LLM Notes

A Preprint

July 18, 2025

Contents

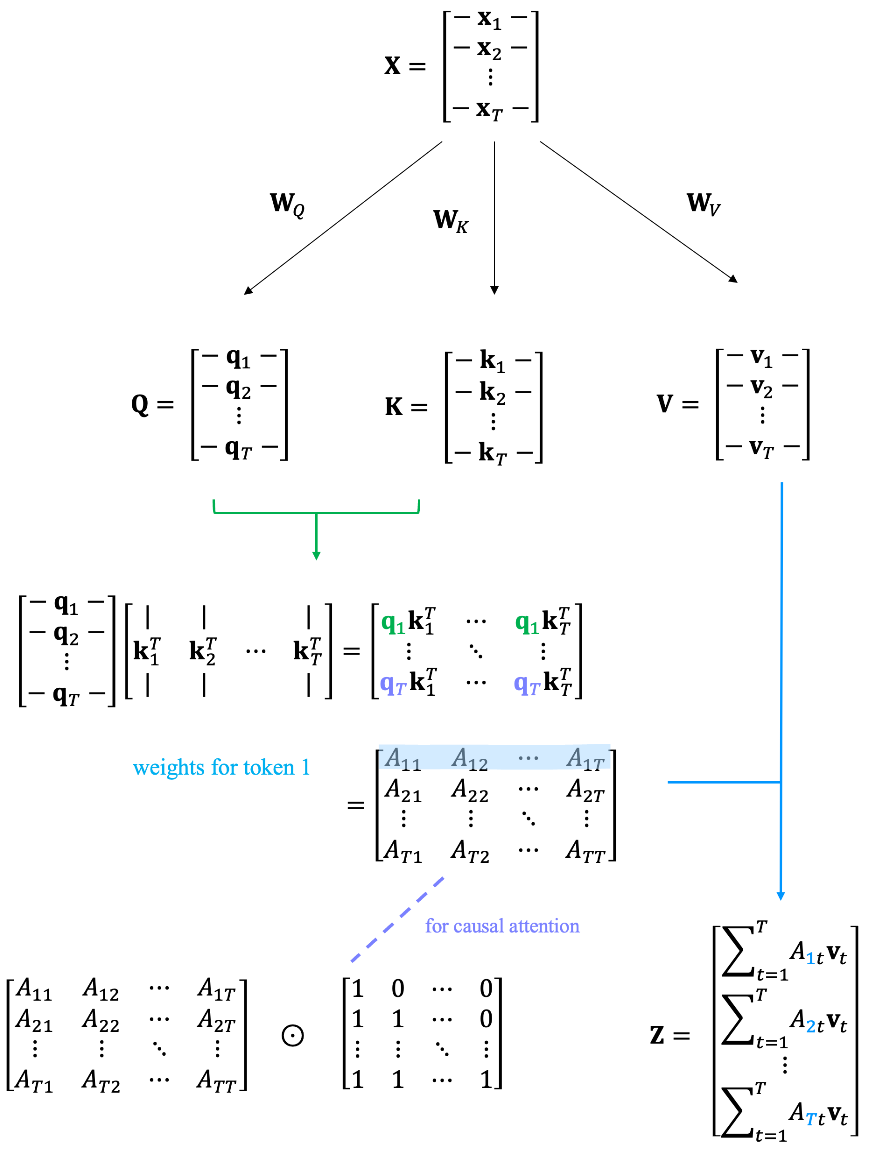
1 Attention

1.1 Self Attention

Given input $\boldsymbol{X} \in \mathbb{R}^{T\times D}$ where T is the sequence length and D is the embedding dimension, we have

$$\boldsymbol{Q} = \boldsymbol{W}_Q\boldsymbol{X} \in \mathbb{R}^{T\times d_k} \quad \boldsymbol{K} = \boldsymbol{W}_K\boldsymbol{X} \in \mathbb{R}^{T\times d_k} \quad \boldsymbol{V} = \boldsymbol{W}_V\boldsymbol{X} \in \mathbb{R}^{T\times d_v} \tag{1}$$

$$\text{Attn}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{2}$$



$$\mathbf{X} = \begin{bmatrix} - \mathbf{x}_1 - \\ - \mathbf{x}_2 - \\ \vdots \\ - \mathbf{x}_T - \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} - \mathbf{q}_1 - \\ - \mathbf{q}_2 - \\ \vdots \\ - \mathbf{q}_T - \end{bmatrix} \quad \mathbf{K} = \begin{bmatrix} - \mathbf{k}_1 - \\ - \mathbf{k}_2 - \\ \vdots \\ - \mathbf{k}_T - \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} - \mathbf{v}_1 - \\ - \mathbf{v}_2 - \\ \vdots \\ - \mathbf{v}_T - \end{bmatrix}$$

$$\begin{bmatrix} - \mathbf{q}_1 - \\ - \mathbf{q}_2 - \\ \vdots \\ - \mathbf{q}_T - \end{bmatrix}\begin{bmatrix} | & | & & | \\ \mathbf{k}_1^T & \mathbf{k}_2^T & \cdots & \mathbf{k}_T^T \\ | & | & & | \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1\mathbf{k}_1^T & \cdots & \mathbf{q}_1\mathbf{k}_T^T \\ \vdots & \ddots & \vdots \\ \mathbf{q}_T\mathbf{k}_1^T & \cdots & \mathbf{q}_T\mathbf{k}_T^T \end{bmatrix}$$

$$= \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1T} \\ A_{21} & A_{22} & \cdots & A_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ A_{T1} & A_{T2} & \cdots & A_{TT} \end{bmatrix}$$

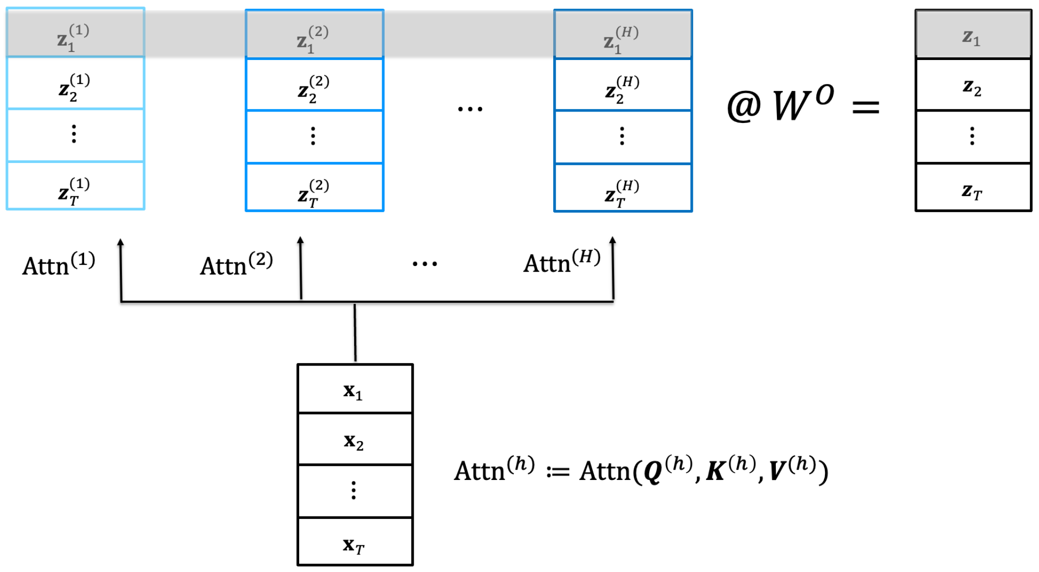
$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1T} \\ A_{21} & A_{22} & \cdots & A_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ A_{T1} & A_{T2} & \cdots & A_{TT} \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$\mathbf{Z} = \begin{bmatrix} \sum_{t=1}^{T} A_{1t}\mathbf{v}_t \\ \sum_{t=1}^{T} A_{2t}\mathbf{v}_t \\ \vdots \\ \sum_{t=1}^{T} A_{Tt}\mathbf{v}_t \end{bmatrix}$$

1.2 Multi-Head Attention

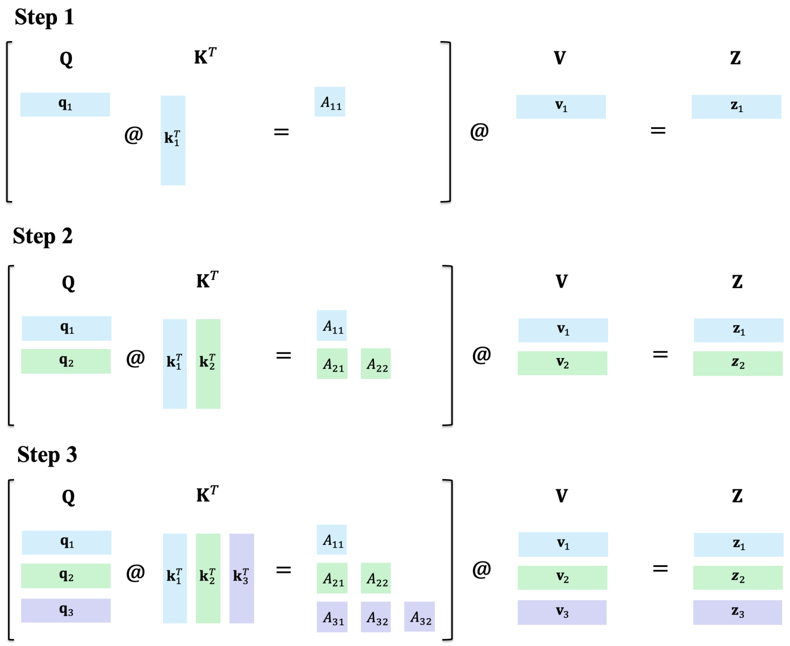
In multi-head attention, input $\boldsymbol{X}$ is first passed through H self-attention layer in parallel. Then, the output from each head is concatenated together and fused by a linear projection

$$\left[\boldsymbol{Z}^{(1)}, \boldsymbol{Z}^{(2)}, \ldots, \boldsymbol{Z}^{(H)}\right] \boldsymbol{W}^{O} = \begin{bmatrix} \boldsymbol{z}_1^{(1)} & \boldsymbol{z}_1^{(2)} & \ldots & \boldsymbol{z}_1^{(H)} \\ \boldsymbol{z}_2^{(1)} & \boldsymbol{z}_2^{(2)} & \ldots & \boldsymbol{z}_2^{(H)} \\ \vdots & \vdots & \ldots & \vdots \\ \boldsymbol{z}_T^{(1)} & \boldsymbol{z}_T^{(2)} & \ldots & \boldsymbol{z}_T^{(H)} \end{bmatrix} \boldsymbol{W}^{O} \tag{3}$$



1.3 KV-Cache

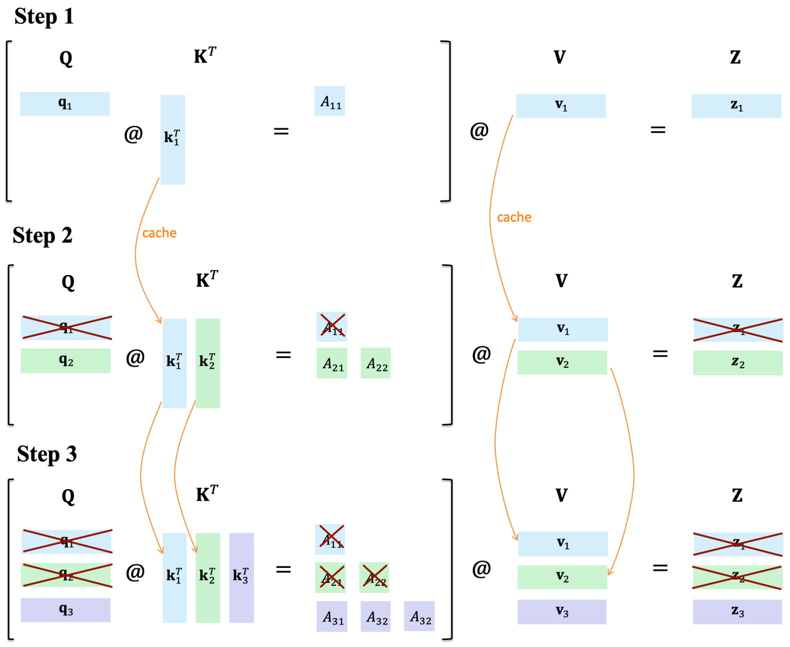
During inference we still use causal masking because this is how the model being trained. Let's look at a simple case where we only give the model a start token <s> and asks it to generate stuff:



KV-cache is built on this two observations:

- At each time step t, due to causal masking $\boldsymbol{k}_{<t}$ and $\boldsymbol{v}_{<t}$ will remain the same
- To predict <token$_{t+1}$> we only need embedding of <token$_t$>.

Therefore, we can make prediction efficiently by drop redundant and unnecessary computation



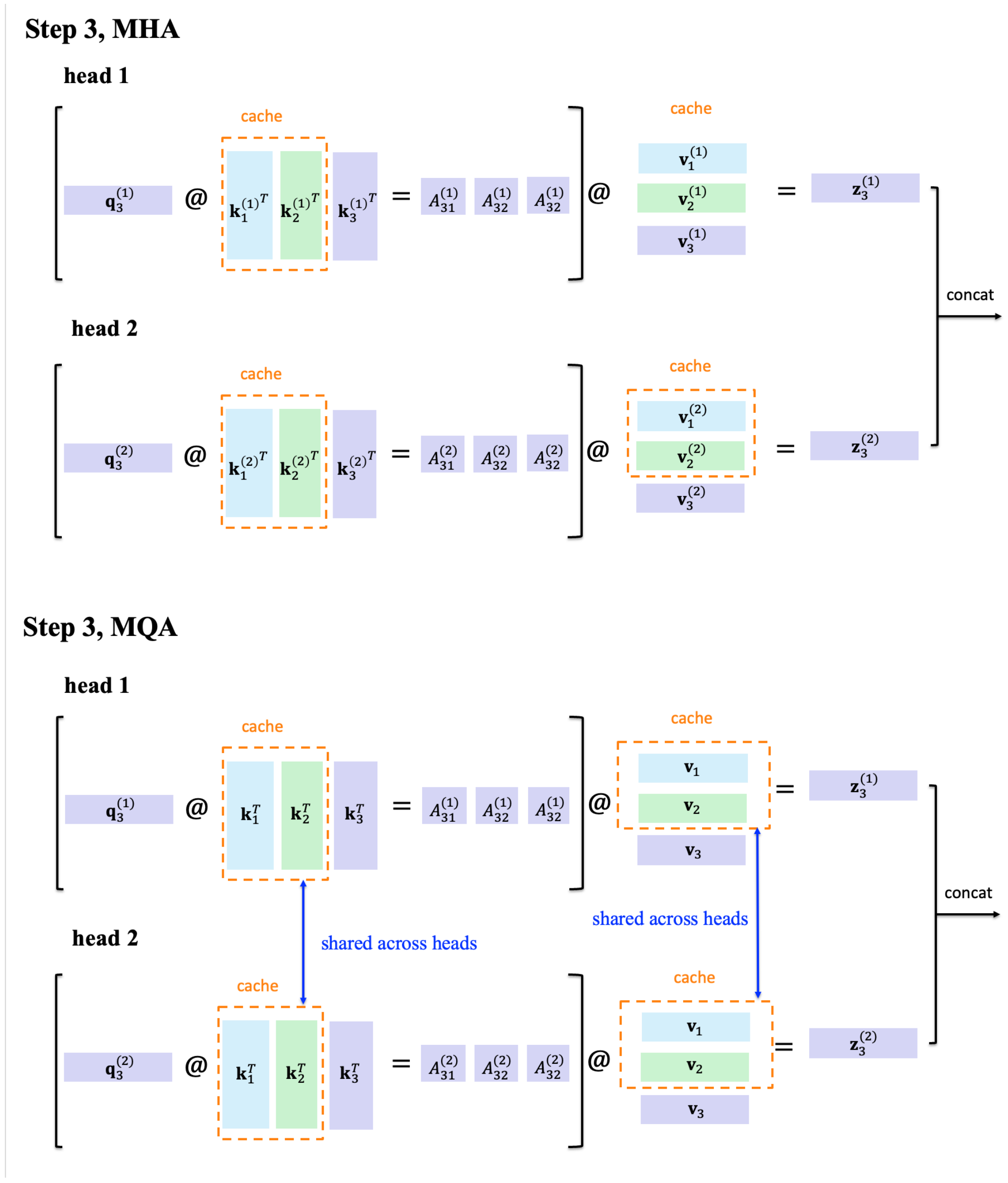
1.4 Multi-Query Attention

In KV-cache, at time step t, the past $t-1$ keys and values are cached, which corresponds to $2 * n_h * (t-1) * h_s$ memory. When t gets larger, storing them all on a single GPU will be impossible and the loading will slow things down.

The idea of multi-query attention (MQA) is quite simple. In MHA, different heads have different query, key and value projection matrix. In MQA, the key and value projection matrics will be shared across all heads, only query project matrix will be different:

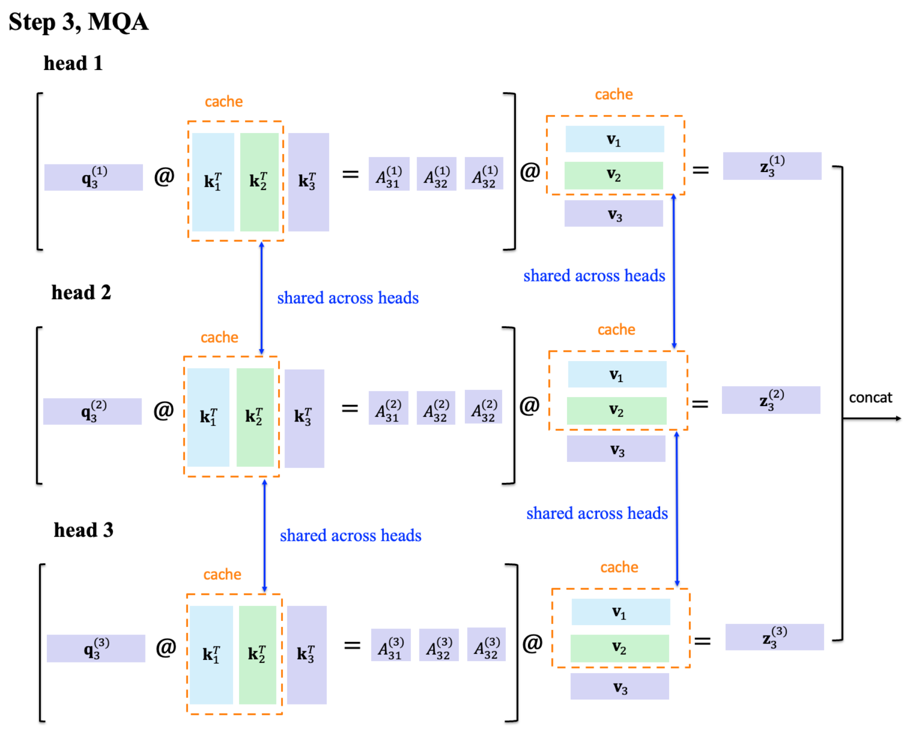
$$\begin{array}{ll} \text{MHA} & \text{MQA} \\ \boldsymbol{Q}^{(h)} = \boldsymbol{W}_Q^{(h)} \boldsymbol{X} & \boldsymbol{Q}^{(h)} = \boldsymbol{W}_Q^{(h)} \boldsymbol{X} \\ \boldsymbol{K}^{(h)} = \boldsymbol{W}_K^{(h)} \boldsymbol{X} & \boldsymbol{K}^{(h)} = \boldsymbol{W}_K \boldsymbol{X} \quad \text{(shared across heads)} \\ \boldsymbol{V}^{(h)} = \boldsymbol{W}_V^{(h)} \boldsymbol{X} & \boldsymbol{V}^{(h)} = \boldsymbol{W}_V \boldsymbol{X} \quad \text{(shared across heads)} \end{array} \tag{4}$$

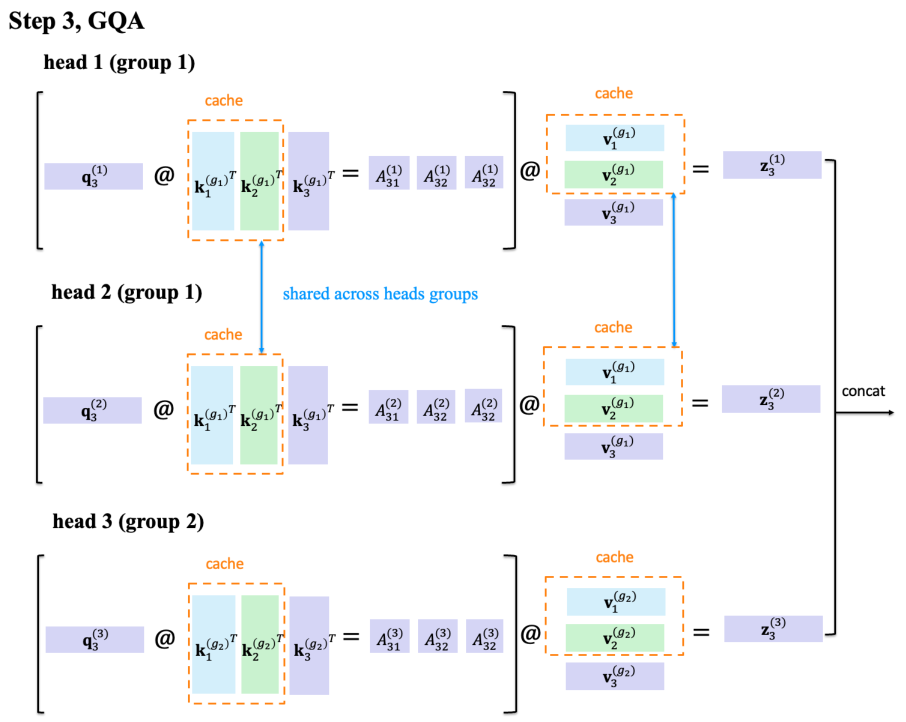
This way, we only need to store one set of KV-cache and share it across all heads.



1.5 Grouped-Query Attention

Sharing the key and value projection across all attention heads might be too brutal, in grouped-query attention, $\boldsymbol{W}_K$ and $\boldsymbol{W}_V$ are shared in grouped heads. So basically different groups have different $\boldsymbol{K}$ and $\boldsymbol{V}$.


Step 3, MQA
head 1
cache
$\mathbf{q}_3^{(1)}$ @ $\mathbf{k}_1^T$ $\mathbf{k}_2^T$ $\mathbf{k}_3^T$ = $A_{31}^{(1)}$ $A_{32}^{(1)}$ $A_{32}^{(1)}$ @ $\mathbf{v}_1$ $\mathbf{v}_2$ $\mathbf{v}_3$ = $\mathbf{z}_3^{(1)}$
shared across heads
head 2
$\mathbf{q}_3^{(2)}$ @ $\mathbf{k}_1^T$ $\mathbf{k}_2^T$ $\mathbf{k}_3^T$ = $A_{31}^{(2)}$ $A_{32}^{(2)}$ $A_{32}^{(2)}$ @ $\mathbf{v}_1$ $\mathbf{v}_2$ $\mathbf{v}_3$ = $\mathbf{z}_3^{(2)}$
concat
head 3
$\mathbf{q}_3^{(3)}$ @ $\mathbf{k}_1^T$ $\mathbf{k}_2^T$ $\mathbf{k}_3^T$ = $A_{31}^{(3)}$ $A_{32}^{(3)}$ $A_{32}^{(3)}$ @ $\mathbf{v}_1$ $\mathbf{v}_2$ $\mathbf{v}_3$ = $\mathbf{z}_3^{(3)}$


Step 3, GQA
head 1 (group 1)
cache
$\mathbf{q}_3^{(1)}$ @ $\mathbf{k}_1^{(g_1)^T}$ $\mathbf{k}_2^{(g_1)^T}$ $\mathbf{k}_3^{(g_1)^T}$ = $A_{31}^{(1)}$ $A_{32}^{(1)}$ $A_{32}^{(1)}$ @ $\mathbf{v}_1^{(g_1)}$ $\mathbf{v}_2^{(g_1)}$ $\mathbf{v}_3^{(g_1)}$ = $\mathbf{z}_3^{(1)}$
shared across heads groups
head 2 (group 1)
$\mathbf{q}_3^{(2)}$ @ $\mathbf{k}_1^{(g_1)^T}$ $\mathbf{k}_2^{(g_1)^T}$ $\mathbf{k}_3^{(g_1)^T}$ = $A_{31}^{(2)}$ $A_{32}^{(2)}$ $A_{32}^{(2)}$ @ $\mathbf{v}_1^{(g_1)}$ $\mathbf{v}_2^{(g_1)}$ $\mathbf{v}_3^{(g_1)}$ = $\mathbf{z}_3^{(2)}$
concat
head 3 (group 2)
$\mathbf{q}_3^{(3)}$ @ $\mathbf{k}_1^{(g_2)^T}$ $\mathbf{k}_2^{(g_2)^T}$ $\mathbf{k}_3^{(g_2)^T}$ = $A_{31}^{(3)}$ $A_{32}^{(3)}$ $A_{32}^{(3)}$ @ $\mathbf{v}_1^{(g_2)}$ $\mathbf{v}_2^{(g_2)}$ $\mathbf{v}_3^{(g_2)}$ = $\mathbf{z}_3^{(3)}$

2 Positional Embedding

References