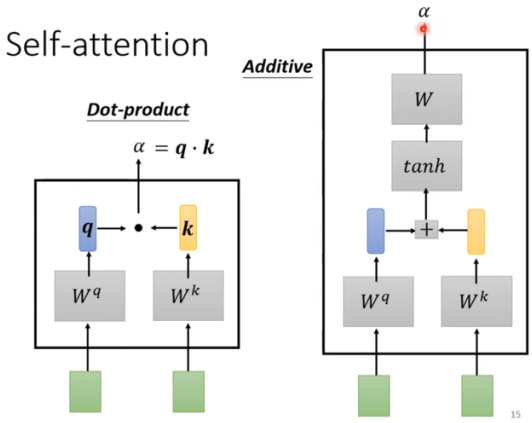
注意力机制

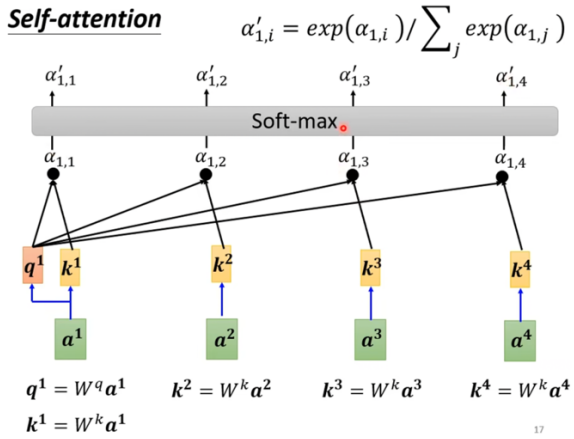
计算 attention score 的模组：

dot product (常用) / additive



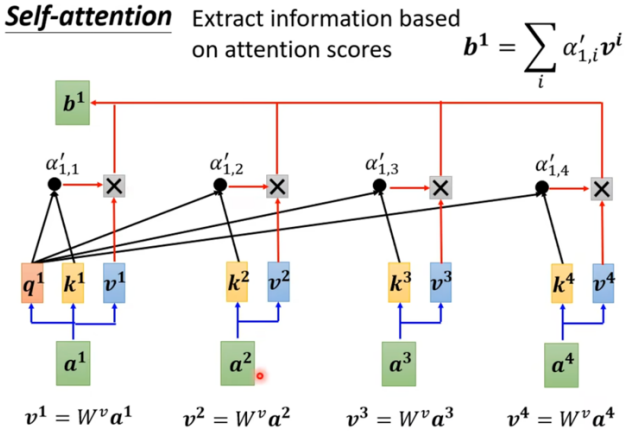
计算 attention score:

具体步骤，q 与 k，计算 a1 与其他每个向量（包括自己）的 attention score(a)。再过一遍 softmax 层（与分类模型一样)。

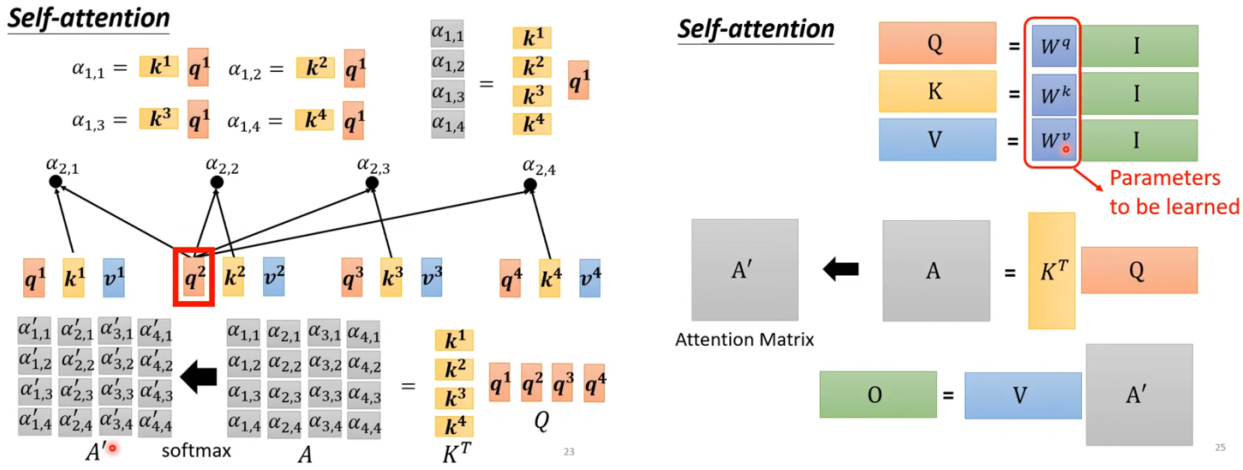


根据 attention score 抽取 information – 计算 b

用 Wv 和第一步的 a 相乘获得 v，再将 v 和第二步的 a 相乘，最后再相加。

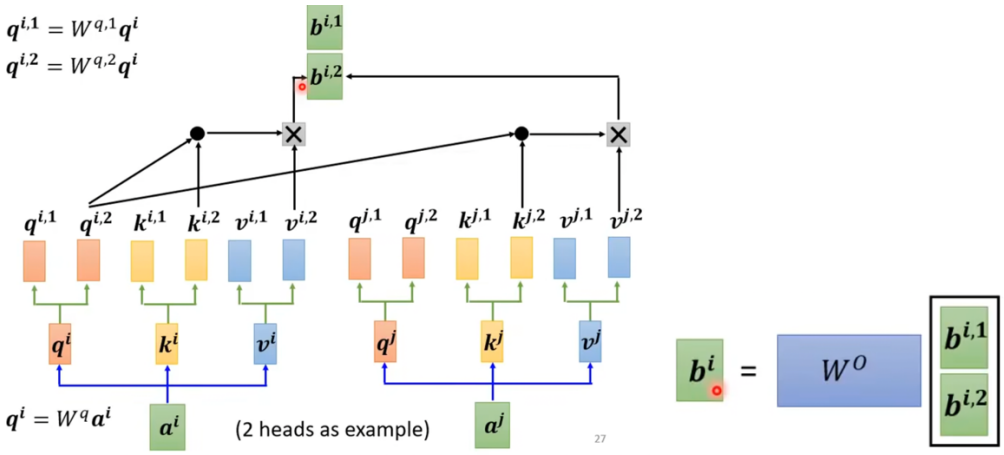


结果的矩阵表达：



只有 Wq, Wk, Wv 这三者是未知的。（透过 training data 找出）

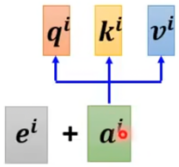
Self-attention 的进阶版本： Multi-head self-attention



Position encoding: (有各式各样的方法)

现在的 self-attention 没有位置信息，各个 input 后续的操作都是一样的。

为此，每一个位置，都应设有一个唯一的 position vector (ei)



Self-attention 与 CNN 的关系：

On the relationship between Self-attention and convolutional layers

Self-attention 与 RNN 的关系：

Fast Autogressive Transformers with Linear attention

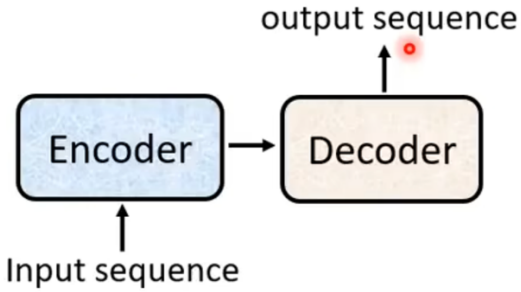
RNN 没有办法平行计算，前者可以

Transformer

是一个 sequence to sequence(seq2seq)的 model，可以做的工作很多。

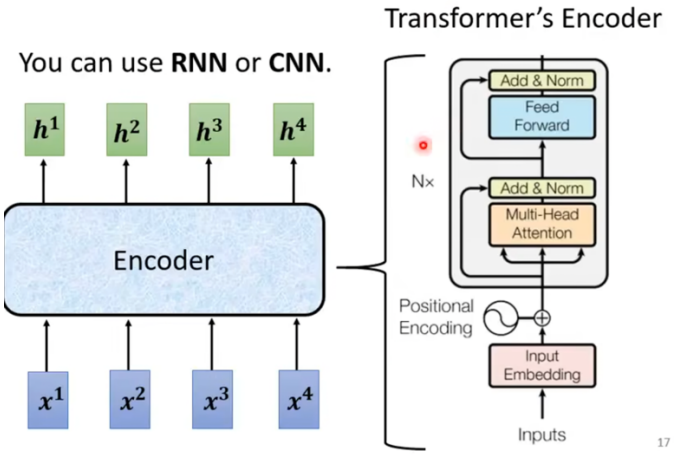
基础结构：

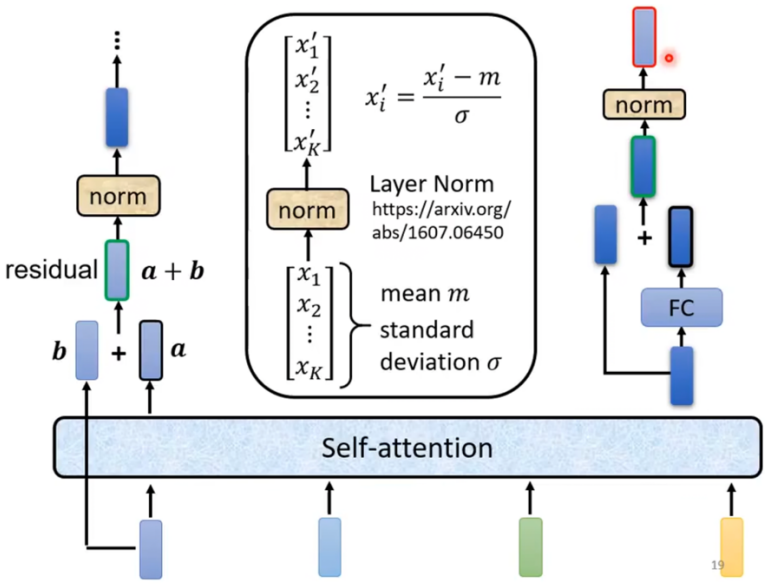
Encoder 处理数据，给 Decoder，之后决定输出。



Transformer Encoder:

给一排向量，输出另外一排向量。

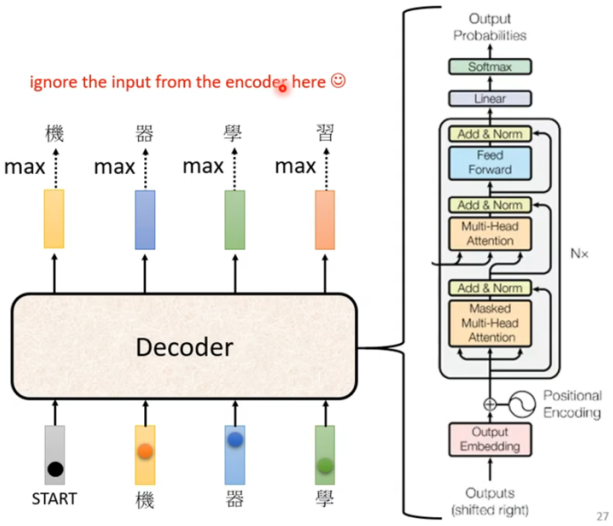




最初，对 input 进行 position encoding。
之后，从左下角开始，做 self-attention，得到的 a 与 input 相加（残差结构）。
之后，过一遍 layer Norm。得到一个值 temp。
再 temp 过一遍 Fully connected network，并将结果与 temp 相加（残差结构）。
最后再过一遍 norm，得到最终的输出结果。

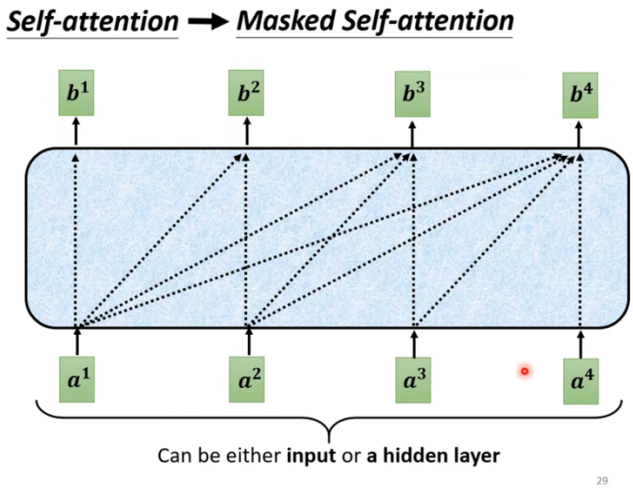
注意，这只是一个 block 的计算操作，在 encoder 中会有 N 个 block。

Transformer Decoder: Auto-regressive

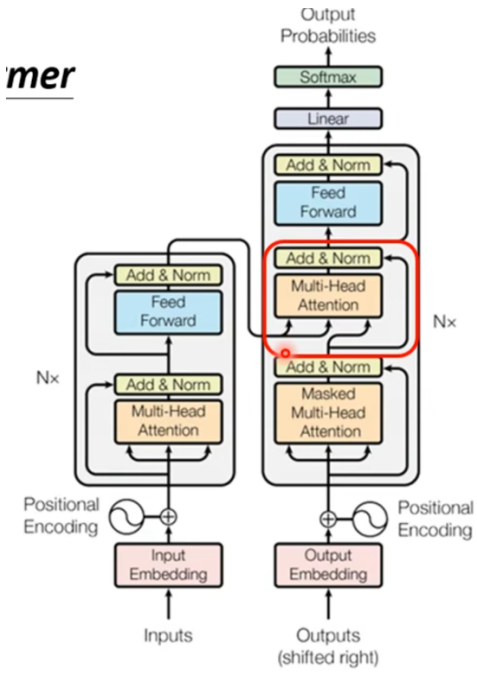


Masked self-attention:

考虑 b1 时，只能看 a1。考虑 b2 时，只能考虑 a1, a2。



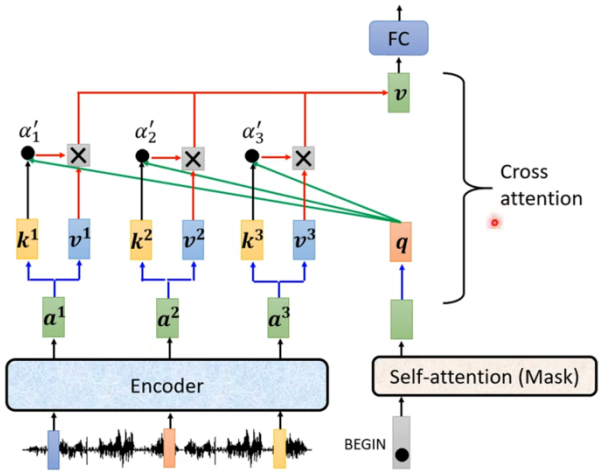
Encoder 与 Decoder 之间的连接：



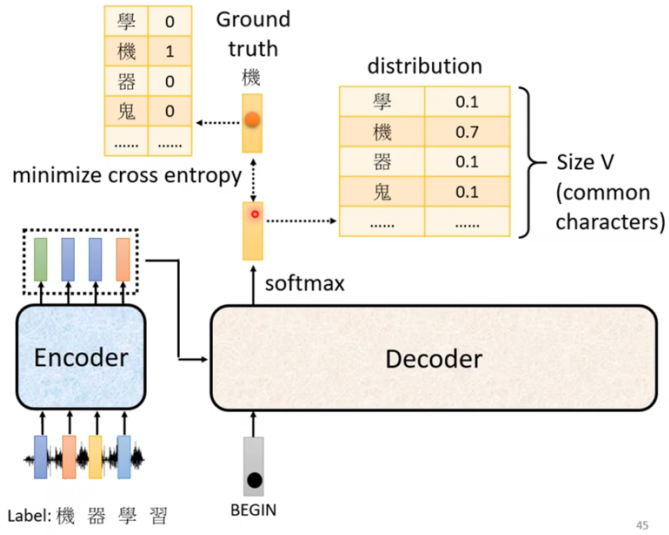
红色标注出来的地方是二者连接的桥梁，可以被称为 cross attention。

Encoder 提供左边两个箭头，Decoder 提供另一个。

q 来自 Decoder，k 与 v 来自 encoder。

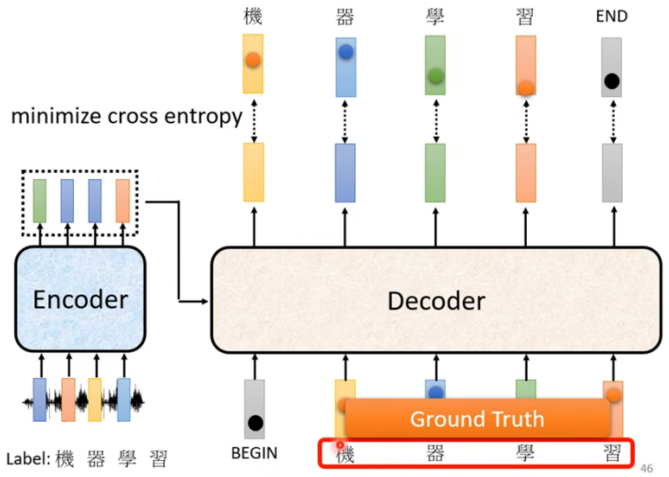


Model Training:



根据声音讯号得到中文的过程：
Decoder 每得到一个字，相当于进行一次分类。Distribution 可以看做一个带有所有中文字的字典。

在训练过程中，我们会给 Decoder 正确答案。(Teacher Forceing: 用正确的答案当作 Decoder 的 input)



有 Begin 的时候，输出机。有 Begin 有机的时候，输出器。有 Begin，机，器的时候，输出学……