Some theoretical results on SVD methods for KV cache compression

Damien Lesens,
with Beheshteh T. Rakhshan and Guillaume Rabusseau

8 July 2025

Plan

Transformer architecture

- Large language models (LLMs) are a groundbreaking advancement in natural language processing
- They enable a wide range of applications and have become a central focus of AI research
- Efficient memory usage and high throughput are critical to scaling and deploying these models effectively

Transformer architecture

Words are embedded into high dimensional vectors and are fed to the model in two phases:

- Pre-filling: the whole prompt is passed to the model, which generates a first token
- Auto-regressive generation: the prompt + the first generated token are passed to the model, generating a new token, and the process repeats

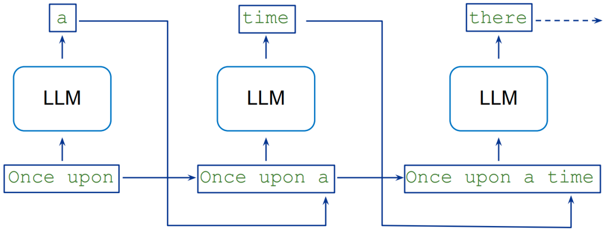


Image credits: "Role-Play with Large Language Models", Shanahan et al.

Transformer architecture

- Many different architectures, but usually composed of a succession of layers which look like this
- Their capabilities are mainly due to the attention computation introduced by the seminal paper "Attention is all you need", Vaswani et al, 2017
- We are interested in optimizing the Multi-Head Attention computation

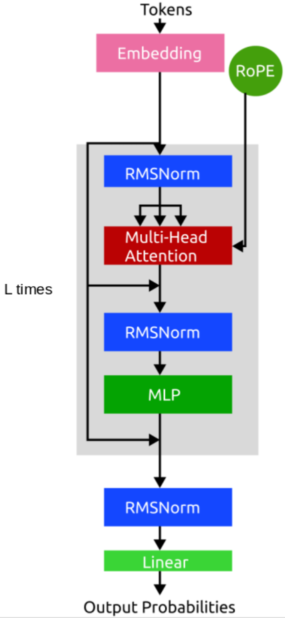


Image source: Medium "LLMs Explained: LLaMA and Its Architecture (Part 1)"

Multi-head attention

We are interested in the multi-head attention computation

- n: sequence length
- $\mathbf{X} \in \mathbb{R}^{n \times D}$: input hidden states
- h: number of attention heads
- $d = D/h$: head dimension

Computes for each head queries, keys and values

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q, \mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K, \mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V \in \mathbb{R}^{n \times d}, i \in [\![1, h]\!]$$

using learnable parameters $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{D \times d}$

$$\mathbf{H}_i = \text{Softmax}(\mathbf{Q}_i\mathbf{K}_i^T/\sqrt{d})\mathbf{V}_i$$

$$\text{MHA}(\mathbf{X}) = [\mathbf{H}_1, \ldots, \mathbf{H}_h]\mathbf{W}_O$$

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Multi-head attention

$$\mathbf{H}_i = \text{Softmax}(\mathbf{Q}_i\mathbf{K}_i^T/\sqrt{d})\mathbf{V}_i$$

$$\text{MHA}(\mathbf{X}) = [\mathbf{H}_1, \ldots, \mathbf{H}_h]\mathbf{W}_O$$

Issue:

- The cost of this computation scales quadratically with the sequence size n: prohibits long text generation

However

- The upper half of $\mathbf{Q}_i\mathbf{K}_i^T$ is set to $-\infty$ so that the attention of a token is not influenced by future tokens.

$$\text{Softmax}(\mathbf{Q}\mathbf{K}^T) = \begin{pmatrix} \alpha_{1,1} & 0 & 0 & \ldots & 0 \\ \alpha_{2,1} & \alpha_{2,1} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \alpha_{n,1} & \alpha_{n,2} & \ldots & \ldots & \alpha_{n,n} \end{pmatrix} \qquad h_j = \sum_{k=1}^{j-1} \alpha_{j,k} v_k$$

KV caching

- Computations for generating a token can be reused to generate the next token
- More precisely keys and values can be cached $\rightarrow$ **KV caching**

For auto-regressive generation,

$$\left.\begin{aligned} \mathbf{K}_i^{(n)} &\leftarrow \text{Concat}(\mathbf{K}_i^{(n-1)}, k_i^{(n)}) \\ \mathbf{V}_i^{(n)} &\leftarrow \text{Concat}(\mathbf{V}_i^{(n-1)}, v_i^{(n)}) \end{aligned}\right\} \text{stored in memory}$$

$$h_i^{(n)} = \text{Softmax}(q_i^{(n)} \mathbf{K}_i^{(n)T} / \sqrt{d}) \mathbf{V}_i^{(n)}$$

The cost of generating the nth token is now $O(n)$,
but the memory size of the cached keys and values scales as $O(n)$

KV caching

$$\left.\begin{aligned} \mathbf{K}_i^{(n)} &\leftarrow \text{Concat}(\mathbf{K}_i^{(n-1)}, k_i^{(n)}) \\ \mathbf{V}_i^{(n)} &\leftarrow \text{Concat}(\mathbf{V}_i^{(n-1)}, v_i^{(n)}) \end{aligned}\right\} \text{ stored in memory}$$

$$h_i^{(n)} = \text{Softmax}(q_i^{(n)} \mathbf{K}_i^{(n)T} / \sqrt{d}) \mathbf{V}_i^{(n)}$$

$\rightarrow$ **long sequence generation is a memory bound problem because of the KV cache**

Example: Llama2-7b (model size 14GB) for a sequence of length 32k the KV cache size is 16GB

How can we compress the KV cache to reduce its memory footprint ?

Dimension reduction

A lot of research in KV cache compression techniques

KV cache tensors (shape (n, L, h, d)) can be compressed along different dimensions[1]:

- n: token eviction
- L number of layers: modify architecture such that the cache is shared across layers
- h: Multi Query Attention (MQA), Grouped Query Attention (GQA), details on these later
- bit precision: Quantization
- d: hidden dimension

[1] Survey: "A survey on large language model acceleration based on kv cache management", Haoyang et al., 2024

Dimension reduction

A lot of research in KV cache compression techniques

KV cache tensors (shape (n, L, h, d)) can be compressed along different dimensions[1]:

- n: token eviction
- L number of layers: modify architecture such that the cache is shared across layers
- h: Multi Query Attention (MQA), Grouped Query Attention (GQA), details on these later
- bit precision: Quantization
- d: hidden dimension

[1] Survey: "A survey on large language model acceleration based on kv cache management", Haoyang et al., 2024

SVD methods

- For a given layer and a given head, use a low-rank approximation of the key and value cache

$$\mathbf{K}_i \approx \mathbf{E}_i \mathbf{F}_i$$

  with $\mathbf{E}_i \in \mathbb{R}^{n \times R}$ and $\mathbf{F}_i \in \mathbb{R}^{R \times d}$
- Memory is $O(nR + Rd)$ instead of $O(nd)$
- Post-training, only need a pass of the model on a small calibration set of tokens
  $\rightarrow$ little time and ressources spent
- Introduced by Palu[1], LORC[2], MatryoshkaKV[3]

[1] "Palu: KV-Cache Compression with Low-Rank Projection", Chang et al., 2025

[2] "Lorc: Low-rank compression for llms kv cache with a progressive compression strategy", Zhang et al., 2024

[3] "MatryoshkaKV: Adaptive KV Compression via Trainable Orthogonal Projection", Lin et al., 2024

SVD methods

How to compute this low-rank approximation ?

- Take a collection of sequences from a text dataset (very small compared to the train dataset).
- Pass each sequence into the model and collect the KV caches
- Perform a Singular Value Decomposition of $\mathbf{K} = \mathbf{U}_K \mathbf{\Sigma}_K \mathbf{V}_K^T$ and keep only the R first right singular vectors

SVD methods

How to compute this low-rank approximation ?

- Take a collection of sequences from a text dataset (very small compared to the train dataset).
- Pass each sequence into the model and collect the KV caches
- Perform a Singular Value Decomposition of $\mathbf{K} = \mathbf{U}_K \boldsymbol{\Sigma}_K \mathbf{V}_K^T$ and keep only the R first right singular vectors

Singular Value Decomposition (SVD), writes $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ $(n_1 \geq n_2)$ as

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_{i=1}^{n_2} \sigma_i u_i v_i^T$$

with $\mathbf{U} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times n_2}$ orthogonal, $\boldsymbol{\Sigma}$ diagonal with positive decreasing entries $(\sigma_i)_i$

SVD methods

- Approximate $\mathbf{K}$ as $\mathbf{K} \approx \mathbf{K}\mathbf{V}_{K,:R}\mathbf{V}_{K,:R}^T = \mathbf{U}_{K,:R}\mathbf{\Sigma}_{K,:R}\mathbf{V}_{K,:R}^T$
- The basis $\mathbf{V}_{K,:R}$ does not depend on the sequence length $\rightarrow$ **it generalizes well to another key cache**

$$\mathbf{K}' \approx \mathbf{K}'\mathbf{V}_{K,:R}\mathbf{V}_{K,:R}^T \text{ with } \mathbf{K}' \neq \mathbf{K}$$

- Do the same thing for $\mathbf{V}$:
$\mathbf{V} = \mathbf{U}_V\mathbf{\Sigma}_V\mathbf{V}_V^T$ and $\mathbf{V} \approx \mathbf{V}\mathbf{V}_{V,:R}\mathbf{V}_{V,:R}^T$

SVD methods

- The attention computation is now

$$\mathbf{H}_i = \text{Softmax}(\mathbf{Q}_i \mathbf{V}_{K,:R} \mathbf{V}_{K,:R}^T \mathbf{K}_i^T / \sqrt{d}) \mathbf{V}_i \mathbf{V}_{V,:R} \mathbf{V}_{V,:R}^T$$

$$= \text{Softmax}(\mathbf{Q}_i \mathbf{V}_{K,:R} (\mathbf{K}_i \mathbf{V}_{K,:R})^T / \sqrt{d}) (\mathbf{V}_i \mathbf{V}_{V,:R}) \mathbf{V}_{V,:R}^T$$

- Store only $\mathbf{K}_i \mathbf{V}_{K,:R}, \mathbf{V}_i \mathbf{V}_{V,:R} \in \mathbb{R}^{n \times R}$
- For auto-regressive generation

$$\mathbf{K}_i^{(n)} \leftarrow \text{Concat}(\mathbf{K}_i^{(n-1)}, k_i^{(n)} \mathbf{V}_{K,:R})$$

$$\mathbf{V}_i^{(n)} \leftarrow \text{Concat}(\mathbf{V}_i^{(n-1)}, v_i^{(n)} \mathbf{V}_{V,:R})$$

and use $\tilde{q}_i^{(n)} = q_i^{(n)} \mathbf{V}_{K,:R}$

SVD methods

- This method works well because singular values of $\mathbf{K}$ and $\mathbf{V}$ decay fast, so they can be well approximated with $R < d$.
- The SVD provides the optimal low rank approximation according to the Frobenius norm (Eckart–Young–Mirsky theorem).

the solution of

$$\min_{\mathbf{P}} \|\mathbf{K} - \mathbf{KP}\|_F^2 \text{ subject to } \mathrm{rank}(\mathbf{P}) \leq R$$

is $\mathbf{P}_{K,R} = \mathbf{V}_{K,:R}\mathbf{V}_{K,:R}^T$ with $\mathbf{K} = \mathbf{U}_K\mathbf{\Sigma}_K\mathbf{V}_K^T$

SVD methods

- However, we are not interested in approximating $\mathbf{K}$ and $\mathbf{V}$ but the output of the attention

$$\widetilde{\mathsf{MHA}}(\mathbf{X}) = [\mathsf{Softmax}(\tilde{\mathbf{Q}}_i \tilde{\mathbf{K}}_i^T / \sqrt{d}) \tilde{\mathbf{V}}_i]_i \tilde{\mathbf{W}}_O$$
$$\approx [\mathsf{Softmax}(\mathbf{Q}_i \mathbf{K}_i^T / \sqrt{d}) \mathbf{V}_i]_i \mathbf{W}_O$$

$$\text{with } \tilde{\mathbf{Q}}_i \approx \mathbf{Q}_i,\ \tilde{\mathbf{K}}_i \approx \mathbf{K}_i,\ \tilde{\mathbf{V}}_i \approx \mathbf{V}_i$$

- $\mathbf{Q}_i$ and $\mathbf{W}_O$ also impact the result

Taking queries into account

- Works in the litterature like Eigen[4] and Zack[5] observe that queries should be taken into account.

- Indeed, $\mathbf{P}_{K,R} = \mathbf{V}_{K,:R}\mathbf{V}_{K,:R}^T$ is an orthogonal projection, i.e. $\mathbf{P}^2 = \mathbf{P}$ and $\mathbf{P}^T = \mathbf{P}$ so

$$\mathbf{Q}_i\mathbf{V}_{K,:R}\mathbf{V}_{K,:R}^T\mathbf{K}_i^T = \mathbf{Q}_i\mathbf{P}_{K,R}\mathbf{K}_i^T$$

$$= (\mathbf{Q}_i\mathbf{P}_{K,R})(\mathbf{K}_i\mathbf{P}_{K,R})^T$$

  We are projecting the row space of $\mathbf{K}$ AND $\mathbf{Q}$

- Same thing for $\mathbf{V}$: take $\mathbf{W}_O$ into account

[4] "Eigen attention: Attention in low-rank space for kv cache compression", Saxena et al., 2024

[5] "ZACK: Zero-Overhead LLM Inference Acceleration via Dimensionality Compression of the Key-Value Cache", Zhang and Shen, 2025

Taking queries into account

Zack and Eigen find a projection by doing an SVD on $\begin{pmatrix} \mathbf{K} \\ \mathbf{Q} \end{pmatrix}$

Instead of solving

$$\min_{\mathbf{P}} \|\mathbf{K} - \mathbf{KP}\|_F^2 \text{ subject to rank}(\mathbf{P}) \leq R$$

they solve

$$\min_{\mathbf{P},\ \mathsf{rk}(\mathbf{P}) \leq R} \| \begin{pmatrix} \mathbf{K} \\ \mathbf{Q} \end{pmatrix} - \begin{pmatrix} \mathbf{K} \\ \mathbf{Q} \end{pmatrix} \mathbf{P} \|_F^2 = \min_{\mathbf{P},\ \mathsf{rk}(\mathbf{P}) \leq R} \|\mathbf{K} - \mathbf{KP}\|_F^2 + \|\mathbf{Q} - \mathbf{QP}\|_F^2$$

to find a projection matrix that will approximate $\mathbf{K}$ and $\mathbf{Q}$ at the same time.

$\rightarrow$ performs better that doing an SVD on $\mathbf{K}$ only

Approximating the attention matrix

- However we are not interested in approximating $\mathbf{K}$ and $\mathbf{Q}$ at the same time but the attention matrix $\mathbf{K}\mathbf{Q}^T$
- We would want to use $\mathbf{A}$ and $\mathbf{B}$ solution of

$$\min_{\mathbf{A},\mathbf{B}\in\mathbb{R}^{d\times R}} \|\mathbf{K}\mathbf{A}\mathbf{B}^T\mathbf{Q}^T - \mathbf{K}\mathbf{Q}^T\|_F^2$$

- This problem has a closed form solution, given by doing an SVD of $\mathbf{K}\mathbf{Q}^T$ (see e.g. DRONE[6])

[6] "DRONE: Data-aware Low-rank Compression for Large NLP Models", Patrick et al.

Approximating the attention matrix

$$\min_{\mathbf{A},\mathbf{B}\in\mathbb{R}^{d\times R}} \|\mathbf{KAB}^T\mathbf{Q}^T - \mathbf{KQ}^T\|_F^2$$

- choose $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{KAB}^T\mathbf{Q}^T$ is the best rank R approximation of $\mathbf{KQ}^T$
- $\mathbf{KQ}^T = \mathbf{U\Sigma V}^T \approx \mathbf{U}_{:R}\mathbf{\Sigma}_{:R}\mathbf{V}_{:R}^T$
- we want $\mathbf{KAB}^T\mathbf{Q}^T = \mathbf{U}_{:R}\mathbf{U}_{:R}^T\mathbf{KQ}^T = \mathbf{U}_{:R}\mathbf{\Sigma}_{:R}\mathbf{V}_{:R}^T$
- $\mathbf{P}_{KQ^T} = \mathbf{AB}^T = \mathbf{Q}^T\mathbf{V}_{:R}\mathbf{\Sigma}_{:R}^{-1}\mathbf{U}_{:R}^T\mathbf{K}$ works

Approximating the attention matrix

- Doing an SVD of $\mathbf{K}\mathbf{Q}^T \in \mathbb{R}^{n \times n}$ is costly compared to $\mathbf{K} \in \mathbb{R}^{n \times d}$ as $n \gg d$
- Solution:
  do an SVD of $\mathbf{K} = \mathbf{U}_K \boldsymbol{\Sigma}_K \mathbf{V}_K^T$ and $\mathbf{Q} = \mathbf{U}_Q \boldsymbol{\Sigma}_Q \mathbf{V}_Q^T$,
  and finally an SVD of $\boldsymbol{\Sigma}_K \mathbf{V}_K^T \mathbf{V}_Q \boldsymbol{\Sigma}_Q = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T \in \mathbb{R}^{d \times d}$
- Another formula is $\mathbf{P}_{KQ^T} = (\mathbf{V}_K \boldsymbol{\Sigma}_K^{-1} \mathbf{U})_{:R} (\mathbf{V}_K \boldsymbol{\Sigma}_K \mathbf{U})_{:R}^T$
- Same asymptotic cost as Eigen and SVD on $\mathbf{K}$

Approximating the attention matrix

Recap of the method we propose

- Pass the model on a calibration set of sequences
- Gather the query, key and value cache
- Solve for each head and each layer

$$\min_{\mathbf{A},\mathbf{B}\in\mathbb{R}^{d\times R}} \|\mathbf{K}_i\mathbf{A}\mathbf{B}^T\mathbf{Q}_i^T - \mathbf{K}_i\mathbf{Q}_i^T\|_F^2$$

$$\min_{\mathbf{C},\mathbf{D}\in\mathbb{R}^{d\times R}} \|\mathbf{V}_i\mathbf{C}\mathbf{D}^T\mathbf{W}_{O,i} - \mathbf{V}_i\mathbf{W}_{O,i}\|_F^2$$

- Store $\mathbf{K}_i\mathbf{A}$,$\mathbf{V}_i\mathbf{C}$ and incorporate $\mathbf{B}$,$\mathbf{D}$ into the attention computation to reduce the memory from $O(nd)$ to $O(nR)$

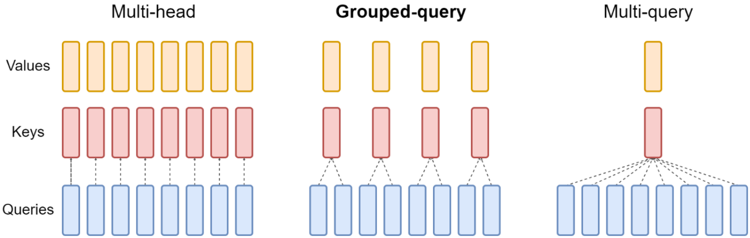
Grouped Query Attention

- This method needs to be compatible with Grouped Query Attention (GQA) introduced by Ainslie et al.[7]
- Dimension reduction technique that is present in most recent LLMs and that we cannot avoid (in all Llama models after Llama2-7B)
- Reduces the number of key and value heads from h to g (number of query heads is still h)

[7] "GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints", Ainslie et al.

Grouped Query Attention

- Trains a model with h heads, then groups heads and mean pools $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$ in each group
- The model then is trained a little to restore performance



$\rightarrow$ Number of query heads and key heads is different

Image credits: "GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints", Ainslie et al.

Compatibility with Grouped Query Attention

- Instead of solving

$$\min_{\mathbf{A},\mathbf{B}\in\mathbb{R}^{d\times R}} \|\mathbf{K}_i\mathbf{A}\mathbf{B}^T\mathbf{Q}_i^T - \mathbf{K}_i\mathbf{Q}_i^T\|_F^2$$

- We would want to solve the optimisation problem for each head group

$$\min_{\mathbf{A},\mathbf{B}_i} \| \sum_{i\in \text{ group}} \mathbf{K}\mathbf{A}\mathbf{B}_i^T\mathbf{Q}_i^T - \mathbf{K}\mathbf{Q}_i^T\|_F^2$$

- One basis for $\mathbf{K}$ but multiple each for each $\mathbf{Q}_i$ in the group

- Previous works do not explain how to handle GQA

Compatibility with Grouped Query Attention

We showed

Theorem

The optimisation problem

$$\min_{\mathbf{A},\mathbf{B}_i} \| \sum_{i \in \textit{ group}} \mathbf{K}\mathbf{A}\mathbf{B}_i^T \mathbf{Q}_i^T - \mathbf{K}\mathbf{Q}_i^T \|_F^2$$

has a closed form solution which can be computed efficiently by doing an SVD on

$$\mathbf{K}(\sum_{i \in \textit{ group}} \mathbf{Q}_i)^T$$

- Like $\mathbf{W}_{K,i}$ weights are averaged to get a single key cache per group, we need to average queries in each group.

Comparing SVD methods

- Approximating the attention matrix $\mathbf{KQ}^T$ makes sense
- Other works decide to approximate other objects
- In which situations are other heuristics in the literature bad at approximating the attention matrix, i.e. at solving

$$\min_{\mathbf{A},\mathbf{B}\in\mathbb{R}^{d\times R}} \|\mathbf{KAB}^T\mathbf{Q}^T - \mathbf{KQ}^T\|_F^2$$

- Do these situations happen with real caches on real models ?

Comparing SVD methods

We can quantify how bad doing an SVD on $\mathbf{K}$ is compared to an SVD on $\mathbf{KQ}^T$

Theorem

Let $\mathbf{K} = \mathbf{U}_K \mathbf{\Sigma}_K \mathbf{V}_K^T$, $\mathbf{Q} = \mathbf{U}_Q \mathbf{\Sigma}_Q \mathbf{V}_Q^T$ *and finally* $\mathbf{\Sigma}_K \mathbf{V}_K^T \mathbf{V}_Q \mathbf{\Sigma}_Q = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ *then*

$$\|\mathbf{K}\mathbf{V}_{K,:R}\mathbf{V}_{K,:R}^T\mathbf{Q}^T - \mathbf{K}\mathbf{Q}^T\|_F^2 = \|\mathbf{K}\mathbf{P}_{KQ^T}\mathbf{Q}^T - \mathbf{K}\mathbf{Q}^T\|_F^2 + \epsilon$$

with

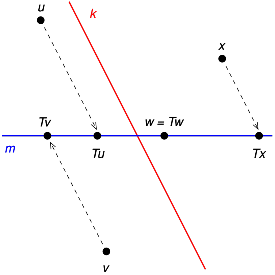
$$\epsilon = \|\mathbf{\Sigma}_{:R}\|_F^2 - \|\mathbf{\Sigma}_{K,:R}\mathbf{V}_{K,:R}^T\mathbf{V}_Q\mathbf{\Sigma}_Q\|_F^2 \geq 0$$

- the gap ϵ is easy to compute
- gives exactly how bad doing only an SVD on $\mathbf{K}$ will be at approximating the attention matrix

Comparing SVD methods Ungoing work

Comparing $\mathbf{P}_{KQ^T}$ and $\mathbf{P}_{Eigen}$, i.e SVD on $\mathbf{KQ}^T$ versus $\begin{pmatrix}\mathbf{K}\\ \mathbf{Q}\end{pmatrix}$

- In general $\mathbf{P}_{KQ^T}$ is an oblique projection
- $\mathbf{P}_{Eigen}$ is always orthogonal
- Even if $\mathbf{P}_{KQ^T}$ is orthogonal, we do not necessarily have Range$(\mathbf{P}_{KQ^T})$ = Range$(\mathbf{P}_{Eigen})$



Future works

- Get a lower bound result for $\mathbf{P}_{KQ^T}$ and $\mathbf{P}_{Eigen}$
- Test on a collection of LLMs whether the theoretical conditions we identify really happen
- Across layers
- Across different calibration set

$\rightarrow$ know for a given model and a given layer which method to use

Conclusion

- SVD methods reduce the dimension of the KV cache to allow long sequence generation
- We argue that it makes more sense to approximate the attention matrix
- Works in the literature do SVD on different objects
- We give conditions under which approximating the attention matrix is better (and we will verify experimentally whenever these conditions do happen)

Conclusion

- SVD methods reduce the dimension of the KV cache to allow long sequence generation
- We argue that it makes more sense to approximate the attention matrix
- Works in the literature do SVD on different objects
- We give conditions under which approximating the attention matrix is better (and we will verify experimentally whenever these conditions do happen)

Thank you for your attention

References I

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

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*, The Thirteenth International Conference on Learning Representations, 2025.

References II

Patrick Chen, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh, *Drone: Data-aware low-rank compression for large nlp models*, Advances in Neural Information Processing Systems (M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, eds.), vol. 34, Curran Associates, Inc., 2021, pp. 29321–29334.

Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen, *A survey on large language model acceleration based on kv cache management*, arXiv preprint arXiv:2412.19442 (2024).

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 preprint arXiv:2410.14731 (2024).

References III

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in Neural Information Processing Systems (I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

Zeyu Zhang and Haiying Shen, *Zack: Zero-overhead llm inference acceleration via dimensionality compression of the key-value cache*, 2025.

References IV

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

Handling positional encodings

- Modern LLMs use positional encodings so that the interaction between tokens depends on their relative position.
- Example: RoPE in Llama. Rows of $\mathbf{K}$ and $\mathbf{Q}$ are multipled by a rotation matrix whose angle is a function of the token index

$$\begin{aligned} q_m k_n^T &= (x_m \mathbf{W}_i^Q \mathbf{R}_{\theta,m}^d)(x_n \mathbf{W}_i^K \mathbf{R}_{\theta,n}^d)^T \\ &= (x_m \mathbf{W}_i^Q)\mathbf{R}_{\theta,m}^d {\mathbf{R}_{\theta,n}^d}^T (x_n \mathbf{W}_i^K) \\ &= (x_m \mathbf{W}_i^Q)\mathbf{R}_{\theta,m-n}^d (x_n \mathbf{W}_i^K) \end{aligned}$$

decays when $m - n$ is large

Handling positional encodings

The KV cache is less low-rank with RoPE applied. Multiple options to handle it:

- Compress before RoPE, but you have to decompress to apply RoPE
- Compress after RoPE, less low-rank but still possible. The calibration set needs to contain the whole range of positional embeddings.
- Some works have tried to remove RoPE from some heads[8]
- Without RoPE one can merge projections into weights and reduce the size of attention weights

[8]"EliteKV: Scalable KV Cache Compression via RoPE Frequency Selection and Joint Low-Rank Projection", Zhou et al.

Formula for GQA

Theorem

The solution of

$$\min_{\mathbf{A},\mathbf{B}_i} \| \sum_{i \in \text{ group}} \mathbf{K}\mathbf{A}\mathbf{B}_i^T\mathbf{Q}_i^T - \mathbf{K}\mathbf{Q}_i^T \|_F^2$$

is given by

$$\mathbf{A} = (\sum_{i \in \text{ group}} \mathbf{Q}_i^T)\mathbf{V}_{:R}\mathbf{\Sigma}_{:R}^{-1} \text{ and } \mathbf{B}_i = \mathbf{U}_{:R}^T\mathbf{K}$$

where $\mathbf{U}$,$\mathbf{\Sigma}$,$\mathbf{V}$ *are obtained from the SVD of*

$$\mathbf{K}(\sum_{i \in \text{ group}} \mathbf{Q}_i)^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

and $\mathbf{U}_{:R}$, $\mathbf{V}_{:R}$ *denote the first* R *columns of* $\mathbf{U}$ *and* $\mathbf{V}$ *respectively, and* $\mathbf{\Sigma}_{:R}$ *the first* R *rows and columns of* $\mathbf{\Sigma}$

Taking queries into account

Doing an SVD on $\begin{pmatrix} \mathbf{K} \\ \mathbf{Q} \end{pmatrix}$ performs better that doing an SVD on $\mathbf{K}$ only