# Attention

## Encoder-Decoder结构



图 1.1E-D结构

一般用于句子对<Source,Target>，我们的目标是给定输入句子Source，期待通过Encoder-Decoder框架来生成目标句子Target。Source和Target可以是同一种语言，也可以是两种不同的语言。而Source和Target分别由各自的单词序列构成：

计算公式：

编码器输出的中间编码

解码器输出的每一维

## Soft Attention

### E-D结构的局限性

由的计算方式可以看出，句子Source中任意单词对生成某个目标单词来说影响力都是相同的，都是根据相同的C计算出来的。而实际上，不同位置的对的影响都是不同的。

比如“Tom chase Jerry”—>“汤姆 追逐 杰瑞”，对的影响明显要比和大。

### Attention机制的应用与计算方式

计算每个的时候，都考虑到Source中每个单词的权重(注意力分配概率)信息，这时每个计算时的C就不再相同了，如图1.2所示。

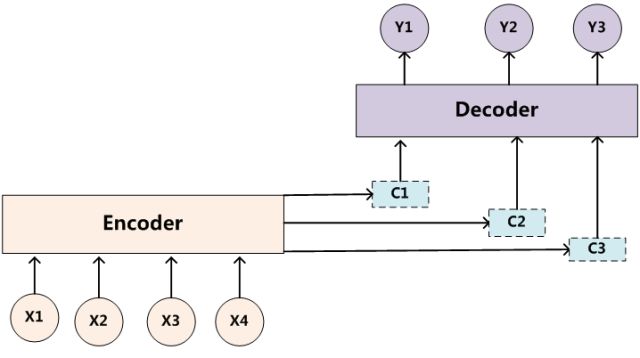


图 1.2加入Attention之后的E-D结构

计算公式：

代表相对与的权重，或者叫注意力分配系数

函数h是对输入的某种转换函数，比如Encoder用的RNN模型,h函数的结果往往是某个时刻输入后隐层节点的状态值。

在解码器中其实就是根据C和计算出来的状态值。

这里，所有相对于的权重向量的计算方式如下：

### Attention机制的本质思想

# Self Attention

## Self与Soft的差异

### 思想上的差异

### 应用上的差异

# Transformer模型结构

## Scaled Dot-Product Attention

一种特殊的Attention结构，输入三个矩阵：Q(query)、K(key)、V(value)。



图 2.1 Scaled Dot-Product Attention

**数学公式：**

计算公式：

最终得到一个的矩阵。

其中：Scale层是正规化，Mask层实现一步的除法操作

## Multi-HeadAttention

并行多个Scaled Dot-Product Attention模块，然后将结果融合，也叫做自注意力(self-Attention)模型。

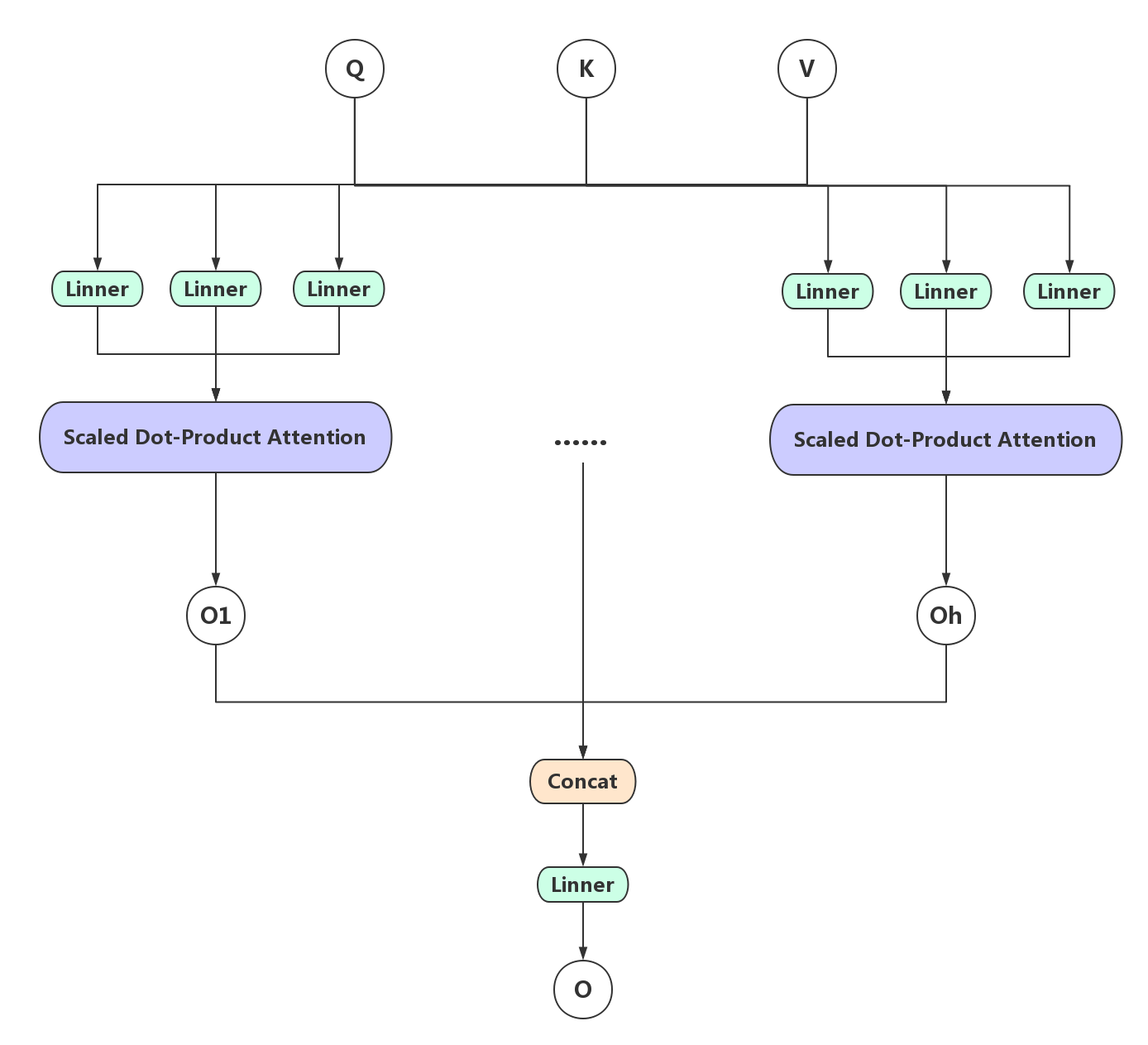


图 2.2Multi-HeadAttention

**各层说明：**

Linner: 线性映射层，即Dense层

Concat: 矩阵融合操作，将若干个矩阵和在一起，Concat(n\*m, n\*m)=n\*2m，最后再使用一个Dense层将维度还原为n\*m

## Transformer结构

网络模型结构如图1.3所示。



图 2.3Transformer结构

Add: 矩阵加法 Norm:正规化

Positional Encoding：位置编码，引入上下文信息，可以参见ELMO和BERT

其中，左侧的Multi-HeadAttention+Feed Forward构成一个Encoder模块，右侧的两个Multi-HeadAttention+Feed Forward构成一个Decoder模型。在论文中，N取值为6，因此Transformer结构的结构也可以表示成图1.4所示。

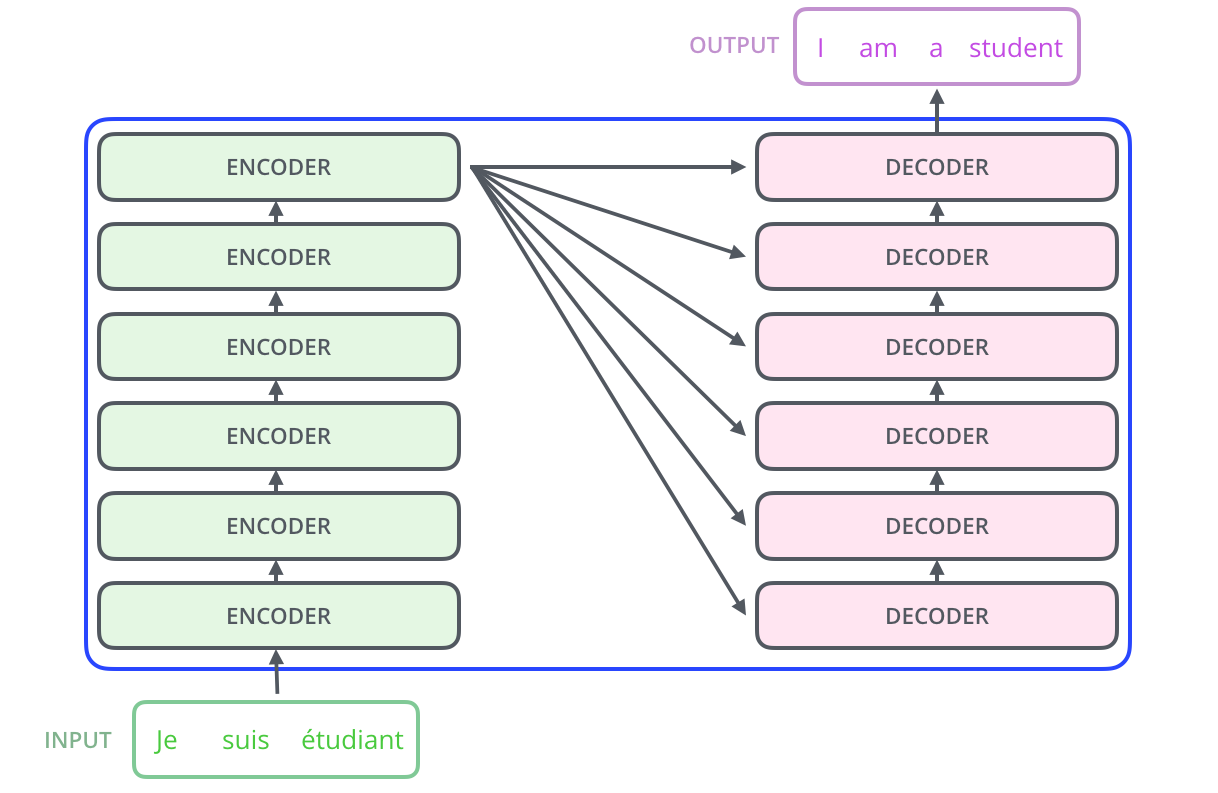


图 2.4Transformer的另一种表示

由上图可以看出，Encoder和Decoder都是一层层串联的，最后一层Encoder的结果会传给所有Decoder作为部分输入。