< Let's Build Intuition >

Some credit note
Sarah Poland's note
thank you

The ducks were swimming near the bank.

The → ducks → were → ...

① problem with long context
② local focus, big picture

He went to the bank for his credit card.
The ducks were swimming near the bank.

① embeddings ② n^{th} token context info from all tokens:
↳ i) relevance of token i to token n (attention)
ii) content of i^{th} token (value - context)

nth token embedding with context:

$$y_n = x_n + \sum_{i=1}^{N} \alpha_{ni} v_i = x_n + \sum_{i=1}^{N} \alpha_{ni} (W^{(v)})^\top x_n$$

① nth token embedding
② learned attention weight (relevance of token i to token n) (i, n)
context provided by ith token
learned context matrix

Now, how do we capture this "attention" – relevance?

let's think of a traditional search engine / query engine for document

query: new iphone

key: apple computer . . . .

key: apple new iphone demo . . .

Conclusion
we need Q, K
↳ giving attention weights
tied with value

Capturing Similarity via Inner Product

Token Keys

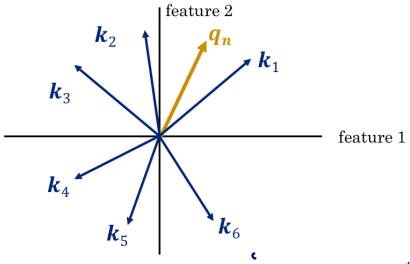
For a weight matrix $W^{(k)} \in \mathbb{R}^{D \times D_k}$, we can define the key vector of the nth token embedding as

$k_n = (W^{(k)})^\top x_n \in \mathbb{R}^{D_k}$

Token Queries

For a weight matrix $W^{(q)} \in \mathbb{R}^{D \times D_k}$, we can define the query vector of the nth token embedding as

$q_n = (W^{(q)})^\top x_n \in \mathbb{R}^{D_k}$



Similarity between q_n and k_i: $z_{n,i} = q_n^\top k_i^*$

$z_{n,1} > z_{n,2} > z_{n,3} > z_{n,6} > z_{n,4} > z_{n,5}$

*Assume vectors have unit norm.

extracts

input X → $K = XW^{(k)}$, $Q = XW^{(q)}$, $V = XW^{(v)}$

< Attention weight >

Token Similarities

$\mathbf{Z} = \mathbf{Q}\,\mathbf{K}^\top \in \mathbb{R}^{N\times N}$

Similarity between q_n and k_i:

$z_{ni} = (\mathbf{Q}\,\mathbf{K}^\top)_{ni} = q_n^\top k_i$

$$Q = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad K = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$$

query token 1 [1 2]
query token 2 [3 4]

K token 1 [5 7]
K token 2 [6 8]

K^T

★ Token Relevance $= QK^T$

but wait, we want $a_{ni} \geq 0$, $\sum_{i=1}^{n} a_{ni} = 1$

$$attn\_scores = softmax(\frac{QK^T}{\sqrt{d_K}})$$

shape=[n, n]

softmax is done over rows

Divide by sqrt(d) to avoid issues with vanishing gradients when d is large

Self-attention
Probability score matrix

	Hello	I	love	you	
Hello	0.8	0.1	0.05	0.05	Softmax()
I	0.1	0.6	0.2	0.1	...
love	0.05	0.2	0.65	0.1	
you	0.2	0.1	0.1	0.6	Softmax()

★ i^{th} row of QK^T represents how much x_i should attend to every other token x_j

YAY! we have attention

① embeddings ② n^{th} token context info from all tokens:

- i) relevance of token i to token n (attention)
- ii) content of i^{th} token (value - content)

nth token embedding with context:

$$y_n = x_n + \sum_{i=1}^{N} \alpha_{ni} v_i = x_n + \sum_{i=1}^{N} \alpha_{ni} (W^{(v)})^\top x_n$$

- x_n: nth token embedding
- α_{ni}: learned attention weight (relevance of token i to token n)
- v_i: context provided by ith token
- $W^{(v)}$: learned context matrix

<Attention weight & value>

Self-attention
Probability score matrix

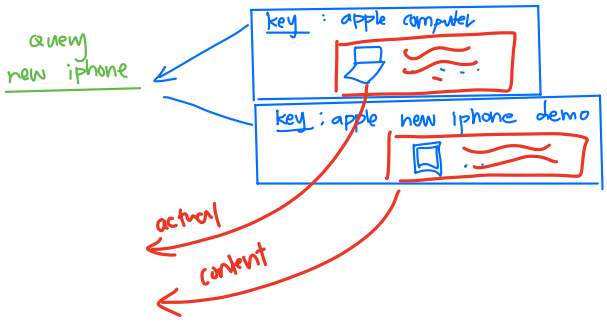
	Hello	I	love	you	
Hello	0.8	0.1	0.05	0.05	Softmax()
I	0.1	0.6	0.2	0.1	...
love	0.05	0.2	0.65	0.1	
you	0.2	0.1	0.1	0.6	Softmax()

$$attn\_scores = softmax(\frac{QK^T}{\sqrt{d}})$$ shape=[n, n]

softmax is done over rows

Divide by sqrt(d) to avoid issues with vanishing gradients when d is large

* ith row of Attention represents how much xi should attend to every other token xj



$$A = \begin{matrix} & \text{token 1} & \text{token 2} \\ \text{token 1} & a_{1,1} & a_{1,2} \\ \text{token 2} & a_{2,1} & a_{2,2} \end{matrix} \qquad V = \begin{matrix} & \text{values} \\ \text{token 1} & \vec{v_1}^T \\ \text{token 2} & \vec{v_2}^T \end{matrix}$$

$$AV = \begin{bmatrix} (a_{1,1} \cdot \vec{v_1}^T) + (a_{1,2} \cdot \vec{v_2}^T) \\ (a_{2,1} \cdot \vec{v_1}^T) + (a_{2,2} \cdot \vec{v_2}^T) \end{bmatrix}$$

how much attention? (1,1) (1,2)

If you want visualization

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 5+14 & 6+16 \\ 15+28 & 18+32 \end{bmatrix}$$

2x2 2x2

↳ 2x1 1x2 = 2x2

2x1 1x2

$$\begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} 5 & 6 \end{bmatrix} = \begin{bmatrix} 5 & 6 \\ 15 & 18 \end{bmatrix}$$

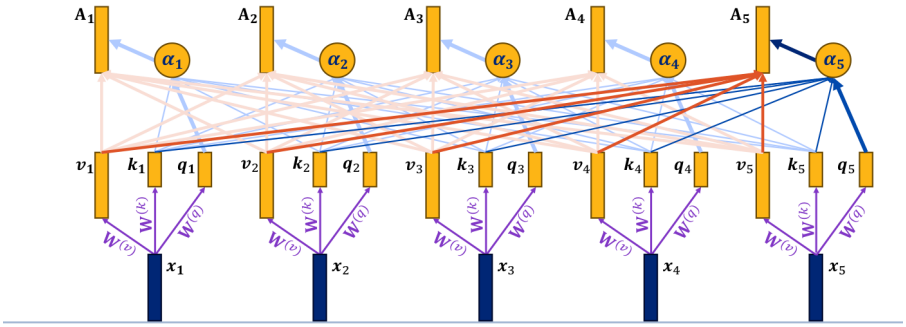
+

$$\begin{bmatrix} 2 \\ 4 \end{bmatrix} \begin{bmatrix} 7 & 8 \end{bmatrix} = \begin{bmatrix} 14 & 16 \\ 28 & 32 \end{bmatrix}$$

Bank next to river gives us global context!

The Self Attention Layer

$$\text{Attention}[\mathbf{K}, \mathbf{Q}, \mathbf{V}] = \boldsymbol{\alpha}\,\mathbf{V} = \sum_{i=1}^{N} \alpha_{ni} \boldsymbol{v}_i$$



nth token embedding with context:

$$y_n = x_n + \sum_{i=1}^{N} \alpha_{ni} v_i = x_n + \sum_{i=1}^{N} \alpha_{ni} (W^{(v)})^T x_n$$

- x_n: nth token embedding
- α_{ni}: learned attention weight (relevance of token i to token n)
- v_i: context provided by ith token
- $W^{(v)}$: learned context matrix

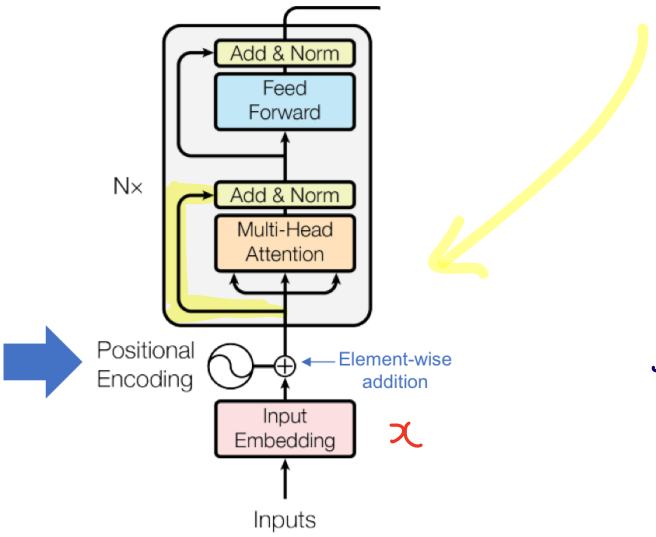
① embeddings

② n^{th} token

context info from all tokens:

- i) relevance of token i to token n (attention)
- ii) content of i^{th} token (value - context)

$$Y = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$



just do multiple versions & stack them up

MHA (v1): multiple heads

Let `h` be the number of heads.
(1) Run h different independent self-attention blocks, to produce h different outputs with shape=[n, d]
(2) **Concatenate** all h outputs, and apply a learned linear transformation to produce the MHA output (shape=[n, d])

$$\mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^{(q)}$$
$$\mathbf{K}_h = \mathbf{X}\mathbf{W}_h^{(k)}$$
$$\mathbf{V}_h = \mathbf{X}\mathbf{W}_h^{(v)}$$
$$\mathbf{H}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h)$$

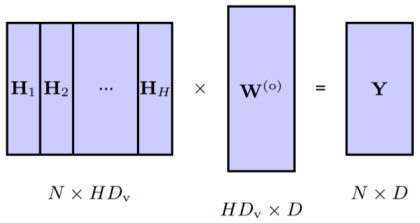
h different Q, K, V, and attention scores!

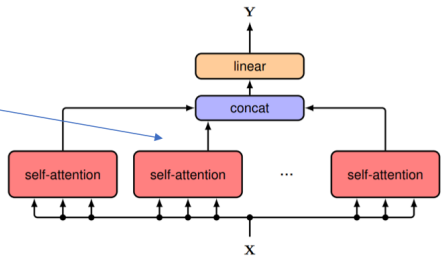
$$\mathbf{Y}(\mathbf{X}) = \text{Concat}\left[\mathbf{H}_1, \ldots, \mathbf{H}_H\right]\mathbf{W}^{(o)}$$

shape=[n, h*d]

Learned linear transform. Shape=[h*d, d]

Note: h*d is pretty large! See next slide for what's done in practice ("split").





Nice property: we can easily stack MHA vertically ("depth")!

Orig paper uses 6 layers, with h=8 heads

Data 188 Introduction to Deep Learning
Spring 2026 Eric Kim

Discussion 08

This week we will work through a set of exercises on attention tensor shapes, the structure of multi-head attention, hand-computed single-head and multi-head forward passes, and the block-matrix view used in practical MHA implementations.

1. Single-Head Attention Shapes

The figure below shows a vanilla single-head attention block. Assume the input tensor has shape $X \in \mathbb{R}^{B\times n\times d}$, where B is the batch size, n is the sequence length, and d is the token embedding dimensionality. Assume further that the dimension of the key vector d_k and dimension of the value vector d_v for the attention are both equal to d. This question asks you to determine the shapes of relevant tensors.

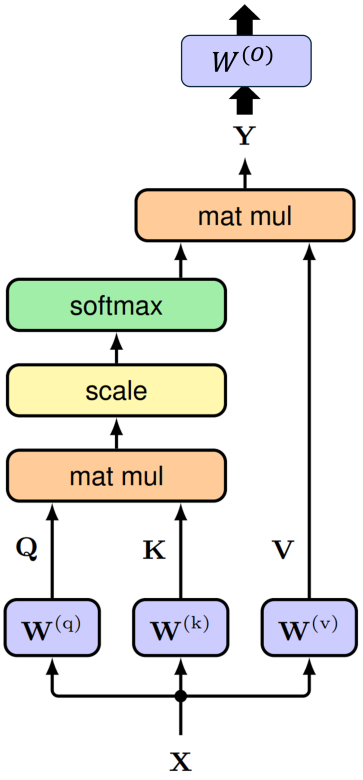


Figure 1: Single-head attention block.

(a) Please provide the shapes for the query/key/value input projection weight matrices $W^{(q)}, W^{(k)}, W^{(v)}$.

Solution: $W^{(q)} \in \mathbb{R}^{d\times d}, W^{(k)} \in \mathbb{R}^{d\times d}, W^{(v)} \in \mathbb{R}^{d\times d}$ since each linear projection weight matrix maps from the input embedding space of dimension d to the query/key/value space of dimension d.

Although not commonly used in practice, we could also design the attention block to have arbitrary query-key/value dimensions. Generally, we could specify the dimension of the key vector d_k and the dimension of the value vector d_v independently. Note that since the query vectors need to do inner product with the key vectors, the dimension of the query vectors should also be d_k. In this case, the shapes of the projection weight matrices would be $W^{(q)} \in \mathbb{R}^{d \times d_k}, W^{(k)} \in \mathbb{R}^{d \times d_k}, W^{(v)} \in \mathbb{R}^{d \times d_v}$.

(b) Please provide the shapes for the query/key/value activations Q, K, V after the input projection.

Solution: $Q \in \mathbb{R}^{B \times n \times d}, K \in \mathbb{R}^{B \times n \times d}, V \in \mathbb{R}^{B \times n \times d}$ since $Q = XW^{(q)}, K = XW^{(k)}, V = XW^{(v)}$.

(c) Please provide the shapes for the result after the first matmul $QK^\top$, the scaled matmul result $\frac{QK^\top}{\sqrt{d}}$, and the softmax result (i.e. the attention weights) $\text{Softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)$. Note that the formulae here are for unbatched inputs.

Solution: The shapes are $QK^\top \in \mathbb{R}^{B \times n \times n}$ since we apply the matmul formula independently for each element in the batch (i.e. we broadcast the matmul over the batch dimension), $\frac{QK^\top}{\sqrt{d}} \in \mathbb{R}^{B \times n \times n}$ has the same shape, and $\text{Softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) \in \mathbb{R}^{B \times n \times n}$ again has the same shape.

(d) Please provide the shapes for the result of the second matmul Y, the output projection weight matrix $W^{(o)}$, and the final output of the attention block after the output projection $YW^{(o)}$.

Solution: $W^{(o)} \in \mathbb{R}^{d \times d}$ because the output projection maps from the pre-linear attention output (which has dimension d) back to the input embedding space (which also has dimension d).

The shapes are $Y = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V \in \mathbb{R}^{B \times n \times d}$ since we apply the matmul formula independently for each element in the batch, $W^{(o)} \in \mathbb{R}^{d \times d}$, and $YW^{(o)} \in \mathbb{R}^{B \times n \times d}$.

2. Multi-Head Attention Mechanism

The figure below shows a sketch of the multi-head attention mechanism. In this question we will review how we can have multiple attention heads from a single input, and how each head's outputs are combined to produce a single MHA output tensor.

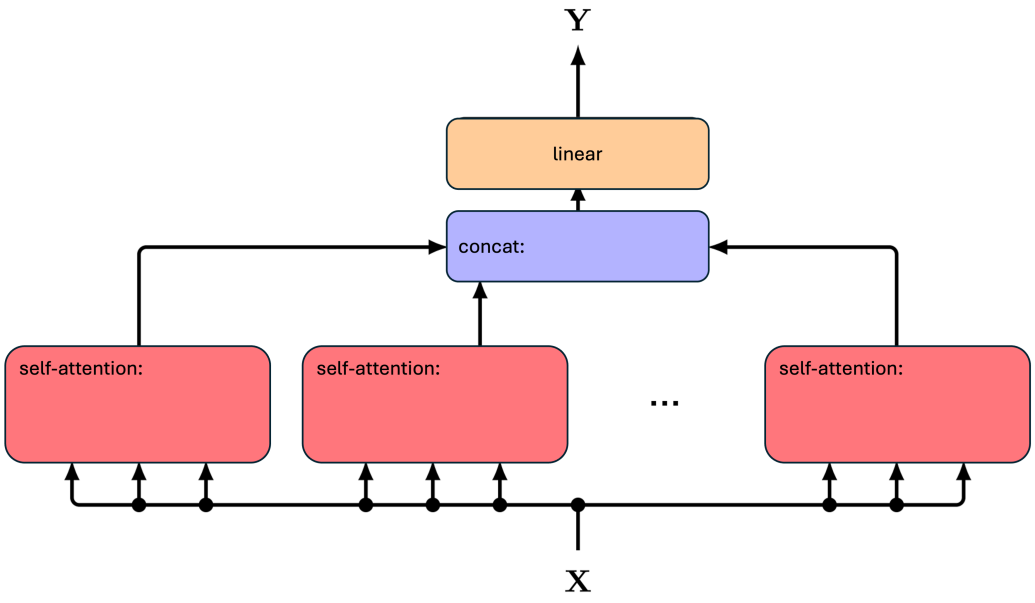


Figure 2: Multi-head attention block.

(a) Please draw on the figure where each query/key/value input projection weight matrices $W_h^{(q)}, W_h^{(k)}, W_h^{(v)}$ are located.

(b) Please draw on the figure where each query/key/value activation Q_h, K_h, V_h and the per-head attention output H_h are calculated.

(c) Please draw on the figure how the per-head attention outputs $H_1, H_2, \ldots, H_{n_{\text{heads}}}$ are combined to produce a single tensor before the final output projection linear layer.

Solution:

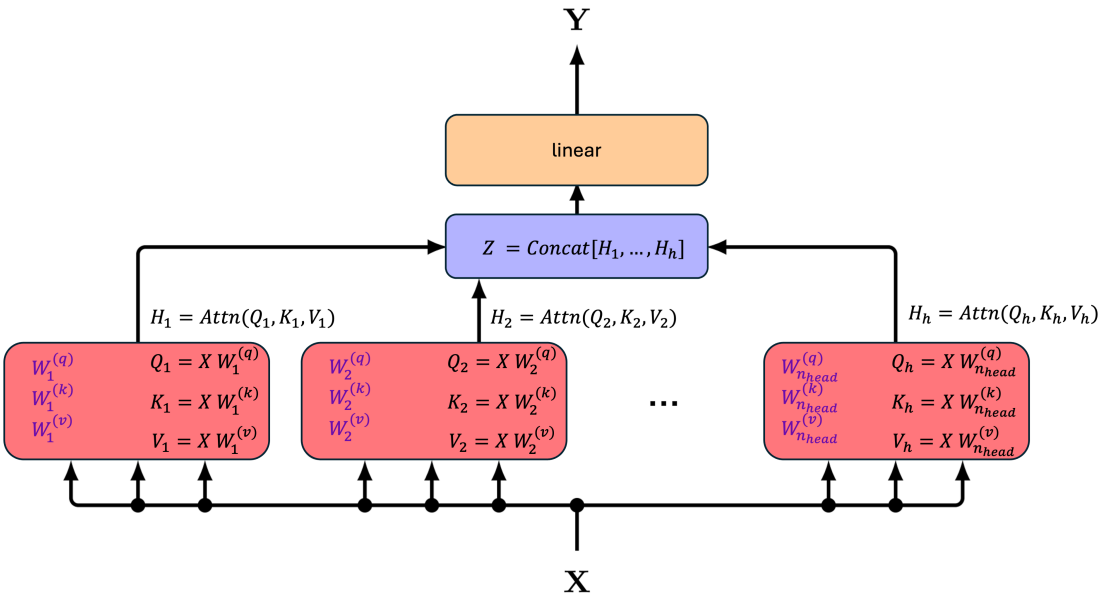


Figure 3: Multi-head attention block with solution annotations.

3. Single Head Attention Forward Pass

Recall the single head attention mechanism:

$$
\begin{aligned}
Q &= XW^{(q)} \\
K &= XW^{(k)} \\
V &= XW^{(v)} \\
H &= \text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \\
Y &= HW^{(o)}
\end{aligned}
$$

In this question, we consider a toy example where sequence length $n = 2$, hidden dimension $d = 2$, and the input is not batched. Assume the input $X \in \mathbb{R}^{n\times d}$, the input projection weights $W^{(q)}, W^{(k)}, W^{(v)} \in \mathbb{R}^{d\times d}$, and the output projection weight $W^{(o)} \in \mathbb{R}^{d\times d}$ bear the following values:

$$
X = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad W^{(q)} = \begin{bmatrix} 3 & -1 \\ 1 & 1 \end{bmatrix}, \quad W^{(k)} = \begin{bmatrix} 2 & -2 \\ 1 & 3 \end{bmatrix}, \quad W^{(v)} = \begin{bmatrix} 2 & -2 \\ 0 & 2 \end{bmatrix}, \quad W^{(o)} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}.
$$

(a) Compute the matrices Q, K, and V.

Solution:

$$\begin{aligned} Q &= XW^{(q)} \\ &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 1 & 1 \end{bmatrix} \\ K &= XW^{(k)} \\ &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \\ V &= XW^{(v)} \\ &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \end{aligned}$$

(b) Given that the attention weights are approximately

$$\text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) \approx \begin{bmatrix} 1.00 & 0.00 \\ 0.50 & 0.50 \end{bmatrix}$$

Please compute the output Y of the single head attention.

Solution: We first compute the output of the attention head H:

$$\begin{aligned} H &= \text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \\ &= \begin{bmatrix} 1.00 & 0.00 \\ 0.50 & 0.50 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix} \end{aligned}$$

Then we compute the final output Y:

$$\begin{aligned} Y &= HW^{(o)} \\ &= \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 2 & 1 \end{bmatrix} \end{aligned}$$

4. MHA with Block Matrices

For multi-head attention with number of heads $n_{\text{heads}} > 1$, in practice we often do not explicitly keep separate projection matrices $W_h^{(q)}, W_h^{(k)}, W_h^{(v)}$ for each head in the forward computation. Instead, we form single large projection matrices $W^{(q)}, W^{(k)}, W^{(v)}$ and compute

$$\begin{aligned} Q &= XW^{(q)} \\ K &= XW^{(k)} \\ V &= XW^{(v)}. \end{aligned}$$

Then each head operates on the appropriate feature chunk of Q, K, and V.

For example, if $n_{\text{heads}} = 2$ and hidden dimension $d = 64$, then each head has size $d_h = 32$. Head 0 operates on $Q[:, :32]$, $K[:, :32]$, $V[:, :32]$, and head 1 operates on $Q[:, 32:]$, $K[:, 32:]$, $V[:, 32:]$.

To show why this is valid, recall the block-matrix identity (here the square brackets denote matrix concatenation instead of indexing)

$$A[B_0 \mid B_1] = [AB_0 \mid AB_1].$$

Here B_0 and B_1 are a split of B along its column axis.

(a) Consider the two-head case $n_{\text{heads}} = 2$ and assume $d = 2d_h$. Write

$$W^{(q)} = [W_0^{(q)} \mid W_1^{(q)}],$$

where $W_0^{(q)}, W_1^{(q)} \in \mathbb{R}^{d \times d_h}$. Show that computing $Q = XW^{(q)}$ and then splitting Q along the feature dimension is equivalent to separately computing $Q_0 = XW_0^{(q)}$ and $Q_1 = XW_1^{(q)}$.

Solution: Using the block-matrix identity,

$$\begin{aligned} Q &= XW^{(q)} \\ &= X[W_0^{(q)} \mid W_1^{(q)}] \\ &= [XW_0^{(q)} \mid XW_1^{(q)}]. \end{aligned}$$

If we define

$$Q_0 = XW_0^{(q)} \quad \text{and} \quad Q_1 = XW_1^{(q)},$$

then

$$Q = [Q_0 \mid Q_1].$$

Therefore, computing the full matrix product $XW^{(q)}$ and then splitting the output into two chunks is exactly the same as separately computing the per-head query matrices Q_0 and Q_1.

(b) Explain why the same argument applies to K and V, and conclude why chunking Q, K, and V after the large projection is equivalent to explicitly keeping track of separate per-head projection matrices.

Solution: The same reasoning applies to the key and value projections:

$$\begin{aligned} W^{(k)} &= [W_0^{(k)} \mid W_1^{(k)}], & K &= XW^{(k)} = [XW_0^{(k)} \mid XW_1^{(k)}] = [K_0 \mid K_1], \\ W^{(v)} &= [W_0^{(v)} \mid W_1^{(v)}], & V &= XW^{(v)} = [XW_0^{(v)} \mid XW_1^{(v)}] = [V_0 \mid V_1]. \end{aligned}$$

So the first d_h columns of Q, K, V are exactly the tensors for head 0, and the next d_h columns are exactly the tensors for head 1. Therefore each head sees exactly the same Q_h, K_h, V_h that it would have seen if we had computed separate per-head projections from the start.

Hence, using a single large projection matrix followed by splitting into chunks is mathematically equivalent to explicitly keeping track of separate $W_h^{(q)}, W_h^{(k)}, W_h^{(v)}$ matrices for each head. The same argument extends immediately from 2 heads to n_{heads} heads by writing each projection matrix as a concatenation of n_{heads} column blocks.

Note: be careful about calculating the attention weights. In PyTorch, usually we could reshape Q, K, V to have shape $B \times n \times n_{\text{heads}} \times d_h$. For HW3, reading the torch batched matrix multiplication documentation or using einops can be helpful.

5. Multi-Head Attention Forward Pass

Recall the multi-head attention equations:

$$\begin{aligned} Q_h &= XW_h^{(q)} \\ K_h &= XW_h^{(k)} \\ V_h &= XW_h^{(v)} \\ H_h &= \text{Attention}(Q_h, K_h, V_h) \\ Y &= \text{Concat}[H_0, ..., H_{n_{\text{heads}}-1}]W^{(o)}. \end{aligned}$$

Adopting the block matrix convention introduced in the previous question, we can group together

$$\begin{aligned} Q &= [Q_0 \mid Q_1 \mid ... \mid Q_{n_{\text{heads}}-1}] = X[W_0^{(q)} \mid W_1^{(q)} \mid ... \mid W_{n_{\text{heads}}-1}^{(q)}] = XW^{(q)}, \\ K &= [K_0 \mid K_1 \mid ... \mid K_{n_{\text{heads}}-1}] = X[W_0^{(k)} \mid W_1^{(k)} \mid ... \mid W_{n_{\text{heads}}-1}^{(k)}] = XW^{(k)}, \\ V &= [V_0 \mid V_1 \mid ... \mid V_{n_{\text{heads}}-1}] = X[W_0^{(v)} \mid W_1^{(v)} \mid ... \mid W_{n_{\text{heads}}-1}^{(v)}] = XW^{(v)}. \end{aligned}$$

In this question, we consider a toy example with $n_{\text{heads}} = 2$, sequence length $n = 2$, hidden dimension $d = 4$, and per-head hidden dimension $d_h = 2$. The input is not batched, so $X, Q, K, V \in \mathbb{R}^{2\times 4}$.

(a) Suppose for a particular input X, the projected query, key, and value matrices are

$$Q = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}, \qquad K = \begin{bmatrix} 2 & 0 & 1 & 1 \\ 1 & 3 & 0 & 2 \end{bmatrix}, \qquad V = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 3 & 1 & 0 & 2 \end{bmatrix}.$$

Compute the per-head query, key, and value matrices $Q_0, Q_1, K_0, K_1, V_0, V_1$ by splitting Q, K, and V along the feature dimension.

Solution: Since each head has feature dimension $d_h = 2$, head 0 uses the first two columns and head 1 uses the last two columns:

$$Q_0 = Q[:, :2] = \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix}, \qquad Q_1 = Q[:, 2:] = \begin{bmatrix} 3 & 4 \\ 7 & 8 \end{bmatrix},$$

$$K_0 = K[:, :2] = \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}, \qquad K_1 = K[:, 2:] = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix},$$

$$V_0 = V[:, :2] = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}, \qquad V_1 = V[:, 2:] = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}.$$

(b) Suppose for another input X, the attention outputs for the two heads are

$$H_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad H_1 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix},$$

and suppose the output projection matrix is

$$W^{(o)} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Compute the concatenated matrix $H = \text{Concat}[H_0, H_1]$ and then compute the final output $Y = HW^{(o)}$.

Solution: We first concatenate the two heads along the feature dimension:

$$\begin{aligned} H &= \text{Concat}[H_0, H_1] \\ &= \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}. \end{aligned}$$

Then we multiply by the output projection matrix:

$$\begin{aligned} Y &= HW^{(o)} \\ &= \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 3 & 1 & 2 & 2 \\ 1 & 3 & 2 & 2 \end{bmatrix}. \end{aligned}$$

