain components in transformer, such as normalization, as they do not impact the understanding of KV cache management.

### 2.1.1 Transformer Decoder

As shown in Figure 1, a decoder-based transformer architecture is composed of multiple stacked Transformer blocks, each designed to process sequential data effectively. Typically, a Transformer block consists of two core components, i.e., a Multi-Head Self-Attention (MHSA) mechanism and a Feed Forward Network (FFN). These blocks are arranged sequentially, where the output of one block is passed as input to the next. This iterative design allows the model to refine its understanding of the input sequence progressively, making it highly effective for tasks such as text generation and language modeling.

**Positional Encoding.** Before the input sequence is processed by the Transformer blocks, it undergoes a preprocessing phase. First, a tokenizer processes the input sentence $X$ by splitting it into discrete units, such as words or subwords. The resulting sequence can be represented as $X = [x_1, x_2, \cdots, x_{|X|}]$. These tokens are then mapped to dense vector representations using an embedding layer, i.e., $\mathbf{X} = \mathbf{I}_X \mathbf{E}^\top$, where $\mathbf{I}_X \in \{0,1\}^{n \times d_{\text{vocab}}}$ represents the one-hot vector of tokenized input $X$, $\mathbf{E} \in \mathbb{R}^{d_{\text{vocab}} \times d_x}$ is the embedding matrix, and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{|X|}] \in \mathbb{R}^{n \times d_x}$ is the resulting matrix of embedded token representations. Since the Transformer architecture does not inherently account for the order of tokens in a sequence, **positional encodings** are added to the token embeddings $\mathbf{X}$ to incorporate positional information. This can be expressed as $\mathbf{X} = \mathbf{X} + PE(X)$, where $PE(X) \in \mathbb{R}^{n \times d_x}$ represents a function Zhao et al. (2023); Zheng et al.

(2021); Su et al. (2024) (e.g., sine and cosine-based positional encoding) that generates positional embeddings for the input $X$.

**Transformer Block.** Once the input features are prepared, they are passed through a series of stacked Transformer blocks. Each block begins with the Multi-Head Self-Attention (MHSA) mechanism, which captures both local and global dependencies. For each token, the self-attention mechanism computes a weighted sum over all other tokens in the sequence, where the weights are derived from the similarity between the tokens. Particularly, since the operations within each transformer block are identical, we use a single transformer block as an example. Specifically, given the input to a block, denoted as $\mathbf{X} \in \mathbb{R}^{|X| \times d}$, the MHSA mechanism computes the query vectors $\mathbf{Q}_i \in \mathbb{R}^{|X| \times d_k}$, key vectors $\mathbf{K}_i \in \mathbb{R}^{|X| \times d_k}$, and value vectors $\mathbf{V}_i \in \mathbb{R}^{|X| \times d_v}$. These vectors are obtained through learned linear transformations as follows:

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_{Q_i}, \quad \mathbf{K}_i = \mathbf{X}\mathbf{W}_{K_i}, \quad \mathbf{V}_i = \mathbf{X}\mathbf{W}_{V_i}, \tag{1}$$

where $\mathbf{W}_{Q_i} \in \mathbb{R}^{d_x \times d_k}$, $\mathbf{W}_{K_i} \in \mathbb{R}^{d_x \times d_k}$ and $\mathbf{W}_{V_i} \in \mathbb{R}^{d_x \times d_v}$ are the learned weight parameters. Then, the self-attention operation is applied to each triple $(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i)$, and obtain the output of the $i$-th attention head $\mathbf{Z}_i$ as follows:

$$\mathbf{Z}_i = \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \text{Softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d_k}}\right)\mathbf{V}_i, \tag{2}$$

where $\sqrt{d_k}$ is a scaling factor to ensure the numerical stability. To capture diverse relationships, multiple attention with $h$ heads are applied to $\mathbf{X}$ in parallel, and their outputs are concatenated with one transformation as follows:

$$\mathbf{Z} = \text{Concat}(\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_h)\mathbf{W}_O, \tag{3}$$

where Concat is concatenation operation and $\mathbf{W}_O \in \mathbb{R}^{d_v \times d_o}$ is the trainable parameters.

Following the self-attention mechanism, the output is passed through a **Feed Forward Network (FFN)**. The FFN is a fully connected neural network that applies two linear transformations separated by a nonlinear activation function $\sigma(\cdot)$ (e.g, ReLU Agarap (2018)) :

$$\text{FFN}(\mathbf{Z}) = \sigma(\mathbf{Z}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_o \times d_1}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_1 \times d_2}$ are two parameters, $\mathbf{b}_1 \in \mathbb{R}^{d_1}$ and $\mathbf{b}_2 \in \mathbb{R}^{d_2}$ are two bias vectors.

### 2.1.2 Auto-regressive Generation Mechanism

LLMs employ an autoregressive mechanism to generate text token by token, with each token conditioned on the previously generated ones. This iterative process ensures that the output sequence remains coherent and contextually appropriate. Formally, given an input sequence of tokens $X = [x_1, x_2, \cdots, x_t]$, the model predicts the next token $x_{t+1}$ at each decoding step $t$ by modeling the conditional probability distribution as follows:

$$P(x_{t+1}|x_1, x_2, \cdots, x_t) = \text{Softmax}(\mathbf{h}_t\mathbf{W}_{\text{out}} + \mathbf{b}_{\text{out}}), \tag{5}$$

where $\mathbf{h}_t \in \mathbb{R}^{d_h}$ represents the hidden state of the LLM regarding $X$ at step $t$, $\mathbf{W}_{\text{out}} \in \mathbb{R}^{d_h \times vocab}$ is the output projection matrix, and $\mathbf{b}_{\text{out}}$ is the bias vector. The softmax function converts the logits into a probability distribution over the vocabulary. Then, at each decoding step, the model generates the next token $x_{t+1}$ by sampling from the predicted probability distribution:

$$x_{t+1} \sim P(x_{t+1}|x_1, x_2, \cdots, x_t). \tag{6}$$

The generated token $x_{t+1}$ is then appended to the sequence $X = [x_1, \cdots, x_t, x_{t+1}]$, and the process continues until a special end-of-sequence (EOS) token is generated or a predefined maximum length is reached.

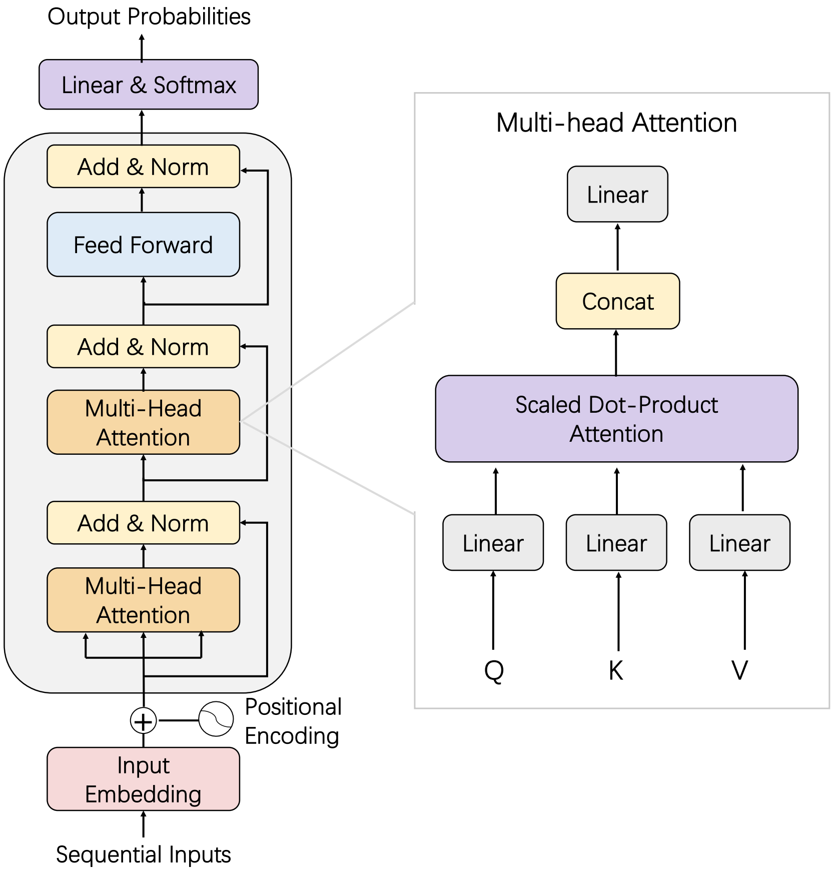

Figure 1: The decoder-only Transformer for LLMs.

### 2.2 Key-Value Cache in Transformer Models

Auto-regressive generation is a powerful mechanism that enables LLMs to produce high-quality, contextually coherent text. However, it presents computational challenges for long sequences, as the Keys and Values need to be recomputed for each token during the generation process. The KV cache optimization addresses this issue by storing the previously computed Keys and Values and reusing them for subsequent token generation, thereby reducing redundant computations and improving inference efficiency.

#### 2.2.1 Auto-regressive Generation with KV Cache

Here, we describe how caching KV pairs of tokens accelerates LLM inference. Specifically, at each decoding step $t$, the model performs self-attention over the entire sequence $X = [x_1, \cdots, x_{t-1}, x_t]$ to generate the next token $x_{t+1}$. This process requires the computation of Keys and Values matrices for all previously processed tokens in $X = [x_1, \cdots, x_t]$. Notably, when generating the token $x_t$, the LLM has already computed the Keys and Values for the tokens in $X[1 : t-1] = [x_1, \cdots, x_{t-1}]$. The KV cache optimizes this process by storing the previously computed Keys and Values matrices for $X[1 : t-1]$ and reusing them, thereby only requiring the computation of Keys and Values for the new token $x_t$. This significantly improves efficiency by eliminating redundant computations.

Formally, at decoding step $t$, the new token embedding $\mathbf{x}_t$ is used to compute the query vector $\mathbf{q}_i^t$, key vector $\mathbf{k}_i^t$, and value vector $\mathbf{v}_i^t$ as follows:

$$\mathbf{q}_i^t = \mathbf{x}_t \mathbf{W}_{Q_i}, \quad \mathbf{k}_i^t = \mathbf{x}_t \mathbf{W}_{K_i}, \quad \mathbf{v}_i^t = \mathbf{x}_t \mathbf{W}_{V_i}, \tag{7}$$

The newly computed $\mathbf{k}_i^t$ and $\mathbf{v}_i^t$ are then appended to the cached key and value matrices from previous steps:

$$\mathbf{K}_i^t = \text{Concat}(\hat{\mathbf{K}}_i^{t-1}, \mathbf{k}_i^t),\ \mathbf{V}_i^t = \text{Concat}(\hat{\mathbf{V}}_i^{t-1}, \mathbf{V}_i^t), \tag{8}$$

where $\hat{\mathbf{K}}_i^{t-1} \in \mathbb{R}^{t-1 \times d_k}$ and $\hat{\mathbf{V}}_i^{t-1} \in \mathbb{R}^{t-1 \times d_v}$ represent the cached key and value matrices of tokens in $X[1:t-1]$. These cached matrices are then used in the scaled dot-product attention computation for token $x_t$. The attention output $\mathbf{z}_i^t$ for the token $x_t$ at step $t$ is calculated as:

$$\mathbf{z}_i^t = \text{Softmax}\left(\frac{\mathbf{q}_i^t \mathbf{K}_i^{t\top}}{\sqrt{d_k}}\right)\mathbf{V}_i^t, \tag{9}$$

Then, a similar KV reuse process can be applied to different attention heads in each layer of the LLM.

### 2.2.2 Time and Space Complexity Analysis

Given a transformer-based $L$-layer LLM with $h$ attention heads per layer and an input sequence of length $X = [x_1, \cdots, x_t]$, we analyze the time saved and the space required to store cached KV pairs. For simplicity, we assume the Keys and Values of $t_c$ tokens are stored for all heads across all LLM layers.

**Saved Time.** For each token, the saved computation time comes from avoiding the repeated computation of Keys and Values in Equation equation 1, self-attention result in Equation equation 2, and linear transformation in Equation equation 3. We omit the time analyze on operations in transformer that do not affect the understanding of KV cache acceleration, such as layer norm and position encoding.

- **QKV Computation.** The time of computing Queries, Keys and Values for each token in Equation equation 1 is $\triangle_1 = O(2d_xd_k + d_xd_v)$.
- **Self-attention Result.** Additionally, computing each attention result $\mathbf{z}_i$ in Equation equation 2 takes $O(t(d_k + d_v))$.
- **Linear Transformation.** To merge the $h$ attention results in Equation equation 3 the time is $\triangle_2 = O(hd_v + d_vd_o)$.

Therefore, for $t_c$ cached tokens across $h$ attention heads and $L$ layers, the total saved computation time is:

$$O\left(L \cdot h \cdot t_c \cdot t \cdot (d_k + d_v) + L \cdot h \cdot t_c\left(\triangle_1 + \triangle_2\right)\right) \tag{10}$$

Thus, the saved time is directly proportional to the number of cached tokens $t_c$, significantly accelerating model computation, especially for longer sequences (when $t$ is large).

**Extra Space.** Compared to computation without caching, additional space is required to store the cached KV pairs for $t_c$ tokens across $h$ attention heads and $L$ layers. Assuming each Key and Value is stored in Float16 precision, the total extra space needed can be expressed as:

$$O(L \cdot h \cdot t_c \cdot 2 \cdot sizeof(Float16)) \tag{11}$$

Thus, for the same LLM model, the extra space required to store the KV pairs primarily depends on the number of cached tokens and the precision of the cached Keys and Values. To address this, existing approaches explore various techniques to reduce the extra space consumption, such as caching only the most important Keys and Values or applying quantization techniques to lower the bit precision of the stored Keys and Values.

## 2.3 Challenges in KV Cache Management

As analyzed in Sec. 2.2.2, reusing cached KV pairs enables the LLM to avoid recomputing past tokens, resulting in significant speedups during inference. However, as sequence lengths grow, the size of the KV cache increases proportionally, placing significant pressure on memory. Consequently, it becomes challenging to manage this cache effectively to accelerate LLM computation without excessive space usage.

- **Cache Eviction Policies:** Determining which items to evict when the cache reaches its capacity is a complex problem. Popular policies Podlipnig & Böszörmenyi (2003) like Least Recently Used (LRU) or Least Frequently Used (LFU) do not align with LLMs patterns, leading to suboptimal performance.
- **Memory Management:** The memory required for the KV cache grows linearly with both the sequence length and the number of layers, which can quickly exceed the hardware memory limits, especially for long sequences. Consequently, managing the collaboration between different types of storage hardware (e.g., GPU, CPU, or external memory) becomes a significant challenge.
- **Latency Bottlenecks:** Accessing and updating the cache at each decoding step can introduce latency, particularly for hardware with limited memory bandwidth.
- **Compression Trade-offs:** Compressing the KV cache can reduce memory usage but may degrade model performance if key information is lost.
- **Dynamic Workloads:** Handling dynamic and unpredictable workloads, where access patterns and data requirements frequently change, requires adaptive caching strategies that can respond in real time.
- **Distributed Coordination:** In distributed KV caches, maintaining coordination across multiple nodes to ensure consistency, fault tolerance, and efficient resource usage adds significant complexity.

## 3 Taxonomy

In the above sections, we analyzed how the number of cached Key-Value (KV) pairs significantly impacts both the computation time and the additional memory required during inference. Efficient KV cache management is critical to balancing performance improvements and resource utilization, especially as sequence lengths and model sizes continue to grow. After carefully reviewing existing approaches, we categorize KV cache optimization strategies into three levels: token-level optimization, model-level optimization, and system-level optimizations. Each level addresses specific aspects of the challenges associated with KV cache management and offers distinct techniques to enhance efficiency. The detailed taxonomy is illustrated in Fig. 2.

- **Token-Level Optimization** refers to improving KV cache management efficiency by focusing on fine-grained the careful selection, organization, and compression at the token level, requiring no architectural changes to the original model. While KV cache selection (Sec. 4.1) focuses on prioritizing and storing only the most relevant tokens. KV cache budget allocation (Sec. 4.2) dynamically distributes memory resources across tokens to ensure efficient cache utilization under limited memory. Furthermore, KV cache merging (Sec. 4.3) reduces redundancy by combining similar or overlapping KV pairs, while KV Cache Quantization (Sec. 4.4) minimizes the memory footprint by reducing the precision of cached KV pairs. Finally, KV cache low-rank decomposition (Sec. 4.5) uses low-rank decomposition technique to reduce cache size.
- **Model-level Optimization** refers to designing an efficient model structure to optimize KV cache management. This can further refer to several strategies: Attention grouping and sharing (Sec. 5.1) methods examine the redundant functionality of key and values and group and share KV cache within or across transformer layers. Architecture alterations (Sec. 5.2 emerge to design new attention mechanisms or construct extrinsic modules for KV optimization. Furthermore, there are also works designing or combining non-transformer architectures 5.3 that adopt other memory efficient designs like recurrent neural networks to optimize the KV cache in traditional transformers.
- **System-level Optimization** refers to optimizing the KV Cache management through two classic low-level aspects: memory management (Sec. 6.1) and scheduling (Sec. 6.2). While memory management techniques focusing on architectural innovations like virtual memory adaptation, intelligent prefix sharing, and layer-aware resource allocation, scheduling strategies have evolved to address diverse optimization goals through prefix-aware methods for maximizing cache reuse, preemptive techniques for fair context switching, and layer-specific mechanisms for fine-grained cache control. In addition, we provide a detailed introduction for hardware accelerator design in Sec. 6.3, including single/multi-GPU, I/O-based solutions, heterogeneous computing and SSD-based solutions.

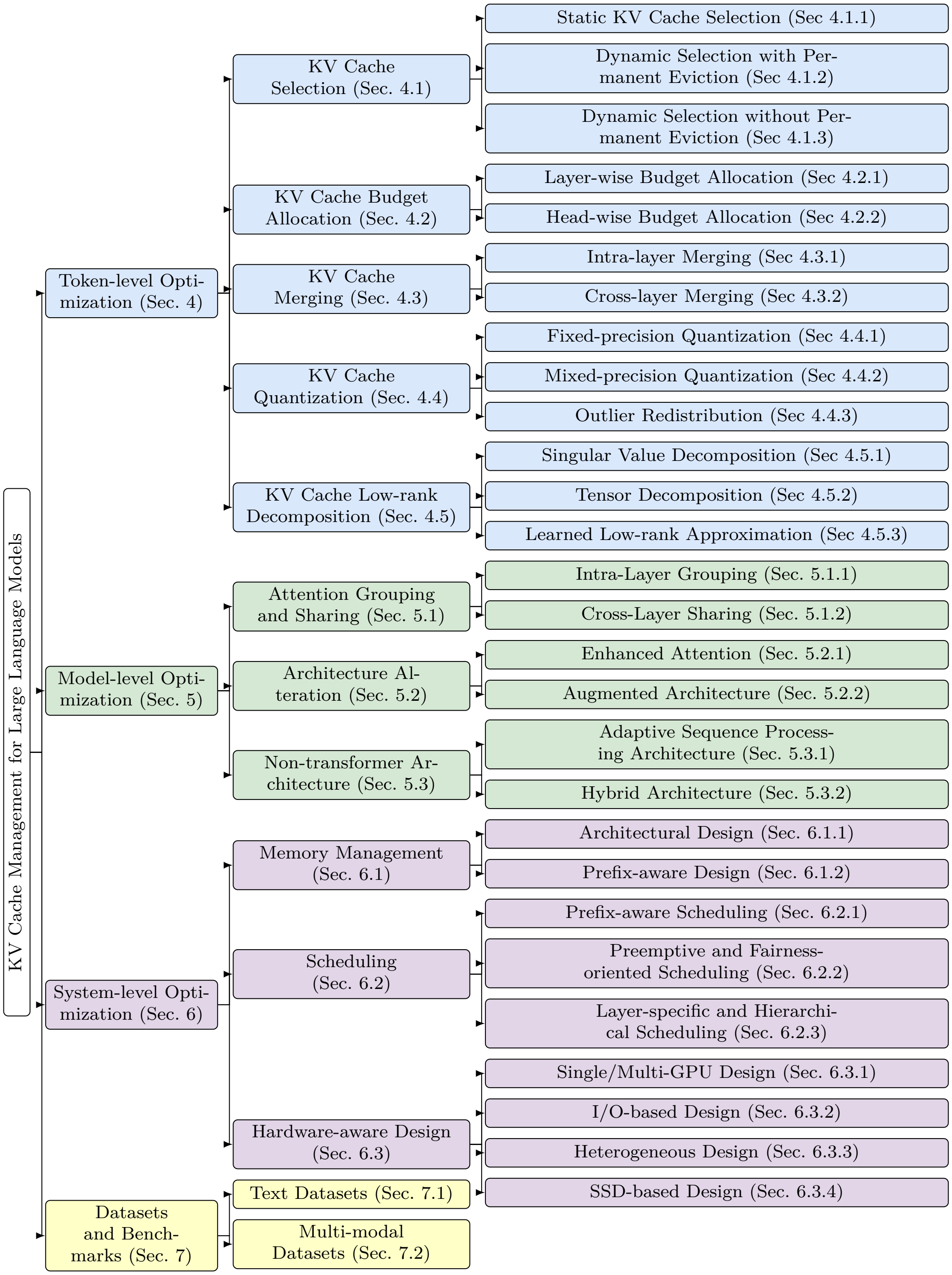
KV Cache Management for Large Language Models
Token-level Optimization (Sec. 4)
KV Cache Selection (Sec. 4.1)
Static KV Cache Selection (Sec 4.1.1)
Dynamic Selection with Permanent Eviction (Sec 4.1.2)
Dynamic Selection without Permanent Eviction (Sec 4.1.3)
KV Cache Budget Allocation (Sec. 4.2)
Layer-wise Budget Allocation (Sec 4.2.1)
Head-wise Budget Allocation (Sec 4.2.2)
KV Cache Merging (Sec. 4.3)
Intra-layer Merging (Sec 4.3.1)
Cross-layer Merging (Sec 4.3.2)
KV Cache Quantization (Sec. 4.4)
Fixed-precision Quantization (Sec 4.4.1)
Mixed-precision Quantization (Sec 4.4.2)
Outlier Redistribution (Sec 4.4.3)
KV Cache Low-rank Decomposition (Sec. 4.5)
Singular Value Decomposition (Sec 4.5.1)
Tensor Decomposition (Sec 4.5.2)
Learned Low-rank Approximation (Sec 4.5.3)
Model-level Optimization (Sec. 5)
Attention Grouping and Sharing (Sec. 5.1)
Intra-Layer Grouping (Sec. 5.1.1)
Cross-Layer Sharing (Sec. 5.1.2)
Architecture Alteration (Sec. 5.2)
Enhanced Attention (Sec. 5.2.1)
Augmented Architecture (Sec. 5.2.2)
Non-transformer Architecture (Sec. 5.3)
Adaptive Sequence Processing Architecture (Sec. 5.3.1)
Hybrid Architecture (Sec. 5.3.2)
System-level Optimization (Sec. 6)
Memory Management (Sec. 6.1)
Architectural Design (Sec. 6.1.1)
Prefix-aware Design (Sec. 6.1.2)
Scheduling (Sec. 6.2)
Prefix-aware Scheduling (Sec. 6.2.1)
Preemptive and Fairness-oriented Scheduling (Sec. 6.2.2)
Layer-specific and Hierarchical Scheduling (Sec. 6.2.3)
Hardware-aware Design (Sec. 6.3)
Single/Multi-GPU Design (Sec. 6.3.1)
I/O-based Design (Sec. 6.3.2)
Heterogeneous Design (Sec. 6.3.3)
SSD-based Design (Sec. 6.3.4)
Datasets and Benchmarks (Sec. 7)
Text Datasets (Sec. 7.1)
Multi-modal Datasets (Sec. 7.2)

Figure 2: Taxonomy of KV Cache Management for Large Language Models.

## 4 Token-level Optimization

In the token level, optimization focuses exclusively on improving KV cache based on the characteristics and patterns of KV pairs of tokens, without considering enhancements from model architecture improvements

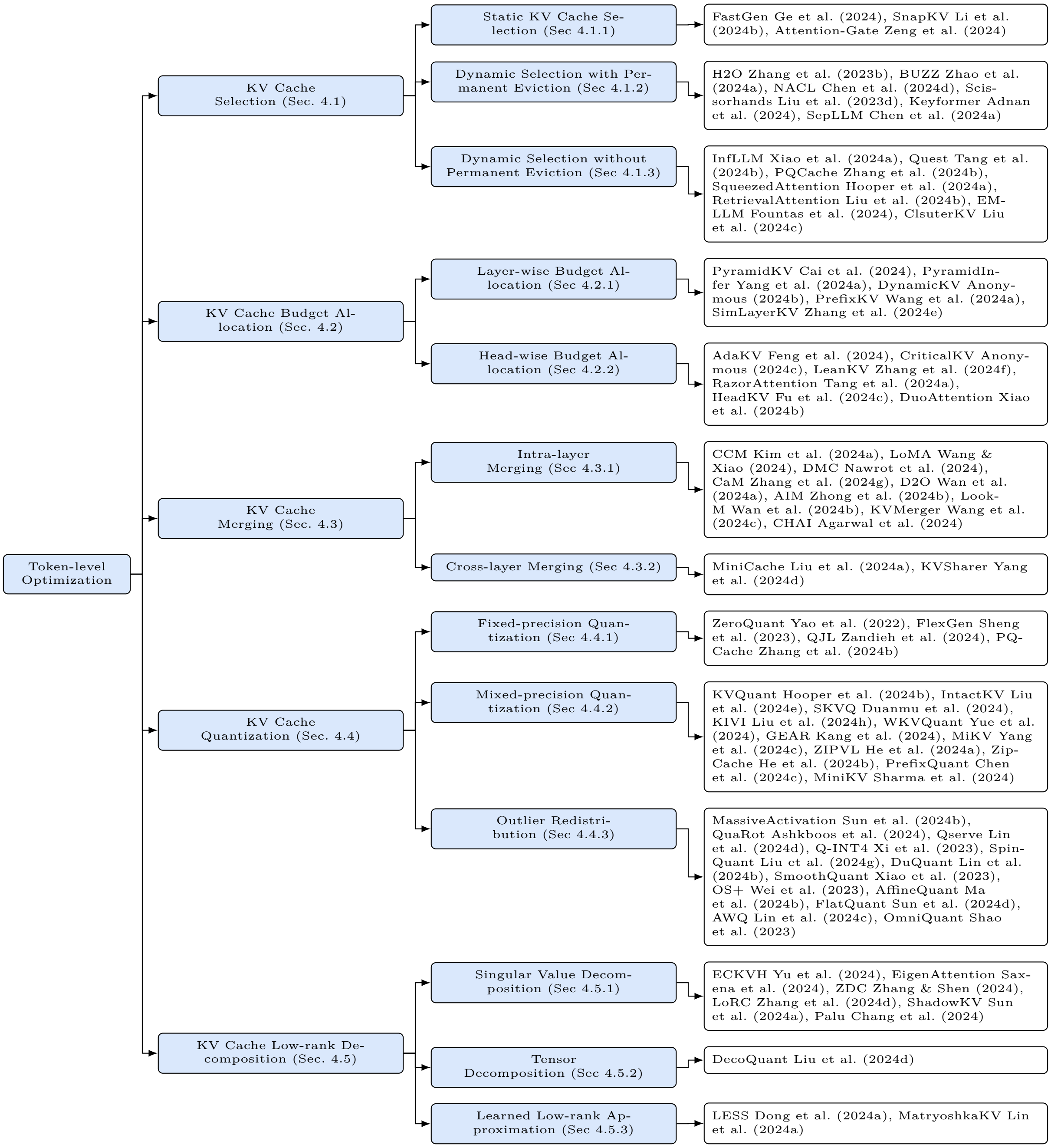
Token-level Optimization
KV Cache Selection (Sec. 4.1)
Static KV Cache Selection (Sec 4.1.1)
FastGen Ge et al. (2024), SnapKV Li et al. (2024b), Attention-Gate Zeng et al. (2024)
Dynamic Selection with Permanent Eviction (Sec 4.1.2)
H2O Zhang et al. (2023b), BUZZ Zhao et al. (2024a), NACL Chen et al. (2024d), Scissorhands Liu et al. (2023d), Keyformer Adnan et al. (2024), SepLLM Chen et al. (2024a)
Dynamic Selection without Permanent Eviction (Sec 4.1.3)
InfLLM Xiao et al. (2024a), Quest Tang et al. (2024b), PQCache Zhang et al. (2024b), SqueezedAttention Hooper et al. (2024a), RetrievalAttention Liu et al. (2024b), EM-LLM Fountas et al. (2024), ClsuterKV Liu et al. (2024c)
KV Cache Budget Allocation (Sec. 4.2)
Layer-wise Budget Allocation (Sec 4.2.1)
PyramidKV Cai et al. (2024), PyramidInfer Yang et al. (2024a), DynamicKV Anonymous (2024b), PrefixKV Wang et al. (2024a), SimLayerKV Zhang et al. (2024e)
Head-wise Budget Allocation (Sec 4.2.2)
AdaKV Feng et al. (2024), CriticalKV Anonymous (2024c), LeanKV Zhang et al. (2024f), RazorAttention Tang et al. (2024a), HeadKV Fu et al. (2024c), DuoAttention Xiao et al. (2024b)
KV Cache Merging (Sec. 4.3)
Intra-layer Merging (Sec 4.3.1)
CCM Kim et al. (2024a), LoMA Wang & Xiao (2024), DMC Nawrot et al. (2024), CaM Zhang et al. (2024g), D2O Wan et al. (2024a), AIM Zhong et al. (2024b), Look-M Wan et al. (2024b), KVMerger Wang et al. (2024c), CHAI Agarwal et al. (2024)
Cross-layer Merging (Sec 4.3.2)
MiniCache Liu et al. (2024a), KVSharer Yang et al. (2024d)
KV Cache Quantization (Sec. 4.4)
Fixed-precision Quantization (Sec 4.4.1)
ZeroQuant Yao et al. (2022), FlexGen Sheng et al. (2023), QJL Zandieh et al. (2024), PQ-Cache Zhang et al. (2024b)
Mixed-precision Quantization (Sec 4.4.2)
KVQuant Hooper et al. (2024b), IntactKV Liu et al. (2024e), SKVQ Duanmu et al. (2024), KIVI Liu et al. (2024h), WKVQuant Yue et al. (2024), GEAR Kang et al. (2024), MiKV Yang et al. (2024c), ZIPVL He et al. (2024a), ZipCache He et al. (2024b), PrefixQuant Chen et al. (2024c), MiniKV Sharma et al. (2024)
Outlier Redistribution (Sec 4.4.3)
MassiveActivation Sun et al. (2024b), QuaRot Ashkboos et al. (2024), Qserve Lin et al. (2024d), Q-INT4 Xi et al. (2023), SpinQuant Liu et al. (2024g), DuQuant Lin et al. (2024b), SmoothQuant Xiao et al. (2023), OS+ Wei et al. (2023), AffineQuant Ma et al. (2024b), FlatQuant Sun et al. (2024d), AWQ Lin et al. (2024c), OmniQuant Shao et al. (2023)
KV Cache Low-rank Decomposition (Sec. 4.5)
Singular Value Decomposition (Sec 4.5.1)
ECKVH Yu et al. (2024), EigenAttention Saxena et al. (2024), ZDC Zhang & Shen (2024), LoRC Zhang et al. (2024d), ShadowKV Sun et al. (2024a), Palu Chang et al. (2024)
Tensor Decomposition (Sec 4.5.2)
DecoQuant Liu et al. (2024d)
Learned Low-rank Approximation (Sec 4.5.3)
LESS Dong et al. (2024a), MatryoshkaKV Lin et al. (2024a)

Figure 3: Taxonomy of the Token-level Optimization for KV Cache Management.

or system parallelization techniques. In general, token-level optimization methods are primarily guided by observations from LLMs and sequential inputs. Existing approaches can be categorized into five main types: KV cache selection, KV cache budget allocation, KV cache merging, KV cache quantization, and KV cache low-rank decomposition. The taxonomy of the token-level optimization is shown in Fig. 3.

## 4.1 KV Cache Selection

KV cache selection mechanisms have emerged as a critical optimization strategy, aimed at reducing memory utilization of KV caches, minimizing inference latency, and enhancing overall throughput in large language models. These optimization objectives have driven the development of various selection methodologies, which can be classified into two distinct categories: (1) **static KV cache selection**, which performs token filtering exclusively during the prefilling phase, with selected tokens remaining fixed throughout subsequent decoding steps; and (2) **dynamic KV cache selection**, which continuously updates KV cache during the decoding phase, enabling adaptive cache management. In dynamic KV cache selection approaches, KV cache tokens that are not selected may be permanently evicted or offloaded to hierarchical caching devices such as CPU memory, implementing a multi-tier storage strategy. Given that real-time KV cache selection during decoding may incur substantial computational overhead, several studies have focused on developing optimized retrieval algorithms to enhance the efficiency of this process. These optimizations include block-level retrieval instead of token-level granularity to reduce search complexity, asynchronous query mechanisms to hide latency, and parallel retrieval pipelines to accelerate the selection process. These optimization efforts aim to mitigate the computational burden while maintaining the effectiveness of token selection. The summary of the KV cache selection is listed in Tab. 2.

### 4.1.1 Static KV Cache Selection

Static KV cache selection methods perform a one-time compression on the KV Cache immediately after the prefilling phase is completed. The model then uses this compressed KV cache for subsequent decoding inference. FastGen Ge et al. (2024) introduces a pattern-aware approach by identifying five fundamental attention structures and implementing targeted selection strategies. These include proximity-based retention for local attention patterns, selective preservation of critical tokens for punctuation-focused attention, frequency-based filtering for sparse attention distributions, and complete token retention for broad attention patterns. SnapKV Li et al. (2024b) simplifies FastGen's approach by focusing solely on retrieving tokens based on their importance scores. It demonstrates that among all prompt tokens, only a portion carries crucial information for response generation, with these tokens maintaining their significance during the generation phase. The approach employs an end-positioned observation window to detect these important contextual tokens. Their corresponding key-value pairs are then concatenated with the tokens from the observation window. Attention-Gate Zeng et al. (2024) introduces a learnable KV-Cache eviction mechanism that processes the entire context sequence and generates token-wise eviction decisions through a parameterized policy network, enabling dynamic in-context memory management.

### 4.1.2 Dynamic Selection with Permanent Eviction

This category of methods performs frequent KV cache selection during the decoding phase, permanently removing unselected KV cache tokens from memory. Early works employ a sliding-window mechanism to address long-text inference challenges, where tokens falling outside the window are permanently evicted and become inaccessible. StreamingLLM Xiao et al. (2024c) uncovers a crucial phenomenon in transformer attention where preserved key-value pairs from initial sequence tokens maintain crucial model performance. This attention sink effect manifests through asymmetric attention weight accumulation at early positions, regardless of semantic significance. The approach leverages this characteristic by incorporating attention sink positions with recent context for efficient processing. LM-Infinite Han et al. (2024) demonstrates that conventional techniques, including sliding-window patterns and relative positional encodings, fail to resolve length generalization issues. The study introduces a novel methodology through the integration of Λ-shaped attention masking and attention distance ceiling mechanisms.

Recent works have explored leveraging attention scores as a criterion for selecting significant KV cache tokens. H2O Zhang et al. (2023b) observes that attention computations are primarily driven by a select group of high-impact tokens, known as Heavy Hitters (H2). This method reformulates cache optimization as a dynamic submodular problem, utilizing cumulative attention scores to guide token retention decisions. Unlike H2O, BUZZ Zhao et al. (2024a) employs a beehive-like structure that selects Heavy Hitters in local KV cache segments. NACL Chen et al. (2024d) identifies a fundamental limitation in H2O, namely their dependence on potentially biased local attention statistics. To overcome this issue, they develop an alternative approach implementing a diversified random eviction strategy for token selection. Scissorhands Liu et al. (2023d) builds upon the temporal significance principle, which suggests that tokens demonstrating historical importance maintain their influence in subsequent computational steps. This observation enables the preservation of

Table 2: Comparison of KV cache selection strategies.

| Method | Initial tokens | Top-$k$ tokens | Recent tokens | Permanent eviction | Dynamic selection | Selection granularity | Remark |
|---|---|---|---|---|---|---|---|
| **FastGen**<br>Ge et al. (2024) | ✓ | ✓ | ✓ | ✓ |  | token | five attention structures |
| **SnapKV**<br>Li et al. (2024b) |  | ✓ | ✓ | ✓ |  | token | observation window-based |
| **Attention-Gate**<br>Zeng et al. (2024) |  | ✓ |  | ✓ |  | token | learned eviction policy |
| **StreamingLLM**<br>Xiao et al. (2024c) | ✓ |  | ✓ | ✓ | ✓ | token | initial and recent tokens |
| **LM-Infinite**<br>Han et al. (2024) | ✓ |  | ✓ | ✓ | ✓ | token | distance ceiling |
| **H2O**<br>Zhang et al. (2023b) |  | ✓ | ✓ | ✓ | ✓ | token | accmulative attention score |
| **BUZZ**<br>Zhao et al. (2024a) | ✓ | ✓ | ✓ | ✓ | ✓ | token | beehive-like structure |
| **Scissorhands**<br>Liu et al. (2023d) |  | ✓ | ✓ | ✓ | ✓ | token | persistence of importance |
| **NACL**<br>Chen et al. (2024d) |  | ✓ | ✓ | ✓ | ✓ | token | diversified random eviction |
| **Keyformer**<br>Adnan et al. (2024) |  | ✓ | ✓ | ✓ | ✓ | token | gumbel logit adjustment |
| **InfLLM**<br>Xiao et al. (2024a) | ✓ | ✓ | ✓ |  | ✓ | block | block-level KV management |
| **Quest**<br>Tang et al. (2024b) |  | ✓ |  |  | ✓ | block | new block representation |
| **PQCache**<br>Zhang et al. (2024b) | ✓ | ✓ | ✓ |  | ✓ | block | product quantization |
| **SqueezedAttention**<br>Hooper et al. (2024a) |  | ✓ |  |  | ✓ | cluster | hierarchical clusters |
| **RetrievalAttention**<br>Liu et al. (2024b) | ✓ | ✓ | ✓ |  | ✓ | token | ANN search |
| **EM-LLM**<br>Fountas et al. (2024) | ✓ | ✓ | ✓ |  | ✓ | event | episodic events |
| **SparQ**<br>Ribar et al. (2024) |  | ✓ | ✓ |  | ✓ | token | low-dimensional retrieval |
| **InfiniGen**<br>Lee et al. (2024) |  | ✓ |  |  | ✓ | token | asynchronous prefetching |
| **RecycledAttention**<br>Xu et al. (2024b) |  | ✓ | ✓ |  | ✓ | token | periodic top-$k$ selection |
| **MagicPIG**<br>Chen et al. (2024g) | ✓ | ✓ | ✓ |  | ✓ | token | Local Sensitive Hash |

repetitive attention patterns through selective token retention. Additionally, Keyformer Adnan et al. (2024) reveals that token removal distorts the underlying softmax probability distribution. Considering the pivotal role of softmax distributions in token significance evaluation, they incorporate regularization techniques to mitigate these distributional perturbations. SepLLM Chen et al. (2024a) observes that separator tokens (e.g., commas, periods, and line breaks) receive disproportionately high attention scores and naturally summarize text segments. Building on this, SepLLM retains separator tokens together with initial tokens, important tokens, and recent tokens in the cache.

#### 4.1.3 Dynamic Selection without Permanent Eviction

The aforementioned permanent eviction-based approaches face two significant limitations. First, the irreversible eviction of tokens potentially impairs the model's performance on long-sequence tasks, particularly in needle-in-a-haystack scenarios, and these methods prove challenging to adapt to multi-turn dialogue contexts. Second, KV cache selection during the decoding phase introduces computational overhead, adversely affecting decoding latency and compromising end-to-end acceleration. To address these challenges, several studies have focused on developing decoding-phase KV cache selection strategies without permanent eviction. These approaches typically employ multi-tier cache systems (e.g., CPU-GPU hierarchical caching) and leverage advanced data structures and system-level enhancements to optimize retrieval efficiency, enabling efficient inference with reduced GPU KV cache footprint.

To accelerate the retrieval of critical tokens, several research efforts have proposed index-based approaches that organize and access KV cache at block or cluster granularity, enabling efficient query and extraction operations. InfLLM Xiao et al. (2024a) maintains full KV cache in blocks while facilitating long sequence processing through a hierarchical storage strategy. The framework employs CPU-GPU memory orchestration, preserving essential tokens and current computational units in GPU memory while offloading less frequently accessed units to CPU memory. To further enhance top-$k$ block retrieval precision, the Quest Tang et al. (2024b) framework presents a refined block representation approach based on minimal and maximal key values in KV cache blocks. PQCache Zhang et al. (2024b) also implements block-based KV cache management and identifies salient tokens through Maximum Inner-Product Search (MIPS), leveraging Product Quantization (PQ) codes and centroids. SqueezedAttention Hooper et al. (2024a) employs K-means clustering in an offline stage to group semantically similar keys, with each group represented by a centroid. During inference, it compares input queries against these centroids to identify and load only the semantically relevant keys from the context. Similarly, RetrievalAttention Liu et al. (2024b) index KV cache tokens using approximate nearest neighbor search (ANNS) techniques. Additionally, EM-LLM Fountas et al. (2024) dynamically segments incoming tokens into episodic events. Besides, it implements a hybrid retrieval mechanism that combines semantic similarity matching with temporal context to efficiently access relevant KV cache segments. Similarly, ClusterKV Liu et al. (2024c) groups tokens into semantic clusters and selectively recalls them during inference, achieving both high accuracy and efficiency for LLMs.

To accelerate top-$k$ token identification, SparQ Ribar et al. (2024) identifies the $r$ most significant elements in the incoming query vector and selectively retrieves the corresponding components along the hidden dimension of the cached key matrix $K$ for approximate attention computation. To overlap prefetching latency, InfiniGen Lee et al. (2024) employs asynchronous prefetching, utilizing indices of salient KV entries selected by queries from the previous layer to retrieve KV cache entries in the current layer. To ensure maximum model performance, RecycledAttention Xu et al. (2024b) sustains the entire KV cache during inference computations, yielding no improvements in memory efficiency. The approach performs periodic top-$k$ token selection to identify salient tokens. Moreover, MagicPIG Chen et al. (2024g) shows that attention-based top-$k$ selection may incur performance degradation. To address this limitation, they introduce a novel heterogeneous computing framework leveraging Locality Sensitive Hashing (LSH) techniques. The system stores LSH hash tables and performs attention estimation on CPU.

#### 4.1.4 Summary and Future Directions

Static KV cache selection algorithms demonstrate superior decoding efficiency overall; however, their efficacy remains to be thoroughly validated in multi-turn dialogues and extended decoding length scenarios. Dynamic KV cache selection algorithms, while adaptive, introduce additional computational overhead during the decoding phase due to frequent cache selection operations. Multi-tier cache architectures and prefetching schemes partially mitigate these challenges, yet their capability to achieve rapid and accurate retrieval within acceptable decoding latency constraints requires further empirical validation, particularly in real-world applications involving long sequences. Furthermore, existing selection methods predominantly rely on attention score-based top-$k$ selection mechanisms. However, based on existing positional encoding schemes, current top-$k$ approaches may not be able to effectively identify and extract relevant tokens in ultra-long sequence tasks.

Table 3: Comparison of KV cache budget allocation strategies. **Extra**: Extra-calibration

| Method | Layer-wise | Head-wise | Retrieval-head | Input-specific | Extra | Remark |
|---|---|---|---|---|---|---|
| **PyramidKV**<br>Cai et al. (2024) | ✓ | | | | | pyramid-shaped |
| **PyramidInfer**<br>Yang et al. (2024a) | ✓ | | | | | pyramid-shaped |
| **DynamicKV**<br>Anonymous (2024b) | ✓ | | | ✓ | | maximize attention retention rate |
| **PrefixKV**<br>Wang et al. (2024a) | ✓ | | | ✓ | | maximize attention retention rate |
| **CAKE**<br>Anonymous (2024a) | ✓ | | | ✓ | | layer-specific preference score |
| **SimLayerKV**<br>Zhang et al. (2024e) | ✓ | | | ✓ | | KV cache compression for lazy layers |
| **AdaKV**<br>Feng et al. (2024) | | ✓ | | ✓ | | minimize attention computation loss |
| **CriticalKV**<br>Anonymous (2024c) | | ✓ | | ✓ | | minimize attention computation loss |
| **LeanKV**<br>Zhang et al. (2024f) | | ✓ | | ✓ | | maximize attention retention rate |
| **RazorAttention**<br>Tang et al. (2024a) | | ✓ | ✓ | | ✓ | echo and induction heads |
| **HeadKV**<br>Fu et al. (2024c) | | ✓ | ✓ | | ✓ | retrieval and reasoning heads |
| **DuoAttention**<br>Xiao et al. (2024b) | | ✓ | ✓ | | ✓ | learned retrieval heads |

### 4.2 KV Cache Budget Allocation

The hierarchical architecture of LLMs leads to diverse information extraction patterns across layers, with each layer's KV-cache contributing differently to model performance. This inherent heterogeneity indicates that uniform KV-cache compression across layers may be suboptimal. KV cache budget allocation addresses this challenge by intelligently distributing memory resources based on each component's importance to prediction accuracy, thereby optimizing memory utilization while minimizing accuracy degradation. Current budget allocation strategies can be categorized into two levels of granularity: **layer-wise** budget allocation, which assigns different compression ratios across model layers, and the more fine-grained **head-wise** budget allocation, which enables precise memory distribution across individual attention heads within each layer, offering more flexible and targeted optimization opportunities. The summary of KV budget allocation is listed in Tab. 3.

#### 4.2.1 Layer-wise Budget Allocation

In contrast to conventional approaches with uniform KV cache sizes, PyramidKV Cai et al. (2024) employs a pyramid-shaped memory allocation strategy, assigning larger cache capacities to lower layers that progressively decrease in upper layers. This design is supported by the observation that lower layers exhibit uniform attention distributions across input sequences, while upper layers show concentrated attention on specific tokens. PyramidInfer Yang et al. (2024a) also adopts a pyramid-shaped budget allocation strategy while selecting tokens with high attention values at each layer. Additionally, during the decoding phase, PyramidInfer dynamically maintains a set of significant tokens through frequent updates driven by attention values. Unlike previous methods, DynamicKV Anonymous (2024b) implements an input-adaptive budget allocation strategy by analyzing attention patterns. Specifically, it computes the average attention scores between recent and historical tokens, identifies the top-$k$ tokens with highest attention values across layers, and proportionally distributes the budget based on the density of significant tokens in each layer. Similarly,

Table 4: The summary of existing KV Cache merging approaches.

| Model | Merge Layer | | Merge Unit | Merge Metric | Merge Type | Training-free |
|---|---|---|---|---|---|---|
| | Intra-layer | Cross-layer | | | | |
| **CCM**<br>Kim et al. (2024a) | ✓ | | Token | Sliding Window | Many-to-One | × |
| **LoMA**<br>Wang & Xiao (2024) | ✓ | | Token | Sliding Window | Many-to-Many | × |
| **DMC**<br>Nawrot et al. (2024) | ✓ | | Token | Learned Merge Indictor | Many-to-One | × |
| **D2O**<br>Wan et al. (2024a) | ✓ | | Token | Cosine Similarity | Two-to-One | ✓ |
| **CaM**<br>Zhang et al. (2024g) | ✓ | | Token | Attention Score | Many-to-One | ✓ |
| **AIM**<br>Zhong et al. (2024b) | ✓ | | Token | Cosine Similarity | Many-to-One | ✓ |
| **Look-M**<br>Wan et al. (2024b) | ✓ | | Token | Cosine Similarity | Many-to-One | ✓ |
| **KVMerger**<br>Wang et al. (2024c) | ✓ | | Token | Weighted Gaussian Kernel | Many-to-One | ✓ |
| **CHAI**<br>Agarwal et al. (2024) | ✓ | | Head | Attention Score | Many-to-One | ✓ |
| **MinCache**<br>Liu et al. (2024a) | | ✓ | Token | Angular Distance | Two-to-One | ✓ |
| **KVSharer**<br>Yang et al. (2024d) | | ✓ | Layer | Euclidean Distance | Many-to-One | ✓ |

PrefixKV Wang et al. (2024a) identifies the most important tokens for each layer by computing the average attention score of tokens within that layer. PrefixKV Wang et al. (2024a) then uses a unified threshold to determine the number of retained tokens, adaptively adjusting the retention for each layer based on its importance distribution. CAKE Anonymous (2024a) examines attention scores through two lenses: the spatial distribution of inter-token attention and the temporal evolution of attention focus. These measurements are combined to compute layer-specific importance scores, which further guide the allocation of memory resources. Additionally, SimLayerKV Zhang et al. (2024e) identifies lazy layers - those exhibiting limited effectiveness in capturing long-range dependencies. The framework then selectively preserves cache entries, maintaining initial and recent tokens for lazy layers while retaining complete KV cache for non-lazy layers.

### 4.2.2 Head-wise Budget Allocation

AdaKV Feng et al. (2024) leverages the observation that attention patterns exhibit distinct concentrations across different heads. It implements head-specific memory allocation by optimizing an L1 loss bound between the original and pruned multi-head attention outputs. Within the constraints of a layer-wise budget, the method distributes cache capacity among heads to maximize the preserved attention information collectively. Building upon AdaKV, CriticalKV Anonymous (2024c) introduces significant enhancements by recognizing that the importance of KV cache entries extends beyond attention weights to encompass value states and pretrained parameter matrices. Leveraging this insight, the framework implements a novel selection algorithm that identifies essential cache entries by minimizing the maximum potential output perturbation. LeanKV Zhang et al. (2024f) implements a fine-grained memory optimization strategy that operates independently for each attention head and input request. The method identifies the smallest subset of tokens necessary to preserve the majority of information flow, allocating cache space based on a predefined attention score threshold - typically maintaining 95% of the total attention mass.

Retrieval head-based methods represent a specialized category of head-wise allocation strategies that focuses on identifying and prioritizing attention heads crucial for extracting key information from long sequences. This approach allocates larger cache budgets to these specialized heads, known as retrieval heads Wu et al. (2024d), due to their significant role in information extraction. RazorAttention Tang et al. (2024a) characterizes two distinct categories of retrieval heads: echo heads, which focus on previously occurring identical tokens, and induction heads, which attend to antecedent tokens that precede current token repetitions. This framework implements differential caching strategies, maintaining complete cache entries for retrieval heads while condensing remote tokens into consolidated compensation tokens for non-retrieval heads. HeadKV Fu et al. (2024c) further enhances RazorAttention by introducing a novel head assessment framework that simultaneously evaluates both retrieval and reasoning capabilities to optimize KV cache allocation strategies. DuoAttention Xiao et al. (2024b) further introduces a parameterized approach to distinguish between two categories of attention mechanisms: retrieval heads, essential for comprehensive long-context processing, and Streaming heads, which primarily engage with recent tokens and attention sinks. This classification is achieved through learned parameters that automatically identify retrieval heads requiring full attention spans.

### 4.2.3 Summary and Future Directions

Despite recent advances and growing attention in KV cache budget allocation research, several critical challenges remain unaddressed. First, the relationship between allocation strategies and model performance requires further investigation. For instance, a notable discrepancy exists between pyramid-shaped allocation strategies Cai et al. (2024); Yang et al. (2024a) advocating larger budgets for lower layers, and retrieval head-based studies Tang et al. (2024a); Fu et al. (2024c) which demonstrate that lower layers rarely exhibit retrieval head characteristics and thus require minimal cache resources. Additionally, the field lacks comprehensive experimental comparisons, particularly regarding the compatibility and performance benefits of head-wise budget allocation strategies with state-of-the-art frameworks like vLLM Kwon et al. (2023) and FlashAttention Dao et al. (2022). Also, existing methods, such as PyramidInfer Yang et al. (2024a), demonstrate some adaptability to input attention patterns. However, future research could target real-time, task-specific allocation strategies that dynamically adjust memory budgets during inference based on input characteristics, task complexity, or downstream requirements.

## 4.3 KV Cache Merging

KV cache merging offers a promising solution by compressing or consolidating KV caches without significantly degrading model accuracy. Rather than a uniform compression strategy, KV cache merging techniques leverage the inherent redundancy within and across layers to dynamically optimize memory utilization. These methods aim to reduce the size of KV caches while preserving critical information necessary for accurate attention computations, enabling efficient inference in resource-constrained settings. Existing KV cache merging strategies can be categorized into two primary approaches: **intra-layer merging**, which focuses on consolidating KV caches within individual layers to reduce memory usage per layer, and **cross-layer merging**, which targets redundancy across layers to eliminate unnecessary duplication. Both approaches offer complementary advantages, providing flexibility to balance memory savings and model performance degradation. The summary of the KV cache merging is listed in Tab. 4.

### 4.3.1 Intra-layer Merging

As the input sequence length increases, the number of Keys and Values grows, leading to higher computational costs for the attention process. To address this, CCM Kim et al. (2024a), LoMA Wang & Xiao (2024), DMC Nawrot et al. (2024) propose to learn a compression module to compress KV of tokens.

Specifically, CCM Kim et al. (2024a) inserts a special indicator token, *[COMP]*, into the input sequence and compresses the accumulating past attention key/value (KV) pairs in each layer between these indicators into a compact memory space. This compression leverages techniques inspired by the Compressive Transformer Rae et al. (2020) and Gisting Mu et al. (2023). Instead of computing attention across all tokens, CCM Kim et al. (2024a) computes attention scores for each new token by referencing the merged token. Similarly, LoMA Wang & Xiao (2024) inserts a special token into the input sequence to determine which consecutive tokens should be compressed. LoMA Wang & Xiao (2024) performs compression using bidirectional attention, repetition zone supervision, and carefully designed attention masks and loss functions. DMC Nawrot et al. (2024) learns a variable to decide whether to append new KV pairs to the cache when necessary or to merge them into existing KV representations using a weighted average.

Note that CCM Kim et al. (2024a), LoMA Wang & Xiao (2024), and DMC Nawrot et al. (2024) require supervised learning to learn a compression module.

Instead, CaM Zhang et al. (2024g), KVMerger Wang et al. (2024c), and D2O Wan et al. (2024a) are training-free, which rely on observations and directly propose rule-based or heuristic-based merging strategies. Specifically, they separate the Keys and Values of tokens in each layer into important (retrained) and unimportant (evicted) tokens. They then keep potentially useful unimportant tokens by merging their Keys and Values with retained important tokens, ensuring that no valuable information is lost. Particularly, D2O Wan et al. (2024a) merges merges the Key (or Value) of a evicted token with one retained token based on cosine similarity. Similar to D2O based on cosine similarity, AIM Zhong et al. (2024b) and Look-M Wan et al. (2024b) merges Keys (resp. Values) of multiple tokens into one. CaM Zhang et al. (2024g) merges the Keys (or Values) of multiple evicted tokens with retained tokens based on attention scores to get the final merged results. Also, KVMerger Wang et al. (2024c) first identifies the merge token sets by clustering consecutive tokens with high cosine similarity, ensuring that only adjacent tokens with strong contextual relevance are grouped together. Then, KVMerger merges the tokens in each merge set into the pivotal token (chosen based on the highest attention score) using Gaussian kernel weights, where closer tokens contribute more to the merged state.

Instead of merging the KV of multiple tokens into one, CHAI Agarwal et al. (2024) observes that heads in multi-head attention often produce highly correlated attention scores for tokens, particularly in the later layers of LLMs. To exploit this redundancy, CHAI Agarwal et al. (2024) clusters attention heads within each layer that produce similar outputs and computes attention for only a single representative head in each cluster. Specifically, within each cluster, CHAI Agarwal et al. (2024) selects one representative head to perform the attention computation, and the computed attention scores are shared across all heads in the cluster.

### 4.3.2 Cross-layer Merging

MiniCache Liu et al. (2024a) observes that KV caches in middle-to-deep layers exhibit high angular similarity, making them ideal for merging. To achieve this, MiniCache Liu et al. (2024a) merges the Key (and Value) pairs of each token from adjacent similar layers into a single shared representation. Specifically, MiniCache Liu et al. (2024a) decomposes KV vectors into magnitude and direction components, storing only the shared directional vectors, token magnitudes, and unmergeable tokens to maximize memory efficiency. Differently, KVSharer Yang et al. (2024d) observes a counterintuitive phenomenon: when the KV caches of two layers differ significantly, sharing one layer's KV cache with another during inference does not cause significant performance degradation. Based on this observation, KVSharer Yang et al. (2024d) computes the Euclidean distance between the KV caches of all layer pairs, ranks the pairs by dissimilarity, and prioritizes the most dissimilar layers for sharing. Since KVSharer Yang et al. (2024d) can share the KV cache of one layer to multiple other layers, the stored KV cache is eliminated significantly.

### 4.3.3 Summary and Future Directions

KV cache merging represents a transformative approach to optimizing memory utilization in LLMs by consolidating or compressing KV caches while maintaining high model accuracy. However, there are several key directions and challenges for future exploration in this domain. Firstly, current KV cache merging methods are typically designed to work across a wide range of tasks, but fine-tuning merging strategies for specific tasks or domains could further enhance efficiency. For example, certain tasks may tolerate more aggressive merging due to inherent redundancy in their attention patterns, while others may require more conservative approaches to preserve accuracy. Adaptive merging mechanisms that adjust compression levels on-the-fly based on task difficulty, sequence length, or available hardware resources are an exciting avenue for future work. Secondly, sparse attention mechanisms, which already reduce the computational complexity of attention by operating on subsets of tokens, could be combined with KV cache merging to achieve even greater efficiency. Exploring how merging complements sparsity-based approaches, such as block-sparse or low-rank attention, could lead to novel hybrid solutions. Thirdly, while empirical results show that merging does not significantly degrade performance, providing theoretical guarantees about the preservation of critical information could enhance the reliability of these methods. Future work might focus

Table 5: The summary of existing mixed-precision quantization models.

| Model | Keys | Values | Important Tokens | | | Outlier storing | Channel Reorder |
|---|---|---|---|---|---|---|---|
| | | | Intial | Middle | Recent | | |
| **KVQuant**<br>Hooper et al. (2024b) | Channel, Pre-RoPE | Per-Token | ✓ | | | ✓ | |
| **KIVI**<br>Liu et al. (2024h) | Channel | Per-Token | | | ✓ | | |
| **SKVQ**<br>Duanmu et al. (2024) | Dynamic outlier-aware | | ✓ | | ✓ | | ✓ |
| **WKVQuant**<br>Yue et al. (2024) | Learnable shifting | | | | ✓ | | |
| **QAQ**<br>Dong et al. (2024b) | Adaptive quantization bits | | ✓ | ✓ | ✓ | ✓ | |
| **MiKV**<br>Yang et al. (2024c) | Dynamic outlier-aware | | ✓ | ✓ | ✓ | | |
| **GEAR**<br>Kang et al. (2024) | Dynamic outlier-aware | | | | ✓ | ✓ | |
| **ZIPVL**<br>He et al. (2024a) | Conventional | | ✓ | ✓ | ✓ | | |
| **CacheGen**<br>Liu et al. (2024f) | Layer-wise, token-locality | | | | | | |
| **Atom**<br>Zhao et al. (2024b) | Group-based | | | | | ✓ | ✓ |

on quantifying the relationship between merging strategies, token importance, and attention accuracy to provide more formal guarantees.

## 4.4 KV Cache Quantization

Quantization technique Lin et al. (2016); Wu et al. (2020); Kwasniewska et al. (2019); Zhou et al. (2018); Jiang & Agrawal (2018) has been widely used to accelerate machine learning models from different aspects, such model parameter quantization Frantar et al. (2022); Dettmers et al. (2024); Bondarenko et al. (2023); Cheng et al. (2017) and data feature quantization Zhou et al. (2023a); Jegou et al. (2010). Similarly, Key-Value (KV) cache quantization is emerging as a highly promising solution to address the memory and computational bottlenecks in LLMs. During autoregressive decoding, LLMs generate key-value pairs for every attention layer across all tokens in the sequence. If we store all KV pairs in the memory with full precision, this cache grows exponentially with longer sequences, increasing the memory and bandwidth requirements significantly. Quantization reduces the precision of numerical representations (e.g., from FP32 to INT8 or INT4), drastically compressing the size of the KV cache. This compression can achieve up to 4x or more memory savings, making it feasible for LLMs to operate on resource-constrained devices like GPUs with limited memory or edge devices.

However, the presence of outliers in Keys and Values poses a significant challenge for low-bit quantization, as these extreme values can lead to substantial performance degradation when compressed into reduced bit representations Dettmers et al. (2022); Bondarenko et al. (2021); Wei et al. (2022). Based on the techniques used, existing KV cache quantization approaches can be classified into three main categories: **Fixed-precision quantization**, where all Keys and Values are quantized to the same bit-width; **Mixed-precision quantization**, which assigns higher precision to critical parts of the cache while using lower precision for less important components; and **Outlier redistribution**, which redistributes or smooths the outliers in Keys and Values to improve quantization quality. These methods collectively enable efficient KV cache compression while mitigating the performance degradation typically associated with low-bit quantization.

### 4.4.1 Fixed-precision Quantization

Fixed-precision quantization proposes quantizing different Keys (different Values) of tokens to the same bit-width. ZeroQuant Yao et al. (2022) propose per-token quantization for Keys and Values. As shown in Fig. 4,

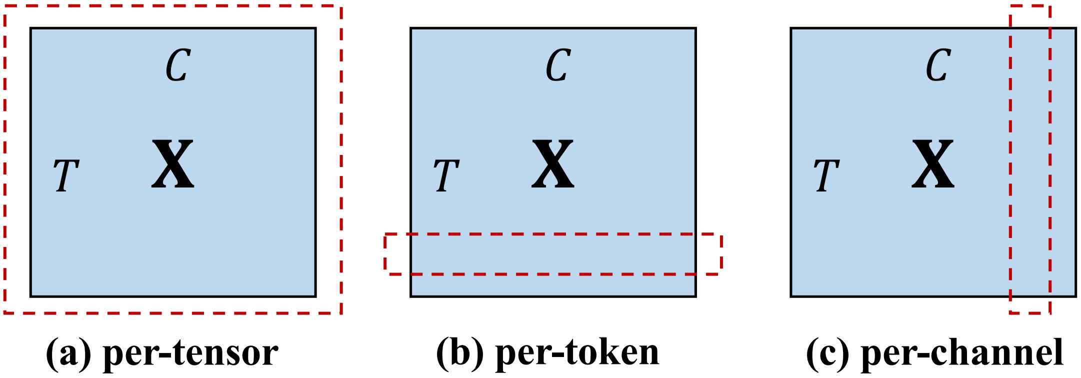

Figure 4: Three types of quantization. Then matrix $\mathbf{X} \in \mathbb{R}^{T\times C}$, where $T$ is the number of tokens and $C$ is the feature dimension.

the per-token quantization approach quantizes tokens individually. Particularly, ZeroQuant Yao et al. (2022) dynamically computes the min-max range for each token during inference. This ensures that each token is quantized based on its unique range, significantly reducing quantization error. Also FlexGen Sheng et al. (2023) and QJL Zandieh et al. (2024) directly perform per-token quantization for Keys and Values, where the scaling factor and zero-point are shared among all elements within the same token. PQCache Zhang et al. (2024b) uses product quantization approaches Jegou et al. (2010); Matsui et al. (2018) to compress KV pairs. However, uniform quantization approaches, which use a fixed bit-width for keys and values across all tokens, can often be suboptimal. It is because they ignore the varying importance of tokens Zhang et al. (2024f) and account for the outlier patterns in Keys and Values Dong et al. (2024b); Hooper et al. (2024b).

#### 4.4.2 Mixed-precision quantization

Unlike fixed-precision quantization, where all Keys or Values are quantized to the same bit-width (e.g., 4-bit or 8-bit), mixed-precision quantization assigns higher or full precision to Keys and Values of critical tokens and parts while using lower precision for less critical parts. The summary of KV mixed-precision quantization is listed in Tab. 5. KVQuant Hooper et al. (2024b) proposes several strategies to quantize Keys and Values smoothly based on observations. Firstly, KVQuant observes that the key values exhibit outliers in specific channels prior to applying Rotary Positional Embedding (RoPE). However, after applying RoPE, the magnitudes of these outlier channels become less consistent, creating a unique challenge for low-precision quantization. Thus, KVQuant Hooper et al. (2024b) proposes to quantize the Keys per channel before applying the RoPE operations and to quantize the Values per token. Secondly, KVQuant Hooper et al. (2024b) observes that KV cache activations contain outliers that skew the quantization range. To address this, they isolate outliers per vector (e.g., per-channel or per-token), store them in a sparse format, and quantize the remaining values to a narrower range. Thirdly, LLMs disproportionately allocate high attention scores to the first token (i.e., attention sink), and quantizing the first token will damage the performance of LLMs. Thus, KVQuant Hooper et al. (2024b) retains the first token in full precision (FP16) while quantizing the rest of the sequence, which is also used by IntactKV Liu et al. (2024e) and SKVQ Duanmu et al. (2024). Similar to KVQuant Hooper et al. (2024b), KIVI Liu et al. (2024h) quantizes the Key cache per-channel, as certain channels exhibit large outliers, and the Value cache per-token, as there are no significant outlier patterns in the Value cache. Additionally, KIVI Liu et al. (2024h) retains the most recent Keys and Values in full precision, while quantizing older KVs. This approach is based on the observation that the most recent KVs are critical for generating subsequent tokens.

Similar to KIVI Liu et al. (2024h), WKVQuant Yue et al. (2024) temporarily retains the most recent Keys and Values in full precision, while quantizing only the past KV cache. This approach helps preserve precision during computation. Additionally, WKVQuant Yue et al. (2024) introduces a two-dimensional quantization strategy, which optimizes parameter matrix to align the values in the KV cache into a smoother and more uniform range, significantly improving quantization quality. GEAR Kang et al. (2024), MiKV Yang et al.

Table 6: The summary of outlier redistribution models in Sec. 4.4.3.

| Model | Operation | Formula | Learn | Remarks |
|---|---|---|---|---|
| **MassiveAct**<br>Sun et al. (2024b) | Add virtual tokens | $\text{softmax}\left(\frac{\mathbf{Q}\begin{bmatrix}\mathbf{K}^T, & \mathbf{k}'\end{bmatrix}}{\sqrt{d}}\right)\begin{bmatrix}\mathbf{V} \\ \mathbf{v}'^T\end{bmatrix}$ | ✓ | Learnable $\mathbf{k}'$, $\mathbf{v}'$ |
| **QuaRot**<br>Ashkboos et al. (2024) | Hadamard rotation | $\mathbf{X}\mathbf{W}^\top = (\mathbf{X}\mathbf{H})(\mathbf{H}^\top\mathbf{W}^\top)$ | × | $\mathbf{H}^\top\mathbf{H} = \mathbf{I}$ |
| **Qserve**<br>Lin et al. (2024d) | Hadamard rotation | $\mathbf{X}\mathbf{W}^\top = (\mathbf{X}\mathbf{H})(\mathbf{H}^\top\mathbf{W}^\top)$ | × | $\mathbf{H}^\top\mathbf{H} = \mathbf{I}$ |
| **Q-INT4**<br>Xi et al. (2023) | Hadamard rotation | $\mathbf{X}\mathbf{W}^\top = (\mathbf{X}\mathbf{H})(\mathbf{H}^\top\mathbf{W}^\top)$ | × | $\mathbf{H}^\top\mathbf{H} = \mathbf{I}$ |
| **SmoothQuant**<br>Xiao et al. (2023) | Scaling | $(\mathbf{X}\,\text{diag}(\mathbf{s})^{-1}) \cdot (\text{diag}(\mathbf{s})\mathbf{W}^\top)$ | × | $\mathbf{s} \in \mathbb{R}^{c_i}$ |
| **QS+**<br>Wei et al. (2023) | Scaling, Shifting | $((\mathbf{X} - \mathbf{z})\,\text{diag}(\mathbf{s})^{-1} \cdot \text{diag}(\mathbf{s}) + \mathbf{z})\mathbf{W}^\top$ | × | $\mathbf{s} \in \mathbb{R}^{c_i}$ |
| **AWQ**<br>Lin et al. (2024c) | Scaling | $\arg\min_{\mathbf{s}} \left\lVert \mathbf{X}\mathbf{W}^\top - \mathbf{X}\,\text{diag}(\mathbf{s})^{-1})Q(\text{diag}(\mathbf{s})\mathbf{W}^\top) \right\rVert$ | ✓ | Quantization $Q(\cdot)$ |
| **OmniQuant**<br>Shao et al. (2023) | Scaling, Shifting | $Q_a\left(\frac{\mathbf{X}-\boldsymbol{\delta}}{\mathbf{s}}\right) Q_w\left(\mathbf{s} \odot \mathbf{W}^\top\right) + \mathbf{B} + \boldsymbol{\delta}\mathbf{W}^\top$ | ✓ | Learnable $Q_a(\cdot)$, $Q_w(\cdot)$ |
| **DuQuant**<br>Lin et al. (2024b) | Rotation, Permutation | $[(\mathbf{X} \cdot \boldsymbol{\Lambda})\hat{\mathbf{R}}_{(1)} \cdot \mathbf{P} \cdot \hat{\mathbf{R}}_{(2)}] \cdot [\hat{\mathbf{R}}_{(2)}^\top \cdot \mathbf{P}^\top \cdot \hat{\mathbf{R}}_{(1)}^\top(\boldsymbol{\Lambda}^{-1} \cdot \mathbf{W}^\top)]$ | × | Matrices $\mathbf{P}$, $\mathbf{R}$ |
| **AffineQuant**<br>Ma et al. (2024b) | Affine transform | $\arg\min_{\mathbf{P}} \left\lVert \mathbf{X}\mathbf{W}^\top - \mathbf{X}\mathbf{P}^{-1}Q(\mathbf{P}\mathbf{W}^\top) \right\rVert_F^2$ | ✓ | Quantization $Q(\cdot)$ |
| **FlatQuant**<br>Sun et al. (2024d) | Affine transform | $\text{AffineQuant} + \mathbf{P} = \mathbf{P}_1 \otimes \mathbf{P}_2$ | ✓ | Decomposition |

(2024c), ZipCache He et al. (2024b) and ZIPVL He et al. (2024a) quantize the KV cache based on the importance of each to achieve efficient and effective compression. First, GEAR Kang et al. (2024) applies quantization to compress the majority of less important entries (e.g., 98%) to ultra-low precision, significantly reducing memory usage. Next, GEAR Kang et al. (2024) employs a low-rank matrix to approximate residual errors, capturing structured patterns in the data. Also, GEAR Kang et al. (2024) uses a sparse matrix stores outliers, correcting individual errors caused by these values. MiKV Yang et al. (2024c) is a mixed-precision KV cache quantization approach. Based on the importance of each token, measured using existing methods like H2O Zhang et al. (2023a) and SnapKV Li et al. (2024b), MiKV Yang et al. (2024c) stores less important KV pairs in low precision while retaining the most important KV pairs in high precision. Instead of approximating the importance weight of each token, ZipCache He et al. (2024b) accurately computes the importance of each token. Instead of computing importance score, PrefixQuant Chen et al. (2024c) observes that token-wise outliers frequently occur at fixed positions (e.g., initial tokens) or low-semantic-value tokens (e.g., ".", "\n"). Based on this observation, PrefixQuant Chen et al. (2024c) identifies high-frequency outlier tokens in LLMs offline and prefixes them in the KV cache, effectively eliminating token-wise outliers. Similarly, MiniKV Sharma et al. (2024) observes that important tokens can be identified before generation and remain consistent throughout the generation process, retaining these important tokens in high precision.

QAQ Dong et al. (2024b) proposes a quality adaptive quantization approach to dynamically determine the suitable quantization bit for each token, based on its importance and sensitivity, while handling outliers and exceptions to maintain model performance. SKVQ Duanmu et al. (2024) introduces the clipped dynamic quantization with channel reorder. First, SKVQ Duanmu et al. (2024) uses a transformation-invariant permutation to group similar channels based on their statistical characteristics and applies clipped dynamic quantization to mitigate the outlier problem. Second, SKVQ Duanmu et al. (2024) maintains high precision for the initial tokens and the most recent tokens while quantizing older tokens. Consequently, SKVQ Duanmu et al. (2024) effectively reduces quantization errors and improves the accuracy of the quantized model. CacheGen Liu et al. (2024f) and AsymKV Tao et al. (2024) use layer-wise asymmetric quantization, assigning higher-bit precision to key matrices in sensitive early layers and lower-bit precision to less sensitive layers, balancing memory efficiency and performance. Particularly, CacheGen Liu et al. (2024f) also exploits token-wise locality by encoding deltas (differences) between KV tensors of nearby tokens instead of raw values. Atom Zhao et al. (2024b) identifies and separates outlier channels, reordering the matrix to group these outlier channels at the end, thereby ensuring regular memory access patterns for improved hardware utilization. Then, Atom Zhao et al. (2024b) quantizes outliers with higher precision, while normal channels are quantized to INT4 for maximum efficiency. In particular, Atom Zhao et al. (2024b) applies fine-grained group quantization by dividing matrices into smaller subgroups (e.g., 128 elements per group) and performing quantization independently within each group.

#### 4.4.3 Outlier Redistribution

As previously mentioned, outliers in the Keys and Values present significant challenges for their quantization. Recent research has proposed two main approaches to address this issue: redistributing the outliers into newly appended virtual tokens or applying equivalent transformation functions to smooth the Keys and Values for improved quantization accuracy. The summary of existing outlier redistribution models are listed in Table. 6.

Specifically, MassiveActivation Sun et al. (2024b) highlights the phenomenon of massive activations in large language models (LLMs), where a small subset of activations is exponentially larger than the rest. To address this, MassiveActivation Sun et al. (2024b) proposes appending a virtual token to the inputs, allowing LLMs to encapsulate the massive outliers within these learned keys and values for each head. Then, we introduce the equivalent transformation function-based approaches. Firstly, QuaRot Ashkboos et al. (2024), Qserve Lin et al. (2024d), and Q-INT4 Xi et al. (2023) redistributes outlier values across all channels by Hadamard rotation, successfully lowering the maximum value of outlier tokens. The Hadamard rotation of activations can be incorporated into the preceding linear layer, thereby redistributing the outliers of Keys and Values into the parameters. Despite this improvement, outlier tokens still exhibit magnitudes hundreds of times greater than normal tokens, causing notable performance issues when using shared quantization scales across tokens Chen et al. (2024c). Expanding on this idea, SpinQuant Liu et al. (2024g) proposes training an orthogonal matrix instead of relying on a random Hadamard matrix to achieve better performance. Similarly, DuQuant Lin et al. (2024b) employs channel permutation to evenly distribute outliers across blocks and utilizes block rotation to further smooth outliers.

SmoothQuant Xiao et al. (2023) leverages a key observation that different tokens show similar patterns of variation across their channels. Based on this insight, it strategically shifts the quantization complexity from activations to weights through an offline process. Specifically, SmoothQuant Xiao et al. (2023) introduces a mathematically equivalent per-channel scaling transformation: $\mathbf{Y} = (\mathbf{X}\text{diag}(\mathbf{s})^{-1}) \cdot (\text{diag}(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$ where $\mathbf{X}$ represents Keys or Values, and the smoothing factor $\mathbf{s} \in \mathbb{R}^{C_i}$ is used to scale $\mathbf{X}$. This transformation achieves two key benefits: it smooths the distribution of Keys and Values to facilitate easier quantization, and it allows the smoothing factors to be efficiently incorporated into the parameters of previous layers during offline processing. In particular, the smooth factor $\mathbf{s}$ is dynamically decided on based on inputs. Similarly, The OS+ Wei et al. (2023) introduces channel-wise shifting to eliminate outlier asymmetry and channel-wise scaling to reduce outlier concentration. These operations are seamlessly migrated to subsequent layers, maintaining equivalence with the floating-point model while improving quantization performance.

Instead of using handcrafted transformations Lin et al. (2024b); Wei et al. (2023); Xiao et al. (2023) to shift the quantization difficulty from activations to weights, AffineQuant Ma et al. (2024b) uses an affine transformation matrix that combines both scaling and rotation transformations. This allows it to optimize weight distributions more effectively, aligning them better with the quantization function and reducing quantization errors. The affine transformation matrix provides richer flexibility compared to SmoothQuant's scalar-based scaling, enabling finer adjustments to the weight and activation distributions. Based on AffineQuant Ma et al. (2024b), FlatQuant Sun et al. (2024d) introduces a fast and learnable affine transformations to enhance the flatness of weights and activations, which decomposes transformations into smaller matrices to reduce memory and computational costs. Similarly, AWQ Lin et al. (2024c) and OmniQuant Shao et al. (2023) proposes differentiable and learnable equivalent transformations, which optimize the equivalent parameters (e.g., channel-wise scaling and shifting) in an end-to-end manner using gradient descent.

#### 4.4.4 Summary and Future Directions

KV cache quantization is a crucial technique for reducing memory and computational overhead in large language models (LLMs) during autoregressive decoding. While significant progress has been made, this field remains dynamic and rapidly evolving, with several promising directions for future research. Firstly, one promising avenue is the development of real-time adaptive quantization methods. These techniques could dynamically adjust quantization levels during inference based on real-time metrics such as token importance, outlier presence, or sequence length. Such an approach could significantly enhance efficiency while maintaining performance, especially for processing long sequences with varying levels of complexity. Secondly, another important direction is extending KV cache quantization to multi-modal and multi-task models. Multi-modal models, which process inputs from diverse domains such as text, vision, and audio, and multi-task scenarios often exhibit highly diverse attention patterns and memory demands. This necessitates the design of more advanced and tailored quantization strategies to balance efficiency and accuracy in these increasingly complex settings.

Thirdly, hybrid quantization techniques also hold significant potential. By combining fixed-precision, mixed-precision, and outlier redistribution methods, researchers could develop more versatile and efficient quantization frameworks. For instance, integrating mixed-precision allocation schemes with outlier smoothing transformations could optimize both memory usage and performance, offering a flexible approach adaptable to a variety of tasks and models. finally, addressing the challenge of outliers remains a critical area of focus. Outliers can have a disproportionate impact on quantization efficiency and model performance. Future research could explore advanced outlier detection mechanisms or innovative encoding techniques to mitigate their effects. Improved handling of outliers could further enhance the effectiveness of quantization methods, enabling more robust and memory-efficient implementations.

### 4.5 KV Cache Low-rank Decomposition

Existing studies have demonstrated that the majority of information within KV caches can be captured by a small subset of their singular values or low-rank components, making low-rank decomposition a powerful tool for compression. By leveraging this property, KV cache low-rank decomposition techniques aim to reduce memory requirements while preserving the essential information required for accurate attention computations. Low-rank decomposition strategies can be classified into three main approaches: **Singular Value Decomposition (SVD)**, which exploits the low-rank structure of KV matrices to retain the most critical singular values; **Tensor Decomposition**, which factorizes KV matrices into smaller components for minimal redundancy; and **Learned Low-rank Approximation**, which incorporates adaptive mechanisms to optimize compression based on learned representations. Each method provides a unique balance of computational efficiency and accuracy retention, enabling scalable and memory-efficient LLM inference.

#### 4.5.1 Singular Value Decomposition

Firstly, ECKVH Yu et al. (2024), EigenAttention Saxena et al. (2024), and ZDC Zhang & Shen (2024) shows that KV caches have a low-rank property, where a small number of top singular values retain most of the information. Using Singular Value Decomposition (SVD), the method compresses KV caches by grouping heads, applying SVD, and retaining top singular values, effectively reducing the number of KV heads with minimal error. Also, ZDC Zhang & Shen (2024) uses an adaptive hybrid compression ratio mechanism to assign higher compression to unimportant tokens in shallower layers while preserving more important tokens in deeper layers, leveraging the similarity of token characteristics in adjacent layers. Secondly, rather than decomposing KV pairs, LoRC Zhang et al. (2024d) employs a low-rank approximation of KV weight matrices and adopts a progressive compression strategy to efficiently compress KV caches without requiring model retraining. Specifically, LoRC Zhang et al. (2024d) uses SVD to compress the Keys and Values parameter matrices (i.e., $\mathbf{W}_i^k$ and $\mathbf{W}_i^v$) as $\mathbf{W}_i^k = \mathbf{U}_i^k \mathbf{\Sigma}_i^k {\mathbf{P}_i^k}^\top$ and $\mathbf{W}_i^v = \mathbf{U}_i^v \mathbf{\Sigma}_i^v {\mathbf{P}_i^v}^\top$. Also, compression is applied conservatively in shallower layers to minimize error amplification and more aggressively in deeper layers. Then, instead of storing $\mathbf{K}^i = \mathbf{X}_i \mathbf{W}_i^k$ and $\mathbf{V}^i = \mathbf{X}_i \mathbf{W}_i^v$, it only stores $\hat{\mathbf{K}}^i = \mathbf{X}_i \mathbf{U}_i^k$ and $\hat{\mathbf{V}}^i = \mathbf{X}_i \mathbf{U}_i^v$, along with $\mathbf{\Sigma}_i^k {\mathbf{P}_i^k}^\top$ and $\mathbf{\Sigma}_i^v {\mathbf{P}_i^v}^\top$. Also, ShadowKV Sun et al. (2024a) performs SVD decomposition directly on pre-RoPE keys to reduce the dimensionality of the key representations. Palu Chang et al. (2024) applies SVD to compress both Keys and Values.

### 4.5.2 Tensor Decomposition

Tensor decomposition Kuleshov et al. (2015); Zhou et al. (2017); Haeffele & Vidal (2015) is a widely used algorithm for factorizing a matrix into a sequential product of local tensors, such as Matrix Product Operator (MPO) Liu et al. (2021) and turker decomposition Malik & Becker (2018) . Taking Matrix Product Operator (MPO) Liu et al. (2021) as an example, the decomposition of a matrix $\mathbf{W} \in \mathbb{R}^{I \times J}$ using MPO can be defined as:

$$\mathrm{TD}(\mathbf{W}) = \prod_{k=1}^{n} \mathcal{T}_{(k)}[d_{k-1}, i_k, j_k, d_k], \tag{12}$$

where $\mathcal{T}_{(k)}$ represents the local tensor of size $d_{k-1} \times i_k \times j_k \times d_k$, with $\prod_{k=1}^{n} i_k = I$ and $\prod_{k=1}^{n} j_k = J$. Here, $n$ denotes the number of local tensors, collectively referred to as the decomposed tensors. As shown in Eaquation equation 12, MPO-based tensor decomposition is well-suited for KV cache compression as it reduces the memory footprint by factorizing large key and value matrices into smaller local tensors, enabling efficient storage while preserving essential information. This approach minimizes redundancy and maintains the structural integrity required for accurate attention computations. DecoQuant Liu et al. (2024d) combines quantization with low-rank decomposition to effectively reduce quantization errors. Specifically, DecoQuant Liu et al. (2024d) leverages the Matrix Product Operator (MPO) to decompose matrices into smaller local tensors. The larger tensors, which contain most of the parameters, are quantized to low-bit precision, while the smaller tensors retain high precision to minimize overall quantization error.

### 4.5.3 Learned Low-rank Approximation

LESS Dong et al. (2024a) introduces a novel learned-kernel-based low-rank approximation approach to efficiently approximate the results of the softmax function. Specifically, LESS Dong et al. (2024a) replaces the softmax with a separable similarity metric, $\phi(\mathbf{q}_t)\psi(\mathbf{K}_t)^\top$, where $\phi$ and $\psi$ are row-wise functions. Here, $\mathbf{q}_t \in \mathbb{R}^{1\times D}$ represents the query, and $\mathbf{K}_t \in \mathbb{R}^{t\times D}$ represents the keys at step $t$. To elaborate, if $\phi$ and $\psi$ are such that: $a_t = \mathrm{softmax}\left(\frac{\mathbf{q}_t\mathbf{K}_t^\top}{\sqrt{D}}\right)\mathbf{V}_t \approx \frac{\phi(\mathbf{q}_t)\psi(\mathbf{K}_t)^\top\mathbf{V}_t}{\phi(\mathbf{q}_t)\psi(\mathbf{K}_t)^\top\mathbf{1}_{S\times 1}}$, then we only need to cache the hidden states $\mathbf{H}_t = \psi(\mathbf{K}_t)^\top\mathbf{V}_t \in \mathbb{R}^{R\times D}$ and the normalization factor $\mathbf{z}_t = \sum_{s=1}^{t}\psi([\mathbf{K}_t]_s) \in \mathbb{R}^{1\times R}$ for inference. Similarly, MatryoshkaKV Lin et al. (2024a) compresses KV caches along the feature dimension by leveraging trainable orthogonal projection matrices.

### 4.5.4 Summary and Future Directions

KV cache low-rank decomposition is a powerful technique for compressing KV caches in LLMs while maintaining the quality of attention computations. Current methods primarily rely on fixed low-rank approximations applied uniformly across all layers or tokens. However, future advancements could focus on dynamic rank adjustment, where the rank is tailored based on token importance, sequence length, or layer-specific properties, enabling a more optimal balance between memory efficiency and performance. Additionally, real-time or streaming applications present a promising avenue for exploration. Since KV caches grow dynamically during inference, lightweight and incremental decomposition methods that can adapt efficiently to expanding sequences will be critical for supporting such scenarios without compromising latency or accuracy.

## 5 Model-level Optimization

In model-level optimization, new architectures or mechanisms are designed for transformers to allow more efficient reuse of KV cache. Typically, these methods require retraining or fine-tuning of the model to come into operation. Nevertheless, efficient transformation pipelines have also been proposed to allow for a fast deployment to new architectures. According to where and how the refinement was made to the models, we separate related works to the grouping and sharing mechanisms within or cross layers (Sec. 5.1), implementing architecture modification or augmentation (Sec. 5.2), and incorporating non-transformer architectures for optimization (Sec. 5.3). The taxonomy of the model-level optimization is shown in Fig. 5.

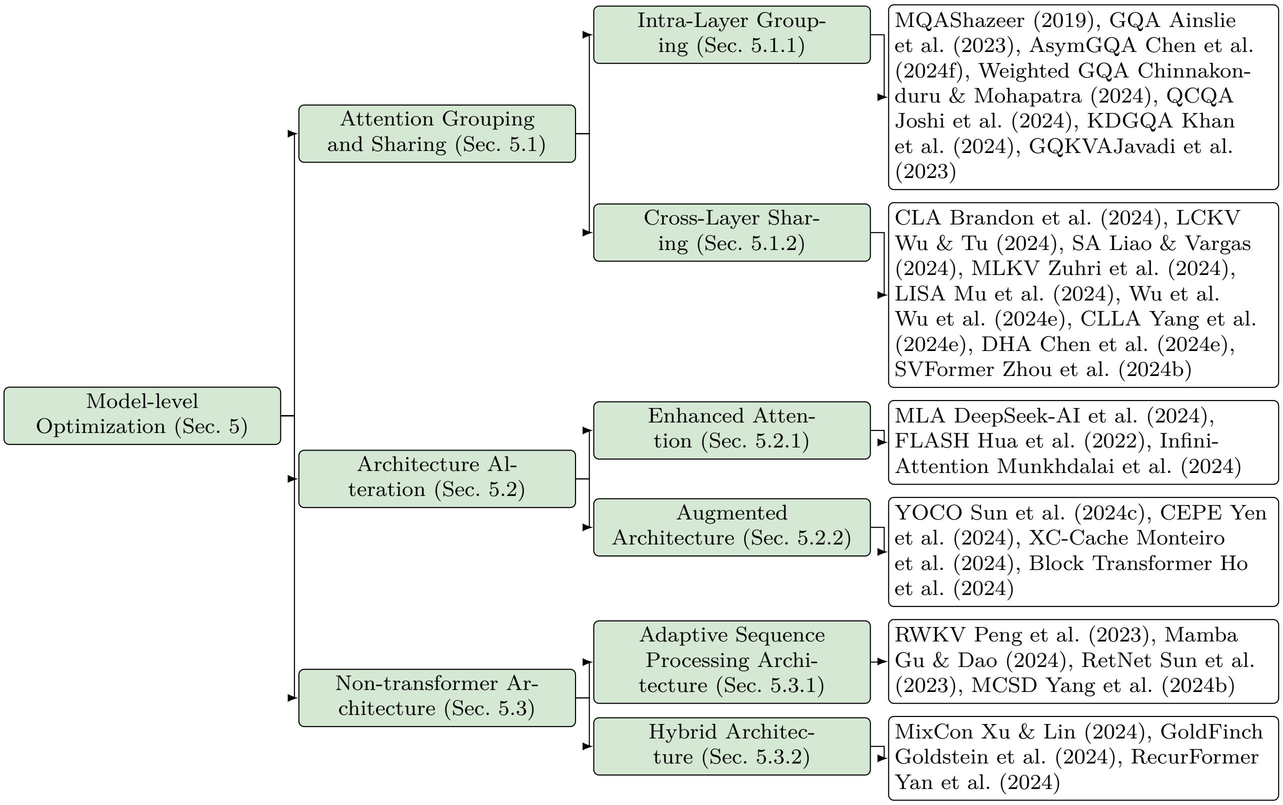
Model-level Optimization (Sec. 5)
Attention Grouping and Sharing (Sec. 5.1)
Intra-Layer Grouping (Sec. 5.1.1)
MQAShazeer (2019), GQA Ainslie et al. (2023), AsymGQA Chen et al. (2024f), Weighted GQA Chinnakonduru & Mohapatra (2024), QCQA Joshi et al. (2024), KDGQA Khan et al. (2024), GQKVAJavadi et al. (2023)
Cross-Layer Sharing (Sec. 5.1.2)
CLA Brandon et al. (2024), LCKV Wu & Tu (2024), SA Liao & Vargas (2024), MLKV Zuhri et al. (2024), LISA Mu et al. (2024), Wu et al. Wu et al. (2024e), CLLA Yang et al. (2024e), DHA Chen et al. (2024e), SVFormer Zhou et al. (2024b)
Architecture Alteration (Sec. 5.2)
Enhanced Attention (Sec. 5.2.1)
MLA DeepSeek-AI et al. (2024), FLASH Hua et al. (2022), Infini-Attention Munkhdalai et al. (2024)
Augmented Architecture (Sec. 5.2.2)
YOCO Sun et al. (2024c), CEPE Yen et al. (2024), XC-Cache Monteiro et al. (2024), Block Transformer Ho et al. (2024)
Non-transformer Architecture (Sec. 5.3)
Adaptive Sequence Processing Architecture (Sec. 5.3.1)
RWKV Peng et al. (2023), Mamba Gu & Dao (2024), RetNet Sun et al. (2023), MCSD Yang et al. (2024b)
Hybrid Architecture (Sec. 5.3.2)
MixCon Xu & Lin (2024), GoldFinch Goldstein et al. (2024), RecurFormer Yan et al. (2024)

Figure 5: Taxonomy of the model based KV optimization for Large Language Models.

### 5.1 Attention Grouping and Sharing

This section explores attention grouping and sharing methods as effective strategies for optimizing key-value (KV) management. We categorize the approaches into two distinct subtypes: intra-layer grouping (Sec. 5.1.1) that focuses on grouping query, key, and value heads within individual layers to reduce redundancy and improve efficiency, and cross-layer sharing 5.1.2 that shares key, value, or attention components across layers to improve information reuse and reduce KV cache requirements. The summary of attention grouping and sharing is listed in Tab. 7.

#### 5.1.1 Intra-layer Grouping

Shazeer first introduced Multi-Query Attention (MQA) Shazeer (2019) that modified the traditional multi-head attention mechanism. In MQA, all attention heads in a transformer block share a single key and value. This simple strategy can greatly accelerate the decoding procedure. The experiments of the author show that MQA would gain much efficiency with only minor quality degradation incurring.

MQA is a radical strategy that would cause not just quality degradation, but also training instability. GQA (Grouped Query Attention) Ainslie et al. (2023) introduced a trade-off solution by dividing the query heads into multiple groups, while each group shares its own keys and values. In addition, an uptraining process is proposed to efficiently convert existing MHA models to GQA configurations by mean-pooling the key and value heads associated with each group. Empirical evaluations demonstrated that GQA models achieve performance close to the original MHA models while offering inference time comparable to MQA.

There were several extensions based on GQA. AsymGQA Chen et al. (2024f) extends GQA by proposing an activation-informed merging strategy. Instead of grouping the heads by uniform clustering, AsymGQA dynamically determines the grouping of quries based on their activations similarities during training and constructs an asymmetric group results, which leads to better optimization and generalization. Weighted

Table 7: The summary of Model-based Attention Grouping and Sharing approaches.

| Method | Applied Location | | Intra-layer Grouped Component | Cross-layer Shared Component | Retraining Required |
|---|---|---|---|---|---|
| | Intra-layer | Cross-layer | | | |
| **MQA**<br>Shazeer (2019) | ✓ | | K, V | - | ✓ |
| **GQA**<br>Ainslie et al. (2023) | ✓ | | K, V | - | Uptrain |
| **AsymGQA**<br>Chen et al. (2024f) | ✓ | | K, V | - | Finetune |
| **Weighted GQA**<br>Chinnakonduru & Mohapatra (2024) | ✓ | | K, V | - | Uptrain & Finetune |
| **QCQA**<br>Joshi et al. (2024) | ✓ | | K, V | - | ✓ |
| **KDGQA**<br>Khan et al. (2024) | ✓ | | K, V | - | ✓ |
| **GQKVA**<br>Javadi et al. (2023) | ✓ | | Q, K, V | - | ✓ |
| **CLA**<br>Brandon et al. (2024) | ✓ | ✓ | K, V | K, V | ✓ |
| **LCKV**<br>Wu & Tu (2024) | | ✓ | - | K, V | ✓ |
| **SA**<br>Liao & Vargas (2024) | | ✓ | - | Attention Weight | ✓ |
| **MLKV**<br>Zuhri et al. (2024) | ✓ | ✓ | K, V | K, V | Uptrain |
| **LISA**<br>Mu et al. (2024) | | ✓ | | Q, K, V | Lightweight adaption |
| **Wu et al.**<br>Wu et al. (2024e) | | ✓ | - | Q, K, V | ✓ |
| **CLLA**<br>Yang et al. (2024e) | | ✓ | - | Q, K, V | ✓ |
| **DHA**<br>Chen et al. (2024e) | ✓ | ✓ | K, V | Q, K, V | Lightweight adaption |
| **SVFormer**<br>Zhou et al. (2024b) | | ✓ | - | V | ✓ |

GQA Chinnakonduru & Mohapatra (2024) introduces additional trainable weights to each key and value head, which can be seamlessly integrated into existing GQA models. By tuning weights during training, it improves the performance of the model without additional inference overhead. QCQA Joshi et al. (2024) utilizes an evolutionary algorithm to identify the optimal query head groupings for GQA, which is guided by a computationally efficient fitness function that leverages the weight-sharing error and the KV cache to evaluate text generation quality and memory capacity. KDGQA Khan et al. (2024) argues that many variances of GQA adopt a fixed grouping strategy, thus lacking dynamic adaptability to the evolving of key-value interactions during training. Their Dynamic Key-Driven GQA address these issues by allocating groups using key head norms adaptively during training, resulting in a flexible strategy to query head grouping and enhance the performance.

GQKVA Javadi et al. (2023) advances the grouping strategy and comes up with a generalized query, key and value grouping mechanism. It first introduces MKVA and GKVA, in which the key and value are grouped to share the same query. Based on this, GQKVA is proposed to separately group the query and key-value pairs. Typically, queries are partitioned into $g_q$ groups, and keys and values are partitioned into $g_{kv}$ groups, and each combination of query and key-value pairs would interact using dot product attention. This results in $g_q \times g_{kv}$ distinct outputs. It generalized different group strategy on query, key and value and preserves good computational efficiency and comparable performance as MHA.

#### 5.1.2 Cross-layer Sharing

Brandon et al. introduce Cross Layer Attention (CLA) Brandon et al. (2024) that extends the ideas of GQA and MQA by sharing the key and value heads between adjacent layers, further reduce the redundancy in the KV cache. This achieves an additional 2× KV cache size reduction compared to MQA, significantly improving memory efficiency without altering computational complexity.

LCKV Wu & Tu (2024) proposes only to compute and cache the key and value for a small subset of layers, even only the top layer, then let queries in bottom layers pair the saved keys and values for inference. This method not only drastically improves the inference speed and reduces memory consumption but is also orthogonal to existing memory-saving techniques, enabling straightforward integration for further optimization. While such a mechanism makes next token computation depend on top layer keys and values of previous tokens, which contradict to the parallel training of transformers, LCKV introduces an approximate training methods to support parallel training.

SA (Shared Attention) Liao & Vargas (2024) proposes reuse of computed attention weights across multiple layers, rather than recalculating them for each layer. Unlike other methods focusing on sharing key-value caches, SA leverages the isotropic tendencies of attention distributions observed in pre-trained LLMs to directly share attention weights, greatly reducing both computational overhead and memory usage.

MLKV (Multi-Layer Key-Value) Zuhri et al. (2024) introduces a simple KV head sharing mechanism across multiple transformer layers. MLKV uses the same single KV head as MQA within a layer, but it also shares this KV head with multiple layers. This extreme strategy reduces the cache size to almost 1% of normal GQA strategies, and experiments show that MLKV still has comparable performance.

LISA (Lightweight Substitute for Attention) Mu et al. (2024) makes a comprehensive analysis for the similarity of attention patterns across layers. Directly sharing attention weights across layers is ineffective because of the misalignment of the attention head and the sensitivity of shallow layers. LISA Mu et al. (2024) addresses challenges by incorporating tiny feed-forward networks to align attention heads between layers and using low-rank matrices to approximate variations in layer-wise attention weights. This achieves a 6× compression of query and key parameters while maintaining high accuracy and perplexity.

Wu et al. Wu et al. (2024e) introduce a unified framework that systematically analyzes and optimizes the cross-layer Key-Value cache sharing mechanism. They consolidate several existing methods, explore novel variants within a cohesive structure, and make thorough evaluations of these methods. The study finds that 2 times reduction to KV cache size can outperform standard transformers in throughput without substantial accuracy loss, while further reduction requires alternative design with additional training costs. With the analysis results, they offer insight into the choice of appropriate KV sharing methods based on the specific requirement or constraints.

CLLA (Cross-Layer Latent Attention) Yang et al. (2024e) introduces an integrated framework combining multiple strategies: attention head size and dimension reduction, cross-layer cache sharing, and KV cache quantization. By unifying these strategies, CLLA achieves extreme KV cache compression to less than 2% of the original model size while maintaining performance levels comparable with uncompressed models.

DHA (Decoupled Head Attention) Chen et al. (2024e) addresses redundancy in MHA and adaptively configures shared groups for key and value heads across layers, reducing KV cache requirements. Observing that clustering and fusing similar heads can reduce KV cache size without significant performance reduction, DHA designs a search, fusion, and continued pre-training framework that can progressively transform MHA checkpoints into DHA models through linear fusion of head parameters, preserving the pre-trained knowledge with small pre-training budget.

Observing that later layers in traditional transformers overly rely on narrow regions of attention, Zhou et al. Zhou et al. (2024b) introduce ResFormer that utilizes residual connections from the value embeddings of the first layer to all subsequent layers, effectively approximating cross-layer attention without incurring significant computational costs. They then propose a simplified variant SVFormer that shares a single value embedding across all layers, dramatically reducing the KV cache size by nearly half while maintaining

Table 8: The summary of Model-based Intra-layer approaches.

| Method | Alteration Type | | KV Cache Management | Retraining Requirement |
|---|---|---|---|---|
| | Enhanced Attention | Augmented Architecture | | |
| **MLA**<br>DeepSeek-AI et al. (2024) | ✓ | | Latent compression | ✓ |
| **FLASH**<br>Hua et al. (2022) | ✓ | | Linear approximation | ✓ |
| **Infini-Attention**<br>Munkhdalai et al. (2024) | ✓ | | Compressive cache | ✓ |
| **YOCO**<br>Sun et al. (2024c) | | ✓ | Single global KV cache | ✓ |
| **CEPE**<br>Yen et al. (2024) | | ✓ | Parallel encoding with cross-attn | Lightweight |
| **XC-Cache**<br>Monteiro et al. (2024) | | ✓ | Encoder cross-attention | ✓ |
| **Block Transformer**<br>Ho et al. (2024) | | ✓ | Hierarchical local KV | Lightweight |

competitive performance. The proposed architectures are flexible to incorporate with other KV-efficient strategies for additional memory savings.

### 5.1.3 Summary and Future Directions

This section highlights innovative strategies for optimizing memory and computational efficiency through intra-layer grouping and cross-layer sharing mechanisms. However, several avenues for improvement remain. First, maintaining performance while optimizing efficiency, especially for precision-sensitive tasks, requires further investigation. Methods that implement radical grouping and sharing mechanisms may compromise the model fidelity for tasks requiring high precision. Second, scalability across diverse model architectures and sizes is essential. Works such as DHA Chen et al. (2024e) and LISA Mu et al. (2024), which rely on specific architectural assumptions, may struggle to generalize to emerging LLMs or non-standard configurations. Third, the dynamics of attention across both time and layers are largely under-explored. Most existing methods rely on static or pre-determined grouping and sharing strategies, neglecting the temporal and contextual variations in attention patterns.

To address these challenges and unlock the full potential of attention optimization, future research should focus on the following aspects. First, developing universal frameworks for attention grouping and sharing that require minimal retraining to enhance adaptability and usability. Second, synergistic integration with other optimization techniques, such as quantization and pruning, has significant potential to achieve even greater efficiency gains. While some works like CLLA Yang et al. (2024e) have begun to address these opportunities, more exploration could be carried out to unlock new levels of efficiency. Third, more dynamic and temporal modeling could be leveraged to adaptively adjust grouping and sharing during runtime to better capture the contextual requirements of different tasks and sequences. Finally, a deeper understanding of the downstream impacts of these techniques on fine-tuning and transfer learning is crucial for their effective application in real-world scenarios.

## 5.2 Architecture Alteration

This section explores architectural modifications to optimize KV cache usage. We categorize these methods into two subsections: methods that refine the attention mechanism for KV cache efficiency (Sec. 5.2.1), and methods that introduce structural changes for better KV management (5.2.2). Many of these works build upon the broader landscape of efficient attention mechanisms (e.g., Linear Transformer Katharopoulos et al. (2020), Performer Choromanski et al. (2020), LinFormer Wang et al. (2020), etc.). Since our focus lies on

Table 9: The summary of Non-Transformer Architectures.

| Method | Key Mechanism | No Traditional KV Cache | KV Cache Compression |
|---|---|---|---|
| **RWKV**<br>Peng et al. (2023) | RNN-like with Transformer parallelism | ✓ | |
| **Mamba**<br>Gu & Dao (2024) | Selective state-space model | ✓ | |
| **RetNet**<br>Sun et al. (2023) | Retention mechanism | | ✓ |
| **MCSD**<br>Yang et al. (2024b) | Slope-decay fusion | ✓ | |
| **MixCon**<br>Xu & Lin (2024) | Transformer + Conba + MoE | ✓ | |
| **GoldFinch**<br>Goldstein et al. (2024) | RWKV + Modified Transformer | | ✓ |
| **RecurFormer**<br>Yan et al. (2024) | Mamba replacing some attention heads | | ✓ |

methods directly impacting KV cache handling, for a comprehensive overview of efficient attention mechanisms, we refer readers to dedicated surveys Zhou et al. (2024c). The summary of architecture alteration for KV reuse is listed in Tab. 8.

### 5.2.1 Enhanced Attention

DeepSeek-V2 DeepSeek-AI et al. (2024) introduced Multi-Head Latent Attention (MLA) that adopts a low-rank KV joint compression mechanism, replacing the full KV cache with compressed latent vectors. The model adopts trainable projection and expansion matrices to do the compression. This compression mechanism significantly reduces the memory requirement of the KV cache and allows the model to handle sequences up to 128K tokens.

FLASH Hua et al. (2022) incorporates the Gated Attention Unit (GAU) to replace the MHA mechanism in traditional transformers. GAU utilizes a single-head attention mechanism with gating functions that selectively modulates importance in information flow. FLASH employs a linear approximation method for attention computation through GAU module, which makes the model efficiently handle long contexts without the quadratic scaling of traditional self-attention, thus mitigating heavy KV cache issues.

Infini-Attention Munkhdalai et al. (2024) adopts representation compression to store long-term content. Furthermore, they introduce a hybrid attention mechanism of masked local attention and long-term linear attention. The masked local attention replaces the standard MHA to let the model only concentrate on local contexts, while the long-term linear attention utilizes compressed memory for far-reaching dependencies and uses linear attention for efficient aggregation. Thus, infini-attention combines both local fine-grained and long-range compressed states, allowing a seamless balance between long-term and short-term context modeling.

### 5.2.2 Augmented Architecture

YOCO Sun et al. (2024c) builds a decoder-decoder architecture composed of two modules: a self-decoder and a cross-decoder. The self-decoder efficiently encodes global key-value caches, while the cross-decoder reuses these caches via cross-attention. This design ensures that key-value pairs are only cached once, substantially reducing GPU memory usage while maintaining global attention capabilities. YOCO's computation flow also enables the prefilling to early exit, allowing faster prefill stages without altering the final output.

CEPE Yen et al. (2024) interleaves additional cross-attention layers between the self-attention and feed-forward layers in the decoder model. It employs a small encoder to process long inputs chunk-by-chunk to encoded representations as cross-attention layers' inputs. In this way, CEPE can prevent the needs for KV cache for every token and reduce computational cost by processing contexts in parallel. This also facilitates an existing LLMs to expand its contexts while preserving the scalability and generalizability.

XC-Cache Monteiro et al. (2024) also utilizes an encoder to interleave cross-attention layers within existing self-attention layers in pre-trained decoder-only models to prevent explicit prompt caching. The encoder processes the context and converts it into a compact set of key-value pairs that summarize the essential information. It also finds that pre-trained causal decoders can be used to replace an encoder for representations extraction, further reducing the training costs on additional encoder.

Block Transformer Ho et al. (2024) introduces a hierarchical global-to-local architecture by combining coarse-grained global attention and fine-grained local attention. In lower layers, tokens are grouped into fixed-size blocks, allowing global context modeling with reduced KV cache overhead. In upper layers, attention operates within individual blocks, enabling lightweight, detailed token decoding with a smaller local KV cache.

#### 5.2.3 Summary and Future Directions

This section explores research that introduces novel attention mechanisms or architectural modifications to improve KV cache management. Although these approaches demonstrate significant progress in enabling longer context windows and faster inference, several challenges remain. First, many methods, such as CEPE Yen et al. (2024) and XC-Cache Monteiro et al. (2024) demonstrate strong performance on retrieval-augmented tasks but may not generalize well across diverse workloads. This necessitates further research into task-adaptive KV cache optimization strategies that dynamically adjust caching behavior to optimize for different task demands. Secondly, integrating these novel mechanisms into existing pretrained models often requires extensive retraining, hindering their adoption in resource-constrained environments. Developing lightweight, modular approaches for retrofitting efficient KV caching into existing architectures is crucial for a wider practical impact. Finally, the robustness and stability of these new mechanisms under real-world conditions, such as noisy or dynamically changing inputs, require further investigation. Addressing these limitations could improve reliability and efficiency in practical deployments.

### 5.3 Non-Transformer Architecture

While transformers are struggling with KV cache issues, researchers have revisited principles from traditional sequential architectures, such as recurrent neural networks (RNNs) Salehinejad et al. (2017), which inherently process sequences without the need for explicit KV caches. Inspired by the lightweight and memory-efficient design of RNNs and efficient attention mechanisms, non-transformer architectures Xu et al. (2024e); Hasani et al. (2022); Smith et al. (2022); Wang et al. (2022b); Gu & Dao (2024); Peng et al. (2023) have emerged, such as Mamba Gu & Dao (2024) and RWKV Peng et al. (2023), offering promising alternatives. While there are a large type of new architectures, we only list methods associated with KV optimization. For further understanding to efficient non-transformer works, please refer to these surveys Zhou et al. (2024c); Xu et al. (2024d); Qu et al. (2024); Patro & Agneeswaran (2024). The summary of non-transformer is listed in Tab. 9.

#### 5.3.1 Adaptive Sequence Processing Architectures

RWKV Peng et al. (2023), which means Receptance Weighted Key Value, is an architecture that combines the strengths of RNNs and transformers to achieve efficient sequence processing. RWKV integrates a linear attention mechanism, enabling parallelizable training like transformers while retaining the efficient inference characteristics of RNNs. By formulating the architecture to operate as either a transformer or an RNN, RWKV achieves constant computational and memory complexity during inference, overcoming the quadratic scaling issues of transformers.

Mamba Gu & Dao (2024) is built based on state space sequence models (SSMs) Gu et al. (2022; 2021). Inspired by the state space systems, SSMs build scalable and memory-efficient long-range sequence modeling frameworks. Mamba improves SSMs by making parameters input-dependent, allowing information to be selectively propagated or forgotten along the sequence based on the current token. This addresses the inability of traditional SSMs to effectively handle the complexity of nonlinear dependencies in natural languages. Mamba omits attention and even MLP blocks, relying entirely on these selective state spaces for sequence modeling. It also develops a hardware-aware parallel algorithm for efficient recurrent computations in training and inference. Mamba achieves linear scaling in sequence length, demonstrating exceptional performance on sequences of up to a million tokens.

RetNet Sun et al. (2023) introduces Retentive Network that combines elements of recurrence and attention, presenting a novel retention mechanism for sequence modeling that offers training parallelism, low-cost inference, and scalable performance together. The proposed Multi-scale Retention Module (MSR) enables support to multiple computation paradigms: the parallel representation is similar to self-attention that adds support to casual masks and parallel training. The recurrent representation is similar to RNN that allows low-cost inference by maintaining state across sequence decoding. The chunkwise recurrent representation constructs a hybrid form to the former representations to further enables handling long sequences. These combined characteristics position RetNet as a strong alternative to transformers without a heavy KV cache mechanism.

MCSD Yang et al. (2024b) features the new block called Multi-Channel Slope and Decay, which is made up of two sections: The slope section can capture local features across short temporal spans, and the decay section can capture global features across long temporal spans. The sections are fused through element-wise operations. During inference, the process would be reformat into a recurrent representation, allowing both spatial and temporal efficiency, minimizing the need for maintaining a large KV cache.

### 5.3.2 Hybrid Architecture

With these non-transformer architecture, some methods construct mixed models to alleviate KV cache necessities while keeping some peculiarities and merits of the self-attention mechanism.

MixCon Xu & Lin (2024) introduces a new architecture called Conba. Inspired by control theory, the Conba layer incorperates a feedback and adaptive control mechanism that can adapt to different sequence-modeling tasks and requirements dynamically with good computational efficiency. Furthermore, MixCon integrates the Mixture of Experts (MoE) module, which dynamically selects the most relevant experts to process parts of the sequence. Combining the transformer layer, the Conba layer, and the MoE module, MixCon constructs a hybrid model with good balance between attention effectiveness and computational efficiency and significantly reduces the total size of the KV cache.

GoldFinch Goldstein et al. (2024) first introduces several new architectures, including the GOLD layer, which combines the Llama and RWKV channel mixer with several improvements, and the enhanced Finch model (RWKV-6) that has significantly reduced parameters without sacrificing efficiency and performance. GoldFinch also proposes a novel mechanism called TokenCat to produce a highly compressed global key cache using the output of Finch layers. GoldFinch builds a hybrid architecture that constructs the key cache in the early layers and consumes the key cache to produce output without the traditional value cache in the top layers, providing a compact and reusable cache pipeline with linear scaling.

RecurFormer Yan et al. (2024) argues that not all transformer heads need to participate in the self-attention mechanism. The work recognizes that certain attention heads show recency-aware behavior which focus on local and short-range dependencies, dissipate the computation resource but gives little contribution. After identifying these heads, RecurFormer replaces them with the Mamba components, achieving straightforward KV cache reduction.

### 5.3.3 Summary and Future Directions

By exploring non-transformer modules such as recurrent and hybrid designs, these methods have introduced novel paradigms that balance performance with computational efficiency, and also alleviate the KV cache issues in traditional transformer architectures. Future research should focus on several key areas. First, improving the scalability of recurrent architectures, such as RWKV Peng et al. (2023) and Mamba Gu & Dao (2024), remains critical. Although these methods reduce memory and computational costs, their performance in capturing ultra-long-range dependencies lags behind transformers. Second, hybrid designs such as MixCon Xu & Lin (2024) and GoldFinch Goldstein et al. (2024) highlight the potential of integrating diverse modules, yet their complexity introduces challenges in training stability and interpretability. Third, the overall generalization capabilities and robustness of non-transformer architectures, while efficient, need require further exploration for diverse input modalities.

# 6 System-level Optimization

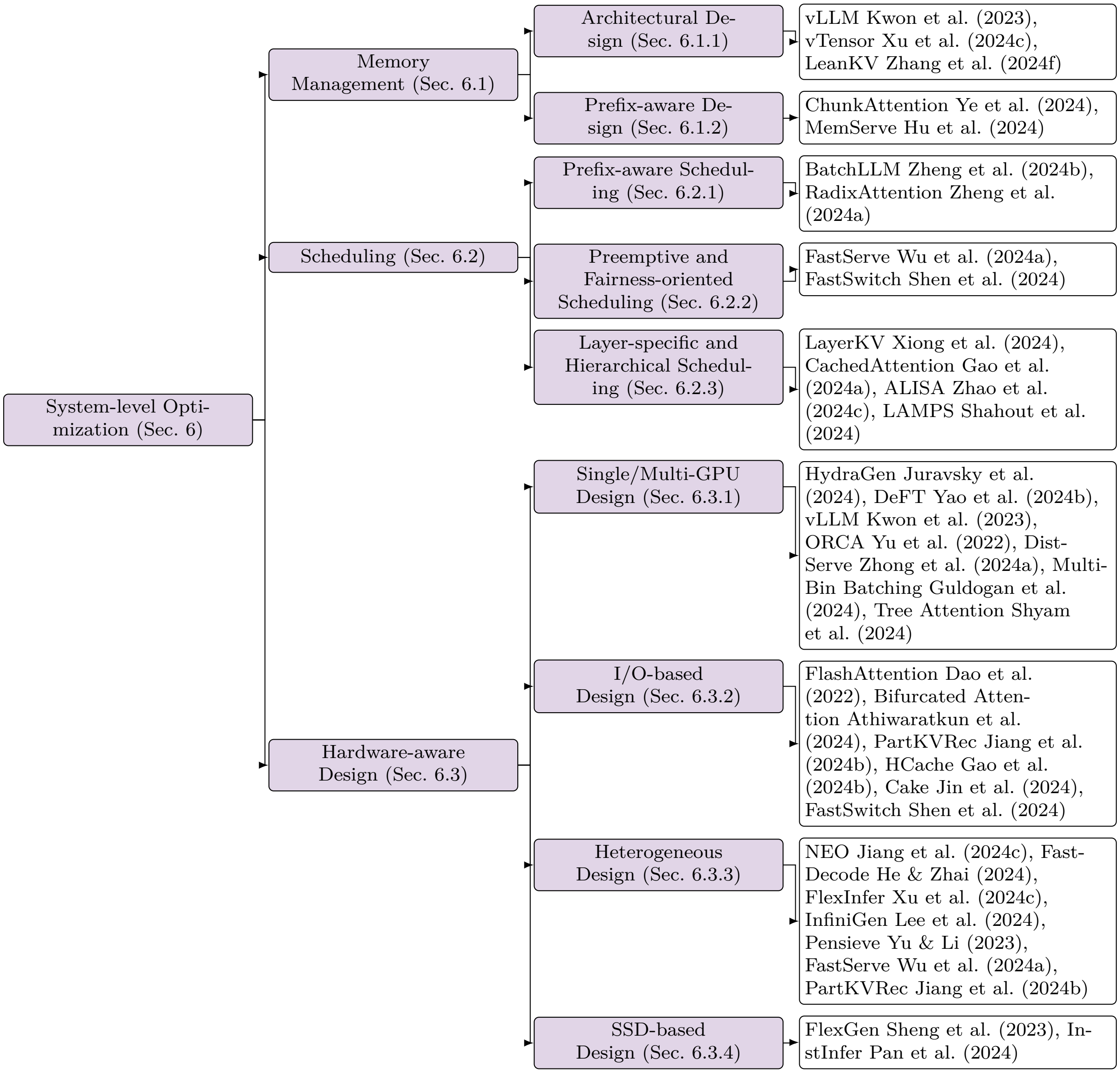
System-level Optimization (Sec. 6)
Memory Management (Sec. 6.1)
Architectural Design (Sec. 6.1.1)
vLLM Kwon et al. (2023), vTensor Xu et al. (2024c), LeanKV Zhang et al. (2024f)
Prefix-aware Design (Sec. 6.1.2)
ChunkAttention Ye et al. (2024), MemServe Hu et al. (2024)
Scheduling (Sec. 6.2)
Prefix-aware Scheduling (Sec. 6.2.1)
BatchLLM Zheng et al. (2024b), RadixAttention Zheng et al. (2024a)
Preemptive and Fairness-oriented Scheduling (Sec. 6.2.2)
FastServe Wu et al. (2024a), FastSwitch Shen et al. (2024)
Layer-specific and Hierarchical Scheduling (Sec. 6.2.3)
LayerKV Xiong et al. (2024), CachedAttention Gao et al. (2024a), ALISA Zhao et al. (2024c), LAMPS Shahout et al. (2024)
Hardware-aware Design (Sec. 6.3)
Single/Multi-GPU Design (Sec. 6.3.1)
HydraGen Juravsky et al. (2024), DeFT Yao et al. (2024b), vLLM Kwon et al. (2023), ORCA Yu et al. (2022), DistServe Zhong et al. (2024a), Multi-Bin Batching Guldogan et al. (2024), Tree Attention Shyam et al. (2024)
I/O-based Design (Sec. 6.3.2)
FlashAttention Dao et al. (2022), Bifurcated Attention Athiwaratkun et al. (2024), PartKVRec Jiang et al. (2024b), HCache Gao et al. (2024b), Cake Jin et al. (2024), FastSwitch Shen et al. (2024)
Heterogeneous Design (Sec. 6.3.3)
NEO Jiang et al. (2024c), FastDecode He & Zhai (2024), FlexInfer Xu et al. (2024c), InfiniGen Lee et al. (2024), Pensieve Yu & Li (2023), FastServe Wu et al. (2024a), PartKVRec Jiang et al. (2024b)
SSD-based Design (Sec. 6.3.4)
FlexGen Sheng et al. (2023), InstInfer Pan et al. (2024)

Figure 6: Taxonomy of the System-level Optimization for KV Cache Management.

Recent system-level optimizations for KV cache in LLM inference can be broadly categorized into three main directions: memory management (Sec. 6.1), scheduling strategies (Sec. 6.2), and hardware-aware designs (Sec. 6.3). These complementary approaches collectively demonstrate the rich design space for system-level optimizations in LLM inference, each addressing different aspects of the performance, efficiency, and resource utilization challenges. The Taxonomy of the system-level optimization is in Fig. 6.

### 6.1 Memory Management

Recent advances in KV cache memory management for large language model (LLM) inference reveal three distinct approaches aimed at enhancing memory efficiency. Architectural designs, exemplified by vLLM with PagedAttention Kwon et al. (2023) and vTensor Xu et al. (2024c), adapt classical operating system principles to create flexible, dynamic memory allocation systems that optimize the use of physical memory through sophisticated mapping and virtual memory abstractions. Prefix-aware designs like ChunkAttention Ye et al. (2024) and MemServe Hu et al. (2024) further refine this approach by organizing data structures to enable efficient cache deduplication and sharing of common prefixes, thereby improving both memory utilization and computational efficiency. Together, these innovations illustrate the potential for significant enhancements in LLM serving via memory management.

#### 6.1.1 Architectural Design

The first category focuses on architectural innovations in memory management, led by vLLM with PagedAttention Kwon et al. (2023), which adapts OS-inspired paging concepts by partitioning KV caches into fixed-size blocks with non-contiguous storage. PagedAttention partitions KV caches into fixed-size blocks that can be stored non-contiguously in physical memory, while vLLM Kwon et al. (2023) implements a virtual memory-like system that manages these blocks through a sophisticated mapping mechanism. This architecture separates logical and physical KV blocks, enabling dynamic memory allocation and flexible block management through block tables that track mapping relationships and fill states. This memory management approach enables efficient memory utilization both within and across requests, demonstrating how classical OS memory management principles can be effectively adapted for LLM inference optimization.

This approach is further enhanced by vTensor Xu et al. (2024c), which introduces a virtual memory abstraction that decouples computation from defragmentation through three key components: the vTensor Scheduler which generates memory management policies based on meta information, the vTensor Operation which translates these policies into CUDA VMM operations, and the vTensor Pool which maintains virtual tensor mappings. VTS processes instructions and creates policies based on memory state tracking, while VTO executes these policies through asynchronous GPU operations. VTP completes the cycle by managing virtual tensor storage and updating meta information for subsequent memory operations.

LeanKV Zhang et al. (2024f) combines unified paging with heterogeneous quantization and dynamic sparsity mechanisms. It implements Hetero-KV quantization to store keys and values at different precisions, complemented by a per-head dynamic sparsity mechanism that adapts memory allocation based on token importance across different attention heads and requests. To efficiently execute these strategies, LeanKV Zhang et al. (2024f) introduces an advanced on-GPU memory management system featuring three key components: unified paging for flexible memory organization, a circular free page list for efficient coordination, and a bidirectional page table for minimal metadata overhead.

#### 6.1.2 Prefix-aware Design

Some latest works emphasize optimizing data organization structures through prefix-aware designs. ChunkAttention Ye et al. (2024) restructures KV cache management by organizing chunks within a prefix tree structure, enabling runtime detection and sharing of common prefixes. It breaks down traditional monolithic KV cache tensors into smaller, manageable chunks organized within a prefix tree structure, enabling efficient runtime detection and sharing of common prefixes across multiple requests. This architectural design brings two significant memory management benefits: efficient KV cache deduplication through prefix tree-based organization, and improved data locality through a two-phase partition algorithm for self-attention computation. By enabling dynamic identification and sharing of common prompt prefixes across multiple requests, ChunkAttention Ye et al. (2024) optimizes both memory utilization and computational efficiency, demonstrating how intelligent chunking and prefix-aware cache management can significantly enhance LLM serving efficiency.

MemServe Hu et al. (2024) extends this concept to distributed settings with its MemPool system, which orchestrates both CPU DRAM and GPU HBM resources across serving instances, managing active and

Table 10: Comparison of Memory Management Techniques for KV Cache Optimization.

| Method | Paged Memory | Virtual Memory | Dynamic Sparsity | Prefix Sharing | Distributed Memory |
|---|---|---|---|---|---|
| **vLLM**<br>Kwon et al. (2023) | ✓ | ✓ | | | |
| **vTensor**<br>Xu et al. (2024c) | | ✓ | | | |
| **LeanKV**<br>Zhang et al. (2024f) | ✓ | | ✓ | | |
| **ChunkAttention**<br>Ye et al. (2024) | | | | ✓ | |
| **MemServe**<br>Hu et al. (2024) | | | | ✓ | ✓ |

historical KV caches through a comprehensive set of distributed memory pool APIs. It presents a prompt token-based indexing layer for historical KV cache retrieval, cross-instance data exchange mechanisms that abstract away hardware heterogeneity, and a global scheduler implementing a prompt tree-based locality-aware policy for enhanced cache reuse, collectively resulting in significant improvements in job completion time and time-to-first-token performance.

These approaches often complement each other, suggesting potential benefits in combining multiple strategies. For instance, LeanKV Zhang et al. (2024f)'s integration of compression with page-based management and MemServe Hu et al. (2024)'s combination of distributed memory management with prefix-aware caching demonstrate the effectiveness of hybrid approaches. The diversity of these solutions reflects both the complexity of KV cache management and the rich opportunity space for continued innovation in optimizing LLM inference systems. Tab.10 provides a comparison of various memory management techniques for KV Cache, highlighting key features such as paged memory, virtual memory, dynamic sparsity, prefix sharing, and distributed memory.

### 6.1.3 Summary and Future Directions

The exploration of memory management strategies for KV caches in large language model inference reveals a promising landscape of innovations that enhance memory efficiency and overall system performance. Architectural advancements, such as those seen in vLLM Kwon et al. (2023) and LeanKV Zhang et al. (2024f), adapt traditional memory management principles for modern AI applications by incorporating paging and virtual memory concepts for dynamic allocation. Prefix-aware designs like ChunkAttention Ye et al. (2024) and MemServe Hu et al. (2024) optimize data organization, enabling the detection and sharing of common prefixes, which reduces redundancy and speeds up inference.

Future work should advance memory management innovations through multiple synergistic directions: investigating adaptive memory hierarchies that dynamically adjust to workload patterns and resource constraints, exploring novel compression techniques that preserve quick access while reducing memory footprint, developing intelligent prefetching mechanisms that anticipate and preload frequently accessed cache entries, researching hardware-aware optimization strategies that leverage emerging memory technologies like computational storage and processing-in-memory units, and designing distributed cache coherence protocols that efficiently maintain consistency across multiple inference nodes. Additionally, the exploration of machine learning-based approaches could enable predictive memory allocation that learns from historical access patterns, while the investigation of specialized data structures could yield more efficient prefix detection and sharing mechanisms. These advancements, combined with research into heterogeneous memory systems that intelligently coordinate different memory types based on access patterns and performance requirements, would significantly enhance the scalability and efficiency of LLM inference systems across diverse deployment scenarios.

## 6.2 Scheduling

Based on these scheduling-oriented works, we can categorize KV cache scheduling optimizations into three main approaches: 1) prefix-aware scheduling strategies, represented by BatchLLM Zheng et al. (2024b) and RadixAttention Zheng et al. (2024a); 2) preemptive and fairness-oriented scheduling, exemplified by FastServe Wu et al. (2024a) and FastSwitch Shen et al. (2024); 3) layer-specific and hierarchical scheduling approaches, demonstrated by LayerKV Xiong et al. (2024), CachedAttention Gao et al. (2024a), and ALISA Zhao et al. (2024c). These approaches collectively address different aspects of scheduling optimization, from memory efficiency to fairness and latency reduction, while specialized solutions like LAMPS Shahout et al. (2024) extend these concepts to specific use cases such as API-augmented LLM requests, demonstrating the rich design space in KV cache scheduling optimization.

Table 11: Comparison of Scheduling Approaches for KV Cache Optimization.

| Method | Prefix-aware | Preemptive | Fairness-oriented | Layer-specific | Hierarchical | Dynamic |
|---|---|---|---|---|---|---|
| **BatchLLM**<br>Zheng et al. (2024b) | ✓ | | | | | |
| **RadixAttention**<br>Zheng et al. (2024a) | ✓ | | | | | ✓ |
| **FastServe**<br>Wu et al. (2024a) | | ✓ | ✓ | | | |
| **FastSwitch**<br>Shen et al. (2024) | | ✓ | ✓ | | | |
| **LayerKV**<br>Xiong et al. (2024) | | | | ✓ | | |
| **CachedAttention**<br>Gao et al. (2024a) | | | | ✓ | ✓ | |
| **ALISA**<br>Zhao et al. (2024c) | | | | ✓ | | ✓ |
| **LAMPS**<br>Shahout et al. (2024) | | | | | ✓ | ✓ |

### 6.2.1 Prefix-aware Scheduling

Unlike traditional LRU-based cache management systems where shared KV contexts might be prematurely evicted or unnecessarily extended in memory, BatchLLM Zheng et al. (2024b) implements explicit global prefix identification and coordinated scheduling of requests sharing common KV cache content. It schedules requests at the granularity of prefix-sharing groups, ensuring optimal KV cache reuse while minimizing cache lifetime - requests with identical prefixes are deliberately scheduled together to maximize KV cache sharing efficiency. This scheduling approach is complemented by a dynamic programming algorithm that optimizes first-level prefix patterns, enabling more efficient KV cache management and reducing scheduling overhead.

RadixAttention Zheng et al. (2024a) builds around a radix tree structure, replacing traditional FCFS scheduling with an intelligent cache-aware approach that prioritizes requests based on matched prefix lengths. It implements dynamic memory management where cached tokens and running requests share the same memory pool, controlled by an LRU eviction policy that strategically removes leaf nodes while preserving valuable ancestor prefixes. This is complemented by a reference counting mechanism that prevents eviction of actively used cache entries during continuous batching while enabling efficient memory reclamation when nodes become unused.

### 6.2.2 Preemptive and Fairness-oriented scheduling

FastServe Wu et al. (2024a) implements a proactive KV cache management strategy that coordinates cache movement between GPU and host memory, overlapping data transmission with computation to minimize latency impact. This is integrated with a skip-join Multi-Level Feedback Queue scheduler that makes KV cache scheduling decisions based on input length information, allowing jobs to enter appropriate priority queues directly while avoiding unnecessary demotions through higher-priority queues. By combining token-level preemption with sophisticated KV cache management and intelligent queue placement, FastServe Wu et al. (2024a) achieves significant performance improvements over traditional run-to-completion systems like vLLM Kwon et al. (2023).

FastSwitch Shen et al. (2024) introduces a fairness-oriented KV cache scheduling system that addresses the overhead challenges of preemptive scheduling in LLM serving. There are three key mechanisms: enhancing I/O utilization through intelligent cache movement scheduling, minimizing GPU idle time during context switches, and eliminating redundant I/O operations in multi-turn conversations. Unlike traditional block-based KV cache memory policies that prioritize memory efficiency at the cost of fragmentation and granularity limitations, FastSwitch Shen et al. (2024) implements a balanced approach that maintains efficient memory usage while facilitating smoother context switching. This integrated scheduling approach enables dynamic priority adjustments for fairness while minimizing the performance impact of context switches.

### 6.2.3 Layer-specific and Hierarchical Scheduling

LayerKV Xiong et al. (2024) introduces a novel layer-wise KV cache scheduling approach to address the growing TTFT (Time to First Token) latency challenges in large-context LLM serving. The contribution lies in its fine-grained, layer-specific KV cache block allocation and management strategy, which departs from traditional monolithic cache management approaches. By implementing layer-wise KV block scheduling and offloading mechanisms, LayerKV Xiong et al. (2024) enables more efficient memory utilization and reduces queuing delays that typically occur when large context windows compete for limited GPU KV cache blocks. It is complemented by an SLO-aware scheduler that optimizes cache allocation decisions based on service level objectives, allowing for dynamic management of memory resources across model layers.

CachedAttention Gao et al. (2024a) introduces a hierarchical scheduling approach consisting of three-tier strategies: layer-wise pre-loading coordinates KV cache movement across storage hierarchies using scheduler-aware fetching and eviction policies, asynchronous saving overlaps I/O operations with GPU computation, and intelligent cache placement decisions are made based on scheduler hints to ensure frequently accessed KV caches reside in faster memory tiers. It also presents a novel positional encoding decoupling mechanism that prevents KV cache invalidation during context window overflow through effective truncation strategies.

ALISA Zhao et al. (2024c) introduces a dual-level KV cache scheduling framework that combines algorithmic sparsity with system-level optimization. At the algorithm level, the Sparse Window Attention mechanism identifies and prioritizes the most important tokens for attention computation, creating a mixture of global dynamic and local static sparse patterns that significantly reduces KV cache memory requirements. At the system-level, its three-phase token-level dynamic scheduler that manages KV tensor allocation and optimizes the trade-off between caching and recomputation. The scheduler makes dynamic decisions about which tokens to cache in GPU memory versus recompute, based on their importance and system resource constraints.

LAMPS Shahout et al. (2024) implements a predictive scheduling mechanism that estimates both pre-API outputs and optimal memory handling strategies during API calls, choosing between preserving, discarding, or swapping KV cache content based on predicted memory waste.

### 6.2.4 Summary and Future Directions

Tab.11 compares scheduling approaches for KV cache optimization based on their support for prefix-awareness, preemptive scheduling, fairness, layer-specific optimizations, hierarchical structures, and dynamic adaptability. The advancements in scheduling strategies for KV cache management in large language model inference highlight a multifaceted approach to optimizing performance, memory efficiency, and fairness. By categorizing these strategies into prefix-aware, preemptive and fairness-oriented, and layer-specific scheduling, we see diverse methodologies addressing different challenges. For instance, prefix-aware strategies like BatchLLM Zheng et al. (2024b) and RadixAttention Zheng et al. (2024a) enhance cache reuse by intelligently grouping requests based on shared prefixes, minimizing cache lifetime and reducing overhead. Meanwhile, preemptive approaches such as FastServe Wu et al. (2024a) and FastSwitch Shen et al. (2024) implement proactive management techniques that optimize cache movement and scheduling, significantly improving latency and ensuring fairness during context switching. Layer-specific scheduling methods like LayerKV Xiong et al. (2024), CachedAttention Gao et al. (2024a), and ALISA Zhao et al. (2024c) further refine cache allocation by implementing fine-grained management strategies tailored to the unique demands of different model layers.

Future work should advance these KV cache scheduling innovations through several interlinked dimensions: developing adaptive hybrid systems that dynamically select optimal scheduling strategies based on real-time workload characteristics, exploring predictive models that anticipate user request patterns to proactively optimize cache allocation, investigating automated parameter tuning mechanisms that adjust scheduling policies across different deployment scenarios, designing context-aware architectures that intelligently balance prefix sharing with fairness requirements, and researching novel cache coherence protocols that efficiently handle distributed inference scenarios. Additionally, the integration of reinforcement learning approaches could enable self-optimizing schedulers that learn from historical usage patterns, while the exploration of hardware-software co-design could yield specialized accelerators that directly support efficient KV cache management operations. These advancements would collectively enhance the robustness, efficiency, and adaptability of LLM inference systems across diverse operational conditions and deployment scales. Finally, considering LLM serving Yao et al. (2024a), different scheduling and sharing for multiple users and queries may lead to potential privacy leaks. Therefore, privacy protection techniques for LLM serving in multi-user scenarios, such as differential privacy Zhao & Chen (2022); Dong & Yi (2021); Dong et al. (2023a), are worth further investigation.

Table 12: Comparison of Hardware-aware Design Approaches for KV Cache Optimization.

| Method | Single/Multi-GPU | I/O-aware | Heterogeneous | SSD-based |
|---|---|---|---|---|
| **Bifurcated Attention** Athiwaratkun et al. (2024) | | ✓ | | |
| **Cake** Jin et al. (2024) | | | | ✓ |
| **DeFT** Yao et al. (2024b) | ✓ | | | |
| **DistServe** Zhong et al. (2024a) | | | ✓ | |
| **FastDecode** He & Zhai (2024) | | ✓ | | |
| **FastSwitch** Shen et al. (2024) | ✓ | | | |
| **FlexGen** Sheng et al. (2023) | | ✓ | | |
| **FlexInfer** Xu et al. (2024c) | | | | ✓ |
| **FlashAttention** Dao et al. (2022) | ✓ | | ✓ | |
| **HCache** Gao et al. (2024b) | | | ✓ | |
| **HydraGen** Juravsky et al. (2024) | ✓ | | | |
| **InfiniGen** Lee et al. (2024) | | | ✓ | |
| **InstInfer** Pan et al. (2024) | | | | |
| **Multi-Bin Batching** Guldogan et al. (2024) | | | | ✓ |
| **NEO** Jiang et al. (2024c) | | | ✓ | |
| **ORCA** Yu et al. (2022) | ✓ | | | |
| **PartKVRec** Jiang et al. (2024b) | | ✓ | | |
| **Pensieve** Yu & Li (2023) | | ✓ | | |
| **Tree Attention** Shyam et al. (2024) | | ✓ | | |
| **vLLM** Kwon et al. (2023) | ✓ | | | |

### 6.3 Hardware-aware Design

Recent hardware-aware optimizations for KV cache management span several key directions based on different hardware architectures and constraints. Single/Multi-GPU designs focus on optimizing memory access patterns, GPU kernel designs for efficient attention computation, and parallel processing with load balancing. IO-based designs optimize data movement across memory hierarchies through asynchronous I/O and intelligent prefetching mechanisms. Heterogeneous designs orchestrate computation and memory allocation across CPU-GPU tiers. SSD-based solutions have evolved from basic offloading approaches to more sophisticated designs, with InstInfer leveraging computational storage drives (CSDs) to perform in-storage attention computation, effectively bypassing PCIe bandwidth limitations. These approaches demonstrate how hardware-aware designs can significantly improve LLM inference efficiency by carefully considering and exploiting the characteristics of different hardware components and their interconnections.

### 6.3.1 Single/Multi-GPU Design

Based on these works focusing on GPU-oriented designs, we can categorize the approaches into several key strategies for KV cache optimization. First, shared prefix optimization approaches like HydraGen Juravsky et al. (2024) and DeFT Yao et al. (2024b) focus on efficient GPU memory utilization through batched prefix computations and tree-structured attention patterns. Rather than maintaining separate KV caches for each sequence with identical prefixes, HydraGen Juravsky et al. (2024) decomposes attention computation to leverage a single shared KV cache for common prefixes across multiple requests. It enables efficient GPU memory utilization through two mechanisms: batched prefix KV cache access across sequences and separate handling of unique suffix KV caches. For DeFT Yao et al. (2024b), its core contributions are twofold: KV-Guided Grouping, which optimizes GPU memory access patterns by intelligently managing shared prefix KV caches to minimize redundant global-to-shared memory transfers, and Flattened Tree KV Splitting, which ensures balanced workload distribution across GPU compute units while minimizing computational redundancy.

Second, distributed processing frameworks exemplified by vLLM Kwon et al. (2023) and ORCA Yu et al. (2022) optimize multi-GPU scenarios through sophisticated memory management and synchronization mechanisms. vLLM Kwon et al. (2023) also implements a KV cache manager that coordinates memory allocation across distributed GPU workers in model-parallel deployments, where each GPU handles a subset of attention heads while sharing the same logical-to-physical block mapping. This GPU-aware design enables efficient memory utilization through near-zero fragmentation and flexible KV cache sharing, while supporting Megatron-LM style tensor parallelism where GPUs execute in SPMD fashion with synchronized block-wise matrix operations. The scheduler broadcasts control messages containing input tokens and block tables to GPU workers, allowing them to independently process their assigned attention heads while maintaining memory coherence through all-reduce operations, effectively eliminating redundant memory management synchronization overhead and maximizing GPU utilization across distributed resources.

ORCA Yu et al. (2022) distributes model layers across GPUs using both intra-layer and inter-layer parallelism, where each worker process manages multiple GPU-controlling threads and coordinates KV cache access through an Attention KV manager. ORCA's GPU-aware design minimizes CPU-GPU synchronization overhead by separating control message communication from tensor data transfer (via NCCL), allowing each GPU thread to efficiently access KV cache memory using request IDs and token indices.

Third, phase-aware designs like DistServe Zhong et al. (2024a) separate prefill and decoding phases across GPU resources to optimize their distinct memory access patterns. Novel batching strategies are represented by Multi-Bin Batching Guldogan et al. (2024), which focuses on length-aware request grouping for improved GPU utilization, while advanced parallel computation frameworks like Tree Attention Shyam et al. (2024) introduce sophisticated reduction algorithms for efficient attention computation across multiple GPUs. DistServe Zhong et al. (2024a) recognizes that prefill and decoding phases have distinct KV cache utilization characteristics and memory access patterns: prefill requires intensive computation with growing KV cache sizes for processing input tokens, while decoding maintains a fixed KV cache size for generating output tokens. By physically separating these phases onto different GPUs, DistServe enables optimized GPU memory management and KV cache access patterns specific to each phase, eliminating interference between prefill's bursty memory access patterns and decoding's steady-state KV cache utilization. Multi-Bin Batching Guldogan et al. (2024) introduces a length-aware batching strategy helps minimize GPU idle time and memory fragmentation that typically occurs when processing requests of varying lengths in the same batch, as it ensures that the KV cache memory allocated for each batch is utilized more uniformly across all requests. Tree Attention Shyam et al. (2024) implements a tree-based reduction algorithm that fundamentally changes how attention values are computed and aggregated across GPUs, enabling more efficient handling of KV cache data through partial reductions that significantly reduce memory bandwidth requirements and peak memory usage.

These approaches can collectively demonstrate how hardware-aware designs can significantly improve the LLM efficiency by carefully considering GPU architecture characteristics and memory hierarchy constraints.

### 6.3.2 I/O-based Design

Recent I/O-focused optimizations for KV cache management span several key dimensions, targeting different levels of the memory hierarchy. At the GPU level, approaches like FlashAttention Dao et al. (2022) and Bifurcated Attention Athiwaratkun et al. (2024) optimize data movement between HBM and SRAM through sophisticated tiling strategies and split attention computations, while CPU-GPU data movement optimizations are addressed by systems like PartKVRec Jiang et al. (2024b), which tackles PCIe bandwidth bottlenecks through hybrid recomputation and transfer strategies, and HCache Gao et al. (2024b), which optimizes intermediate activation storage and restoration.

FlashAttention Dao et al. (2022) employs a tiling strategy that carefully manages KV cache access patterns, reducing redundant memory operations by keeping frequently accessed portions of the KV cache in fast SRAM while systematically fetching and evicting data blocks to minimize HBM accesses. Bifurcated Attention Athiwaratkun et al. (2024) presents an I/O-aware approach to optimize KV cache access patterns during shared-context batch decoding by strategically splitting attention computations into two distinct GEMM operations. It specifically targets the memory bandwidth bottleneck in high-batch scenarios with long contexts by minimizing repeated KV cache accesses, maintaining the same computational FLOPs while drastically reducing memory I/O operations. For PartKVRec Jiang et al. (2024b), its key innovation lies in its hybrid strategy of partial KV cache recomputation on the GPU while simultaneously transferring the remaining cache data from CPU memory, effectively hiding PCIe transfer latency. The implementation employs a sophisticated I/O-aware scheduling system that analyzes input characteristics and hardware capabilities to determine the optimal balance between recomputation and data transfer, dynamically managing KV cache movement to maximize PCIe bandwidth utilization while minimizing GPU idle time. HCache Gao et al. (2024b) strategically stores and restores intermediate activations instead of complete KV cache states, implementing a bubble-free restoration scheduler that carefully balances computation and I/O operations to maximize bandwidth utilization. A key innovation is its chunk-based storage manager that addresses the I/O pattern mismatch between saving (layer-before-token) and restoration (token-before-layer) operations, optimizing data layout and access patterns to reduce I/O overhead. Cake Jin et al. (2024) addresses the fundamental I/O bottleneck in loading cached KV states from disk to GPU memory. It introduces a bidirectional parallelized strategy that simultaneously leverages both computational and I/O resources. This hybrid approach dynamically balances between loading cached KV states from storage and computing them on GPUs, adapting automatically to varying system conditions without manual parameter tuning.

Context management optimizations are exemplified by FastSwitch Shen et al. (2024), which implements efficient context switching mechanisms for multi-user scenarios through granular memory management policies. FastSwitch Shen et al. (2024) addresses I/O inefficiencies in traditional block-based KV cache approaches by implementing a more granular and continuous memory management policy that minimizes I/O overhead during preemption and context switching.

These approaches demonstrate how careful consideration of I/O patterns and memory hierarchy characteristics can significantly improve LLM inference efficiency by minimizing data movement and maximizing bandwidth utilization across different storage tiers.

### 6.3.3 Heterogeneous Design

Recent heterogeneous computing approaches for KV Cache demonstrate diverse strategies for optimizing CPU-GPU collaboration. Systems like NEO Jiang et al. (2024c) and FastDecode He & Zhai (2024) implement strategic workload distribution through CPU offloading of attention computations, while FlexInfer Xu et al. (2024c) introduces virtual memory abstractions for optimal resource coordination.

NEO Jiang et al. (2024c) advances heterogeneous computing for LLM inference by implementing strategic CPU offloading of attention computations and KV cache states. Through asymmetric GPU-CPU pipelining and load-aware scheduling, it optimally balances workloads across both computing platforms, enabling larger GPU batch sizes without latency penalties. For FastDecode He & Zhai (2024), its key contribution lies in its strategic offloading of memory-bound KV cache operations to distributed CPU resources, leveraging the aggregate memory capacity and computing power of multiple CPU nodes rather than treating CPUs as mere storage devices. By utilizing CPUs for KV cache computations and storage while keeping compute-intensive operations on GPUs, it creates an efficient pipeline that maximizes resource utilization across the heterogeneous infrastructure, enabling larger batch sizes and higher throughput. FlexInfer Xu et al. (2024c) orchestrates CPU-GPU resource utilization for LLM inference by introducing the virtual memory-based abstraction vTensor.

Advanced caching and prefetching mechanisms are exemplified by InfiniGen Lee et al. (2024), which employs speculative prefetching for KV cache entries, and Pensieve Yu & Li (2023), which implements multi-tier caching for conversation states. For InfiniGen Lee et al. (2024), its key innovation lies in its prediction mechanism that operates across the heterogeneous architecture, using partial computation of attention inputs and modified query-key weights to identify and prefetch only the most relevant KV cache entries from CPU memory to GPU. Pensieve Yu & Li (2023) introduces a heterogeneous computing architecture specifically designed for multi-turn conversation LLM serving by implementing a sophisticated multi-tier caching strategy across GPU and CPU resources. This stateful approach manages KV cache data across the heterogeneous memory hierarchy, maintaining conversation history states across multiple hardware tiers rather than recomputing them for each interaction.

Sophisticated scheduling and preemption strategies are demonstrated by FastServe Wu et al. (2024a), which focuses on token-level preemption and proactive memory management, and PartKVRec Jiang et al. (2024b), which balances data transfer and recomputation through dynamic scheduling. For FastServe Wu et al. (2024a), its token-level preemption capability is supported by a sophisticated heterogeneous memory management system that proactively coordinates KV cache data movement between GPU and host memory. It implements a skip-join Multi-Level Feedback Queue scheduler that manages computational resources across the CPU-GPU boundary, optimizing both computation scheduling and data movement. PartKVRec Jiang et al. (2024b) employs a scheduler that dynamically optimizes the distribution of tasks across the heterogeneous hardware platform, using a profiler to analyze both hardware capabilities and workload characteristics.

These approaches collectively showcase how heterogeneous architectures can be effectively leveraged to overcome single-device limitations while maintaining efficient resource utilization and minimizing communication overhead between CPU and GPU resources.

### 6.3.4 Solid-state Disk (SSD)-based Design

Recent SSD-based approaches for KV cache management demonstrate an evolution in storage utilization strategies, from traditional extension of the memory hierarchy to computational storage innovations. FlexGen Sheng et al. (2023) introduces an SSD-based approach to KV cache management that extends the memory hierarchy across GPU, CPU memory, and disk storage, optimizing high-throughput LLM inference on resource-constrained hardware through intelligent tensor storage and access pattern optimization determined by linear programming. The system's key innovations include coordinated data placement across all three storage tiers, optimized access patterns to minimize SSD latency impact, aggressive 4-bit compression for both model weights and attention cache, and efficient utilization of SSD storage as a memory hierarchy extension for KV cache management. InstInfer Pan et al. (2024) introduces a more revolutionary approach by leveraging computational storage drives (CSDs) to perform attention computations directly within the storage layer, transforming SSDs from passive storage devices into active computational units and utilizing the high internal bandwidth of flash memory channels to bypass traditional PCIe bandwidth limitations.

These approaches demonstrate how storage devices can be effectively integrated into LLM inference systems, either as memory hierarchy extensions or as computational resources, to enable efficient processing of large models and long sequences in resource-constrained environments. Tab.12 compares hardware-aware design approaches for KV cache optimization across four key features: Single/Multi-GPU support, I/O-awareness, heterogeneous computing, and SSD-based design.

### 6.3.5 Summary and Future Directions

Recent advancements in hardware-aware designs for KV cache management emphasize optimizing performance based on specific hardware architectures and constraints, demonstrating significant enhancements in large language model inference efficiency. Approaches like HydraGen Juravsky et al. (2024) and vLLM Kwon et al. (2023) in single and multi-GPU designs focus on efficient memory access patterns and load balancing, while I/O-based strategies such as FlashAttention Dao et al. (2022) and PartKVRec Jiang et al. (2024b) tackle data movement bottlenecks through intelligent prefetching and scheduling mechanisms. Additionally, heterogeneous designs exemplified by NEO Jiang et al. (2024c) and FastDecode He & Zhai (2024) effectively leverage CPU-GPU collaboration to maximize resource utilization.

Future work should advance this research through multiple interconnected directions: exploring novel architectural designs that combine specialized hardware accelerators with optimized memory hierarchies, investigating hybrid systems that leverage computational storage drives and processing-in-memory capabilities, developing self-adaptive algorithms that dynamically optimize resource allocation based on workload patterns, researching advanced compression techniques that maintain model fidelity while reducing memory requirements, and designing intelligent scheduling mechanisms that efficiently coordinate heterogeneous computing resources including CPUs, GPUs, and custom accelerators. These improvements, working in concert, would enhance both the performance and scalability of LLM inference systems across diverse deployment scenarios, from edge devices to data centers, while maintaining adaptability to emerging hardware innovations and varying computational demands.

## 7 Text and Multi-modal Datasets

In this section, we introduce the text and multi-modal datasets used to evaluate LLM efficiency.

### 7.1 Text Dataset

We collect a lot of long-context datasets from state-of-the-art benchmark frameworks and various papers, including L-EvalAn et al. (2023), M4LEKwan et al. (2023), BAMBOODong et al. (2023b), LongBenchBai et al. (2023), LRATay et al. (2020), SCROLLSShaham et al. (2022), ZEROSCROLLSShaham et al. (2023), LooGLELi et al. (2023a), LongEvalLi* et al. (2023), and StreamingEvalXiao et al. (2024c). Specifically, we categorize these datasets into different tasks, including question answering, text summarization, text reasoning, text retrieval, and text generation.

### 7.1.1 Question Answering (QA) Task

Dataset for this task usually consist of question-answer pairs, and documents that contains the answer to the question. For a model to run such task, documents and questions are usually used as the model input, while the output can differ greatly. Some datasets' answers are closed-ended, meaning that the model should only output its answer in designated form, typically multiple choice answers, while the open-ended answers take a more free form. According to the number of documents involved in a question-answer pair, we can categorize QA task datasets into single-doc QA(QA-SG) and multiple-doc QA(QA-MT). The detailed statistics of the datasets for question answering are provided in Table 13.

- **Qasper Dasigi et al. (2021)** consists of 5049 questions based on 1585 papers on NLP. Question is from NLP practitioners that only have read the abstract and title of a paper, then another set of practitioners answer these questions by reading through the whole paper. The supporting evidences is provided correspondingly. Each instance of the dataset consists of a question, an answer, corresponding paper and supporting evidence. Instances built by LongBench Bai et al. (2023) doesn't require evidence.

- **HotpotQA Yang et al. (2018)** is a typical for a multi-doc QA dataset. It's built based on Wikipedia, and each instance consists of multiple documents, a question, an answer and supporting facts. Supporting facts is a set of paragraph indexes, annotated manually.

Table 13: Text question answering (QA) dataset. In the **Avg. Len:** average length, **Tok:** tokens; **W:** words. In the Instances column, **Doc:** documents, **Q:** questions, **Inst:** instructions. Particularly, AltQA, PaperQA and MeetingQA have two datasets with different length levels, and is separated with /. Particularly, for datasets from L-Eval, the GPT-4 metric means the win-rate against Turbo-16K, judged by GPT-4. $\Delta L$ is the length difference between answer length and ground truth. For NarrativeQA, **MRR:** Mean Reciprocal Rank .

| Task | Name | Source | Instances | Avg Len | Metric | Lang. |
|---|---|---|---|---|---|---|
| QA | AltQA<br>Pal et al. (2023) | Wikipedia | 200/200 | 3243/13,084 Tok | Acc | EN |
| QA | PaperQA(BAMBOO)<br>Dong et al. (2023b) | Paper | 100/100 | 3101/6838 Tok | Acc | EN |
| QA | MeetingQA(BAMBOO)<br>Dong et al. (2023b) | Meeting | 100/100 | 2738/9838 Tok | Acc | EN |
| QA | TriviaQA<br>Joshi et al. (2017) | Web Question, Wiki | 95,956 Q, 662,659 Doc | 17,370 W | EM, F1 | EN |
| QA | TOEFL(L-Eval)<br>An et al. (2023) | TOFEL-QA<br>Tseng et al. (2016) | 15 Doc, 269 Inst | 3907 Tok | Rouge-L, GPT-4, $\Delta L$ | EN |
| QA | Coursera(L-Eval)<br>An et al. (2023) | Video Subtitles | 15 Doc, 172 Inst | 9075 Tok | Rouge-L, GPT-4, $\Delta L$ | EN |
| QA | SFiction(L-Eval)<br>An et al. (2023) | SFGram<br>Schaetti (2018), fiction | 7 Doc, 64 Inst | 16,381 Tok | Rouge-L, GPT-4, $\Delta L$ | EN |
| QA | LongFQA(L-Eval)<br>An et al. (2023) | Financial Transcripts | 6 Doc, 52 Inst | 6032 Tok | Rouge-L, GPT-4, $\Delta L$ | EN |
| QA | CUAD(L-Eval)<br>An et al. (2023) | CUAD<br>Hendrycks et al. (2021) | 20 Doc, 130 Inst | 30,966 Tok | Rouge-L, GPT-4, $\Delta L$ | EN |
| QA | DuoRC<br>Kwan et al. (2023) | Movie | - | 3572 W | Acc | EN |
| QA | NQ<br>Kwiatkowski et al. (2019) | Wiki | 307,373 | 9005 W | Rouge | EN |
| QA-SG | NarrativeQA<br>s Koˇ ciský et al. (2018) | Story | 1572 Doc | 62,528 Tok | BLEU, METEOR,<br>Rouge-L, MRR | EN |
| QA-SG | NarrativeQA(LongBench)<br>Bai et al. (2023) | Story | 200 | 18,409 W | F1 | EN |
| QA-SG | Qasper<br>Dasigi et al. (2021) | Paper | 1585 | 5001 W | F1 | EN |
| QA-SG | Qasper(LongBench)<br>Bai et al. (2023) | Paper | 200 | 3619 W | F1 | EN |
| QA-SG | MultifieldQA-en<br>Bai et al. (2023) | Paper, Legal,<br>Gov, Google | 200 | 4459 W | F1 | EN |
| QA-SG | MultifieldQA-zh<br>Yang et al. (2018) | Paper, Legal,<br>Gov, Google | 200 | 6701 W | F1 | ZH |
| QA-SG | QuALITY<br>Pang et al. (2021) | Story, magazine | 381 Doc, 6737 Q | 4203 W | EM | EN |
| QA-MT | HotpotQA<br>Yang et al. (2018) | Wiki | 112,779 | 1138 W | EM, F1 | EN |
| QA-MT | HotpotQA(LongBench)<br>Bai et al. (2023) | Wiki | 200 | 9151 W | F1 | EN |
| QA-MT | 2WikiMultihopQA<br>Ho et al. (2020) | Wiki | 192,606 Q | 639 W | EM, F1 | EN |
| QA-MT | MuSiQue<br>Trivedi et al. (2021) | Wiki | 24,814 | 1827 W | F1 | EN |
| QA-MT | DuReader<br>He et al. (2017) | Baidu | 200,000 Q, 1,000,000 Doc | 396 W | BLEU, Rouge-L | ZH, EN |
| QA+RET | NewsQA(M4LE)<br>Kwan et al. (2023) | News | - | 3679 W | Acc | EN |
| QA+RET | C3(M4LE)<br>Kwan et al. (2023) | Textbook | - | 3797 W | Acc | ZH |

- **AltQA Pal et al. (2023)** is based on google's NQ Kwiatkowski et al. (2019) dataset. The answer are all numerical. The original document is "altered" so that each occurrences of the numerical answer is different from the original document, so as to avoid data contamination from pretraining. This dataset is also used in BAMBOO Dong et al. (2023b) benchmark.

Table 14: Text Dataset-Summarization. In the **Avg. Len:** average length, **Tok:** tokens; **W:** words. In the Instances column, **Doc:** documents, **Q:** questions, **Inst:** instructions. Particularly, SPACE has the concept of 'Entity', and R/Ent stands for reviews per entity. Sum stands for summary. In the Metric column, **EM:** Exact Match. **PM:** Partial Match. **Acc:** Accuracy. For MultiNews, **Rouge-SU** skip bigrams when having a distance larger than 4 words. Particularly, LooGLE utilizes GPT-4 for its QA and summarization task, using it for answer's semantic judgement.

| **Task** | **Name** | **Source** | **Instances** | **Avg Len** | **Metric** | **Lang.** |
|---|---|---|---|---|---|---|
| SUM | CNN/Dailymail<br>Nallapati et al. (2016) | News | 300,000 | 766 W | Rouge-1/2/L | EN |
| SUM | XSum<br>Narayan et al. (2018) | News | 400,000 | 431 W | Rouge-1/2/L | EN |
| SUM | QMSum<br>Zhong et al. (2021) | Meeting | 232 Meets, 1808 Q | 9070 W | Rouge-1/2/L | EN |
| SUM | MultiNews<br>Fabbri et al. (2019) | News | 51,216 | 5866 W | Rouge-1/2/SU | EN |
| SUM-QB+<br>Reasoning+<br>QA | LooGLE<br>Li et al. (2023a) | Papers, Wiki,<br>Movie, TV | 776 Doc, 6448 Q | 19,367 W<br>24,005 Tok | BLEU, Rouge, METEOR,<br>BERT, GPT4,<br>EM, PM | EN, ZH |
| SUM | GovReport<br>Huang et al. (2021) | Gov | 19,466 | 9409.4 W | Rouge-1/2/L | EN |
| SUM | VCSUM<br>Wu et al. (2023a) | Meeting | 239 | 14,107 Tok | F1, Gold Rouge-1 | ZH |
| SUM | SummScreenFD<br>Chen et al. (2021b) | TV | 269,000 | 6613 Tok | Rouge | EN |
| SUM | BigPatent<br>Sharma et al. (2019) | Patent | 1,341,362 | 3573 W | Rouge-1/2/L | EN |
| SUM | SPACE<br>Angelidis et al. (2021) | Review | 50 Entities,<br>1,140,000 Reviews,<br>100R/Ent,<br>1050 Sum | 15,532 W | Rouge-1/2/L | EN |
| SUM | SQuALITY<br>Wang et al. (2022a) | Story | 625 | 5200 W | Rouge-1/2/L,<br>METEOR,<br>BERT | EN |
| SUM+RET | CNNNews(M4LE)<br>Kwan et al. (2023) | News | - | 3754 W | Rouge-L | EN |
| SUM+RET | CEPSUM(M4LE)<br>Kwan et al. (2023) | E-Commerce | - | 4003 W | Rouge-L | ZH |
| SUM+RET | LCSTS(M4LE)<br>Kwan et al. (2023) | News | - | 4102 W | Rouge-L | ZH |
| SUM+RET | NCLS(M4LE)<br>Kwan et al. (2023) | NCLS Zhu et al. (2019) | - | 3470 W | Rouge-L | EN, ZH |
| SUM+RET | WikiHow<br>Kwan et al. (2023) | Wiki | - | 3514 W | Rouge-L | EN |
| SUM+RET | News2016<br>Kwan et al. (2023) | News | - | 3785 W | Rouge-L | ZH |
| SUM | Pubmed(M4LE)<br>Kwan et al. (2023) | Medical | 1267 | 3678 W | Rouge-L | EN |
| SUM | BookSum(M4LE)<br>Kwan et al. (2023) | Book | - | 2643 W | Rouge-L | EN |
| SUM | CNewsum(M4LE)<br>Kwan et al. (2023) | News | 690 | 1883 W | Rouge-L | ZH |
| SUM | CLTS+(M4LE)<br>Kwan et al. (2023) | News | - | 3158 W | Rouge-L | ZH |
| SUM | Arxiv(M4LE)<br>Kwan et al. (2023) | Paper | 1550 | 3748 W | Rouge-L | EN |

- **PaperQA** and **MeetingQA** from BAMBOO Dong et al. (2023b) benchmark are question answering tasks in the form of multiple-choice. Each instance of the two datasets consists of question , evidence, answer and corresponding content.
- **NarrativeQA** s Koˇ ciský et al. (2018) uses complex narratives that are self-contained as input documents. Both books and movie scripts are used. For question construction, annotators are only given a story summary, and are asked to write questions based on it. For each story(1572 stories in total), about 30 question-answer pairs are constructed from each summary-story pair. Notably, because of the consistency in story context, the task can be simplified to selecting a correct answer from all answers that relates to the story.
- **MultifieldQA** Bai et al. (2023) is an original dataset from Longbench. Its contents covers scientific papers, legal documents, government reports and google results. The dataset has both Chinese and English version, and each instance consists of context built on documents, and a question-answer pair.
- **2WikiMultihopQA** Ho et al. (2020) is a multi-document QA dataset built on Wikipedia and Wikidata. WikiData is a Knowledge Graph database, from which the author was able to extract the (subject entity, property, object entity) triple that corresponds to a Wikipidia document. These triples are used as evidences in each QA pair, as a way for model to show its inference process. The dataset consists of 192,606 questions in total.
- **Musique Trivedi et al. (2021)** is also a multi-document dataset(or multi-hop dataset, as the paper refers to). Its data is extracted from existing single-hop QA datasets. These single-hop QAs are then composed into multi-hop QA pairs. In addition, Musique add some unanswerable QA pairs in order to further test model's ability. There are 24,814 answerable questions in Musique, and each answerable question corresponds to an unanswerable question.
- **DuReader** He et al. (2017) is a multi-document QA dataset, whose data is based on Baidu search results. It consists of 200,000 questions, 1,000,000 documents and 420,000 answers. Each instance contains a question, multiple possible answers(also possible to be empty), and multiple documents.
- **TriviaQA** Joshi et al. (2017) is a multi-document reading comprehension QA dataset. All QA pairs are from 14 trivia websites, written by trivia enthusiasts. For each QA pair, 6 supporting documents(evidence) are provided, collected from Bing search API as well as Wikipedia. The total number of QA pairs is 95,956, with a total of 662,659 supporting documents, the average length of each document is 2895 words.
- **TOEFL(L-Eval)** An et al. (2023) collect lectures from the TOEFL Practice Online as context . Each instance consists of a long input of lectures, multiple instructions(questions) and corresponding answers.
- **Coursera(L-Eval)** An et al. (2023) is a dataset built on Coursera website. Similar to TOFEL, Each instance consists of a long input of lectures, multiple instructions and corresponding answers.
- **SFiction(L-Eval)** An et al. (2023) is based on scientific fictions, in which context real-world principles don't apply. The questions contained in the documents ask the model to answer it based on either contextual information or real-world knowledge, as a way to test model hallucination.
- **LongFQA(L-Eval)** An et al. (2023) is an open-ended QA dataset on finance based on earnings call transcripts.
- **CUAD(L-Eval)**An et al. (2023) is drawn from the CUAD Hendrycks et al. (2021) dataset, which use legal contract as its context.
- **QuALITY** Pang et al. (2021) is a multiple-choice single-document QA dataset. It uses science fictions, magazine articles and nonfiction articles as input documents. The question is written by those that have read the full document. Each instance contains a document, a multiple-choice questions and corresponding answers. Notably, part of the questions are unanswerable.
- **NewsQA** Kwan et al. (2023) and **DuoRC** Kwan et al. (2023) are English QA datasets, constructed from news and movie plots, respectively.

Table 15: Text Reasoning/Classification Datasets. **CLS:** Classification. In the **Avg. Len:** average length, **Tok:** tokens; **W:** words. In the Instances column, **Doc:** documents, **Inst:** instructions. In the Metric column, **EM:** Exact Match. **Acc:** Accuracy.

| **Task** | **Name** | **Source** | **Instances** | **Avg Len** | **Metric** | **Lang.** |
|---|---|---|---|---|---|---|
| CLS/Reasoning | Long ListOps<br>Tay et al. (2020) | Generated | 100,003 | 3106 W | Acc | EN |
| Reasoning | ContractNLI<br>Koreeda & Manning (2021) | Legal | 10,319 | 2254 Tok | EM | EN |
| CLS | LSHT(LongBench)<br>Bai et al. (2023) | News | 200 | 22,337 W | Acc | ZH |
| Reasoning | GSM(16 shot)<br>An et al. (2023) | GSM8K<br>Cobbe et al. (2021) | 100 Doc, 100 Inst | 5557 Tok | Rouge-L, GPT-4, $\Delta L$ | EN |
| Reasoning | SenHallu(BAMBOO)<br>Dong et al. (2023b) | Paper | 200/200 | 3170/6357 Tok | Precision, Recall, F1 | EN |
| Reasoning | AbsHallu(BAMBOO)<br>Dong et al. (2023b) | Paper | 200/200 | 3314/6445 Tok | Precision, Recall, F1 | EN |
| CLS | MNDS News<br>Petukhova & Fachada (2023) | News | 10,917 | 637 W | Acc | EN |

- **C3** Kwan et al. (2023) is a multiple-choice QA dataset, based on second-language Chinese exams.
- **NQ** Kwiatkowski et al. (2019) is a QA dataset based on Wikipedia pages. Each instance(or example, as referred to in original paper) consists of a question, corresponding wikipedia page, a long answer and a short answer.

### 7.1.2 Text Summarization Task

A summarization dataset is a curated collection of texts and their corresponding summaries. They typically include diverse content, such as news articles, scientific papers, or conversational data, paired with concise and accurate summaries. The detailed statistics of the datasets for text summarization are listed in Table 14.

- **CNN/Dailymail** Nallapati et al. (2016), **GovReport** Huang et al. (2021), and **XSum** Narayan et al. (2018) include a document and its corresponding summary in each instance. CNN/Dailymail is based on over 300,000 news articles, GovReport is based on 14,466 long government reports, and XSum is based on BBC news.
- **MultiNews** Fabbri et al. (2019) is a multi-doc summary dataset, each instance consists of multiple news and a summary.
- **Loogle** Li et al. (2023a) is based on papers, WikiPedia, movie and TV scripts. Each long input text corresponds to mutiple question-answer-summary triad. In total there are 776 documents and 6,448 questions. Average document length is 19.367 words.
- **VCSUM** Wu et al. (2023a) is based on real-world Chinese meeting transcripts. Each meeting tarnscript corresponds to a headline, segmentation summaries and an overall summary. There're 239 meetings in total.
- **SummScreenFD** Chen et al. (2021b) is based on TV transcripts. Each instance consists of a TV transcript containing conversations, scenes and actor actions, and a summary(recapitulation, as referred to in original paper).
- **BigPatent** Sharma et al. (2019) is based on 1,341,362 patent documents. The highlight of this dataset is that important information is distributed evenly in patent documents, compared to other types of documents. Each instance contains a document and its corresponding summary(human written abstract).

- **SPACE** Angelidis et al. (2021) is based on reviews of 50 hotels. The highlight of the dataset is that the summaries are written in 6 different aspects, based on the hotel's review. Each hotel constructs an instance, containing the hotel's name, multiple reviews, summaries of different aspects and an overall summary.
- **SQuality** Wang et al. (2022a) is based on the same stories domain as QuALITY Pang et al. (2021) dataset. It's a query-based summarization dataset. Each instance contains a story, multiple summarization questions, and multiple summarizations that corresponds to each questions. There are 625 QA pairs in total.
- **CNNNews(M4LE)** Kwan et al. (2023) is based on CNN English news. Each instance of the dataset is paired with a multi-sentence summary.
- **CEPSUM(M4LE)** Kwan et al. (2023) is based on product information from Chinese e-commerce platform. Each instance contains a product description and corresponding summary.
- **LCSTS(M4LE)** Kwan et al. (2023) is a summarization dataset in Chinese. It consists of over 2 million posts from a Chinese micro-blogging website, each post is paired with a summary. M4LE selects instances whose article has over 30 words.
- **NCLS(M4LE)** Kwan et al. (2023) is a summarization dataset with articles and corresponding summaries in different language, which highlights model's cross-lingual ability. Original NCLS is constructed from CNNNews and LCSTS.
- **WikiHow(M4LE)** Kwan et al. (2023) is based on procedural descriptions on Wikipedia. Each article is entitled with a beginning of "How to...". Each paragraph of the article describes one step in the procedure, and corresponds to short summary. These summaries are then put together as the suymmary of the article.
- **News2016(M4LE)** Kwan et al. (2023) is based on ove 2 million news articles in Chinese. For each article, its title is used as golden summary. M4LE remove instances whose length is less than 200 words or over 800 words.
- **PubMed(M4LE)** Kwan et al. (2023) is based on medical papers. In M4LE, each paper's abstract is used as the summary of the paper.
- **BookSum(M4LE)** Kwan et al. (2023) is a dataset containing 405 English books, whose contents covers plays, novels and short stories. Each chapter of the content corresponds to a human-written summary.
- **CNewsum(M4LE)** Kwan et al. (2023) is based on 304,307 news articles in Chinese. Each article corresponds to a human-written summary.
- **CLTS+(M4LE)** Kwan et al. (2023) is based on CLTS Zhu (2020). CLTS contains over 180,000 Chinese articles, and CLTS+ uses back translation to make summaries more abstractive. M4LE selects part of these instances for benchmark.
- **Arxiv(M4LE)** Kwan et al. (2023) is based on papers collected from arXiv.org. For each paper, its abstract is used as golden summary.

### 7.1.3 Text Reasoning Task

A reasoning task involves the ability of a model to draw logical conclusions, make inferences, or solve problems based on given information. It requires understanding relationships, patterns, or rules within the data to arrive at accurate and coherent outcomes.Natural Language Inference(NLI) can be considered a subset of reasoning. It highlights model's ability to perform logical inference instructed by natural language.In an NLI task, the typical goal is to determine the relationship between two pieces of text: a premise and a hypothesis. The detailed statistics of the datasets for text reasoning are listed in Tab. 15.

- **Long Listops** Tay et al. (2020) is a mathematical reasoning dataset. It inputs an listop expression, instructing the model to perform calculation and output the exact numeric answer. A listop expression has a hierarchical structure that involves a set of simple mathematical operators. The final answer is a number in 0-9, described in original paper as "a ten-way classification task".

Table 16: Text Dataset-Retrieval. In the **Avg. Len:** average length, **W:** words. Particularly, LongEval, StreamingEval and TopicRet is more of a data generation method, which makes their length and instance number flexible, denoted by '-'. In the Metric column, **Acc:** Accuracy. **F1** Rajpurkar et al. (2016b) calculates unigram overlap between model output and answers after processing elments like white-spaces and stop-words.

| **Task** | **Name** | **Source** | **Instances** | **Avg Len** | **Metric** | **Lang.** |
|---|---|---|---|---|---|---|
| CLS/RET | TREC(LongBench)<br>Bai et al. (2023) | Web Question | 200 | 5177 W | Acc | EN |
| RET | LongEval<br>Li* et al. (2023) | Conversations | - | - | Acc | EN |
| RET | StreamingEval<br>Xiao et al. (2024c) | LongChat Li* et al. (2023) | - | - | Acc | EN |
| RET | TopicRet(L-Eval)<br>An et al. (2023) | LongChat Li* et al. (2023) | - | - | Acc | EN |
| RET | DRCD(M4LE)<br>Kwan et al. (2023) | Wiki | - | 3617 W | Acc | ZH |
| CLS+RET | MARC<br>Kwan et al. (2023) | E-Commerce | 2200 | 3543 W | F1 | EN, ZH |
| CLS+RET | Online Shopping(M4LE)<br>Kwan et al. (2023) | E-Commerce | 2200 | 3714 W | F1 | ZH |
| CLS+RET | MNDS News(M4LE)<br>Kwan et al. (2023) | MNDS News Petukhova & Fachada (2023) | - | 3805 W | Acc | EN |
| CLS+RET | THUCNews(M4LE)<br>Kwan et al. (2023) | News | - | 3721 W | Acc | ZH |

- **GSM** Cobbe et al. (2021) is a mathematcal reasoning dataset, which describes mathematical problems in natural language and ask the model to solve it.
- **ContractNLI** Koreeda & Manning (2021) uses contracts as context, and provides hypothesis, answer, and added evidence to each instance as well. The task requires model to judge the relationship between the hypothesis and context. Each instance contains 607 contracts, each contract has 17 annotated hypothesis and corresponding answers.
- **LSHT(LongBench)** Bai et al. (2023) is a Chinese classification dataset. It's based on Xinhua News. The model is asked to classify the input news articles into different categories.
- **SenHallu** Dong et al. (2023b) and **AbsHallu** Dong et al. (2023b)use content and a related hypothesis as model's input, and instruct the model to determine whether the hypothesis is true based on the content. The false hypothesis(hallucination, as referred to by original paper) is generated by GPT.
- **MNDS News** Petukhova & Fachada (2023) is a classification dataset consisting of 10.917 news articles. The news articles have 17 first level categories and 109 second-level categories.

### 7.1.4 Text Retrieval Task

A retrieval task in LLM benchmarks evaluates a model's ability to retrieve relevant information from a large collection of data based on a given query. It tests the model's understanding of the query, semantic matching, and efficiency in identifying the most relevant documents or pieces of information. The detailed statistics of the datasets for text retrieval are listed in Table 16.

- **LongChat** Li* et al. (2023) has two subtask dataset for retrieval. Coarse-grained Topic Retrieval dataset use a long document that talk about a number of different topics, and instrutct the model to retrieve the first topic of the document. Fine-grained Line retrieval, on the other hand, is more challenging, which

Table 17: Text Dataset-Generation.In the **Avg. Len:** average length, **Tok:** tokens; **W:** words. In the Instances column, **Doc:** documents, **Inst:** instructions. In the Metric column, **EM:** Exact Match. **Acc:** Accuracy.

| Task | Name | Source | Instances | Avg Len | Metric | Lang. |
|---|---|---|---|---|---|---|
| GEN | LCC<br>Guo et al. (2023) | Code | 360000 | 1337 W | EM, Edit Sim | Python/CSharp/Java |
| GEN | RepoBench-P(LongBench)<br>Bai et al. (2023) | Code | 500 | 4206 W | Edit Sim | Python/Java |
| GEN/RET | MultiDoc2Dial<br>Feng et al. (2021) | Doc2Dial<br>Feng et al. (2020) | 488 Doc,<br>4796 Dialogues | 4283 T | F1, EM,<br>SacreBLEU, Recall | EN |
| GEN | OpenReview(L-Eval)<br>An et al. (2023) | ASAP-Review<br>Yuan et al. (2021) | 20 Doc 60 Inst | 11,170 Tok | Rouge-L, GPT-4, $\Delta L$ | EN |
| GEN | ASAP-Review<br>Yuan et al. (2021) | Paper | 8877 Papers,<br>25,986 Reviews | 6782 W/Paper | Rouge-1/2/L, BERT | EN |
| GEN | ShowsPred<br>Dong et al. (2023b) | TV Shows | 100/100 | 2389/4860 Tok | Acc | EN |
| GEN | MeetingPred<br>Dong et al. (2023b) | Meeting | 100/100 | 3689/11578 Tok | Acc | EN |
| GEN-Code | PrivateEval<br>Dong et al. (2023b) | Code | 152/152 | 3149/6230 Tok | Pass@1 | EN, Python |
| GEN-Code | CodeU(L-Eval)<br>An et al. (2023) | Code | 90 Doc 10 Inst | 31,575 Tok | Rouge-L, GPT-4, $\Delta L$ | Python |

present the model with multiple lines that contain a diffrernt number and label, with similar line patterns. The model is asked to retrieve the number of a specific labeled line.Notably, such dataset can be easily constructed or generated, so it's easy to create an ultra long dataset of this type. Because the dataset is easily constructed by definition, the length of the dataset and the number of instances is indefinite.

- **StreamingEval**Xiao et al. (2024c) construct a line retrieval task based on LongChat, which makes a query in every 10 lines, with its answer about 20 lines above, so as to evaluate the streaming conversation scenario.
- **TopicRet** An et al. (2023) on the other hand, is based on the coarse-grained topic retrieval task, but ask about the second or third topic instead of the first one, so as to make the task more challenging.
- **DRCD(M4LE)** Kwan et al. (2023) is a reading comprehension dataset. In M4LE, DRCD is constructed into two subset, one(DRCD explicit) require model to return the articles' IDs related to a given topic, and another subset(DRCD semantic) requires the model to answer specific questions given multiple paragraphs.
- **MARC** Kwan et al. (2023) consists of bilingual(namely English and Chinese) reviews. The model is asked to identify all positive reviews and retrieve them.
- **Online Shopping(M4LE)** Kwan et al. (2023) is based on 60K product reviews on Chinese e-commerce platforms. Reviews are categorized into positive and negative.

### 7.1.5 Text Generation Task

Generation tasks require model to generate contents based on the given instructions and context. The detailed statistics of the datasets for text generation are listed in Table 17.

- **MultiDoc2Dial** Feng et al. (2021) gives model a dialogue history and all involved documents, and instruct model to generate the next turn of the dialogue.
- **OpenReview(L-Eval)** An et al. (2023), which is based on **ASAP-Review** Yuan et al. (2021), provides LLM with a paper and instruct it to generate a review.
- **ShowsPred** and **MeetingPred** Dong et al. (2023b) use dialogue history as input, and ask model to infer which role said the last turn of the conversation. Apart from natural language context, code generation is also an important implementation for LLMs.

Table 18: Text Dataset-Aggregation. In the **Avg. Len:** average length, **Tok:** tokens; **W:** words. In the Instances column, **Doc:** documents, **Inst:** instructions. In the Metric column, **Acc:** Accuracy. **ES:** Exponential Similarity, **CI:** Concordance Index

| Task | Name | Source | Instances | Avg Len | Metric | Lang. |
|---|---|---|---|---|---|---|
| AGG | SpaceDigest Shaham et al. (2023) | Reviews | 500 | 5481 W | ES | EN |
| AGG | BookSumSort Shaham et al. (2023) | Literature | 500 | 6840 W | CI | EN |
| AGG | PassageRetrieval-en Bai et al. (2023) | Wiki | 200 | 9289 W | Acc | EN |
| AGG | PassageRetrieval-zh Bai et al. (2023) | C4 Dataset | 200 | 6745 W | Acc | ZH |
| AGG | PassageCount Bai et al. (2023) | Wiki | 200 | 11,141 W | Acc | EN |
| AGG | ShowsReport(BAMBOO) Dong et al. (2023b) | TV Shows | 200/200 | 2992/6411 Tok | CI | EN |
| AGG | ReportSumSort(BAMBOO) Dong et al. (2023b) | Reports | 150/150 | 3753/8309 Tok | CI | EN |

- **LCC** Guo et al. (2023) gives model long code snippets as context, and instruct model to generate the following line of code.
- **RepoBench-P** Liu et al. (2023c) requires model to retrieve toe most relevant code snippets from a long input, and then generate code according to the instruction.
- **PrivateEval** Dong et al. (2023b) use API documents and a code snippet as input, and instruct the model to generate code acccordingly. Notably, to avoid data contamination caused by pre-training, the keywords in API documents are modified, making the document "private".
- **CodeU** Dong et al. (2023b) use the same practice of modifying keyword, only that it uses modified source code of public library, rather than API document, as an input.

### 7.1.6 Aggregation Task

Aggregation task involves understanding and aggregating information from the whole input to answer complex instructions, such as calculating the percentage of positive comments given a set of comments of different attitudes. The detailed statistics of the datasets for text aggregation are listed in Table 18.

- **SpaceDigest** Shaham et al. (2023) give the model a set of hotel reviews, and ask the model to output the percentage of positive reviews in the context.
- **BookSumSort** Shaham et al. (2023), **ReportSumSort** Dong et al. (2023b), and **ShowsSort** Dong et al. (2023b) use shuffled paragraphs from book summaries, TV transcripts or government reports as context, and ask the model to sort them in the correct order.
- **PassageCount** Bai et al. (2023) selects multiple passage, duplicates some of the paragraphs, and put all those paragraphs into an instance after shuffling. The model is then asked to determine how many documents are used to construct this instance.
- **PassageRetrieval** Bai et al. (2023), on the other hand, selects 30 wikipedia passages, and use GPT-3.5-Turbo to write a summary for one of them. Then these passages and the generated summary are used as the model input. The model is then instructed to tell which passage was the summary generated from.

### 7.1.7 Evaluation Metric for Text Datasets

General evaluation metrics used by text datasets mentioned above include **Exact Match** Rajpurkar et al. (2016a), **Partial Match**, **Accuracy**, **Recall**, **Precision**, **F1**, **BLEU** Papineni et al. (2002), **SacreBLEU** Post (2018), **Rouge** Lin (2004), **METEOR** Denkowski & Lavie (2011), **BERT** Zhang et al. (2020), **Edit Similarity**, **Pass@k** Chen et al. (2021a) , **Exponential Similarity**, **Concordance Index**, **Mean Reciprocal Rank**. In addition to general evaluation metrics, some more specific metrics are used in particular benchmarks. For datasets from L-Eval An et al. (2023), the **GPT-4** metric means the win-rate against Turbo-16K, judged by GPT-4. $\Delta L$ is the length difference between answer length and ground truth. For LooGLE Li et al. (2023a), it utilizes **GPT-4** for its QA and summarization task, using it for answer's semantic judgment.

- **Exact Match (EM)** Rajpurkar et al. (2016a) is a metric used to evaluate the accuracy of models in tasks like question answering or text generation. It measures the percentage of predictions that exactly match the ground truth answer, considering both the content and format.
- **Partial Match (PM)** metric evaluates the similarity between a model's output and the reference by allowing partial credit for partially correct answers. Unlike strict metrics like Exact Match (EM), PM accounts for overlaps or shared elements, such as keywords or phrases, making it more flexible in assessing performance.
- **Accuracy** is a metric used to evaluate the overall performance of a model by measuring the proportion of correctly predicted instances (both positive and negative) out of the total instances.
- **Recall** is a metric used to evaluate a model's ability to retrieve all relevant instances in a dataset. It is calculated as the ratio of correctly retrieved relevant items to the total number of relevant items, emphasizing completeness.
- **Precision** is a metric used to evaluate the accuracy of a model by measuring the proportion of correctly predicted positive instances out of all predicted positive instances.
- **F1** is a performance measure that combines Precision and Recall into a single score using their harmonic mean. It provides a balanced evaluation, especially useful in datasets with imbalanced classes, by considering both false positives and false negatives.
- **BLEU** Papineni et al. (2002), is a widely used metric for evaluating the quality of machine-generated text, especially in machine translation. It works by comparing n-grams in the generated output with reference texts to measure overlap, while applying penalties for overly short outputs to ensure fluency.
- **SacreBLEU** Post (2018) is a standardized version of the BLEU metric used to evaluate machine translation quality. It simplifies BLEU's implementation by fixing preprocessing steps like reference handling to ensure consistent and reproducible results across different systems.
- **Rouge** Lin (2004) and its variants measure model's performance by calculating overlap between model output and reference answer with unigram(**Rouge-1**), bigram(**Rouge-2**), LCS(**Rouge-L**), etc. **Gold Rouge-1** in VCSUM dataset refers to using high-quality reference summaries (gold standards) for evaluation, ensuring reliable and meaningful comparisons.
- **METEOR** Denkowski & Lavie (2011) (Metric for Evaluation of Translation with Explicit ORdering) is a text evaluation metric designed to assess the quality of machine translation.
- **BERT** Zhang et al. (2020) metric, often referred to as BERTScore, is a text evaluation metric that uses contextual embeddings from the BERT model to compare similarity between generated and reference texts.
- **Edit Similarity** is a metric that measures the similarity between two text sequences based on the minimum number of edit operations required to transform one sequence into another. It is derived from the concept of edit distance such as Levenshtein distance.
- **Pass@k** Chen et al. (2021a) evaluates the performance of a model by measuring the percentage that at least one of the top k generated outputs contains a correct solution. In datasets we surveyed, only **Pass@1** is used.
- **Exponential Similarity** is a metric that measures the similarity between two items by exponentially weighting their differences, giving more importance to smaller discrepancies.
- **Concordance Index** is a metric used to evaluate the predictive accuracy of models, particularly in survival analysis or ranking tasks.

Table 19: Multimodal Dataset. Specfically, for data type, **Img**: Image; **T**: text; **V:** Video. For task abbreviation, **Conv**: conversation task; **Desc**: description task; **Reas:** reasoning task; **Perc:** perception task; **Pred:** prediction; **NTH:** needle in the haystack; **SUMM:** summary. For instance and average column, **Q**: questions; **W**: words; **s**: seconds. For example, **54 Img, 150 Q** denote that there are 54 images with 150 questions.

| Tasks | Name | Data | Source | Instance | Average | Metric | Language |
|---|---|---|---|---|---|---|---|
| Conv, Desc, Reas | LLaVA-Bench<br>Liu et al. (2023b) | Img, T | COCO, In-The-Wild | 54 Img, 150 Q | 1 Img, 59.9 W | Relative Score | EN |
| Perc, Reas | MMBench<br>Yuan Liu et al. (2023) | Img, T | Internet | 2948 Q | 1 Img, 114.5 W | Acc | EN/CN |
| Pred, Count,<br>NIH, Retrieval | MileBench<br>Song et al. (2024) | Img, T | Public,<br>self-building | 6440 Q | 15.2 Img, 422.3 W | Acc, ROUGE-L | EN |
| Reas, NIH, SUMM,<br>Desc, Order, Count | MLVU<br>Zhou et al. (2024a) | V, T | Public,<br>self-collection | 1334 V, 2593 Q | 704.6s V, 39.7 W | M-Avg, G-Avg | EN |
| Reas, Retrieval | LongVideoBench<br>Wu et al. (2024b) | V, T | web-collected | 3763 V, 6678 Q | 730.5s V, 49.5 W | Acc | EN |
| Perc, Recognition, Reas | Video-MME<br>Fu et al. (2024a) | V, T | YouTube | 900 V, 2700 Q | 1017.9s V | Acc | EN |
| Desc, Reas | NExT-QA<br>Xiao et al. (2021) | V, T | YouTube,<br>TV Show, Public | 1000 V, 47962 Q | 44s V, 25.5 W | Acc, WUPS | EN |
| Perc, Count, Reas | MVBench<br>Li et al. (2023b) | V, T | Public | 4000 Q | 16.7s V, 31.3 W | Acc | EN |
| Desc | MSVD-QA<br>Xu et al. (2017) | V, T | MSVD | 1970 V, 50505 Q | 10s V | Acc | EN |
| Desc | MSRVYY-QA<br>Xu et al. (2017) | V, T | MSRVTT | 10000 V, 243690 Q | 15s V | Acc | EN |

- **Mean Reciprocal Rank (MRR)** is an evaluation metric commonly used in information retrieval and recommendation systems to measure the quality of ranked results. It calculates the reciprocal of the rank of the first relevant item in a result list and averages it across all queries.

## 7.2 Multimodal Datasets and Evaluation Metric

### 7.2.1 Multimodal Datasets

Multimodal datasets have emerged to address the need for a comprehensive understanding of the complex real world by integrating diverse data types such as text, images, audio, and video. These datasets drive advancements in AI, particularly in machine learning and deep learning, by offering rich and diverse data to train more robust and versatile models. We analyze the multimodal benchmarks listed in Table 19, highlighting their distinct focuses. Each benchmark is built upon one or more multimodal datasets, involving their collection, processing, and the use of specific validation metrics. Below, we provide a detailed introduction and description of each multimodal benchmark.

- **LLaVA-Bench** Liu et al. (2023b) The benchmark is structured around image-ground-truth textual description-question-answer triplets, segmented across COCO and In-The-Wild datasets. It assesses a model's proficiency in multimodal instruction adherence and visual reasoning. By employing a suite of tasks and metrics, it quantifies the model's ability to comprehend and act on visual-language directives, articulate comprehensive descriptions, and engage in intricate reasoning processes.

- **MMBench** Yuan Liu et al. (2023) This benchmark serves as a bilingual multimodal benchmark, facilitating a comparative analysis of VLM performance across English and Chinese linguistic contexts. It distinctively assesses multimodal models using a hierarchical taxonomy of abilities, stringent quality assurance measures, and a dual-language evaluation framework. Unlike other benchmarks, MMBench Yuan Liu et al. (2023) incorporates the CircularEval strategy for comprehensive evaluation and utilizes LLMs for precise extraction of choices, setting it apart from its counterparts.

- **MileBench** Song et al. (2024) evaluates the multi-modal long-context capabilities of LLMs, including both diagnostic and realistic evaluation sets. It emphasizes long-context and multi-image tasks. This unique focus allows it to capture the complexity and diversity of real-world multimodal challenges, setting it apart from existing benchmarks. The dataset in MileBench Song et al. (2024) is characterized by its inclusion of long texts integrated with multiple images, reflecting real-world scenarios where context is key. It contains a diverse range of tasks that require both comprehension and generation.

- **MLVU** Zhou et al. (2024a) is a holistic benchmark, designed to gauge the capabilities of multi-modal LLMs in comprehending video content, transcends the constraints of its predecessors by significantly increasing video durations, encompassing diverse video genres, and crafting a spectrum of assessment tasks. This benchmark offers an extensive array of tasks and video genres to evaluate the comprehensive competencies of MLLMs. It highlights the substantial potential for enhancement in current methodologies and emphasizes the critical factors of context length, image comprehension quality, and the selection of LLM architecture for future progress.

- **LongVideoBench** Wu et al. (2024b) This benchmark offers an extensive benchmarking framework aimed at assessing the capacity of large multimodal models (LMMs) to comprehend lengthy videos with subtitles, extending up to an hour. It places a strong focus on the retrieval and reasoning capabilities over extended, interwoven video and language data streams, tackling the challenge of single-frame bias and underscoring its proficiency in evaluating multimodal comprehension in long contexts.

- **Video-MME** Fu et al. (2024a) A benchmark for comprehensive evaluation, it assesses the proficiency of Multi-modal Large Language Models (MLLMs) in analyzing videos. This dataset comprises a wide array of 900 videos spanning diverse domains and subfields, ensuring extensive scenario coverage. It encompasses videos with lengths ranging from 11 seconds to 1 hour to gauge model flexibility across various time frames. Furthermore, it incorporates various data modalities, including subtitles and audio tracks, to evaluate the comprehensive competencies of MLLMs. The benchmark aims to test the models' capacity for sequential visual data comprehension, with an emphasis on temporal reasoning and the processing of multi-modal inputs.

- **NExT-QA** Xiao et al. (2021) Advancing video comprehension from mere description to explanation of causal, temporal, and descriptive actions, a video question answering (VideoQA) benchmark has been established. This benchmark boasts a dataset with 5,440 videos and approximately 52K manually annotated question-answer pairs, sorted into causal, temporal, and descriptive categories. It poses a challenge to QA models to engage in reasoning about causal and temporal actions and to decipher complex object interactions within daily activities. Distinguished from other video benchmarks, this benchmark specifically focuses on causal and temporal action reasoning within realistic videos that are rich in object interactions. It stands as one of the largest manually annotated VideoQA datasets, offering support for both multiple-choice and open-ended questions, and includes a variety of videos that mirror real-life scenarios.

- **MVBench** Li et al. (2023b) Featuring a substantial dataset, the benchmark comprises 200 multiple-choice question-answer (QA) pairs for each of the 20 temporal understanding tasks, amassing a total of 4,000 QA pairs. It draws from a variety of videos across 11 public datasets, spanning diverse domains and scenes, thereby testing models' abilities to comprehend temporal sequences. The benchmark automates the generation of multiple-choice QA pairs from existing video annotations, minimizing human involvement and ensuring a fair evaluation process.

- **MSVD-QA** Xu et al. (2017) The MSVD dataset is a collection of 1,970 video clips with descriptive captions, initially for video captioning. It features diverse real-world scenarios and assesses multimodal learning models' capabilities in understanding video content and generating natural language descriptions.

- **MSRVTT-QA** Xu et al. (2017) The MSR-VTT dataset comprises 10,000 video clips with 20 human-transcribed sentences each, focusing on connecting video content with language descriptions. It evaluates multimodal learning models' ability to comprehend video information and translate it into coherent captions, testing their video understanding and language generation skills in a more complex and diverse environment.

### 7.2.2 Evaluation Metric for Multimodal Datasets

The evaluation metrics for multimodal datasets include **Relative Score**, **Accuracy**, **ROUGE-L**, **M-Avg**, **G-Avg**, **WUPS**. Several common metrics, including **Accuracy**, **ROUHE-L**, have been introduced in Sec. 7.1.7. Here, we only introduce the special metrics of multimodal datasets, which include **Relativa Score**, **M-Avg**, **G-Avg**, **WUPS** as follows:

- **Relative Score** This metric is used in LLaVA-Bench to evaluate the performance of multimodal models by comparing their outputs to a reference model, typically text-based GPT-4. It is calculated as the percentage ratio of the candidate model's score to the reference model's score, based on dimensions such as helpfulness, relevance, accuracy, and level of detail.

- **M-Avg** Multiple-Choice Average is calculated as the mean accuracy across all multiple-choice tasks in the MLVU benchmark. The accuracy for each task is determined by the proportion of correctly predicted answers compared to the total number of questions within that task.

- **G-Axg** Generation Average s calculated as the mean score across all generation tasks in the MLVU benchmark. Each task is evaluated on multiple dimensions (e.g., Accuracy, Relevance, Completeness, and Reliability) using GPT-4, with scores ranging from 1 to 5. The overall score for each task is the average of these dimensions, and G-Avg is the mean of these task-level scores.

- **WUPS** K et al. (2012) Wu-Palmer Similarity measures the semantic similarity between two words based on their positions in a taxonomy (e.g., WordNet). It calculates how closely related two words are by considering their least common ancestor (LCS).

## 8 Conclusion

Advancements in LLMs have driven significant progress on various fields, but their high computational and memory demands during inference pose challenges, especially for long-context and real-time applications. KV cache management offers an effective solution by optimizing memory, reducing redundant computation, and improving performance. This survey reviews KV cache management strategies across token-level, model-level, and system-level optimizations. Token-level optimizations focus on fine-grained control of KV cache through selection, budget allocation, merging, quantization, and low-rank decomposition, enabling efficient resource allocation without altering model architectures. Model-level optimizations leverage architectural innovations, such as attention grouping and non-transformer designs, to enhance the efficiency of KV reuse. System-level optimizations further complement these efforts by employing advanced memory management, scheduling techniques, and hardware-aware designs to optimize resource utilization across diverse computing environments.

Despite the progress made, substantial opportunities remain for future exploration. Key areas include the development of real-time, task-specific budget allocation strategies, dynamic workload handling, advanced distributed coordination for KV cache in multi-node systems, and hardware-aware innovations to leverage emerging architectures like computational storage and processing-in-memory. Additionally, integrating reinforcement learning and adaptive algorithms could enable more intelligent and responsive KV cache management, further enhancing LLM efficiency across diverse deployment scenarios.

## References

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: KV cache reduction through key tokens selection for efficient generative inference. In *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13-16, 2024*. mlsys.org, 2024.

Abien Fred Agarap. Deep learning using rectified linear units. *arXiv preprint arXiv:1803.08375*, 2018.

Saurabh Agarwal, Bilge Acun, Basil Hosmer, Mostafa Elhoushi, Yejin Lee, Shivaram Venkataraman, Dimitris Papailiopoulos, and Carole-Jean Wu. CHAI: Clustered Head Attention for Efficient LLM Inference, April 2024. URL http://arxiv.org/abs/2403.08058.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints, December 2023. URL http://arxiv.org/abs/2305.13245.

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. L-Eval: Instituting Standardized Evaluation for Long Context Language Models, 2023. URL https://arxiv.org/abs/2307.11088.

Stefanos Angelidis, Reinald Kim Amplayo, Yoshihiko Suhara, Xiaolan Wang, and Mirella Lapata. Extractive Opinion Summarization in Quantized Transformer Spaces. *Transactions of the Association for Computational Linguistics*, 9:277–293, March 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00366. URL https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00366/98621/Extractive-Opinion-Summarization-in-Quantized.

Anonymous. CAKE: Cascading and adaptive KV cache eviction with layer preferences. In *Submitted to The Thirteenth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=EQgEMAD4kv. under review.

Anonymous. DynamicKV: Task-aware adaptive KV cache compression for long context LLMs. In *Submitted to The Thirteenth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=uHkfU4TaPh. under review.

Anonymous. Identify critical KV cache in LLM inference from an output perturbation perspective. In *Submitted to The Thirteenth International Conference on Learning Representations*, 2024c. URL https://openreview.net/forum?id=lRTDMGYCpy. under review.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

Ben Athiwaratkun, Sujan Kumar Gonugondla, Sanjay Krishna Gouda, Haifeng Qian, Hantian Ding, Qing Sun, Jun Wang, Jiacheng Guo, Liangfu Chen, Parminder Bhatia, Ramesh Nallapati, Sudipta Sengupta, and Bing Xiang. Bifurcated attention: Accelerating massively parallel decoding with shared prefixes in llms, 2024. URL https://arxiv.org/abs/2403.08845.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A Bilingual, Multitask Benchmark for Long Context Understanding, 2023. URL https://arxiv.org/abs/2308.14508.

William Berrios, Gautam Mittal, Tristan Thrush, Douwe Kiela, and Amanpreet Singh. Towards language models that can see: Computer vision through the LENS of natural language. *CoRR*, abs/2306.16410, 2023. doi: 10.48550/ARXIV.2306.16410. URL https://doi.org/10.48550/arXiv.2306.16410.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Understanding and overcoming the challenges of efficient transformer quantization. *arXiv preprint arXiv:2109.12948*, 2021.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. *Advances in Neural Information Processing Systems*, 36:75067–75096, 2023.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan Kelly. Reducing Transformer Key-Value Cache Size with Cross-Layer Attention, May 2024. URL `http://arxiv.org/abs/2405.12981`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic KV cache compression based on pyramidal information funneling. *CoRR*, abs/2406.02069, 2024.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S Abdelfattah, and Kai-Chiang Wu. Palu: Compressing kv-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. Sepllm: Accelerate large language models by compressing one segment into one separator. *arXiv preprint arXiv:2412.12094*, 2024a.

Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. Driving with llms: Fusing object-level vector modality for explainable autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14093–14100. IEEE, 2024b.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021a. URL `https://arxiv.org/abs/2107.03374`.

Mengzhao Chen, Yi Liu, Jiahao Wang, Yi Bin, Wenqi Shao, and Ping Luo. Prefixquant: Static quantization beats dynamic through prefixed outliers in llms. *arXiv preprint arXiv:2410.05265*, 2024c.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. SummScreen: A Dataset for Abstractive Screenplay Summarization, 2021b. URL `https://arxiv.org/abs/2104.07091`.

Yilong Chen, Guoxia Wang, Junyuan Shang, Shiyao Cui, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, Yu Sun, Dianhai Yu, and Hua Wu. NACL: A general and effective KV cache eviction framework for llms at inference time. *CoRR*, abs/2408.03675, 2024d.

Yilong Chen, Linhao Zhang, Junyuan Shang, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, and Yu Sun. DHA: Learning Decoupled-Head Attention from Transformer Checkpoints via Adaptive Heads Fusion, June 2024e. URL `http://arxiv.org/abs/2406.06567`.

Yuang Chen, Cheng Zhang, Xitong Gao, Robert D. Mullins, George Anthony Constantinides, and Yiren Zhao. Optimised Grouped-Query Attention Mechanism for Transformers. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*, July 2024f. URL `https://openreview.net/forum?id=13MMghY6Kh`.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Léon Bottou, Zhihao Jia, and Beidi Chen. Magicpig: LSH sampling for efficient LLM generation. *CoRR*, abs/2410.16179, 2024g.

Jian Cheng, Jiaxiang Wu, Cong Leng, Yuhang Wang, and Qinghao Hu. Quantized cnn: A unified approach to accelerate and compress convolutional networks. *IEEE transactions on neural networks and learning systems*, 29(10):4730–4743, 2017.

Sai Sena Chinnakonduru and Astarag Mohapatra. Weighted Grouped Query Attention in Transformers, July 2024. URL http://arxiv.org/abs/2407.10855.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2020.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168.

Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, Yang Zhou, Kaizhao Liang, Jintai Chen, Juanwu Lu, Zichong Yang, Kuei-Da Liao, Tianren Gao, Erlong Li, Kun Tang, Zhipeng Cao, Tong Zhou, Ao Liu, Xinrui Yan, Shuqi Mei, Jianguo Cao, Ziran Wang, and Chao Zheng. A survey on multimodal large language models for autonomous driving. In *IEEE/CVF Winter Conference on Applications of Computer Vision Workshops, WACVW 2024 - Workshops, Waikoloa, HI, USA, January 1-6, 2024*, pp. 958–979. IEEE, 2024. doi: 10.1109/WACVW60836.2024.00106. URL https://doi.org/10.1109/WACVW60836.2024.00106.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 1280–1297. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.70. URL https://doi.org/10.18653/v1/2024.acl-long.70.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers, 2021. URL https://arxiv.org/abs/2105.03011.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model, June 2024. URL `http://arxiv.org/abs/2405.04434`.

Michael Denkowski and Alon Lavie. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar F. Zaidan (eds.), *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pp. 85–91, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL `https://aclanthology.org/W11-2107`.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Tianyu Ding, Tianyi Chen, Haidong Zhu, Jiachen Jiang, Yiqi Zhong, Jinxin Zhou, Guangzhi Wang, Zhihui Zhu, Ilya Zharkov, and Luming Liang. The efficiency spectrum of large language models: An algorithmic survey. *arXiv preprint arXiv:2312.00678*, 2023.

Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. Get more with less: Synthesizing recurrence with kv cache compression for efficient llm inference. *arXiv preprint arXiv:2402.09398*, 2024a.

Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. Qaq: Quality adaptive quantization for llm kv cache. *arXiv preprint arXiv:2403.04643*, 2024b.

Wei Dong and Ke Yi. Residual sensitivity for differentially private multi-way joins. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 432–444, 2021.

Wei Dong, Qiyao Luo, and Ke Yi. Continual observation under user-level differential privacy. In *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 2190–2207. IEEE, 2023a.

Zican Dong, Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. Bamboo: A comprehensive benchmark for evaluating long text modeling capacities of large language models. *arXiv preprint arXiv:2309.13345*, 2023b.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. GLM: general language model pretraining with autoregressive blank infilling. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 320–335. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.26. URL `https://doi.org/10.18653/v1/2022.acl-long.26`.

Haojie Duanmu, Zhihang Yuan, Xiuhong Li, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. Skvq: Sliding-window key and value cache quantization for large language models. *arXiv preprint arXiv:2405.06219*, 2024.

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

Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. Multi-News: a Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model, 2019. URL https://arxiv.org/abs/1906.01749.

Song Feng, Hui Wan, Chulaka Gunasekara, Siva Sankalp Patel, Sachindra Joshi, and Luis A. Lastras. doc2dial: A Goal-Oriented Document-Grounded Dialogue Dataset, 2020. URL https://arxiv.org/abs/2011.06623.

Song Feng, Siva Sankalp Patel, Hui Wan, and Sachindra Joshi. MultiDoc2Dial: Modeling Dialogues Grounded in Multiple Documents. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6162–6176, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.498. URL https://aclanthology.org/2021.emnlp-main.498.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. *CoRR*, abs/2407.11550, 2024.

Zafeirios Fountas, Martin Benfeghoul, Adnan Oomerjee, Fenia Christopoulou, Gerasimos Lampouras, Haitham Bou-Ammar, and Jun Wang. Human-like episodic memory for infinite context llms. *CoRR*, abs/2407.09450, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Chaoyou Fu, Yuhan Dai, Yondong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*, 2024a.

Daocheng Fu, Xin Li, Licheng Wen, Min Dou, Pinlong Cai, Botian Shi, and Yu Qiao. Drive like a human: Rethinking autonomous driving with large language models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 910–919, 2024b.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. *CoRR*, abs/2410.19258, 2024c.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-efficient large language model serving for multi-turn conversations with cachedattention, 2024a. URL https://arxiv.org/abs/2403.19708.

Shiwei Gao, Youmin Chen, and Jiwu Shu. Fast state restoration in LLM serving with hcache. *CoRR*, abs/2410.05004, 2024b. doi: 10.48550/ARXIV.2410.05004. URL https://doi.org/10.48550/arXiv.2410.05004.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for llms. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Daniel Goldstein, Fares Obeid, Eric Alcaide, Guangyu Song, and Eugene Cheah. GoldFinch: High Performance RWKV/Transformer Hybrid with Linear Pre-Fill and Extreme KV-Cache Compression, July 2024. URL http://arxiv.org/abs/2407.12077.

Giovani Gracioli, Ahmed Alhammad, Renato Mancuso, Antônio Augusto Fröhlich, and Rodolfo Pellizzoni. A survey on cache management mechanisms for real-time embedded systems. *ACM Computing Surveys (CSUR)*, 48(2):1–36, 2015.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, May 2024. URL http://arxiv.org/abs/2312.00752.

Albert Gu, Isys Johnson, Karan Goel, Khaled Kamal Saab, Tri Dao, Atri Rudra, and Christopher Re. Combining Recurrent, Convolutional, and Continuous-time Models with Linear State Space Layers. In *Advances in Neural Information Processing Systems*, November 2021. URL `https://openreview.net/forum?id=yWd42CWN3c`.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the Parameterization and Initialization of Diagonal State Space Models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. arXiv, 2022.

Ozgur Guldogan, Jackson Kunde, Kangwook Lee, and Ramtin Pedarsani. Multi-bin batching for increasing LLM inference throughput, 2024. URL `https://openreview.net/forum?id=WVmarX0RNd`.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. LongCoder: A Long-Range Pre-trained Language Model for Code Completion, 2023. URL `https://arxiv.org/abs/2306.14893`.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 2023.

Benjamin D Haeffele and René Vidal. Global optimality in tensor factorization, deep learning, and beyond. *arXiv preprint arXiv:1506.07540*, 2015.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 3991–4008. Association for Computational Linguistics, 2024.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid structural state-space models. *arXiv preprint arXiv:2209.12951*, 2022.

Jiaao He and Jidong Zhai. Fastdecode: High-throughput gpu-efficient llm serving using heterogeneous pipelines, 2024. URL `https://arxiv.org/abs/2403.11421`.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. DuReader: a Chinese Machine Reading Comprehension Dataset from Real-world Applications, 2017. URL `https://arxiv.org/abs/1711.05073`.

Yefei He, Feng Chen, Jing Liu, Wenqi Shao, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. Zipvl: Efficient large vision-language models with dynamic token sparsification and kv cache compression. *arXiv preprint arXiv:2410.08584*, 2024a.

Yefei He, Luoming Zhang, Weijia Wu, Jing Liu, Hong Zhou, and Bohan Zhuang. Zipcache: Accurate and efficient kv cache quantization with salient token identification. *arXiv preprint arXiv:2405.14256*, 2024b.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. CUAD: An Expert-Annotated NLP Dataset for Legal Contract Review, 2021. URL `https://arxiv.org/abs/2103.06268`.

Namgyu Ho, Sangmin Bae, Taehyeon Kim, Hyunjik Jo, Yireun Kim, Tal Schuster, Adam Fisch, James Thorne, and Se-Young Yun. Block Transformer: Global-to-Local Language Modeling for Fast Inference, November 2024. URL `http://arxiv.org/abs/2406.02657`.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL `https://www.aclweb.org/anthology/2020.coling-main.580`.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Monishwaran Maheswaran, June Paik, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. Squeezed attention: Accelerating long context length llm inference, 2024a. URL https://arxiv.org/abs/2411.09688.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024b.

Cunchen Hu, Heyang Huang, Junhao Hu, Jiang Xu, Xusheng Chen, Tao Xie, Chenxi Wang, Sa Wang, Yungang Bao, Ninghui Sun, and Yizhou Shan. Memserve: Context caching for disaggregated llm serving with elastic memory pool, 2024. URL https://arxiv.org/abs/2406.17565.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer Quality in Linear Time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9099–9117. PMLR, 2022. URL https://proceedings.mlr.press/v162/hua22a.html.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient Attentions for Long Document Summarization, April 2021. URL http://arxiv.org/abs/2104.02112. arXiv:2104.02112.

Farnoosh Javadi, Walid Ahmed, Habib Hajimolahoseini, Foozhan Ataiefard, Mohammad Hassanpour, Saina Asani, Austin Wen, Omar Mohamed Awad, Kangling Liu, and Yang Liu. GQKVA: Efficient Pre-training of Transformers by Grouping Queries, Keys, and Values, December 2023. URL http://arxiv.org/abs/2311.03426.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023. doi: 10.48550/ARXIV.2310.06825. URL https://doi.org/10.48550/arXiv.2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *CoRR*, abs/2401.04088, 2024a. doi: 10.48550/ARXIV.2401.04088. URL https://doi.org/10.48550/arXiv.2401.04088.

Chaoyi Jiang, Lei Gao, Hossein Entezari Zarch, and Murali Annavaram. Efficient llm inference with i/o-aware partial kv cache recomputation, 2024b. URL https://arxiv.org/abs/2411.17089.

Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. *Advances in Neural Information Processing Systems*, 31, 2018.

Xuanlin Jiang, Yang Zhou, Shiyi Cao, Ion Stoica, and Minlan Yu. Neo: Saving gpu memory crisis with cpu offloading for online llm inference, 2024c. URL https://arxiv.org/abs/2411.01142.

Ming Jin, Qingsong Wen, Yuxuan Liang, Chaoli Zhang, Siqiao Xue, Xue Wang, James Zhang, Yi Wang, Haifeng Chen, Xiaoli Li, et al. Large models for time series and spatio-temporal data: A survey and outlook. *arXiv preprint arXiv:2310.10196*, 2023.

Shuowei Jin, Xueshen Liu, Qingzhao Zhang, and Z. Morley Mao. Compute or load kv cache? why not both?, 2024. URL https://arxiv.org/abs/2410.03065.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension, 2017. URL `https://arxiv.org/abs/1705.03551`.

Vinay Joshi, Prashant Laddha, Shambhavi Sinha, Om Ji Omer, and Sreenivas Subramoney. QCQA: Quality and Capacity-aware grouped Query Attention, June 2024. URL `http://arxiv.org/abs/2406.10247`.

Jordan Juravsky, Bradley Brown, Ryan Ehrlich, Daniel Y. Fu, Christopher Ré, and Azalia Mirhoseini. Hydragen: High-throughput llm inference with shared prefixes, 2024. URL `https://arxiv.org/abs/2402.05099`.

Manjula Shenoy. K, K. C. Shet, and U. Dinesh Acharya. A new similarity measure for taxonomy based on edge counting, 2012. URL `https://arxiv.org/abs/1211.4709`.

Christoforos Kachris. A survey on hardware accelerators for large language models. *arXiv preprint arXiv:2401.09890*, 2024.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. Gear: An efficient kv cache compression recipefor near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527*, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Zohaib Khan, Muhammad Khaquan, Omer Tafveez, Burhanuddin Samiwala, and Agha Ali Raza. Beyond Uniform Query Distribution: Key-Driven Grouped Query Attention, August 2024. URL `http://arxiv.org/abs/2408.08454`.

Jang-Hyun Kim, Junyoung Yeom, Sangdoo Yun, and Hyun Oh Song. Compressed context memory for online language model interaction. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL `https://openreview.net/forum?id=64kSvC4iPg`.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. Speculative decoding with big little decoder. *Advances in Neural Information Processing Systems*, 36, 2024b.

Yuta Koreeda and Christopher Manning. ContractNLI: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 1907–1919, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL `https://aclanthology.org/2021.findings-emnlp.164`.

Volodymyr Kuleshov, Arun Chaganty, and Percy Liang. Tensor factorization via matrix factorization. In *Artificial Intelligence and Statistics*, pp. 507–516. PMLR, 2015.

Kwan, Wai-Chung, Zeng, Xingshan, Wang, Yufei, Sun, Yusen, Liand Liangyou, Shang, Lifeng, Liu, Qun, Wong, and Kam-Fai. M4le: A multi-ability multi-range multi-task multi-domain long-context evaluation benchmark for large language models. *arXiv preprint arXiv:2310.19240*, 2023.

Alicja Kwasniewska, Maciej Szankin, Mateusz Ozga, Jason Wolfe, Arun Das, Adam Zajac, Jacek Ruminski, and Paul Rad. Deep learning optimization for edge devices: Analysis of training quantization parameters. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pp. 96–101. IEEE, 2019.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew

Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:453–466, November 2019. ISSN 2307-387X. doi: 10.1162/tacl_a_00276. URL https://direct.mit.edu/tacl/article/43518.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pp. 611–626. ACM, 2023.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. Infinigen: Efficient generative inference of large language models with dynamic kv cache management, 2024. URL https://arxiv.org/abs/2406.19707.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Dacheng Li*, Rulin Shao*, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph E. Gonzalez, Ion Stoicaand Xuezhe Ma, , and Hao Zhang. How long can open-source llms truly promise on context length?, June 2023. URL https://lmsys.org/blog/2023-06-29-longchat.

Haoyang Li and Lei Chen. Cache-based GNN system for dynamic graphs. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pp. 937–946. ACM, 2021. doi: 10.1145/3459637.3482237. URL https://doi.org/10.1145/3459637.3482237.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. LooGLE: Can Long-Context Language Models Understand Long Contexts?, 2023a. URL https://arxiv.org/abs/2311.04939.

Kunchang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023b.

Yiming Li, Yanyan Shen, Lei Chen, and Mingxuan Yuan. Orca: Scalable temporal graph neural network training with theoretical guarantees. *Proc. ACM Manag. Data*, 1(1):52:1–52:27, 2023c. doi: 10.1145/3588737. URL https://doi.org/10.1145/3588737.

Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, et al. Scbench: A kv cache-centric analysis of long-context methods. *arXiv preprint arXiv:2412.10319*, 2024a.

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

Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. Prompt compression for large language models: A survey. *arXiv preprint arXiv:2410.12388*, 2024c.

Bingli Liao and Danilo Vasconcellos Vargas. Beyond KV Caching: Shared Attention for Efficient LLMs, July 2024. URL http://arxiv.org/abs/2407.12866.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, Tianqi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. MatryoshkaKV: Adaptive KV Compression via Trainable Orthogonal Projection, October 2024a. URL http://arxiv.org/abs/2410.14731.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International conference on machine learning*, pp. 2849–2858. PMLR, 2016.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: activation-aware weight quantization for on-device LLM compression and acceleration. In Phillip B. Gibbons, Gennady Pekhimenko, and Christopher De Sa (eds.), *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13-16, 2024*. mlsys.org, 2024c. URL `https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html`.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4A8KV4 quantization and system co-design for efficient LLM serving. *CoRR*, abs/2405.04532, 2024d. doi: 10.48550/ARXIV.2405.04532. URL `https://doi.org/10.48550/arXiv.2405.04532`.

Zhiqi Lin, Cheng Li, Youshan Miao, Yunxin Liu, and Yinlong Xu. Pagraph: Scaling gnn training on large graphs via computation-aware caching. In *Proceedings of the 11th ACM Symposium on Cloud Computing*, pp. 401–415, 2020.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: KV cache compression in depth dimension for large language models. *CoRR*, abs/2405.14366, 2024a. doi: 10.48550/ARXIV.2405.14366. URL `https://doi.org/10.48550/arXiv.2405.14366`.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. Retrievalattention: Accelerating long-context LLM inference via vector retrieval. *CoRR*, abs/2409.10516, 2024b.

Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression. *arXiv preprint arXiv:2412.03213*, 2024c.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023a. URL `http://papers.nips.cc/paper_files/paper/2023/hash/6dcf277ea32ce3288914faf369fe6de0-Abstract-Conference.html`.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023b.

Peiyu Liu, Ze-Feng Gao, Wayne Xin Zhao, Zhi-Yuan Xie, Zhong-Yi Lu, and Ji-Rong Wen. Enabling lightweight fine-tuning for pre-trained language model compression based on matrix product operators. *arXiv preprint arXiv:2106.02205*, 2021.

Peiyu Liu, Ze-Feng Gao, Wayne Xin Zhao, Yipeng Ma, Tao Wang, and Ji-Rong Wen. Unlocking data-free low-bit quantization with matrix decomposition for kv cache compression. *arXiv preprint arXiv:2405.12591*, 2024d.

Ruikang Liu, Haoli Bai, Haokun Lin, Yuening Li, Han Gao, Zhengzhuo Xu, Lu Hou, Jun Yao, and Chun Yuan. Intactkv: Improving large language model quantization by keeping pivot tokens intact. *arXiv preprint arXiv:2403.01241*, 2024e.

Tianyang Liu, Canwen Xu, and Julian McAuley. RepoBench: Benchmarking Repository-Level Code Auto-Completion Systems, 2023c. URL `https://arxiv.org/abs/2306.03091`.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, et al. Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 38–56, 2024f.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant–llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024g.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023d.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024h.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. Deepseek-vl: Towards real-world vision-language understanding. *CoRR*, abs/2403.05525, 2024. doi: 10.48550/ARXIV.2403.05525. URL https://doi.org/10.48550/arXiv.2403.05525.

Qianli Ma, Zhen Liu, Zhenjing Zheng, Ziyang Huang, Siying Zhu, Zhongzhong Yu, and James T Kwok. A survey on time-series pre-trained models. *IEEE Transactions on Knowledge and Data Engineering*, 2024a.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. Affinequant: Affine transformation quantization for large language models. *arXiv preprint arXiv:2403.12544*, 2024b.

Osman Asif Malik and Stephen Becker. Low-rank tucker decomposition of large tensors using tensorsketch. *Advances in neural information processing systems*, 31, 2018.

Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh. A survey of product quantization. *ITE Transactions on Media Technology and Applications*, 6(1):2–10, 2018.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. Towards efficient generative large language model serving: A survey from algorithms to systems. *arXiv preprint arXiv:2312.15234*, 2023.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Comput. Surv.*, 56(2):30:1–30:40, 2024. doi: 10.1145/3605943. URL https://doi.org/10.1145/3605943.

João Monteiro, Étienne Marcotte, Pierre-André Noël, Valentina Zantedeschi, David Vázquez, Nicolas Chapados, Christopher Pal, and Perouz Taslakian. XC-Cache: Cross-Attending to Cached Context for Efficient LLM Inference. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pp. 15284–15302. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.findings-emnlp.896.

Jesse Mu, Xiang Li, and Noah D. Goodman. Learning to compress prompts with gist tokens. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/3d77c6dcc7f143aa2154e7f4d5e22d68-Abstract-Conference.html.

Yongyu Mu, Yuzhang Wu, Yuchun Fan, Chenglong Wang, Hengyu Li, Qiaozhi He, Murun Yang, Tong Xiao, and Jingbo Zhu. Cross-layer Attention Sharing for Large Language Models, 2024. URL https://arxiv.org/abs/2408.01890.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave No Context Behind: Efficient Infinite Context Transformers with Infini-attention, August 2024. URL http://arxiv.org/abs/2404.07143.

Ramesh Nallapati, Bowen Zhou, Cicero Dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 280–290, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL `http://aclweb.org/anthology/K16-1028`.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization, 2018. URL `https://arxiv.org/abs/1808.08745`.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *CoRR*, abs/2307.06435, 2023. doi: 10.48550/ARXIV.2307.06435. URL `https://doi.org/10.48550/arXiv.2307.06435`.

Piotr Nawrot, Adrian Lancucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. Dynamic Memory Compression: Retrofitting LLMs for Accelerated Inference. In *Forty-First International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=tDRYrAkOB7`.

Arka Pal, Deep Karkhanis, Manley Roberts, Samuel Dooley, Arvind Sundararajan, and Siddartha Naidu. Giraffe: Adventures in expanding context lengths in llms, 2023. URL `https://arxiv.org/abs/2308.10882`.

Xiurui Pan, Endian Li, Qiao Li, Shengwen Liang, Yizhou Shan, Ke Zhou, Yingwei Luo, Xiaolin Wang, and Jie Zhang. Instinfer: In-storage attention offloading for cost-effective long-context llm inference, 2024. URL `https://arxiv.org/abs/2409.04992`.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. QuALITY: Question Answering with Long Input Texts, Yes!, 2021. URL `https://arxiv.org/abs/2112.08608`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://aclanthology.org/P02-1040`.

Seungcheol Park, Jaehyeon Choi, Sojin Lee, and U Kang. A comprehensive survey of compression algorithms for language models. *arXiv preprint arXiv:2401.15347*, 2024.

Badri Narayana Patro and Vijay Srinivas Agneeswaran. Mamba-360: Survey of state space models as transformer alternative for long sequence modelling: Methods, applications, and challenges. *arXiv preprint arXiv:2404.16112*, 2024.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Jiaju Lin, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Bolun Wang, Johan S. Wind, Stanislaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the Transformer Era, December 2023. URL `http://arxiv.org/abs/2305.13048`.

Alina Petukhova and Nuno Fachada. MN-DS: A Multilabeled News Dataset for News Articles Hierarchical Classification. *Data*, 8(5):74, April 2023. ISSN 2306-5729. doi: 10.3390/data8050074. URL `https://www.mdpi.com/2306-5729/8/5/74`.

Stefan Podlipnig and Laszlo Böszörmenyi. A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)*, 35(4):374–398, 2003.

Matt Post. A call for clarity in reporting BLEU scores. In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor (eds.), *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6319. URL `https://aclanthology.org/W18-6319`.

Jianing Qiu, Lin Li, Jiankai Sun, Jiachuan Peng, Peilun Shi, Ruiyang Zhang, Yinzhao Dong, Kyle Lam, Frank P-W Lo, Bo Xiao, et al. Large ai models in health informatics: Applications, challenges, and the future. *IEEE Journal of Biomedical and Health Informatics*, 2023.

Haohao Qu, Liangbo Ning, Rui An, Wenqi Fan, Tyler Derr, Hui Liu, Xin Xu, and Qing Li. A survey of mamba. *arXiv preprint arXiv:2408.01129*, 2024.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=SylKikSYDH`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016a. URL `http://arxiv.org/abs/1606.05250`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016b. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://aclanthology.org/D16-1264`.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient LLM inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Tomáš Koˇ ciský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, TBD:TBD, 2018. URL `https://TBD`.

Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for kv cache compression. *arXiv preprint arXiv:2408.05646*, 2024.

Nils Schaetti. Sfgram: a dataset containing thousands of scienc-fiction books and novels. `https://github.com/nschaetti/EchoTorch`, 2018.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. SCROLLS: Standardized CompaRison Over Long Language Sequences, 2022. URL `https://arxiv.org/abs/2201.03533`.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. ZeroSCROLLS: A Zero-Shot Benchmark for Long Text Understanding, 2023. URL `https://arxiv.org/abs/2305.14196`.

Rana Shahout, Cong Liang, Shiji Xin, Qianru Lao, Yong Cui, Minlan Yu, and Michael Mitzenmacher. Fast inference for augmented large language models, 2024. URL https://arxiv.org/abs/2410.18248.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

Akshat Sharma, Hangliang Ding, Jianping Li, Neel Dani, and Minjia Zhang. Minikv: Pushing the limits of llm inference via 2-bit layer-discriminative kv cache, 2024. URL https://arxiv.org/abs/2411.18077.

Eva Sharma, Chen Li, and Lu Wang. BIGPATENT: A Large-Scale Dataset for Abstractive and Coherent Summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2204–2213, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1212. URL https://www.aclweb.org/anthology/P19-1212.

Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need, November 2019. URL http://arxiv.org/abs/1911.02150.

Ao Shen, Zhiyao Li, and Mingyu Gao. Fastswitch: Optimizing context switching efficiency in fairness-aware large language model serving, 2024. URL https://arxiv.org/abs/2411.18424.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single GPU. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31094–31116. PMLR, 2023. URL https://proceedings.mlr.press/v202/sheng23a.html.

Luohe Shi, Hongyi Zhang, Yao Yao, Zuchao Li, and Hai Zhao. Keep the cost down: A review on methods to optimize llm's kv-cache consumption. *arXiv preprint arXiv:2407.18003*, 2024.

Vasudev Shyam, Jonathan Pilault, Emily Shepperd, Quentin Anthony, and Beren Millidge. Tree attention: Topology-aware decoding for long-context attention on gpu clusters, 2024. URL https://arxiv.org/abs/2408.04093.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Dingjie Song, Shunian Chen, Guiming Hardy Chen, Fei Yu, Xiang Wan, and Benyou Wang. Milebench: Benchmarking mllms in long context. *arXiv preprint arXiv:2404.18532*, 2024.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024a.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models. *CoRR*, abs/2402.17762, 2024b. doi: 10.48550/ARXIV.2402.17762. URL https://doi.org/10.48550/arXiv.2402.17762.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive Network: A Successor to Transformer for Large Language Models, August 2023. URL http://arxiv.org/abs/2307.08621.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You Only Cache Once: Decoder-Decoder Architectures for Language Models, May 2024c. URL http://arxiv.org/abs/2405.05254.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024d.

Zhaoxuan Tan and Meng Jiang. User modeling in the era of large language models: Current research and future directions. *arXiv preprint arXiv:2312.11518*, 2023.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient KV cache compression through retrieval heads. *CoRR*, abs/2407.15891, 2024a.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: query-aware sparsity for efficient long-context LLM inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b.

Yehui Tang, Yunhe Wang, Jianyuan Guo, Zhijun Tu, Kai Han, Hailin Hu, and Dacheng Tao. A survey on transformer compression. *arXiv preprint arXiv:2402.05964*, 2024c.

Qian Tao, Wenyuan Yu, and Jingren Zhou. Asymkv: Enabling 1-bit quantization of kv cache with layer-wise asymmetric quantization configurations. *arXiv preprint arXiv:2410.13212*, 2024.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long Range Arena: A Benchmark for Efficient Transformers, 2020. URL https://arxiv.org/abs/2011.04006.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023. doi: 10.48550/ARXIV.2302.13971. URL https://doi.org/10.48550/arXiv.2302.13971.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop Questions via Single-hop Question Composition, 2021. URL https://arxiv.org/abs/2108.00573.

Bo-Hsiang Tseng, Sheng-Syun Shen, Hung-Yi Lee, and Lin-Shan Lee. Towards Machine Comprehension of Spoken Content: Initial TOEFL Listening Comprehension Test by Machine, August 2016. URL http://arxiv.org/abs/1608.06378. arXiv:1608.06378.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, et al. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 1, 2023.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, and Mi Zhang. D2O: dynamic discriminative operations for efficient generative inference of large language models. *CoRR*, abs/2406.13035, 2024a. doi: 10.48550/ARXIV.2406.13035. URL https://doi.org/10.48550/arXiv.2406.13035.

Zhongwei Wan, Ziang Wu, Che Liu, Jinfa Huang, Zhihong Zhu, Peng Jin, Longyue Wang, and Li Yuan. LOOK-M: look-once optimization in KV cache for efficient multimodal long-context inference. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pp. 4065–4078. Association for Computational Linguistics, 2024b. URL https://aclanthology.org/2024.findings-emnlp.235.

Alex Wang, Richard Yuanzhe Pang, Angelica Chen, Jason Phang, and Samuel R. Bowman. SQuALITY: Building a long-document summarization dataset the hard way. *arXiv preprint 2205.11465*, 2022a.

Ao Wang, Hui Chen, Jianchao Tan, Kefeng Zhang, Xunliang Cai, Zijia Lin, Jungong Han, and Guiguang Ding. Prefixkv: Adaptive prefix kv cache is what vision instruction-following models need for efficient generation, 2024a. URL `https://arxiv.org/abs/2412.03409`.

Junxiong Wang, Jing Nathan Yan, Albert Gu, and Alexander M Rush. Pretraining without attention. *arXiv preprint arXiv:2212.10544*, 2022b.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748*, 2024b.

Yumeng Wang and Zhenyang Xiao. LoMA: Lossless Compressed Memory Attention, February 2024. URL `http://arxiv.org/abs/2401.09486`.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive KV cache merging for llms on long-context tasks. *CoRR*, abs/2407.08454, 2024c. doi: 10.48550/ARXIV.2407.08454. URL `https://doi.org/10.48550/arXiv.2407.08454`.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414, 2022.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*, 2023.

Bingyang Wu, Yinmin Zhong, Zili Zhang, Shengyu Liu, Fangyue Liu, Yuanhang Sun, Gang Huang, Xuanzhe Liu, and Xin Jin. Fast distributed inference serving for large language models, 2024a. URL `https://arxiv.org/abs/2305.05920`.

Han Wu, Mingjie Zhan, Haochen Tan, Zhaohui Hou, Ding Liang, and Linqi Song. VCSUM: A Versatile Chinese Meeting Summarization Dataset, 2023a. URL `https://arxiv.org/abs/2305.05280`.

Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding, 2024b. URL `https://arxiv.org/abs/2407.15754`.

Haoyi Wu and Kewei Tu. Layer-Condensed KV Cache for Efficient Inference of Large Language Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 11175–11188. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.602. URL `https://doi.org/10.18653/v1/2024.acl-long.602`.

Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and Philip S. Yu. Multimodal large language models: A survey. In Jingrui He, Themis Palpanas, Xiaohua Hu, Alfredo Cuzzocrea, Dejing Dou, Dominik Slezak, Wei Wang, Aleksandra Gruca, Jerry Chun-Wei Lin, and Rakesh Agrawal (eds.), *IEEE International Conference on Big Data, BigData 2023, Sorrento, Italy, December 15-18, 2023*, pp. 2247–2256. IEEE, 2023b. doi: 10.1109/BIGDATA59044.2023.10386743. URL `https://doi.org/10.1109/BigData59044.2023.10386743`.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. A survey on large language models for recommendation. *World Wide Web*, 27(5):60, 2024c.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. *CoRR*, abs/2404.15574, 2024d.

You Wu, Haoyi Wu, and Kewei Tu. A Systematic Study of Cross-Layer KV Sharing for Efficient LLM Inference, October 2024e. URL `http://arxiv.org/abs/2410.14442`.

Haocheng Xi, Changhao Li, Jianfei Chen, and Jun Zhu. Training transformers with 4-bit integers. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/99fc8bc48b917c301a80cb74d91c0c06-Abstract-Conference.html`.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *arXiv preprint arXiv:2401.07851*, 2024.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory, 2024a. URL `https://arxiv.org/abs/2402.04617`.

Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL `https://proceedings.mlr.press/v202/xiao23c.html`.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. *CoRR*, abs/2410.10819, 2024b.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024c. URL `https://openreview.net/forum?id=NG7sS51zVF`.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9777–9786, June 2021.

Yi Xiong, Hao Wu, Changxu Shao, Ziqing Wang, Rui Zhang, Yuhong Guo, Junping Zhao, Ke Zhang, and Zhenxuan Pan. Layerkv: Optimizing large language model serving with layer-wise kv cache management, 2024. URL `https://arxiv.org/abs/2410.00428`.

Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *ACM Multimedia*, 2017.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. Large language models for generative information extraction: A survey. *Frontiers of Computer Science*, 18(6):186357, 2024a.

Fangyuan Xu, Tanya Goyal, and Eunsol Choi. Recycled attention: Efficient inference for long-context language models, 2024b. URL `https://arxiv.org/abs/2411.05787`.

Jiale Xu, Rui Zhang, Cong Guo, Weiming Hu, Zihan Liu, Feiyang Wu, Yu Feng, Shixuan Sun, Changxu Shao, Yuhong Guo, Junping Zhao, Ke Zhang, Minyi Guo, and Jingwen Leng. vtensor: Flexible virtual tensor management for efficient llm serving, 2024c. URL `https://arxiv.org/abs/2407.15309`.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*, 2023.

Rui Xu, Shu Yang, Yihui Wang, Bo Du, and Hao Chen. A survey on vision mamba: Models, applications and challenges. *arXiv preprint arXiv:2404.18861*, 2024d.

Xin Xu and Zhouchen Lin. MixCon: A Hybrid Architecture for Efficient and Adaptive Sequence Modeling. In Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto Bugarín-Diz, José M. Alonso-Moral, Senén Barro, and Fredrik Heintz (eds.), *Frontiers in Artificial Intelligence and Applications*. IOS Press, October 2024. ISBN 978-1-64368-548-9. doi: 10.3233/FAIA240593. URL https://ebooks.iospress.nl/doi/10.3233/FAIA240593.

Xiongxiao Xu, Yueqing Liang, Baixiang Huang, Zhiling Lan, and Kai Shu. Integrating mamba and transformer for long-short range time series forecasting. *arXiv preprint arXiv:2404.14757*, 2024e.

Ruiqing Yan, Linghan Zheng, Xingbo Du, Han Zou, Yufeng Guo, and Jianfei Yang. RecurFormer: Not All Transformer Heads Need Self-Attention, October 2024. URL http://arxiv.org/abs/2410.12850.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. Pyramidinfer: Pyramid KV cache compression for high-throughput LLM inference. In *ACL*, pp. 3258–3270. Association for Computational Linguistics, 2024a.

Hua Yang, Duohai Li, and Shiman Li. MCSD: An Efficient Language Model with Diverse Fusion, July 2024b. URL http://arxiv.org/abs/2406.12230.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024c.

Yifei Yang, Zouying Cao, Qiguang Chen, Libo Qin, Dongjie Yang, Hai Zhao, and Zhi Chen. Kvsharer: Efficient inference via layer-wise dissimilar KV cache sharing. *CoRR*, abs/2410.18517, 2024d. doi: 10.48550/ARXIV.2410.18517. URL https://doi.org/10.48550/arXiv.2410.18517.

Zhen Yang, J. N. Han, Kan Wu, Ruobing Xie, An Wang, Xingwu Sun, and Zhanhui Kang. Lossless KV Cache Compression to 2%, October 2024e. URL http://arxiv.org/abs/2410.15252.

Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. Llm4drive: A survey of large language models for autonomous driving. In *NeurIPS 2024 Workshop on Open-World Agents*, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering, September 2018. URL http://arxiv.org/abs/1809.09600. arXiv:1809.09600.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving with cached knowledge fusion. *arXiv preprint arXiv:2405.16444*, 2024a.

Jinwei Yao, Kaiqi Chen, Kexun Zhang, Jiaxuan You, Binhang Yuan, Zeke Wang, and Tao Lin. Deft: Decoding with flash tree-attention for efficient tree-structured llm inference, 2024b. URL https://arxiv.org/abs/2404.00242.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022.

Lu Ye, Ze Tao, Yong Huang, and Yang Li. Chunkattention: Efficient self-attention with prefix-aware kv cache and two-phase partition, 2024. URL https://arxiv.org/abs/2402.15220.

Howard Yen, Tianyu Gao, and Danqi Chen. Long-Context Language Modeling with Parallel Context Encoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 2588–2610. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.142. URL `https://doi.org/10.18653/v1/2024.acl-long.142`.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In Marcos K. Aguilera and Hakim Weatherspoon (eds.), *16th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2022, Carlsbad, CA, USA, July 11-13, 2022*, pp. 521–538. USENIX Association, 2022. URL `https://www.usenix.org/conference/osdi22/presentation/yu`.

Hao Yu, Zelan Yang, Shen Li, Yong Li, and Jianxin Wu. Effectively compress kv heads for llm. *arXiv preprint arXiv:2406.07056*, 2024.

Lingfan Yu and Jinyang Li. Stateful large language model serving with pensieve. *CoRR*, abs/2312.05516, 2023. doi: 10.48550/ARXIV.2312.05516. URL `https://doi.org/10.48550/arXiv.2312.05516`.

Jiayi Yuan, Hongyi Liu, Shaochen Zhong, Yu-Neng Chuang, Songchen Li, Guanchu Wang, Duy Le, Hongye Jin, Vipin Chaudhary, Zhaozhuo Xu, et al. Kv cache compression, but what must we give in return? a comprehensive benchmark of long context capable approaches. *arXiv preprint arXiv:2407.01527*, 2024.

Weizhe Yuan, Pengfei Liu, and Graham Neubig. Can We Automate Scientific Reviewing?, 2021. URL `https://arxiv.org/abs/2102.00176`.

Yuanhan Zhang Yuan Liu, Haodong Duan et al. Mmbench: Is your multi-modal model an all-around player? *arXiv:2307.06281*, 2023.

Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. Wkvquant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*, 2024.

Amir Zandieh, Majid Daliri, and Insu Han. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. *arXiv preprint arXiv:2406.03482*, 2024.

etc Zefan-Cai. Awesome-llm-kv-cache: A curated list of awesome llm inference papers with codes, 2024. URL `https://github.com/Zefan-Cai/Awesome-LLM-KV-Cache`. Open-source software available at https://github.com/Zefan-Cai/Awesome-LLM-KV-Cache.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. GLM-130B: an open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/forum?id=-Aw0rrrPUF`.

Zihao Zeng, Bokai Lin, Tianqi Hou, Hao Zhang, and Zhijie Deng. In-context kv-cache eviction for llms via attention-gate. *CoRR*, abs/2410.12876, 2024.

Duzhen Zhang, Yahan Yu, Jiahua Dong, Chenxing Li, Dan Su, Chenhui Chu, and Dong Yu. Mm-llms: Recent advances in multimodal large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 12401–12430. Association for Computational Linguistics, 2024a. doi: 10.18653/V1/2024.FINDINGS-ACL.738. URL `https://doi.org/10.18653/v1/2024.findings-acl.738`.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. Pqcache: Product quantization-based kvcache for long context llm inference. *arXiv preprint arXiv:2407.12820*, 2024b.

Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(8):5625–5644, 2024c. doi: 10.1109/TPAMI.2024.3369699. URL https://doi.org/10.1109/TPAMI.2024.3369699.

Rongzhi Zhang, Kuang Wang, Liyuan Liu, Shuohang Wang, Hao Cheng, Chao Zhang, and Yelong Shen. Lorc: Low-rank compression for llms kv cache with a progressive compression strategy. *arXiv preprint arXiv:2410.03111*, 2024d.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020. URL https://arxiv.org/abs/1904.09675.

Xuan Zhang, Cunxiao Du, Chao Du, Tianyu Pang, Wei Gao, and Min Lin. Simlayerkv: A simple framework for layer-level KV cache reduction. *CoRR*, abs/2410.13846, 2024e.

Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John C. S. Lui, and Haibo Chen. Unifying kv cache compression for large language models with leankv, 2024f. URL https://arxiv.org/abs/2412.03131.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. Cam: Cache merging for memory-efficient llms inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024g. URL https://openreview.net/forum?id=LCTmppB165.

Zeyu Zhang and Haiying Shen. Zero-delay qkv compression for mitigating kv cache and network bottlenecks in llm inference. *arXiv preprint arXiv:2408.04107*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023a.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2O: heavy-hitter oracle for efficient generative inference of large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/6ceefa7b15572587b78ecfcebb2827f8-Abstract-Conference.html.

Junqi Zhao, Zhijin Fang, Shu Li, Shaohui Yang, and Shichao He. Buzz: Beehive-structured sparse kv cache with segmented heavy hitters for efficient llm inference, 2024a. URL https://arxiv.org/abs/2410.23079.

Liang Zhao, Xiaocheng Feng, Xiachong Feng, Bin Qin, and Ting Liu. Length extrapolation of transformers: A survey from the perspective of position encoding. *arXiv preprint arXiv:2312.17044*, 2023.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209, 2024b.

Ying Zhao and Jinjun Chen. A survey on differential privacy for unstructured data content. *ACM Computing Surveys (CSUR)*, 54(10s):1–28, 2022.

Youpeng Zhao, Di Wu, and Jun Wang. Alisa: Accelerating large language model inference via sparsity-aware kv caching, 2024c. URL https://arxiv.org/abs/2403.17312.

Jianqiao Zheng, Sameera Ramasinghe, and Simon Lucey. Rethinking positional encoding. *arXiv preprint arXiv:2107.02561*, 2021.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024a. URL https://arxiv.org/abs/2312.07104.

Zhen Zheng, Xin Ji, Taosong Fang, Fanghao Zhou, Chuanjie Liu, and Gang Peng. Batchllm: Optimizing large batched llm inference with global prefix sharing and throughput-oriented token batching, 2024b. URL https://arxiv.org/abs/2412.03594.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. QMSum: A New Benchmark for Query-based Multi-domain Meeting Summarization, 2021. URL https://arxiv.org/abs/2104.05938.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. In Ada Gavrilovska and Douglas B. Terry (eds.), *18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024*, pp. 193–210. USENIX Association, 2024a. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

Yiwu Zhong, Zhuoming Liu, Yin Li, and Liwei Wang. Aim: Adaptive inference of multi-modal llms via token merging and pruning. *arXiv preprint arXiv:2412.03248*, 2024b.

Daquan Zhou, Kai Wang, Jianyang Gu, Xiangyu Peng, Dongze Lian, Yifan Zhang, Yang You, and Jiashi Feng. Dataset quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17205–17216, 2023a.

Hongjian Zhou, Fenglin Liu, Boyang Gu, Xinyu Zou, Jinfa Huang, Jinge Wu, Yiru Li, Sam S Chen, Peilin Zhou, Junling Liu, et al. A survey of large language models in medicine: Progress, application, and challenge. *arXiv preprint arXiv:2311.05112*, 2023b.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. Mlvu: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*, 2024a.

Pan Zhou, Canyi Lu, Zhouchen Lin, and Chao Zhang. Tensor factorization for low-rank tensor completion. *IEEE Transactions on Image Processing*, 27(3):1152–1163, 2017.

Yiren Zhou, Seyed-Mohsen Moosavi-Dezfooli, Ngai-Man Cheung, and Pascal Frossard. Adaptive quantization for deep neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Zhanchao Zhou, Tianyi Wu, Zhiyun Jiang, and Zhenzhong Lan. Value Residual Learning For Alleviating Attention Concentration In Transformers, December 2024b. URL http://arxiv.org/abs/2410.17897.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024c.

Junnan Zhu, Qian Wang, Yining Wang, Yu Zhou, Jiajun Zhang, Shaonan Wang, and Chengqing Zong. NCLS: Neural Cross-Lingual Summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3052–3062, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1302. URL https://www.aclweb.org/anthology/D19-1302.

Xiaodan Zhu. *Natural Language Processing and Chinese Computing: 9th CCF International Conference, NLPCC 2020, Zhengzhou, China, October 14-18, 2020, Proceedings, Part I*. Number v.12430 in Lecture Notes in Computer Science Ser. Springer International Publishing AG, Cham, 2020. ISBN 9783030604509.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.

Bohan Zhuang, Jing Liu, Zizheng Pan, Haoyu He, Yuetian Weng, and Chunhua Shen. A survey on efficient training of transformers. *arXiv preprint arXiv:2302.01107*, 2023.

Zayd Muhammad Kawakibi Zuhri, Muhammad Farid Adilazuarda, Ayu Purwarianti, and Alham Fikri Aji. MLKV: Multi-Layer Key-Value Heads for Memory Efficient Transformer Decoding, October 2024. URL http://arxiv.org/abs/2406.09297.