关注就是一切

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

摘要

主流的序列转换模型基于复杂的递归或卷积神经网络，其中包括一个编码器和一个解码器。 性能最好的模型还通过注意力机制将编码器和解码器连接起来。 我们提出了一种新的简单网络架构--"转换器"（Transformer），它完全基于注意力机制，无需递归和卷积。 在两项机器翻译任务上的实验表明，这些模型的质量更优，同时可并行化程度更高，所需的训练时间也大大减少。 我们的模型在 WMT 2014 英德翻译任务中达到了 28.4 BLEU，比现有的最佳结果（包括集合）提高了 2 BLEU 以上。 在 WMT 2014 英法翻译任务中，我们的模型在 8 个 GPU 上训练了 3.5 天后，单个模型的 BLEU 得分达到了 41.8 分，是目前最先进的单个模型 BLEU 得分，而这只是文献中最佳模型训练成本的一小部分。 我们将 Transformer 成功应用于英语选区解析，并同时使用大量和有限的训练数据，从而证明 Transformer 可以很好地推广到其他任务中。

---

* 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

递归神经网络，尤其是长短期记忆[13]和门控递归[7]神经网络，已被牢固确立为序列建模和转译问题（如语言建模和机器翻译）的最先进方法[35, 2, 5]。 自此以后，许多人继续努力推动递归语言模型和编码器-解码器架构的发展[38, 24, 15]。

递归模型通常按照输入和输出序列的符号位置进行计算。 将位置与计算时间的步长对齐，它们会生成隐藏状态 ht 的序列，作为前一个隐藏状态 ht-1 和位置 t 的输入的函数。这种固有的序列性质排除了训练实例内的并行化，而在序列长度较长时，这一点变得至关重要，因为内存约束限制了跨实例的批处理。 最近的研究通过因式分解技巧[21]和条件计算[32]显著提高了计算效率，同时也提高了后者的模型性能。 然而，顺序计算的基本限制仍然存在。

注意机制已成为各种任务中令人信服的序列建模和转导模型的一个组成部分，可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，除了少数情况[27]，这种注意机制都是与递归网络结合使用的。

在这项工作中，我们提出了 "转换器"（Transformer）这一模型架构，它摒弃了递归机制，而是完全依赖注意力机制来绘制输入和输出之间的全局依赖关系。 Transformer 可以大大提高并行化程度，在 8 个 P100 GPU 上只需训练 12 个小时，翻译质量就能达到新的水平。

2 背景

减少顺序计算的目标也是扩展神经 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，将来自两个任意输入或输出位置的信号联系起来所需的运算次数随位置间距离的增加而增加，ConvS2S 为线性增加，ByteNet 为对数增加。 这就增加了学习远距离位置之间依赖关系的难度[12]。 在 Transformer 中，这将被减少到一个恒定的操作数，但代价是由于平均注意力加权位置而降低了有效分辨率，我们在 3.2 节中介绍了多头注意力来抵消这一影响。

自我注意（有时也称为内部注意）是一种注意机制，它将单个序列的不同位置联系起来，以计算序列的表征。 自我注意已成功应用于多种任务中，包括阅读理解、抽象概括、文本引申和学习与任务无关的句子表征 [4,27,28,22]。

端到端记忆网络基于递归注意机制，而不是序列对齐递归，在简单语言问题解答和语言建模任务中表现出色[34]。

然而，据我们所知，Transformer 是第一个完全依靠自我注意来计算输入和输出表示而不使用序列对齐 RNN 或卷积的转导模型。 在下面的章节中，我们将描述 Transformer，激发自我注意，并讨论它与 [17, 18] 和 [9] 等模型相比的优势。

3 模型结构

最具竞争力的神经序列转换模型通常采用编码器-解码器结构。 [5, 2, 35]. 在此结构中，编码器将输入符号序列 (x1, ..., xn) 映射为连续表示序列 z = (z1, ..., zn)。给定 z 之后，解码器则逐个生成输出符号序列 (y1, ..., ym)。在每一步中，模型是自回归的 [10]，即在生成下一个符号时，会将之前生成的符号作为额外输入进行消耗。

Transformer 遵循这种整体架构，使用堆叠的自注意力层和逐点全连接层（point-wise fully connected layers）来构建编码器和解码器，分别如图 1 的左半部分和右半部分所示。

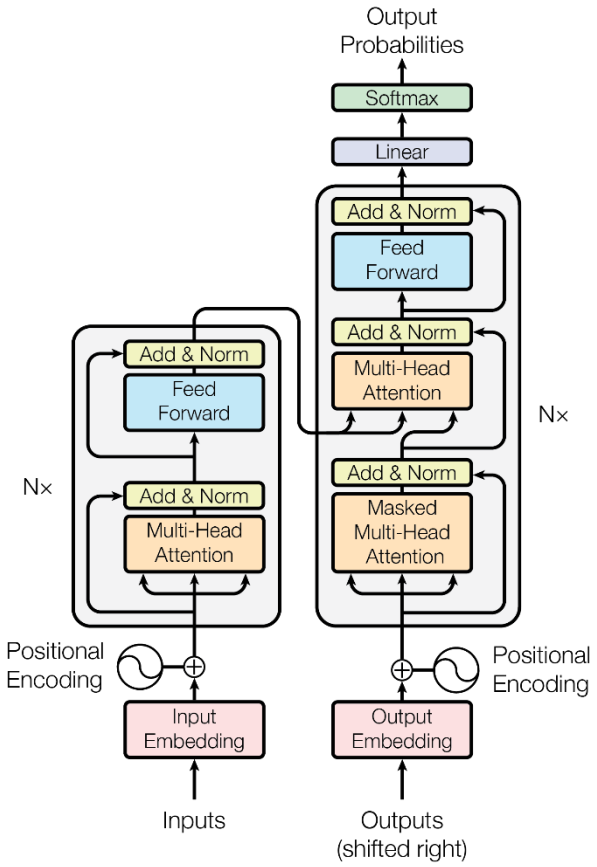

Figure 1: The Transformer - model architecture.

3.1 编码器和解码器堆栈

编码器 编码器由 N = 6 层相同的层堆叠组成。 每一层都有两个子层。 第一个是多头自注意机制，第二个是简单的位置全连接前馈网络。 我们在两个子层的每个周围都采用了残差连接[11]，然后进行层归一化[1]。 也就是说，每个子层的输出都是 LayerNorm(x + Sublayer(x))，其中 Sublayer(x) 是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会产生维数为 dmodel = 512 的输出。

解码器同样由 N = 6 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（multi-head attention）。与编码器类似，我们在每个子层周围使用残差连接，并随后进行层归一化。我们还修改了解码器堆栈中的自注意力子层，以防止位置关注后续的位置。这种掩蔽机制，结合输出嵌入向右偏移一个位置的事实，确保了位置 i 的预测只能依赖于位置小于 i 的已知输出。

3.2 注意机制

注意力函数可以被描述为将一个查询（query）和一组键值对（key-value pairs）映射到一个输出，其中查询、键、值和输出都是向量。输出是值的加权和，每个值的权重由查询与相应键之间的兼容性函数计算得出。

3.2.1 标度点积注意力

我们将这种特殊的注意力称为 "缩放点积注意力"（图 2）。 输入包括查询和维度为 dk 的键，以及维度为 √ dv 的值。 我们计算查询与所有密钥的点积，将每个点积除以 dk，然后应用软最大函数获得值的权重.

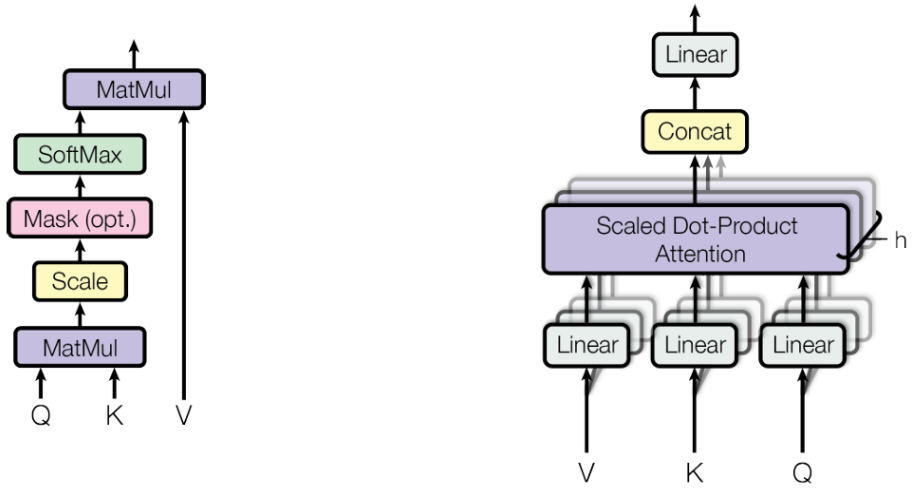

图 2：（左）标度点积注意力。 (右图）多头注意由多个并行运行的注意层组成。

在实践中，我们会同时计算一组查询的注意力函数，这些查询被打包成一个矩阵 Q，键和值也被打包成矩阵 K 和 V。 我们计算的输出矩阵为

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

最常用的两种注意函数是加法注意[2]和点积（乘法）注意。 点积注意力与我们的算法相同，$\frac{1}{\sqrt{d_k}}$的缩放因子除外。加法注意使用单隐层前馈网络计算相容函数。 虽然两者的理论复杂度相似，但点积注意力在实际应用中速度更快，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。

虽然在 dk 值较小的情况下，这两种机制的表现类似，但在 dk 值较大的情况下，加法注意比点积注意更胜一筹 [3]。 我们猜测，对于较大的 dk 值，点积的幅度会越来越大，从而将 softmax 函数推向梯度极小的区域。 为了消除这种影响，我们用以下方法缩放点积$\frac{1}{\sqrt{d_k}}$.

3.2.2 多头注意力机制

我们发现，与其使用 dmodel 维度的键、值和查询来执行单一的注意力函数，不如将查询、键和值分别线性投影到 dk、dk 和 dv 维度，并将不同的学习过的线性投影进行 h 次投影。 然后，我们在每个投影版本的查询、键和值上并行执行注意力函数，得到 dv 维的输出值。 如图 2 所示，这些值被串联起来并再次投影，从而得到最终值。

多头注意力允许模型在不同位置共同关注来自不同表征子空间的信息。 而在单注意头的情况下，平均化会抑制这一点。

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

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

在这项工作中，我们采用了 h = 8 个并行注意力层，或称为 "头"。 我们使用 dk = dv = dmodel/h = 64。 由于减少了每个头的维度，总计算成本与全维度的单头注意力相似。

3.2.3 注意力在我们模型中的应用

The Transformer 通过三种不同方式实现多头关注:

- 在 "编码器-解码器注意 "层中，查询来自前一个解码器层，而记忆键和记忆值则来自编码器的输出。 这样，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（如 [38, 2, 9]）中典型的编码器-解码器注意机制。

- 编码器包含自注意层。 在自注意层中，所有的键、值和查询都来自同一个地方，在这种情况下，就是编码器中上一层的输出。 编码器中的每个位置都可以关注编码器上一层的所有位置。
- 同样，解码器中的自关注层允许解码器中的每个位置关注解码器中包括该位置在内的所有位置。 我们需要防止解码器中的信息向左流动，以保持自动回归特性。 我们通过屏蔽（设置为-∞）软最大输入中对应非法连接的所有值，在缩放点积关注中实现了这一点。 见图 2。

3.3 定位前馈网络

除了注意力子层外，我们的编码器和解码器中的每一层都包含一个全连接的前馈网络，该网络分别对每个位置进行相同的处理。 这包括两个线性变换，中间有一个 ReLU 激活。

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

虽然不同位置的线性变换相同，但各层使用的参数不同。 另一种描述方法是两个内核大小为 1 的卷积。 输入和输出的维度为 dmodel = 512，内层的维度为 dff = 2048。

3.4 嵌入 and Softmax

与其他序列转换模型类似，我们使用学习到的嵌入将输入标记和输出标记转换为维数为 dmodel 的向量。 我们还使用通常的学习线性变换和 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，我们在两个嵌入层之间共享相同的权重矩阵和预软最大线性变换，类似于 [30]。 在嵌入层中，我们将这些权重乘以根号下(dmodel)。

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次数。其中，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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.3 位置编码

由于我们的模型不包含递归（recurrence）也不包含卷积（convolution），为了使模型能够利用序列中符号的顺序信息，我们必须向模型注入一些关于序列中符号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的底部向输入嵌入添加“位置编码”

（positional encodings）。位置编码与嵌入具有相同的维度 dmodel，以便两者可以相加。位置编码有多种选择，包括学习得到的位置编码和固定的位置编码 [9]。

在这项工作中，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 是位置，i 是维数。 也就是说，位置编码的每个维度对应一个正弦波。 波长形成一个从 2π 到 10000-2π 的几何级数。 我们之所以选择这个函数，是因为我们假设它可以让模型轻松学会按相对位置进行关注，因为对于任何固定的偏移 k，PEpos+k 都可以表示为 PEpos 的线性函数。

我们还尝试了使用学习到的位置嵌入[9]，发现两个版本产生的结果几乎相同（见表 3 第（E）行）。 我们之所以选择正弦波版本，是因为它可以让模型推断出比训练时遇到的序列长度更长的序列。

4 为什么要自注意力机制

在本节中，我们将比较自注意层与递归层和卷积层的各个方面，后者通常用于将一个可变长度的符号表示序列 (x1,...xn) 映射到另一个等长序列 (z1,...,zn)，xi,zi∈Rd，例如典型序列转换编码器或解码器中的隐藏层。 我们使用自注意的动机有三个。

一是每层的总计算复杂度。 另一个是可并行化的计算量，以所需的最小顺序运算次数来衡量。

第三是网络中长距离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。 影响学习此类依赖关系能力的一个关键因素是前向和后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，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，自注意层连接所有位置的连续操作数不变，而递归层则需要 O(n) 次连续操作。 就计算复杂度而言，当序列长度 n 小于表示维度 d 时，自注意层的计算速度要快于递归层，机器翻译中最先进的模型所使用的句子表示，如单词片[38]和字节对[31]表示，通常就是这种情况。 为了提高涉及超长序列任务的计算性能，可以限制自我关注只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。 这将把最大路径长度增加到 O(n/r)。 我们计划在今后的工作中进一步研究这种方法。

核宽 k < n 的单个卷积层无法连接所有输入和输出位置对。 在内核连续的情况下，这样做需要堆叠 O(n/k) 个卷积层；在扩张卷积的情况下，则需要堆叠 O(logk(n) 个卷积层[18]，从而增加了网络中任意两个位置之间最长路径的长度。 可分离卷积[6]则大大降低了复杂度，达到 O(k - n - d + n - d2)。 即使在 k = n 的情况下，可分离卷积的复杂度也相当于自注意层和点式前馈层的组合，而我们在模型中采用的正是这种方法。

作为附带的好处，自我关注可以产生更多可解释的模型。 我们将从模型中检查注意力分布，并在附录中介绍和讨论实例。 各个注意力头不仅明显学会执行不同的任务，而且许多注意力头似乎还表现出与句子的句法和语义结构有关的行为。

5 训练

本节将介绍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，该数据集包含约 450 万个句子对。 句子使用字节对编码[3]进行编码，其中有大约 37000 个共享的源目标词汇。 对于英语-法语，我们使用了规模更大的 WMT 2014 英语-法语数据集，该数据集包含 3600 万个句子，并将标记拆分为 32000 个词块词汇[38]。 句对按大致序列长度分组。 每个训练批包含一组句对，其中包含约 25000 个源词块和 25000 个目标词块。

5.2 硬件和时间表

我们在一台配备 8 个 NVIDIA P100 GPU 的机器上训练了我们的模型。对于使用文中描述的超参数的基础模型，每个训练步骤大约耗时 0.4 秒。我们总共训练了基础模型 100,000 步，即 12 小时。对于较大的模型（如表 3 最后一行所述），每步训练时间是 1.0 秒。较大模型的训练总步数为 300,000 步，耗时 3.5 天。

5.3 优化器

我们使用了 adam 优化器[20]， β 1 = 0.9， β 2 = 0.98，ϵ = 10-9。 在训练过程中，我们根据公式改变学习率：

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

这相当于在第一个热身训练步数中线性地提高学习率，之后则按步数的平方根反比例降低学习率。 我们使用的是 warmup_steps = 4000。

5.4 正则化

在训练过程中，我们采用了三种正则化方法：

表 2：在英语到德语和英语到法语的 newstest2014 测试中，Transformer 的 BLEU 分数优于之前的先进模型，而训练成本仅为之前的一小部分。

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

残差滤波 我们对每个子层的输出进行滤波 [33]，然后将其添加到子层输入中并进行归一化处理。 此外，我们还对编码器和解码器堆栈中的嵌入和位置编码之和进行了滤除。 在基础模型中，我们使用 Pdrop = 0.1 的比率。

标签平滑 在训练过程中，我们采用了值 $\epsilon ls = 0.1$ 的标签平滑[36]。 这样做会增加模型的不确定性，从而降低复杂度，但却能提高准确度和 BLEU 得分。

6 结果

6.1 机器翻译

在 WMT 2014 英-德翻译任务中，大转换器模型（表 2 中的转换器（big））的 BLEU 值比之前报道的最佳模型（包括集合）高出 2.0 以上，达到了 28.4 的新的最先进 BLEU 值。该模型的配置见表 3 底行。 8 个 P100 GPU 的训练耗时 3.5 天。 即使是我们的基本模型，也超越了之前发布的所有模型和集合，而训练成本只是任何竞争模型的一小部分。

在 WMT 2014 英语到法语的翻译任务中，我们的大模型获得了 41.0 的 BLEU 分数，超过了之前发布的所有单一模型，而训练成本还不到之前最先进模型的 1/4。 为英译法而训练的 Transformer（大）模型使用了 Pdrop = 0.1，而不是 0.3。

对于基本模型，我们使用的是通过平均最近 5 个检查点得到的单一模型，这些检查点以 10 分钟的间隔写入。 对于大型模型，我们取最后 20 个检查点的平均值。 我们使用波束搜索，波束大小为 4，长度惩罚 $\alpha = 0.6$ [38]。 这些超参数是在开发集上实验后选择的。 我们将推理过程中的最大输出长度设定为输入长度 + 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 数量和每个 GPU 的持续单精度浮点运算能力的估计值相乘，估算出训练一个模型所使用的浮点运算次数。

表 3：转换器架构的变体，未列出的值与基础模型的值相同，所有指标均为英译德翻译开发集，2013 年新测试，列出的复杂度是按字片段计算的，根据我们的字节怀表编码，不应与按字数计算的复杂度相比较。

	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。开发集 newstest2013 上英译德性能的变化。我们使用了上一节所述的波束搜索，但没有使用检查点平均法。 表 3 列出了这些结果。

如第 3.2.2 节所述，在表 3 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，但计算量保持不变。 虽然单头注意力比最佳设置差 0.9 BLEU，但注意力头数过多也会导致质量下降。

在表 3 行（B）中，我们发现减小关注键大小 dk 会降低模型质量。 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更有益处。 我们在（C）行和（D）行进一步观察到，正如预期的那样，模型越大越好，而 dropout 对避免过度拟合很有帮助。 在第（E）行中，我们用学习到的位置嵌入[9]替换了正弦位置编码，观察到的结果与基础模型几乎完全相同。

6.3 英语成分句法分析

为了评估 Transformer 是否可以推广到其他任务，我们在英语成分句法分析（English Constituency Parsing）上进行了实验。该任务提出了特定的挑战：输出受到强烈的结构约

表 4：转换器在英语选区解析方面具有良好的通用性（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

束，并且显著长于输入。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]。

我们在宾州树库（Penn Treebank）[25]中的华尔街日报（WSJ）部分，大约 40,000 个训练句子上，训练了一个 4 层的 Transformer 模型，模型维度 dmodel = 1024。我们还在半监督设置下进行了训练，使用了更大规模的高置信度语料库和 BerkeleyParser 语料库，包含大约 1700 万个句子 [37]。对于仅使用 WSJ 的设置，我们使用了 16,000 个词汇单元；而在半监督设置下，则使用了 32,000 个词汇单元。

我们仅在第 22 节开发集上进行了少量实验，以选择注意力和残差（第 5.4 节）、学习率和波束大小，所有其他参数均与英译德基础翻译模型保持一致。 在推理过程中，我们将最大输出长度增加到输入长度 + 300。 在仅 WSJ 和半监督设置中，我们使用的波束大小均为 21，$\alpha = 0.3$。

表 4 中的结果表明，尽管缺乏针对特定任务的调整，但我们的模型表现出人意料的好，其结果优于除递归神经网络语法 [8] 以外的所有以前报道过的模型。

与 RNN 序列到序列模型[37]相比，Transformer 的性能优于 BerkeleyParser [29]，即使仅在由 40K 个句子组成的 WSJ 训练集上进行训练也是如此。

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的序列转换模型，用多头自我注意力取代了编码器-解码器架构中最常用的递归层。

对于翻译任务，Transformer 的训练速度明显快于基于递归层或卷积层的架构。 在 WMT 2014 英译德和 WMT 2014 英译法翻译任务上，我们都达到了新的技术水平。 在前一项任务中，我们的最佳模型甚至优于之前报告的所有集合。

我们对基于注意力的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的输入和输出模式的问题上，并研究局部、受限的注意力机制，以有效处理大型输入和输出，如图像、音频和视频。 我们的另一个研究目标是减少生成的顺序。

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

感谢 Nal Kalchbrenner 和 Stephan Gouws 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 attentionbased 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

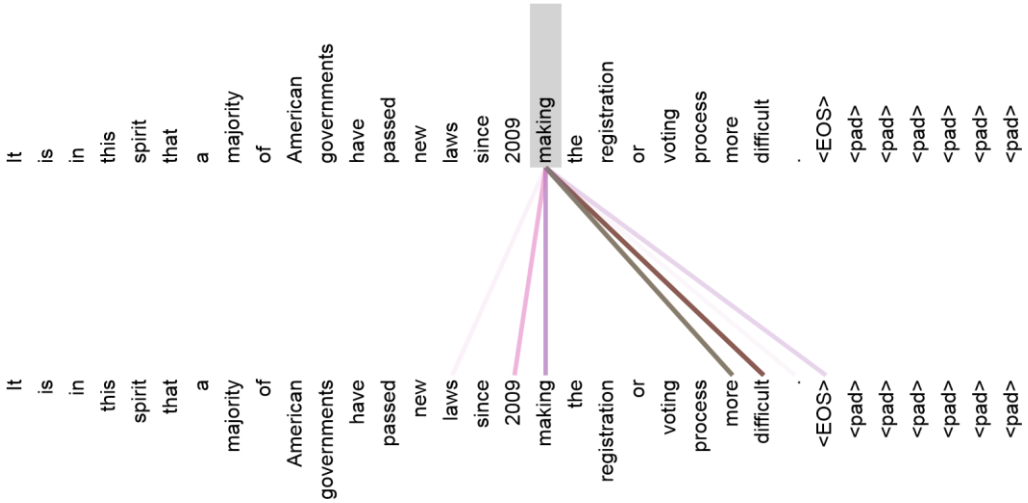

图 3：第六层中第五层编码器自我注意中长距离依赖关系的注意机制示例。 许多注意头注意动词 "making "的远距离依赖关系，完成短语 "making...more difficult"。 此处显示的注意力只针对 "制造 "一词。 不同的颜色代表不同的注意头。 最好用彩色观看。

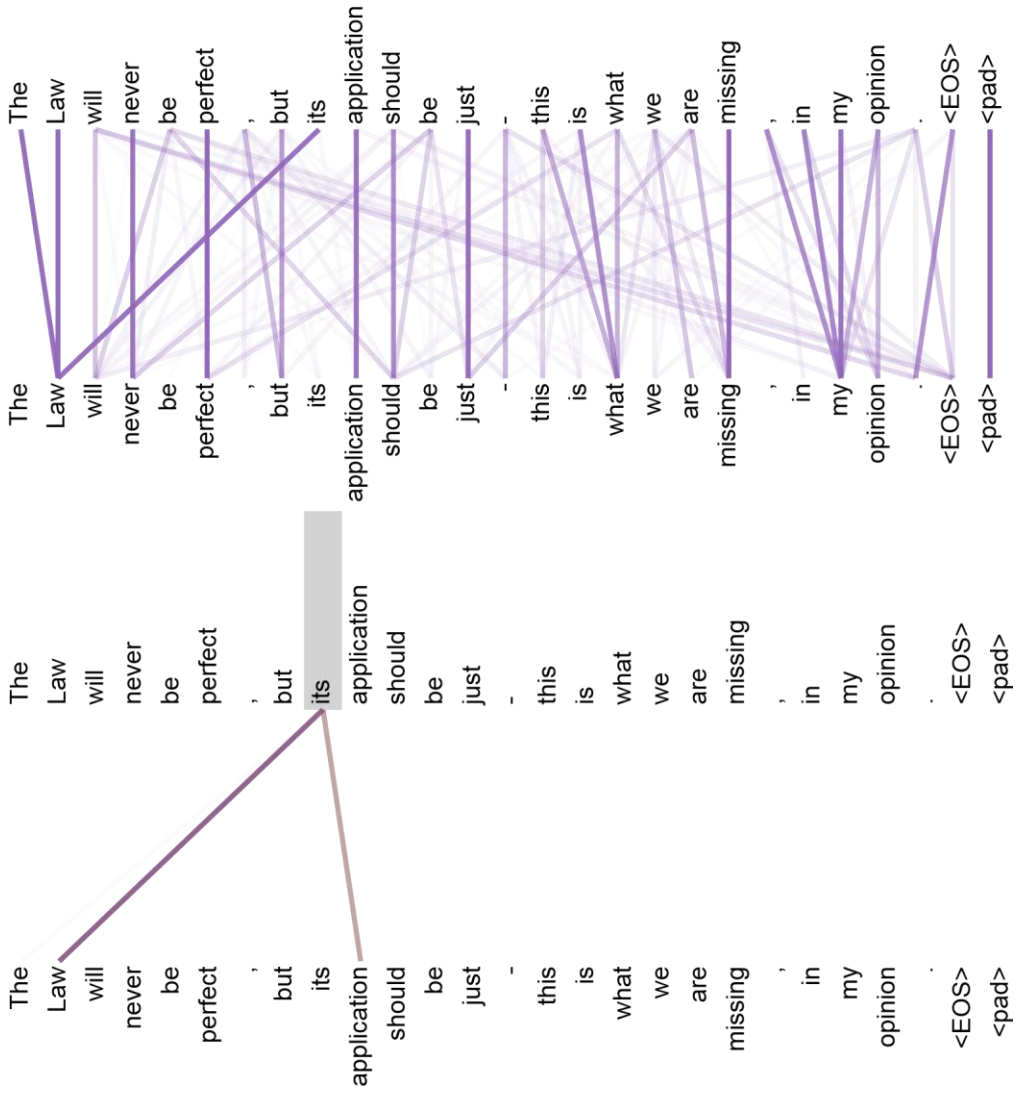

图 4：同样位于 6 层第 5 层的两个注意头显然参与了拟声词的解决。 上图：注意头 5 的全部注意力。 下图 第 5 和第 6 注意头仅对 "its "一词的单独注意。 请注意，这个词的注意力非常敏锐。

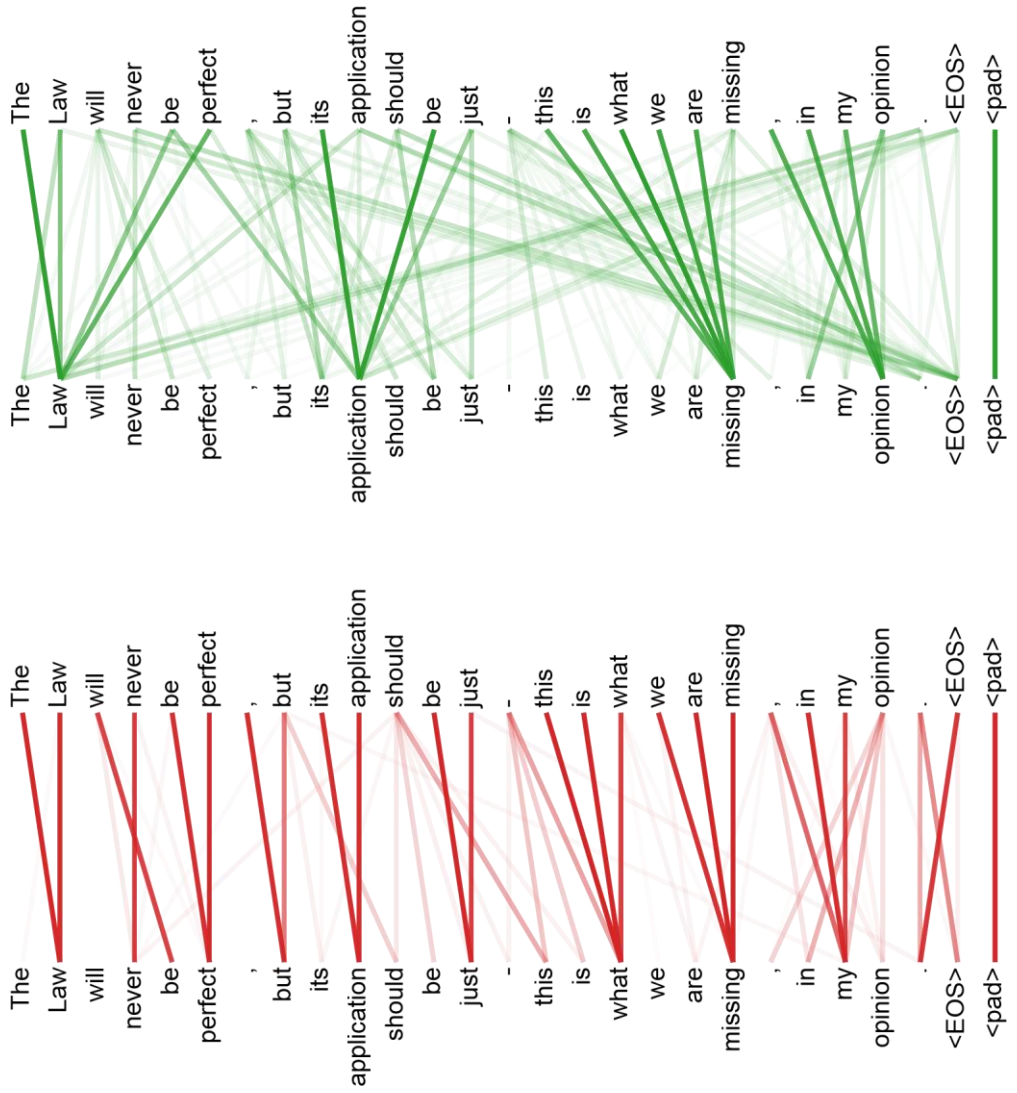

图 5：许多注意头的行为似乎与句子结构有关。 我们在上面给出了两个这样的例子，它们来自第 5 层第 6 层编码器自我注意中的两个不同的注意头。