Attention Is All You Need

Abstract

They propose a new type of simple network architecture, the Transformer, based solely on attention mechanisms,dispensing with recurrence and convolutions entirely.

Conclusion

The Transformer is significant faster and better than other models. They also want to make generation less sequential.

Introduction

传统的RNN网络因为时序问题无法实现并行运算，并且保存历史消息需要大量储存开销。本文的Transformer完全不使用RNN，只使用attention，可以并行运算。

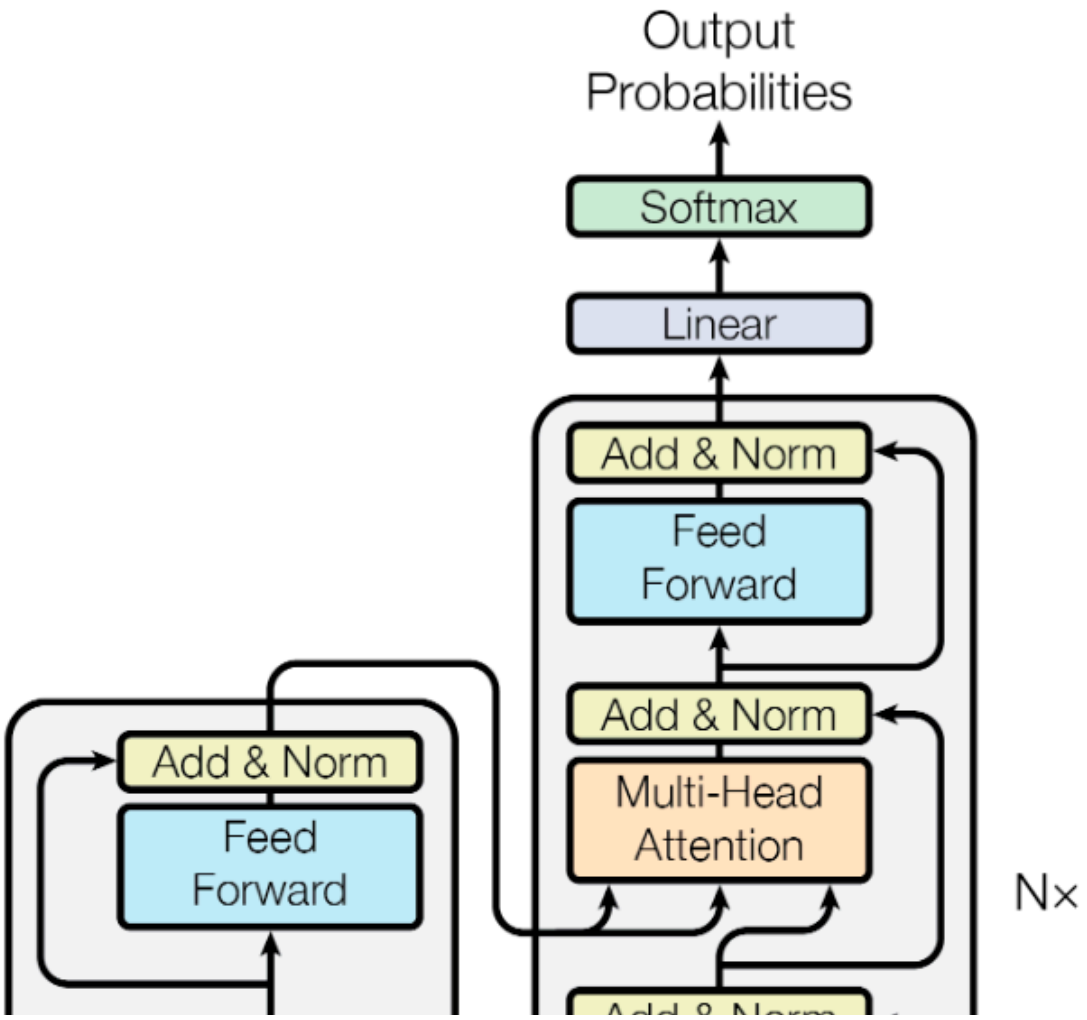
Background

卷积神经网络无法很好作用于长序列消息，因为他需要很深的网络；然而卷积可以有多输出通道，一个输出通道可以识别一种模式，值得学习，这也是多头（muti-head）的来由。

Model Architeture

Encoder：原始文本输入（x1，x2....xn)经过编码后为机器可以理解的（z1,z2.....zn)

Decoder将上述（z1,z2.....zn)解码为（y1，y2...ym），n和m不一定相等，且yi是一个一个输出的，过去的输出也是新的输入，此类称为自回归模型。



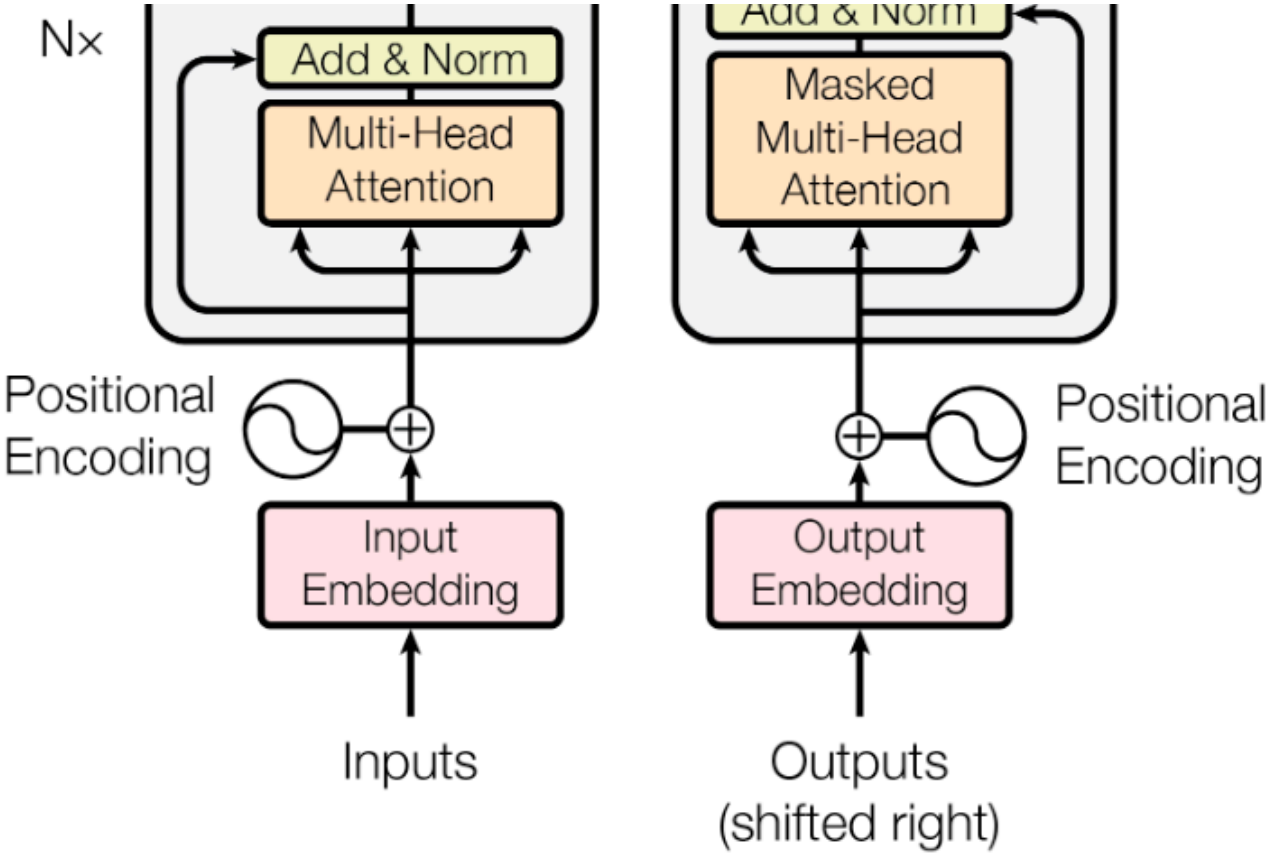


Figure 1: The Transformer - model architecture.

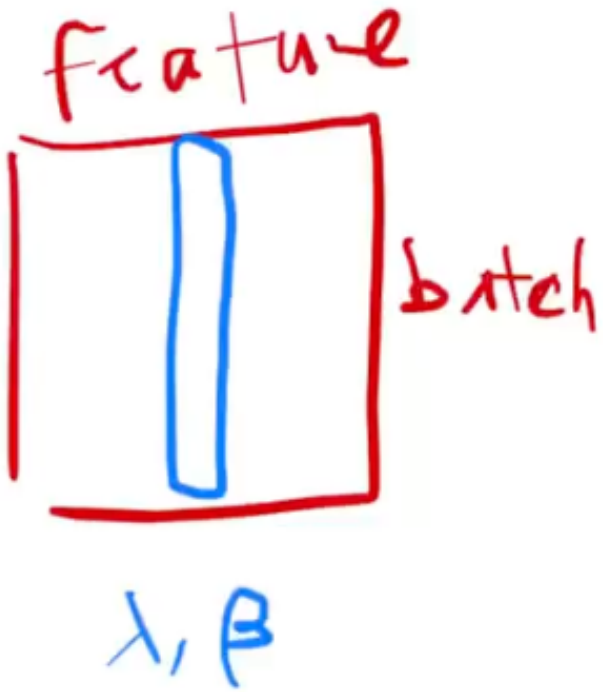
Encoder：

使用了6个完全一样的layer，每个layer都有2个sublayer，第一个是multi-head，第二个是MLP。每个子层输出为

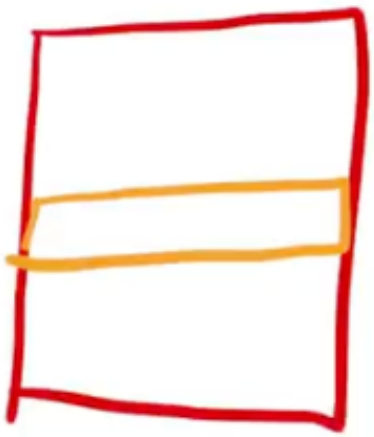
$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$$

固定输出维度为512，懒得调投影了，简单。

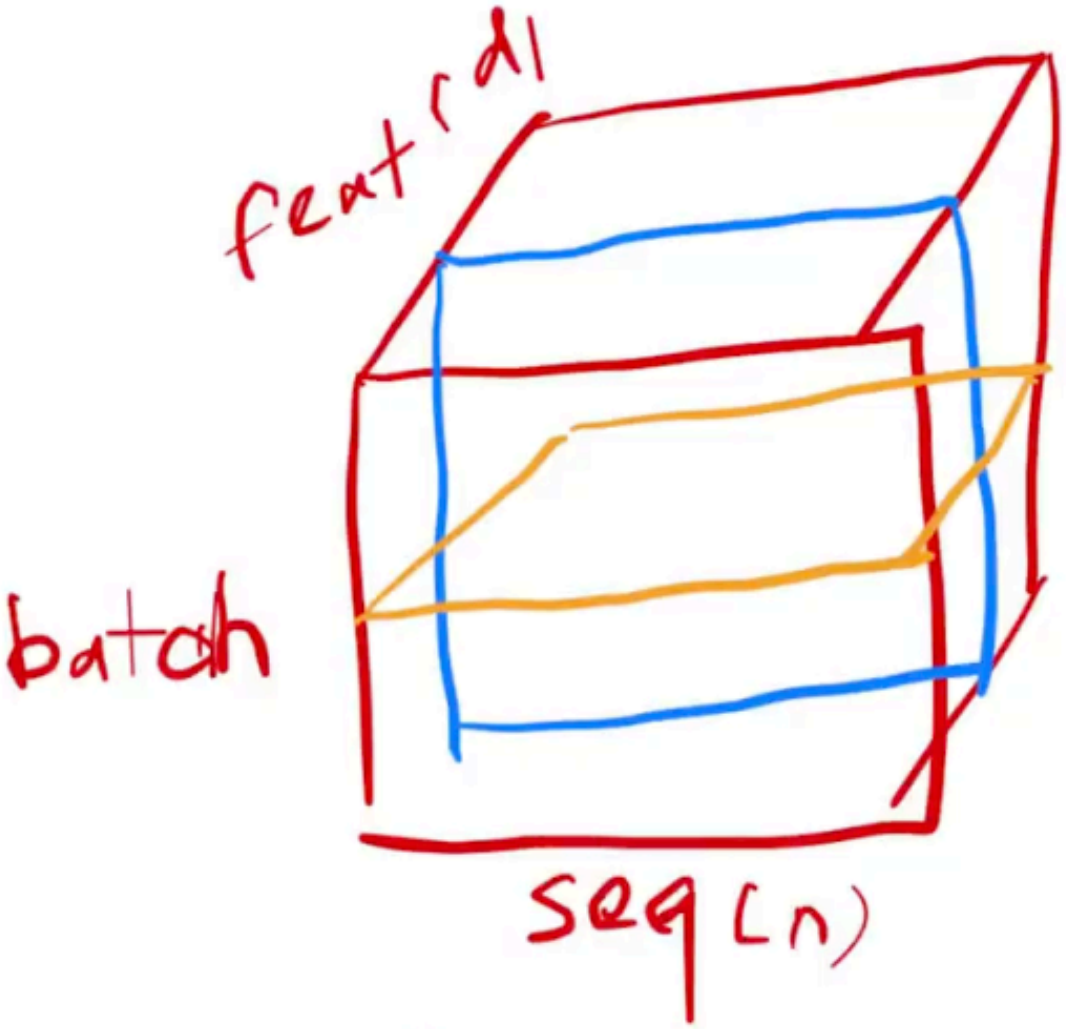
Batchnorm：在一小批数据中将特征（列）均值变为0，方差变为1（向量剪均值，再除以方差）



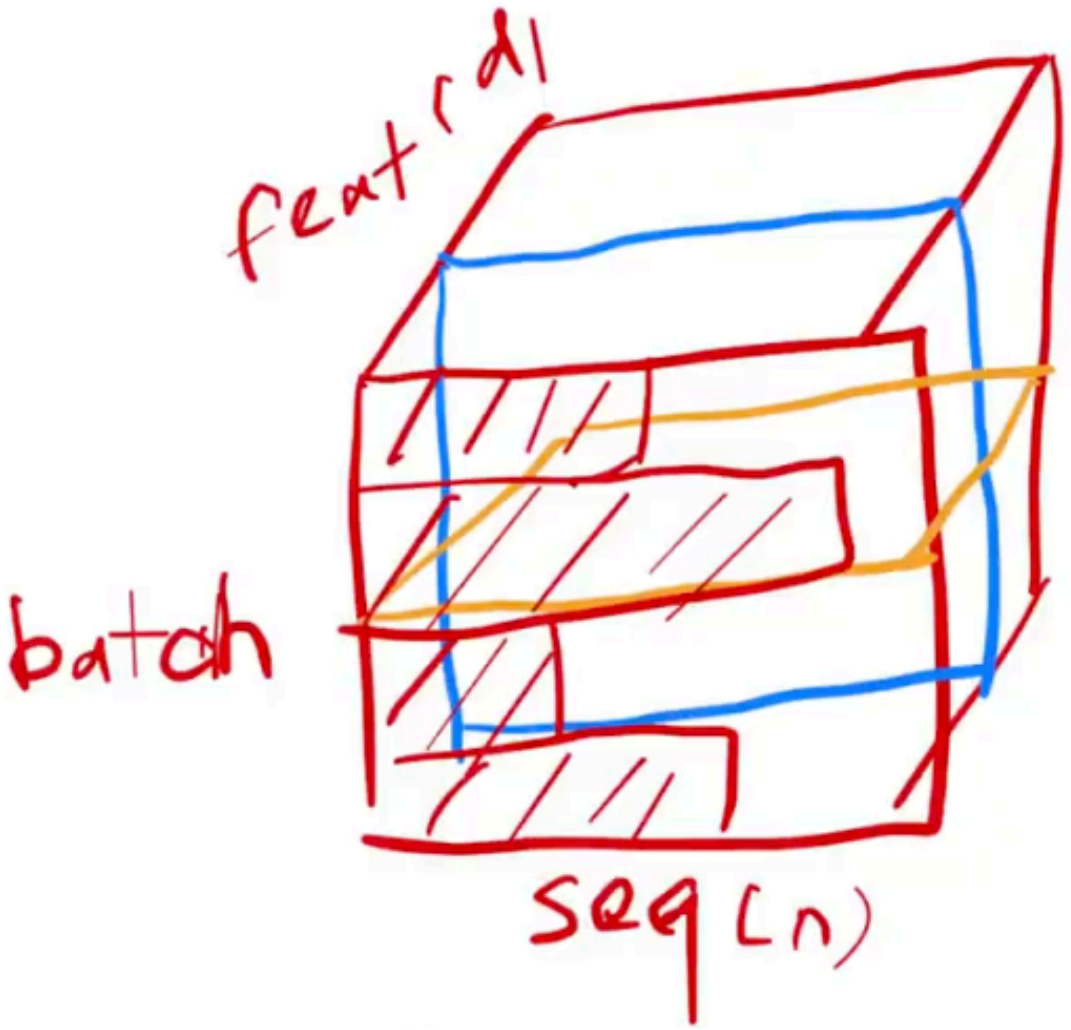
Layernorm：和batchnorm几乎一样，但是是对每个样本（行）处理。



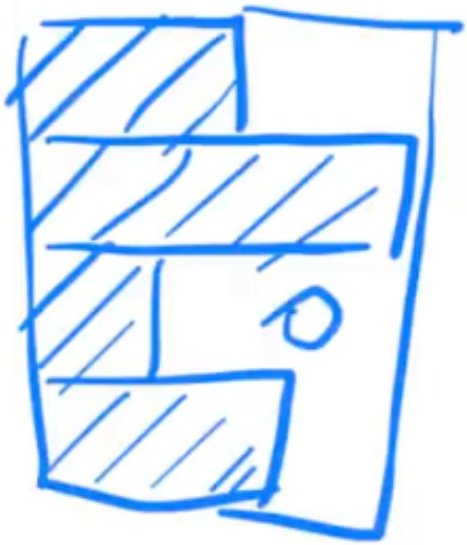
一般数据输入为三维


feat (d)
batch
seq (n)

且多使用layernorm，原因是数据通常变长



举例，batch均值是算阴影面积，



而且如果新数据出了这货，

那么方差和均值浮动就很大

而layernorm就没有这个问题

Decoder：

解码器一样由6个同样的层（每个包含3个子层），第三个子层使用muti-head架构。解码用的是自回归，同时为了保证不看到t后面的输入，使用掩码mask注意力机制。

Attention

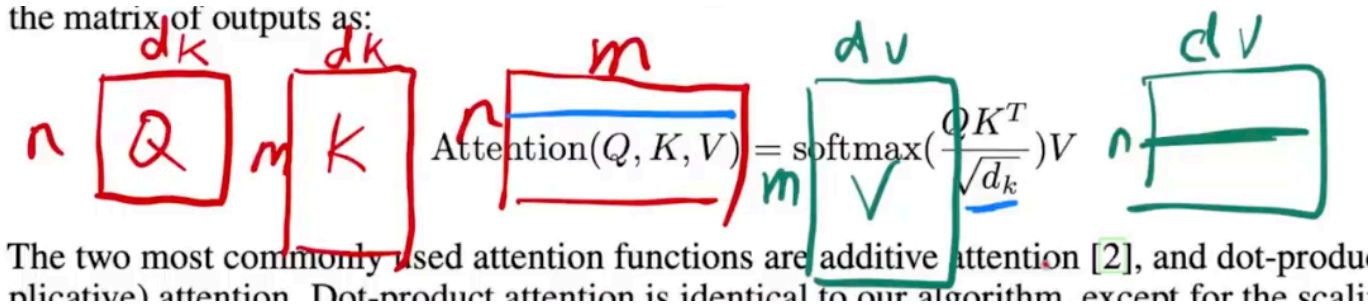
将query和key、value对映射成输出的函数，ouput是value的加权和。key谁离query近，他对应的value就有更大权值。

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

Q和K维度是dk，V维度是dv，通过对Q和K做内积（用距离衡量相似度），再除以$\sqrt{d_k}$，

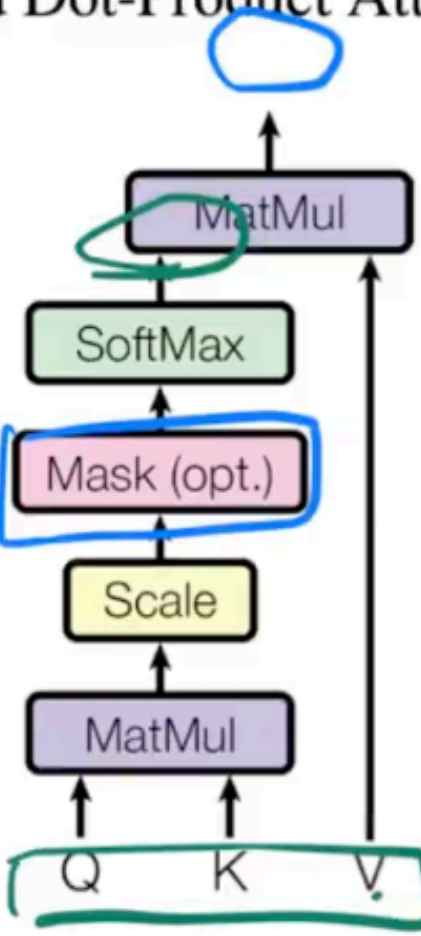
（如果向量很长的话，相对差距就会变大，梯度会变小，训不动，所以要除个数）

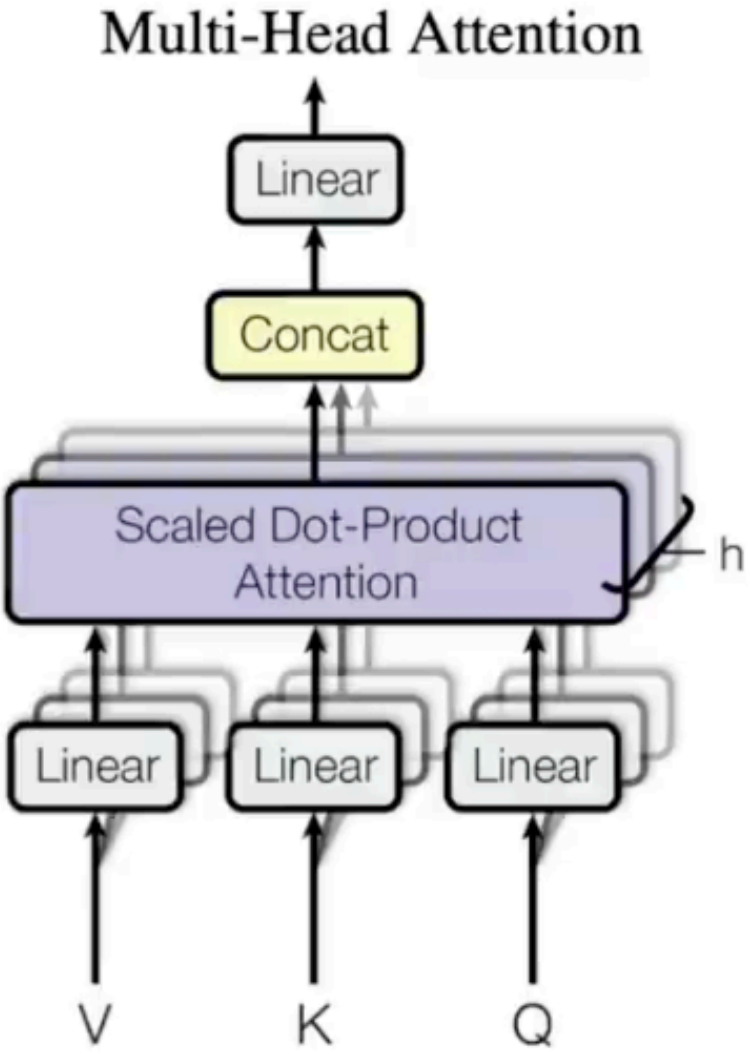
丢到softmax就可以得到value的权重，再乘V则可得到output



对于t时刻的Qt，应该关注的是K1、K2...Kt-1、Kt，但是计算时实际上会看到所有K，所以把后面的K换成很大的负数，至此在softmax中由指数作用趋近于0，忽略其影响。

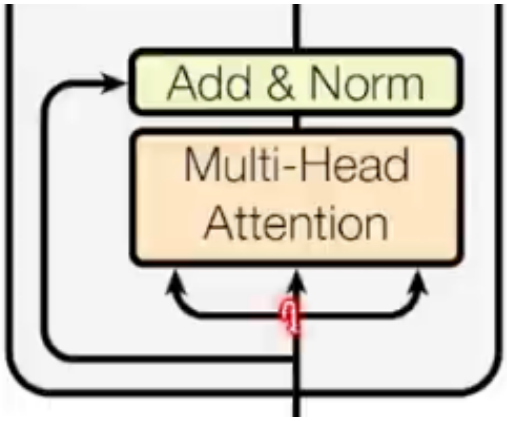
Scaled Dot-Product Attention



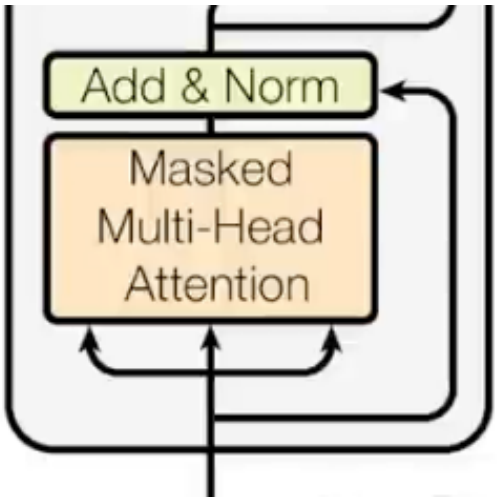


多头就是模拟多通道，用h次机会学习不同的模式

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

编码器的注意力层： 

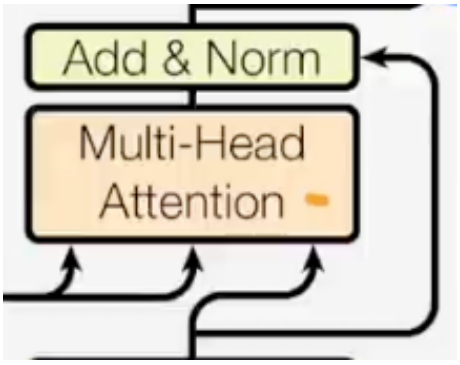
其实是同一个输入复制了三遍变成了Q、K、V，就是自己本身，所以叫自注意力。输出其实就是输入本身的加权和（通过相似度)。多头就是学习了h次距离空间。

解码器的第一个自注意力层： 

和上面相似，只是长度可能变成了m，还有个mask（后面的权重设为0)

解码器的第二个自注意力层：

输入Key和Value来自编码器输出，Query来自解码器上一个自注意力层输出。

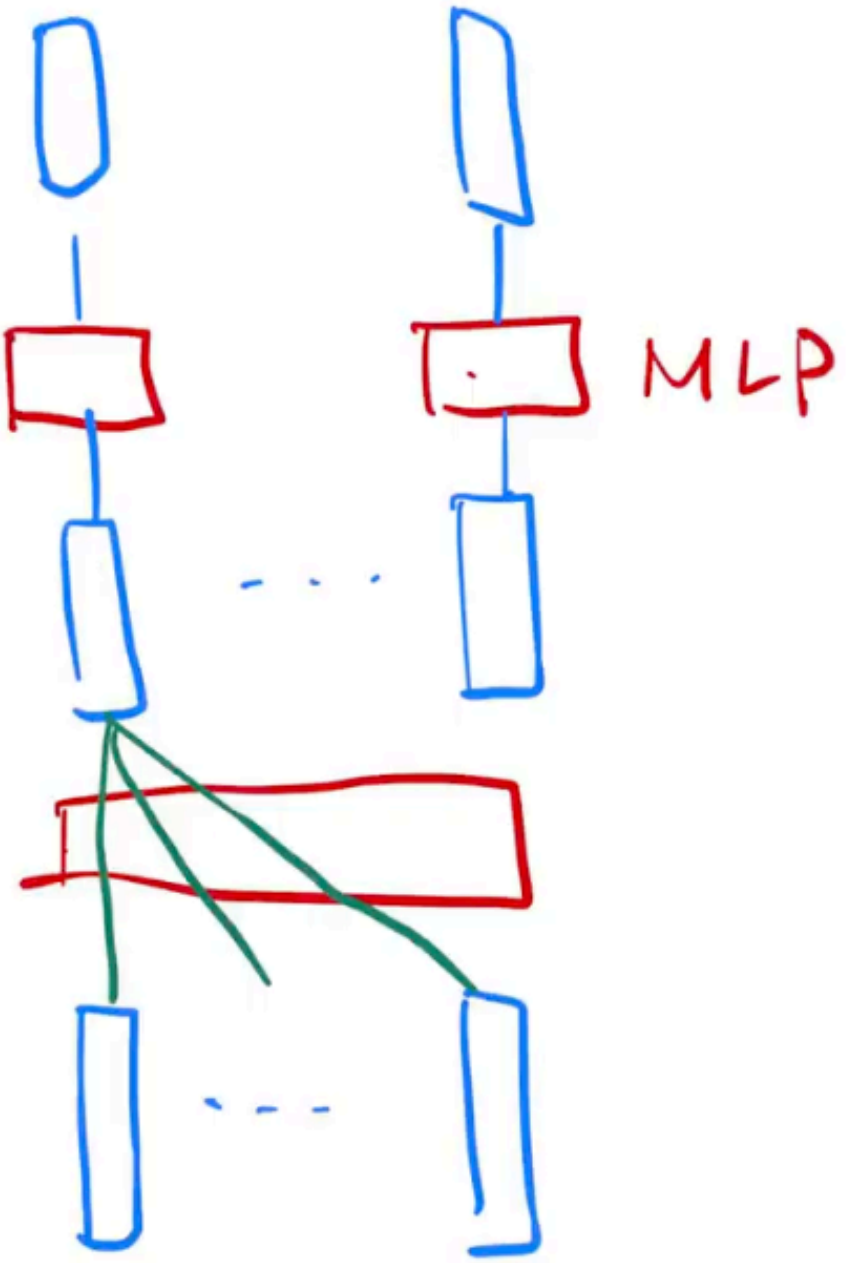


编码器和解码器的互动其实就是：解码器的输入在挑编码器的输出，看看哪个长得像，cross attention

Feed-Forward

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

说白了就是单隐藏层MLP，把两个线性层放在一起，用relu增强非线性。



Attention就是全局地抽取序列中的感兴趣信息并且汇聚，用MLP形成语义空间的向量

Embeddings and Softmax

乘√d是为了增大权重

Positional Encoding

为了弥补时序信息，增加位置编码来表示。