注意力就是你所需要的一切

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

主流的序列转换模型基于复杂的循环神经网络或卷积神经网络，其中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。我们提出一种新的简单网络架构——Transformer，该架构完全基于注意力机制，彻底摒弃循环和卷积。针对两项机器翻译任务的实验表明，这些模型在质量上更优，同时具有更强的并行能力，训练所需时间也显著更少。在 WMT 2014 英译德翻译任务中，我们的模型取得了 28.4 BLEU，较包括集成模型在内的现有最佳结果提高了超过 2 个 BLEU 点。在 WMT 2014 英译法翻译任务中，我们的模型取得了新的单模型最高 BLEU 分数 41.8；在 8 个 GPU 上训练 3.5 天即可达到这一结果，仅为文献中最佳模型训练成本的一小部分。我们还将 Transformer 成功应用于英语成分句法分析，证明了该模型能够良好地推广到其他任务，并且适用于大规模和有限训练数据两种情形。

注：*贡献相同。作者列举顺序为随机排列。Jakob 提议用自注意力替代 RNN，并开始评估这一想法。Ashish 与 Illia 设计并实现了最初的 Transformer 模型，并深度参与了这项工作的各个方面。Noam 提出了缩放点积注意力、多头注意力和无参数位置表示，并成为几乎所有细节工作的另一位核心参与者。Niki 在我们最初的代码库和 tensor2tensor 中设计、实现、调试并评估了无数模型变体。Llion 也试验了新颖的模型变体，负责最初的代码库、高效推理和可视化。Lukasz 和 Aidan 花费了无数个漫长的日子设计和实现 tensor2tensor，以替代我们早期的代码库，大幅改进结果并极大加速研究。

† 在 Google Brain 工作期间完成。

‡ 在 Google Research 工作期间完成。

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]，已经确立为序列建模和转换问题（如语言建模和机器翻译 [35, 2, 5]）中的最先进方法。此后，大量研究持续推动循环语言模型和编码器—解码器架构 [38, 24, 15] 的发展边界。

循环模型通常沿输入和输出序列的符号位置分解计算。将这些位置与计算时间步对齐后，它们根据前一隐藏状态 h_{t-1} 和位置 t 的输入，生成隐藏状态序列 h_t。这种固有的顺序性使训练样本内部无法并行化；随着序列长度增加，这一点变得至关重要，因为内存限制了跨样本的批处理规模。近期研究通过分解技巧 [21] 和条件计算 [32]，在提高计算效率方面取得了显著进展，后者同时也改善了模型性能。然而，顺序计算这一根本性约束依然存在。

注意力机制已经成为各种任务中高效序列建模和转换模型不可或缺的组成部分，使模型能够不受输入或输出序列中距离的限制来建模依赖关系 [2, 19]。然而，除少数情况外 [27]，这类注意力机制都与循环网络结合使用。

在本文中，我们提出 Transformer，这是一种摒弃循环、完全依靠注意力机制来建立输入与输出之间全局依赖关系的模型架构。Transformer 能够实现显著更强的并行化能力，并且在 8 个 P100 GPU 上训练短至 12 小时后，即可达到新的翻译质量最高水平。

2 背景

减少顺序计算的目标也是 Extended Neural GPU [16]、ByteNet [18] 和 ConvS2S [9] 的基础；这些模型都使用卷积神经网络作为基本构件，对所有输入和输出位置并行计算隐藏表示。在这些模型中，关联任意两个输入或输出位置的信号所需的操作数会随位置间距离增加：对于 ConvS2S 呈线性增长，对于 ByteNet 呈对数增长。这使得模型更难学习远距离位置之间的依赖关系 [12]。在 Transformer 中，所需操作数被降为常数，但代价是有效分辨率降低，因为模型会对注意力加权的位置进行平均；我们通过第 3.2 节所述的多头注意力抵消了这一影响。

自注意力有时也称为内部注意力，是一种关联单个序列中不同位置、以计算该序列表示的注意力机制。自注意力已经成功应用于多种任务，包括阅读理解、抽象式摘要、文本蕴含以及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，而非与序列位置对齐的循环结构；研究表明，它们在简单语言问答和语言建模任务上表现良好 [34]。

然而，据我们所知，Transformer 是第一个完全依靠自注意力计算输入和输出表示、而不使用与序列对齐的 RNN 或卷积的转换模型。在以下各节中，我们将介绍 Transformer，阐述采用自注意力的动机，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换模型都采用编码器—解码器结构 [5, 2, 35]。其中，编码器将符号表示的输入序列 $(x_1, ..., x_n)$ 映射为连续表示序列 $\boldsymbol{z} = (z_1, ..., z_n)$。给定 $\boldsymbol{z}$ 后，解码器逐个元素地生成符号输出序列 $\left(y_1, ..., y_m\right)$。在每一步中，模型都是自回归的 [10]：生成下一个符号时，将此前生成的符号作为额外输入。

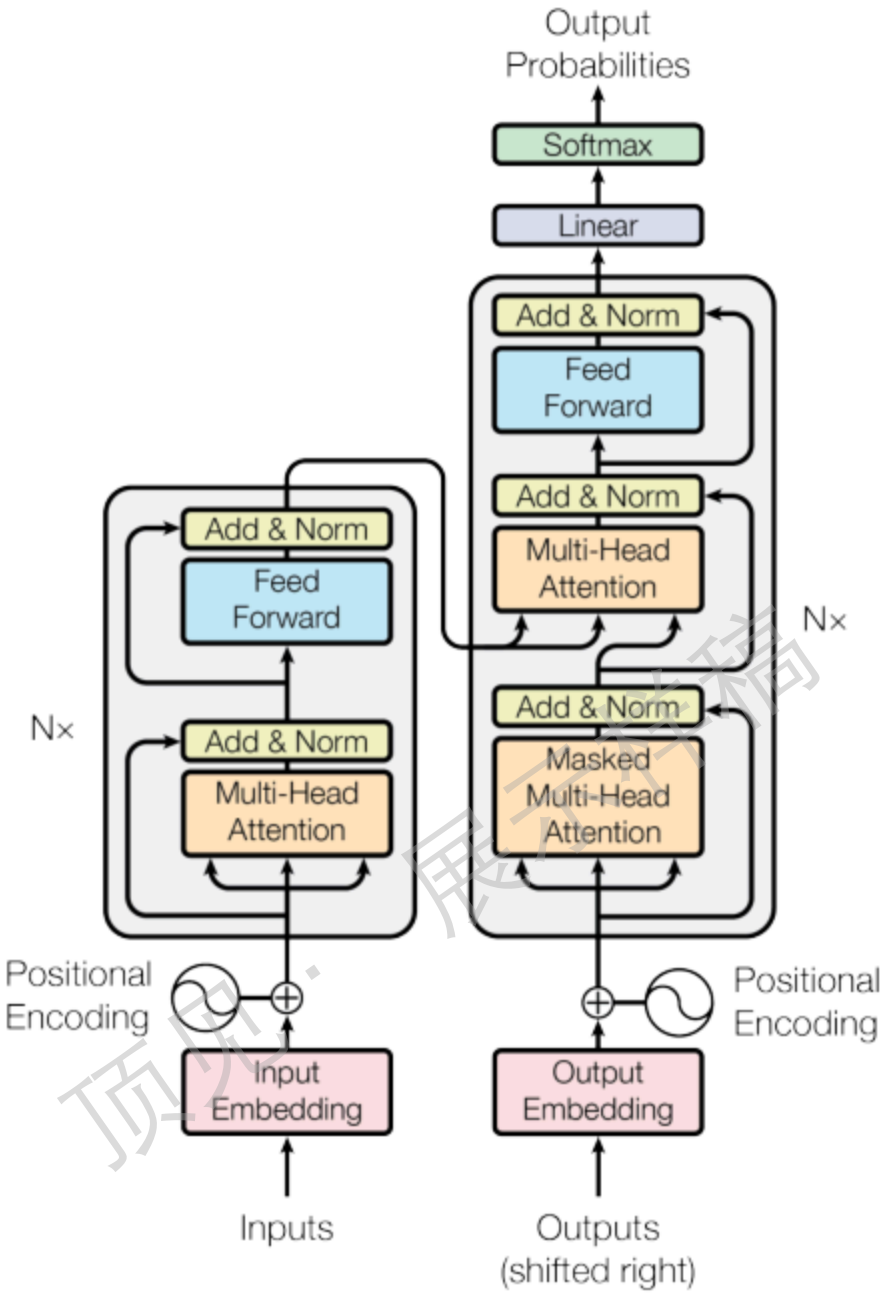

图 1：Transformer——模型架构。

Transformer 遵循这一总体架构，对编码器和解码器均使用堆叠的自注意力层和逐位置全连接层，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 $N = 6$ 个相同层堆叠而成。每层包含两个子层。第一个是多头自注意力机制，第二个是简单的逐位置全连接前馈网络。我们在两个子层外都采用残差连接 [11]，随后进行层归一化 [1]。也就是说，每个子层的输出为 $\text{LayerNorm}(x + \text{Sublayer}(x))$，其中 $\text{Sublayer}(x)$ 是该子层所实现

的函数。为便于使用这些残差连接，模型中的所有子层以及嵌入层都产生维度为 $d_{\text{model}} = 512$ 的输出。

解码器：解码器同样由 $N = 6$ 个相同层堆叠而成。除每个编码器层中的两个子层外，解码器还插入第三个子层，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，我们在各个子层外采用残差连接，随后进行层归一化。我们还修改了解码器堆栈中的自注意力子层，以阻止位置关注后续位置。这种掩码与输出嵌入向后偏移一个位置相结合，确保位置 i 的预测只能依赖位置 i 之前的已知输出。

3.2 注意力

注意力函数可以描述为将一个查询和一组键值对映射为一个输出，其中查询、键、值和输出均为向量。输出计算为加权求和

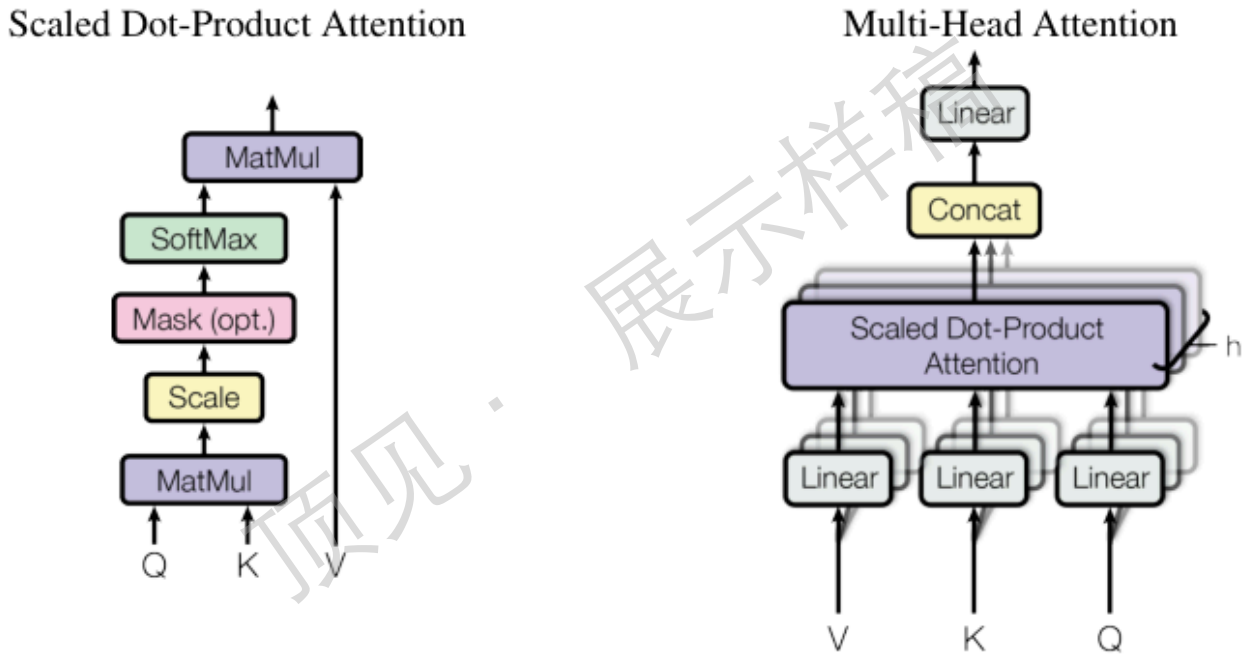

图 2：（左）缩放点积注意力。（右）多头注意力由若干并行运行的注意力层组成。

得到，其中分配给每个值的权重由查询与相应键之间的兼容性函数计算得到。

3.2.1 缩放点积注意力

我们将特定的注意力形式称为“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，将每个点积除以 $\sqrt{d_k}$ ，再应用 softmax 函数以获得各个值的权重。

实际计算中，我们同时对一组查询计算注意力函数，并将其打包成矩阵 Q。键和值也分别打包成矩阵 K 和 V。我们将输出矩阵计算为：

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

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。除缩放因子 $\frac{1}{\sqrt{d_k}}$ 外，点积注意力与我们的算法完全相同。加性注意力使用带有单个隐藏层的前馈网络计算兼容性函数。尽管二者在理论复杂度上相近，但在实践中点积注意力速度更快、空间效率更高，因为它可以使用高度优化的矩阵乘法代码实现。

当 d_k 较小时，两种机制的表现相近；而当 d_k 较大时，加性注意力优于未进行缩放的点积注意力 [3]。我们推测，当 d_k 较大时，点积的绝对值会变大，使 softmax 函数进入梯度极小的区域 [4] 为抵消这一影响，我们将点积乘以 $\frac{1}{\sqrt{d_k}}$ 进行缩放。

3.2.2 多头注意力

我们没有对维度为 d_{model} 的键、值和查询执行单个注意力函数，而是发现，将查询、键和值分别通过不同的、可学习的线性投影进行 h 次投影更有利，投影后的维度分别为 d_k, d_k 和 d_v。随后，我们在这些投影后的查询、键和值上并行执行注意力函数，得到维度为 d_v 的

注：[4] 为说明点积为何会变大，假设 q 和 k 的分量是均值为 0、方差为 1 的相互独立随机变量。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。

输出值。将这些输出拼接后再次进行投影，从而得到最终的值，如图 2 所示。

多头注意力使模型能够在不同位置联合关注来自不同表示子空间的信息。使用单个注意力头时，平均操作会抑制这种能力。

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

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

在本文中，我们采用 $h = 8$ 个并行注意力层，即注意力头。对于每个注意力头，我们使用 $d_k = d_v = d_{\text{model}}/h = 64$。由于每个注意力头的维度降低，总计算成本与完整维度的单头注意力相近。

3.2.3 注意力在我们模型中的应用

Transformer 以三种不同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。这使解码器中的每个位置都能够关注输入序列中的所有位置。这模拟了序列到序列模型（如 [38, 2, 9]）中典型的编码器—解码器注意力机制。

编码器包含自注意力层。在自注意力层中，所有键、值和查询都来自同一位置来源，在此即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中的所有位置。

同样，解码器中的自注意力层允许解码器的每个位置关注解码器中截至该位置（包括该位置）的所有位置。我们需要阻止信息向左流动，以保持自回归属性。在缩放点积注意力内部，我们通过屏蔽（将其设为 $-\infty$）softmax 输入中对应非法连接的所有值来实现这一点。见图 2。

3.3 逐位置前馈网络

除注意力子层外，我们的编码器和解码器中的每一层都包含一个全连接前馈网络，该网络对每个位置分别且以相同方式应用。它由两个线性变换组成，中间使用 ReLU 激活函数。

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

2.

虽然不同位置使用相同的线性变换，但不同层使用不同的参数。另一种描述方式是使用两个核大小为 1 的卷积。输入和输出的维度为 $d_{\text{model}} = 512$，内层维度为 $d_{ff} = 2048$。

3.4 嵌入和 softmax

与其他序列转换模型类似，我们使用学习得到的嵌入，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得到的线性变换和 softmax 函数，将解码器输出转换为下一个词元的预测概率。在我们的模型中，两个嵌入层与 softmax 前的线性变换共享同一个权重矩阵，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、每层复杂度和顺序操作的最小数量。n 是序列长度，d 是表示维度，k 是卷积核大小，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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

表格中文译文

表 1：不同层类型的最大路径长度、每层复杂度和顺序操作的最小数量。n 是序列长度，d 是表示维度，k 是卷积核大小，r 是受限自注意力中的邻域大小。

层类型	每层复杂度	顺序操作	最大路径长度
自注意力	$O\left(n^2 \cdot d\right)$	$O(1)$	$O(1)$
循环	$O\left(n \cdot d^2\right)$	$O(n)$	$O(n)$
卷积	$O\left(k \cdot n \cdot d^2\right)$	$O(1)$	$O\left(\log_k(n)\right)$
受限自注意力	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 位置编码

由于我们的模型不包含循环和卷积，为了使模型能够利用序列的顺序，我们必须注入有关序列中词元相对位置或绝对位置的信息。为此，我们在编码器和解码器堆栈底部的输入嵌入中加入“位置编码”。位置编码与嵌入具有相同的维度 d_{model}，因此二者可以相加。位置编码有多种选择，既可以是学习得到的，也可以是固定的 [9]。

在本文中，我们使用不同频率的正弦和余弦函数：

$$
\begin{aligned}
PE_{(\text{pos},2i)} &= \sin\left(\text{ pos }/10000^{2i/d_{\text{model}}}\right) \\
PE_{(\text{pos},2i+1)} &= \cos\left(\text{ pos }/10000^{2i/d_{\text{model}}}\right)
\end{aligned}
$$

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曲线。波长从 2π 到 $10000 \cdot 2\pi$ 构成几何级数。我们选择这一函数，是因为我们假设它能够使模型轻松学习依据相对位置进行关注：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。

我们还试验了使用学习得到的位置嵌入 [9]，发现两个版本产生的结果几乎完全相同（见表 3 中的第（E）行）。我们选择正弦版本，是因为它可能使模型能够外推到长于训练期间所遇到长度的序列。

4 为什么采用自注意力

本节将自注意力层的各个方面与通常用于将一个可变长度的符号表示序列 $\left(x_1, \ldots, x_n\right)$ 映射为另一个等长序列 $\left(z_1, \ldots, z_n\right)$ 的循环层和卷积层进行比较，其中 $x_i, z_i \in \mathbb{R}^d$，例如典型序列转换编码器或解码器中的隐藏层。为了说明采用自注意力的动机，我们考虑三个目标。

第一个目标是每层的总计算复杂度。第二个目标是可并行化的计算量，以所需顺序操作的最小数量进行衡量。

第三个指标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换任务中的一项关键挑战。影响学习此类依赖关系能力的一个关键因素，是前向和反向信号必须在网络中 traversing 的路径长度。输入序列和输出序列中任意位置组合之间的路径越短，就越容易学习长程依赖关系[12]。因此，我们还比较了由不同层类型组成的网络中任意输入位置与输出位置之间的最大路径长度。

如表1所示，自注意力层通过固定数量的顺序执行操作连接所有位置，而循环层则需要 $O(n)$ 个顺序操作。就计算复杂度而言，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；对于当前机器翻译领域最先进模型所使用的句子表示（例如词片段[38]和字节对[31]表示），通常都是这种情况。为了提高涉及超长序列任务的计算性能，可以将自注意力限制为仅考虑输入序列中以相应输出位置为中心、大小为 r 的邻域。这会将最大路径长度增加到 $O(n/r)$。我们计划在未来工作中进一步研究这一方法。

当卷积核宽度为 $k < n$ 时，单个卷积层无法连接所有输入位置与输出位置的组合。在连续卷积核的情况下，需要堆叠 $O(n/k)$ 个卷积层；在膨胀卷积的情况下，则需要 $O\left(\log_k(n)\right)$ 个卷积层[18]，从而增加网络中任意两个位置之间最长路径的长度。卷积层通常比循环层的开销更大，增加的倍数为 k。然而，可分离卷积[6]能够显著降低复杂度，使其降至 $O\left(k \cdot n \cdot d + n \cdot d^2\right)$。不过，即使 $k = n$，可分离卷积的复杂度仍等于一个自注意力层与一个逐点前馈层的组合，这正是我们在模型中采用的方法。

作为一个附带好处，自注意力还可能产生更具可解释性的模型。我们检查了模型中的注意力分布，并在附录中给出和讨论了相关示例。不同的注意力头不仅明显学会了执行不同的任务，而且其中许多似乎表现出与句法和语义结构相关的行为。

5 训练

本节介绍我们模型的训练方案。

5.1 训练数据与批处理

我们在标准的 WMT 2014 英语—德语数据集上进行训练，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，该编码具有一个约37000个词元的共享源语言—目标语言词汇表。对于英语—法语任务，我们使用规模大得多的 WMT 2014 英语—法语数据集，其中包含3600万句子，并将词元切分为一个包含32000个词片段的词汇表[38]。我们按照近似序列长度将句子对组成批次。每个训练批次包含一组句子对，其中约有25000个源语言词元和25000个目标语言词元。

5.2 硬件与训练安排

我们在一台配备8块 NVIDIA P100 GPU的机器上训练模型。对于使用论文中所述超参数的基础模型，每个训练步骤约耗时0.4秒。基础模型总共训练了100000步，即12小时。对于大模型（见表3最下面一行），每步耗时1.0秒。大模型训练了300000步（3.5天）。

5.3 优化器

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

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

3.

这相当于在最初的 warmup_steps 个训练步骤中线性增加学习率，此后使其按照步数的平方根倒数成比例地下降。我们使用的 warmup_steps = 4000。

5.4 正则化

我们在训练过程中采用三种正则化方法：

表2：Transformer 在英语—德语和英语—法语 newstest2014 测试集上以一小部分训练成本取得了优于此前最先进模型的 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	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

表格中文译文

模型	BLEU（EN-DE）	BLEU（EN-FR）	训练成本（FLOPs，EN-DE）	训练成本（FLOPs，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（基础模型）	27.3	38.1	\multicolumn{2}{	c

模型	BLEU（EN-DE）	BLEU（EN-FR）	训练成本（FLOPs，EN-DE）	训练成本（FLOPs，EN-FR）
Transformer（大模型）	28.4	41.8	\multicolumn{2}{	c

残差随机失活 我们将随机失活[33]应用于每个子层的输出，在其加到子层输入并进行归一化之前执行。此外，我们还对编码器堆栈和解码器堆栈中的嵌入与位置编码之和应用随机失活。对于基础模型，我们使用的比率为 $P_{\text{drop}} = 0.1$。

标签平滑 在训练过程中，我们采用取值为 $\epsilon_{ls} = 0.1$ 的标签平滑[36]。这会损害困惑度，因为模型学会变得更加不确定，但能够提高准确率和 BLEU 分数。

6 结果

6.1 机器翻译

在 WMT 2014 英语—德语翻译任务中，大型 Transformer 模型（表2中的 Transformer（大模型））比此前报告的最佳模型（包括集成模型）高出2.0个以上 BLEU 分数，创下28.4这一新的最先进 BLEU 分数。该模型的配置列于表3的最后一行。训练在8块 P100 GPU 上耗时3.5天。即使是我们的基础模型，也以任一竞争模型训练成本的一小部分超过了此前发表的所有模型和集成模型。

在 WMT 2014 英语—法语翻译任务中，我们的大模型取得了41.0的 BLEU 分数，超过了此前发表的所有单模型，而训练成本不到此前最先进模型的 1/4。用于英语—法语训练的 Transformer（大模型）采用的随机失活率为 $P_{\text{drop}} = 0.1$，而不是0.3。

对于基础模型，我们通过对最后5个检查点取平均得到一个单模型，这些检查点每隔10分钟保存一次。对于大模型，我们对最后20个检查点取平均。我们采用束搜索，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定的。在推理过程中，我们将最大输出长度设为输入长度 + 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、所使用的 GPU 数量以及每个 GPU 持续单精度浮点运算能力的估计值相乘，来估计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，我们以不同方式改变基础模型，并测量其在英语—德语翻译任务开发集上的性能变化。

注：[5] 对于 K80、K40、M40 和 P100，我们分别采用2.8、3.7、6.0和9.5 TFLOPS 的数值。

表3：Transformer 架构的不同变体。未列出的取值与基础模型相同。所有指标均来自英语—德语翻译开发集 newstest2013。所列困惑度按照我们的字节对编码以每个词片段计算，不应与按词计算的困惑度进行比较。

	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

表格中文译文

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	训练步数	PPL（开发集）	BLEU（开发集）	参数 $\times 10^6$
基础模型	6	512	2048	8	64	64	0.1	0.1	100 K	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		

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	**训练步数**	**PPL（开发集）**	**BLEU（开发集）**	**参数 $\times 10^6$**
							0.2		5.47	25.7		
(E)						位置嵌入而非正弦函数			4.92	25.7		
大模型	6	1024	4096	16			0.3		300K	4.33	26.4	213

开发集 newstest2013。我们采用了上一节所述的束搜索，但没有进行检查点平均。结果见表3。

在表3的（A）行中，我们改变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，具体如第3.2.2节所述。尽管单头注意力比最佳设置低0.9个 BLEU 分数，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，减小注意力键的大小 d_k 会损害模型质量。这表明确定兼容性并不容易，采用比点积更复杂的兼容性函数可能会有所帮助。我们还在（C）和（D）行中观察到，正如预期的那样，更大的模型表现更好，而随机失活对于避免过拟合非常有帮助。在（E）行中，我们用学习式位置嵌入替换正弦位置编码[9]，并观察到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，我们在英语成分句法分析上进行了实验。该任务具有一些特殊挑战：输出受到严格的结构约束，并且明显长于输入。此外，在小数据情形下，RNN 序列到序列模型尚未取得最先进的结果[37]。

我们在《华尔街日报》（WSJ）Penn Treebank[25]部分数据上训练了一个4层 Transformer，其 $d_{\text{model}} = 1024$，训练句子约4万条。我们还在半监督设置下对其进行训练，使用规模更大的高置信度语料库和 BerkleyParser 语料库，共约1700万句子[37]。在仅使用 WSJ 的设置下，我们使用包含16000个词元的词汇表；在半监督设置下，则使用包含32000个词元的词汇表。

我们仅进行了少量实验，以在第22节开发集上选择随机失活率（包括注意力随机失活和残差随机失活，第5.4节）、学习率和束大小；所有其他参数均保持与英语—德语基础翻译模型相同。在推理过程中，我们

表4：Transformer 能够良好地泛化到英语成分句法分析（结果来自 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

表格中文译文

分析器	训练方式	WSJ 23 F1
Vinyals & Kaiser 等（2014）[37]	仅 WSJ，判别式	88.3
Petrov 等（2006）[29]	仅 WSJ，判别式	90.4
Zhu 等（2013）[40]	仅 WSJ，判别式	90.4
Dyer 等（2016）[8]	仅 WSJ，判别式	91.7
Transformer（4层）	仅 WSJ，判别式	91.3
Zhu 等（2013）[40]	半监督	91.3
Huang & Harper（2009）[14]	半监督	91.3
McClosky 等（2006）[26]	半监督	92.1
Vinyals & Kaiser 等（2014）[37]	半监督	92.1
Transformer（4层）	半监督	92.7
Luong 等（2015）[23]	多任务	93.0
Dyer 等（2016）[8]	生成式	93.3

我们将最大输出长度增加到输入长度 + 300。对于仅 WSJ 和半监督设置，我们均使用束大小21以及 $\alpha = 0.3$。

表4中的结果表明，尽管缺乏针对任务的调优，我们的模型仍表现得出乎意料地好，所得结果优于此前报告的所有模型，唯一例外是循环神经网络语法模型[8]。

与 RNN 序列到序列模型[37]相比，即使仅在包含4万句子的 WSJ 训练集上训练，Transformer 仍优于 BerkeleyParser[29]。

7 结论

在本文中，我们介绍了 Transformer，这是第一个完全基于注意力机制的序列转换模型，用多头自注意力替代了编码器—解码器架构中最常用的循环层。

对于翻译任务，Transformer 的训练速度显著快于基于循环层或卷积层的架构。在 WMT 2014 英语—德语和 WMT 2014 英语—法语翻译任务上，我们均取得了新的最先进结果。在前一个任务中，我们的最佳模型甚至超过了此前报告的所有集成模型。

我们对基于注意力的模型的未来感到振奋，并计划将其应用于其他任务。我们计划将 Transformer 扩展到输入和输出模态不是文本的问题，并研究局部的、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和输出。使生成过程更少依赖顺序执行是我们的另一项研究目标。

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 提出的富有成效的意见、修正与启发。

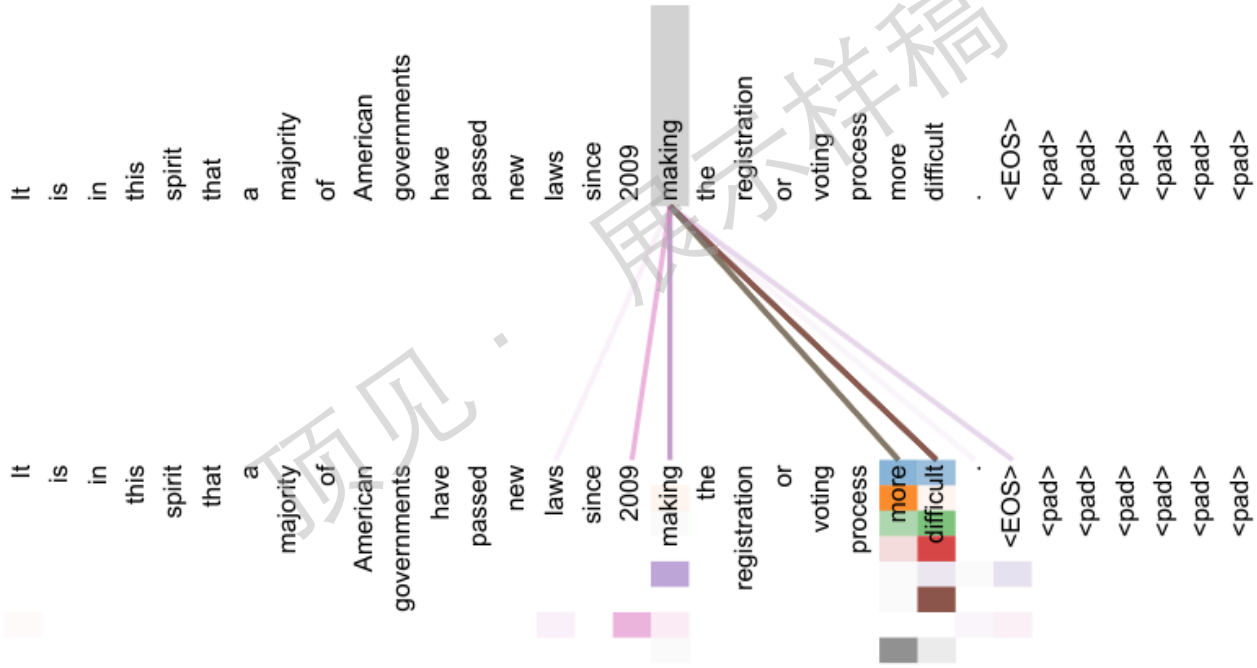

图3：编码器第5层（共6层）的自注意力中跟踪长距离依赖关系的注意力机制示例。许多注意力头关注动词“making”的远距离依赖关系，从而补全短语“making...more difficult”。此处仅展示针对词“making”的注意力。不同颜色表示不同的头。彩色查看效果最佳。

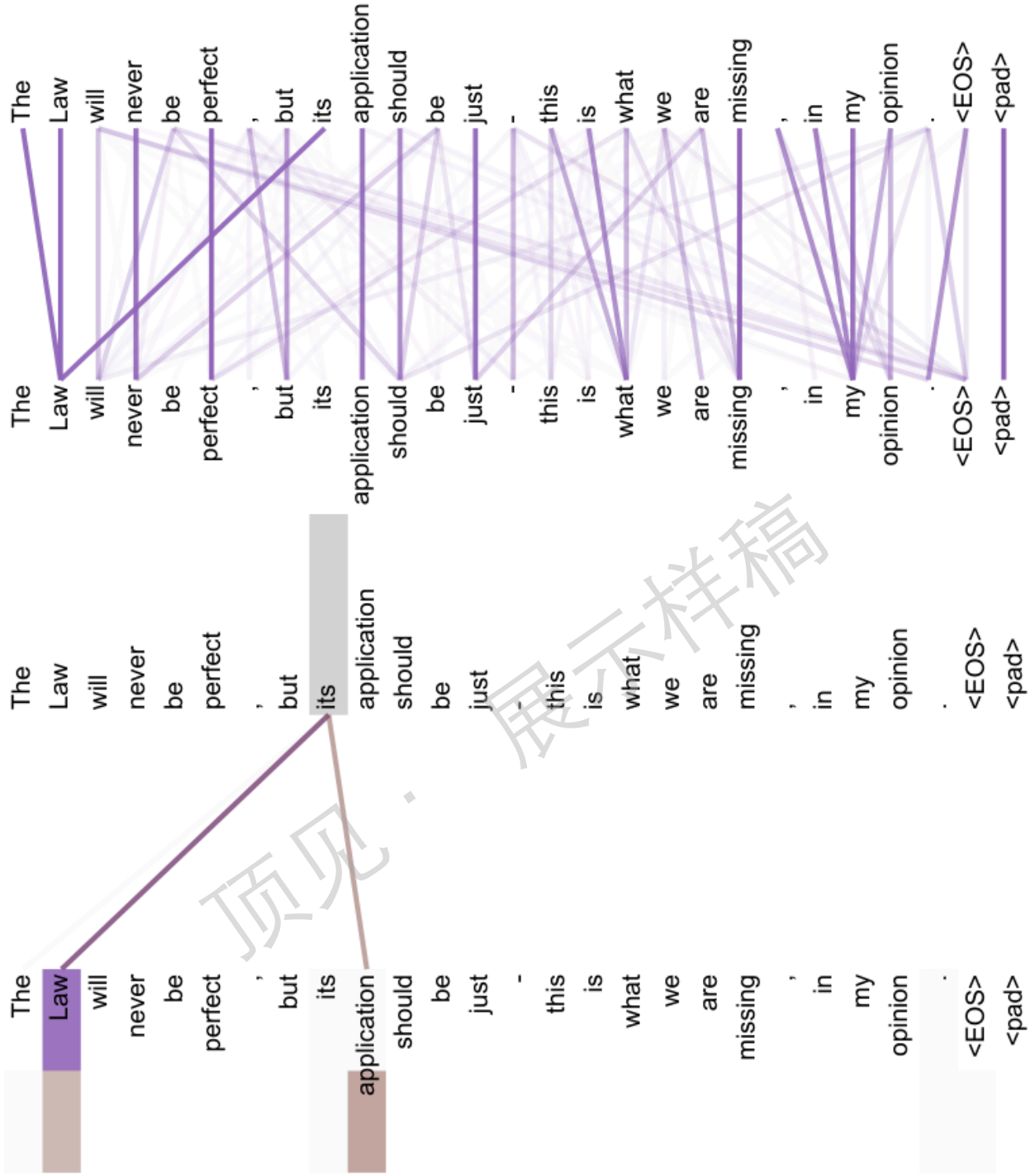

图4：两个同样位于第5层（共6层）的注意力头，显然参与了回指消解。上：第5个头的完整注意力。下：仅针对词“its”的第5和第6个注意力头的孤立注意力。注意，对于该词，注意力非常尖锐。

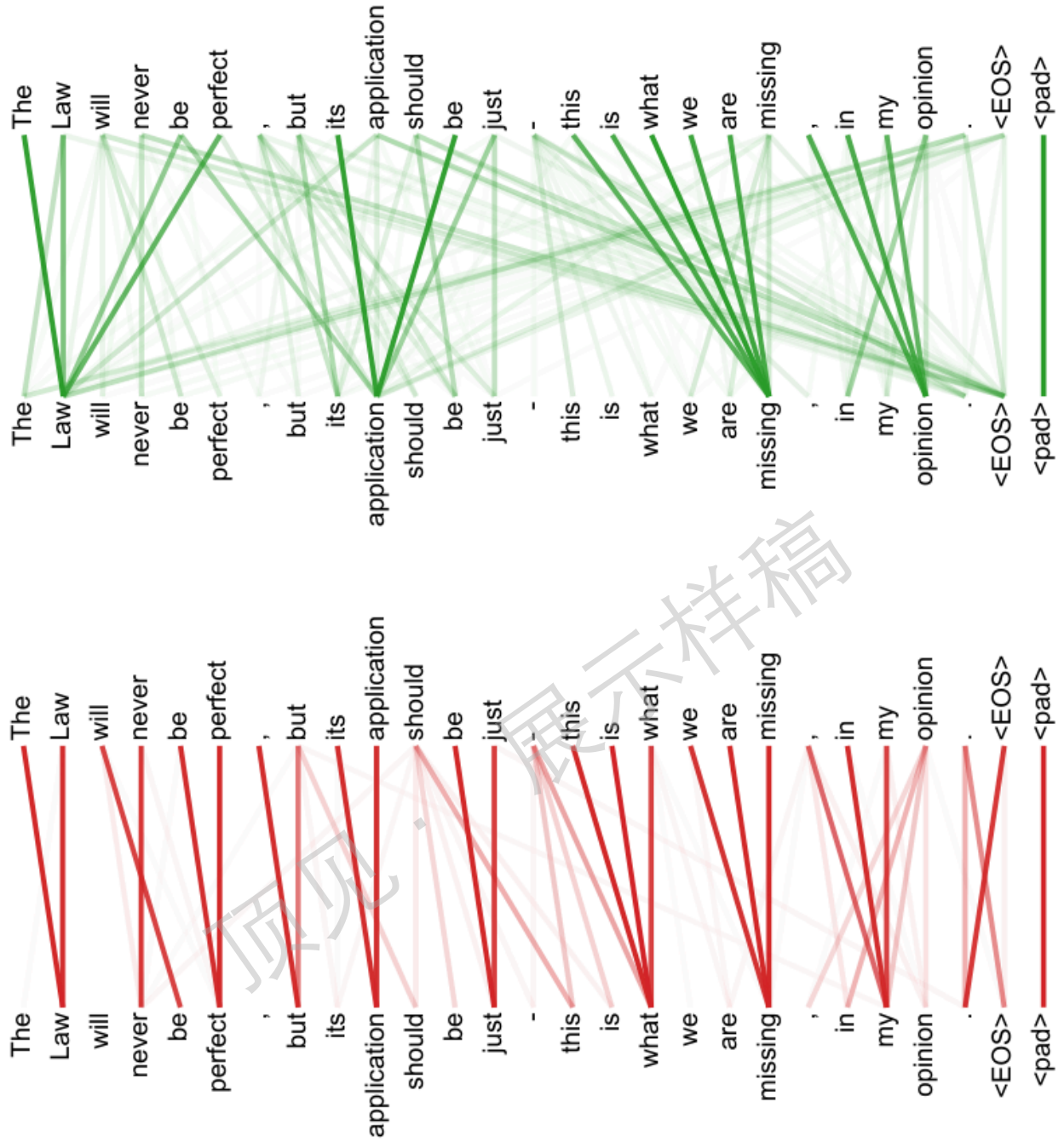

图5：许多注意力头表现出似乎与句子结构相关的行为。上面给出了两个此类示例，分别来自编码器第5层（共6层）自注意力中的两个不同注意力头。这些头显然学会了执行不同的任务。

参考文献

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 《层归一化》[Layer normalization]. arXiv preprint arXiv:1607.06450, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio.《通过联合学习对齐与翻译的神经机器翻译》[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.《神经机器翻译架构的大规模探索》[Massive exploration of neural machine translation architectures]. CoRR, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata.《用于机器阅读的长短期记忆网络》[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.《使用RNN编码器-解码器学习短语表示用于统计机器翻译》[Learning phrase representations using rnn encoder-decoder for statistical machine translation]. CoRR, abs/1406.1078, 2014.

[6] Francois Chollet.《Xception：基于深度可分离卷积的深度学习》[Xception: Deep learning with depthwise separable convolutions]. arXiv preprint arXiv:1610.02357, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio.《门控循环神经网络在序列建模上的实证评估》[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.《循环神经网络文法》[Recurrent neural network 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》[Convolu tional sequence to sequence learning]. arXiv preprint arXiv:1705.03122v2, 2017.

[10] Alex Graves.《使用循环神经网络生成序列》[Generating sequences with recurrent neural networks]. arXiv preprint arXiv:1308.0850, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.《用于图像识别的深度残差学习》[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》[Gradient flow in recurrent nets: the difficulty of learning long-term dependencies], 2001.

[13] Sepp Hochreiter and Jürgen Schmidhuber.《长短期记忆》[Long short-term memory]. Neural computation, 9(8):1735-1780, 1997.

[14] Zhongqiang Huang and Mary Harper. 《跨语言潜在标注的自训练PCFG文法》［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. 《探索语言建模的极限》［Exploring the limits of language modeling］. arXiv preprint arXiv:1602.02410, 2016.

[16] Łukasz Kaiser and Samy Bengio. 《主动记忆能替代注意力机制吗?》［Can active memory replace attention?］? In Advances in Neural Information Processing Systems, (NIPS), 2016.

[17] Łukasz Kaiser and Ilya Sutskever. 《神经GPU学习算法》［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 Ko ray Kavukcuoglu. 《线性时间的神经机器翻译》［Neural machine translation in linear time］. 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》［Structured attention networks］. In International Conference on Learning Representations, 2017.

[20] Diederik Kingma and Jimmy Ba. 《Adam：一种随机优化方法》［Adam: A method for stochastic optimization］. In ICLR, 2015.

[21] Oleksii Kuchaiev and Boris Ginsburg. 《LSTM网络的分解技巧》［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. 《一种结构化的自注意力句子嵌入》［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》［Multi-task sequence to sequence learning］. arXiv preprint arXiv:1511.06114, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 《基于注意力的神经机器翻译的有效方法》［Effective approaches to attention based neural machine translation］. 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》［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. 《有效的句法分析自训练》［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. 《一种可分解的注意力模型》［A decomposable attention model］. In Empirical Methods in Natural Language Processing, 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. 《用于抽象式摘要的深度强化模型》［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、紧凑且可解释的树标注》［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. 《利用输出嵌入改进语言模型》［Using the output embedding to improve language models］. arXiv preprint arXiv:1608.05859, 2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. 《基于子词单元的稀有词神经机器翻译》［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. 《超大规模神经网络：稀疏门控混合专家层》［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：一种防止神经网络过拟合的简单方法》［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》［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. 《使用神经网络的序列到序列学习》［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.《重新思考计算机视觉的Inception架构》[Rethinking the inception architecture for computer vision]. CoRR, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton.《将文法视为外语》[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.《谷歌的神经机器翻译系统：弥合人工翻译与机器翻译之间的差距》[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.《用于神经机器翻译的具有前馈连接的深度循环模型》[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.《快速准确的移进-归约成分句法分析》[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.