注意力就是您所需的一切

阿希什·瓦斯瓦尼* 谷歌大脑 avaswani@google.com

诺姆·沙泽尔* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

雅各布·乌什科雷特* 谷歌研究院 usz@google.com

Llion Jones* Google Research llion@google.com

艾丹·N·戈麦斯* † 多伦多大学 aidan@cs.toronto.edu

ukasz Kaiser* Google Brain lukaszkaiser@google.com

伊利亚·波洛苏欣* ‡ illia.polosukhin@gmail.com

摘要

主流的序列转换模型基于复杂的循环或卷积神经网络，这些网络包含编码器和解码器。性能最佳的模型还通过注意力机制连接编码器和解码器。我们提出了一种新的简单网络架构——Transformer，它完全基于注意力机制，彻底摒弃了循环和卷积结构。在两个机器翻译任务上的实验表明，这些模型在质量上更优，同时具有更高的并行化能力，且训练所需时间显著减少。我们的模型在WMT 2014英德翻译任务中取得了28.4 BLEU的分数，相比现有最佳结果（包括集成模型）提高了超过2 BLEU。在WMT 2014英法翻译任务中，我们的模型在8个GPU上训练3.5天后，取得了41.8 BLEU的单模型最优成绩，其训练成本仅为文献中最佳模型的一小部分。我们通过将Transformer成功应用于英语成分句法分析（无论训练数据规模大小），证明了该模型能良好泛化至其他任务。

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

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

1 引言

循环神经网络，尤其是长短期记忆网络[13]和门控循环神经网络[7]，已在序列建模和转换问题（如语言建模和机器翻译）中被牢固确立为最先进的方法[35, 2, 5]。此后，大量研究持续推动着循环语言模型和编码器-解码器架构的发展边界[38, 24, 15]。

循环模型通常沿着输入和输出序列的符号位置分解计算。通过将位置与计算时间步骤对齐，它们生成一系列隐藏状态 h_t，该状态是前一个隐藏状态 h_{t-1} 和位置 t 的输入的函数。这种固有的顺序性质阻碍了训练示例内的并行化，这在序列较长时变得至关重要，因为内存限制制约了跨示例的批处理。最近的研究通过分解技巧[21]和条件计算[32]显著提升了计算效率，同时后者还提升了模型性能。然而，顺序计算的根本限制依然存在。

注意力机制已成为各种任务中引人注目的序列建模和转换模型不可或缺的一部分，它允许对依赖关系进行建模，而无需考虑其在输入或输出序列中的距离[2, 19]。然而，除少数情况外[27]，此类注意力机制通常与循环网络结合使用。

在本工作中，我们提出了Transformer模型架构，它摒弃了循环结构，完全依赖注意力机制来捕捉输入与输出之间的全局依赖关系。Transformer能够实现更显著的并行化，在仅使用八块P100 GPU训练十二小时后，即可达到翻译质量的新最优水平。

2 背景

减少序列计算的目标也构成了扩展神经GPU[16]、ByteNet[18]和ConvS2S[9]的基础，这些模型均使用卷积神经网络作为基本构建模块，并行计算所有输入和输出位置的隐藏表示。在这些模型中，关联两个任意输入或输出位置信号所需的操作次数随位置间距离增长而增加——ConvS2S呈线性增长，ByteNet呈对数增长。这使得学习远距离位置间的依赖关系变得更加困难[12]。在Transformer中，这一操作被减少至常数级别，尽管代价是因对注意力加权位置进行平均而降低了有效分辨率，我们通过3.2节所述的多头注意力机制来抵消这一影响。

自注意力，有时也称为内部注意力，是一种将单个序列的不同位置关联起来以计算该序列表示的注意力机制。自注意力已在多种任务中成功应用，包括阅读理解、抽象摘要、文本蕴含以及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，而非序列对齐的循环结构，并已在简单语言问答和语言建模任务中展现出良好性能[34]。

然而，据我们所知，Transformer是首个完全依赖自注意力机制来计算输入和输出表示的转换模型，无需使用序列对齐的RNN或卷积网络。在接下来的章节中，我们将描述Transformer架构，阐释自注意力的原理，并讨论其相较于[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换模型都采用编码器-解码器结构[5, 2, 35]。其中，编码器将输入符号表示序列($x_1, ..., x_n$)映射为连续表示序列$\mathbf{z} = (z_1, ..., z_n)$。在给定$\mathbf{z}$的条件下，解码器逐步生成输出符号序列($y_1, ..., y_m$)，每次生成一个元素。该模型在每一步都采用自回归方式[10]，在生成下一个符号时会将已生成的符号作为额外输入。

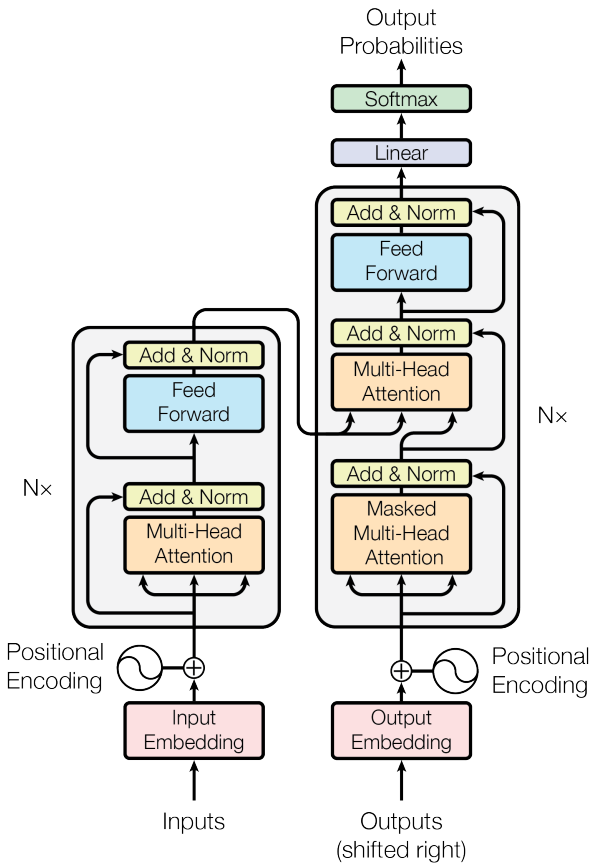

图1：Transformer模型架构。

Transformer遵循这一整体架构，在编码器和解码器中均采用堆叠的自注意力层和逐点全连接层，分别如图1左右两部分所示。

3.1 编码器与解码器堆栈

编码器：编码器由N =6个相同的层堆叠而成。每层包含两个子层。第一层是多头自注意力机制，第二层是简单的位置级全连接前馈网络。我们在每个子层周围采用残差连接[11]，随后进行层归一化[1]。即每个子层的输出为LayerNorm(x + Sublayer(x))，其中Sublayer(x)是该子层自身实现的函数。为便于残差连接，模型中所有子层及嵌入层均生成维度为d_{model} =512的输出。

解码器：解码器同样由N =个相同的层堆叠而成。除了每个编码器层中的两个子层外，解码器还插入了第三个子层，该子层对编码器堆栈的输出执行多头注意力操作。与编码器类似，我们在每个子层周围采用残差连接，并进行层归一化。我们还修改了解码器堆栈中的自注意力子层，以防止位置关注到后续位置。这种掩码机制，加上输出嵌入向右偏移一个位置的事实，确保了对位置i的预测只能依赖于小于i的已知输出位置。

3.2 注意力机制

注意力函数可以描述为将一个查询和一组键值对映射到一个输出，其中查询、键、值和输出都是向量。输出被计算为加权和

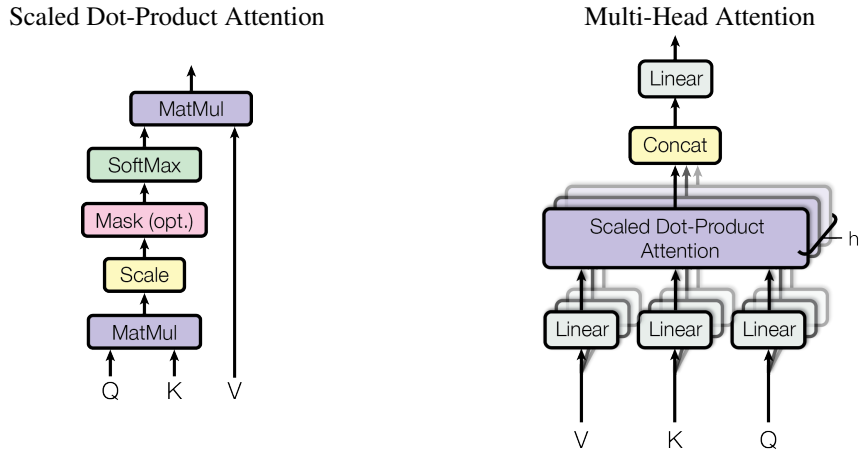

图2：（左）缩放点积注意力。（右）多头注意力由多个并行运行的注意力层组成。

其中每个值的权重由查询与对应键的兼容性函数计算得出。

3.2.1 缩放点积注意力

我们将我们特别的注意力称为“缩放点积注意力”（图2）。输入包括维度为d_k的查询和键，以及维度为d_v的值。我们计算查询与所有键的点积，将每个结果除以$\sqrt{d_k}$，然后应用softmax函数来获得值的权重。

在实践中，我们同时计算一组查询的注意力函数，将它们打包成矩阵 Q。键和值也分别打包成矩阵 K 和 V。我们通过以下方式计算输出矩阵：

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

两种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 {v*}。加性注意力使用具有单个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，但在实践中点积注意力更快且空间效率更高，因为它可以通过高度优化的矩阵乘法代码实现。

当d_k值较小时，两种机制表现相似；但对于较大的d_k值，加性注意力机制的表现优于未缩放的点积注意力机制[3]。我们推测，当d_k值较大时，点积的幅值会变得很大，从而将softmax函数推入梯度极小的区域[4]。为了抵消这种影响，我们将点积缩放为$\frac{1}{\sqrt{d_k}}$。

3.2.2 多头注意力机制

我们并没有使用单一的注意力函数来处理d_{model}维度的键、值和查询，而是发现将查询、键和值分别通过不同的、可学习的线性投影h次，线性投影到d_k、d_k和d_v维度更为有效。然后，我们对这些投影后的查询、键和值并行执行注意力函数，从而得到d_v维度的

[4]To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k.

输出值。这些值被连接起来并再次投影，得到最终的值，如图2所示。

多头注意力机制允许模型在不同位置同时关注来自不同表示子空间的信息。仅使用单一注意力头时，平均操作会抑制这种能力。

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

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 $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$。

在本工作中，我们采用了$h = 8$个并行注意力层（或称注意力头）。每个注意力头使用 $d_k = d_v = d_\text{model}/h = 64$维表示。由于每个头的维度降低，总计算成本与使用全维度的单头注意力模型相近。

3.2.3 注意力机制在我们模型中的应用

Transformer在三个不同的方面使用了多头注意力机制：

- 在“编码器-解码器注意力”层中，查询来自前一个解码器层，而记忆键和值则来自编码器的输出。这使得解码器中的每个位置都能关注输入序列中的所有位置。这模仿了序列到序列模型中典型的编码器-解码器注意力机制，例如[38, 2, 9]。

- 编码器包含自注意力层。在自注意力层中，所有的键、值和查询都来自同一个地方，即编码器中前一层的输出。编码器中的每个位置都可以关注到编码器前一层的所有位置。

- 同样，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所有直到并包括该位置的位置。我们需要防止解码器中的信息向左流动，以保持自回归特性。我们通过在缩放点积注意力内部，将softmax输入中对应非法连接的所有值屏蔽（设置为$-\infty$）来实现这一点。参见图2。

3.3 位置前馈网络

除了注意力子层之外，我们的编码器和解码器中的每个层都包含一个全连接前馈网络，该网络分别且相同地应用于每个位置。它由两个线性变换组成，中间通过ReLU激活函数连接。

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

虽然线性变换在不同位置上是相同的，但它们在不同层之间使用不同的参数。另一种描述方式是将其视为两个核大小为1的卷积。输入和输出的维度为$d_\text{model} = 512$，而内层的维度为 $d_{ff} = 2048$。

3.4 嵌入与Softmax

与其他序列转导模型类似，我们使用学习到的嵌入将输入标记和输出标记转换为维度为 d_model的向量。我们还使用常规的学习线性变换和softmax函数将解码器输出转换为预测的下一个标记概率。在我们的模型中，两个嵌入层和softmax前的线性变换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$\sqrt{d_\text{model}}$。

表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为序列长度，d为表示维度，k为卷积核大小，r为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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 位置编码

由于我们的模型不包含递归和卷积，为了让模型能够利用序列的顺序信息，我们必须注入一些关于序列中标记的相对或绝对位置的信息。为此，我们在编码器和解码器堆栈底部的输入嵌入中添加了“位置编码”。位置编码的维度与嵌入的维度相同，均为d_{model}，因此两者可以相加。位置编码有多种选择，包括可学习的和固定的[9]。

在本工作中，我们使用了不同频率的正弦和余弦函数：

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

其中 pos 表示位置，i 表示维度。也就是说，位置编码的每个维度都对应一个正弦曲线。其波长构成从 2π 到 $10000 \cdot 2\pi$ 的几何级数。我们选择该函数是因为我们假设，对于任意固定偏移量 k，PE_{pos+k} 都可以表示为 PE_{pos} 的线性函数，这将使模型能够轻松学习通过相对位置进行注意力聚焦。

我们还尝试了使用学习到的位置嵌入[9]，发现两个版本产生了几乎相同的结果（见表3第(E)行）。我们选择了正弦版本，因为它可能使模型能够外推到比训练时遇到的序列长度更长的序列。

4 为何使用自注意力

在本节中，我们将自注意力层的各个方面与常用于将一个可变长度的符号表示序列（$x_1, ..., x_n$）映射到另一个等长序列（$z_1, ..., z_n$）的循环层和卷积层进行比较，其中涉及 $x_i, z_i \in \mathbb{R}^d$，例如典型序列转导编码器或解码器中的隐藏层。为了说明我们采用自注意力的动机，我们考虑了三个关键需求。

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

第三是网络中长距离依赖之间的路径长度。学习长距离依赖是许多序列转导任务中的关键挑战。影响学习此类依赖能力的一个关键因素是前向和反向信号在网络中必须穿越的路径长度。输入和输出序列中任意位置组合之间的路径越短，学习长距离依赖就越容易[12]。因此，我们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所示，自注意力层以恒定次数的顺序执行操作连接所有位置，而循环层需要$O(n)$次顺序操作。在计算复杂度方面，当序列长度

长度 n 小于表示维度 d，这在机器翻译中最先进的模型所使用的句子表示中最为常见，例如词片[38]和字节对[31]表示。为了提高涉及非常长序列的任务的计算性能，自注意力可以限制为仅考虑输入序列中围绕相应输出位置的大小为 r 的邻域。这将使最大路径长度增加到 $O(n/r)$。我们计划在未来的工作中进一步研究这种方法。

一个卷积核宽度为$k < n$的单一卷积层并不能连接所有输入和输出位置。对于连续卷积核，需要堆叠$O(n/k)$个卷积层才能实现；而对于空洞卷积[18]，则需要$O(log_k(n))$层，这会增加网络中任意两个位置间最长路径的长度。通常，卷积层的计算成本比循环层高出k倍。然而，可分离卷积[6]将计算复杂度显著降低至$O(k \cdot n \cdot d + n \cdot d^2)$。但即使采用$k = n$，可分离卷积的复杂度仍相当于自注意力层与逐点前馈层的组合复杂度，这也正是我们模型中采用的方法。

作为额外的优势，自注意力机制能够产生更具可解释性的模型。我们在附录中检查了模型的注意力分布，并展示和讨论了相关示例。不仅单个注意力头明显学会了执行不同的任务，许多注意力头还表现出与句子的句法和语义结构相关的行为。

5 训练

本节描述了我们模型的训练机制。

5.1 训练数据与批处理

我们在标准的WMT 2014英德数据集上进行了训练，该数据集包含约450万个句子对。句子采用字节对编码[3]进行编码，共享的源-目标词汇表约有37000个标记。对于英法翻译，我们使用了规模更大的WMT 2014英法数据集，包含3600万个句子，并将标记分割为32000个词片词汇表[38]。句子对根据近似序列长度进行批量组合。每个训练批次包含一组句子对，其中约含25000个源语言标记和25000个目标语言标记。

5.2 硬件与时间安排

我们在配备8块NVIDIA P100 GPU的单台机器上训练模型。对于采用本文所述超参数的基础模型，每个训练步骤耗时约0.4秒。基础模型共训练10万步，总时长12小时。对于大型模型（参数详见表3末行），单步训练时间为1.0秒。大型模型共训练30万步，耗时3.5天。

5.3 优化器

我们使用了Adam优化器[20]，其参数为$\beta_1 = 0.9$、$\beta_2 = 0.98$和$\epsilon = 10^{-9}$。在训练过程中，我们根据以下公式调整学习率：

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

这对应于在前$warmup_steps$个训练步骤中线性增加学习率，之后按步骤数的平方根倒数比例降低学习率。我们使用了$warmup_steps = 4000$。

5.4 正则化

我们在训练过程中采用了三种正则化方法：

表2：Transformer在英德和英法newstest2014测试中以更低的训练成本获得了比以往最先进模型更优的BLEU分数。

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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

残差丢弃 我们在每个子层的输出上应用丢弃[33]，然后将其添加到子层输入并进行归一化。此外，我们在编码器和解码器堆栈中，对嵌入向量和位置编码的和也应用丢弃。对于基础模型，我们使用的丢弃率为P_{drop} =0.1。

在训练过程中，我们采用了值为 $\epsilon_{ls} = 0.1$ 的标签平滑技术[36]。这会增加困惑度，因为模型学会了更加不确定，但能提高准确率和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，大型Transformer模型（表2中的Transformer (big)）以超过2.0 BLEU的优势超越了先前报道的最佳模型（包括集成模型），创下了28.4 BLEU的最新最优成绩。该模型的配置列于表3最末行。训练在8块P100 GPU上耗时3.5天。即使我们的基础模型也超越了所有先前发布的模型及集成模型，而训练成本仅为任何竞争模型的一小部分。

在WMT 2014英语到法语翻译任务中，我们的大型模型取得了41.0的BLEU分数，超越了所有先前发布的单一模型，且训练成本不到先前最先进模型的1/4。用于英语到法语训练的Transformer（大型）模型使用了P_{drop} =0.1的dropout率，而非0.3。

对于基础模型，我们使用通过平均最后5个检查点得到的单一模型，这些检查点以10分钟为间隔保存。对于大型模型，我们平均了最后20个检查点。我们采用束搜索方法，束宽为4，长度惩罚设为α =0.6 [38]。这些超参数是在开发集上经过实验后确定的。在推理过程中，我们将最大输出长度设置为输入长度+50，但会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训练成本与文献中的其他模型架构进行了比较。我们通过将训练时间、使用的GPU数量以及每个GPU的持续单精度浮点运算能力估计值相乘，来估算训练模型所使用的浮点运算次数[5]。

6.2 模型变体

为了评估Transformer不同组件的重要性，我们以多种方式调整基础模型，测量其在英语到德语翻译任务上性能的变化。

[5]We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

表3：Transformer架构的变体。未列出的值与基础模型相同。所有指标均基于英语到德语的翻译开发集newstest2013。列出的困惑度按我们的字节对编码以每词片为单位计算，不应与每词困惑度进行比较。

	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

开发集，newstest2013。我们使用了上一节所述的束搜索方法，但未进行检查点平均。我们在表3中展示了这些结果。

在表3的(A)行中，我们按照第3.2.2节的描述，在保持计算量不变的情况下，调整注意力头的数量以及注意力键与值的维度。虽然单头注意力比最佳设置低0.9 BLEU，但注意力头过多时质量也会下降。

在表3的(B)行中，我们观察到减小注意力键尺寸d_k会损害模型质量。这表明确定兼容性并非易事，且比点积更复杂的兼容性函数可能更有益处。在(C)和(D)行中我们进一步观察到，正如预期那样，更大的模型表现更好，而dropout对于避免过拟合非常有帮助。在(E)行中，我们将正弦位置编码替换为可学习的位置嵌入[9]，发现其结果与基础模型几乎相同。

6.3 英语成分句法分析

为了评估Transformer是否能够泛化到其他任务，我们进行了英语成分句法分析的实验。这项任务提出了特定的挑战：输出受到强烈的结构约束，且长度显著超过输入。此外，在数据量有限的情况下，RNN序列到序列模型尚未能取得最先进的结果[37]。

我们在《华尔街日报》(WSJ) 的宾州树库 [25] 部分训练了一个 4 层 $d_{model} = 1024$ 的 Transformer 模型，训练句子约 4 万句。同时，我们还在半监督环境下对其进行了训练，使用了规模更大的高置信度语料库和 BerkeleyParser 语料库 [37]，其中包含约 1700 万句。在仅使用 WSJ 数据时，我们采用了 16K 词符的词汇表；而在半监督设置中，词汇表规模为 32K 词符。

我们仅进行了少量实验来选择dropout（包括注意力dropout和残差dropout，见第5.4节）、学习率以及集束大小，这些实验均在Section 22开发集上进行，其余所有参数均保持与英德基础翻译模型一致。在推理过程中，我们

表4：Transformer在英语成分句法分析中泛化能力良好（结果基于WSJ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

将最大输出长度增加到输入长度的+ 300倍。在仅使用WSJ和半监督设置中，我们均采用了21的束宽和$\alpha = 0.3$的参数。

我们在表4中的结果显示，尽管缺乏针对特定任务的调优，我们的模型表现却出人意料地好，除递归神经网络语法[8]外，其效果优于所有先前报告的模型。

与RNN序列到序列模型[37]相比，Transformer即使在仅使用4万句WSJ训练集进行训练的情况下，其表现也优于Berkeley-Parser[29]。

7 结论

在这项工作中，我们提出了Transformer，这是首个完全基于注意力机制的序列转导模型，它使用多头自注意力取代了编码器-解码器架构中最常用的循环层。

在翻译任务中，Transformer的训练速度显著快于基于循环或卷积层的架构。在WMT 2014英德翻译和WMT 2014英法翻译任务中，我们均取得了新的最优结果。在前一项任务中，我们最佳模型的表现甚至超越了所有先前报道的集成模型。

我们对基于注意力模型的未来充满期待，并计划将其应用于其他任务。我们计划将Transformer扩展到涉及文本以外输入输出模态的问题，并研究局部、受限的注意力机制，以高效处理图像、音频和视频等大型输入输出。使生成过程减少序列化依赖也是我们的另一个研究目标。

我们用于训练和评估模型的代码可在 https://github.com/tensorflow/tensor2tensor 获取。

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

参考文献

[1] Jimmy Lei Ba、Jamie Ryan Kiros 与 Geoffrey E Hinton。层归一化。*arXiv preprint arXiv:1607.06450*，2016年。[2] Dzmitry Bahdanau、Kyunghyun Cho 与 Yoshua Bengio。通过联合学习对齐与翻译的神经机器翻译。*CoRR*，abs/1409.0473，2014年。[3] Denny Britz、Anna Goldie、Minh-Thang Luong 与 Quoc V. Le。神经机器翻译架构的大规模探索。*CoRR*，abs/1703.03906，2017年。[4] Jianpeng Cheng、Li Dong 与 Mirella Lapata。用于机器阅读的长短期记忆网络。*arXiv preprint arXiv:1601.06733*，2016年。

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, Yoshua Bengio。使用RNN编码器-解码器学习短语表示以用于统计机器翻译。*CoRR*, abs/1406.1078, 2014。

[6] Francois Chollet. Xception：使用深度可分离卷积进行深度学习。*arXiv preprint arXiv:1610.02357*，2016。

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, Yoshua Bengio。门控循环神经网络在序列建模上的实证评估。*CoRR*, abs/1412.3555, 2014。

[8] Chris Dyer、Adhiguna Kuncoro、Miguel Ballesteros 和 Noah A. Smith。循环神经网络语法。收录于 *Proc. of NAACL*，2016年。

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin。卷积序列到序列学习。*arXiv preprint arXiv:1705.03122v2*，2017。

[10] Alex Graves。使用循环神经网络生成序列。*arXiv preprint arXiv:1308.0850*，2013。

[11] 何恺明、张翔宇、任少卿、孙剑。深度残差学习在图像识别中的应用。载于 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*，第770–778页，2016年。

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi 和 Jürgen Schmidhuber。循环神经网络中的梯度流：学习长期依赖的困难，2001。

[13] Sepp Hochreiter 和 Jürgen Schmidhuber。长短期记忆。*Neural computation*, 9(8):1735–1780, 1997。

[14] 黄忠强与玛丽·哈珀。跨语言潜在标注自训练PCFG语法。载于 *Proceedings of the 2009 Conference on Empirical Methods in NaturalLanguage Processing*，第832–841页。ACL，2009年8月。

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, Yonghui Wu。探索语言建模的极限。*arXiv preprint arXiv:1602.02410*，2016。

[16] ukasz Kaiser 与 Samy Bengio。主动记忆能否替代注意力？收录于 *Advances in Neural Information Processing Systems, (NIPS)*，2016年。

[17] ukasz Kaiser 与 Ilya Sutskever。神经GPU学习算法。发表于 *International Conference on Learning Representations (ICLR)*，2016年。

[18] Nal Kalchbrenner、Lasse Espeholt、Karen Simonyan、Aaron van den Oord、Alex Graves 和 Koray Kavukcuoglu。线性时间内的神经机器翻译。*arXiv preprint arXiv:1610.10099v2*，2017。

[19] Yoon Kim、Carl Denton、Luong Hoang 和 Alexander M. Rush。结构化注意力网络。发表于 *International Conference on Learning Representations*，2017年。

[20] Diederik Kingma 和 Jimmy Ba。Adam：一种随机优化方法。发表于 *ICLR*，2015年。[21] Oleksii Kuchaiev 和 Boris Ginsburg。LSTM网络的因式分解技巧。*arXiv preprint arXiv:1703.10722*，2017年。

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 一种结构化的自注意力句子嵌入。*arXiv preprint arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals 和 Lukasz Kaiser。多任务序列到序列学习。*arXiv preprint arXiv:1511.06114*，2015年。

[24] Minh-Thang Luong, Hieu Pham 和 Christopher D Manning。基于注意力的神经机器翻译的有效方法。*arXiv preprint arXiv:1508.04025*，2015。

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz 和 Beatrice Santorini。构建一个大型英语注释语料库：宾州树库。*Computational linguistics*, 19(2):313–330, 1993。

[26] David McClosky、Eugene Charniak 和 Mark Johnson。用于解析的有效自训练方法。载于 *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*，第 152–159页。ACL，2006年6月。

[27] Ankur Parikh、Oscar Täckström、Dipanjan Das 和 Jakob Uszkoreit。一种可分解的注意力模型。发表于 *Empirical Methods in Natural Language Processing*，2016年。

[28] Romain Paulus、Caiming Xiong 和 Richard Socher。一种用于抽象摘要的深度强化模型。*arXiv preprint arXiv:1705.04304*，2017。

[29] Slav Petrov、Leon Barrett、Romain Thibaux 和 Dan Klein。学习准确、紧凑且可解释的树标注。载于 *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*，第 433–440 页。ACL，2006 年 7 月。

[30] Ofir Press 与 Lior Wolf。利用输出嵌入改进语言模型。*arXiv preprint arXiv:1608.05859*, 2016。

[31] Rico Sennrich, Barry Haddow 和 Alexandra Birch。使用子词单元进行稀有词的神经机器翻译。*arXiv preprint arXiv:1508.07909*，2015年。

[32] Noam Shazeer、Azalia Mirhoseini、Krzysztof Maziarz、Andy Davis、Quoc Le、Geoffrey Hinton 和 Jeff Dean。极其庞大的神经网络：稀疏门控的专家混合层。*arXiv preprint arXiv:1701.06538*，2017。

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: 一种防止神经网络过拟合的简单方法。*Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston 和 Rob Fergus。端到端记忆网络。收录于 C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama 和 R. Garnett 编辑的 *Advances in Neural Information Processing Systems 28*，第 2440–2448 页。Curran Associates, Inc., 2015。

[35] Ilya Sutskever、Oriol Vinyals 和 Quoc VV Le。基于神经网络的序列到序列学习。收录于 *Advances in Neural Information Processing Systems*，第 3104–3112 页，2014 年。

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna。重新思考计算机视觉中的Inception架构。*CoRR*, abs/1512.00567, 2015。

[37] Vinyals、Kaiser、Koo、Petrov、Sutskever 和 Hinton。将语法视为外语。收录于 *Advances in Neural Information Processing Systems*，2015年。

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, 等。谷歌的神经机器翻译系统：弥合人类与机器翻译之间的鸿沟。*arXiv preprint arXiv:1609.08144*, 2016。

[39] 周杰、曹颖、王旭光、李鹏、徐伟。用于神经机器翻译的具有快进连接的深度循环模型。*CoRR*，abs/1606.04199，2016。

[40] 祝慕华、张岳、陈文亮、张民、朱靖波。快速准确的移进-归约成分句法分析。载于 *Proceedings of the 51st Annual Meeting of the ACL (Volume1: Long Papers)*，第434–443页。ACL，2013年8月。

Attention Visualizations

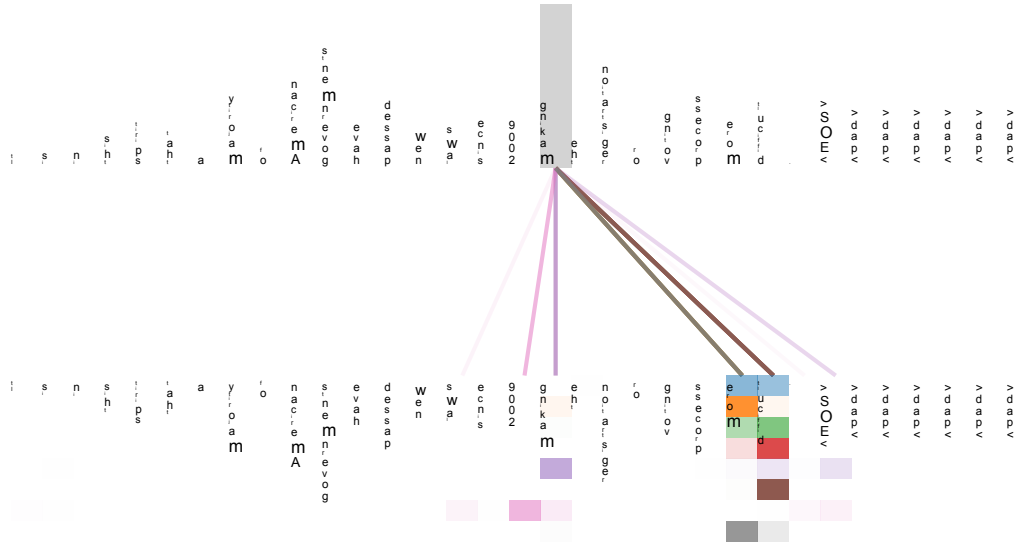

图3：第5层（共6层）编码器自注意力机制中，注意力机制遵循长距离依赖关系的示例。许多注意力头聚焦于动词"making"的远距离依赖，补全了"making...more difficult"这一短语。此处仅展示单词"making"的注意力分布。不同颜色代表不同的注意力头。建议彩色查看效果。

图4：两个注意力头，同样位于6层中的第5层，显然参与了指代消解。顶部：头5的完整注意力分布。底部：仅针对单词“its”在注意力头5和6中的独立注意力分布。请注意，该词的注意力分布非常集中。

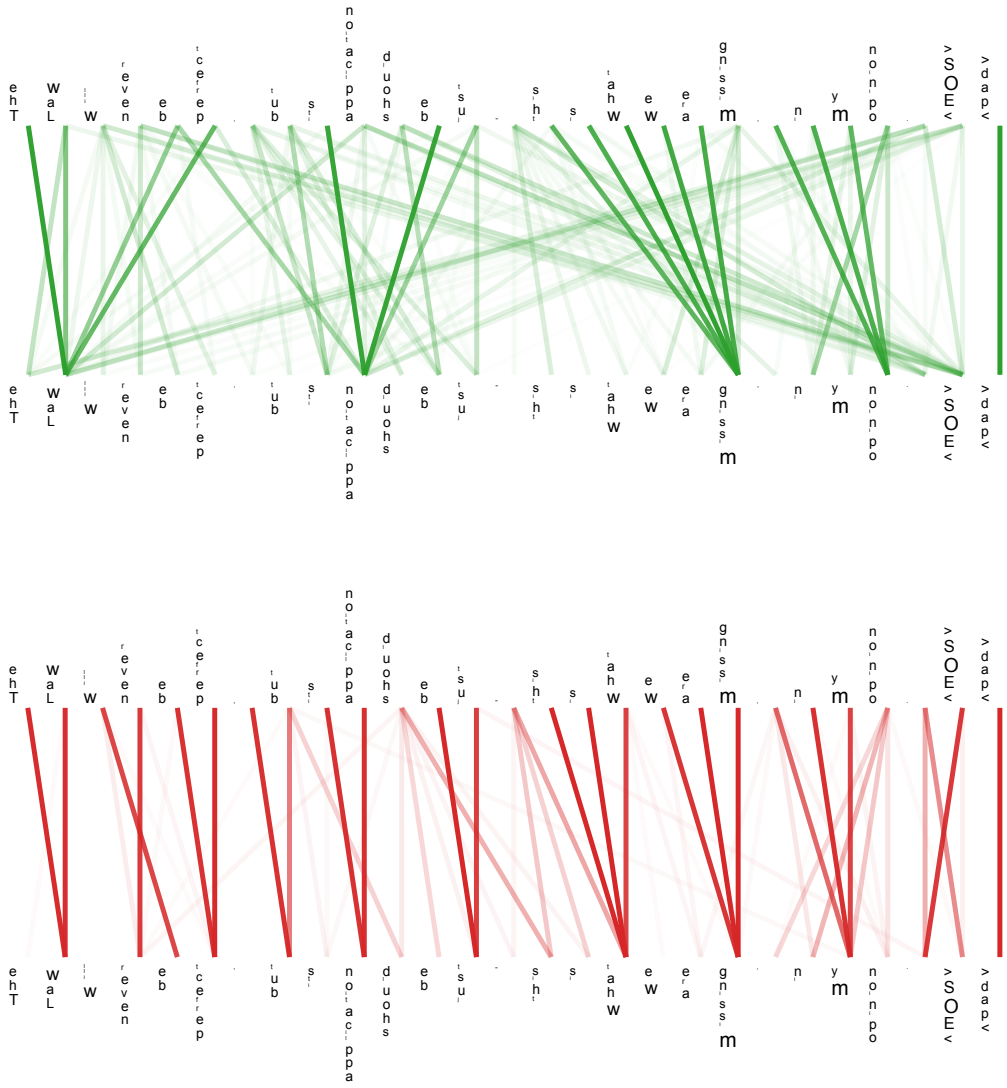

图5：许多注意力头表现出与句子结构相关的行为。我们在上面给出了两个这样的例子，它们来自6层编码器自注意力机制中第5层的两个不同头部。这些头部显然学会了执行不同的任务。