Attention Is All You Need

摘要

主流序列转导模型基于复杂的循环神经网络或卷积神经网络，这些神经网络包含一个编码器和一个解码器。 性能最好的模型还通过attention机制将编码器和解码器连接起来。 我们提出一种新的简单的网络架构Transformer，仅基于attention机制并完全避免循环和卷积。 对两个机器翻译任务的实验表明，这些模型在质量上更加优越、并行性更好并且需要的训练时间显著减少。 我们的模型在WMT 2014英语-德语翻译任务上达到28.4 BLEU，超过现有最佳结果（包括整合模型）2个BLEU。 在WMT 2014英语-法语翻译任务中，我们的模型建立了单模型新的最先进的BLEU分数41.8，它在8个GPU上训练了3.5天，这个时间只是目前文献中记载的最好的模型训练成本的一小部分。通过在解析大量训练数据和有限训练数据的两种情况下将其应用到English constituency，我们表明Transformer可以很好地推广到其他任务。

1 简介

在序列建模和转换问题中，如语言建模和机器翻译[35, 2, 5]，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，已经被确立为最先进的方法。 自那以后，许多努力一直在推动循环语言模型和编码器-解码器架构的界限[38, 24, 15]。

循环模型通常是对输入和输出序列的符号位置进行因子计算。 通过在计算期间将位置与步骤对齐，它们根据前一步的隐藏状态h_{t-1}和输入产生位置t的隐藏状态序列h_t。这种固有的顺序特性阻碍样本训练的并行化，这在更长的序列长度上变得至关重要，因为有限的内存限制样本的批次大小。 最近的工作通过巧妙的因子分解[21]和条件计算[32]在计算效率方面取得重大进展，后者还同时提高了模型性能。 然而，顺序计算的基本约束依然存在。

在各种任务中，attention机制已经成为序列建模和转导模型不可或缺的一部分，它可以建模依赖关系而不考虑其在输入或输出序列中的距离[2, 19]。 除少数情况外[27]，这种attention机制都与循环网络一起使用。

在这项工作中我们提出Transformer，这种模型架构避免循环并完全依赖于attention机制来绘制输入和输出之间的全局依赖关系。 Transformer允许进行

更多的并行化，并且可以在八个P100 GPU上接受少至十二小时的训练后达到翻译质量的新的最佳结果。

2 背景

减少顺序计算的目标也构成扩展的神经网络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，关联任意两个输入和输出位置的信号所需的操作次数会随着位置之间的距离而增加，ConvS2S是线性增加，而ByteNet是对数增加。 这使得学习远距离位置之间的依赖关系变得更加困难[12]。 在Transformer中，这中操作减少到固定的次数，尽管由于对用attention权重化的位置取平均降低了效果，但是我使用Multi-Head Attention进行抵消，具体描述见 3.2。

Self-attention，有时称为intra-attention，是一种attention机制，它关联单个序列的不同位置以计算序列的表示。 Self-attention已成功用于各种任务，包括阅读理解、摘要概括、文本蕴涵和学习与任务无关的句子表征[4, 27, 28, 22]。

端到端的内存网络基于循环attention机制，而不是序列对齐的循环，并且已被证明在简单语言的问题回答和语言建模任务中表现良好[34]。

然而，就我们所知，Transformer是第一个完全依靠self-attention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、引出self-attention并讨论它相对[17, 18]和[9]几个模型的优势。

3 模型架构

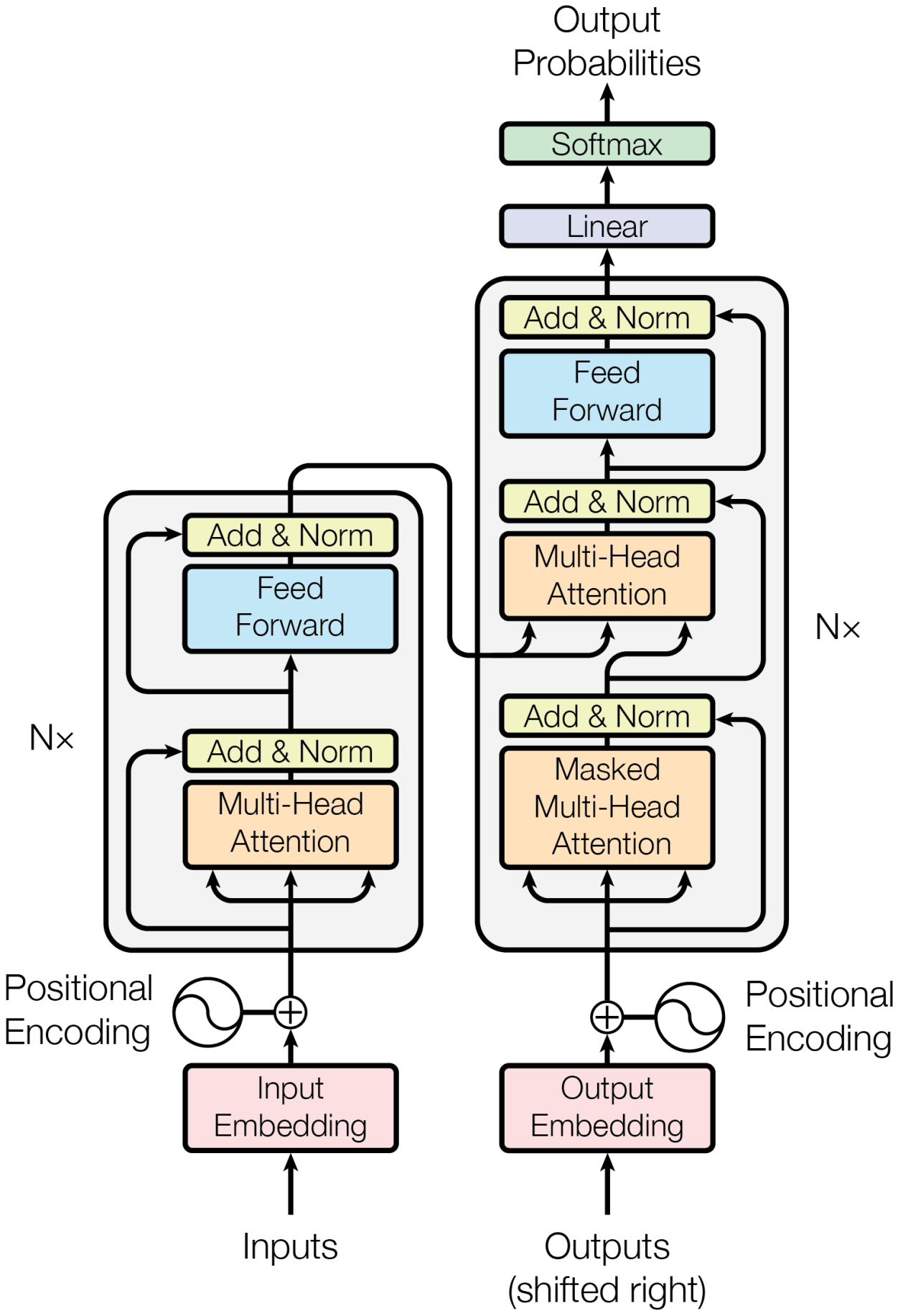


图1： Transformer — 模型架构。

大部分神经序列转导模型都有一个编码器-解码器结构[5, 2, 35]。 这里，编码器映射一个用符号表示的输入序列($x_1,...,x_n$) 到一个连续的表示$\mathbf{z} = (z_1,...,z_n)$。 根据$\mathbf{z}$，解码器生成符号的一个输出序列($y_1,...,y_m$) ，一次一个元素。 在每一步中，模型都是自回归的[10]，当生成下一个时，消耗先前生成的符号作为附加输入。Transformer遵循这种整体架构，编码器和解码器都使用self-attention堆叠和point-wise、完全连接的层，分别显示在图1的左边和右边。

3.1 编码器和解码器堆栈

编码器： 编码器由$N = 6$ 个完全相同的层堆叠而成。 每一层都有两个子层。 第一层是一个multi-head self-attention机制，第二层是一个简单的、位置完全连接的前馈网络。 我们对每个子层再采用一个残差连接[11] ，接着进行层标准化[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产生的输出维度都为$d$model = 512。

解码器： 解码器同样由$N = 6$ 个完全相同的层堆叠而成。 除了每个编码器层中的两个子层之外，解码器还插入第三个子层，该层对编码器堆栈的输出执行multi-head attention。 与编码器类似，我们在每个子层再采用残差连接，然后进行层标准化。 我们还修改解码器堆栈中的self-attention子层，以防止位置关注到后面的位置。 这种掩码结合将输出嵌入偏移一个位置，确保对位置的预测 i只能依赖小于i的已知输出。

3.2 Attention

Attention函数可以描述为将query和一组key-value对映射到输出，其中query、key、value和输出都是向量。 输出为value的加权和，其中分配给每个value的权重通过query与相应key的兼容函数来计算。

3.2.1 缩放版的点积attention

我们称我们特殊的attention为“缩放版的点积attention” （图 2）。 输入由query、d_k 维的key和d_v 维的value组成。 我们计算query和所有key的点积、用

$\sqrt{d_k}$

相除，然后应用一个softmax函数以获得值的权重。

在实践中，我们同时计算一组query的attention函数，并将它们组合成一个矩阵Q。 key和value也一起打包成矩阵 K和 V。 我们计算输出矩阵为：

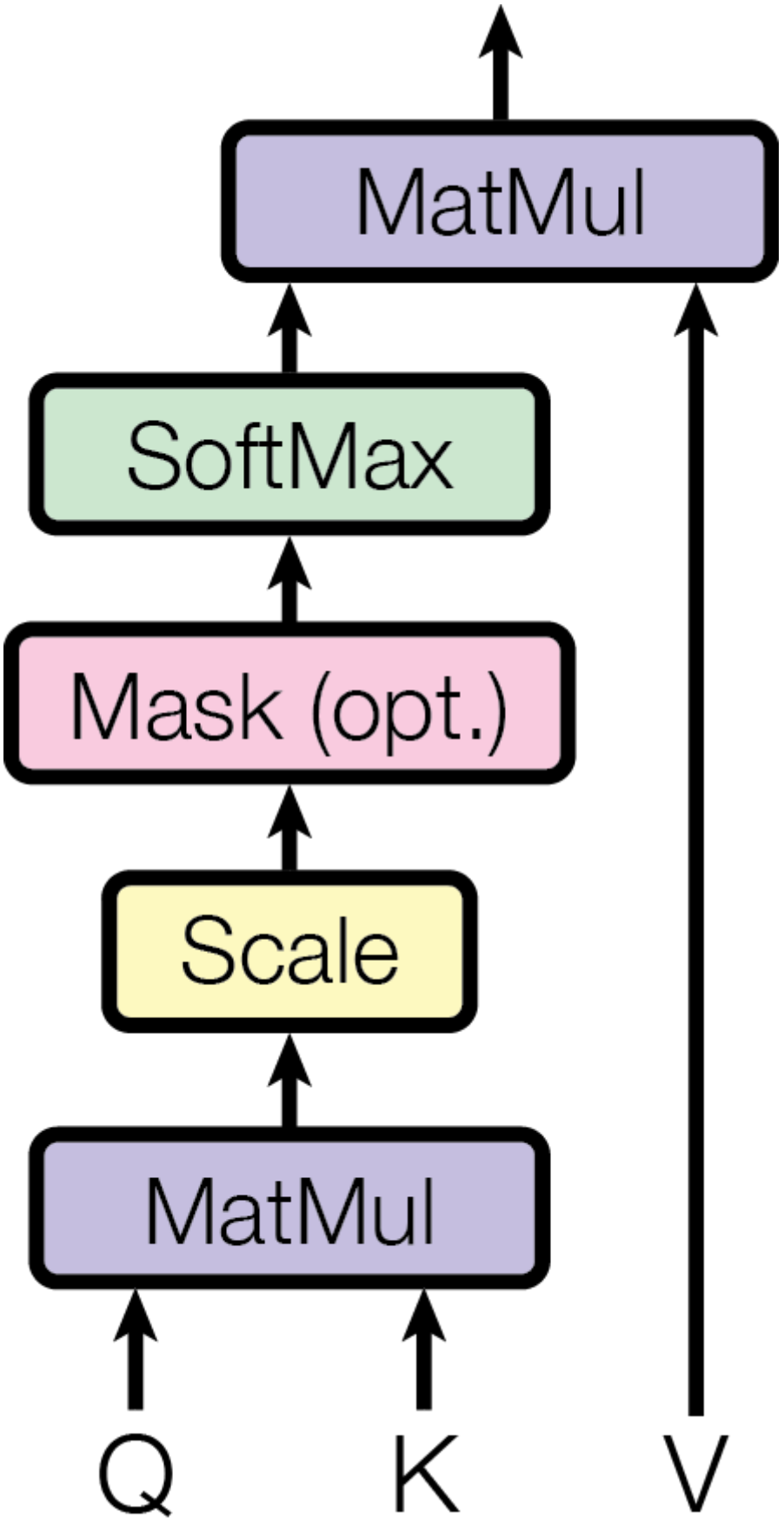
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

两个最常用的attention函数是加法attention[2]和点积（乘法）attention。 除了缩放因子
$\frac{1}{\sqrt{d_k}}$
之外，点积attention与我们的算法相同。 加法attention使用具有单个隐藏层的前馈网络计算兼容性函数。 虽然两者在理论上的复杂性相似，但在实践中点积attention的速度更快、更节省空间，因为它可以使用高度优化的矩阵乘法代码来实现。

当d_k的值比较小的时候，这两个机制的性能相差相近，当d_k比较大时，加法attention比不带缩放的点积attention性能好[3]。 我们怀疑，对于很大的d_k值，点积大幅度增长，将softmax函数推向具有极小梯度的区域4。 为了抵消这种影响，我们缩小点积
$\frac{1}{\sqrt{d_k}}$
倍。

3.2.2 Multi-Head Attention

缩放版的点积Attention



Multi-Head Attention

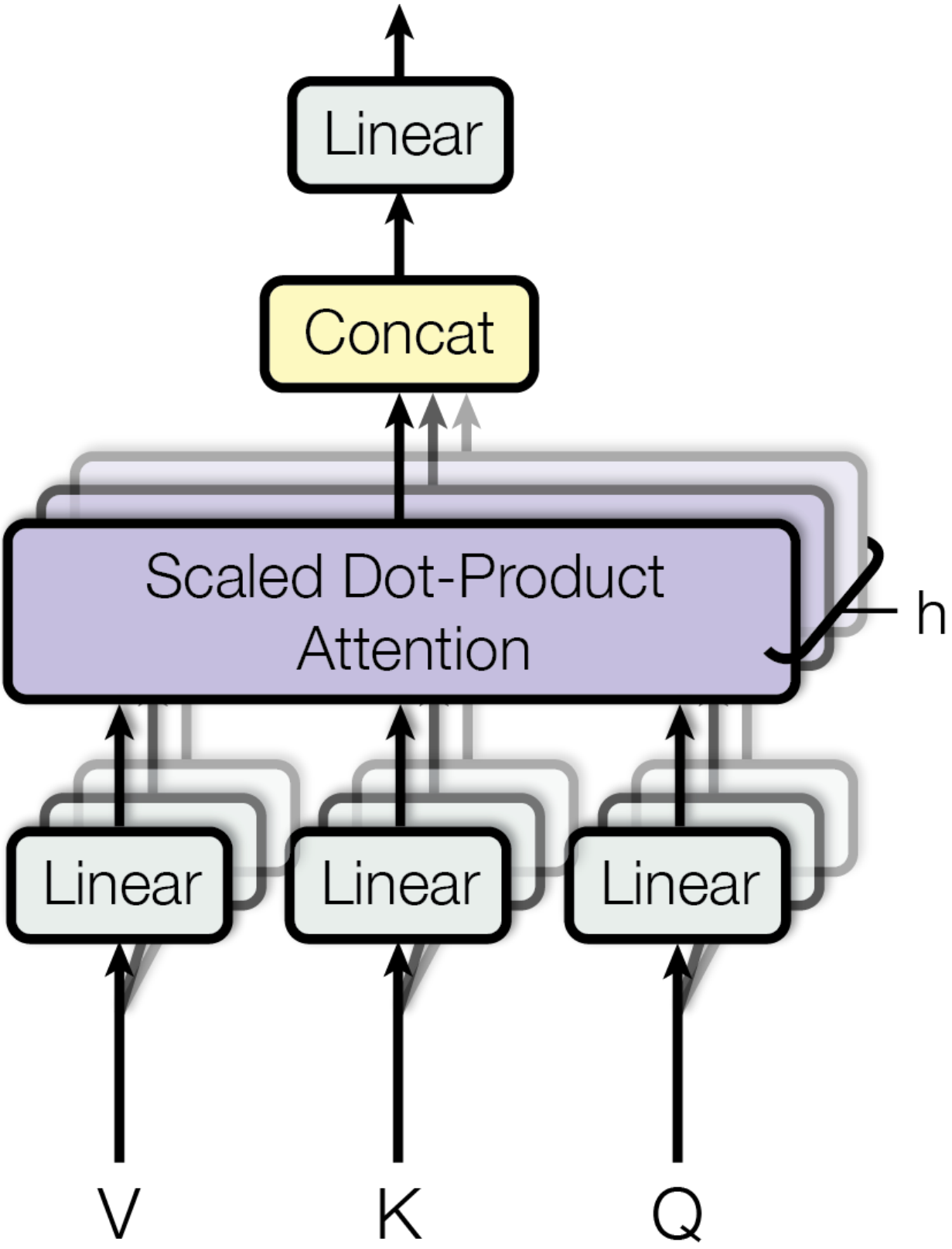

Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

图2： （左）缩放版的点积attention。 （右）Multi-Head Attention，由多个并行运行的attention层组成。

我们发现将query、key和value分别用不同的、学到的线性映射h倍到d_k、d_k和d_v维效果更好，而不是用d model维的query、key和value执行单个attention函数。 基于每个映射版本的query、key和value，我们并行执行attention函数，产生d_v维输出值。 将它们连接并再次映射，产生最终值，如图所示 2。

Multi-head attention允许模型的不同表示子空间联合关注不同位置的信息。 如果只有一个attention head，它的平均值会削弱这个信息。

MultiHead(Q,K,V)	= Concat(head1,...,headh)WO		
其中 headi	= Attention($QWiQ$, $KWiK$, $VWiV$)		

其中，映射为参数矩阵$WiQ \in \mathbb{R}d\text{model}\times dk$，$WiK \in \mathbb{R}d\text{model}\times dk$，$WiV \in \mathbb{R}d\text{model}\times dv$及$WO \in \mathbb{R}hdv\times d\text{model}$。

在这项工作中，我们采用$h = 8$个并行attention层或head。 对每个head，我们使用$dk = dv = d\text{model}/h = 64$。 由于每个head的大小减小，总的计算成本与具有全部维度的单个head attention相似。

3.2.3 Attention在我们的模型中的应用

Transformer使用以3种方式使用multi-head attention：

- 在“编码器—解码器attention”层，query来自上面的解码器层，key和value来自编码器的输出。 这允许解码器中的每个位置能关注到输入序列中的所有位置。 这模仿序列到序列模型中典型的编码器—解码器的attention机制，例如[38, 2, 9]。
- 编码器包含self-attention层。 在self-attention层中，所有的key、value和query来自同一个地方，在这里是编码器中前一层的输出。 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。
- 类似地，解码器中的self-attention层允许解码器中的每个位置都关注解码器中直到并包括该位置的所有位置。 我们需要防止解码器中的向左信

息流来保持自回归属性。 通过屏蔽softmax的输入中所有不合法连接的值（设置为-∞），我们在缩放版的点积attention中实现。 见图 2.

3.3 基于位置的前馈网络

除了attention子层之外，我们的编码器和解码器中的每个层都包含一个完全连接的前馈网络，该前馈网络单独且相同地应用于每个位置。 它由两个线性变换组成，之间有一个ReLU激活。

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

尽管线性变换在不同位置上是相同的，但它们层与层之间使用不同的参数。 它的另一种描述方式是两个内核大小为1的卷积。 输入和输出的维度为$d$model = 512，内部层的维度为d_{ff} = 2048。

3.4 嵌入和Softmax

与其他序列转导模型类似，我们使用学习到的嵌入将输入词符和输出词符转换为维度为$d$model的向量。 我们还使用普通的线性变换和softmax函数将解码器输出转换为预测的下一个词符的概率。 在我们的模型中，两个嵌入层之间和pre-softmax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

$\sqrt{d_{\text{model}}}$

。

3.5 位置编码

由于我们的模型不包含循环和卷积，为了让模型利用序列的顺序，我们必须注入序列中关于词符相对或者绝对位置的一些信息。 为此，我们将“位置编码”添加到编码器和解码器堆栈底部的输入嵌入中。 位置编码和嵌入的维度$d$model相同，所以它们俩可以相加。 有多种位置编码可以选择，例如通过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对应于一个正弦曲线。 这些波长形成一个几何级数，从2π到$10000 \cdot 2\pi$。 我们选择这个函数是因为我们假设它允许模型很容易学习对相对位置的关注，因为对任意确定的偏移k，PE_{pos+k}可以表示为PE_{pos}的线性函数。

我们还使用学习到的位置嵌入9进行了试验，发现这两个版本产生几乎相同的结果（参见表 3 行(E)）。 我们选择了正弦曲线，因为它可以允许模型推断比训练期间遇到的更长的序列。

4 为什么选择Self-Attention

本节，我们比较self-attention与循环层和卷积层的各个方面，它们通常用于映射变长的符号序列表示$(x_1,...,x_n)$到另一个等长的序列$(z_1,...,z_n)$，其中$x_i,z_i \in \mathbb{R}^d$，例如一个典型的序列转导编码器或解码器中的隐藏层。 我们使用self-attention是考虑到解决三个问题。

一个是每层计算的总复杂度。 另一个是可以并行的计算量，以所需的最小顺序操作的数量来衡量。

第三个是网络中长距离依赖之间的路径长度。 学习长距离依赖性是许多序列转导任务中的关键挑战。 影响学习这种依赖性能力的一个关键因素是前向和后向信号必须在网络中传播的路径长度。 输入和输出序列中任意位置组合之间的这些路径越短，学习远距离依赖性就越容易[12]。 因此，我们还比较了由不同图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。

表1： 不同图层类型的最大路径长度、每层复杂度和最少顺序操作数。 n为序列的长度，d为表示的维度，k为卷积的核的大小，r为受限self-attention中邻域的大小。

图层类型	每层复杂度	顺序	最大路径长度
		操作	
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
循环	$O(n \cdot d^2)$	$O(n)$	$O(n)$
卷积	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention	$O(r \cdot n \cdot d$	$O(1)$	$O(n/r)$

Attention (受限)	)	O(1)	O(n/r)

如表1所示，self-attention层将所有位置连接到恒定数量的顺序执行的操作，而循环层需要$O(n)$ 顺序操作。 在计算复杂性方面，当序列长度n小于表示维度d时，self-attention层比循环层快，这是机器翻译中最先进的模型最常见情况，例如单词[38]表示法和字节对[31]表示法。 为了提高涉及很长序列的任务的计算性能，可以将self-attention限制在仅考虑大小为r的邻域。 这会将最大路径长度增加到$O(n/r)$。 我们计划在未来的工作中进一步调查这种方法。

核宽度为$k < n$的单层卷积不会连接每一对输入和输出的位置。 要这么做，在邻近核的情况下需要$O(n/k)$ 个卷积层， 在扩展卷积的情况下需要$O(log_k(n))$ 个层[18]，它们增加了网络中任意两个位置之间的最长路径的长度。 卷积层通常比循环层更昂贵，与因子k有关。然而，可分卷积[6]大幅减少复杂度到$O(k \cdot n \cdot d + n \cdot d^2)$。 然而，即使$k = n$，一个可分卷积的复杂度等同于self-attention层和point-wise前向层的组合，即我们的模型采用的方法。

间接的好处是self-attention可以产生更可解释的模型。 我们从我们的模型中研究attention的分布，并在附录中展示和讨论示例。 每个attention head不仅清楚地学习到执行不同的任务，许多似乎展现与句子的句法和语义结构的行为。

5 训练

本节介绍我们的模型训练方法。

5.1 训练数据和批次

我们在标准的WMT 2014英语-德语数据集上进行了训练，其中包含约450万个句子对。 这些句子使用字节对编码[3]进行编码，源语句和目标语句共享大约37000个词符的词汇表。 对于英语-法语翻译，我们使用大得多的WMT 2014英法数据集，它包含3600万个句子，并将词符分成32000个word-piece词汇表[38]。 序列长度相近的句子一起进行批处理。 每个训练批次的句子对包含大约25000个源词符和25000个目标词符。

5.2 硬件和时间

我们在一台具有8个NVIDIA P100 GPU的机器上训练我们的模型。 使用本文描述的超参数的基础模型，每个训练步骤耗时约0.4秒。 我们的基础模型共训练了

10万步或12小时。 For our big models,(described on the bottom line of table 3), step time was 1.0 seconds. 大模型训练了30万步（3.5天）。

5.3 优化器

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

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

这对应于在第一次*warmup_steps*步骤中线性地增加学习速率，并且随后将其与步骤数的平方根成比例地减小。 我们使用*warmup_steps* = 4000。

5.4 正则化

训练期间我们采用三种正则化：

残差丢弃 我们将丢弃[33]应用到每个子层的输出，在将它与子层的输入相加和规范化之前。 此外，在编码器和解码器堆栈中，我们将丢弃应用到嵌入和位置编码的和。 对于基本模型，我们使用$P_{drop} = 0.1$丢弃率。

Label Smoothing 在训练过程中，我们使用的label smoothing的值为$\epsilon_{ls} = 0.1$[36]。 这让模型不易理解，因为模型学得更加不确定，但提高了准确性和BLEU得分。

6 结果

6.1 机器翻译

表2： Transformer在英语-德语和英语-法语newstest2014测试中获得的BLEU分数比以前的最新模型的分数更好，且训练成本只是它们的一小部分。

模型	BLEU			训练成本（FLOPs）					
	cmidrule								
	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**		$2.3 \cdot 10^{19}$					

在WMT 2014英语-德语翻译任务中，大型transformer模型（表2中的Transformer (big)）比以前报道的最佳模型（包括整合模型）高出2.0 个BLEU以上，确立了一个全新的最高BLEU分数为28.4。 该模型的配置列在表3的底部。 训练在8 个P100 GPU上花费3.5 天。 即使我们的基础模型也超过了以前发布的所有模型和整合模型，且训练成本只是这些模型的一小部分。

在WMT 2014英语-法语翻译任务中，我们的大型模型的BLEU得分为41.0，超过了之前发布的所有单一模型，训练成本低于先前最先进模型的1／4 。 英语-法语的Transformer (big) 模型使用丢弃率为$P_{drop} = 0.1$，而不是0.3。

对于基础模型，我们使用的单个模型来自最后5个检查点的平均值，这些检查点每10分钟写一次。 对于大型模型，我们对最后20个检查点进行了平均。 我们使用beam search，beam大小为4 ，长度惩罚$\alpha = 0.6$ [38]。 这些超参数是在开发集上进行实验后选定的。 在推断时，我们设置最大输出长度为输入长度+50，但在可能时尽早终止[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体系结构进行了比较。 我们通过将训练时间、所使用的GPU的数量以及每个GPU的持续单精度浮点能力的估计相乘来估计用于训练模型的浮点运算的数量5。

6.2 模型的变体

表3： Transformer架构的变体。 未列出的值与基本模型的值相同。 所有指标都基于英文到德文翻译开发集newstest2013。 Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	*d*model	*d*ff	*h*	*dk*	*dv*	*Pdrop*	*ϵls*	
base	6	512	2048	8	64	64	0.1	0.1	
(A)				1	512	512			
				4	128	128			
				16	32	32			
				32	16	16			
(B)					16				
					32				
(C)	2								
	4								
	8								
		256			32	32			

		1024			128	128			
			1024						
			4096						
(D)							0.0		
							0.2		
								0.0	
								0.2	
(E)		位置嵌入而不是正弦曲线							
big	6	1024	4096	16			0.3		

为了评估Transformer不同组件的重要性，我们以不同的方式改变我们的基础模型，测量开发集newstest2013上英文-德文翻译的性能变化。 我们使用前一节所述的beam搜索，但没有平均检查点。 我们在表中列出这些结果 3.

在表3的行（A）中，我们改变attention head的数量和attention key和value的维度，保持计算量不变，如3.2.2节所述。 虽然只有一个head attention比最佳设置差0.9 BLEU，但质量也随着head太多而下降。

在表3行（B）中，我们观察到减小key的大小d_k会有损模型质量。 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更有用。 我们在行（C）和（D）中进一步观察到，如预期的那样，更大的模型更好，并且丢弃对避免过

度拟合非常有帮助。 在行（E）中，我们用学习到的位置嵌入[9]来替换我们的正弦位置编码，并观察到与基本模型几乎相同的结果。

6.3 English Constituency Parsing

表4： The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

分析器	**训练**	**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]	半监督	91.3
Huang & Harper（2009) [14]	半监督	91.3
McClosky et al. (2006) [26]	半监督	92.1
Vinyals & Kaiser el al. (2014) [37]	半监督	92.1
Transformer (4 layers)	半监督	92.7
Luong et		

Luong et al. (2015) [23]	多任务	93.0
Dyer et al. (2016) [8]	生成的	93.3

To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. 这项任务提出特别的挑战：输出受到很强的结构性约束，并且比输入要长很多。 此外，RNN序列到序列模型还没有能够在小数据[37]中获得最好的结果。

我们用d_{model} = 1024 在Penn Treebank[25]的Wall Street Journal（WSJ）部分训练了一个4层transformer，约40K个训练句子。 我们还使用更大的高置信度和BerkleyParser语料库，在半监督环境中对其进行了训练，大约17M个句子[37]。 我们使用了一个16K词符的词汇表作为WSJ唯一设置，和一个32K词符的词汇表用于半监督设置。

We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other parameters remained unchanged from the English-to-German base translation model. 在推断过程中，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beam大小21 和α = 0.3 。

表4中我们的结果表明，尽管缺少特定任务的调优，我们的模型表现得非常好，得到的结果比之前报告的除循环神经网络语法[8]之外的所有模型都好。

与RNN序列到序列模型[37]相比，即使仅在WSJ训练40K句子组训练时，Transformer也胜过BerkeleyParser [29]。

7 结论

在这项工作中，我们提出了Transformer，第一个完全基于关注的序列转导模型，用multi-headed self-attention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可以比基于循环或卷积层的体系结构训练更快。 在WMT 2014英语-德语和WMT 2014英语-法语翻译任务中，我们取得了最好的结果。 在前面的任务中，我们最好的模型甚至胜过以前报道过的所有整合模型。

我们对基于attention的模型的未来感到兴奋，并计划将它们应用于其他任务。 我们计划将Transformer扩展到除文本之外的涉及输入和输出模式的问题，并调查局部的、受限attention机制以有效处理大型输入和输出，如图像、音频和视频。 让生成具有更少的顺序性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以在https://github.com/tensorflow/tensor2tensor上找到。

致谢 我们感谢Nal Kalchbrenner和Stephan Gouws富有成效的评论、更正和灵感。

4为了说明点积为什么变大，假设q和k的组成是均值为0和方差为1的独立随机变量。 那么它们的点积$q \cdot k = \sum_{i=1} dk\ qi\ ki$的均值为0，方差为$dk$。

5对于K80、K40、M40和P100，我们分别使用2.8、3.7、6.0和9.5 TFLOPS的值。

参考

[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.

Attention的可视化

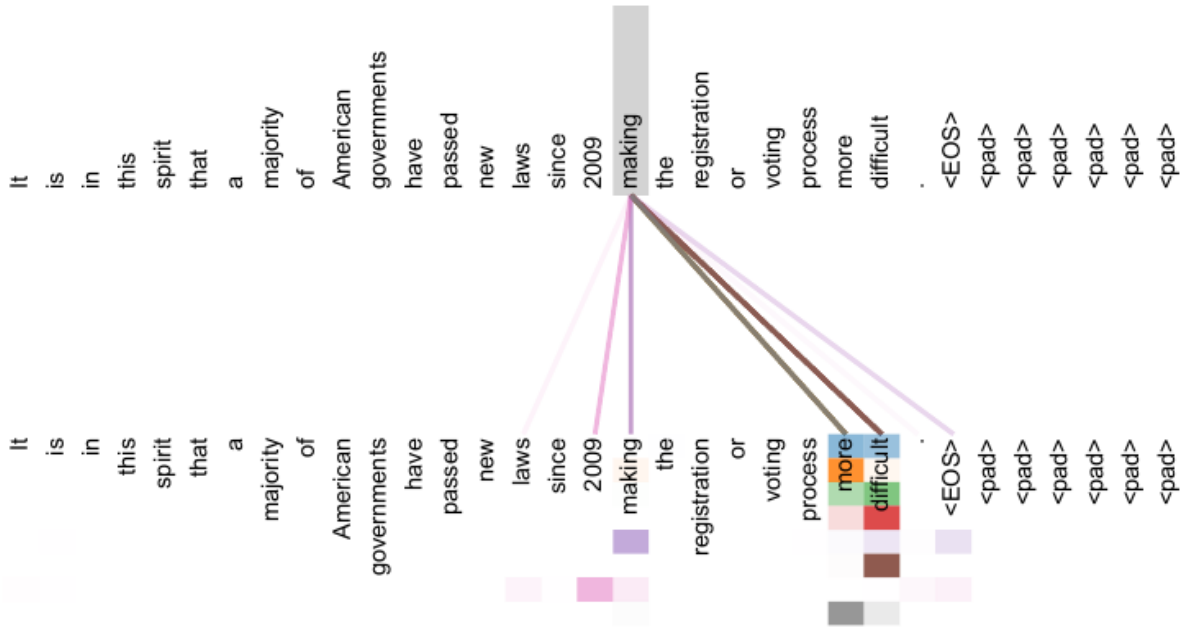


图3： Attention机制的一个示例， 5/6层的编码器self-attention中的长距离依赖的attention。 很多attention head都关注与动词'making'的远距离依赖关系，正好补全'making...more difficult'这个短语。 这里只显示“making”一词的attention。 不同的颜色代表不同的head。 Best viewed in color.

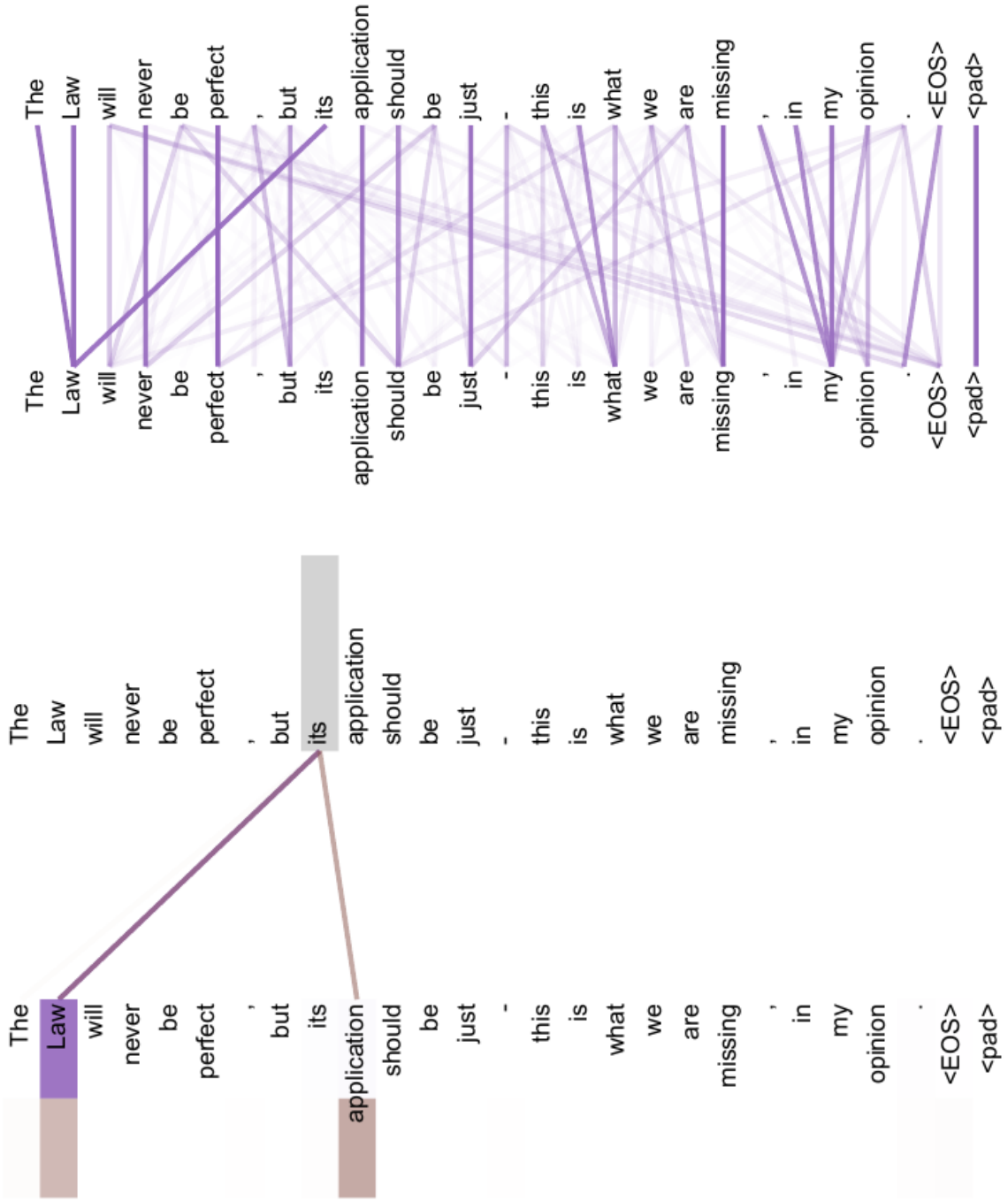


图4： 两个attention head，也在5/6层，显然有逆向照应（下文的词返指或代替上文的词）。 顶部：head 5的完整的attention。 底部：仅将attention

heads 5和6中单词'its'的attention分离出来。 请注意，这个词的attention非常明确。

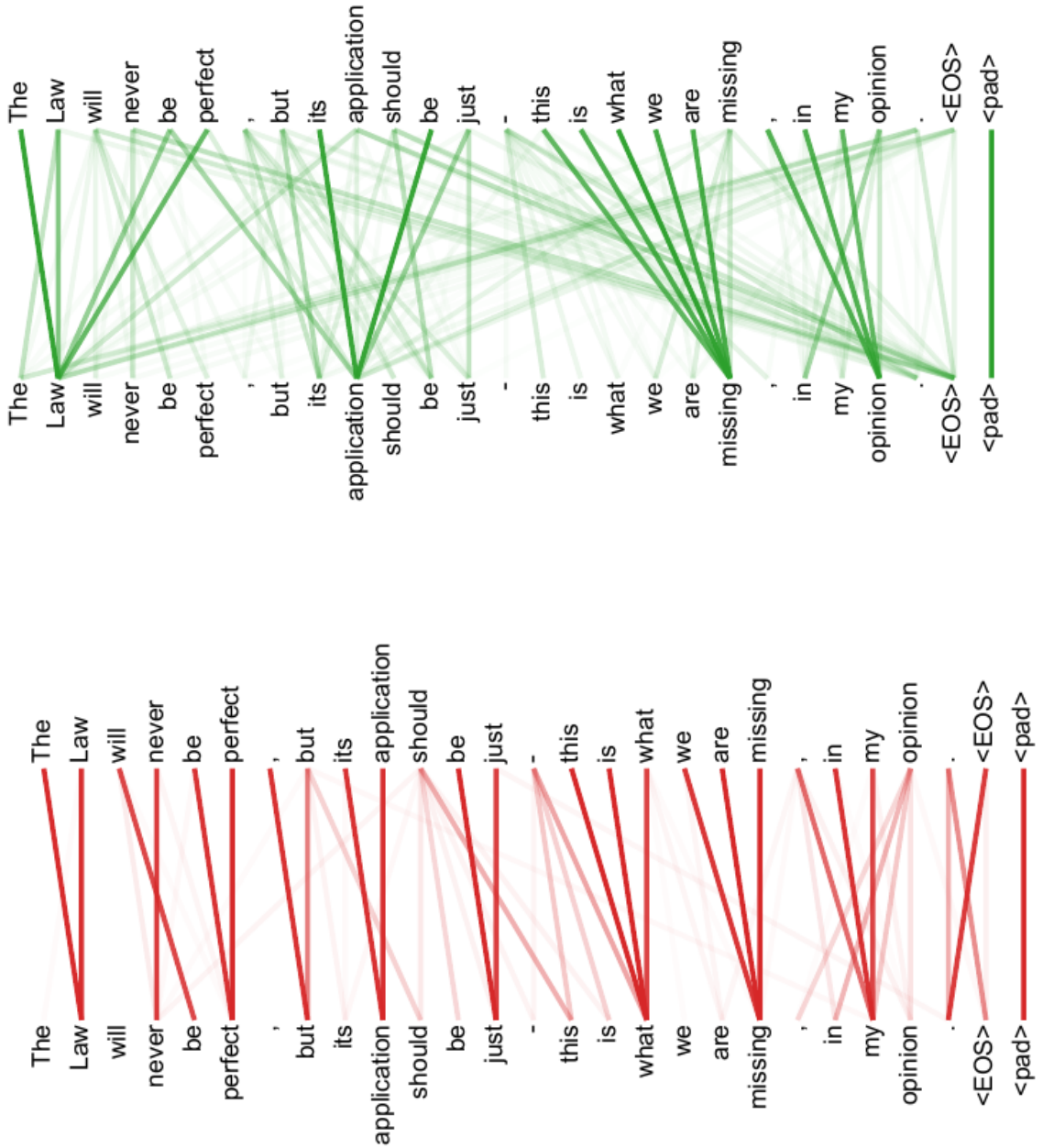


图5： 很多attention head表现出的行为似乎与句子的结构有关。 我们给出了两个这样的例子，来自编码器5/6层self-attention的两个不同的head。Head清楚地学会了执行不同的任务。