《Attention Is All You Need》

摘要：

1. **本文的背景：**2017 年之前，主流序列转换模型都是基于 RNN or CNN 的，而表现较为出色的 RNN 架构在解码器上不能有效进行并行计算，导致绝大多数模型有着算法效率低下的问题。
2. **所提出的解决方法：**一个新的简单网络结构——**Transformer**，其核心结构仅仅是基于注意力机制，而不完全是 RNN 或 CNN。
3. **模型表现：**在 WMT 2014 英语到德语翻译任务以及英语到法语翻译任务上，分别取得了 28.4BLEU 和 41.8BLEU 的分数，是当时该项目的最好结果。
4. **其他研究分析：**在大型、有限的训练数据中，通过将 Transformer 成功应用于英语句法解析，表明了其可以很好地适用于其他任务，即具有一定的泛用性。

问题背景：

（1） 长短期记忆（LSTM）和门控循环神经网络（GRU）作为序列建模和转换问题的主流方法，使得许多相关研究以循环神经网络和**编码器-解码器体系结构**进行。

（2） 由于 RNN 需要将序列中的每个位置数据与计算时刻的步骤对齐，即所产生的一系列隐藏状态$\boldsymbol{h_t}$，需要用上一时刻隐藏状态$\boldsymbol{h_{t-1}}$和当前时刻t的信息作为输入来获得，而这种固有的顺序性导致了训练样本难以进行并行化计算。

（3） 本论文的研究者们发现**注意力机制**能在不考虑输入或输出序列中的距离的情况下，对依赖关系进行建模，但是当时现行的这些注意力机制却通常都与 RNN 一起绑定使用。

（4） 如何避免顺序计算的网络架构，构建并行计算的模块，是一个能有效提高算法效率的热门问题。

Transformer 的提出：

Transformer 是第一个完全依靠 self-attention（自注意力机制）来计算其输入和输出的表示，而不使用 RNN 或卷积的模型。因此，相比传统的 RNN 模型记忆长度有限且无法并行化（只有计算完t_i时刻后的数据才能计算t_{i+1}时刻的数据），Transformer 可以完全做到并行化计算。【这也是 Transformer 的主要优势】

Transformer 的模型架构：

（1）整体架构；

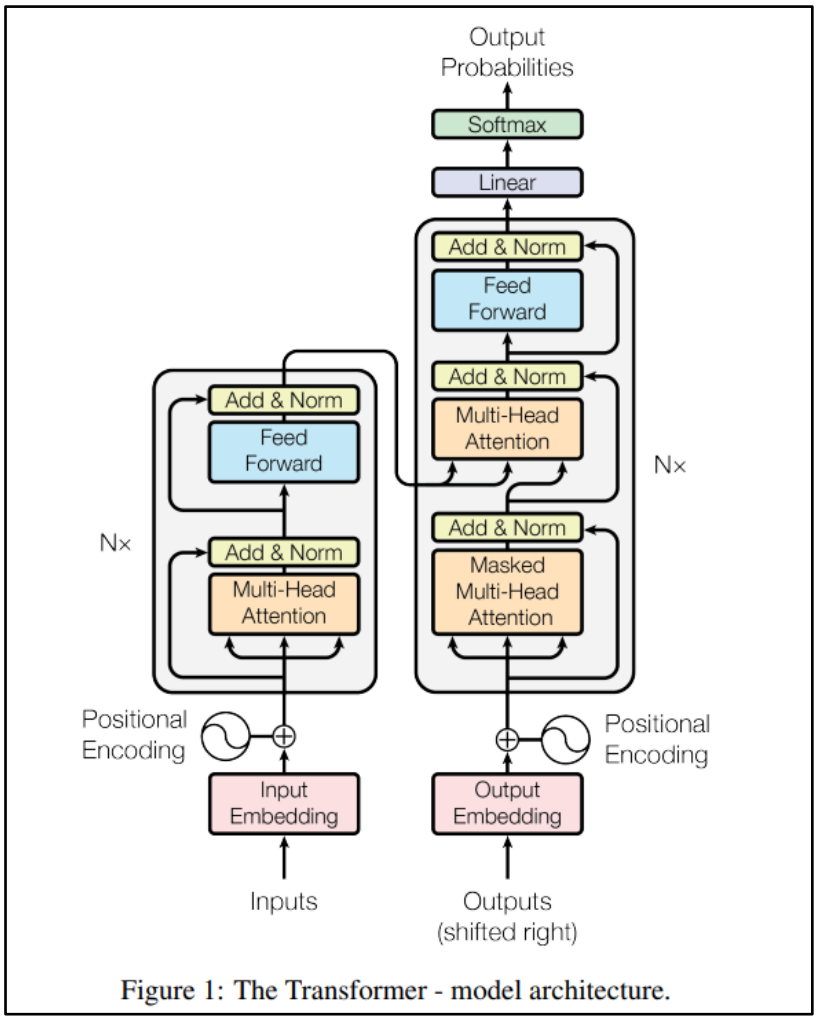


Figure 1: The Transformer - model architecture.

如上图所示，Transformer 主要遵循编码器（图中左边）—解码器（图中右边）结构。其中，每个编码器由$N_x = 6$个相同的模块组成。每个模块又主要由两个子层构成，分别是多头自注意力机制（Multi-Head Attention）和前馈网络（Feed Forward）。此外，对两个子层使用残差连接（Add），并加入层归一化（Norm）。

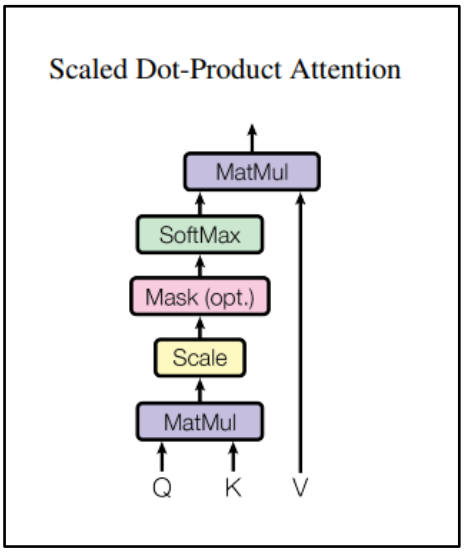
同样，解码器也由$N_x = 6$个相同的模块组成。除了与编码器模块中一样的两个子层外，解码器还加入了第三个子层，该子层对编码器的输出执行带掩蔽的多头注意力机制（Masked Multi-Head Attention），即确保对位置i的预测只依赖于小于i 位置的已知输出来计算。

（2）注意力机制；

1˚缩放点积注意力：（细节模块）

上述结构中的 attention 函数可以被简单描述为将 query 和一组 key-value 映

射到输出 outputs，其中 query、key、value 和 outputs 都是向量。outputs 被计算为 value 的加权求和，其中分配给每个 value 的权重由 query 与对应的 key 的兼容函数来计算得出。

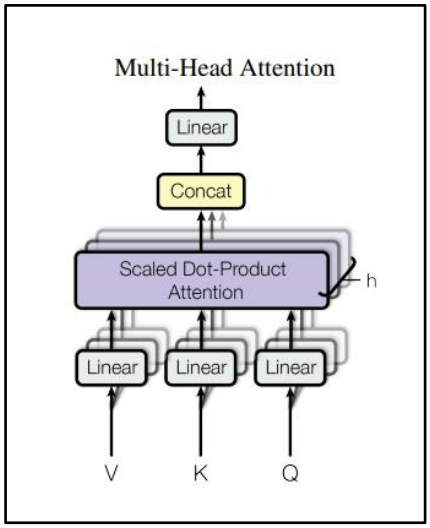


如图所示，这是 attention 函数的详细结构，也可用公式表达为：

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (1)$$

其中，$\sqrt{d_k}$为缩放因子，由 key 的维度d_k决定。此外，之所以采用点积是因为它能利用高度优化的矩阵乘法来实现计算（有效并行化）。

2°多头注意力：（整体模块）



如图所示，将 query、key 和 value 向量分别用不同的、所学到的线性映射层以h倍降维到d_k、d_k和d_v维，而不是直接用d_{model}维的 query、key 和 value 向量

来计算单个 attention 函数，具体公式如下：

$$MultiHead(Q,K,V) = Concat(head_1, head_2, \dots, head_h)W^O$$

$$where\ head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \qquad (2)$$

其中，$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}, W^O \in \mathbb{R}^{hd_{\text{v}} \times d_{\text{model}}}$；

如公式 (2)所示，基于每个映射模块的 query、key 和 value，通过并行执行 attention 函数，产生d_v维的输出值，然后将它们连接（Concat）并再次映射，来产生最终值。【实际上，就是多了一个能调参的超参数$\boldsymbol{h}$】

3°多头注意力模块的应用；

在编码器中：

包含了 self-attention 层，即在 self-attention 层中，所有的 key、value 和 query 来自相同的位置（即将输入复制成三份）。因此，在编码器中的每个位置都可以注意到编码器上一层的所有位置。

在解码器中：

在 Masked Multi-Head Attention 模块中，使用 mask 层来屏蔽所在计算的位置之后的位置信息，同时也采用 self-attention 层。

而在 Multi-Head Attention 层中，query 来自前一个解码器层，而 key 和 value 来自编码器的输出。

（3）位置前馈网络；

在上图中，可以注意到在每个编码器和解码器模块中都包含了一个完全连接的前馈网络。该前馈网络单独且相同地应用于每个位置，而它主要包括两个线性变换以及中间的一个$ReLU$激活，具体计算公式如下：

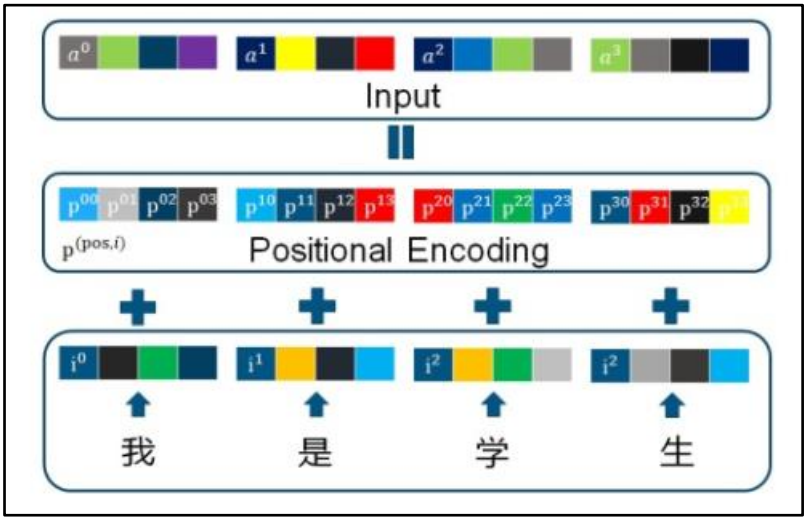
$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

实际上，它的主要目的是把输出结果映射到我们想要的语义空间中。

（4）位置编码；

由于该模型不包含循环和卷积，因此为了让模型能够利用序列的顺序，各模块的输入就必须加入序列中关于词符相对或者绝对位置的一些信息。为此，采用将“位置编码”添加到编码器和解码器堆栈底部的输入的方式。

且由于“位置编码”和输入的维度d_{model}相同，所以它们可以相加，如下图所示：



关于自注意力机制的分析：

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

如上表所示，通过比较 self-attention 与循环层和卷积层的各个方面，可以清楚地得出，self-attention 不仅在计算的总复杂度$O(n^2 d)$上优于其他结构，还能有效进行并行计算$O(1)$。（这也是自注意力机制的优势所在，因此如今受到研究者的追捧。）

实验细节与结果：

（1）优化算法；

使用 Adam 算法优化，其中$\beta 1 = 0.9, \beta 2 = 0.98$ 及$\epsilon = 10^{-9}$，并根据下述公式在训练过程中改变学习率$lrate$：

$$lrate = d_{model}^{-0.5} \cdot \min\left(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}\right)$$

其中，$warmup_{steps} = 4000$。

（2）正则化；

训练主要采用两种正则化：

1）Residual Dropout：将 dropout 应用到每个子层的输出中，$drop = 0.1$。

2）Label Smoothing：在训练过程中，使用 label smoothing 的值为$\epsilon ls = 0.1$。

尽管这让模型不易理解，但这有效提高了准确性和 BLEU 得分。

（3）结果；

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

从图中可以看到，Transformer 模型在 WMT 2014 的任务上取得了最优的分数。

总结：

Transformer 作为一个既 CNN 和 RNN 之后的新模型架构，有效解决了当时 RNN 所存在的无法有效并行计算的问题，同时它也在翻译任务上取得了更优异的结果。总之，这给当时的深度学习研究带来了新的可能。而如今，注意力机制也已被应用于多项其他研究之中。