Eigen Attention: Attention in Low-Rank Space for KV Cache Compression

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, Kaushik Roy
Purdue University
`{saxenau, gsaha, choudh23, kaushik}@purdue.edu`

Abstract

Large language models (LLMs) represent a groundbreaking advancement in the domain of natural language processing due to their impressive reasoning abilities. Recently, there has been considerable interest in increasing the context lengths for these models to enhance their applicability to complex tasks. However, at long context lengths and large batch sizes, the key-value (KV) cache, which stores the attention keys and values, emerges as the new bottleneck in memory usage during inference. To address this, we propose Eigen Attention, which performs the attention operation in a low-rank space, thereby reducing the KV cache memory overhead. Our proposed approach is orthogonal to existing KV cache compression techniques and can be used synergistically with them. Through extensive experiments over OPT, MPT, and Llama model families, we demonstrate that Eigen Attention results in up to 40% reduction in KV cache sizes and up to 60% reduction in attention operation latency with minimal drop in performance. Code is available at https://github.com/UtkarshSaxena1/EigenAttn.

1 Introduction

The recent boom in artificial intelligence applications and their widespread public adoption can be attributed to the human-like capabilities of large language models (LLMs). LLMs have demonstrated remarkable performance across a wide range of natural language processing (NLP) tasks (lmsys.org, 2024). The maximum number of tokens these models can process simultaneously to understand and generate text is referred to as the context/sequence length, and this closely determines their performance limits. Hence, there has been considerable interest in increasing the context length to enhance their capabilities (Zhang et al., 2024a; Ding et al., 2024; Achiam et al., 2023). Longer context lengths open up new possibilities, such as summarizing lengthy documents, retrieving information to answer questions about extensive texts, and analyzing code. To make applications enabled by LLMs more accessible, developing techniques to serve these models efficiently is crucial. One standard technique to accelerate LLM inference on GPUs is caching the intermediate attention keys and values through a KV cache to avoid expensive re-computation for every generated token. However, it is observed that at long context lengths, the KV cache becomes the new memory and latency bottleneck (Pope et al., 2022). Furthermore, while batching multiple requests together amortizes the weight access cost of LLMs, it exacerbates the KV cache memory overhead. Consider the weight and KV cache memory footprint for the Llama-2 7B model with a batch size of 16 and a context length of 32k tokens. Here, the weight memory occupies 14 GB, while the KV cache requires a significantly higher memory of 256 GB at 16-bit precision. In essence, increasing the context/sequence length of LLMs has transformed LLM inference into a memory-bound problem. The entire KV cache must be bought on-chip from the off-chip GPU memory for each newly generated token while the computation core stays idle, waiting for data.

Existing methods to address the KV cache bottleneck can be broadly classified into four distinct categories. First, some approaches focus on reducing the number of KV cache heads in the multi-head attention block through grouped query and multi-query attention (Ainslie et al., 2023; Shazeer, 2019). Second, a few methods aim to alleviate the memory overhead by utilizing a low-precision quantized KV cache (Hooper et al., 2024; Zirui Liu et al., 2024). Third, certain strategies involve evicting KV cache values associated with unimportant tokens, thereby caching only the keys and values of important tokens determined through some metrics (Zhang et al., 2023; Adnan et al., 2024). Finally,

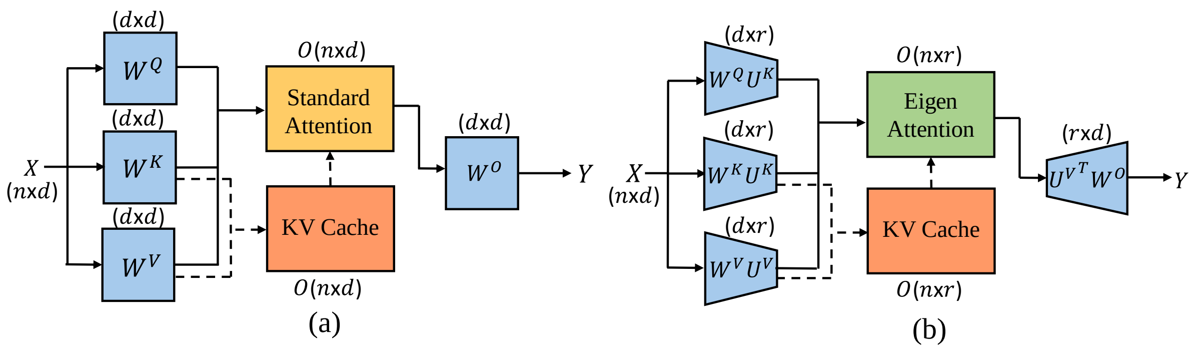


Figure 1: Comparison between (a) Standard Attention and (b) Eigen Attention. Eigen Attention utilizes lower dimensional ($r \ll d$) query, key, and value projection matrices than the standard attention operation, leading to KV cache compression and compute FLOPs benefits.

Table 1: KV cache size, number of parameters, and total FLOPs for Standard vs Eigen Attention computed for sequence length n, hidden dimension d in standard attention and hidden dimension r ($\ll d$) in Eigen Attention.

	Standard Attention	Eigen Attention
KV Cache Size	$2.n.d$	$2.n.r$
# Parameters	$4.d^2$	$4.d.r$
FLOPs (generation phase)	$4.d^2 + 2.n.d$	$4.d.r + 2.n.r$

some approaches tackle the issue from a systems perspective by utilizing the CPU and disk memory for the KV cache or by extending virtual memory and incorporating paging techniques into the attention mechanism (Kwon et al., 2023a).

In contrast to the above-mentioned techniques, we propose Eigen Attention, a strategy to alleviate the KV cache overhead through low-rank approximation. Eigen Attention leverages the observation that attention inputs in LLMs (i.e., key, query, and value) can be reasonably approximated using a few principal basis vectors or eigenvectors (Yu and Wu, 2023). Expressing keys and values as a linear combination of these principal vectors essentially reduces their dimension, leading to a lower memory footprint of the KV cache. To achieve this, we use a very small subset of training data as a calibration dataset to generate a set of query, key, and value matrices for the trained model. Subsequently, we obtain the basis vectors through Singular Value Decomposition (SVD) on these matrices and choose the most important directions through a pre-defined error threshold. Eigen Attention is a post-training technique that can be applied without requiring any additional fine-tuning. Moreover, our approach is orthogonal to existing techniques for KV cache compression and can be used in conjunction with them. Figure 1 illustrates the differences between standard attention and our proposed Eigen Attention. Columns of the $\mathbf{U}^K$ matrix are the principal basis vectors for keys and queries. Similarly, $\mathbf{U}^V$ contains the important basis vectors for the value matrix. We project the attention inputs into a low-rank space defined by the eigenvectors $\mathbf{U}^K$ and $\mathbf{U}^V$ by assimilating these projections into the corresponding weight matrices $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$ and $\mathbf{W}^O$ offline. During inference, the low-dimensional $\mathbf{W}^K\mathbf{U}^K$ and $\mathbf{W}^V\mathbf{U}^V$ are multiplied with the input vector $\mathbf{X}$ to generate lower-dimensional K and V matrices. While the primary goal of Eigen Attention is to reduce the memory overhead of the KV cache, the proposed low-rank approximation also leads to a compute-efficient implementation of the attention block (Section 5.2). For the standard dimension d and the compressed dimension $r(\ll d)$, Table 1 shows the reduction in KV cache size and floating point operations (FLOPs) achieved through Eigen Attention. In summary, we make the following contributions:

- We propose Eigen Attention, a novel mechanism to efficiently serve LLMs by compressing the KV cache through low-rank approximations.

- We demonstrate that the low-rank approximation employed by Eigen Attention enhances the compute efficiency of the attention block.

- Extensive experiments across various models and language tasks show that Eigen Attention compresses the KV cache by up to 40% along with upto 60% reduction in the attention operation latency.

2 Background

Multi-Head Attention. A typical LLM consists of L decoder layers, each with two components: multi-head attention (MHA) and the fully connected feed-forward network (FFN). For an input token embedding $\mathbf{X} \in \mathbb{R}^{n \times d}$, the MHA block performs attention operations in parallel across h heads. Here, n is the sequence length, and d is the hidden dimensionality of the model. For each attention head $i \in \{1, 2, ...h\}$, $\mathbf{X}$ is transformed into key, query, and value matrices as follows:

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q, \;\; \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K, \;\; \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V \quad (1)$$

Here, $\mathbf{W}_i^Q \in \mathbb{R}^{d \times d_h}, \mathbf{W}_i^K \in \mathbb{R}^{d \times d_h}, \mathbf{W}_i^V \in \mathbb{R}^{d \times d_h}$ are learnable weight matrices with $d_h = \frac{d}{h}$. The attention operation $\mathbf{A}$ at each head i is computed, and the results from all heads are concatenated to obtain the final output of the MHA block, as shown in Equation 2.

$$\begin{aligned} \mathrm{h}_i &= \mathbf{A}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \mathrm{Softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_h}}\right)\mathbf{V}_i, \\ \mathrm{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= [\mathrm{h}_1, \mathrm{h}_2, .., \mathrm{h}_h]\mathbf{W}^O \end{aligned} \quad (2)$$

LLM Inference. The inference consists of the prefill and generation phases. In the prefill phase, the keys $\mathbf{K}_i$ and $\mathbf{V}_i$ values are computed for an input token embedding $\mathbf{X} \in \mathbb{R}^{b \times n \times d}$ with batch size b (as shown in Equation 1) and cached in memory for the generation phase. The total size of KV cache (in bits) can be derived by $2*b*n*d*h*L*p$, where L corresponds to the number of decoder layers in the LLM, and p corresponds to the precision of cached vectors.

In the generation phase, the model uses and updates the KV cache to generate the output autoregressively, one token at a time. Note that this phase is memory-bound. For an incoming token embedding $\mathbf{x} \in \mathbb{R}^{b \times 1 \times d}$, the key $\mathbf{k}_i$ and value $\mathbf{v}_i$ computed through Equation 1 are appended to the KV cache:

$$\mathbf{K}_i \leftarrow \mathrm{Concat}(\mathbf{K}_i, \mathbf{k}_i), \mathbf{V}_i \leftarrow \mathrm{Concat}(\mathbf{V}_i, \mathbf{v}_i)$$

The query $\mathbf{q}_i$ obtained through Equation 1 is used to compute the attention output for each head:

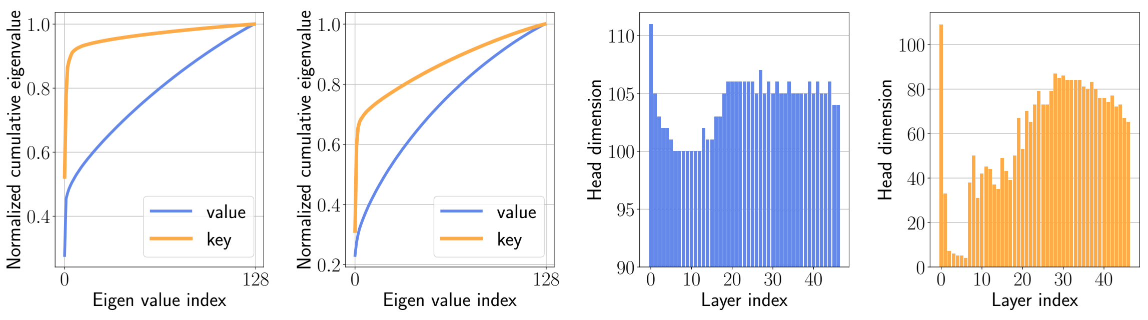
$$\mathrm{h}_i = \mathrm{Softmax}\left(\mathbf{q}_i\mathbf{K}_i^T / \sqrt{d_h}\right)\mathbf{V}_i \quad (3)$$

Then, similar to Equation 2, the heads' outputs are concatenated and multiplied with $\mathbf{W}^O$ to produce the final output of the attention block.

3 Related Works

KV Cache Compression. As mentioned in Section 2, the KV cache size is computed as $2*b*n*d*h*L*p$. KV cache compression methods target these factors to reduce its overall memory footprint. Multi-query attention (Shazeer, 2019) and grouped query attention (Ainslie et al., 2023) reduce the number of attention heads h. Quantization based methods reduce the precision p (Yang et al., 2024; Kang et al., 2024; Zirui Liu et al., 2024; Hooper et al., 2024). Notably, works like KIVI (Zirui Liu et al., 2024) and KV Quant (Hooper et al., 2024) have demonstrated that the precision of $\mathbf{K}$ and $\mathbf{V}$ matrices can be reduced to as low as 2-bits. Several works attempt to reduce the sequence length n by only caching $\mathbf{K}$ and $\mathbf{V}$ corresponding to a subset of tokens (Beltagy et al., 2020). Another strategy is to cache K and V according to token importance (Zhang et al., 2023; Adnan et al., 2024; Liu et al., 2024; Niu et al., 2024; Li et al., 2024; Zhang et al., 2024b). H_2O (Zhang et al., 2023) finds important tokens by monitoring accumulated attention scores. KeyFormer (Adnan et al., 2024) improves H_2O by considering the importance of discarded tokens as well. Recently, (Wu and Tu, 2024) demonstrated that the cached $\mathbf{K}$ and $\mathbf{V}$ can be reused across the decoder layers, essentially reducing L. In contrast to these techniques, Eigen Attention reduces the embedding dimension d of each cached $\mathbf{K}$ and $\mathbf{V}$ vector. It is orthogonal to the existing KV cache compression techniques and can be used in conjunction with them.

Low-Rank Approximation. Various works in literature have leveraged low-rank approximation for performing efficient LLM inference. Recent works (Yu and Wu, 2023; Feng et al., 2022) have shown that while the weight matrices for transformers-based models are not inherently sparse, the activations are. LoRD (Kaushal et al., 2023) leverages this observation to compress the weight matrix of LLMs by representing it as a product of two low-rank matrices. LoSparse (Li et al., 2023) combines pruning and low-rank approximation and expresses the weight matrix as a sum of a low-rank matrix and a sparse matrix. Fisher-weighted SVD (Hsu et al., 2022) proposes to utilize Fisher information to weigh the importance of weights before performing SVD-based low-rank



a Layer #15 Head #16 b Layer #25 Head #16 c Value d Key

Figure 2: Eigenvalue spectrum analysis for OPT-30b model. (a), (b) The Y-axis is the normalized cumulative eigenvalue value after performing SVD on the key value representation matrix, and the X-axis is an index of the largest eigenvalue. (c), (d) Dimensions of the low-rank matrices with normalized cumulative eigenvalue of 0.9.

approximation. In this work, we focus on reducing the KV cache's memory footprint by storing low-dimensional keys and values to efficiently enable long sequence lengths for LLMs.

4 Methodology

This section describes Eigen Attention, which achieves KV cache compression by performing the attention operation in a low-rank space determined by a few basis vectors. We first discuss generating these principal basis vectors and then describe our proposed approach detailed in Algorithm 1.

4.1 Basis Vector Generation

The first step towards computing attention in low-rank space is determining the principle basis vectors that span this low-dimensional space. To achieve this, we create representation matrices $\mathbf{R}^{Q}_{l,i}$, $\mathbf{R}^{K}_{l,i}$ and $\mathbf{R}^{V}_{l,i}$ corresponding to query, key, and value respectively, for each layer l and attention head i. We drop the layer index l for ease of clarity. These representation matrices are obtained by performing a forward pass for n_s samples taken from the calibration dataset, as shown in Equation 4. Note that this calibration dataset is a subset of WikiText (Merity et al., 2016), which is commonly used for pretraining LLMs.

$$\begin{aligned}\mathbf{R}^{Q}_{i} &= [(\mathbf{Q}^{1}_{i})^{T}, (\mathbf{Q}^{2}_{i})^{T}, ..., (\mathbf{Q}^{n_s}_{i})^{T}]\\ \mathbf{R}^{K}_{i} &= [(\mathbf{K}^{1}_{i})^{T}, (\mathbf{K}^{2}_{i})^{T}, ..., (\mathbf{K}^{n_s}_{i})^{T}]\\ \mathbf{R}^{V}_{i} &= [(\mathbf{V}^{1}_{i})^{T}, (\mathbf{V}^{2}_{i})^{T}, ..., (\mathbf{V}^{n_s}_{i})^{T}]\end{aligned} \tag{4}$$

Here, $\mathbf{R}^{Q/K/V}_{i} \in \mathbb{R}^{(n_s.n)\times d_h}$. The key and query representation matrices are concatenated as $\mathbf{R}^{KQ}_{i} = [\mathbf{R}^{Q}_{i}, \mathbf{R}^{K}_{i}]$ followed by an SVD operation $\mathbf{R}^{KQ}_{i} \xrightarrow{\text{SVD}} \Sigma^{d_h}_{i=1}\sigma_i u_i v^{T}_{i}$. We obtain a r-rank approximation $(\mathbf{R}^{KQ}_{i})_r$ according to the following criteria (Saha et al., 2021),

$$||(\mathbf{R}^{KQ}_{i})_r||^2_F \geq \epsilon_{th}||\mathbf{R}^{KQ}_{i}||^2_F \tag{5}$$

ϵ_{th} is the threshold hyperparameter determining the degree of low-rank approximation. We create a unified basis for key and query to aid in low-rank attention, as shown in Section 4.2. The orthogonal basis vectors spanning the low-rank space for key and query are given by $\mathbf{U}^{KQ}_{i} = [u_1, u_2, ..., u_r]$, which we represent concisely as $\mathbf{U}^{K}_{i}$. Similarly, we generate $\mathbf{U}^{V}_{i}$ for the value representation matrix $\mathbf{R}^{V}_{i}$. Please refer to Appendix A.1 for more details on SVD. Note that a unique low-rank basis is obtained for each head in the attention layer, and we keep the rank r the same across heads by taking the maximum rank across heads. Figure 2 (a), (b) show the spectrum of eigenvalue distribution of the keys and values for OPT-30b, with keys having a lower rank than the values. In Figure 2 (c), (d), we plot rank r obtained by keeping ϵ_{th} as 0.9 for different layers of the model. As shown, some layers' dimensions can be reduced to nearly zero.

4.2 Eigen Attention

To understand our proposed low-rank attention, consider the basis vectors $\mathbf{U}^{K}_{i}, \mathbf{U}^{V}_{i}$ such that $(\mathbf{U}^{K/V}_{i})^{T}.\mathbf{U}^{K/V}_{i} = \mathbf{I}$ due to orthogonality. The attention inputs can be projected to the low-rank space spanned by these vectors as,

$$\begin{aligned}\mathbf{Q}'_{i} &= \mathbf{Q}_{i}\mathbf{U}^{K}_{i}(\mathbf{U}^{K}_{i})^{T}\\ \mathbf{K}'_{i} &= \mathbf{K}_{i}\mathbf{U}^{K}_{i}(\mathbf{U}^{K}_{i})^{T}\\ \mathbf{V}'_{i} &= \mathbf{V}_{i}\mathbf{U}^{V}_{i}(\mathbf{U}^{V}_{i})^{T}\end{aligned} \tag{6}$$

Here, $\{\mathbf{Q}'_{i}, \mathbf{K}'_{i}, \mathbf{V}'_{i}\} \in \mathbb{R}^{n\times d_h}$ are low rank attention inputs with rank r. Then, we compute Eigen

Attention as follows:

$$\begin{aligned}\mathbf{A}' &= \text{Softmax}(\frac{\mathbf{Q}_i'\mathbf{K}_i'^T}{\sqrt{d_h}})\mathbf{V}_i' \\ &= \text{Softmax}(\frac{\mathbf{Q}_i\mathbf{U}_i^K(\mathbf{U}_i^K)^T\mathbf{K}^\mathbf{T}}{\sqrt{d_h}})\mathbf{V}_i\mathbf{U}_i^V(\mathbf{U}_i^V)^T\end{aligned} \tag{7}$$

For an appropriate rank r, $\mathbf{U}_i^{K/V}(\mathbf{U}_i^{K/V})^T \approx \mathbf{I}$ (Yu and Wu, 2023). Hence, attention with low-rank inputs (i.e., Eigen Attention) can approximate the full rank attention ($\mathbf{A}' \approx \mathbf{A}$). Further, the basis vectors $\mathbf{U}_i^K$ and $\mathbf{U}_i^V$ used to compute Eigen Attention can be seamlessly merged with the weight projection matrices of the attention layer:

$$\begin{aligned}\mathbf{W}_i^{Q'} &\leftarrow \mathbf{W}_i^Q\mathbf{U}_i^K \\ \mathbf{W}_i^{K'} &\leftarrow \mathbf{W}_i^K\mathbf{U}_i^K \\ \mathbf{W}_i^{V'} &\leftarrow \mathbf{W}_i^V\mathbf{U}_i^V \\ \mathbf{W}_i^{O'} &\leftarrow (\mathbf{U}_i^V)^T\mathbf{W}_i^O\end{aligned} \tag{8}$$

Here, $\{\mathbf{W}_i^{Q'}, \mathbf{W}_i^{K'}, \mathbf{W}_i^{V'}\} \in \mathbb{R}^{d_h \times r}$ and $\mathbf{W}_i^{O'} \in \mathbb{R}^{r \times d_h}$. This transformation reduces the output dimension of projection matrices, effectively decreasing the embedding dimension of keys, queries, and values. Consequently, both the number of parameters and floating-point operations (FLOPs) in the attention layer are reduced. More importantly, Eigen Attention significantly lowers the KV cache memory footprint (refer Table 1).

4.3 Rotational Position Embedding

Positional information can be incorporated in text processed by LLMs in several ways. Different models employ absolute or relative positional embeddings at different levels of model computations. Recent LLM demonstrations employ rotary positional embeddings (RoPE) (Su et al., 2024). RoPE transforms the keys and queries before performing the attention operation as shown below:

$$\mathbf{Q}_i^{pos} = \mathbf{Q}_i\mathbf{R}; \quad \mathbf{K}_i^{pos} = \mathbf{K}_i\mathbf{R} \tag{9}$$

LLMs with RoPE are trained with a fixed dimensional $\mathbf{R}$, making them incompatible with any modification to the embedding dimension of the keys or queries. To integrate RoPE with Eigen Attention, we introduce minor modifications. Specifically, we leave the query to be full rank and transform the key back to a high dimension before applying the RoPE rotation matrix. The query, key dot product with Eigen Attention is given by,

$$\mathbf{Q}_i^{pos}(\mathbf{K}_i^{pos})^T = \mathbf{Q}_i\mathbf{R}\mathbf{R}^T(\mathbf{U}^K)^T(\mathbf{U}^K\mathbf{K}_i) \tag{10}$$

We store the low-dimensional representation of the key (i.e., $\mathbf{U}^K\mathbf{K}_i$) in the KV cache but perform an additional transformation through $(\mathbf{U}^K)^T$ before applying RoPE (Figure 6). To mitigate the parameter overhead associated with this additional transformation, we propose to share $\mathbf{U}^K$ across all the attention heads. Similar to the standard Eigen Attention, we merge $\mathbf{U}^K$ into $\mathbf{W}_i^K$, with the value computation unchanged.

Algorithm 1 Eigen Attention

Input: LLM Decoder layers $\{\mathbf{L}\}_{l=1}^L$, error budget e_b, step size ϵ_s and inputs to first decoder X_1.

```
1:  procedure EIGENATTENTION()
2:      for $l = 1...L$ do
3:          $\mathbf{X}_{l+1}, \mathbf{R}^{KQ}, \mathbf{R}^V \leftarrow \texttt{forward}(\mathbf{L}_l, X_l)$
4:          $\epsilon_{th}^l \leftarrow 1.0$
5:          while $e \leq e_b$ do
6:              $\epsilon_{th}^l \leftarrow \epsilon_{th}^l - \epsilon_s$
7:              $\mathbf{U}_l^K \leftarrow \texttt{SVD}(\mathbf{R}_l^{KQ}, \epsilon_{th}^l)$
8:              $\mathbf{U}_l^V \leftarrow \texttt{SVD}(\mathbf{R}_l^V, \epsilon_{th}^l)$          ▷ eq.5
9:              $\mathbf{W}_l^Q \leftarrow \mathbf{W}_l^Q\mathbf{U}_l^K$
10:             $\mathbf{W}_l^K \leftarrow \mathbf{W}_l^Q\mathbf{U}_l^K$
11:             $\mathbf{W}_l^V \leftarrow \mathbf{W}_l^Q\mathbf{U}_l^V$
12:             $\mathbf{W}_l^O \leftarrow (\mathbf{U}_l^V)^T\mathbf{W}_l^Q$          ▷ eq. 8
13:             $\mathbf{X}_{l+1}' \leftarrow \texttt{forward}(\mathbf{L}_l, X_l)$
14:             $e_b = \frac{||\mathbf{X}_{l+1}' - \mathbf{X}_{l+1}||^2}{||\mathbf{X}_{l+1}||^2}$          ▷ error
15:         end while
16:     end for
17:     return $\{\mathbf{L}\}_{l=1}^L$          ▷ Low-rank Layers
18: end procedure
```

4.4 Layer-wise Rank Allotment

Eigen Attention introduces layer-wise threshold ϵ_{th}, which determines the accuracy of low-rank approximation according to Equation 5. We observe that the same ϵ_{th} across attention layers introduces different errors at the output of the LLM decoder layer. This implies that the layers that incur lower errors due to the low-rank approximation can be further compressed by lowering the ϵ_{th}. To achieve this, we introduce a layer-wise threshold selection methodology based on the normalized output error of each decoder layer. The threshold ϵ_{th} is reduced by a step size ϵ_s until the decoder layer output

Table 2: Perplexity (PPL, lower is better) results on Wikitext and C4 and Accuracy (Acc, higher is better) results on lm-eval-harness tasks: PiQA, WinoGrande, Arc-e, Arc-c, HellaSwag. Results are evaluated at different levels of KV cache compression obtained by Eigen Attention. The baseline represents standard attention with an uncompressed KV cache. The performance degradation from the baseline is shown in brackets.

Model	Params	KV Cache	PPL ↓		Acc ↑					
			Wikitext	C4	PiQA	WinoG	Arc-e	Arc-c	HellaS	Avg-Acc
OPT	30b	Baseline	9.56	10.69	0.78	0.68	0.70	0.35	0.54	0.61
		0.8x	9.61 (+0.05)	10.75 (+0.06)	0.78	0.67	0.70	0.34	0.54	0.61 (-0.00)
		0.7x	9.94 (+0.38)	10.98 (+0.29)	0.78	0.66	0.69	0.35	0.54	0.60 (-0.01)
		0.6x	10.80 (+1.24)	11.47 (+0.78)	0.76	0.64	0.69	0.32	0.52	0.59 (-0.02)
	66b	Baseline	9.34	10.28	0.79	0.69	0.72	0.37	0.56	0.63
		0.8x	9.36 (+0.02)	10.29 (+0.01)	0.79	0.69	0.72	0.37	0.56	0.63 (-0.00)
		0.7x	9.51 (+0.17)	10.40 (+0.12)	0.78	0.68	0.72	0.37	0.56	0.62 (-0.01)
		0.6x	9.68 (+0.34)	10.54 (+0.26)	0.78	0.68	0.70	0.36	0.55	0.62 (-0.01)
MPT	7b	Baseline	7.68	9.60	0.79	0.69	0.75	0.41	0.57	0.64
		0.8x	8.02 (+0.34)	10.14 (+0.54)	0.79	0.68	0.74	0.39	0.56	0.63 (-0.01)
		0.7x	8.45 (+0.77)	10.80 (+1.20)	0.78	0.68	0.72	0.38	0.54	0.62 (-0.02)
		0.6x	9.61 (+1.93)	12.26 (+2.66)	0.76	0.68	0.69	0.35	0.52	0.60 (-0.04)
	30b	Baseline	6.40	8.44	0.80	0.70	0.79	0.47	0.60	0.67
		0.8x	6.48 (+0.08)	8.55 (+0.11)	0.81	0.71	0.79	0.47	0.60	0.68 (+0.01)
		0.7x	6.66 (+0.26)	8.82 (+0.38)	0.80	0.71	0.78	0.46	0.59	0.67 (-0.00)
		0.6x	7.01 (+0.61)	9.38 (+0.94)	0.79	0.70	0.77	0.44	0.57	0.66 (-0.01)
Llama-2	7b	Baseline	5.47	6.97	0.78	0.69	0.76	0.43	0.57	0.65
		0.8x	5.96 (+0.49)	7.82 (+0.85)	0.76	0.66	0.72	0.40	0.54	0.61 (-0.04)
		0.7x	6.28 (+0.81)	8.55 (+1.58)	0.75	0.63	0.70	0.38	0.52	0.60 (-0.05)
		0.6x	7.48 (+2.01)	10.07 (+3.10)	0.74	0.63	0.65	0.33	0.48	0.56 (-0.09)
	13b	Baseline	4.88	6.47	0.79	0.72	0.79	0.48	0.60	0.68
		0.8x	5.06 (+0.18)	6.77 (+0.30)	0.78	0.72	0.78	0.45	0.59	0.66 (-0.02)
		0.7x	5.32 (+0.44)	7.19 (+0.72)	0.77	0.70	0.77	0.45	0.57	0.65 (-0.03)
		0.6x	6.10 (+1.22)	9.34 (+2.87)	0.73	0.65	0.70	0.36	0.48	0.58 (-0.10)
	70b	Baseline	3.32	5.52	0.82	0.78	0.83	0.54	0.65	0.72
		0.8x	3.44 (+0.12)	5.74 (+0.22)	0.81	0.76	0.81	0.51	0.63	0.71 (-0.01)
		0.7x	3.60 (+0.28)	5.95 (+0.43)	0.81	0.76	0.81	0.51	0.61	0.70 (-0.02)
		0.6x	3.78 (+0.46)	6.23 (+0.71)	0.80	0.75	0.81	0.50	0.59	0.69 (-0.03)
Llama-3	8b	Baseline	6.14	8.88	0.80	0.73	0.80	0.51	0.60	0.69
		0.8x	6.72 (+0.58)	9.93 (+1.05)	0.78	0.73	0.78	0.48	0.57	0.67 (-0.02)
		0.7x	7.22 (+1.08)	10.91 (+2.03)	0.78	0.71	0.77	0.45	0.55	0.65 (-0.04)
		0.6x	8.51 (+2.37)	13.20 (+4.32)	0.76	0.67	0.74	0.42	0.50	0.62 (-0.07)
	70b	Baseline	2.85	6.73	0.82	0.81	0.87	0.60	0.66	0.75
		0.8x	3.22 (+0.37)	7.12 (+0.39)	0.82	0.80	0.85	0.59	0.66	0.74 (-0.01)
		0.7x	3.59 (+0.74)	7.48 (+0.75)	0.82	0.79	0.84	0.57	0.65	0.73 (-0.02)
		0.6x	5.54 (+2.69)	10.39 (+3.66)	0.78	0.73	0.77	0.45	0.54	0.65 (-0.10)

reaches a specified layerwise error budget. This condenses the layer-wise rank search to two hyper-parameters (i.e., error budget e_b, and step size ϵ_s), which is kept the same for all LLM layers. Error budget e_b can be increased to increase KV cache compression (Table 9). For all our experiments, we keep $\epsilon_s = 0.02$.

5 Experiments

5.1 Setup

We evaluate Eigen Attention across three model families: OPT (Zhang et al., 2022), MPT (MosaicML-MPT), and Llama (Touvron et al., 2023; Llama-3), each with distinct position encoding schemes. OPT employs learnable position embeddings, MPT utilizes AliBi (Press et al., 2021), and the Llama model family employs RoPE (Su et al., 2024). We conduct evaluations on both language generation and zero-shot tasks. The language generation tasks include perplexity evaluation on Wikitext-2 (Merity et al., 2016) and C4 (Dodge et al., 2021) datasets. The zero-shot tasks are obtained from lm-eval-harness framework (Gao et al., 2023): PiQA (Bisk et al., 2020), Winogrande (WinoG) (Sakaguchi et al., 2021), Arc-easy/challenge (Clark et al., 2018), and HellaSwag (HellaS) (Zellers et al., 2019).

To emphasize that our approach is orthogonal to existing compression techniques, we implement it alongside Grouped Query Attention (Ainslie et al., 2023) (present in Llama-2 70b and Llama-3) and Quantization (Zirui Liu et al., 2024).

5.2 Results

We demonstrate the compression benefits achieved by Eigen Attention through our results in Table 2 on various families and sizes of models for a number of language tasks. In particular, we report perplexity (PPL) on Wikitext and C4 datasets and accuracy (Acc) on various zero-shot benchmarks at three KV cache compression ratios: 0.8x, 0.7x, and 0.6x. As expected, increasing the degree of KV cache compression increases the perplexity while

Table 3: Perplexity and Accuracy results after fine-tuning. The baseline represents standard attention with an uncompressed KV cache. Degradation from baseline is shown in brackets.

Model	KV Cache	PPL ↓		Acc ↑					
		Wikitext	C4	PiQA	WinoG	Arc-e	Arc-c	HellaS	Avg-Acc
MPT-7b	Baseline	7.68	9.6	0.79	0.69	0.75	0.41	0.57	0.64
	0.6x	9.21 (+1.53)	11.44 (+2.14)	0.79	0.69	0.72	0.39	0.55	0.63 (-0.01)
Llama-2-7b	Baseline	5.47	6.97	0.78	0.69	0.76	0.43	0.57	0.65
	0.6x	6.55 (+1.07)	8.55 (+1.58)	0.78	0.67	0.74	0.41	0.54	0.63 (-0.02)
Llama-2-13b	Baseline	4.88	6.47	0.79	0.72	0.79	0.48	0.60	0.68
	0.6x	5.64 (+0.76)	7.77 (+1.30)	0.78	0.68	0.75	0.44	0.58	0.65 (-0.03)
Llama-3-8b	Baseline	6.14	8.88	0.80	0.73	0.80	0.51	0.60	0.69
	0.6x	7.60 (+1.47)	11.44 (+2.56)	0.80	0.70	0.79	0.48	0.58	0.67 (-0.02)

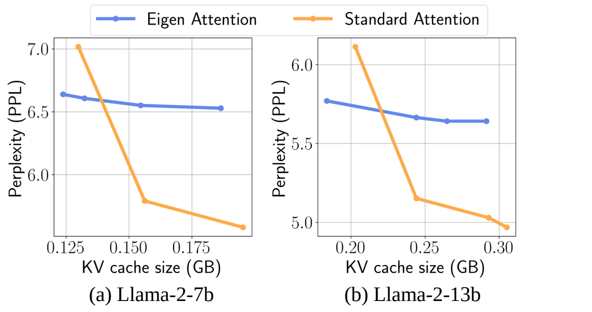


Figure 3: PPL on Wikitext with different KV cache sizes in GB (n = 2048) obtained via different quantization precision and group size. For Eigen Attention, we compress the KV cache to 0.6x and then apply quantization.

Table 4: Comparison of Eigen Attention with H_2O. Combination of Eigen Attention and H_2O leads to the best tradeoff between accuracy and KV cache size.

Model	Method	KV Cache	Avg. 0-shot Acc ↑
Llama-2-7b	H_2O	0.2x	0.54
	H_2O	0.4x	0.64
	Eigen Attn	0.6x	0.63
	Eigen Attn + H_2O	0.24x	0.62
Llama-3-8b	H_2O	0.2x	0.56
	H_2O	0.4x	0.67
	Eigen Attn	0.6x	0.67
	Eigen Attn + H_2O	0.24x	0.65

Table 5: Average latency per attention layer during token generation phase.

Model	Baseline Attn	Eigen Attn 0.6x
OPT-66b	1.57 ms	0.62 ms (-60%)
Llama-2-70b	0.79 ms	0.77 ms (-3%)
Llama-3-70b	0.74 ms	0.80 ms (+8%)

reducing accuracy on benchmark tasks. On average, perplexity increases by 0.32, 0.69, and 1.79 while accuracy drops by 1%, 2%, and 3% at 0.8x, 0.7x, and 0.6x KV cache compression, respectively. Within a model family, we find larger models to be more resilient to KV cache compression. Notably, the OPT-66b model incurs only 1% degradation in zero-shot accuracy with a 40% compression for the KV cache. Within the Llama-2 family, the average PPL gap to baseline reduces from 2.56 in the 7b model to 0.59 in the 70b parameter model. Across these three distinct model families, we find the KV cache of MPT models to be the most compressible while the Llama-3 models to be the least. For an iso-parameter size of 70b, the Llama-2 model is more resilient to KV cache compression than the Llama-3 model, with both employing grouped query attention.

Eigen Attention followed by Fine-tuning: We attempt to mitigate the increase in perplexity and the degradation in accuracy observed with the smaller LLMs by fine-tuning. We use LoRA (Hu et al., 2022) to fine-tune these models on the Alpaca dataset (Taori et al., 2023) for 2000 steps. Specifically, we fine-tune the query, key, value, and output projection matrices in the attention layer for MPT, Llama-2, and Llama-3 models and report the results in Table 3. Fine-tuning helps improve the performance of Eigen Attention models, making them perform closer to the baseline. The perplexity gap to baseline reduces from 2.56 (before fine-tuning) to 1.55 after fine-tuning. Similarly, the average zero-shot accuracy gap is reduced from 7% (before fine-tuning) to within 2% of baseline. We find the most improvements in Llama-3-8b, while the least improvements with MPT-7b LLM.

Eigen Attention with Quantization: We compare performance after quantizing key and value in standard and Eigen Attention. Note that the KV cache in Eigen Attention is first compressed to 0.6x before applying quantization. Figure 3 shows perplexity for the Llama-2-7b and Llama-2-13b models on the Wikitext (Merity et al., 2016) dataset across various KV cache sizes obtained by performing different levels of quantization. Similar to KIVI (Zirui Liu et al., 2024) and KV-Quant (Hooper et al., 2024), we perform per channel quantization for key and per token quantization for value. We implement integer quantization at different pre-

Table 6: Total inference latency (in seconds) for Llama-3-8b model at different prompt and generation length.

Batch Size	Prompt tokens	Generated tokens	Baseline Attn	Eigen Attn (0.6x)
1	8192	2048	76	82 (+8%)
1	8192	16384	785	880 (+12%)
1	8192	32768	2048	2350 (+13%)
32	2048	512	180	156 (-13%)
32	2048	2048	826	859 (+4%)
32	2048	4096	**OOM**	2366

Table 7: Perplexity and zero-shot accuracy when using different calibration datasets for MPT-7b model.

Calib. dataset	KV Cache	Wikitext (PPL)	C4 (PPL)	Zero-shot Acc. (%)
C4	0.8	8.03	9.96	0.63
	0.7	8.57	10.47	0.63
	0.6	9.63	11.44	0.62
Alpaca	0.8	8.12	10.07	0.64
	0.7	3.02	10.88	0.62
	0.6	10.86	12.42	0.61

cision and group sizes, leading to varied KV cache sizes (Table 8). For iso-KV cache size, the key and value are quantized to lower precision in quantized standard attention as compared to quantized Eigen Attention. We make two observations: (1) at large KV cache size, the quantized standard attention outperforms quantized Eigen Attention because less error is incurred due to quantization compared to the low-rank decomposition of attention matrices, (2) as quantization precision is reduced to lower the KV cache size, quantized Eigen Attention outperforms quantized standard attention. This is due to a much more severe quantization in standard attention, which induces higher approximation error than Eigen Attention.

Eigen Attention with token pruning: We compare Eigen Attention with a popular token pruning approach H_2O (Zhang et al., 2023), which only keeps K and V corresponding to a subset of tokens within the KV Cache. Table 4 shows the average zero shot accuracy comparison at different KV cache compression ratios. We observe that H_2O at 0.4x KV cache achieves similar accuracy to Eigen Attention at 0.6x KV cache. Superior performance of H_2O over Eigen Attention can be attributed to the dynamic nature of the algorithm. While Eigen Attention compresses KV cache based on statistics derived offline, H_2O is able to achieve better generalization by making decisions to eject tokens from KV cache at runtime. However, both Eigen Attention and H_2O compress KV cache along different dimensions and can be combined to achieve further compression as seen in Table 4. The combination of Eigen Attention and H_2O achieves the best performance highlighting the orthogonality of both approaches. By reducing the hidden dimension of K and V by 60% using Eigen Attention and using H_2O to keep only 40% of tokens in KV cache, we are able to compress KV cache to 0.24x while achieving accuracy comparable to Eigen Attention at 0.6x and H_2O at 0.4x KV cache.

Latency Comparisons: We compare the latency of Eigen Attention and standard attention baseline for different models. In Table 5, we show average latency per attention layer during the token generation phase for generating 128 tokens with a 2048-token synthetic prompt on 2 NVIDIA A100 GPUs. We observe an impressive 60% latency improvement for the OPT-66b model, which can be attributed to the FLOPs reduction from Eigen Attention and the reduced latency in fetching the compressed KV cache from memory. For models with RoPE embedding (i.e., Llama-2-70b and Llama-3), we perform an additional transformation before the attention dot product (Figure 6), which diminishes the latency gains achieved with Eigen Attention. For Llama-2-70b, we observe only a 3% latency improvement, while for Llama-3-70b, there is a latency penalty of 8%. We further analyze end to end inference latency at different context lengths and batch sizes in Table 6 for Llama-3-8b on a single NVIDIA A100 GPU. We see that increasing the context length increases the latency overhead with Eigen Attention due to the presence of additional transformation to manage RoPE embedding. For the maximum sequence length of 40k, there is a 13% latency overhead with Eigen Attention for Llama-3-8b. When batch size is increased, the KV cache size increases which increases the memory bounded nature of inference. In this scenario, additional transformation required by Eigen Attention leads to much lesser latency overhead and often faster inference than baseline attention. Most notably, at high batch size and context length, baseline attention leads to out of memory (OOM) error on GPU while Eigen Attention does not. This highlights the main contribution of our work: to enable long context inference by compressing KV cache.

5.3 Ablation Studies

We analyze our approach by varying the number of calibration samples used to compute the representation matrix for low-rank approximation and the number of steps used to fine-tune the models.

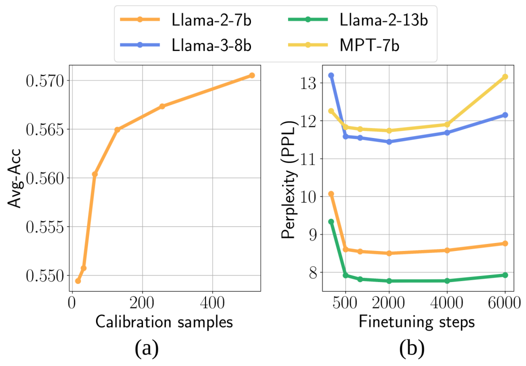


Figure 4: Ablation Study. (a) Average accuracy (Avg-Acc) on zero-shot tasks trained on Llama-2-7b with an increasing number of calibration samples. (b) Perplexity (PPL) vs fine-tuning steps on the C4 dataset for MPT and Llama family of models.

Impact of calibration dataset: Table 7 shows the performance of MPT-7b model when different calibration datasets are used for obtaining low rank space of K, V vectors. We select samples from C4 (Dodge et al., 2021) and Alpaca (Taori et al., 2023) datasets for obtaining the low rank basis. We see that compressing using C4 calibration dataset achieves better perplexity on the C4 dataset itself while achieving slightly lower perplexity on Wikitext dataset (results in Table 2). Performance on zero shot tasks remains similar regardless of the calibration dataset choice.

Number of Calibration Samples: Figure 4(a) shows average accuracy (Avg-Acc) on zero-shot benchmarks with 0.6x KV cache for different numbers of calibration samples used to generate the representation matrix for low-rank approximation. Increasing the number of calibration samples enhances the Avg-Acc, as more samples lead to a more generalized set of basis vectors. However, more samples also increase the size of representation matrices (Equation 4), requiring significantly higher GPU memory for performing SVD. Using more than 128 calibration samples leads to out-of-memory errors. We average representations from samples to handle this, thereby reducing representation matrix dimensions. For instance, representations from every 2 (4) samples are averaged for 256 (512) samples.

Fine-tuning Steps: In Figure 4(b), we evaluate perplexity (PPL) for the C4 dataset evaluated on the MPT and Llama models for a range of fine-tuning steps. With just 500 steps, the PPL improves by 1.2 on average across all the models. 2000 steps are ideal, beyond which we observe an average increase of 0.61 in the PPL. We posit that this is due to model overfitting on finetuning data.

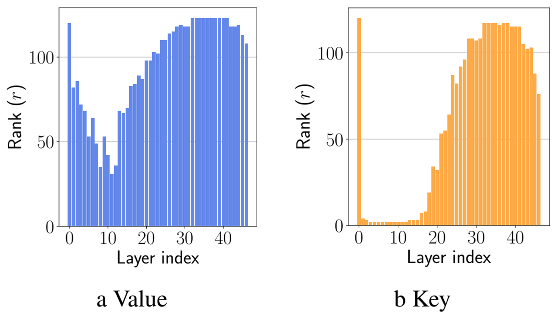


Figure 5: Layerwise rank assignment for key and value determined by Eigen Attention for OPT-30b with 40% compressed KV cache.

5.4 Layerwise Rank Visualization

Figure 5 shows the rank r assigned to key and query projection layers by Eigen Attention at 40% KV cache compression. We observe that the initial layers (with the exception of first layer) are compressed more than the later layers. Also, keys are assigned a lower rank and are compressed more than values, which concurs with our eigenvalue spectrum analysis in Figure 2. Specifically, for 40% KV cache compression, keys are compressed by 54% while values are compressed by only 26%.

6 Conclusion

In this work, we propose Eigen Attention, a novel technique that reduces the memory overhead associated with KV cache in LLMs. Eigen Attention is inspired by the observation that keys, queries, and values can be approximated using a few basis vectors, enabling the possibility of performing the attention operation in a low-rank space with minimal performance loss. To achieve this, we project the attention inputs into low-rank subspaces defined by a set of principal basis vectors computed offline using a calibration dataset in a one-shot manner. These projections are integrated into the weight matrices, which are utilized during inference to generate lower dimensional key and value matrices, thereby reducing the KV cache memory footprint. Our approach is orthogonal to existing KV cache compression strategies and can be used in synergy. Extensive experiments across a range of LLMs and language tasks demonstrate that Eigen Attention reduces the KV cache memory footprint by 40% with minimal performance loss and achieves up to a 60% improvement in attention operation latency compared to the standard attention baseline.

Limitations

Eigen Attention takes a step towards efficiently enabling longer context lengths, thereby opening avenues for enhancing the capabilities of the current state-of-the-art LLMs. While we demonstrate low-rank basis generation using the Wikitext dataset (Merity et al., 2016), we do not extensively study the best dataset for basis generation. Additionally, although our proposed approach has the potential to make LLMs ubiquitous, it does not mitigate the risks of misuse of these models for malicious activities. A strong commitment to user data protection, robust ethical guidelines, and transparency mechanisms is essential to address this issue effectively.

Acknowledgements

This work was supported by the Center for the Co-Design of Cognitive Systems (COCOSYS), a DARPA sponsored JUMP center of Semiconductor Research Corporation (SRC), National Science Foundation and United States Department of Energy.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. 2024. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Preprint*, arXiv:2403.09054.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. 2024. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. *Preprint*, arXiv:2104.08758.

Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael Jordan, and Zheng-Jun Zha. 2022. Rank diminishing in deep neural networks. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

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

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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

Ayush Kaushal, Tejas Vaidhya, and Irina Rish. 2023. Lord: Low rank decomposition of monolingual code llms for one-shot compression. *Preprint*, arXiv:2309.14021.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023a. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023b. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 20336–20350. PMLR.

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

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for llm compression and acceleration. *Preprint*, arXiv:2306.00978.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2024. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36.

Llama-3. Introducing meta llama 3: The most capable openly available llm to date. Accessed: 2010-06-07.

lmsys.org. 2024. Lmsys chatbot arena leaderboard.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

MosaicML-MPT. Introducing mpt-7b: A new standard for open-source, commercially usable llms. Accessed: 2010-06-07.

Yue Niu, Saurav Prakash, and Salman Avestimehr. 2024. Atp: Enabling fast llm serving via attention on top principal keys. *Preprint*, arXiv:2403.02352.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference. *Preprint*, arXiv:2211.05102.

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

Gobinda Saha, Isha Garg, and Kaushik Roy. 2021. Gradient projection memory for continual learning. In *International Conference on Learning Representations*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

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

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Haoyi Wu and Kewei Tu. 2024. Layer-condensed kv cache for efficient inference of large language models. *Preprint*, arXiv:2405.10637.

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

Hao Yu and Jianxin Wu. 2023. Compressing transformers: Features are low-rank, but weights are not! *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11007–11015.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024a. Soaring from 4k to 400k: Extending llm's context with activation beacon. *arXiv preprint arXiv:2401.03462*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yichi Zhang, Bofei Gao, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, Wen Xiao, et al. 2024b. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. *Preprint*, arXiv:2306.14048.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024. Kivi : Plug-and-play 2bit kv cache quantization with streaming asymmetric quantization.

A Appendix

A.1 SVD for Matrix Approximation

Singular Value Decomposition (SVD) can be used to obtain a low-rank approximation for any matrix $\mathbf{Z} \in \mathbb{R}^{m \times n}$ by factorizing it into three matrices as $\mathbf{Z} = \mathbf{U\Sigma V}$. Here, $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\mathbf{\Sigma}$ is a diagonal matrix which contains the sorted singular values. For $\mathbf{Z}$ with a rank $r \leq \min(m, n)$, it can be expressed as $\mathbf{Z} = \sum_{i=1}^{r} \sigma_i u_i v_i^T$, where $u_i \in \mathbf{U}$, $v_i \in \mathbf{V}$ and $\sigma_i \in \mathbf{\Sigma}$. For a k-rank approximation with $k < r$, we have $\mathbf{Z}_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ such that the top k values from $\mathbf{\Sigma}$ are chosen to represent $\mathbf{Z}_k$, a low-rank approximation of $\mathbf{Z}$.

A.2 Eigen Attention with RoPE

We introduce some modifications to Eigen Attention algorithm to handle compatibility with LLMs employing RoPE empedding (Section 4.3). The comparison of Eigen Attention with standard attention in the presence of RoPE is shown in Figure 6.

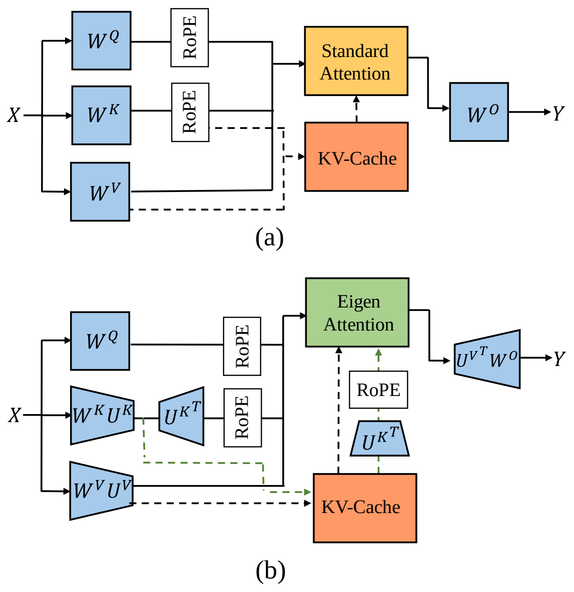


Figure 6: Comparison between (a) Standard Attention and (b) Eigen Attention for LLMs with RoPE (Touvron et al., 2023). Eigen Attention stores low dimensional representation of key and value in KV cache and applies an additional transformation before applying RoPE.

A.3 Quantization Results

Table 8: Perplexity (PPL) on Wikitext after applying quantization to eigen attention and standard attention.

Model	Method	Precision	Group Size	KV Cache size (GB)	PPL (Wikitext)
Llama-2-13b	Eigen Attention	3	128	0.18	5.77
		4	128	0.24	5.66
		4	32	0.26	5.64
		4	16	0.29	5.64
	Standard Attention	2	512	0.20	6.11
		2	32	0.24	5.15
		2	16	0.29	5.03
		3	128	0.31	4.97
Llama-2-7b	Eigen Attention	4	16	0.19	6.53
		4	128	0.15	6.55
		3	32	0.13	6.61
		3	64	0.12	6.64
	Standard Attention	2	512	0.13	7.02
		2	32	0.16	5.79
		3	128	0.20	5.58

Section 5.2 shows results for performing quanti-

zation with Eigen Attention and standard attention. The data corresponding to Figure 3 is provided in Table 8.

A.4 Hyperparameters

Table 9: Error budget e_b for Eigen Attention. KV cache size is computed for a sequence length of 2048.

Model	Parameters	KV cache compression	KV cache size (GB)	e_b
OPT	30b	0.8x	2.10	0.008
		0.7x	1.87	0.015
		0.6x	1.60	0.024
	66b	0.8x	3.69	0.005
		0.7x	3.22	0.012
		0.6x	2.80	0.015
MPT	7b	0.8x	0.80	0.011
		0.7x	0.70	0.015
		0.6x	0.60	0.019
	30b	0.8x	2.12	0.002
		0.7x	1.83	0.003
		0.6x	1.59	0.004
Llama-2	7b	0.8x	0.83	0.025
		0.7x	0.71	0.070
		0.6x	0.59	0.090
	13b	0.8x	1.29	0.035
		0.7x	1.12	0.050
		0.6x	0.93	0.070
	70b	0.8x	3.44	0.005
		0.7x	3.60	0.010
		0.6x	3.78	0.017
Llama-3	8b	0.8x	6.72	0.045
		0.7x	7.22	0.060
		0.6x	8.51	0.090
	70b	0.8x	3.22	0.025
		0.7x	3.59	0.035
		0.6x	5.54	0.070

We use the Hugging Face Transformers library (Wolf et al., 2020) to implement Eigen Attention in PyTorch (Paszke et al., 2019). All the experiments were performed on NVIDIA A100 (80GB) GPUs. All models are downloaded from Hugging Face Hub. We show results on: OPT-30b, OPT-66b, MPT-7b, MPT-30b, Llama-2-7b, Llama-2-13b, Llama-2-70b, Llama-3-8b and, Llama-3-70b.

All the models are compressed using 512 (n_s) samples of sequence length (n) 2048 from Wikitext dataset (Merity et al., 2016). Eigen Attention introduces a hyperparameter for error budget (e_b), which is tuned to achieve the required KV cache compression, as listed in Table 9. We keep ϵ_s = 0.02 for all our runs.

For fine-tuning, we use $lora\_r = 64$, $lora\_alpha = 64$, $sequence\_length = 2048$, $lora\_dropout = 0.05$ and use the default values from the Hugging Face PEFT library (Mangrulkar et al., 2022) for all the other hyperparameters. We observe that fine-tuning on the Alpaca dataset (Taori et al., 2023) performs better than C4 (Dodge et al., 2021).

A.5 Artifact Licenses

According to their license, all the LLMs used in this paper fall under acceptable use cases. The licenses for the models are linked for perusal: OPT-30b, OPT-66b, MPT-7b, MPT-30b, Llama-2-7b, Llama-2-13b, Llama-2-70b, Llama-3-8b and, Llama-3-70b.

A.6 Future Work

We describe two key future directions for Eigen Attention: (1) integrating Eigen Attention with efficient LLM serving frameworks like vLLM (Kwon et al., 2023b), which employ additional approximation techniques (e.g., weight quantization (Lin et al., 2024)) to achieve high throughput inference, and (2) finding the best combination of various compression techniques described in Section 3 to achieve extreme KV cache compression.