Attention is all you need

Abstract

- 提出了一个simple的架构；
- 仅仅依赖于注意力机制，没有卷积和循环神经网络；
- 这里我学习了一下BLUE SCORE；
- 机器翻译结果很好，也有较好的泛化能力。

Conclusion

- 第一个仅依赖注意力的序列转录模型，把所有的循环层替换成了多头注意力机制；
- 在机器翻译任务上好且快；

Introduction

- RNN的缺点：时序信息比较长、难以并行

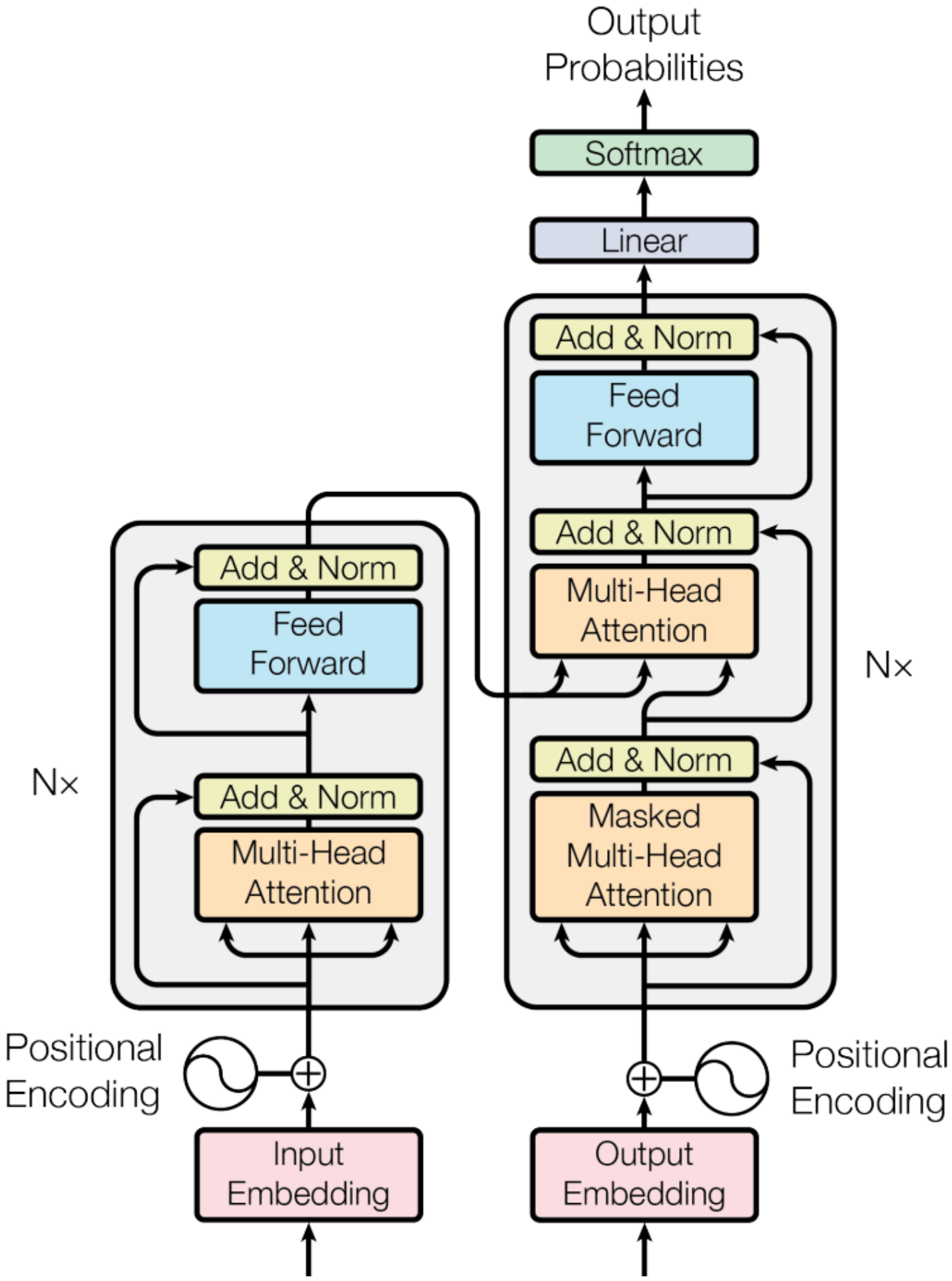
Background

相关工作就是要说明和你类似的工作相关性在哪，区别在哪。

- 有的工作用CNN代替RNN，这就很难考虑到很长的信息，因为CNN只看每个小小的窗口；但是Transformer可以考虑任意两个*像素点*之间的相关性。
- CNN有多个输出的通道，希望Transformer也可以做到这一点，于是使用了多头注意力机制。

Model Architecture

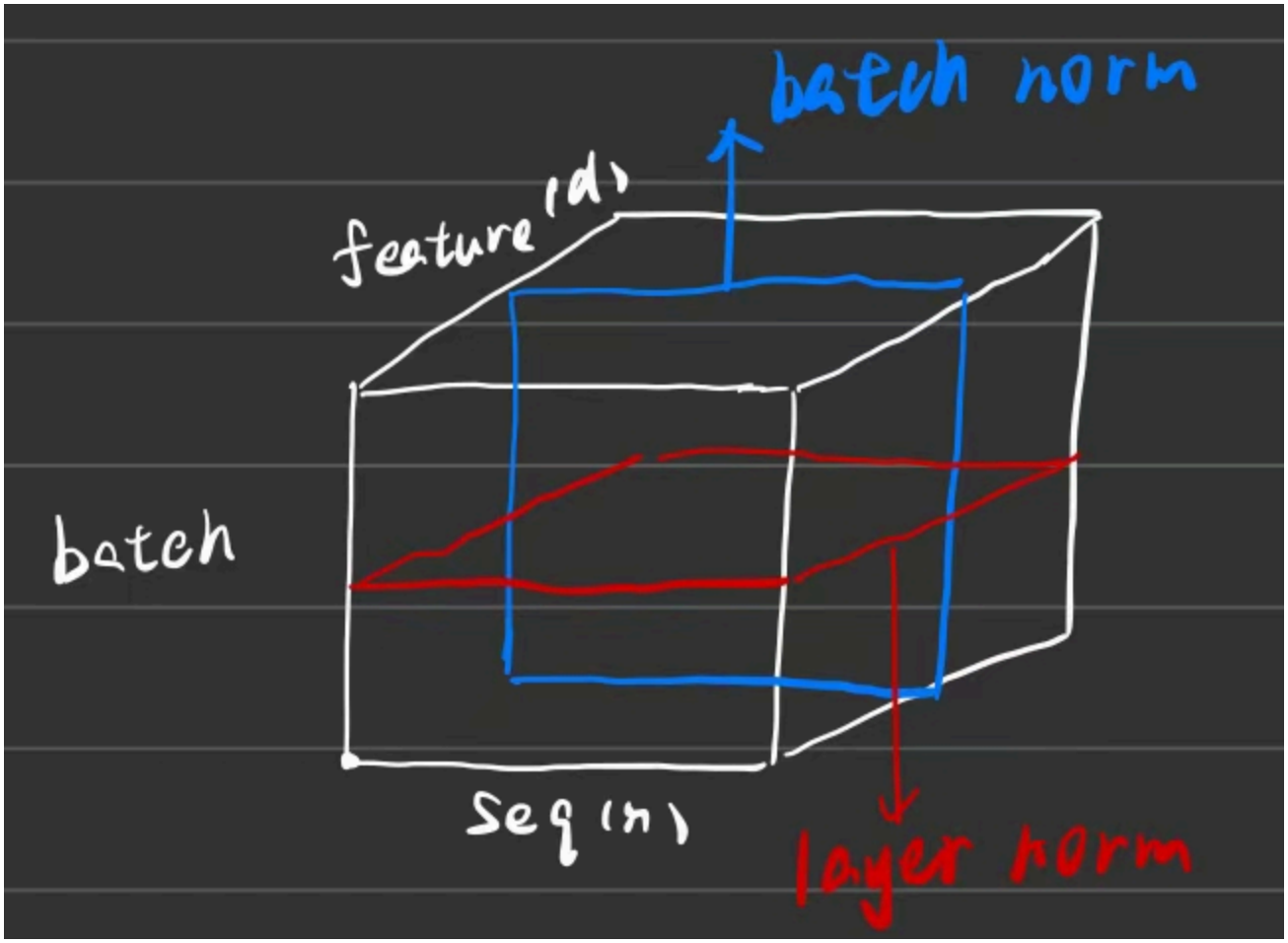
- 编码器的输入为一个句子$(x_1, x_2, ..., x_n)$，输出为它们经过处理后的表示$z = (z_1, z_2, ..., z_n)$，可以理解为是原始的输入转变为机器可以理解的向量。
- 解码器是给输入z之后，得到输出$(y_1, y_2, ..., y_m)$，输出的时候是一个一个输出的（自回归）


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding

Inputs Outputs (shifted right)

Encoder：由6个layer构成，每个layer都有两个sublayer：第一个sublayer是多头注意力网络，第二个就是普通的MLP。并且每个sublayer都做了residual connection和layer norm。写成公式，每一层的输出实际上就是$LayerNorm(x + Sublayer(x))$。同时，每一层的输出的维度都是512。

Layer Norm：把方差变成1，均值变成0。



- 首先，每个batch中有很多样本，每个样本都是一个向量序列，每个向量都有很多特征，因此是三维的；
- batch norm如蓝色所示，是很多样本序列中每个向量在同一个特征维度的切片求均值方差；而layer norm如红色所示，是同一个样本内部求均值方差；
- 与batch norm相比，layer norm只在一个样本内部求和处理，相对来说更加稳定。

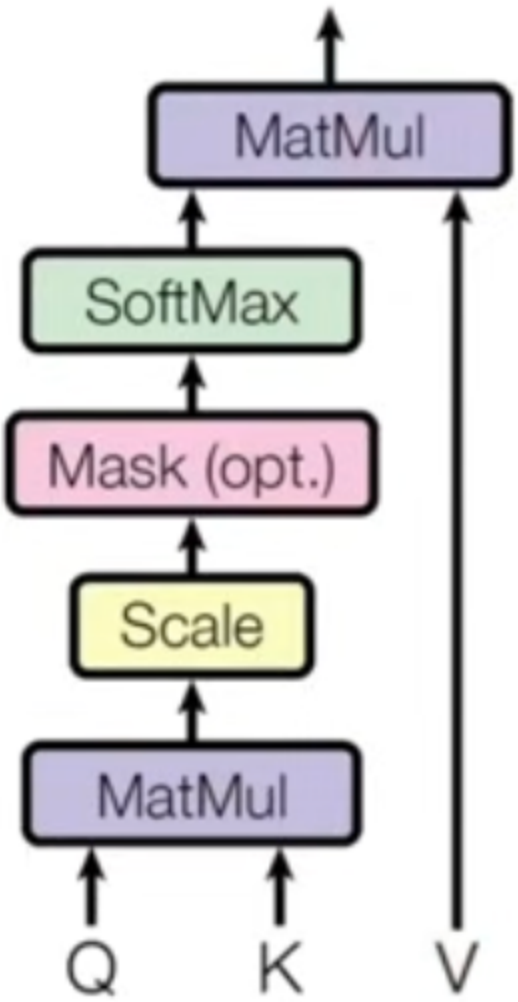
Decoder：由6个layer构成，每个layer都有三个sublayer。和Encoder的操作基本类似，但是第一个sublayer是**掩码**的注意力机制，因为Decoder的输入是上一步自己的输出，因此它无法看到后面的输入。

Attention：注意力函数是query，key-value对的映射，输出是value的求和，但是每个value的权重取决于相应的query和key的相似度。Transformer使用的是“Scaled Dot-product Attention”，就是简单的内

积，但这就要求query和key的维度是一样的，我们设定为d_k。具体公式为：

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

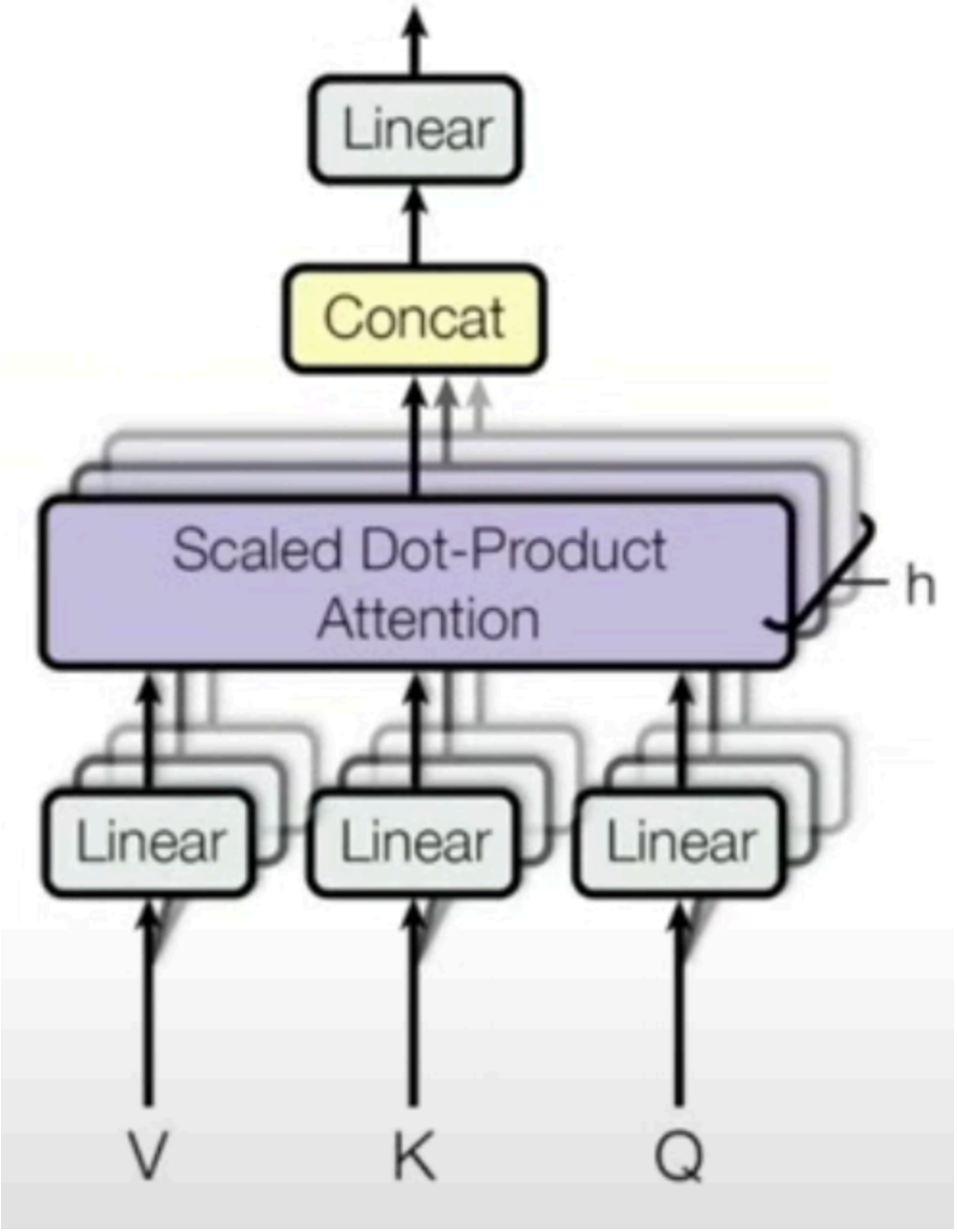
其中，Q、K、V分别是query、key和value的合并。Q是$n \times d_k$的矩阵，K是$m \times d_k$的矩阵，V是$m \times d_v$的矩阵，因此最终的结果是一个$n \times d_v$的矩阵。同时，进行$softmax$也很合乎常理，因为权重之和一般为1。



为什么要除去$\sqrt{d_k}$？ 因为当Q，K的维度过于大的时候，相似度较大的和相似度较小的值会相差非常大，值会更向两端靠，经过softmax就会靠拢于0和1，这就会导致梯度比较小，训练跑不动。
为什么要Mask，怎么实现？ Decoder中的每个q_t按理说只能考虑$k_1, k_2, ..., k_{t-1}$，但是实际上它也会和后面的k进行计算，只不过算出来的值我们赋给它一个很大的负数，那么经过softmax后权重就会变成0。

Multi-Head Attention：
我们首先把Q，K和V投影到低维的空间h次，做h次的注意力函数的计算，得到h个输出，把h个输出合并

到一起，再线性投影回来，得到结果。**多头注意力是为了匹配不同的相似和计算模式。**

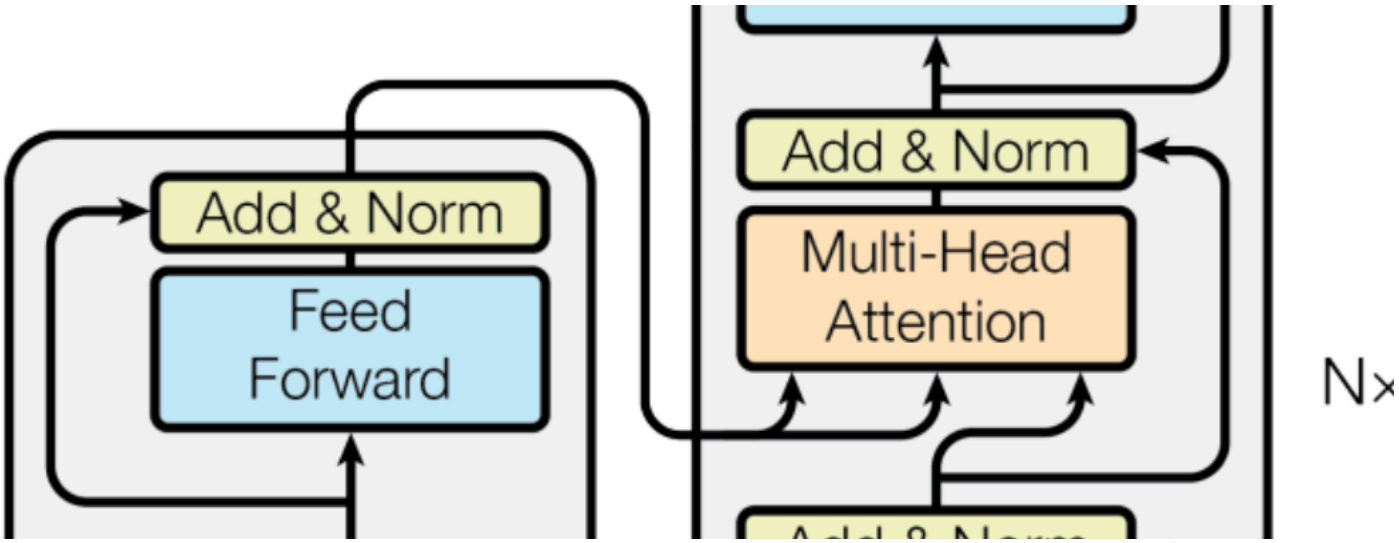


计算的公式如下：

$$MultiHead(Q,K,V) = Concat(head_1, head_2, ..., head_h)W^O$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Transformer中h的值为8。

编码器和解码器之间传递信息：



这里多头注意力的Q来自Decoder上一步的输出，K，V则来自于Encoder的输出（这里不是自注意力机制）。

Feed-Forward：对序列的每个位置单独应用，但是神经网络都是一样的。

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

宏观上来理解：Attention部分已经抽取了序列中的讯息，后面的MLP只需要按照我们的需求投影到相应的空间，所以可以单独做MLP。

Embeddings&softmax

- Embedding：把token映射到d维的向量，解码器、编码器都需要embedding，它们权重一样（方便训练）。
- 训练出来的embedding需要乘上$\sqrt{d_{model}}$。

  Positional Encoding：Attention并没有考虑时序信息。

  用一个和embedding后大小相同但是表示位置的向量与embedding相加。

Why Attention

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

第一列是矩阵计算的复杂度，第二列表示self-attention是并行计算，第三列表示不管距离多远，都可以直接计算query和key的相关性。

Training

优化器用的是Adam，学习率结合了衰减和warm-up

BLEU SCORE

BLEU（Bilingual Evaluation Understudy）是机器翻译和自然语言处理领域中常用的自动评估指标，主要用于评估机器生成文本与人类参考文本之间的**相似性**。它的评分系统基于词汇的**重叠情况**，评估生成的翻译与一个或多个参考翻译之间的匹配度。

计算方式如下：

- n-gram精确度：通过计算机器生成文本和参考文本之间的n-gram（n个连续词）的匹配度。n-gram可以是单词（1-gram）、词对（2-gram）等。**较高的n值通常可以捕捉到更长的短语匹配。**
- 惩罚机制（Brevity Penalty, BP）：为了避免生成过短的文本得分过高，BLEU引入了一种惩罚机制，**如果生成的文本比参考文本短得太多，得分会相应降低。**

BLEU的值介于0和1之间，1表示生成的文本与参考文本完全相同，具体公式为：

$$BLEU = BP \times exp(\sum_{n=1}^{N} w_n \log p_n)$$

- BP 是惩罚系数（当生成文本短于参考文本时会惩罚得分）
- p_n是n-gram精确度，即生成文本与参考文本在n-gram上的匹配情况
- w_n是权重（通常设置为均匀分布）