LLM NOTES

A PREPRINT

July 18, 2025

Contents

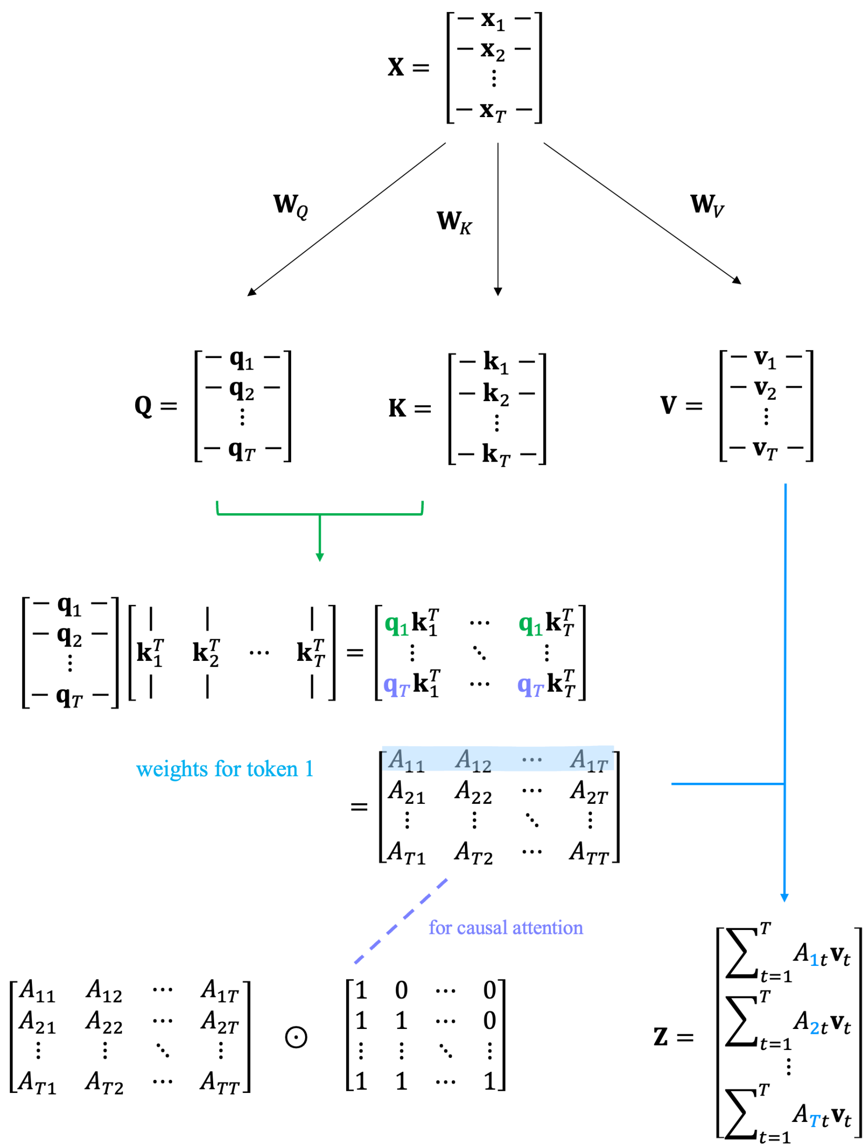
1 Attention

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

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

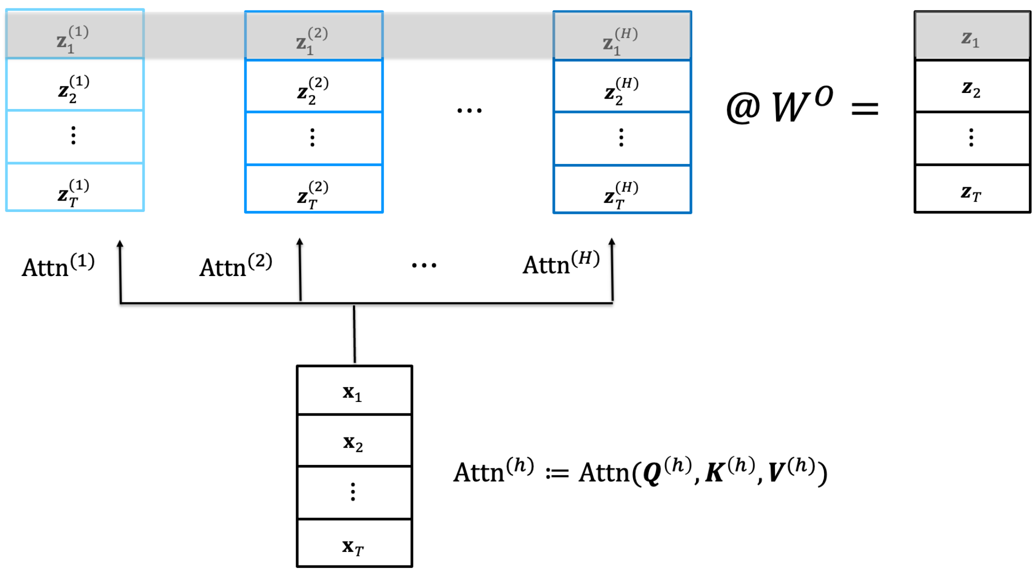
$$\operatorname{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \operatorname{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{2}$$



1.2 Multi-Head Attention

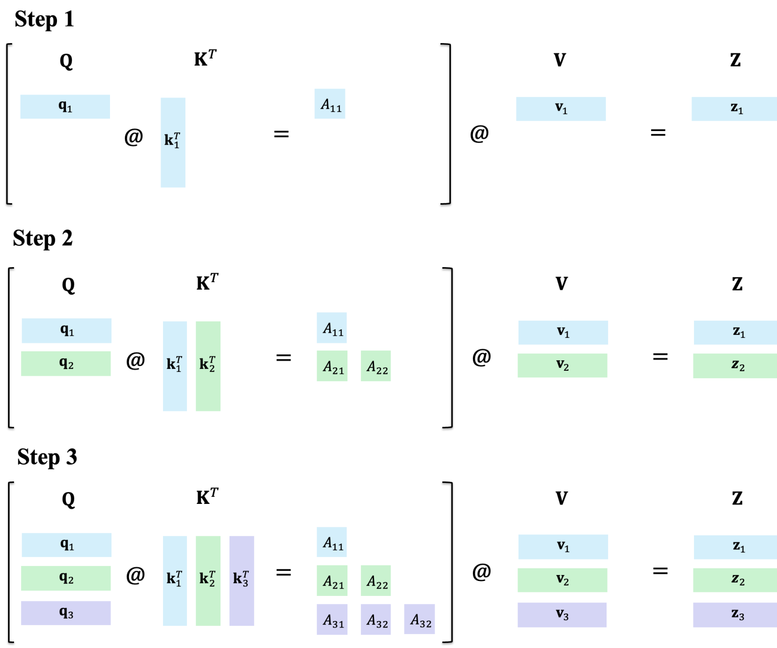
In multi-head attention, input $\boldsymbol{X}$ is first passed through H self-attention layer in parallel. Then, the output from each head is concatenated together and fused by a linear projection

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



1.3 KV-Cache

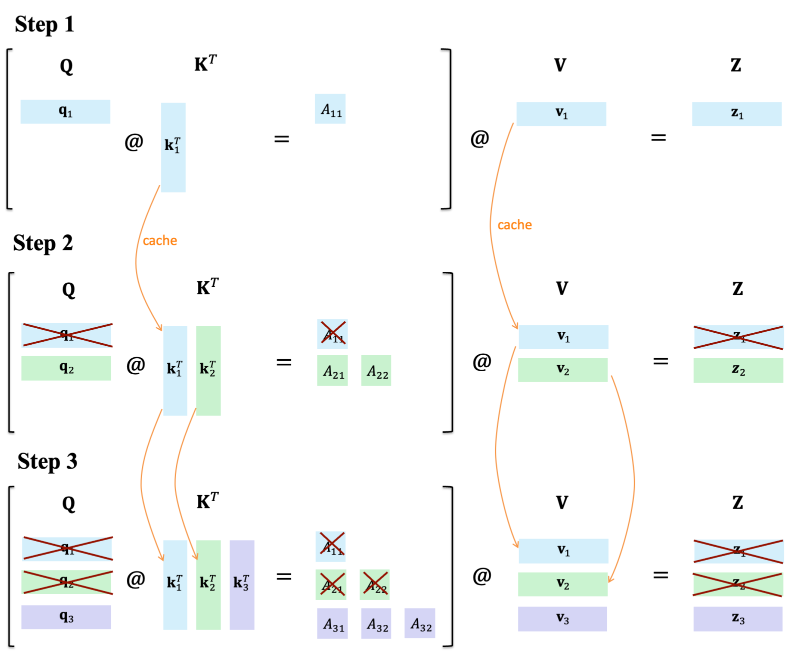
During inference we still use causal masking because this is how the model being trained. Let's look at a simple case where we only give the model a start token <s> and asks it to generate stuff:



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



Basically we have

Token 1: [K1, V1] → Cache: [K1, V1]

Token 2: [K2, V2] → Cache: [K1, K2], [V1, V2]

...

Token n: [Kn, Vn] → Cache: [K1, K2, ..., Kn], [V1, V2, ..., Vn]

2 Positional Embedding

References