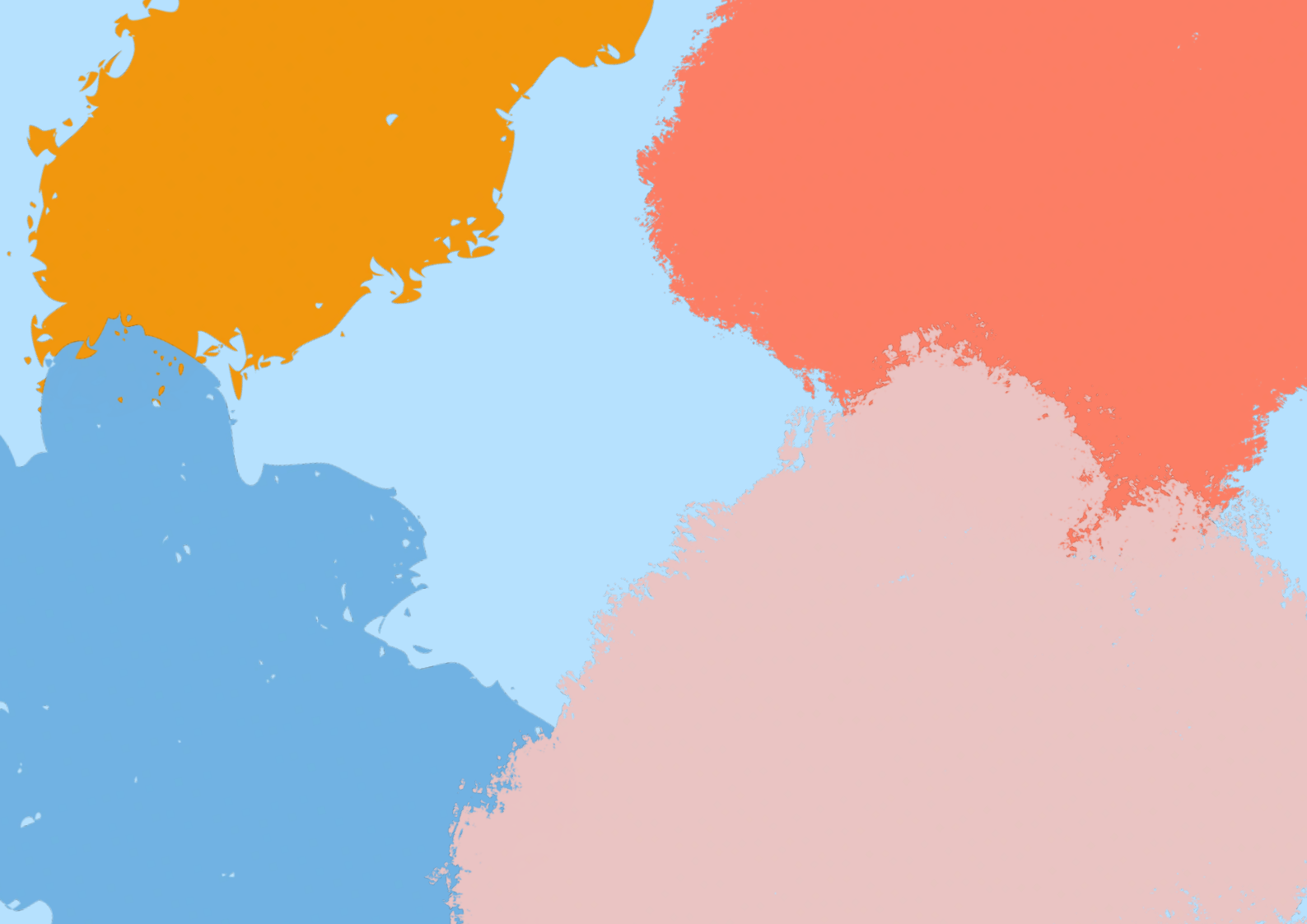
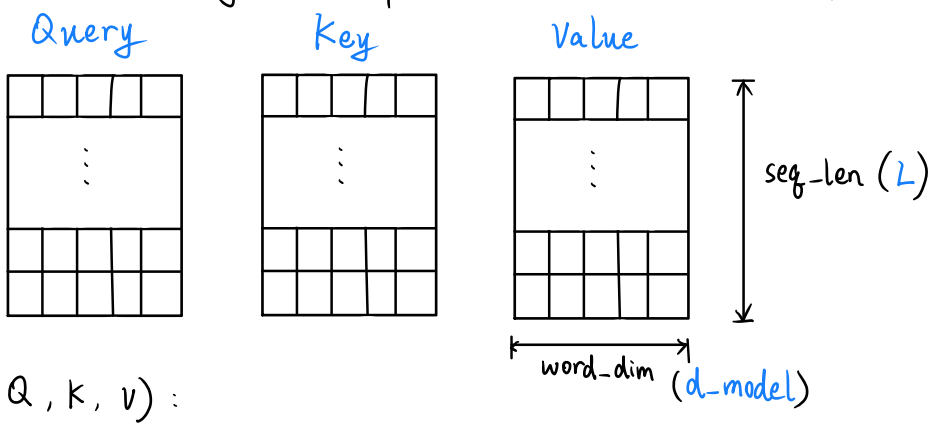


Attention：目的是计算两个序列的相似度关系矩阵（分数），也就是Q与V的相似度（矩重）

每一行代表一个词（向量），Query是你要算哪些词对Key-Value的注意力分数



def attention(Q, K, V)：

(1) $Q \cdot K^T$：每个要被查询的q对于每一个词k的关系，得到Q与K的关系矩阵

(2) $softmax(Q \cdot K^T)$：将关系矩阵归一化成和为1的权重矩阵

(3) $Softmax(Q \cdot K^T)V$：权重矩阵与真值V做乘法得到"分数"

(4) $softmax(\frac{Q \cdot K^T}{\sqrt{d_k}})V$：$d_k$为$Q \cdot K^T$的维度（$d_k \cdot d_k$），$d_k$高的话softmax会放大值之间的差异

Self-Attention：只关注自身序列的注意力分数

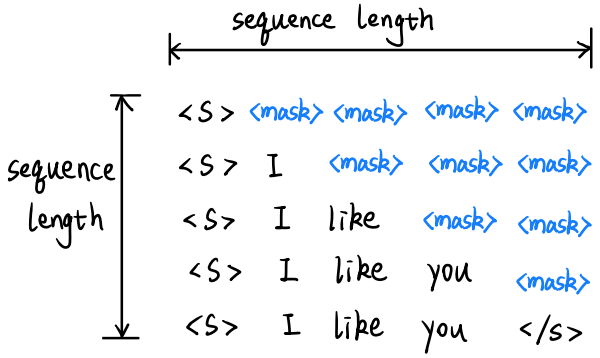
def self-attention = attention(x, x, x)

Mask self-Attention：盖住某些token不让model学习（只看的到历史信息而看不到未来）

① 为对任意句子进行补全

② 为并行学习

③ 一般为上三角矩阵

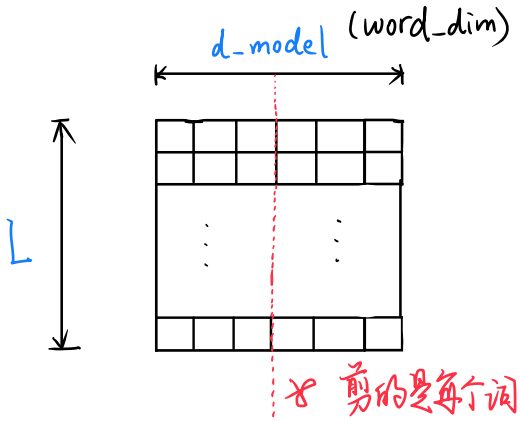


计算注意力分数时，score += mask即为忽略上三角部分的注意力分数

MultiHead-Attention：每个头学习到的不同语义特征

$$MultHead(Q,K,V) = concat(head_1, head_2, \cdots head_h)W^O$$

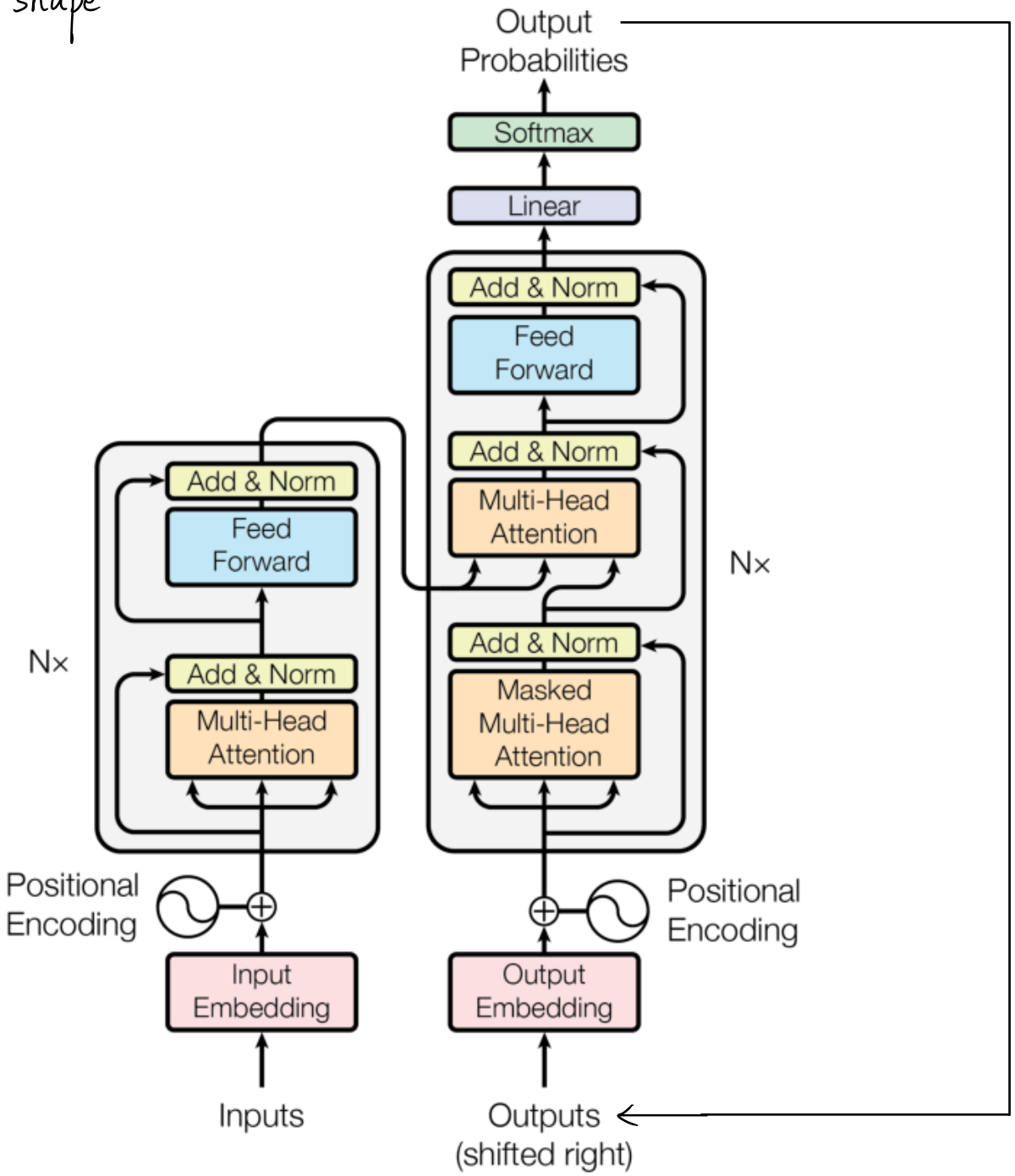
, where $head_n = Attention(QW_n^Q, KW_n^K, VW_n^V)$



(仍能表征词与词之间的关系，但仅停留在保留的k个维度上)

* 剪的是每个词的维度，所以每个head中的注意力分数仍是关于所有词的

code with shape



Embedding 层：

一个batch的句子

Input : (batch_size, seq_len, 1) → 每个词的index值

→ 句子长度(词个数)

↓ search in table(vocab_num, emb_dim) → 每个词→词向量的维度

Output : (batch_size, seq_len, emb_dim)

Postional Encoding : 提供时序信息

Input : (batch_size, seq_len, emb_dim)

$$PE(pos, i) = \begin{cases} \sin\left(\frac{pos}{10000^{\frac{i}{emb\_dim}}}\right), & \text{for } 2 \mid i \\ \cos\left(\frac{pos}{10000^{\frac{i}{emb\_dim}}}\right), & \text{for } 2 \nmid i \end{cases} \quad (i \geq 1)$$

pos 是行, i 是列
(词位) (向量位)

Output = Input + PE(Input, i)

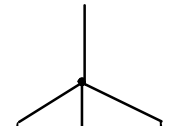
Output : (batch_size, seq_len, dim)

self-Attention 层 (mask/None mask) :

Input : (batch_size, seq_len, dim)

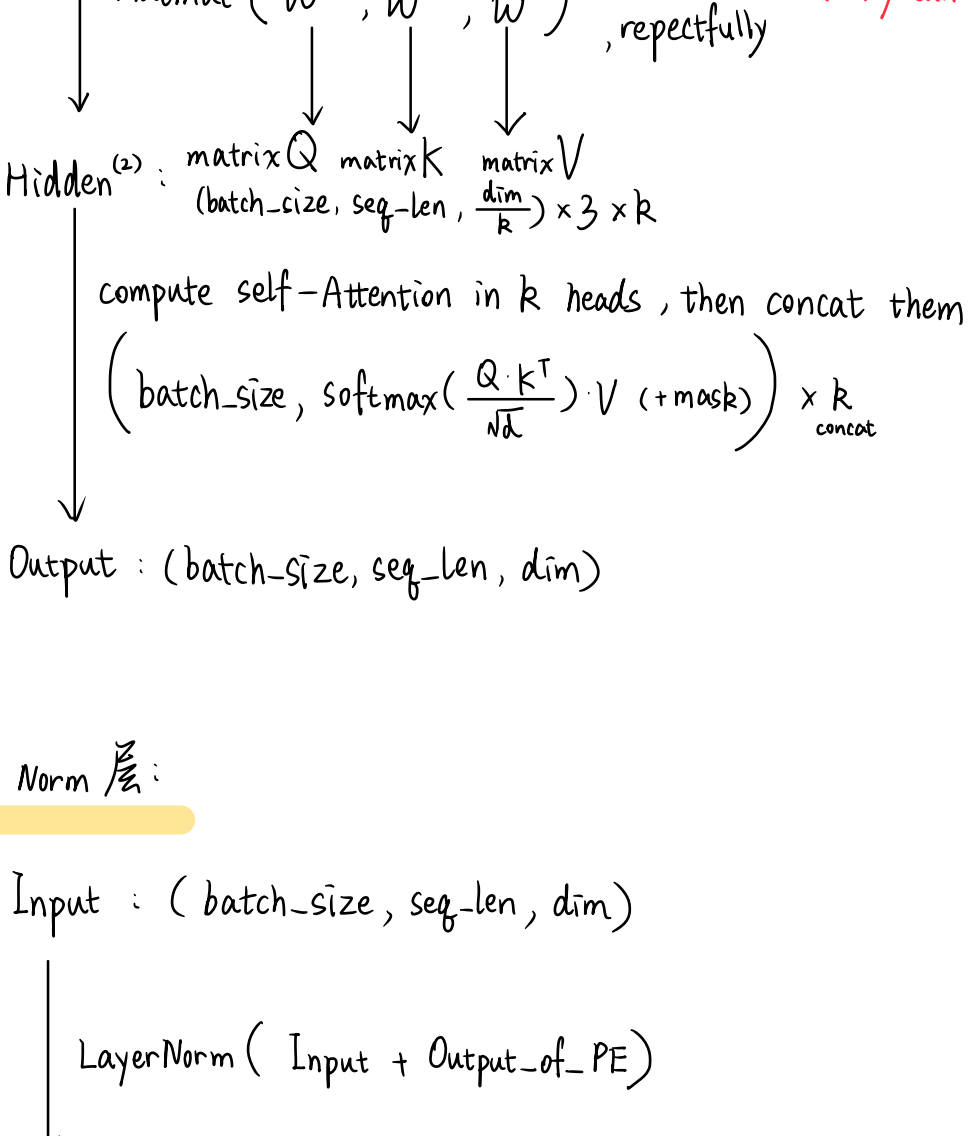
↓ multi-head (suppose split in to k heads) * must obey: k | dim

Hidden$^{(1)}$: (batch_size, seq_len, $\frac{dim}{k}$) × k



* these 3 weights are nn.Parameters
(They can be trained through backpropogate)

matmul (W^Q , W^K , W^V)



Add & Norm 层：

Input : (batch_size, seq_len, dim)

LayerNorm (Input + Output_of_PE)

Output : (batch_size, seq_len, dim)

feed forward 层：

Input : (batch_size, seq_len, dim)

MLP : nn.Sequential [nn.Linear(dim, 4* dim)
nn.Linear(4* dim, dim)]

Output : (batch_size, seq_len, dim)

cross-Attention 层：

Q 为 De coder 的输出，而 K-V 为 Encoder 的输出