注意力就是你所需的一切

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

主流的序列转导模型基于复杂的循环或卷积神经网络，其中包括一个编码器和一个解码器。表现最好的模型还通过注意力机制连接编码器和解码器。本文提出一种新的简单网络架构Transformer，完全基于注意力机制，完全免除递归和卷积。在两个机器翻译任务上的实验表明，这些模型在质量上优于其他模型，同时具有更强的并行性，训练时间明显减少。该模型在WMT 2014英德翻译任务上取得了28.4个BLEU，比现有的最佳结果(包括集成)提高了2个BLEU以上。在WMT 2014英法翻译任务中，该模型在8个gpu上训练3.5天后建立了新的单模型的最先进的BLEU分数为41.8，只占文献中最好模型训练成本的一小部分。Transformer可以很好地推广到其他任务，在大量和有限的训练数据下成功地将其应用于英语短语句法分析。

1 简介

循环神经网络，特别是长短期记忆[13]和门控循环[7]神经网络，在序列建模和转导问题(如语言建模和机器翻译)中已经成为最先进的方法[35, 2, 5]。此后，许多努力继续推动循环语言模型和编码器-解码器架构的边界[38, 24, 15]。

递归模型通常沿着输入和输出序列的符号位置进行因子计算。在计算时间内将位置与步骤对齐，它们生成一个隐藏状态序列h_t，作为前一个隐藏状态h_{t-1}的函数和位置的输入t。这

*同等贡献。列表顺序是随机的。Jakob提出用self-attention代替rnn，并开始评估这一想法。Ashish与Illia一起设计和实现了第一个Transformer模型，并至关重要地参与了这项工作的各个方面。Noam提出了尺度点积注意力、多头注意力和无参数的位置表示，成为几乎涉及每个细节的另一个人。Niki在我们的原始代码库和tensor2tensor中设计、实现、调优和评估了无数的模型变体。Llion还尝试了新的模型变体，负责我们的初始代码库，以及高效的推理和可视化。Lukasz和Aidan花了无数天的时间设计和实现tensor2tensor的各个部分，替换了我们早期的代码库，极大地改善了结果并极大地加速了我们的研究。

†工作时大脑处于谷歌。

‡在谷歌研究期间进行的工作。

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

种固有的顺序性排除了训练样本中的并行化，在序列长度较长的情况下，并行化变得至关重要，因为内存约束限制了样本之间的批处理。最近的工作通过因子分解技巧[21]和条件计算[32]在计算效率方面取得了显著的改进，同时也提高了后者情况下的模型性能。然而，顺序计算的基本约束仍然存在。

注意力机制已经成为各种任务中引人注目的序列建模和转导模型的组成部分，允许对依赖关系进行建模，而无需考虑它们在输入或输出序列中的距离[2, 19]。然而，除了极少数情况[27]，在所有情况下，这种注意力机制都与循环网络一起使用。

本文提出Transformer，一种避免递归的模型架构，完全依靠注意力机制来获取输入和输出之间的全局依赖关系。Transformer允许更多的并行化，并可以在8个P100 gpu上训练仅12个小时后达到翻译质量的新水平。

2 背景

减少顺序计算的目标也构成了扩展的神经GPU [16], ByteNet [18]和ConvS2S [9]的基础，所有这些都使用卷积神经网络作为基本构建块，并行计算所有输入和输出位置的隐藏表示。在这些模型中，将来自两个任意输入或输出位置的信号关联起来所需的操作数随着位置之间的距离而增长，对于ConvS2S是线性增长，对于ByteNet是对数增长。这使得学习遥远位置之间的依赖关系变得更加困难[12]。在Transformer中，这减少为恒定数量的操作，尽管以降低有效分辨率为代价，这是由于平均注意力加权位置，我们用多头注意力来抵消这种影响，如3.2节所述。

自注意力，有时称为内部注意力，是一种关联单个序列的不同位置的注意力机制，以便计算序列的表示。自注意力已成功应用于各种任务，包括阅读理解、抽象摘要、文本蕴涵和学习任务无关的句子表示[4, 27, 28, 22]。

端到端记忆网络基于循环注意力机制而不是序列对齐的递归，已被证明在简单语言问答和语言建模任务上表现良好[34]。

然而，Transformer是第一个完全依赖自注意力来计算其输入和输出表示的转导模型，而不使用序列对齐的rnn或卷积。在以下几节中，我们将描述Transformer，激发自注意力，并讨论其相对于[17, 18]和[9]等模型的优势。

3 模型架构

大多数具有竞争力的神经序列转导模型具有编码器-解码器结构[5, 2, 35]。在这里，编码器将符号表示的输入序列$(x_1, ..., x_n)$映射到连续表示序列$\mathbf{z} = (z_1, ..., z_n)$。给定$\mathbf{z}$，解码器然后生成符号的输出序列$(y_1, ..., y_m)$，每次一个元素。在每个步骤中，模型都是自回归的[10]，在生成下一个步骤时，将之前生成的符号作为额外的输入。

Transformer遵循这种整体架构，编码器和解码器使用堆叠的自注意力和逐点全连接层，分别如图1的左半部分和右半部分所示。

3.1 编码器和解码器堆栈

编码器： 编码器由一堆$N = 6$相同的层组成。每一层有两个子层。第一个是多头自注意力机制，第二个是简单的、按位置全连接的前馈网络。我们在每两个子层周围采用残差连接[11]，然后进行层归一化[1]。也就是说，每个子层的输出是$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$，其中$\mathrm{Sublayer}(x)$是子层自身实现的功能。为了促进这些残差连接，模型中的所有子层以及嵌入层产生维度$d_{\text{model}} = 512$的输出。

解码器： 解码器也由$N = 6$相同层的堆栈组成。除了每个编码器层中的两个子层外，解码器插入第三个子层，该子层对编码器堆栈的输出执行多头注意力。与编码器类似，我们在每个子层周围采用残差连接，然后进行层归一化。还修改了解码器堆栈中的自注意力子层，以防止位置关注后续位置。这种屏蔽，结合输出嵌入被一个位置偏移的事实，确保对位置i的预测只能依赖于位置小于i的已知输出。

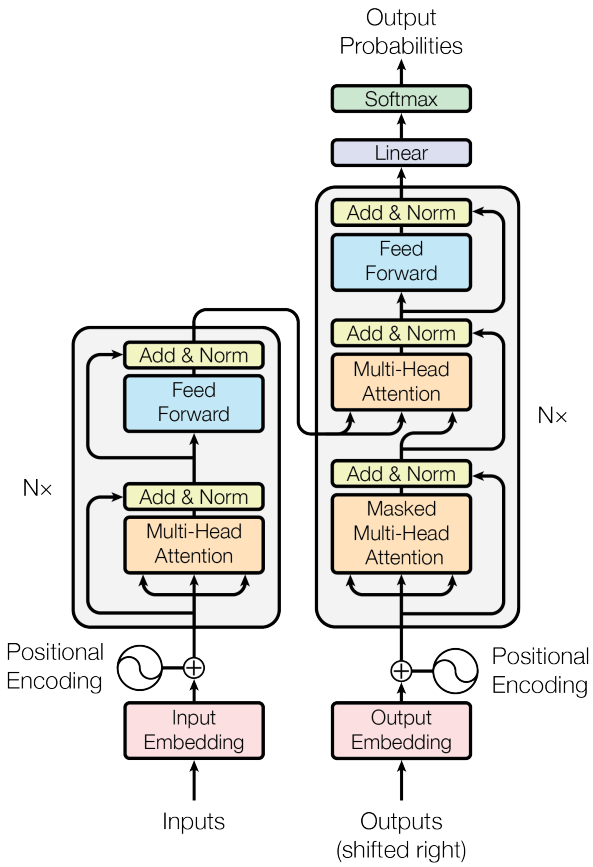

Figure 1: Transformer模型架构。

3.2 注意

注意力函数可以描述为将查询和一组键值对映射到输出，其中查询、键、值和输出都是向量。输出被计算为值的加权和，其中分配给每个值的权重是由查询与相应键的兼容性函数计算的。

3.2.1 缩放点积注意力

我们将这种特别的注意力称为"缩放点积注意力"(图2)。输入由维度d_k的查询和键以及维度d_v的值组成。我们计算查询与所有键的点积，每个键除以$\sqrt{d_k}$，并应用softmax函数获得值的权重。

在实践中，我们同时计算一组查询的注意力函数，打包成一个矩阵Q。键和值也打包成矩阵K和V。我们将输出矩阵计算为:

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

两个最常用的注意力函数是加性注意力[2]和点积(乘性)注意力。点积注意力与我们的算法相同，除了缩放因子$\frac{1}{\sqrt{d_k}}$。加性注意力使用具有单隐藏层的前馈网络计算兼容性函数。虽然两者在理论复杂度上相似，但点积注意力在实践中要快得多，空间效率更高，因为它可以使用高度优化的矩阵乘法代码来实现。

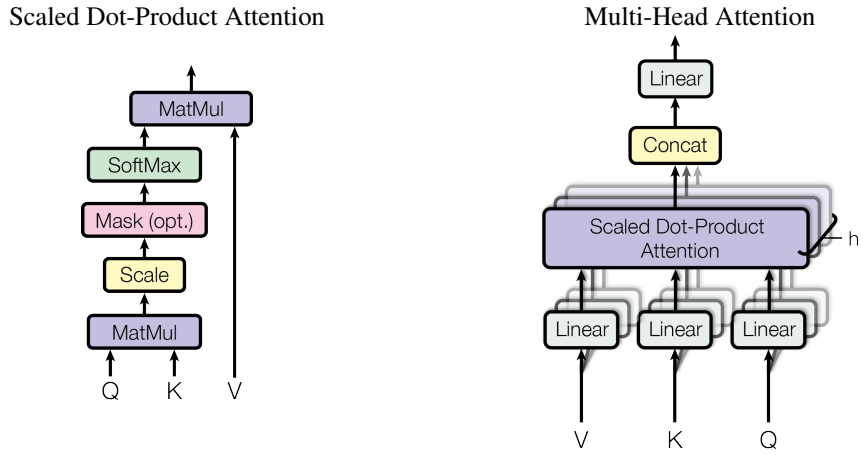

Figure 2: (左)缩放点积注意力。(右)多头注意力由多个并行运行的注意力层组成。

虽然对于较小的d_k值，两种机制的表现类似，但对于较大的d_k[3]值，添加注意力的表现优于点积注意力。我们怀疑，对于d_k的大值，点积在量级上增长很大，将softmax函数推到其梯度极小的区域[4]。为了抵消这种影响，我们将点积按$\frac{1}{\sqrt{d_k}}$进行缩放。

3.2.2 多头注意

与使用d_{model}维度的键、值和查询执行单个注意力函数不同，我们发现用不同的时间线性投影查询、键和值h是有益的，分别学习到d_k、d_k和d_v维度的线性投影。然后，在这些投影版本的查询、键和值上，我们并行执行注意力函数，产生d_v维的输出值。这些值被连接起来并再次投影，从而得到最终的值，如图2所示。

多头注意力允许模型在不同位置共同关注来自不同表示子空间的信息。如果只有一个注意力头，平均会抑制这种情况。

$$\begin{aligned}\text{MultiHead}(Q,K,V) &= \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O \\ \text{where head}_\text{i} &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

其中投影是参数矩阵$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$，$W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$，$W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$和$W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$。

在这项工作中，我们采用了$h = 8$并行注意力层(parallel attention layers，简称heads)。对于每一个，我们使用$d_k = d_v = d_{\text{model}}/h = 64$。由于每个头的维度降低，总的计算成本与全维度的单头注意力相似。

3.2.3 注意力在模型中的应用

Transformer以三种不同的方式使用多头注意力。

- 在“编码器-解码器注意力”层中，查询来自前一个解码器层，记忆键和值来自编码器的输出。这允许解码器中的每个位置都参与输入序列中的所有位置。这模仿了序列到序列模型(如[38, 2, 9])中典型的编码器-解码器注意力机制。

[4]为了说明点积变大的原因，假设q和k的分量是均值为0，方差为1的独立随机变量。那么它们的点积$q \cdot k = \sum_{i=1}^{d_k} q_i k_i$的均值为0，方差为$d_k$。

- 编码器包含自注意力层。在self-attention层中，所有的键、值和查询都来自同一个地方，在这种情况下，是编码器中前一层的输出。编码器中的每个位置都可以处理编码器前一层的所有位置。
- 类似地，解码器中的自注意力层允许解码器中的每个位置关注解码器中的所有位置，直到并包括该位置。我们需要防止解码器中向左的信息流，以保持自回归特性。我们通过屏蔽(设置为$-\infty$) softmax输入中对应于非法连接的所有值，在缩放的点积注意力中实现这一点。参见图2。

3.3 位置前馈网络

除了attention子层之外，编码器和解码器中的每一层都包含一个完全连接的前馈网络，该网络分别相同地应用于每个位置。它由两个线性变换组成，中间有一个ReLU激活函数。

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

虽然在不同位置上的线性变换是相同的，但每一层使用不同的参数。另一种描述方法是将其描述为两个卷积核大小为1的卷积。输入和输出的维度是$d_{\text{model}} = 512$，内层的维度是$d_{ff} = 2048$。

3.4 嵌入和Softmax

与其他序列转导模型类似，我们使用学习的嵌入将输入标记和输出标记转换为维度d_{model}的向量。我们还使用通常学习的线性变换和softmax函数将解码器输出转换为预测的下一个token概率。在我们的模型中，我们在两个嵌入层和pre-softmax线性变换之间共享相同的权重矩阵，类似于[30]。在嵌入层中，我们将这些权重乘以$\sqrt{d_{\text{model}}}$。

3.5 位置编码

由于我们的模型不包含递归和卷积，为了让模型利用序列的顺序，我们必须注入一些关于单词在序列中相对或绝对位置的信息。为此，我们在编码器和解码器堆栈的底部向输入嵌入添加"位置编码"。位置编码与嵌入具有相同的维度d_{model}，因此可以将两者相加。有许多位置编码的选择，学习和固定[9]。

在这项工作中，我们使用不同频率的正弦函数和余弦函数:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

其中pos是位置，i是尺寸。也就是说，位置编码的每个维度对应一个正弦信号。波长从2π到$10000 \cdot 2\pi$呈几何级数。我们选择这个函数，因为我们假设它将允许模型轻松地学习通过相对位置参加，因为对于任何固定偏移k, PE_{pos+k}可以表示为PE_{pos}的线性函数。

我们还尝试使用学习的位置嵌入[9]，并发现两个版本产生了几乎相同的结果(参见表3行(E))。我们选择正弦版本，因为它可以允许模型推断出比训练期间遇到的序列长度更长的序列。

4 为什么是自我注意力

在本节中，我们将自注意力层的各个方面与递归层和卷积层进行比较，递归层和卷积层通常用于将一个变长符号表示序列$(x_1, ..., x_n)$映射到另一个等长序列$(z_1, ..., z_n)$和$x_i, z_i \in \mathbb{R}^d$，例如典型序列转导编码器或解码器中的隐藏层。激励我们使用自我注意力我们考虑三个需求。

一个是每层的总计算复杂度。另一个是可并行化的计算量，由所需的最小顺序操作数来衡量。

第三是网络中长程依赖关系之间的路径长度。学习长程依赖关系是许多序列转导任务的关键挑战。影响学习这种依赖关系的能力的一个关键因素是信号在网络中必须遍历的前向和

Table 1: 不同层类型的最大路径长度、每层复杂度和最小顺序操作数。n是序列长度，d是表示维度，k是卷积的内核大小，r是受限自注意力中的邻域大小。

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

后向路径的长度。输入和输出序列中任何位置组合之间的这些路径越短，就越容易学习长程依赖关系[12]。因此，本文还比较了由不同层类型组成的网络中任意两个输入和输出位置之间的最大路径长度。

如表1所述，自注意力层将所有位置与固定数量的顺序执行操作连接起来，而循环层需要$O(n)$顺序操作。在计算复杂度方面，当序列长度n小于表示维度d时，自注意力层比循环层更快，这是机器翻译中最先进的模型使用的句子表示最常见的情况，如词块[38]和字节对[31]表示。为了提高涉及非常长的序列的任务的计算性能，自注意力可以限制为只考虑以各自输出位置为中心的输入序列中大小为r的邻域。这将增加到$O(n/r)$的最大路径长度。我们计划在未来的工作中进一步研究这种方法。

具有内核宽度$k < n$的单个卷积层不会连接所有的输入和输出位置对。这样做需要一个$O(n/k)$卷积层的堆栈，在连续内核的情况下，或$O(log_k(n))$在扩张卷积[18]的情况下，增加网络中任何两个位置之间最长路径的长度。卷积层通常比循环层昂贵，是循环层的k倍。然而，可分离卷积[6]将复杂度大大降低到$O(k \cdot n \cdot d + n \cdot d^2)$。然而，即使是$k = n$，可分离卷积的复杂性等于自注意力层和按点前馈层的组合，这是我们在我们的模型中采用的方法。

作为附带好处，自注意力可以产生更可解释的模型。我们将检查模型的注意力分布，并在附录中给出和讨论示例。个体注意力头不仅清楚地学会了执行不同的任务，而且许多注意力头似乎表现出与句子的句法和语义结构相关的行为。

5 培训

本节描述了我们模型的训练机制。

5.1 训练数据和批处理

我们在标准的WMT 2014英德数据集上进行了训练，该数据集由大约450万个句子对组成。句子使用字节对编码[3]，它有一个共享的源-目标词汇表约37000个token。对于英语-法语，我们使用了更大的WMT 2014英语-法语数据集，包括3600万个句子，并将token分割为3.2万个单词词汇表[38]。句子对按近似序列长度进行批处理;每个训练批次包含一组句子对，其中大约包含25000个源标记和25000个目标标记。

5.2 硬件和时间表

我们在一台拥有8个NVIDIA P100 gpu的机器上训练我们的模型。对于使用本文中描述的超参数的基本模型，每个训练步骤大约需要0.4秒。我们对基础模型进行了总共10万步或12小时的训练。对于我们的大型模型，(在表3的底线上描述)，步长时间是1.0秒。大型模型训练了30万步(3.5天)。

5.3 优化器

我们使用Adam优化器[20]与$\beta_1 = 0.9$, $\beta_2 = 0.98$和$\epsilon = 10^{-9}$。我们在训练过程中根据下面的公式改变学习率:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \tag{3}$$

Table 2: Transformer在英语到德语和英语到法语的newstest2014测试中取得了比之前最先进的模型更好的BLEU分数，而训练成本只占很小一部分。

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**	$2.3 \cdot 10^{19}$	

这对应于第一个$warmup_steps$训练步骤的学习率线性增加，然后与步骤数的逆平方根成比例地减少它。我们使用$warmup_steps = 4000$。

5.4 正则化

在训练过程中，我们使用了三种类型的正则化。

剩余漏　我们将dropout [33]应用于每个子层的输出，然后将其添加到子层输入并归一化。此外，我们将dropout应用于编码器和解码器堆栈中的嵌入和位置编码的总和。对于基础模型，我们使用$P_{drop} = 0.1$。

标签平滑　在训练过程中，我们使用了值的标签平滑$\epsilon_{ls} = 0.1$[36]。这伤害了困惑，因为模型学会了更不确定，但提高了准确性和BLEU分数。

6 结果

6.1 机器翻译

在WMT 2014英语到德语翻译任务中，big transformer模型(表2中的transformer (big))比之前报告的最好模型(包括集成)的表现超过2.0 BLEU，建立了一个新的最先进的BLEU分数28.4。该模型的配置列在表3的底部。培训在8 P100 gpu上花费了3.5天。即使我们的基础模型也超过了所有之前发布的模型和集成，而训练成本只是任何竞争模型的一小部分。

在WMT 2014英语到法语翻译任务中，我们的大模型达到了41.0的BLEU分数，超过了之前发表的所有单个模型，而低于之前最先进模型的训练成本$1/4$。为英语-法语训练的Transformer (big)模型使用的辍学率为$P_{drop} = 0.1$，而不是0.3。

对于基础模型，我们使用一个通过平均最后5个检查点得到的单一模型，每10分钟编写一次。对于大型模型，我们对最后20个检查点取平均值。我们使用光束搜索，光束大小为4，长度惩罚为$\alpha = 0.6$[38]。这些超参数是在开发集上进行实验后选择的。我们在推理期间将最大输出长度设置为输入长度+ 50，但在可能的情况下提前终止[38]。

表2总结了我们的结果，并将我们的翻译质量和培训成本与文献中的其他模型架构进行了比较。我们通过乘以训练时间、使用的GPU数量和每个GPU持续单精度浮点容量的估计来估计用于训练模型的浮点操作数量[5]。

Table 3: Transformer架构的变体。未列出的值与基本模型中的值相同。所有指标都在英语-德语翻译开发集newstest2013上。根据我们的字节对编码，列出的困惑度是每个单词的，不应该与每个单词的困惑度进行比较。

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

6.2 模型变化

为了评估Transformer不同组件的重要性，我们以不同的方式改变了基础模型，测量了开发集newstest2013上英德翻译性能的变化。我们使用了上一节中描述的波束搜索，但没有使用检查点平均。我们将这些结果显示在表3中。

在表3 rows (A)中，我们改变attention头的数量以及attention键和值维度，保持计算量不变，如3.2.2节所述。虽然单头注意力比最佳设置差0.9 BLEU，但过多的头也会导致质量下降。

在表3 rows (B)中，我们观察到减少注意力密钥大小d_k会损害模型质量。这表明确定兼容性并不容易，比点积更复杂的兼容性函数可能是有益的。我们在(C)和(D)行中进一步观察到，正如预期的那样，模型越大越好，dropout在避免过拟合方面非常有用。在行(E)中，我们将正弦位置编码替换为学习到的位置嵌入[9]，并观察到与基本模型几乎相同的结果。

6.3 英语句法成分分析

为了评估Transformer是否可以泛化到其他任务，我们在英语短语句法分析上进行了实验。这项任务提出了特定的挑战:输出受到强烈的结构约束，并且明显比输入长。此外，RNN序列到序列模型还未能在小数据体系中取得最先进的结果[37]。

我们用宾夕法尼亚树库[25]的华尔街日报(WSJ)部分的$d_{model} = 1024$训练了一个4层transformer，大约40K训练句子。我们还在半监督环境中训练它，使用来自大约1700万个句子的更大的高置信度和BerkleyParser语料库[37]。我们为WSJ only设置使用了16K个代币的词汇，为半监督设置使用了32K个代币的词汇。

我们只进行了少量实验来选择dropout、注意力和残差(5.4)、学习率和波束大小在section 22开发集上，所有其他参数都保持不变，从英语到德语基本翻译模型。在推理过程中，我们将最大输出长度增加到输入长度+ 300。对于WSJ和半监督设置，我们都使用了21和$\alpha = 0.3$的光束大小。

[5]我们对K80、K40、M40和P100分别使用2.8、3.7、6.0和9.5 TFLOPS。

Table 4: Transformer可以很好地推广到英语句法分析中(结果见《华尔街日报》第23节)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

我们在表4中的结果表明，尽管缺乏特定任务的调优，我们的模型表现得惊人地好，产生了比所有之前报道的模型更好的结果，除了循环神经网络语法[8]。

与RNN序列到序列模型[37]相比，即使只在WSJ的40K句子训练集上训练，Transformer的表现也优于BerkeleyParser [29]。

7 结论

本文提出Transformer，第一个完全基于注意力的序列转导模型，用多头自注意力替换了编码器-解码器架构中最常用的循环层。

对于翻译任务，Transformer的训练速度明显快于基于循环或卷积层的架构。在WMT 2014英语到德语和WMT 2014英语到法语的翻译任务中，我们取得了新的技术水平。在前一个任务中，我们最好的模型甚至超过了所有之前报告的集成。

我们对基于注意力的模型的未来感到兴奋，并计划将它们应用于其他任务。计划将Transformer扩展到涉及文本以外的输入和输出模态的问题，并研究局部的、受限的注意力机制，以有效处理图像、音频和视频等大量输入和输出。我们的另一个研究目标是减少生成的顺序性。

我们用于训练和评估模型的代码可以在`https://github.com/tensorflow/tensor2tensor`上找到。

致谢　我们感谢Nal Kalchbrenner和Stephan Gouws 他们富有成效的评论、纠正和启发。

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.

[10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841. ACL, August 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[16] Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.

[17] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *International Conference on Learning Representations (ICLR)*, 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099v2*, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. *arXiv preprint arXiv:1703.10722*, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[26] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159. ACL, June 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In *Empirical Methods in Natural Language Processing*, 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 433–440. ACL, July 2006.

[30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 434–443. ACL, August 2013.

注意力可视化

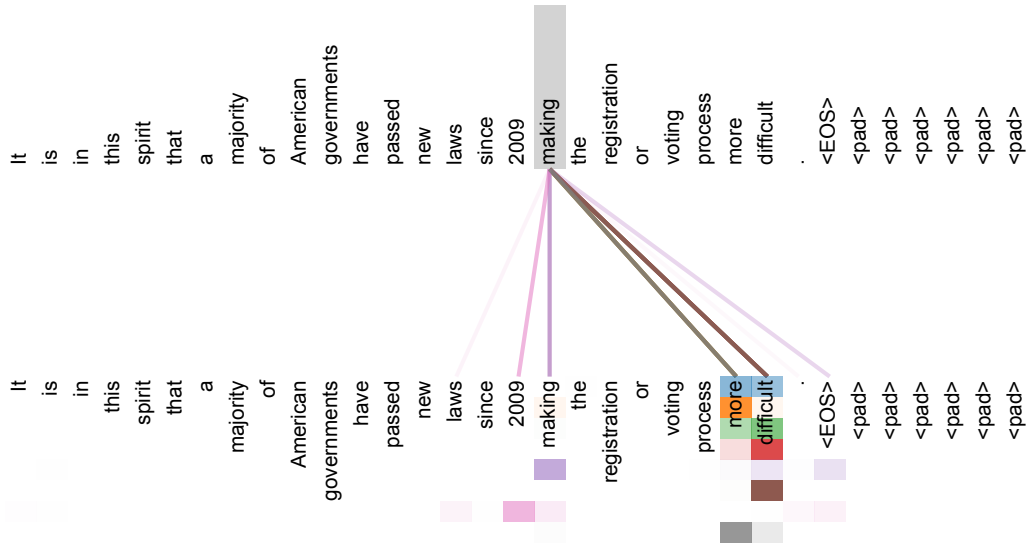

Figure 3: 注意力机制遵循6中的第5层编码器自注意力中的长距离依赖关系的示例。许多注意力头关注动词“making”的一个遥远的依赖关系，完成短语“making...更困难”。这里只关注“making”一词。不同的颜色代表不同的头。彩色效果最好。

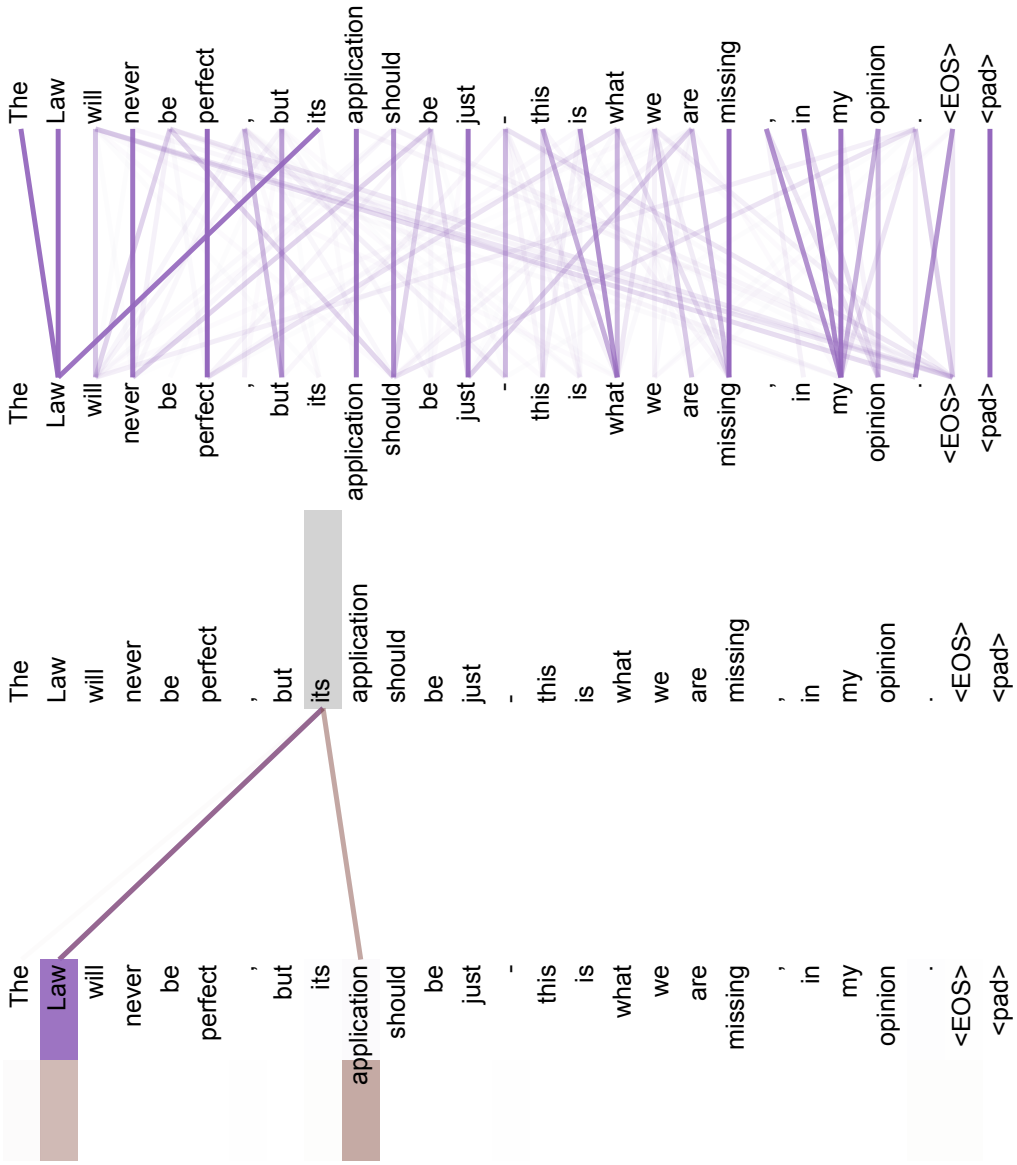

Figure 4: 两个注意头，也在第5层，显然参与回指消解。上图:全神贯注于第5头。下图:仅从单词“its”中分离出注意头5和6。请注意，这个词的关注度非常高。

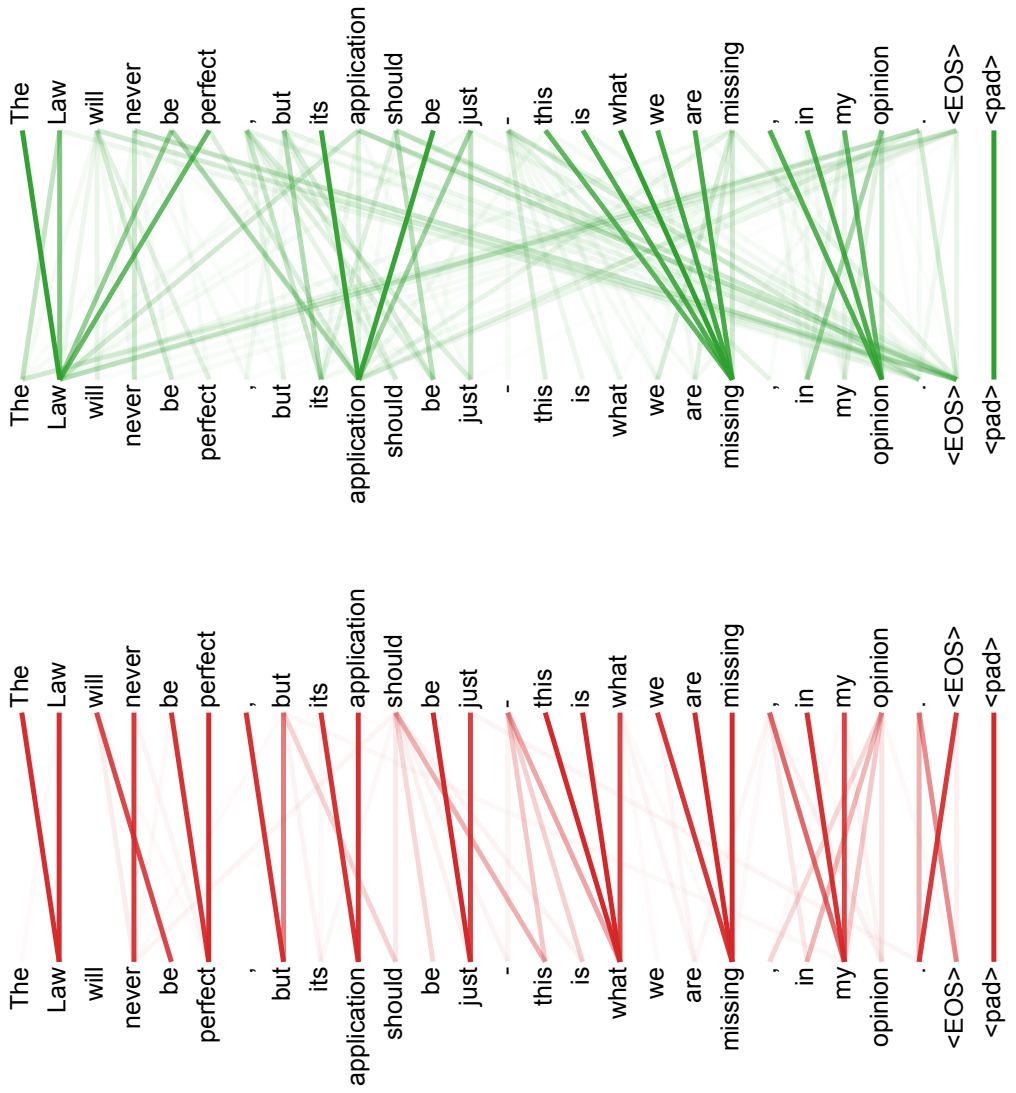

Figure 5: 许多注意力头表现出的行为似乎与句子的结构有关。我们在上面给出了两个这样的例子，来自6中的第5层的编码器自注意力的两个不同头部。大脑显然学会了执行不同的任务。