Transformer

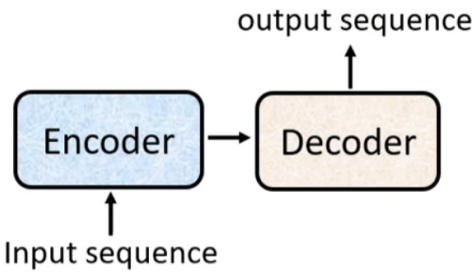
Seq2seq引入

- 运用场景：输出不知道具体长度（机器自己决定）
    - e.g.：输入声音信号，输出中文字，长度没有绝对的关系 / 反过来的语音合成

很多NLP任务可以看作QA任务，QA任务又可以利用Seq2seq解决，但各自化模型效果更好。一些运用：句法分析（把句法看作另一种语言，把翻译套用语法分析）、多标签分类（自己决定class）、目标识别

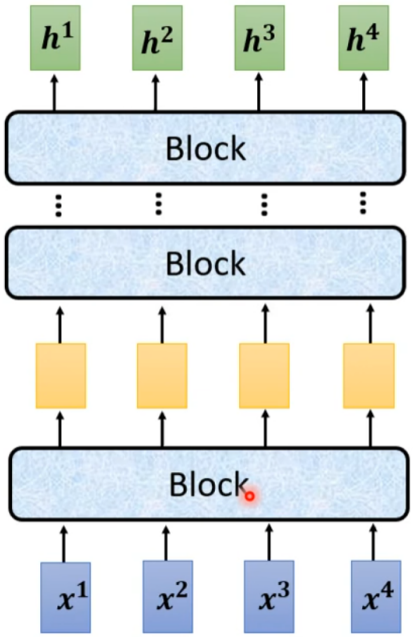
Transformer

Seq2seq模型主要是由编码器和解码器两部分组成



Encoder

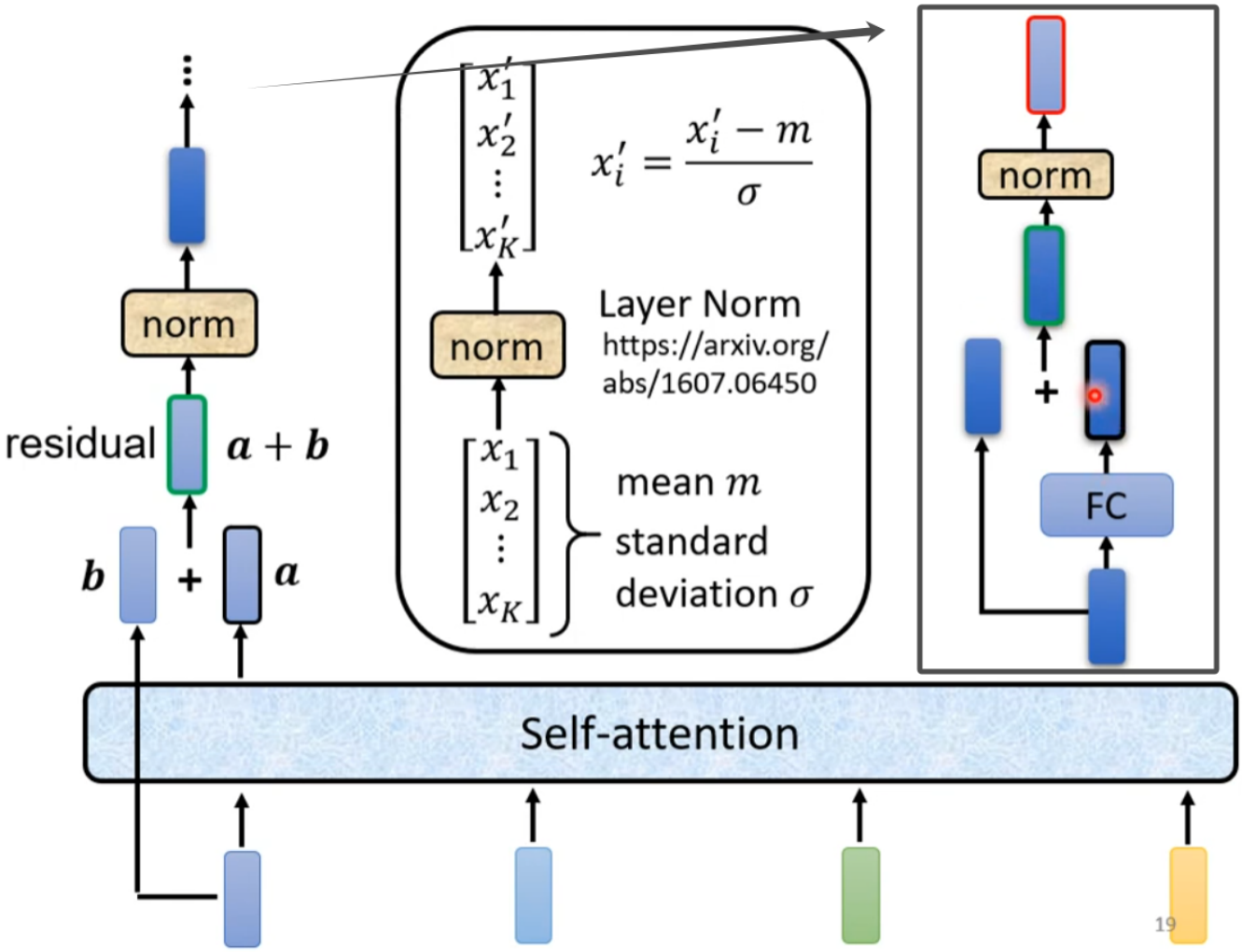
Seq2seq模型的编码器能够实现输入一个序列的向量然后输出一个序列的向量，这个可以通过自注意力，RNN，CNN实现



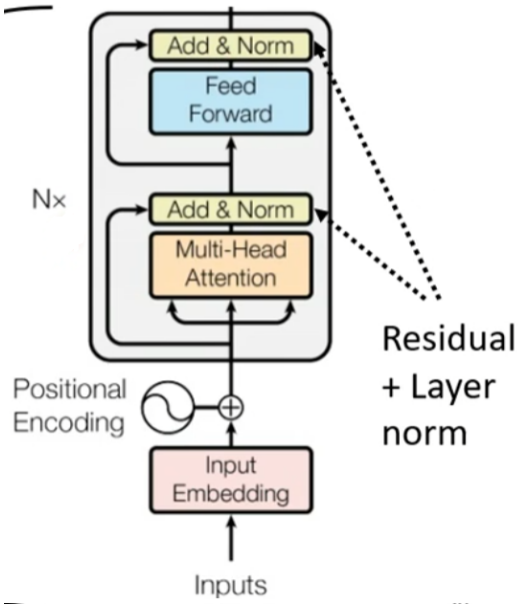
而在Tranformer的编码器里，在每个使用**自注意力**的基础上两次引入了**残差机制**并利用**Layer Norm**归一化（图中公式分母课件打错了应该是x_i）。

补充：

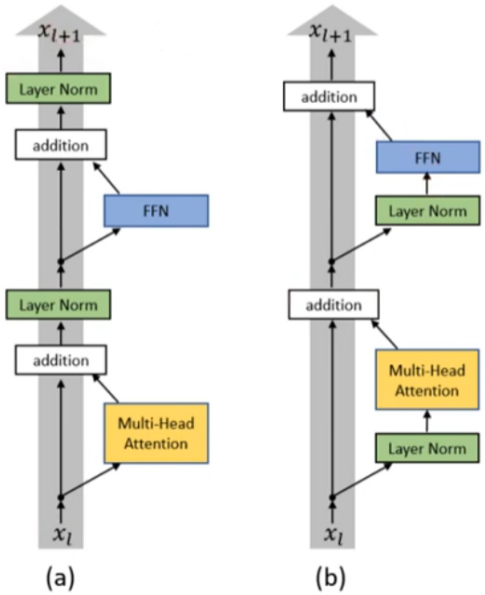
- BN受限于受限于batch size，难以处理动态神经网络中的变长序列的mini-bach。
- LN对不同时间步进行横向标准化，使得每一个时间步都有自己的分布，从而可以处理单一样本、变长序列，而且训练和测试处理方式一致。



在解码器重利用多头注意力（如果需要提取token不同的特征就需要多个的关键字q，那么就利用矩阵算出不同的Q以及对应K,V，最后利用输出权重W^O计算出输出O）和残差机制，再利用线性层变换输出



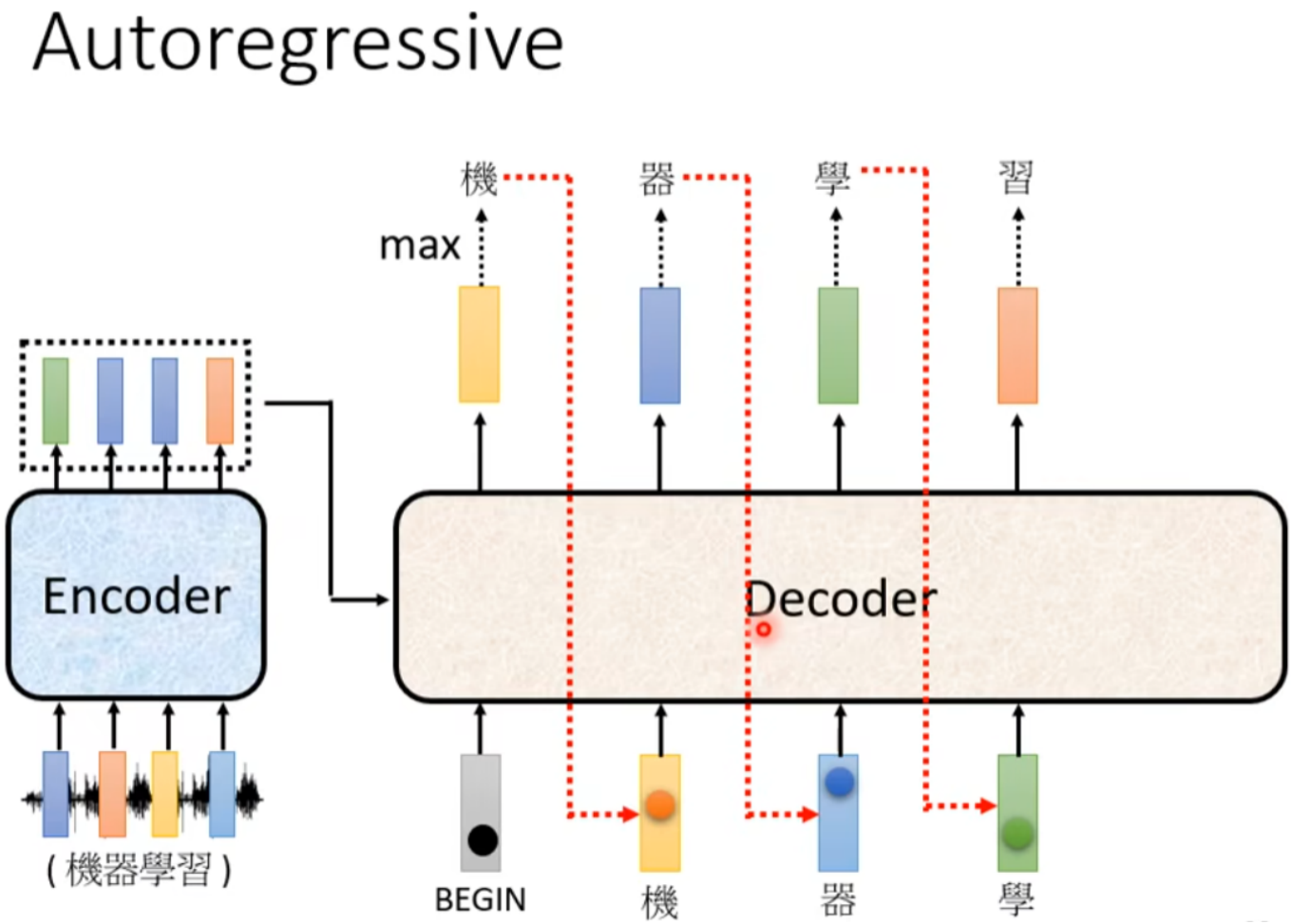
同时，对于Transformer编码器模块的顺序不一定这样是最好的（b结构中先归一化再用神经网络计算权重的效果更好），还有归一化还可以再优化。



Decoder

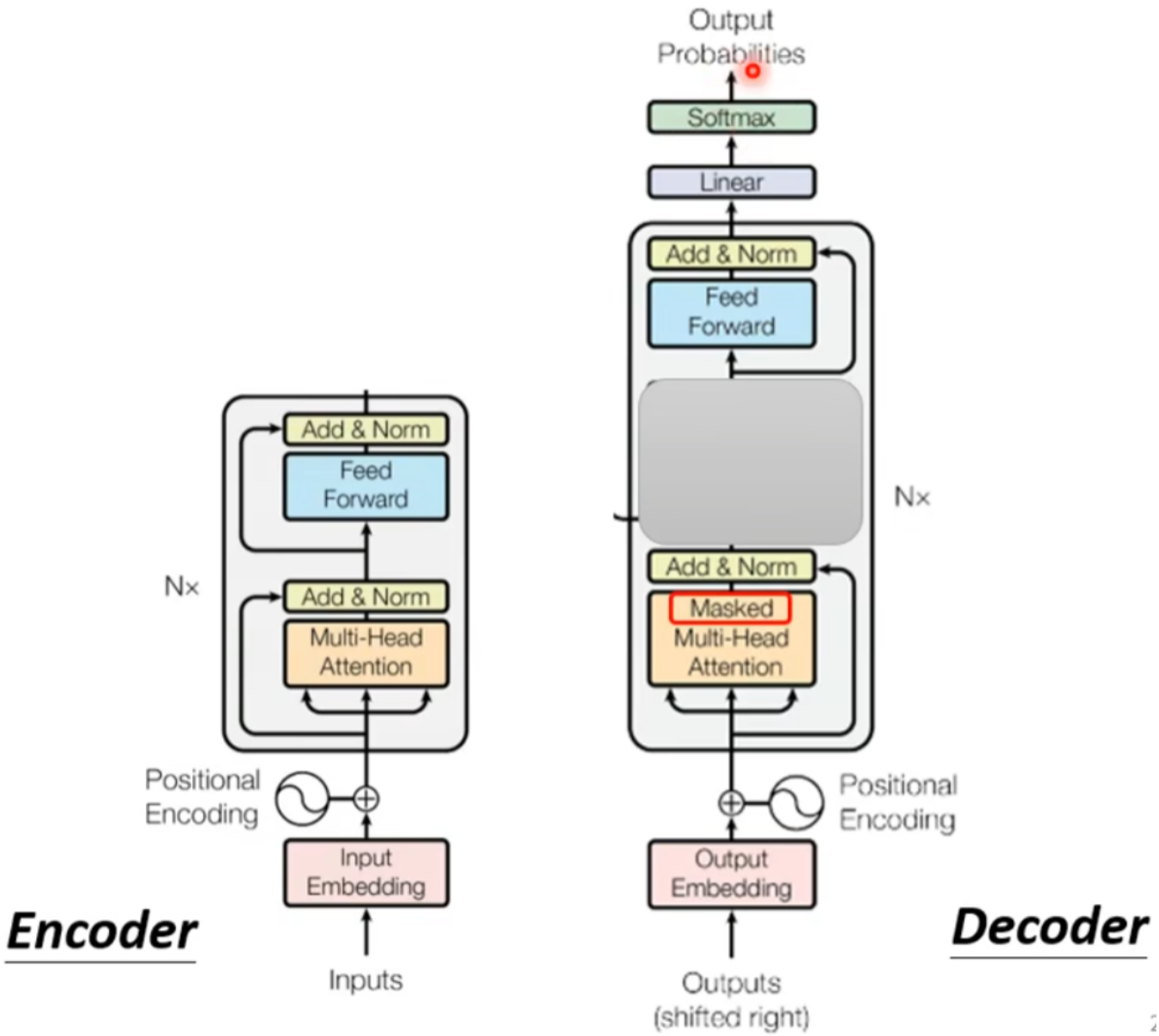
- Autoregressive(AT)

从Encoder的处理结果进入Decoder时首先需要增加一个BEGIN (special token)，输出用softmax选择最后可能的结果（机率）。之后每次的输入是自己上一次的输出。

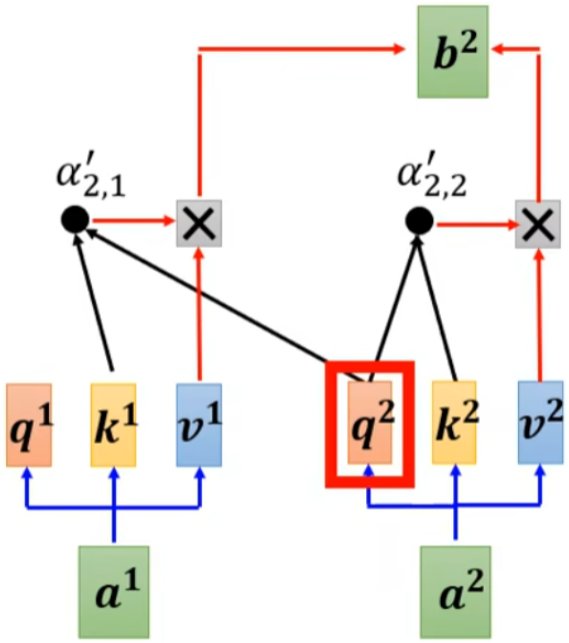


是否会产生连带错误?

比较Transformer的编码器和解码器，我们将解码器遮住，会发现两者其余部分非常接近。

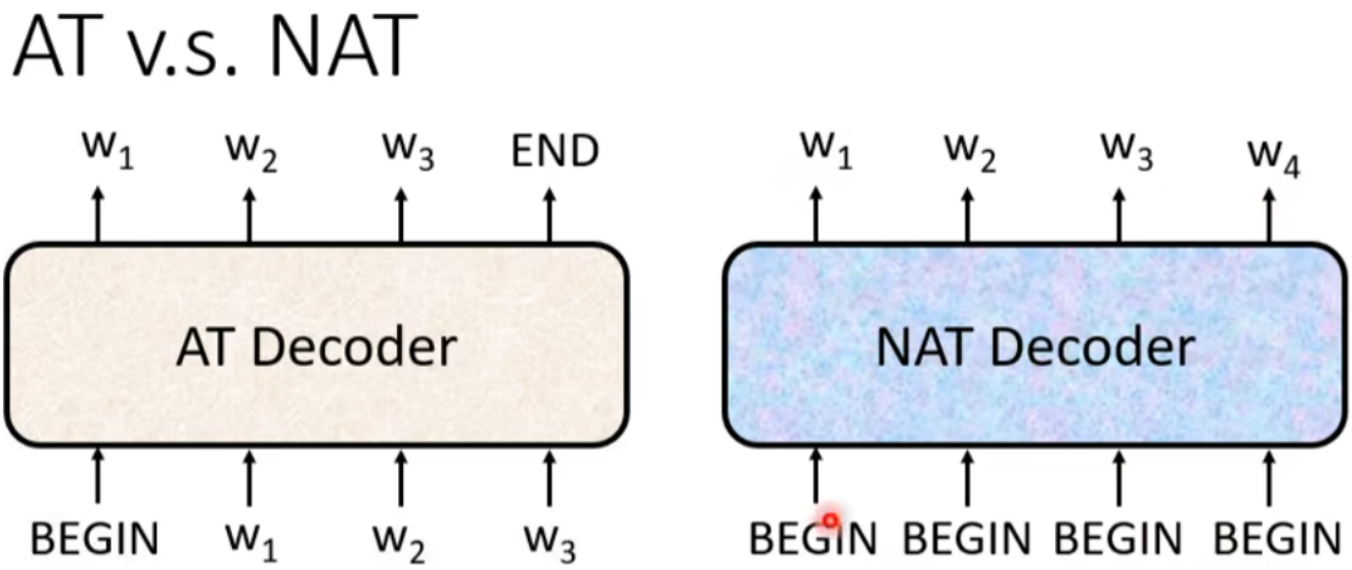


需要注意的是，在解码器中的核心从Self-attention变成**Masked** Self-attention：原本前后都是相关的，但现在前者不能考虑后者的关系。（是因为后者的生成依赖之前的输出，因此之后的输出都还没出现）



- Non-Autoregressive(NAT)

NAT是直接丢入一堆BEGIN，但此时到底要生成多长就取决于输入的token数量（小于等于），而AT是算END的概率自己决定。

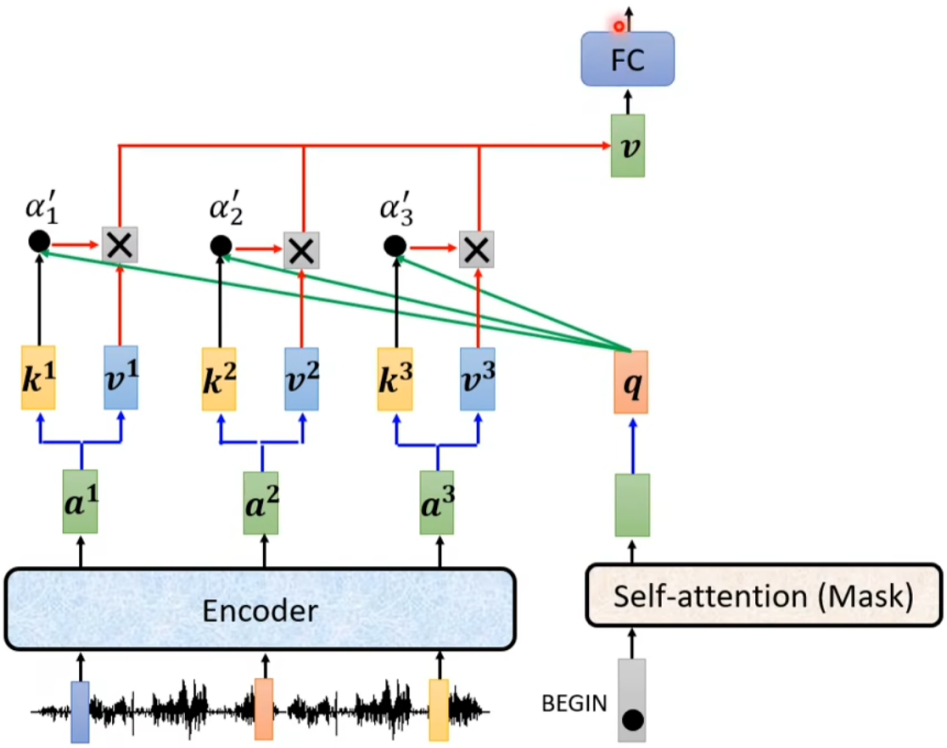


- 那么如何决定输入BEGIN的数量：
  - 其他的预测模型预测输出长度
  - 给一个特别长的序列，然后忽略输出是END之后的输出token

这样方式，可以平行处理，一个步骤产生（快）；可以控制输出的长度。但效果可能不如AT，因为依赖后处理或迭代优化。

- 交叉注意力

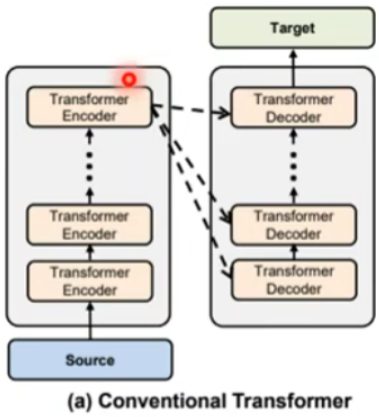
Q来自于Decoder，K, V来自于Encoder的最后一层。

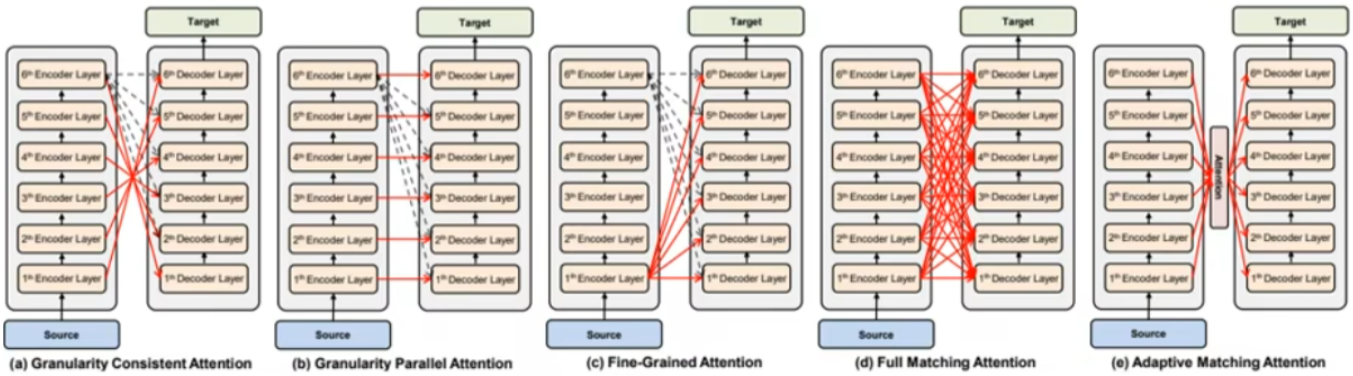


但K, V肯定可以来自不同的层，效果也不同。

Cross Attention

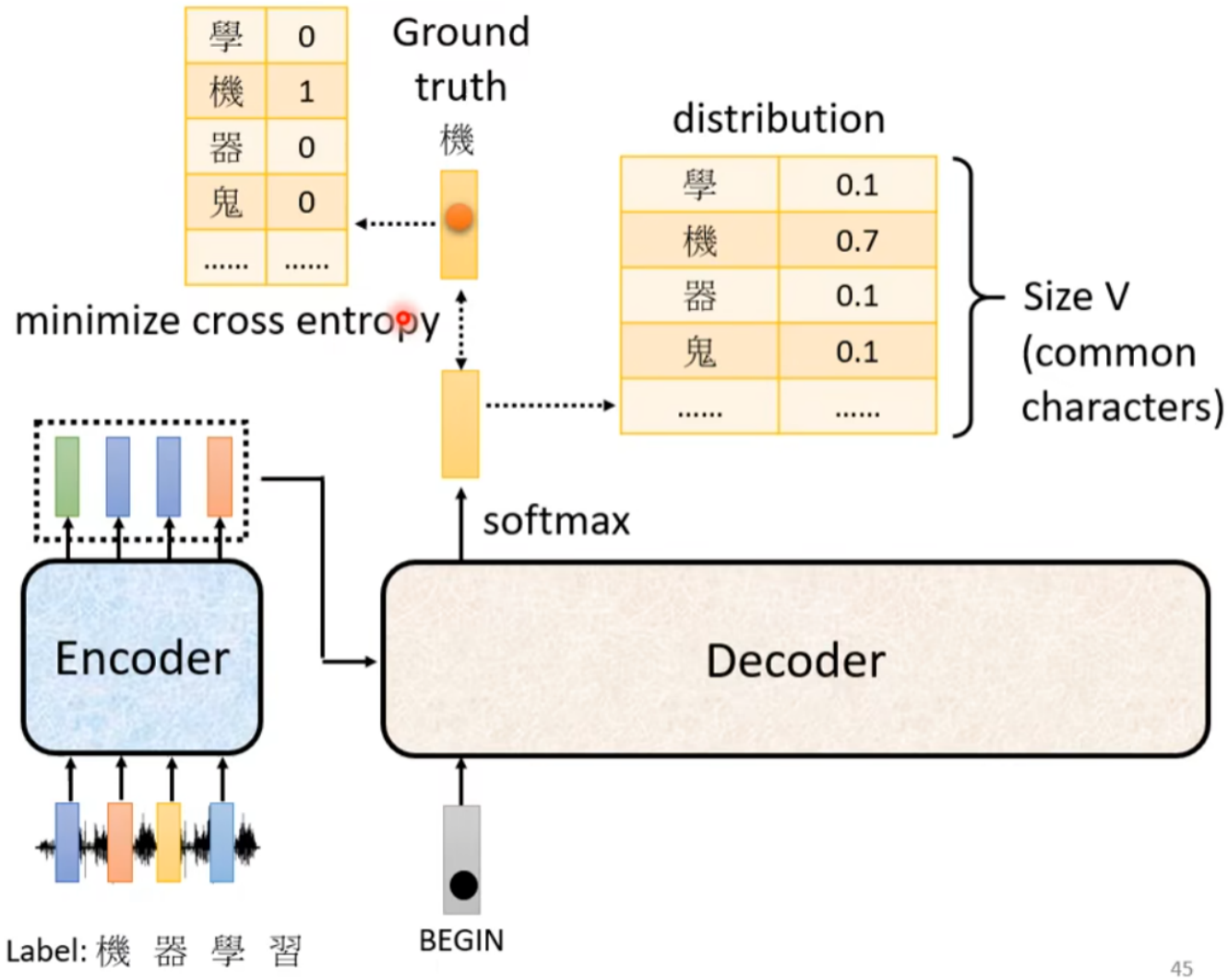
Source of image:
https://arxiv.org/abs/2005.08081





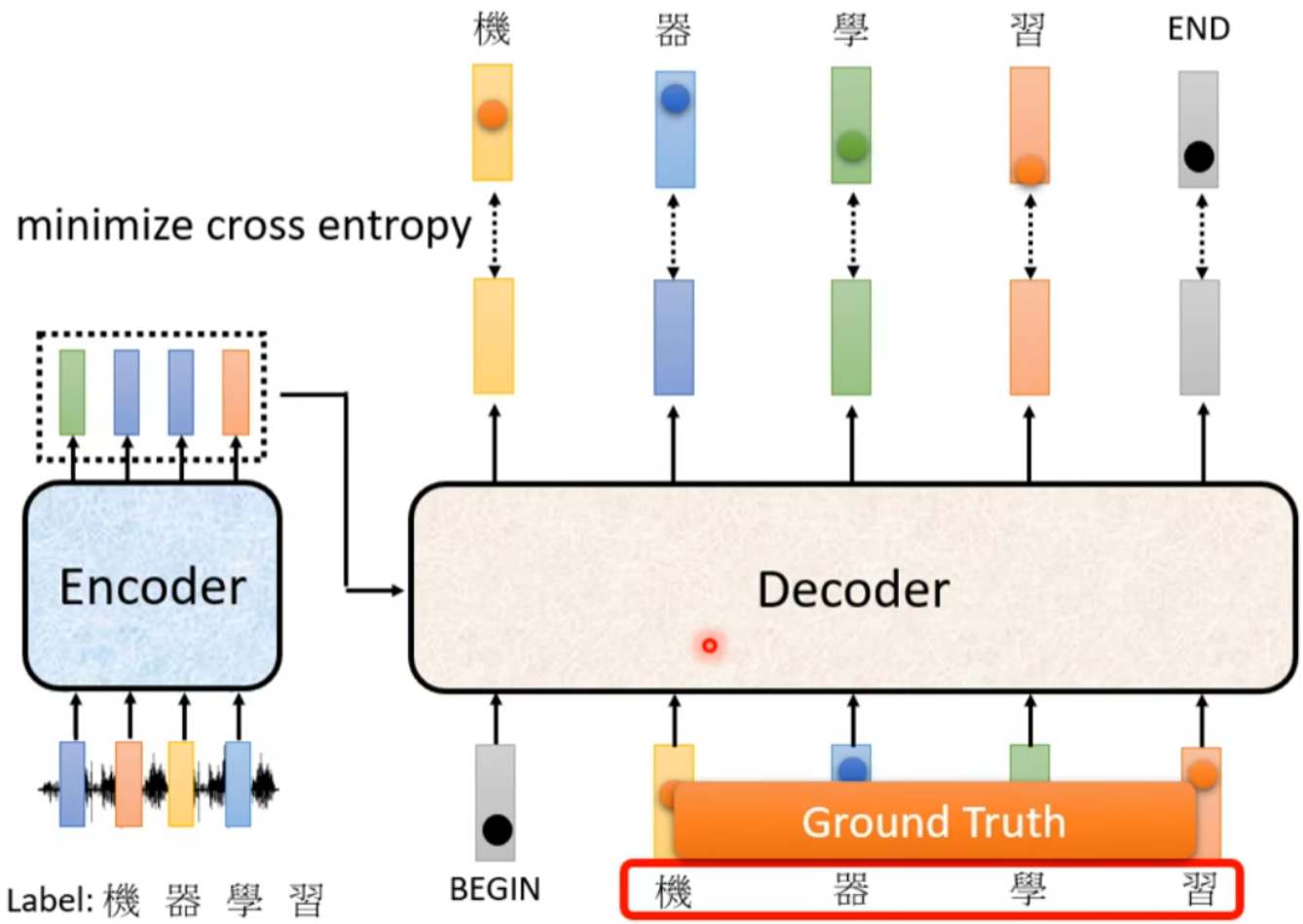
训练

（1）损失函数：类似于分类，每一个token的输出计算loss，所有交叉熵的和越小越好



（2）Teatcher Forcing：把真实标签作为输入

通过在训练时直接使用真实标签作为下一步的输入，而不是使用模型的预测值，从而加速训练并防止错误累积。



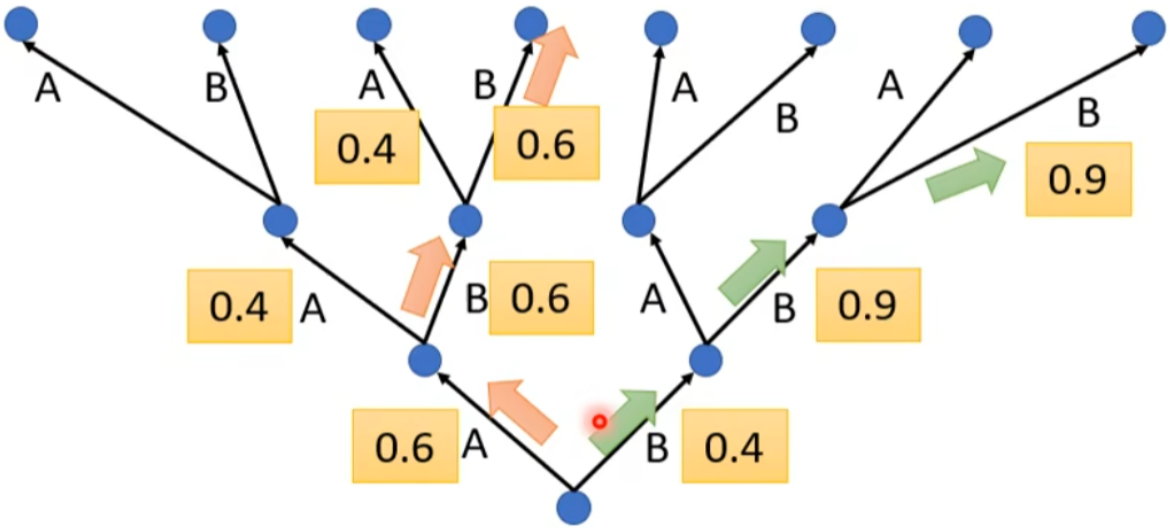
训练时，模型总是看到真实标签作为输入，而推理时缺用自己的预测值作为下一步输入，导致测试时性能下降。可以采用Scheduled Sampling故意引入一些错误的信息，提高模型能力。

（3）其他训练Tips：

- 复制机制Copy Mechanism：人名，特有名词，摘要（一般要百万篇文章训练）
- 引导注意：强迫对Attention有一个固定的样貌
- Beam Search：找到分数最高的路。但有时有用（有唯一最佳答案），有时没用（需要一点创造力，可以加点噪声 -> 不是人类觉得好的就是最好）

Beam Search

Assume there are only two tokens (V=2).



The red path is ***Greedy Decoding***.

The green path is the best one.