注意力就是一切

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* 谷歌研究院 nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* 谷歌研究院 llion@google.com

Aidan N. Gomez* † 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

主导的序列转导模型基于复杂的循环或卷积神经网络，包括编码器和解码器。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，完全基于注意力机制，摒弃了循环和卷积。在两个机器翻译任务上的实验表明，这些模型在质量上更优，同时更易于并行化，并且训练所需时间大大减少。在 WMT 2014 英德翻译任务上，我们的模型取得了 28.4 BLEU 的成绩，比现有最佳结果（包括集成模型）提高了 2 BLEU 以上。在 WMT 2014 英法翻译任务上，我们的模型在八个 GPU 上训练 3.5 天后，取得了 41.8 的新单模型最先进 BLEU 分数，仅为文献中最佳模型训练成本的一小部分。我们通过将 Transformer 成功应用于英语成分句法分析（无论是在大规模还是有限的训练数据下），证明了 Transformer 能很好地泛化到其他任务。

*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，特别是长短期记忆网络 [13] 和门控循环神经网络 [7]，已牢固确立为序列建模和转换问题的最先进方法，例如语言建模和机器翻译 [35, 2, 5]。此后，已有大量努力继续推动循环语言模型和编码器-解码器架构 [38, 24, 15] 的边界。

循环模型通常沿着输入和输出序列的符号位置分解计算。将位置与计算时间步对齐，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} 和位置 t 输入的函数。这种固有的顺序性阻碍了训练样本内的并行化，这在更长的序列长度下变得至关重要，因为内存限制限制了跨样本的批处理。最近的工作通过分解技巧 [21] 和条件计算 [32] 在计算效率方面取得了显著的改进，同时在后一种情况下也提高了模型性能。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，它允许在不考虑输入或输出序列中距离的情况下对依赖关系进行建模 [2, 19]。然而，除了少数情况 [27] 外，这种注意力机制都与循环网络结合使用。

在这项工作中，我们提出了 Transformer 模型架构，该模型摒弃了循环结构，完全依赖于注意力机制来建立输入和输出之间的全局依赖关系。Transformer 模型能够实现显著更多的并行计算，并且在仅用八个 P100 GPU 训练十二小时后，即可达到新的翻译质量的最新水平。

2 背景

减少顺序计算的目标也构成了扩展神经GPU [16]、ByteNet [18]和ConvS2S [9] 的基础，它们都使用卷积神经网络作为基本构建块，为所有输入和输出位置并行计算隐藏表示。在这些模型中，将任意两个输入或输出位置的信号关联起来所需的运算次数随着位置之间的距离而增长，对于ConvS2S是线性的，对于ByteNet是对数的。这使得学习远距离位置之间的依赖关系更加困难 [12]。在Transformer中，这被减少为常数次运算，尽管代价是由于平均注意力加权位置而导致有效分辨率降低，但我们通过第3.2节所述的多头注意力来抵消这种影响。

自注意力（有时也称为内部注意力）是一种注意力机制，它将单个序列的不同位置关联起来，以计算该序列的表示。自注意力已成功应用于多种任务，包括阅读理解、抽象摘要、文本蕴含和学习与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，并且在简单语言问答和语言建模任务上表现良好[34]。

据我们所知，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，而无需使用序列对齐的 RNN 或卷积。在接下来的章节中，我们将描述 Transformer，阐述自注意力的原理，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]。在这里，编码器将符号表示的输入序列（$x_1, ..., x_n$）映射到连续表示的序列（$\mathbf{z} = (z_1, ..., z_n)$）。给定$\mathbf{z}$，解码器然后一次生成一个符号的输出序列（$y_1, ..., y_m$）。在每一步，模型都是自回归的[10]，在生成下一个符号时将先前生成的符号作为附加输入。

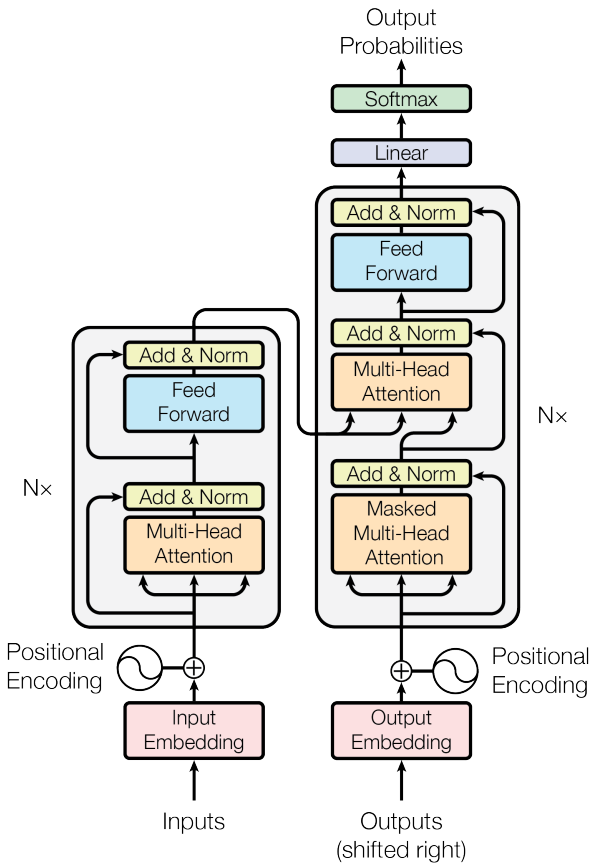

图 1：Transformer - 模型架构。

Transformer 遵循此总体架构，在编码器和解码器中都使用了堆叠的自注意力机制和逐点全连接层，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 $N = 6$ 个相同的层堆叠而成。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、位置感知的全连接前馈网络。我们在两个子层周围都使用了残差连接 [11]，然后是层归一化 [1]。也就是说，每个子层的输出是 LayerNorm(x + Sublayer(x)) ，其中 Sublayer(x) 是由子层本身实现的函数。为了方便这些残差连接，模型中的所有子层以及嵌入层都输出 $d_{\text{model}} = 512$ 维度的输出。

解码器：解码器也由一个由 $N =$ 个相同的层组成的堆栈组成。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。与编码器类似，我们在每个子层周围使用残差连接，然后进行层归一化。我们还修改了解码器堆栈中的自注意力子层，以防止位置关注后续位置。这种掩码，加上输出嵌入偏移一个位置的事实，确保位置 i 的预测只能依赖于小于 i 的位置的已知输出来。

3.2 注意力

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。输出是加权和计算的

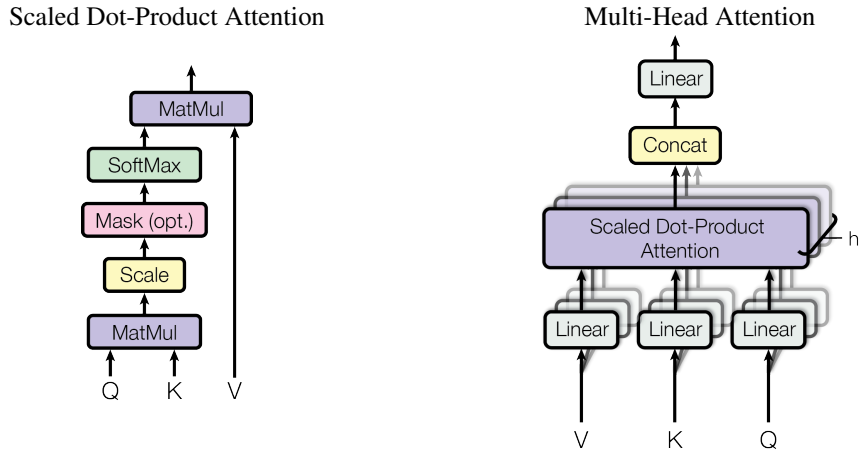

图 2：（左）缩放点积注意力。（右）多头注意力由多个并行运行的注意力层组成。

值的，其中分配给每个值（value）的权重是通过查询与相应键（key）的兼容性函数计算得出的。

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）。输入由维度为 d_k 的查询和键，以及维度为 d_v 的值组成。我们计算查询与所有键的点积，将每个点积除以 $\sqrt{d_k}$，然后应用 softmax 函数以获得值上的权重。

在实践中，我们同时计算一组查询的注意力函数，将它们打包到一个矩阵 Q 中。键和值也被打包到矩阵 K 和 V 中。我们计算输出矩阵为：

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

最常用的两种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个隐藏层的前馈网络来计算兼容性函数。虽然这两种在理论复杂度上相似，但点积注意力在实践中速度更快、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。

当d_k值较小时，这两种机制表现相似，但对于较大的d_k值，加性注意力在没有缩放的情况下优于点积注意力[3]。我们怀疑，对于较大的d_k值，点积的幅度会变大，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$\frac{1}{\sqrt{d_k}}$。

3.2.2 多头注意力

与其执行一次具有d_{model}维键、值和查询的注意力函数，我们发现通过不同的、学习到的线性投影将查询、键和值分别线性投影到h次，维度分别为d_k、d_k和d_v，这样更有益。然后，我们在这些投影后的查询、键和值版本上并行执行注意力函数，得到d_v维的

[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进行投影，从而得到最终值，如图 2 所示。

多头注意力机制允许模型在不同位置联合关注来自不同表示子空间的信息。使用单个注意力头，平均会抑制这种能力。

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

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，我们使用了 $h = 8$ 个并行注意力层（或称头）。对于其中的每一个，我们使用了 $d_k = d_v = d_\text{model}/h = 64$。由于每个头的维度减小，总计算成本与全维度单头注意力相似。

3.2.3 注意力在我们模型中的应用

Transformer 使用多头注意力机制，有三种不同的用法：

- 在“编码器-解码器注意力”层中，查询来自前一个解码器层，而记忆键和值来自编码器的输出。这使得解码器中的每个位置都可以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，在本例中是编码器前一层的输出。编码器中的每个位置都可以关注编码器前一层的任何位置。

- 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位置，直到并包括该位置。我们需要防止解码器中的信息向左流动，以保持自回归属性。我们在缩放点积注意力内部通过屏蔽掉（设置为 $-\infty$）softmax 输入中所有对应于非法连接的值来实现这一点。参见图 2。

3.3 位置感知前馈网络

除了注意力子层外，我们编码器和解码器中的每一层都包含一个全连接的前馈网络，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，中间有一个 ReLU 激活。

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

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

3.4 嵌入与Softmax

与其他的序列转换模型类似，我们使用学习到的嵌入将输入词元和输出词元转换为维度为 d_model 的向量。我们还使用通常的学习到的线性变换和 softmax 函数将解码器输出转换为预测的下一个词元概率。在我们的模型中，我们共享两个嵌入层和预 softmax 线性变换之间的相同权重矩阵，这与 [30] 类似。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、每层复杂度以及最小顺序操作数。n 是序列长度，d 是表示维度，k 是卷积的核大小，r 是受限自注意力中邻域的大小。

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循环和卷积，为了让模型能够利用序列的顺序，我们必须注入关于序列中词元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将“位置编码”添加到输入嵌入中。位置编码与嵌入具有相同的维度 d_{model}，因此两者可以相加。位置编码有多种选择，包括学习到的和固定的 [9]。

在本工作中，我们使用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 是位置，i 是维度。也就是说，位置编码的每个维度都对应一个正弦波。波长从 2π 到 $10000 \cdot 2\pi$ 形成一个几何级数。我们选择此函数是因为我们假设它可以让模型轻松学习按相对位置进行注意力，因为对于任何固定的偏移量 k，PE_{pos+k} 可以表示为 PE_{pos} 的线性函数。

我们也尝试了使用学习到的位置嵌入 [9]，发现这两个版本产生了几乎相同的结果（见表 3 第 (E) 行）。我们选择使用正弦版本，因为它可能允许模型外推到比训练期间遇到的序列长度更长的序列。

4 为什么是自注意力

在本节中，我们将自注意力层与通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，例如典型序列转换编码器或解码器中的隐藏层（$x_i, z_i \in \mathbb{R}^d$）。为了说明我们使用自注意力，我们考虑了三个期望。

一是每层的总计算复杂度。二是可并行计算的计算量，以所需的最少顺序操作次数衡量。

第三是网络中长距离依赖的路径长度。学习长距离依赖是许多序列转换任务中的一个关键挑战。影响学习此类依赖能力的一个关键因素是前向和后向信号在网络中必须遍历的路径长度。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，自注意力层将所有位置连接起来，其顺序执行的操作数量是恒定的，而循环层需要 $O(n)$ 次顺序操作。在计算复杂度方面，当序列长度增加时，自注意力层的速度比循环层快。

长度 n 小于表示的维度 d，这在机器翻译的最新模型中使用的句子表示中最为常见，例如词片段 [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，自注意力可以被限制为仅考虑输入序列中以相应输出位置为中心的、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。

宽度为$k < n$的单个卷积层无法连接所有输入和输出位置对。在连续核的情况下，这需要堆叠$O(n/k)$个卷积层，或者在扩张卷积的情况下堆叠$O(log_k(n))$个卷积层[18]，这会增加网络中任意两个位置之间最长路径的长度。卷积层的成本通常是循环层的k倍。然而，可分离卷积[6]将复杂度显著降低到$O(k \cdot n \cdot d + n \cdot d^2)$。然而，即使有$k = n$，可分离卷积的复杂度也等于自注意力层和逐点前馈层的组合，而这正是我们在模型中所采用的方法。

作为附带的好处，自注意力可以产生更具可解释性的模型。我们检查了我们模型中的注意力分布，并在附录中展示和讨论了示例。不仅单个注意力头清楚地学会了执行不同的任务，而且许多注意力头似乎还表现出与句子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方案。

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，该数据集包含约450万个句子对。句子使用字节对编码[3]进行编码，该编码具有约37000个标记的共享源-目标词汇表。对于英法，我们使用了明显更大的WMT 2014英法数据集，包含3600万个句子，并将标记分割成一个32000个词片段的词汇表[38]。句子对通过近似序列长度进行批处理。每个训练批次包含一组句子对，包含约25000个源标记和25000个目标标记。

5.2 硬件和进度

我们在配备 8 个 NVIDIA P100 GPU 的一台机器上训练了我们的模型。对于使用论文中描述的超参数的基础模型，每个训练步骤大约需要 0.4 秒。我们总共训练了基础模型 100,000 个步骤，耗时 12 小时。对于我们的大模型（在表 3 的最后一行描述），步长为 1.0 秒。大模型训练了 300,000 个步骤（3.5 天）。

5.3 优化器

我们使用了 Adam 优化器 [20]，其 $\beta_1 = 0.9$、$\beta_2 = 0.98$ 和 $\epsilon = 10^{-9}$。我们根据以下公式在训练过程中改变了学习率：

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

这对应于在最初的 $warmup\_steps$ 个训练步骤中线性增加学习率，之后根据步数的平方根的倒数成比例地减小学习率。我们使用了 $warmup\_steps = 4000$。

5.4 正则化

我们在训练过程中使用了三种正则化方法：

表 2：Transformer 在英语到德语和英语到法语 newstest2014 测试中取得了比之前最先进模型更好的 BLEU 分数，而训练成本却大大降低。

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 Dropout 我们将 dropout [33] 应用于每个子层的输出，在将其加到子层输入并进行归一化之前。此外，我们在编码器和解码器堆栈的嵌入和位置编码的总和上应用 dropout。对于基础模型，我们使用的比率为 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 [36]。这会损害困惑度，因为模型会变得更加不确定，但会提高准确率和 BLEU 分数。

6 结果

6.1 机器翻译

在 WMT 2014 英德翻译任务上，大型 Transformer 模型（表 2 中的 Transformer (big)）比之前报道的最佳模型（包括集成模型）的 BLEU 分数高出 2.0，创下了 28.4 的新 state-of-the-art BLEU 分数。该模型的配置列在表 3 的最后一行。训练在 8 个 P100 GPU 上花费了 3.5 天。即使是我们的基础模型也超越了所有先前发布的模型和集成模型，而训练成本仅为任何竞争模型的一小部分。

在 WMT 2014 英法翻译任务上，我们的大型模型取得了 41.0 的 BLEU 分数，性能优于所有先前发布的单一模型，而训练成本仅为先前最先进模型的 $1/4$。训练用于英法翻译的 Transformer (big) 模型使用了 $P_{drop} = 0.1$ 的 dropout 率，而不是 0.3。

对于基础模型，我们使用了通过平均最后 5 个检查点获得的单个模型，这些检查点以 10 分钟的间隔写入。对于大型模型，我们平均了最后 20 个检查点。我们使用了束搜索，束大小为 4，长度惩罚为 α =0.6 [38]。这些超参数是在开发集上进行实验后选择的。我们将推理期间的最大输出长度设置为输入长度 +50，但尽可能提前终止 [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 数量以及每块 GPU 的持续单精度浮点容量估算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，我们以不同的方式改变了我们的基础模型，并衡量了在英语到德语翻译上的性能变化。

[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。所有指标均在英语到德语翻译开发集 newstest2013 上计算。列出的困惑度是根据我们的字节对编码计算的每个词片段（per-wordpiece）的困惑度，不应与每个词（per-word）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

开发集，newstest2013。我们使用了上一节所述的束搜索，但没有进行检查点平均。我们将这些结果呈现在表 3 中。

在表 3 的行 (A) 中，我们改变了注意力头的数量以及注意力键和值的维度，同时保持计算量不变，具体如 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 的行 (B) 中，我们观察到减小注意力键大小 d_k 会损害模型质量。这表明确定兼容性并不容易，并且比点积更复杂的兼容性函数可能是有益的。我们在行 (C) 和 (D) 中进一步观察到，正如预期的那样，更大的模型更好，而 dropout 在避免过拟合方面非常有帮助。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]，并观察到与基线模型几乎相同的结果。

6.3 英语选区分析

为了评估Transformer能否泛化到其他任务，我们在英语成分句法分析上进行了实验。这项任务带来了特定的挑战：输出受到严格的结构约束，并且比输入长得多。此外，在小数据环境下，RNN序列到序列模型未能达到最先进的结果[37]。

我们在一份包含约 40K 训练句的 Penn Treebank [25] 的华尔街日报 (WSJ) 部分上训练了一个 $d_{model} = 1024$ 的 4 层 Transformer。我们还在半监督设置下训练了它，使用了来自 [37] 的更大置信度语料库和 BerkleyParser 语料库，共约 17M 句。对于仅使用 WSJ 的设置，我们使用了 16K 词汇量，对于半监督设置，我们使用了 32K 词汇量。

我们仅在第 22 节的开发集上进行了少量实验来选择 dropout、注意力机制和残差连接（第 5.4 节）、学习率和束搜索大小，所有其他参数均与英德基础翻译模型保持不变。在推理过程中，我们

表 4：Transformer 在英语成分句法分析上泛化良好（结果见 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。我们在 WSJ 仅设置和半监督设置中都使用了束宽为 21 和 $\alpha = 0.3$。

我们在表 4 中的结果表明，尽管缺乏特定任务的调整，我们的模型表现出人意料地好，除了循环神经网络语法 [8] 外，其结果优于所有先前报道的模型。

与RNN序列到序列模型[37]相比，Transformer即使仅在40K句子的WSJ训练集上进行训练，其性能也优于Berkeley-Parser[29]。

7 结论

在这项工作中，我们提出了 Transformer，这是第一个完全基于注意力机制的序列转换模型，它用多头自注意力机制取代了编码器-解码器架构中最常用的循环层。

对于翻译任务，Transformer 的训练速度比基于循环层或卷积层的架构快得多。在 WMT 2014 英语到德语和 WMT 2014 英语到法语的翻译任务上，我们都达到了新的最先进水平。在前者任务中，我们最好的模型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感到兴奋，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的输入和输出模态的问题，并研究局部、受限的注意力机制，以有效地处理图像、音频和视频等大型输入和输出。使生成过程的顺序性降低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 的宝贵意见、修正和启发。

参考文献

[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, a nd Yoshua Bengio. 使用RNN编码器-解码器学习短语表示用于统计机器翻译. *CoRR*, abs/1406 .1078, 2014.

[6] Francois Chollet. Xception: 深度学习与深度可分离卷积. *arXiv preprint arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 经验性评估门控循环神经网络在序列建模上的表现. *CoRR*, abs/1412.3555, 2014.

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

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and 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] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for im- age 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. 循环网络中的梯度流：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):1735−17 80，1997。

[14] 黄忠强和玛丽·哈珀。跨语言的带潜在标注的自训练PCFG语法。在 *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*，第 832−841页。ACL，2009年8月。

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 探索语言模型的极限。*arXiv preprint arXiv:1602.02410*, 2016。

[16] ukasz Kaiser 和 Samy Bengio。主动记忆可以取代注意力吗？在 *Advances in Neural Information Processing Systems, (NIPS)*，2016。

[17] ukasz Kaiser 和 Ilya Sutskever。Neural GPUs 学习算法。在 *International Conference on Learning Representations (ICLR)*，2016。

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and K oray Kavukcuoglu. 神经机器翻译的线性时间实现. *arXiv preprint arXiv:1610.10099v2*, 2017.

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

[20] Diederik Kingma 和 Jimmy Ba。Adam：一种随机优化方法。在 *ICLR*，2015 年。 [21] Ol eksii Kuchaiev 和 Boris Ginsburg。LSTM 网络中的因子分解技巧。 *arXiv preprint arXiv:1703.10722*，2017 年。

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, a nd 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. 多任务序列到序列学习. *arXiv preprint arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 有效的基于注意力的神经机器翻译方法。*arXiv preprint arXiv:1508.04025*, 2015.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 构建大型英语标注语料库：宾州树库。*Computational linguistics*，19(2):313–330，1993。

[26] David McClosky, Eugene Charniak, and Mark Johnson. 有效的自训练用于解析。在 *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, and Jakob Uszkoreit. A decomposable attentio n 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. 学习准确、紧凑且可解释的树注释。在 *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 和 Lior Wolf。使用输出嵌入改进语言模型。*arXiv preprint arXiv:1608.05859*, 2 016。

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. 使用子词单元的神经机器翻译稀有词. *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 Salakhutdi - nov. 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. 端到端记忆网络。在 C . Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, 和 R. Garnett, 编辑, *Advances in Neural Information Processing Systems 28*, 页码 2440–2448。Curran Associates, Inc. , 2015。

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural ne tworks. 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] 永辉吴，迈克·舒斯特，志峰陈，国乐，穆罕默德·诺鲁兹，沃尔夫冈·马赫雷，马克西姆·克里昆，元曹，秦高，克劳斯·马赫雷等。谷歌的神经机器翻译系统：弥合人类与机器翻译之间的差距。*arXiv preprint arXiv:1609.08144*，2016。

[39] 乔杰，曹颖，王旭光，李鹏，徐伟。用于神经机器翻译的具有快速前馈连接的深度递归模型。*CoRR*, abs/1606.04199, 2016。

[40] 诸慕华，张悦，陈文亮，张敏，朱景波。快速准确的移近-规约成分分析。在 *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*，第 434–443 页。AC L，2013 年 8 月。

Attention Visualizations

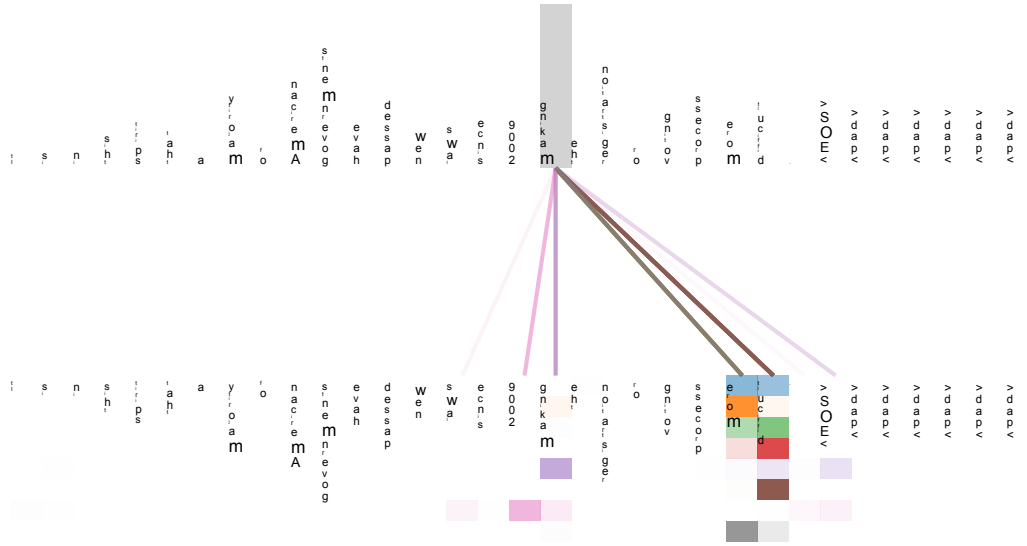

图 3：一个在第 5 层（共 6 层）的编码器自注意力中遵循长距离依赖的注意力机制示例。许多注意力头关注动词“making”的一个远距离依赖，完成了短语“making...more difficult”。此处显示的注意力仅针对单词“making”。不同的颜色代表不同的头。最好在彩色下查看。

图 4：两个注意力头，同样在第 6 层的第 5 层，似乎参与了照应消解。顶部：头 5 的完整注意力。底部：仅来自单词“its”的注意力头 5 和 6 的孤立注意力。请注意，该单词的注意力非常集中。

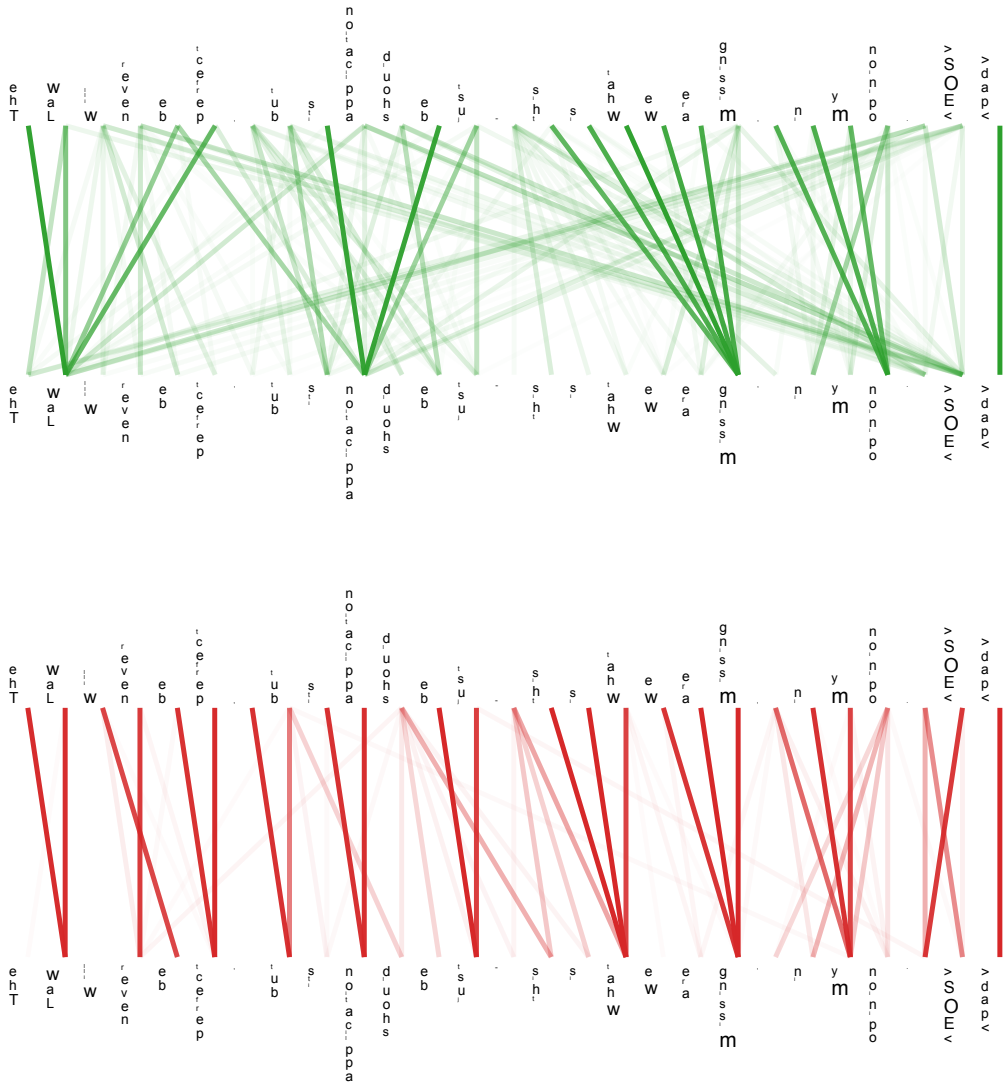

图 5：许多注意力头表现出似乎与句子结构相关的行为。我们在上面给出了两个这样的例子，分别来自第 6 层编码器自注意力中的两个不同头。这些头显然学会了执行不同的任务。