# MILO: EFFICIENT MANY-SHOT IN-CONTEXT LEARNING WITH BLOCK-WISE LOW-RANK COMPRESSION

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

## ABSTRACT

Many-shot in-context learning (ICL) enables large language models (LLMs) to adapt to complex tasks by conditioning on thousands of demonstration examples, but this paradigm shifts the inference efficiency bottleneck to the key-value (KV) cache memory. Due to the linear scaling behavior of the KV cache, storing these intermediate tensors has become a paramount challenge for both online serving and on-device deployment. To address this issue, we propose a novel compression framework, termed **MILO**, that exploits the low-rank redundancy inherent in many-shot contexts. Specifically, MILO features a block-wise low-rank compression strategy that compresses the KV cache at the block granularity, where each block contains multiple many-shot examples. Furthermore, to handle the heterogeneous context density across different blocks, MILO dynamically allocates rank budgets based on the information entropy, preserving the fidelity of critical blocks while aggressively compressing redundant ones. Experimental results on Qwen2.5 models demonstrate that our method achieves up to 50% reduction in KV cache memory and 1.8× throughput improvement, with negligible performance degradation on classification and reasoning benchmarks, significantly outperforming prior baselines.

## 1 INTRODUCTION

In-context learning (ICL) (Brown et al., 2020; Agarwal et al., 2024; Jiang et al., 2024; Song et al., 2024) is a powerful capability of large language models (LLMs) Meta (2024); Brown et al. (2020); Yang et al. (2024a) that enables the model to learn a new task at inference time by conditioning on a few demonstration examples or 'shots' provided directly in the prompt, without any parameter updates. Traditionally, it has been limited to "few-shot" learning due to the small context windows of prior models Brown et al. (2020). However, with the recent expansion of model context windows to a million tokens or more, a new paradigm of 'many-shot ICL' has emerged Agarwal et al. (2024); Jiang et al. (2024); Song et al. (2024); Xiao et al. (2025). Through augmenting prompts with hundreds or even thousands of examples, many-shot ICL has been shown to significantly boost performance on complex downstream tasks, often approaching the accuracy of fully fine-tuned models Agarwal et al. (2024); Jiang et al. (2024).

Despite its effectiveness across a wide range of tasks, many-shot ICL introduces significant efficiency challenges for practical deployment Dong et al. (2022); Xiao et al. (2025); Golchin et al. (2025). Long-context inputs place significantly higher memory and compute requirements for inference due to the quadratic complexity of the attention operation and the existence of key-value (KV) cache Ott et al. (2019). To maintain high performance and low latency, many-shot ICL often requires careful and query-dependent example selection to construct proper inputs an Luo et al. (2024); Golchin et al. (2025). Recent works have attempted to minimize the inference costs by pre-encoding the KV cache of many-shot examples Xiao et al. (2025); Ou et al. (2026). Notably, DBSA applies dynamic sparse attention to alleviate the compute burden, and Adapshot further introduces semantic-aware KV cache reuse Ou et al. (2026). However, none of the prior works discuss the storage and memory efficiency aspects of the KV cache. For instance, storing the KV cache for 90k examples necessitates an additional 11.1 GB for 8B dense models Xiao et al. (2025). As the long-context capabilities are continuously expanded, with models like Gemini now supporting 10M tokens, the growing memory footprint for encoding prior examples makes many-shot ICL increasingly memory-bound and impractical for serving with stringent service-level objectives (SLOs).



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this work, we present an efficient framework, termed **MILO**, to address the above-mentioned KV cache challenges in many-shot ICL. **Our key idea is to exploit the observed low-rank nature of the KV cache of pre-encoded many-shot examples. This low-rank redundancy arises because many examples share similar semantic features, leading to highly correlated representations within the KV cache in latent space.** By storing the KV cache in low-dimensional space, MILO can achieve up to 50% storage cost reduction and 1.8× speedup against prior works with minimal accuracy degradation. However, achieving this performance requires solving two unique challenges.

First, it is essential to choose the appropriate granularity for low-rank compression to achieve a balance between memory efficiency and model performance. Compression itself is a lossy process, where a naive, global approach risks losing fine-grained information across different examples and, most importantly, induces redundant computation overhead during decompression Chang et al. (2024). Conversely, attempting to compress on an example-by-example basis fails to capture the global semantics and results in a significant performance drop due to accumulated information loss from each example Sun et al. (2024). To resolve this trade-off, MILO employs a block-wise low-rank compression (BLC) method. Specifically, the KV caches are compressed at the granularity of a block, where each block contains multiple many-shot examples. Such a granularity can strike the best trade-offs between model accuracy and inference efficiency, thereby improving the end-to-end quality of service (QoS).

A second challenge lies in the fact that the information density within the KV cache is highly heterogeneous across the blocks. While some blocks focus on local semantic patterns that are highly redundant, others often retain critical information required for tasks like complex chain-of-thought (CoT) reasoning Wei et al. (2022); Gu et al. (2025) and test-time scaling Muennighoff et al. (2025); Snell et al. (2024); Ji et al. (2025); Zhao et al. (2025b). Applying a uniform compression rank across the entire cache is therefore often suboptimal. To address this issue, we propose a dynamic rank adaptation mechanism that allocates higher rank budgets based on the entropy of each block and aggressively compresses redundant ones. This ensures that the compression algorithm can preserve the fidelity of retrieval capabilities while minimizing the aggregate KV cache memory footprint. Furthermore, we design and implement a suite of system optimizations, e.g., customized kernels and CUDA graphs and streams, tailored to MILO, to achieve real-world end-to-end benefits.

In summary, our contributions are as follows:

- We identify the KV cache memory challenges in many-shot ICL and propose a new framework, termed MILO, by exploiting the low-rank redundancy inherent in the KV cache during many-shot ICL.
- MILO features block-wise low-rank compression (BLC) that balances reconstruction efficiency with model representation quality, and a dynamic rank adaptation mechanism to allocate ranks based on the information entropy.
- Evaluations on Qwen2.5 models demonstrate that our framework significantly reduces KV cache storage and improves inference latency with negligible accuracy degradation across a suite of standard ICL benchmarks.

## 2 BACKGROUND

**Many-Shot ICL.** The paradigm of in-context learning (ICL) was first popularized by GPT-3 Brown et al. (2020), demonstrating that LLMs can adapt to new tasks by conditioning on a few demonstration examples without parameter updates. While early work focused on standard "few-shot" settings (typically 1∼8 examples) due to limited context windows (e.g., 2k or 4k tokens), the recent advent of long-context models—capable of processing up to 10M tokens—has unlocked the potential for "many-shot" ICL. Recent studies indicate that scaling the number of demonstrations from dozens to thousands yields monotonic performance improvements, rivaling the performance of fine-tuned models on complex reasoning and extraction tasks Agarwal et al. (2024); Jiang et al. (2024).

**Inference Optimization for Many-Shot ICL.** Despite the promising results of many-shot ICL, this approach shifts the bottleneck from offline training to online inference, where the prompt length scales linearly with the number of shots. The quadratic complexity of self-attention and, more critically, the linear growth of the key-value (KV) cache imposes severe memory and latency con-

straints Ott et al. (2019). Several recent works have attempted to resolve such inference challenges. Notably, DBSA explores pre-encoding mechanisms with dynamic sparse attention to reduce compute latency Xiao et al. (2025). Adapshot proposes dynamic context budget allocation and KV cache reuse to further improve both accuracy performance and system throughput Ou et al. (2026). However, the challenge of managing the memory footprint for the cached intermediate states of thousands of examples remains an open problem that our work aims to address.

**Low-rank Compression.** Several works have explored compressing LLMs in low-rank space. Prior efforts have generally focused on weight and activation compression using Fisher information Yuan et al. (2023); Hsu et al. (2022). Recently, as context length continues to scale up, several works aim to further compress KV cache for more efficient memory management Chang et al. (2024); Yu et al. (2024a); Lin et al. (2024); Saxena et al. (2024); Sun et al. (2024); Yu et al. (2024b); Zhang et al. (2024). Notably, Palu applies a low-rank projection matrix in zero-shot settings Chang et al. (2024), and LoRC proposes inter-layer progressive compression Zhang et al. (2024) Yu et al. (2024b) explores compressing KV cache at the head level by converting multi-head attention (MHA) to group-query attention (GQA) with LoRA fine-tuning Hu et al. (2021). *However, existing works focus on the compression within a limited context window and do not consider the low-rank nature across examples. Our work aims to extend low-rank compression to many-shot settings and fill in the gap.*

**Sparse Attention and Quantization.** Another line of inference-time approaches is based on the observation that not all tokens are created equal, thus creating the opportunity of sparse attention. $H_2O$ Zhang et al. (2023), StreamingLLM Xiao et al. (2023), and Scissorhands Liu et al. (2023), aim to reduce memory by evicting tokens deemed less important, typically retaining the most important ones for token generation. On the system level, ALISA Zhao et al. (2024) employs a co-design approach to maximize throughput performance on resource-constrained platforms. DBSA applies a similar block sparse attention method for retrieval-based many-shot in-context learning, achieving competitive performance against fine-tuning baselines Xiao et al. (2025). Quantization is also a standard method for reducing KV cache memory footprint. Techniques like KIVI Liu et al. (2024) and KVQuant Hooper et al. (2024) compress the KV cache by reducing numerical precision, storing entries in 4-bit or 2-bit formats. *Orthogonal to these works, our method aims to compress the representation of the KV cache from the perspective of low-rank approximation, which can be combined with quantization and sparse attention for further gains.*

## 3 OBSERVATION

To reduce memory footprint, recent works have explored the low-rank nature of the KV cache Chang et al. (2024); Yu et al. (2024a); Lin et al. (2024); Saxena et al. (2024); Sun et al. (2024). However, these methods focus on data-dependent compression, which often demands additional training or finetuning to achieve competitive compression rates Chang et al. (2024); Saxena et al. (2024). Furthermore, these studies are often limited to zero- or few-shot learning settings with limited context lengths. In our study, we aim to uncover the low-rank characteristics of the KV cache in many-shot ICL settings. Figures 1 and 2 visualize the singular value distributions of pre-RoPE key and value caches on Qwen2.5 models with many-shot inputs.

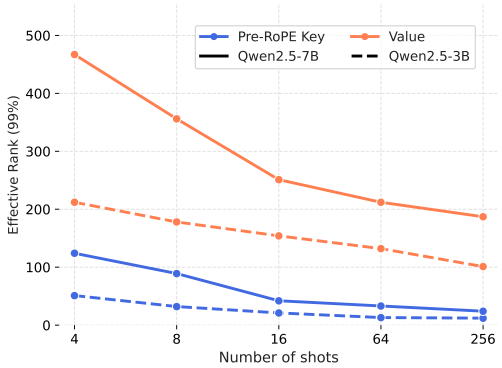


Figure 1: Top-1% Effective rank of pre-RoPE key and value over different numbers of shots on the MathQA dataset.

Here we have three key observations. First, the key cache exhibits a much stronger low-rank nature than the value cache in many-shot ICL settings, as shown in both Figure 1 and 2. This is consistent with prior findings that pre-RoPE keys are much better suited for low-rank compression Sun et al. (2024). Second, counterintuitively, scaling the number of shots in ICL creates more KV storage, but simultaneously introduces more redundancy in low-rank space, as shown in Figure 1. Third, across different blocks and different layers, the low-rank behavior is heterogeneous, as indicated in

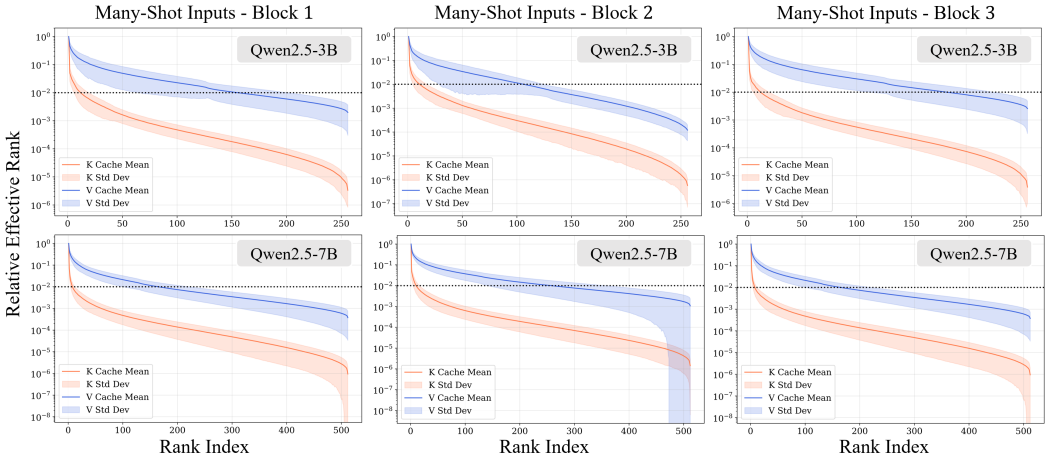


Figure 2: Observation of the low-rank characteristics of the key and value cache in Qwen2.5 models across different blocks of examples on MATHQA datasets. Here we sample three blocks, where each block contains 64 examples. Effective rank is defined as the square of singular values. We calculate the mean and std across all attention layers.

Figure 2. These observations both motivate and validate our solution to exploit the low-rank nature of many-shot contexts.

## 4 METHODOLOGY

In this section, we present our framework for efficient many-shot in-context learning. We first formulate the memory bottleneck problem in standard transformer attention. We then introduce block-wise low-rank compression, our approach for granular cache compression, and a dynamic rank allocation mechanism with efficient attention. Next, we present some system optimization techniques for MILO.

### 4.1 BACKGROUND

Consider a transformer-based LLM with $L$ layers and $H$ attention heads per layer. In the many-shot setting, the model processes a sequence of inputs $X = \{x_1, \ldots, x_N\}$, where $N$ represents a long context (e.g., $N > 10^5$). The standard self-attention mechanism for a specific head $h$ at layer $l$ computes the output $\mathbf{O}_h$ as:

$$\mathbf{O}_h = \text{Attn}(\mathbf{Q_h}, \mathbf{K_h}, \mathbf{V_h}) = \text{Softmax}\left(\frac{\mathbf{Q}_h \mathbf{K}_h^\top}{\sqrt{d}}\right)\mathbf{V}_h \tag{1}$$

where $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h \in \mathbb{R}^{N\times d}$ denote the Query, Key, and Value matrices, respectively, and $d$ is the head dimension. The primary bottleneck in many-shot ICL is the storage of the intermediate key-value tensors, i.e., the KV cache, $(\mathbf{K}, \mathbf{V})$, which grows linearly with the sequence length $N$ Ott et al. (2019). The total memory footprint $\mathcal{M}$ required to store the KV cache in FP16 precision is calculated as:

$$\mathcal{M} = 2 \cdot N \cdot L \cdot H \cdot d \cdot 2 \tag{2}$$

As the number of shots continues to grow, $\mathcal{M}$ can often exceed the high-bandwidth memory (HBM) capacity of modern GPUs, forcing evictions or offloading that degrade performance. Our objective is to compress $(\mathbf{K}, \mathbf{V})$ into approximations $(\hat{\mathbf{K}}, \hat{\mathbf{V}})$ such that the memory usage is minimized while preserving the model's accuracy performance.

### 4.2 BLOCK-WISE LOW-RANK COMPRESSION

Global compression methods often treat the entire context matrix as a single entity, but this approach risks losing fine-grained details essential for many-shot ICL. It also induces redundant computation

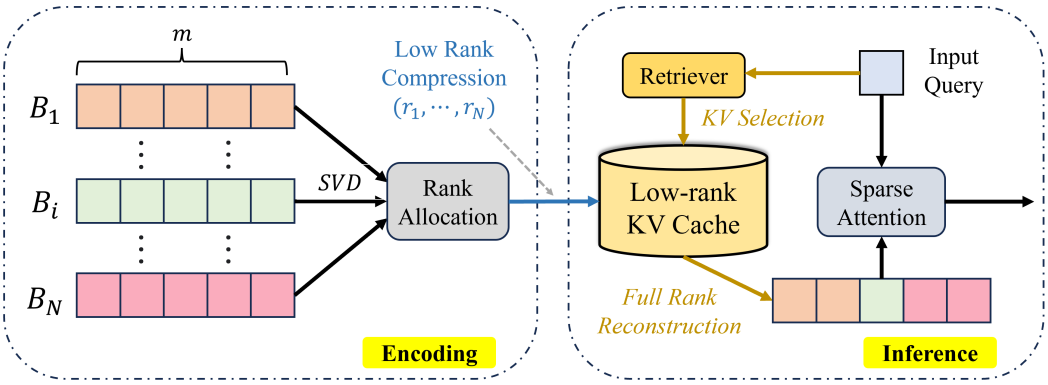


Figure 3: Overview of MILO. During the encoding stage, the KV cache for many-shot contexts is split into $N$ blocks, with each block containing $m$ examples. MILO first applies SVD to compute the appropriate rank for each block based on entropy analysis, then performs subsequent low-rank compression with rank budgets $(r_1, ..., r_N)$. During inference, the input query goes through the retriever to conduct KV cache selection, where the selected compressed KV cache is reconstructed in parallel to perform subsequent sparse attention.

during decoding, as only a handful of many-shot examples are selected for the entire KV cache Xiao et al. (2025). On the other hand, compressing each example or token leads to improved decoding efficiency, but suffers significant performance degradation, as the information loss from each example accumulates Sun et al. (2024); Yuan et al. (2023). To address the above-mentioned limitations, we exploit the insights from Section 3, and propose a **block-wise low-rank compression**. Specifically, we partition the KV cache into $N$ blocks, each containing multiple many-shot examples. Given a block of key-value pairs $\mathbf{K}_i, \mathbf{V}_i \in \mathbb{R}^{B\times d}$, where $B$ is the block size (total tokens across blocked examples) and $d$ is the head dimension, we apply low-rank approximation to each block independently. Given that key and value often exhibit different levels of low-rank, for each block $i$, we perform a decomposition for $\mathbf{K}_i$ and $\mathbf{V}_i$ separately:

$$\mathbf{K}_i \approx \mathbf{A}_i^{\mathbf{k}}\mathbf{\Sigma}_i^{\mathbf{k}}(\mathbf{B}_i^{\mathbf{k}})^\top, \quad \mathbf{V}_i \approx \mathbf{A}_i^{\mathbf{v}}\mathbf{\Sigma}_i^{\mathbf{v}}(\mathbf{B}_i^{\mathbf{v}})^\top \tag{3}$$

where $\mathbf{A}_i \in \mathbb{R}^{B\times r_i}$, $\mathbf{\Sigma}_i \in \mathbb{R}^{r_i\times r_i}$ is a diagonal matrix of singular values, and $\mathbf{B}_i \in \mathbb{R}^{d\times r_i}$. $r_i$ denotes the rank allocated to block $i$, satisfying $r_i \ll \min(B, d)$. Instead of storing $2Bd$ parameters per block, BLC stores the factorized matrices, reducing the storage cost to $(r_i^{\mathbf{k}}+r_i^{\mathbf{v}})(B+d)$. Assume a uniform block size $B$, the overall compression ratio $\rho$ is given by:

$$\rho = \sum_{i=1}^{N} \frac{(r_i^{\mathbf{k}} + r_i^{\mathbf{v}})(B + d)}{2 \cdot N \cdot B \cdot d} \tag{4}$$

This granularity allows us to capture local semantic dependencies while maintaining a high compression ratio.

### 4.3 Dynamic Rank Allocation

As shown in Figure 2, the information density in many-shot prompts is highly heterogeneous. This echoes prior works that a uniform rank allocation for the KV cache at each layer is often suboptimal Chang et al. (2024); Zhang et al. (2024). Here, we further demonstrate that this phenomenon also extends to many-shot ICL at the block level. Specifically, in a typical many-shot setting, the context window is populated with numerous demonstration blocks, but different blocks exhibit substantially different spectral complexity and therefore require different representation capacities under compression. Therefore, applying a uniform rank wastes representation capacity on highly redundant blocks while under-allocating rank to information-dense blocks.

Here, we formulate rank selection as a global resource allocation problem under a fixed total rank budget and dynamically assign the rank $r_i$ for each block based on its corresponding entropy. The notion of entropy is often used to measure such expressiveness and information density in neural networks Zhao et al. (2025a); Jaynes (1957); Chan et al. (2021); Yuan et al. (2023). Specifically, for

block $i$, we perform singular value decomposition (SVD) to obtain the singular values $\{\sigma_1, ..., \sigma_d\}$, where $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_d$. Following prior works Zhao et al. (2025a); Chan et al. (2021), we define the retained spectral information at rank $d$ as:

$$\mathbf{H}_i(d) \triangleq \sum_{j=1}^{d} \log(1 + \frac{\sigma_j^2}{\epsilon^2}) \tag{5}$$

where $\epsilon$ is a small constant for numerical stability. Given a total rank budget $R_{\text{tot}}$, MILO allocates block-specific ranks $\{r_i\}_{i=1}^N$ by solving the following optimization problem:

$$\max_{\{r_i\}} \quad \sum_{i=1}^{N} H_i(r_i), \quad \text{s.t.} \quad \sum_{i=1}^{N} r_i \leq R_{\text{tot}}, \quad r_{\min} \leq r_i \leq r_{\max}. \tag{6}$$

This formulation allocates more rank to blocks with slowly decaying spectra while aggressively compressing redundant blocks. Since $H_i(r)$ is additive over singular components, the allocation can be efficiently computed using their marginal information gains. The gain from increasing the rank of block $i$ from $r$ to $r+1$ is

$$\Delta H_{i,r} = H_i(r+1) - H_i(r) = \log\left(1 + \frac{\sigma_{i,r+1}^2}{\epsilon^2}\right). \tag{7}$$

Starting from $r_i = r_{\min}$, we can greedily assign each remaining rank unit to the block with the largest marginal gain until $\sum_i r_i = R_{\text{tot}}$. This produces a heterogeneous rank allocation while strictly satisfying the global compression budget.

### 4.4 MILO WORKFLOW

Combining the above two proposed techniques, we present MILO, a new framework for efficient many-shot ICL. Figure 3 and Algorithm 1 further demonstrate the detailed workflow of our framework. Here, for simplicity, we assume $\mathbf{K}$ and $\mathbf{V}$ are compressed to the same rank, while in practice they can be compressed to different ranks according to the entropy distribution. It is divided into an encoding stage and an inference stage. During encoding, the KV cache for a many-shot context is partitioned into multiple blocks, each containing a specific number of examples. These blocks undergo SVD to determine an optimal rank allocation. By analyzing the entropy of each block, the system applies low-rank compression based on assigned rank budgets and stores the resulting data in a low-rank KV cache. In the subsequent inference stage, an input query is first evaluated by a retriever to perform KV selection from this compressed storage. The selected low-rank cache segments then undergo full-rank reconstruction in parallel. Finally, the reconstructed KV cache and the original input query are processed together through a sparse attention mechanism to complete the generation.

**Algorithm 1** MILO Workflow

**Input:** Many-shot context $\mathcal{C}$ with $N$ blocks $\{B_1, B_2, \ldots, B_N\}$, where each $B_i$ contains $m_i$ examples. Low-rank Cache $LC$. Input Query $\mathbf{Q}$.
**Output:** Attention outputs $\mathbf{O}$
*Stage 1: Encoding*
**for** $i == 1$ to $N$ **do**
  *# Perform SVD*
  $\mathbf{A}, \mathbf{\Sigma}, \mathbf{B}^\top = \text{SVD}((\mathbf{K}_i, \mathbf{V}_i))$
  *// Get Optimal Rank* $r_i$
  $r_i = \text{argmin}_{r_i} \mathbf{H}(\cdot)$
  *// Compress and Store*
  $LC.Store(\mathbf{A}[:r_i], \mathbf{\Sigma}[:r_i], \mathbf{B}[:r_i])$
**end for**
*Stage 2: Inference*
*# Process Query to the Retriever*
$(k_1, k_2, ..., k_N) = Retriver(\mathbf{Q})$
**for** $j == k_1$ to $k_N$ **do**
  *// Retrieve Selected Block*
  $\mathbf{A}_j, \mathbf{\Sigma}_j, \mathbf{B}_j = LC.Retreive(j)$
  *// Full-rank Reconstruction*
  $(\mathbf{K}_i, \mathbf{V}_i) \approx \mathbf{A}_i \mathbf{\Sigma}_i \mathbf{B}_i{}^\top$
**end for**
$\mathcal{O} = \text{Attn}(\mathbf{Q}, [\mathbf{K}_{k_1}, ...\mathbf{K}_{k_N}], [\mathbf{V}_{k_1}, ...\mathbf{V}_{k_N}])$
**return** $\mathcal{O}$

### 4.5 SYSTEM OPTIMIZATIONS

To translate MILO's memory savings into end-to-end speedup, we optimize the reconstruction path with fused Triton kernels, CUDA streams, and CUDA Graphs.

Table 1: Accuracy comparison of MILO against baseline methods on various many-shot ICL tasks. All experiments are conducted using 512-shot examples under the same rank budget of 50%.

| Model | Method | BANKING77 | CLINIC150 | TREC | NLU | MATHQA | SVAMP | Avg. |
|---|---|---|---|---|---|---|---|---|
| Qwen2.5-3B | Full Cache | 0.772 | 0.736 | 0.884 | 0.804 | 0.328 | 0.536 | 0.677 |
| | ASVD | 0.722 | 0.694 | 0.812 | 0.723 | 0.241 | 0.441 | 0.605 |
| | Palu | 0.740 | 0.712 | 0.856 | 0.768 | 0.264 | 0.484 | 0.637 |
| | **MILO** | 0.744 | 0.720 | 0.860 | 0.780 | 0.276 | 0.500 | 0.647 |
| Qwen2.5-7B | Full Cache | 0.812 | 0.808 | 0.932 | 0.860 | 0.420 | 0.684 | 0.753 |
| | ASVD | 0.785 | 0.742 | 0.837 | 0.815 | 0.311 | 0.594 | 0.680 |
| | Palu | 0.802 | 0.763 | 0.861 | 0.824 | 0.356 | 0.592 | 0.700 |
| | **MILO** | 0.804 | 0.794 | 0.894 | 0.834 | 0.394 | 0.632 | 0.725 |

**Fused Low-Rank Reconstruction.** A naive PyTorch-based implementation requires separate kernels for loading low-rank factors, applying singular-value scaling, reconstructing KV tensors, and writing them back to global memory. We instead implement a fused Triton kernel that performs cache retrieval, scaling, and reconstruction in a single pass. The kernel tiles the token and head dimensions, computes the reconstructed keys and values in registers and on-chip SRAM, and writes them directly to the KV buffer consumed by attention. This reduces both intermediate HBM traffic and intermediate kernel-launch overhead.

**Parallel Reconstruction with CUDA Streams.** Rather than reconstructing them sequentially on the default stream, MILO dispatches independent blocks to multiple CUDA streams to enable block-level parallelism and synchronizes only before sparse attention. This improves GPU utilization for small reconstruction kernels and reduces reconstruction latency on the critical path.

**CUDA Graph Execution.** We further use CUDA Graphs to reduce CPU launch overhead for repeated inference. Frequently occurring reconstruction and attention execution paths are captured and replayed for compatible request configurations, avoiding repeated kernel-launch overhead for short GPU operations.

# 5 Evaluation

## 5.1 Experimental Setup

**Models and Datasets.** We conduct experiments on Qwen family models, which are widely used for long-context understanding and reasoning tasks in real-world deployment. Specifically, we choose Qwen2.5-3B and Qwen2.5-7B Yang et al. (2024b). In terms of datasets, we focus on natural language understanding and mathematical reasoning tasks, including Banking77 Loukas et al. (2023), Clinic150 Larson et al. (2019), TREC Hovy et al. (2001), NLU Liu et al. (2019), MathQA Amini et al. (2019), and SVAMP Patel et al. (2021). We report model performance as the average accuracy scores of the above tasks. For system performance, we report both the memory and throughput results in an end-to-end fashion. All models are running in half-precision (FP16).

**Baselines.** We compare our method against standard full caching and state-of-the-art low-rank methods:

- **Full Cache:** We use the standard uncompressed KV cache as the accuracy performance upper bound and system efficiency lower bound.
- **ASVD (example-based)** Yuan et al. (2023): ASVD performs activation-aware SVD and compresses it at the granularity of an example for each incoming query.
- **Palu (global-based)** Chang et al. (2024): Palu employs a low-rank projection matrix and applies KV cache compression to the global context.

**Implementation Details.** We conduct our experiments on a single NVIDIA L4-24 GB GPU, with a 2.60 GHz Intel Xeon CPU with 512 GB DRAM and PCIe Gen-4. Our method is implemented on top of FlexGen Sheng et al. (2023), Transformers Wolf et al. (2020), PyTorch Paszke et al. (2019), and

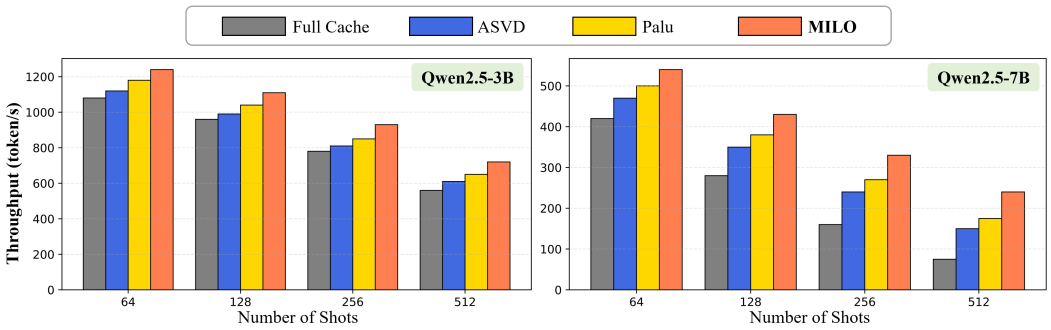


Figure 4: End-to-end system performance comparison of MILO against prior methods on the MathQA dataset under the same rank budget of 50% on a single NVIDIA L4 GPU.

Triton Tillet et al. (2019). In terms of selecting demonstration examples, we follow prior works Xiao et al. (2025); Ou et al. (2026) and use the standard BM25 retriever Robertson & Zaragoza (2009).

### 5.2 Main Results

**Accuracy Results.** Table 1 reports the detailed accuracy results of our method and baseline methods. Under an identical rank budget constraint, our method outperforms prior state-of-the-art low-rank methods, improving the accuracy performance by up to 4.5%. Specifically, on Qwen2.5-3B, MILO demonstrates a consistent lead across all individual tasks, substantially narrowing the empirical performance gap to the accuracy performance upper bound. A similar trend can be observed on the larger Qwen2.5-7B model, where MILO retains a greater proportion of the full cache representation capacity, significantly outperforming SOTA methods.

**Throughput Results.** We further evaluate the system performance of MILO under a memory-constrained deployment. Since the KV cache often cannot remain resident in GPU memory, we consider a KV offloading configuration where pre-encoded cache blocks for many-shot examples are stored in CPU memory and transferred to the GPU on demand during query processing Sheng et al. (2023); Zhao et al. (2024). Figure 4 reports the end-to-end throughput comparison. Here, we have two key observations. First, compared with full KV offloading, MILO achieves an average speedup of 1.6×, thanks to the reduced KV volume by performing low-rank compression and efficient kernel implementation. For instance, for the 7B model with 512-shot inputs, MILO can sustain up to 3.2× more throughput. Second, MILO outperforms Palu and ASVD by up to 37% and 60%, respectively. Third, the larger the model size, the higher the improvement MILO brings. This is due to the increased KV cache size transferred during offload, thus making the inference more memory-bound.

**Combining KV Quantization.** Here, we further apply asymmetric INT8 Sheng et al. (2023) and KIVI Liu et al. (2024) quantization on top of MILO and evaluate the impact on model accuracy and KV cache memory reduction ratio. Table 2 shows that MILO can be effectively combined with quantization methods, providing complementary memory savings while preserving model accuracy.

Table 2: Accuracy and KV-cache memory reduction results when combining MILO with quantization.

| Model | Method | KV Reduction | Accuracy |
|---|---|---|---|
| Qwen2.5-3B | Full Cache | 1× | 0.677 |
| | MILO | 2× | 0.647 |
| | MILO + INT8 | 4× | 0.621 |
| | MILO + KIVI | 4.7× | 0.608 |
| Qwen2.5-7B | Full Cache | 1× | 0.753 |
| | MILO | 2× | 0.725 |
| | MILO + INT8 | 4× | 0.712 |
| | MILO + KIVI | 5.3× | 0.701 |

### 5.3 Performance Analysis

**Impact of Rank Budget and Block Size.** Figure 5 (a) and (b) demonstrate the sensitivity of MILO to the compression ratio and block size (number of examples per block). Here, we have two key observations. First, as the compression ratio increases, all low-rank methods experience gradual accuracy degradation due to the reduced rank budget. However, MILO consistently maintains higher accuracy than ASVD and Palu across the entire range. This indicates that the proposed block-wise

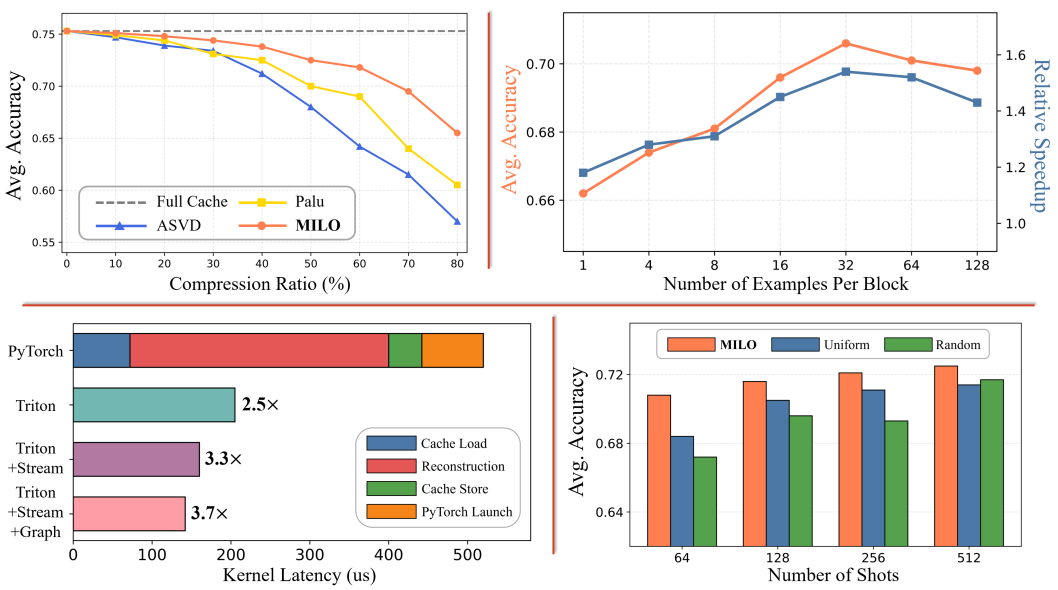


Figure 5: Performance Analysis. Top Left (a): Impact of compression ratio (%) on model performance for MILO and baseline methods; Top Right (b): Impact of block size (number of examples per block) on latency and model accuracy for MILO.; Bottom Left (c): Performance breakdown for MILO customized Triton kernels; Bottom Right (d): Impact of rank allocation strategy on model accuracy for MILO. All experiments are conducted using Qwen2.5-7B.

adaptive allocation preserves more task-relevant information under constrained rank budgets. Second, increasing the block size initially improves both model accuracy and system efficiency, as larger blocks expose more cross-example redundancy that can be captured by the low-rank representation. However, both accuracy and speedup begin to decline after a certain block size (32), suggesting that overly large blocks reduce compression and retrieval granularity while increasing reconstruction overhead.

**Kernel Performance Analysis.** We further evaluate the effectiveness of the system optimizations for MILO, as shown in Figure 5 (c). The native PyTorch implementation incurs substantial latency from separate cache loading, low-rank reconstruction, cache storage, and repeated kernel launches. Replacing these operations with our fused Triton implementation reduces kernel latency by approximately $2.5\times$, primarily by eliminating intermediate HBM traffic. Enabling parallel block reconstruction with CUDA streams further improves the speedup to approximately $3.3\times$ and combining with CUDA graph achieves a total speedup of approximately $3.7\times$ over the PyTorch implementation.

**Impact of Rank Allocation.** We also compare the proposed entropy-based rank allocation strategy against uniform and random allocation under the same total rank budget in Figure 5 (d). We can see that MILO consistently achieves the highest model accuracy. As the number of shots increases, the performance gap narrows, but MILO continues to provide the strongest accuracy, demonstrating that adaptive rank allocation makes more effective use of a fixed compression budget.

## 6 Conclusion

In this work, we present MILO, a block-wise low-rank KV-cache compression framework for efficient many-shot in-context learning. By exploiting heterogeneous low-rank redundancy across demonstration blocks and dynamically allocating a global rank budget, MILO reduces KV cache memory footprint by 50% while preserving model accuracy. Combining with customized fused Triton kernels, MILO further improves the system performance, achieving up $1.8\times$ speedup upon existing baseline methods, demonstrating a practical path toward efficient many-shot ICL inference.

## AI USE STATEMENT

In this work, we have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. Additionally, we used generative AI tools for **edit a research paper to improve readability** and **propose a title or keywords for a research paper**. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal M. P. Behbahani, Aleksandra Faust, and Hugo Larochelle. Many-shot in-context learning. *ArXiv*, abs/2404.11018, 2024.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *North American Chapter of the Association for Computational Linguistics*, 2019.

an Luo, Xin Xu, Yue Liu, Panupong Pasupat, and Mehran Kazemi. In-context learning with retrieved demonstrations for language models: A survey. *ArXiv*, abs/2401.11624, 2024.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

Kwan Ho Ryan Chan, Yaodong Yu, Chong You, Haozhi Qi, John Wright, and Yi Ma. Redunet: A white-box deep network from the principle of maximizing rate reduction. *ArXiv*, abs/2105.10446, 2021.

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: Kv-cache compression with low-rank projection. In *International Conference on Learning Representations*, 2024.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. A survey on in-context learning. In *Conference on Empirical Methods in Natural Language Processing*, 2022.

Shahriar Golchin, Yanfei Chen, Rujun Han, Manan S. Gandhi, Tianli Yu, Swaroop Mishra, Mihai Surdeanu, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Towards compute-optimal many-shot in-context learning. *ArXiv*, abs/2507.16217, 2025.

Qingqing Gu, Dan Wang, Yue Zhao, Xiaoyu Wang, Zhonglin Jiang, Yong Chen, Hongyan Li, and Luo Ji. Chain-of-conceptual-thought elicits daily conversation in large language models. In *Pacific Rim International Conference on Artificial Intelligence*, 2025.

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*, abs/2401.18079, 2024.

Eduard H. Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. Toward semantics-based answer pinpointing. In *Human Language Technology - The Baltic Perspective*, 2001.

Yen-Chang Hsu, Ting Hua, Sung-En Chang, Qiang Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. *ArXiv*, abs/2207.00112, 2022.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021.

Edwin T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Jia Xu, Linjian Mo, and Min Zhang. A survey of test-time compute: From intuitive inference to deliberate reasoning. 2025.

Yixing Jiang, Jeremy Irvin, Ji Hun Wang, Muhammad Ahmed Chaudhry, Jonathan H. Chen, and Andrew Y. Ng. Many-shot in-context learning in multimodal foundation models. *ArXiv*, abs/2405.09798, 2024.

Stefan Larson, Anish Mahendran, Joseph Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. *ArXiv*, abs/1909.02027, 2019.

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. *ArXiv*, abs/2410.14731, 2024.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. Benchmarking natural language understanding services for building conversational agents. *ArXiv*, abs/1903.05566, 2019.

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. *ArXiv*, abs/2305.17118, 2023.

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*, abs/2402.02750, 2024.

Lefteris Loukas, Ilias Marios Stogiannidis, Prodromos Malakasiotis, and Stavros Vassos. Breaking the bank with chatgpt: Few-shot text classification for finance. *ArXiv*, abs/2308.14634, 2023.

Meta. Introducing llama 3.1: Our most capable models to date, 2024. URL `https://ai.meta.com/blog/meta-llama-3-1/`.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Fei-Fei Li, Hanna Hajishirzi, Luke S. Zettlemoyer, Percy Liang, Emmanuel J. Candes, and Tatsunori Hashimoto. s1: Simple test-time scaling. *ArXiv*, abs/2501.19393, 2025.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *North American Chapter of the Association for Computational Linguistics*, pp. 6151–6162, 2019.

Jie Ou, Jinyu Guo, Shiyao Guo, Yuang Li, Ruiqi Wu, Zhaokun Wang, Wenyi Li, and Wenhong Tian. Adapshot: Adaptive many-shot in-context learning with semantic-aware kv cache reuse. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42946–42960, 2026.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Jun-Jie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *ArXiv*, abs/1912.01703, 2019.

Arkil Patel, S. Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *North American Chapter of the Association for Computational Linguistics*, 2021.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389, 2009.

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

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark W. Barrett, Joseph Gonzalez, Percy Liang, Christopher Ré, Ioan Cristian Stoica, and Ce Zhang. High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pp. 31094—-31116. PMLR, 2023.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *ArXiv*, abs/2408.03314, 2024.

Mingyang Song, Mao Zheng, and Xuan Luo. Can many-shot in-context learning help llms as evaluators? a preliminary empirical study. In *International Conference on Computational Linguistics*, 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*, abs/2410.21465, 2024.

Philippe Tillet, Hsiang-Tsung Kung, and David D. Cox. Triton: an intermediate language and compiler for tiled neural network computations. *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2019.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Emily Xiao, Chin-Jou Li, Yilin Zhang, Graham Neubig, and Amanda Bertsch. Efficient many-shot in-context learning with dynamic block-sparse attention. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *ArXiv*, abs/2309.17453, 2023.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. Qwen2 technical report. *ArXiv*, abs/2407.10671, 2024a.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024b.

Hao Yu, Zelan Yang, Shen Li, Yong Li, and Jianxin Wu. Effectively compress kv heads for llm. *ArXiv*, abs/2406.07056, 2024a. URL https://api.semanticscholar.org/CorpusID:270380364.

Hao Yu, Zelan Yang, Shen Li, Yong Li, and Jianxin Wu. Effectively compress kv heads for llm. *ArXiv*, abs/2406.07056, 2024b.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models. *ArXiv*, abs/2312.05821, 2023.

Rongzhi Zhang, Kuan 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*, abs/2410.03111, 2024.

Zhenyu (Allen) 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. *ArXiv*, abs/2306.14048, 2023.

Youpeng Zhao, Di Wu, and Jun Wang. Alisa: Accelerating large language model inference via sparsity-aware kv caching. *ArXiv*, abs/2403.17312, 2024.

Youpeng Zhao, Ming Lin, Huadong Tang, Qiang Wu, and Jun Wang. Merino: Entropy-driven design for generative language models on iot devices. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 22840–22848, 2025a.

Youpeng Zhao, Jinpeng Lv, Di Wu, Jun Wang, and Chris Gooley. Are we scaling the right thing? a system perspective on test-time scaling. *ArXiv*, abs/2509.19645, 2025b.