1 Overview of Multi-Head Attention

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. This is achieved by linearly projecting the queries, keys, and values multiple times with different learned projections, and then applying attention to each of these projected versions in parallel.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O \tag{1}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2}$$

In the implementation provided, this projection is simplified by splitting the input matrices directly.

2 Function Explanations

2.1 split_heads

This function splits the last dimension of the input tensor into multiple heads.

Algorithm 1 split_heads(X, n_heads)

1: **Input:** X of shape (seq_len, d_k), n_heads
2: **Output:** Reshaped X with shape (n_heads, seq_len, d_k/n_heads)
3: seq_len, d_k ← X.shape
4: Assert d_k is divisible by n_heads
5: d_head ← d_k / n_heads
6: X_reshaped ← reshape X to (seq_len, n_heads, d_head)
7: **return** transpose X_reshaped to (n_heads, seq_len, d_head)

Mathematically, this operation transforms a matrix $X \in \mathbb{R}^{L \times D}$ into $X' \in \mathbb{R}^{h \times L \times \frac{D}{h}}$ where L is the sequence length, D is the feature dimension, and h is the number of heads.

2.2 combine_heads

This function reverses the split_heads operation to combine the outputs from multiple attention heads.

This transforms $X \in \mathbb{R}^{h \times L \times \frac{D}{h}}$ back to $X' \in \mathbb{R}^{L \times D}$.

2.3 scaled_dot_product_attention

This function implements the core attention mechanism.

Mathematically:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

Algorithm 2 combine_heads(X)

Input: X of shape (n_heads, seq_len, d_head)
Output: Combined tensor of shape (seq_len, n_heads*d_head)
n_heads, seq_len, d_head ← X.shape
X_transposed ← transpose X to (seq_len, n_heads, d_head)
return reshape X_transposed to (seq_len, n_heads * d_head)

Algorithm 3 scaled_dot_product_attention(Q, K, V, mask)

Input: Q, K, V of shapes (n_heads, seq_len, d_head), optional mask
Output: Attention output of shape (n_heads, seq_len, d_head)
d_head ← K.shape[2]
attention_scores ← matmul(Q, transpose(K, (0, 2, 1)))
attention_scores ← attention_scores / $\sqrt{d\_head}$
if mask is not None **then**
    attention_scores ← attention_scores + mask
end if
attention_weights ← exp(attention_scores)
attention_weights ← attention_weights / sum(attention_weights, axis=-1, keepdims=True)
output ← matmul(attention_weights, V)
return output

2.4 multi_head_attention

This function orchestrates the entire multi-head attention process.

Algorithm 4 multi_head_attention(Q, K, V, n_heads)

Input: Q, K, V of shapes (seq_len, d_model), n_heads
Output: Multi-head attention output of shape (seq_len, d_model)
Q_split ← split_heads(Q, n_heads)
K_split ← split_heads(K, n_heads)
V_split ← split_heads(V, n_heads)
attention_output ← scaled_dot_product_attention(Q_split, K_split, V_split)
combined_output ← combine_heads(attention_output)
return combined_output

3 Detailed Walkthrough with Example

Given the example inputs:

$$Q = K = V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad n\_heads = 2 \tag{4}$$

Let's trace through the computation step by step:

3.1 Step 1: Splitting Heads

For input matrices of shape (2, 2) with 2 heads, we get:

$$d\_head = 2/2 = 1 \tag{5}$$

$$Q\_reshaped = \text{reshape}(Q, (2, 2, 1)) = \begin{bmatrix} [1] & [0] \\ [0] & [1] \end{bmatrix} \tag{6}$$

$$Q\_split = \text{transpose}(Q\_reshaped, (1, 0, 2)) = \begin{bmatrix} [1] & [0] \\ [0] & [1] \end{bmatrix} \tag{7}$$

Due to the specific example, the reshape and transpose operations produce similar-looking outputs. For clarity:

$$Q\_split = \begin{bmatrix} [1] & [0] \\ [0] & [1] \end{bmatrix} \in \mathbb{R}^{2\times2\times1} \tag{8}$$

Similarly for K_split and V_split.

3.2 Step 2: Scaled Dot-Product Attention

$$\text{attention\_scores} = Q\_split \cdot K\_split^T \tag{9}$$

$$= \begin{bmatrix} [1] \cdot [1], [1] \cdot [0] \\ [0] \cdot [1], [0] \cdot [0] \end{bmatrix} \tag{10}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{11}$$

After scaling by $1/\sqrt{d\_head} = 1/\sqrt{1} = 1$:

$$\text{scaled\_attention\_scores} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{12}$$

Applying softmax (row-wise):

$$\text{attention\_weights} = \text{softmax}(\text{scaled\_attention\_scores}) \tag{13}$$

$$= \begin{bmatrix} \frac{e^1}{e^1+e^0} & \frac{e^0}{e^1+e^0} \\ \frac{e^0}{e^0+e^0} & \frac{e^0}{e^0+e^0} \end{bmatrix} \tag{14}$$

$$= \begin{bmatrix} \frac{e^1}{e^1+1} & \frac{1}{e^1+1} \\ 0.5 & 0.5 \end{bmatrix} \tag{15}$$

$$= \begin{bmatrix} \frac{2.718}{3.718} & \frac{1}{3.718} \\ 0.5 & 0.5 \end{bmatrix} \tag{16}$$

$$\approx \begin{bmatrix} 0.731 & 0.269 \\ 0.5 & 0.5 \end{bmatrix} \tag{17}$$

Computing the weighted sum of values:

$$\text{output} = \text{attention\_weights} \cdot V\_split \tag{18}$$

$$= \begin{bmatrix} 0.731 \cdot [1] + 0.269 \cdot [0] \\ 0.5 \cdot [1] + 0.5 \cdot [0] \end{bmatrix} \tag{19}$$

$$= \begin{bmatrix} [0.731] \\ [0.5] \end{bmatrix} \tag{20}$$

Similarly for the second head, we get:

$$\text{output\_head2} \approx \begin{bmatrix} [0.5] \\ [0.731] \end{bmatrix} \tag{21}$$

3.3 Step 3: Combining Heads

$$\text{attention\_output} = \begin{bmatrix} [0.731] & [0.5] \\ [0.5] & [0.731] \end{bmatrix} \tag{22}$$

$$\text{combined\_output} = \text{combine\_heads}(\text{attention\_output}) \tag{23}$$

$$= \begin{bmatrix} 0.731 & 0.5 \\ 0.5 & 0.731 \end{bmatrix} \tag{24}$$

This matches the output shown in the example:

$$\text{Output} = \begin{bmatrix} 0.73105858 & 0.5 \\ 0.5 & 0.73105858 \end{bmatrix} \tag{25}$$

4 Analysis and Significance

The multi-head attention mechanism allows the model to:

1. **Attend to different representation subspaces:** By splitting into multiple heads, the model can focus on different aspects of the input data simultaneously.

2. **Capture different dependencies:** Different heads can learn to attend to different types of relationships in the data.

3. **Enhance the representational power:** Multiple attention heads increase the model's capacity to represent complex functions.

The specific output [[0.73105858, 0.5], [0.5, 0.73105858]] reflects how the attention mechanism is focusing on the diagonal elements of the input matrices, which is expected given the identity-like input matrices. The value 0.73105858 is approximately $\frac{e^1}{e^1+1} \approx 0.731$, which is the softmax of 1 in the context of the given attention scores.