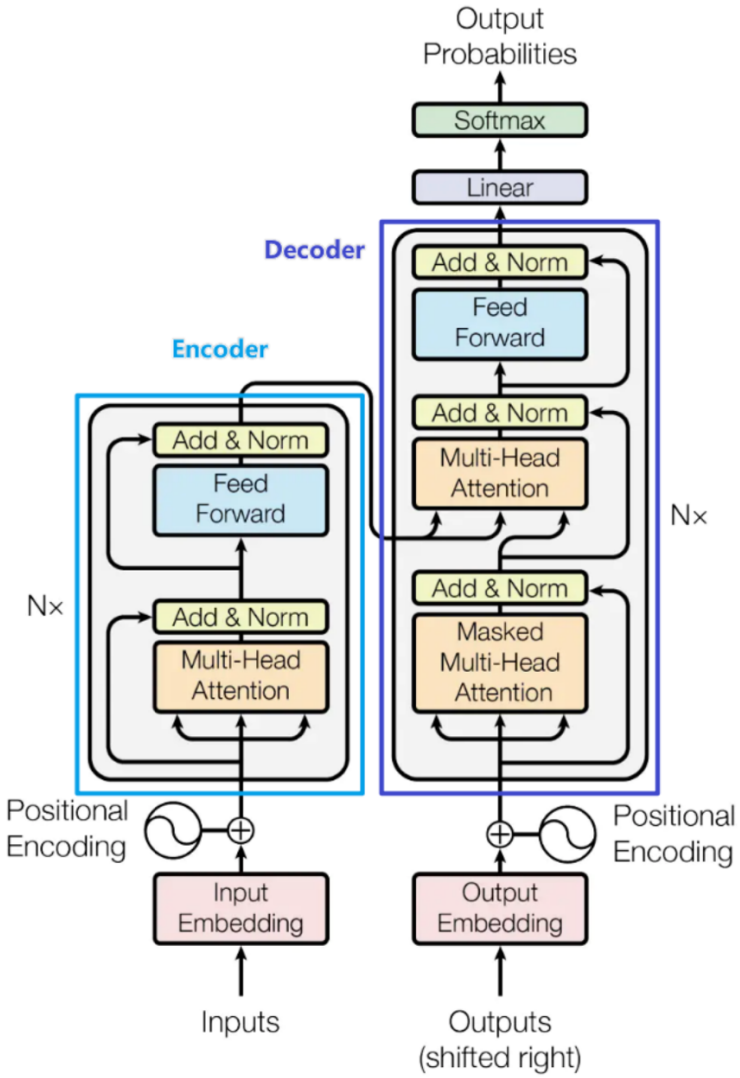
Transformer

- **Transformer 模型由 N 个 Encoder 层和 N 个 Decoder 层组合而成。**



1. Encoder 层

- 先来看 Encoder 部分，从最下方的$\boldsymbol{Inputs}$开始，这里就输入了一个序列$\boldsymbol{a_1, a_2, \ldots, a_N}$（比如在 NLP 中，输入了一个句子），然后获得每一项$\boldsymbol{a_i}$的 Embedding（嵌入），这里的 Embedding 其实是$\boldsymbol{a_i}$的特征向量。
- 接着利用前面提到的位置编码 Position Encoding 方式对序列的各个位置进行编码，并把位置编码向量与序列的特征向量 Embedding 直接相加，得到下一层（Multi-Head Attention)的输入$\boldsymbol{X'}$。
- 可以看到 Multi-Head Attention 模块的$\boldsymbol{Q}$、$\boldsymbol{K}$、$\boldsymbol{V}$来自于同一个输入$\boldsymbol{X'}$（模块输入的三个箭头来源相同），所以它是一个 Multi-Head self-Attention，多头自注意力机制模块。Transformer 的 Attention Score 的计算方法采用的是缩放点积相关性:

$$\alpha_{i,j} = \frac{\left(q^i, k^j\right)}{\sqrt{d}}$$

- 经过多头自注意力机制模块的输出下一步经残差模块和 Normalization 模块 （Add & Norm），这可以缓解深层次网络梯度弥散、网络退化等问题。
- 经残差 + 正则化模块后，将进入**位置前馈网络** (Position-Wise Feed-Forward Network，FFN) 层，同样该层的输出也需要再经过残差块 + 正则化模块。这里需要特别补充的是，位置前馈网络层对于 Transformer 实现良好性能至关重要。

  Position-Wise Feed-Forward Netword 是一个**全连接网络**，包含两个线性变换和一个非线性函数 (ReLU)

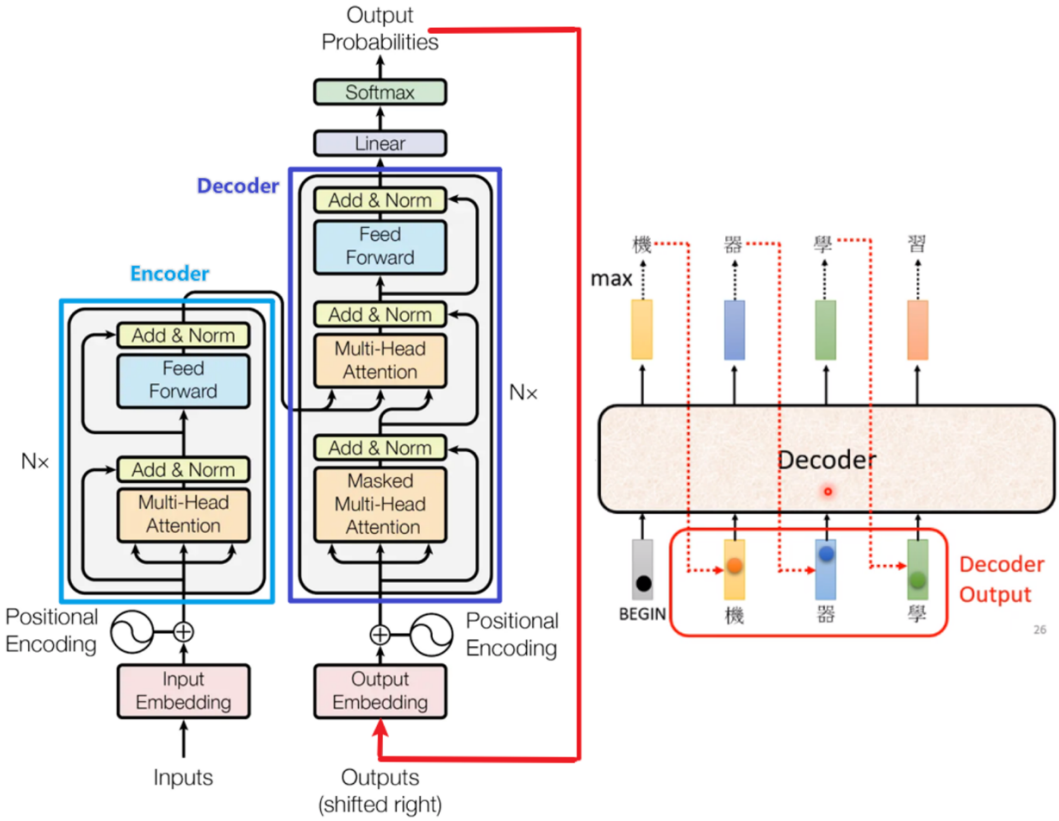
  $$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

  这个线性变换在不同的位置都是一样的，并且在不同的层之间使用不同的参数。

  这个 Position-Wise 可以看作**两个核大小为 1x1 的一维卷积层**。

  研究者观察到**简单地堆叠 Self-Attention 模块会导致等级崩溃问题以及 token 均匀性归纳偏差，而前馈层是缓解此问题的重要构建块之一**。
- 可以看到 Encoder 部分都是关注上下文信息的，而 Self-Attention 是支持并行化的，因此 Encoder 可以并行处理输入的序列，然后输出一整个序列的 Embedding。

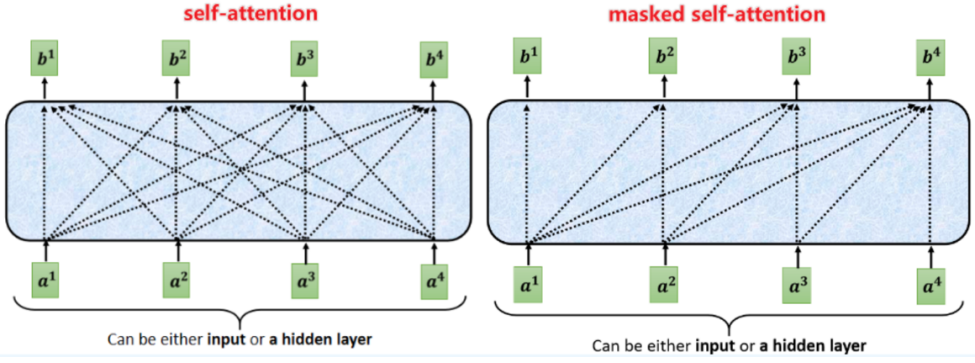
2. Decoder 层



- 从原框架的右下角开始看（上图 Decoder 的最下面），这里输入是一个$Outputs$，Decoder 的第一个输入来自于目标序列，在 NLP 中，常常会在输入句子的开头加上一个表示开始的字符（比如$[CLSI/BEGIN]$等等），这个字符经过 Embedding 后就是 Decoder 的第一个输入。

  结合右图，可以发现 Decoder 的机制类似于 RNN，是串联的，前一个输出会作为后一项的输入，所以在原始的 Transformer 框架中，加了一个红色箭头表示其输出又作为 Decoder 的输入，对$Outputs$进行一个解释。

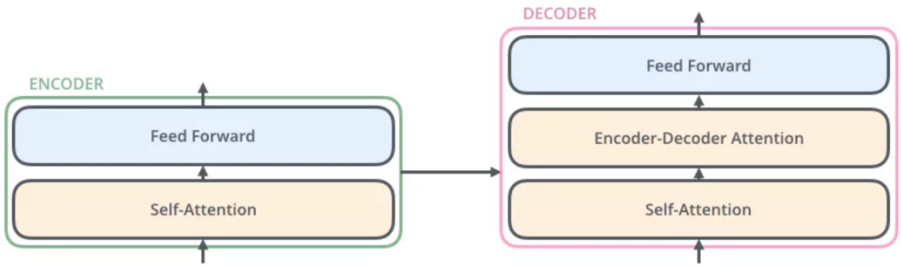
- Decoder 的输入进入 Masked Multi-Head Attention 模块，可以看到该模块三个箭头的输入均来自于同一个 Embedding，所以是一个 Self-Attention 模块，额外加了 Mask 机制，Self-Attention 和 Masked Self-Attention 的差别如下图所示，当前预测位置只能获取已经输出了的信息，即b_1只能获取a_1的信息，b_2只能获取a_2的信息……以此类推。



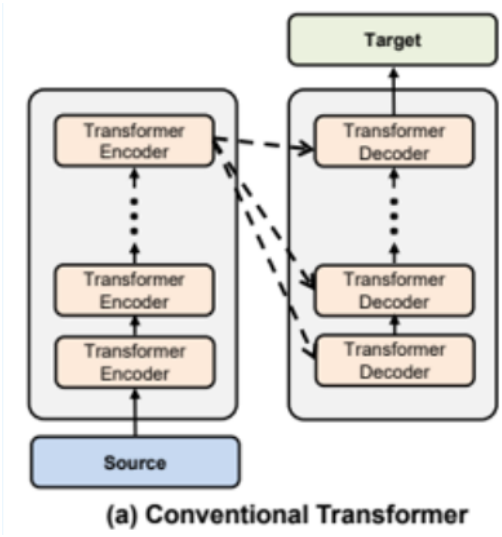
- 接下来经残差块 + 正则化模块后，输入另外一个 Multi-Head Attention 模块，注意该模块的输入箭头，**有两个箭头来自于 Encoder 的输出，一个箭头来自于 Decoder 上一层的输出**。该 Attention 模块为 Encoder-Decoder 的 Cross-Attention 模块。
- 经过该 Cross-Attention 模块后，再经过与 Encoder 类似的残差块 + 正则化模块，FFN 模块 + 残差块 + 正则化模块，最后接上与下游任务相关的线性层（比如分类、线性回归等），逐步获取序列的输出。

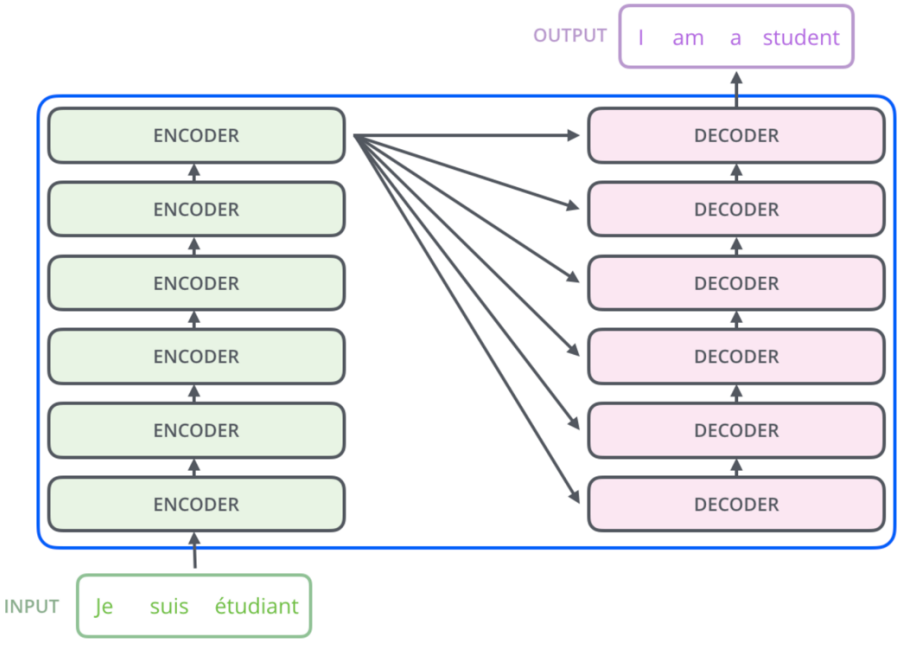
3. Encoder-Decoder 交互 Cross-Attention

- 与编码器对应，如下图，解码器在编码器的 Self-Attention 和 FFNN 中间插入了一个 Encoder-Decoder Attention 层，这个层帮助解码器聚焦于输入序列最相关的部分（类似于 Seq2Seq 模型中的 Attention）



- Encoder-Decoder 之间交互存在多种形式，传统 Transformer 中 Encoder 和 Decoder 交互的方式是：第 N 个 Encoder 层最后的输出与每一层 Decoder 进行交互，如下图所示:





- 在详解 Attention 机制时，提到$\boldsymbol{Q}$、$\boldsymbol{K}$、$\boldsymbol{V}$三项可以来自不同矩阵，选择不同的$\boldsymbol{Q}$、$\boldsymbol{K}$、$\boldsymbol{V}$就形成了不同的 Attention 变形，比如当$\boldsymbol{Q}=\boldsymbol{K}=\boldsymbol{V}$时，就是 Self-Attention 机制，那么这里 Encoder-Decoder 交互的 Cross-Attention 实际上就是将 Decoder 内该模块上一层的输出作为$\boldsymbol{Q}$，而 Encoder 最后一层的输出（一整个序列的 Embedding）作为$\boldsymbol{K}$和$\boldsymbol{V}$（$\boldsymbol{K}=\boldsymbol{V}$）。可以直观理解为，哪个 Key 可以更好地回答这个 Query。