7月20日

一、李宏毅2021春机器学习课程第5.1节：Transformer（一）

1 序列到序列的模型

Transformer本质上就是一个Sequence-to-sequence的model，我们经常缩写为Seq2seq，所有我们就先来讨论一下什么是Seq2seq的model。

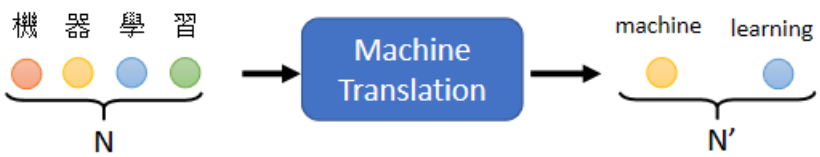
上一节在讲自注意力机制的时候就提到过，**input是一个sequence时，output有三种可能**：

- 每一个向量都有一个对应的Label，就比如词性标注的问题。
- 输入多个向量，只需要输出一个Label，就比如Sentiment Analysis的问题，分析一句话的情感是正面还是负面。
- 机器要自己决定应该输出多少个Label，这就是Seq2seq的模型要处理的问题。

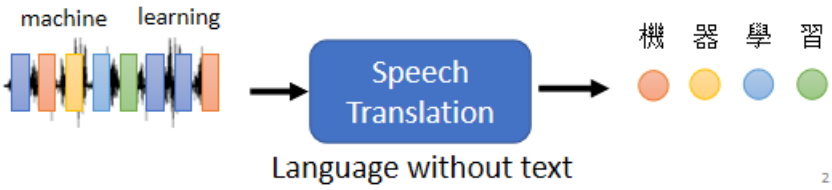
举例来说，Seq2seq一个很好的应用就是**语音辨识**：

在做语音辨识的时候，输入是声音讯号，声音讯号其实就是一串的vector，输出是语音辨识的结果，也就是输出的这段声音讯号所对应的文字。**输出的长度由机器自己决定**，由机器自己去听这段声音讯号的内容，并决定应该要输出几个文字。

还有很多其他的例子，比如说机器翻译：



甚至可以做更复杂的问题,比如说做语音翻译：



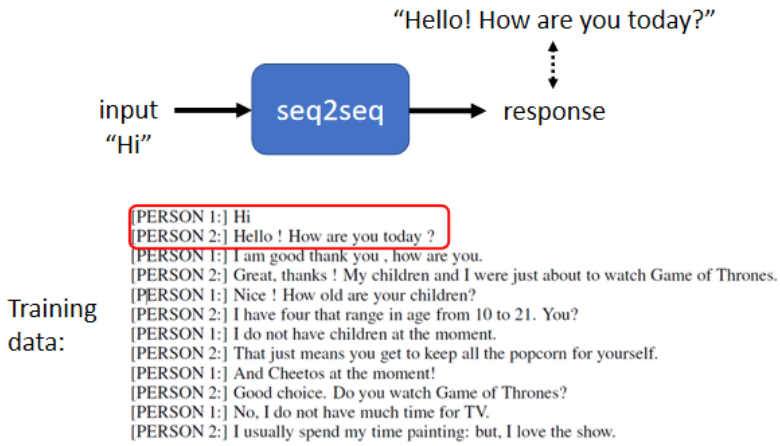
接下来就要介绍一下Seq2seq模型常见的一些应用，你可以发现这个模型的应用范围是非常广泛的。

---

2 Seq2seq model 常见的应用场景

2.1 聊天机器人

可以用Seq2seq model来训练一个聊天机器人，聊天机器人就是你对它说一句话，它要给你一个回应，**输入输出都是文字**，而文字又可以看成是 **vector sequence**，因此可以考虑用 Seq2seq model 来处理。

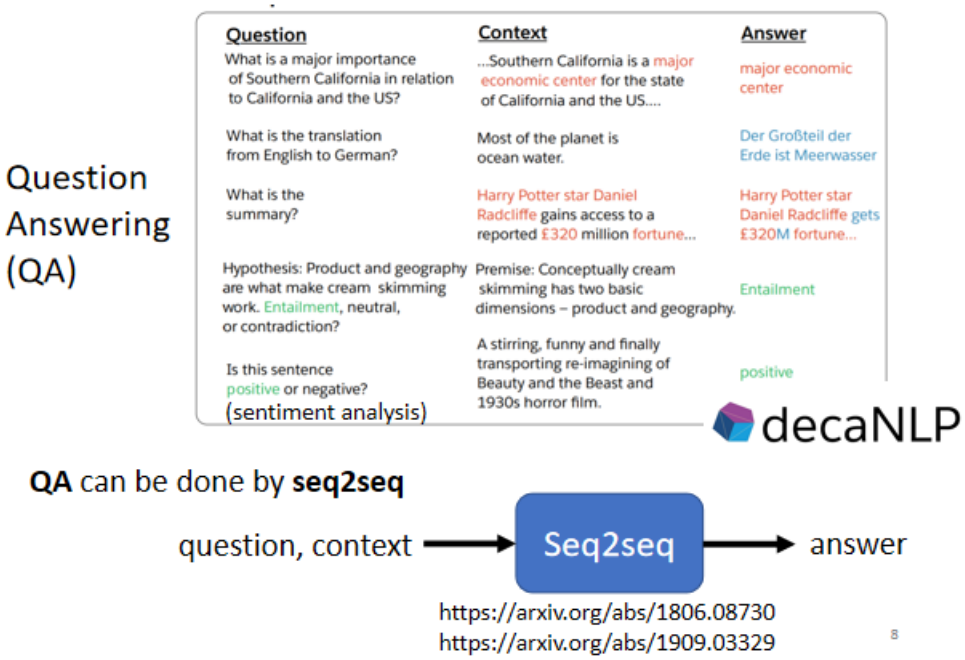


为了训练我们的模型，需要**收集大量人的对话**，像电视剧、电影的台词等等，可以很容易收集到很多人跟人之间的对话。假设在对话里面某一个人说"Hi"，另外一个人回答说"Hello, How are you today"，那我们就可以教机器看到输入是Hi，那你的输出就要跟"Hello, How are you today"越接近越好，这是一个基本的训练思路。

2.2 问题回答 (QA)

事实上Seq2Seq model 在自然语言处理（NLP）的领域应用的也非常广泛，特别是question answering任务上的应用。而**其实很多自然语言处理的任务，都可以看成是question answering，QA的任务。**

Question Answering 就是给机器读一段文字，然后你问机器一个问题，希望他可以给你一个正确的答案。

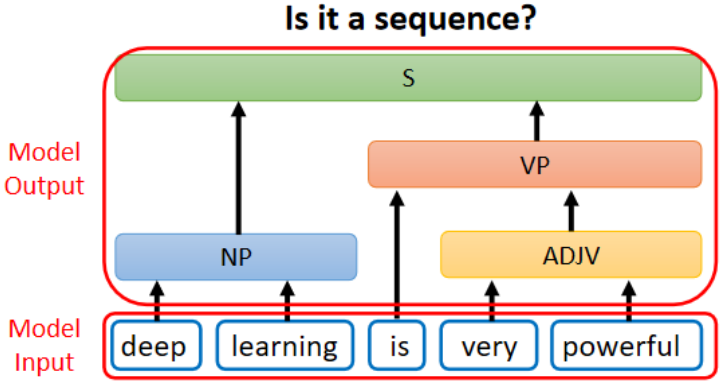


- 假设你今天想做的是机器翻译的任务，那机器读的文章就是一个英文句子，**问题**就是这个句子的中文翻译是什么，然后输出的**答案**就是对应的中文。
- 或者你想要让机器自动作自动摘要的任务，就是给机器读一篇长的文章，让机器把长文章中的重点摘录出来，那你就是给机器一段文字，**问题**是这段文字的摘要是什么，然后输出的**答案**就是这篇文章的摘要。
- 又或者是你想要让机器做 Sentiment analysis的任务，就是机器要自动判断一个句子是正面还是负面的。你就给机器要判断正面还是负面的文章，**问题**就是这个句子是正面还是负面的，然后输出的**答案**就是这个句子对应的情感。

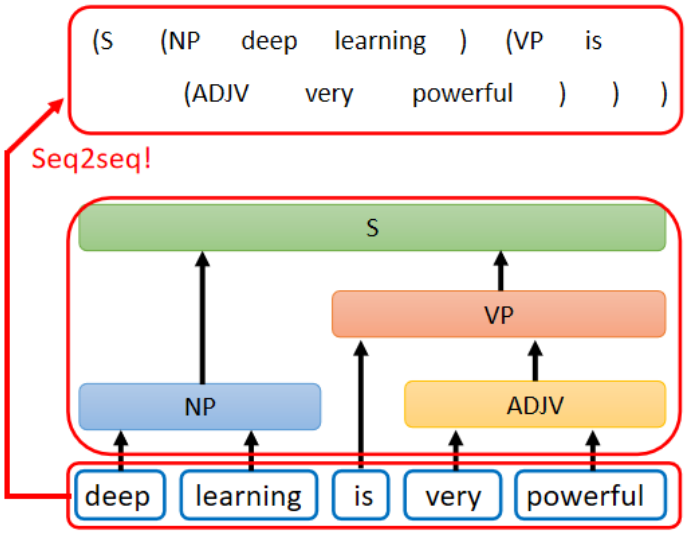
所以各式各样的NLP的问题，往往都可以看作是QA的问题，而**QA的问题,就可以用Seq2Seq model来解。**

2.3 文法剖析

还有一些**你不觉得它是一个Seq2Seq model 的问题，但你仍然可以用 Seq2Seq model 硬解这个问题。**就比如说**文法剖析的问题**：给机器一段文字，机器要做的事情是产生一个**文法的剖析树**，告诉我们deep加learning合起来是一个名词片语，very加powerful合起来是一个形容词片语，形容词片语加is以后会变成一个动词片语，动词片语加名词片语合起来,是一个句子。



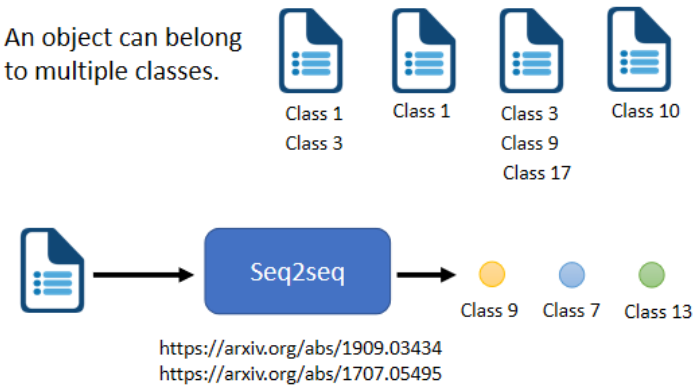
这个输出看起来不像是一个 Sequence，输出是一个树状的结构，但**事实上一个树状的结构，也可以硬是把他看作是一个 Sequence**：



这一个 **Sequence就代表了这一个 tree 的 structure**，你先把 tree 的 structure 转换成一个 Sequence 以后，你就可以用 Seq2Seq model 硬解这个问题。看起来挺离谱的哈？我反正觉得挺离谱的，但是实际上是真的可以做到的，就比如这篇paper Grammar as a Foreign Language (arxiv.org) 就是这样做的。

2.4 多标签分类

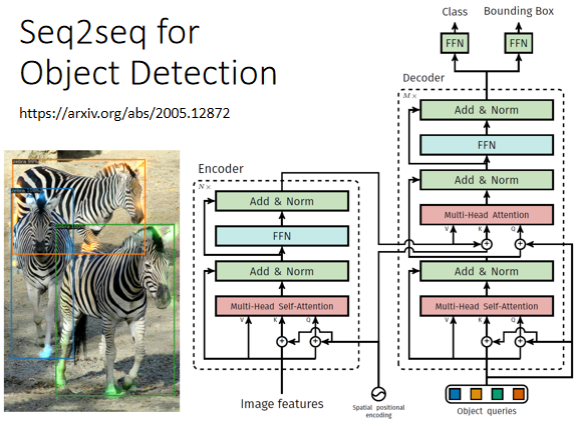
Seq2Seq model 也可以应用在多标签分类的问题上，所谓的 multi-label classification ，意思是说**同一个东西，它可以属于多个class**，举例来说，你在做文章分类的时候，一篇文章可能既属于类别1，又属于类别2。



由于一篇文章属于多少个类别是不确定的，所有我们用 Seq2Seq model 来解决这个问题，让机器自己决定输出几个类别。

2.5 对象检测

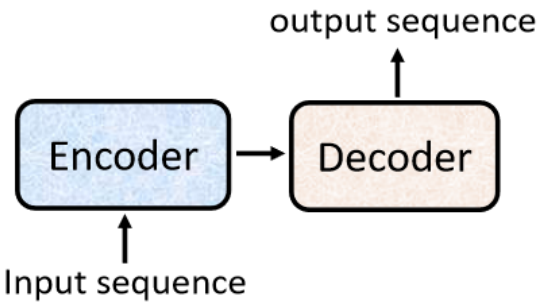
object detection，这个看起来跟 Seq2Seq model 八竿子打不着的问题，也可以用 Seq2Seq model 硬解，这里就放一下论文的链接：End-to-End Object Detection with Transformers (arxiv.org)



---

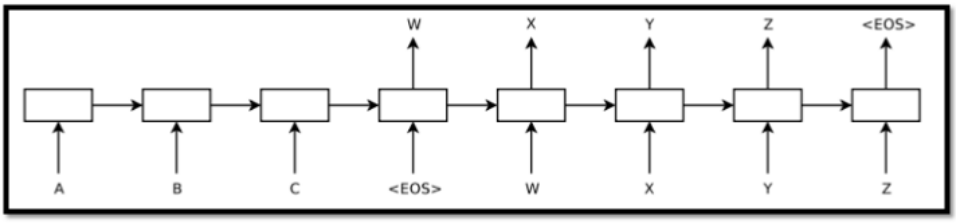
3 Seq2seq model的整体架构

现在我们要来看看这个似乎啥都能做的 seq2seq model 究竟是怎么实现的。**一般的 seq2seq model 会分成两个部分，一个部分是Encoder，另一个部分是Decoder**：



我们**输入一个sequence到Encoder，Encoder负责处理这个sequence，再把处理好的结果丢给Decoder，由Decoder决定要输出什么样的sequence**，等一下还会具体介绍Encoder和Decoder内部的架构。

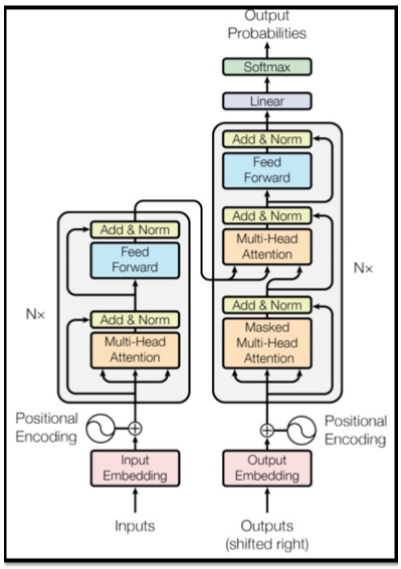
seq2seq model 的起源其实非常早，在2014年的9月，就有一篇把 seq2seq model用在翻译上的论文：Sequence to Sequence Learning with Neural Networks (arxiv.org)



Sequence to Sequence Learning with Neural Networks

https://arxiv.org/abs/1409.3215

而**在今天讲到 seq2seq model 的时候，大家更多指的是我们今天的主角，也就是transformer。**它有一个Encoder架构，有一个Decoder架构，还有很多花花绿绿的block，接下来就是要详细介绍一下每一个花花绿绿的block分别在做的事情是什么。



Transformer

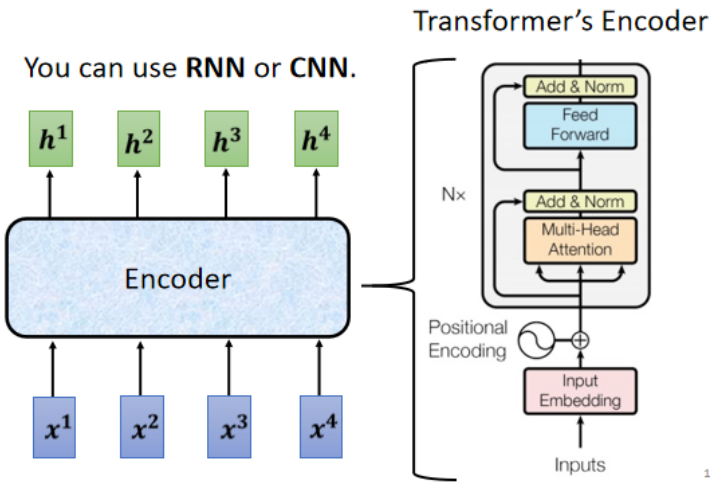
https://arxiv.org/abs/1706.03762

它有一个Encoder架构,有一个Decoder架构,它裡面有很多花花绿绿的block,等一下就会讲一下,这裡面每一个花花绿绿的block,分别在做的事情是什麼

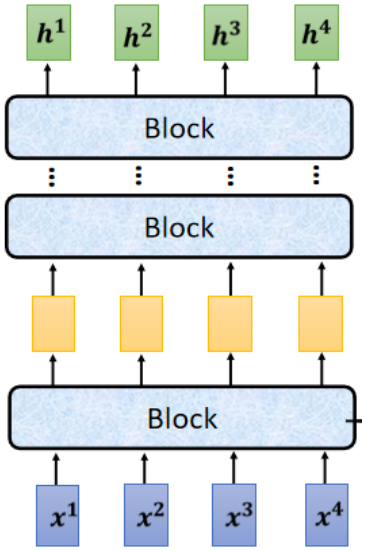
---

4 编码器的具体架构

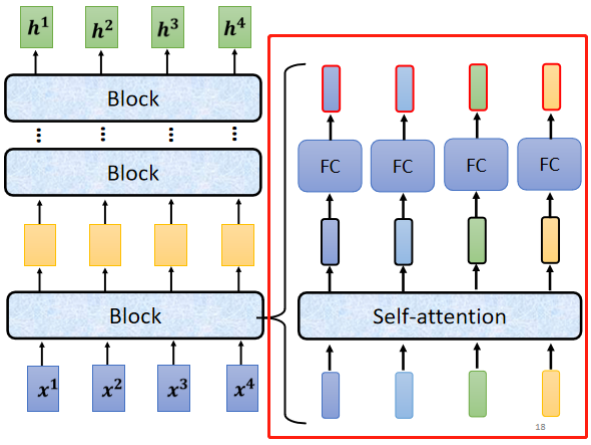
seq2seq model Encoder 要做的事情就是**给一排向量，输出另外一排向量。**



这件事情听起来很简单，很多模型都可以做到，可能第一个想到的就是上一节刚刚讲完的 self-attention，而**事实上在transformer里面，它的Encoder用的就是self-attention**，上面的这张图来自于原始论文，看起来有点复杂，我们用下面这张简化后的图，来仔细地解释一下Encoder的架构。

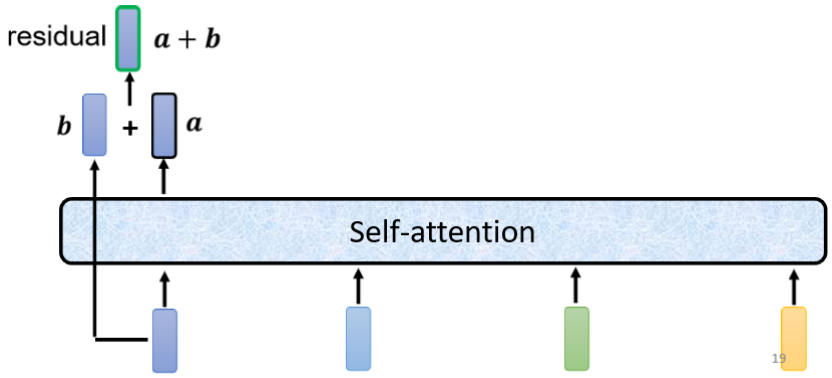


现在的Encoder里面，会**分成很多的block**，每一个block都是输入一排向量，输出一排向量，最后一个block会输出最终的vector sequence，**每一个block也并不是neural network的一层，而是做了好几个layer在做的事情。**在transformer的Encoder里面，每一个block做的事情大概是这样的：



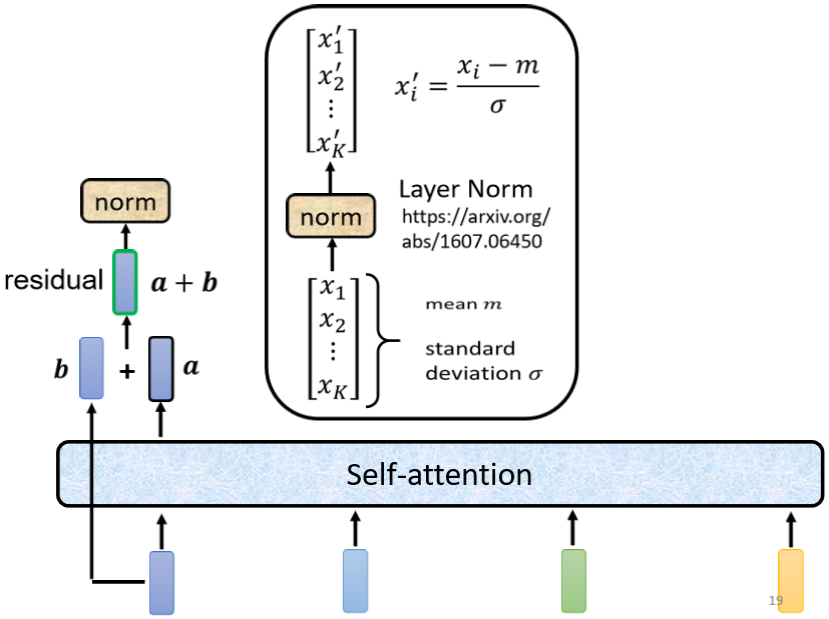
- 首先input一排vector以后，对它们做self-attention，考虑整个sequence的资讯，然后输出另外一排vector。
- 接下来这一排vector，会分别丢到不同的fully connected network里面，再输出另外一排vector，这一排vector就是这个block的输出。

当然这是简化版本的一个大致描述，事实上在原来的**transformer里面做的事情要更复杂一些。**



刚刚我们说self-attention层是输入一排vector，考虑整个sequence的资讯，然后输出另外一排vector。事实上在transformer里面，我们**不只是输出这个vector**，我们还要**把这个vector加上它的input**，再得到新的output 。这个做法就叫做residual connection，主要是**为了解决网络退化和梯度破碎问题**，具体解释可以参考这篇文章残差网络解决了什么，为什么有效？ - 知乎 (zhihu.com)，这里就不再进一步说明。

得到residual的结果以后，还需要再做一件事情叫做normalization，这里使用的不是batch normalization，而是layer normalization。

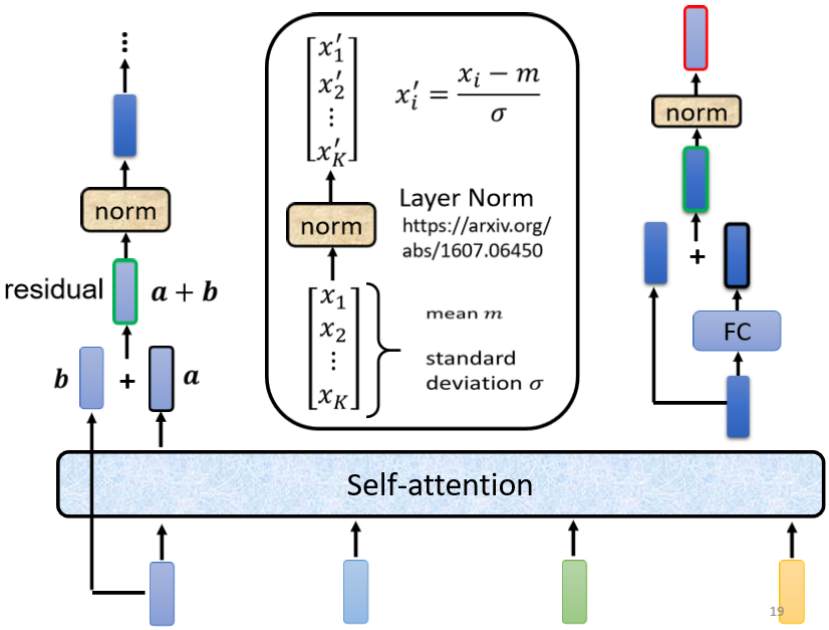


layer normalization做的事情比batch normalization更简单一点，要注意的是，**batch normalization是对不同example，不同feature的同一个dimension，去计算均值和标准差**；而**layer normalization是对同一个example，同一个feature里面不同的dimension，去计算均值和标准差。**

计算出均值和标准差以后，就可以做一个normalize，把输入vector里面的每一个dimension减掉均值m，再除以s标准差σ以后就得到layer normalization的输出：

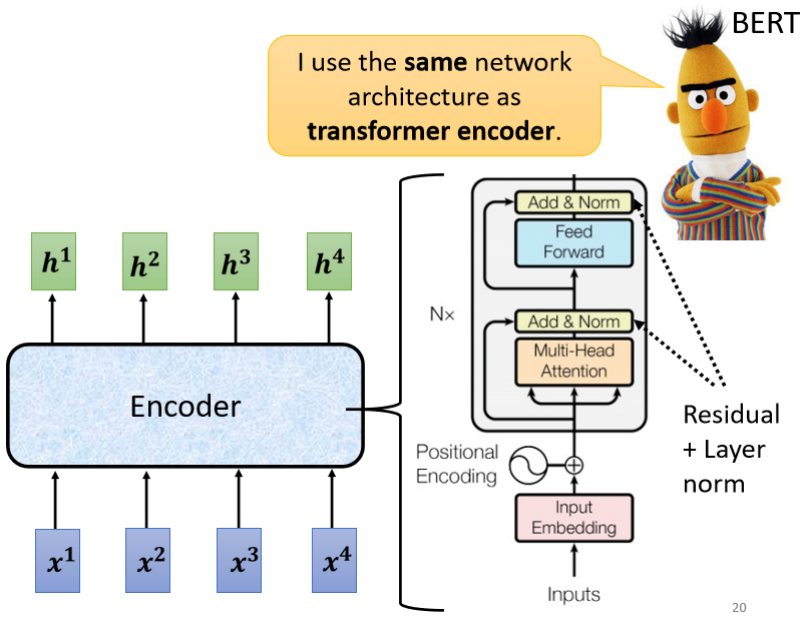
$$x_i' = \frac{x_i - m}{\sigma}$$

这个输出才是self-attention层真正的输出，它会作为下一个全连接层的输入。



而**fully connected network 这边也设计了residual的架构**，也就是说我们会把全连接层的input跟它的output加起来，才得到新的输出。并且得到residual的结果以后，也需要**再做一次layer normalization**，才得到全连接层真正的输出。

现在回到原论文里Encoder的图片，其实就是我们刚刚讲到的过程：



- 首先我们的输入通过**input embedding层得到嵌入表达**，这里还需要**加上positional encoding**，来让我们之后的self-attention能够考虑到输入位置的资讯。
- 接下来进入**Multi-Head Attention层**，这个就是self-attention的block，这边特别强调了它是Multi-Head的self-attention
- **Add&norm**，**就是residual加layer normalization**，Multi-Head Attention层的输出要做residual加layer normalization之后，才会输入下一个模块，也就是全连接层。
- **全连接的 feed forward network 的输出也需要再做一次 Add&norm，才是整个block的输出。**
- 然后这个block会**重复n次**，这个复杂的block其实在之后会讲到的一个非常重要的模型**BERT**里面会再用到。 **BERT其实就是transformer的encoder。**

这就是transformer的Encoder的具体架构介绍啦，**至于Decoder的架构又是怎样的，且听下回分解。**