一. 算法总览

在第一章提到了Transfomrer在近些年NLP发展中的地位，在本章将深入地研究Tranformer的内部原理。Transformer在各种NLP任务中都表现出色，例如机器翻译、语言生成、问答系统等。更为重要的是，它的算法思想奠定了近些年NLP发展的方向，本文将介绍Transformer的原理，包括它的核心组成部分和基本工作原理。

首先，Transformer的核心组成部分是自注意力机制，它可以使模型对输入序列中不同位置之间的关系进行建模。在Transformer中，每个位置都有一个向量表示，这些向量可以组合成一个矩阵作为模型的输入。模型的任务是将这个输入矩阵转换为输出矩阵，其中每个位置都有一个向量表示。

自注意力机制的核心思想是将每个位置的向量表示与其他位置的向量表示进行比较，并计算它们之间的相似度得分。这个得分可以被看作是一个权重，用于对其他位置的向量表示进行加权求和。通过这种方式，每个位置的向量表示可以被更新为一个考虑到输入序列中其他位置的加权向量和，这样可以更好地捕捉序列中的关系。

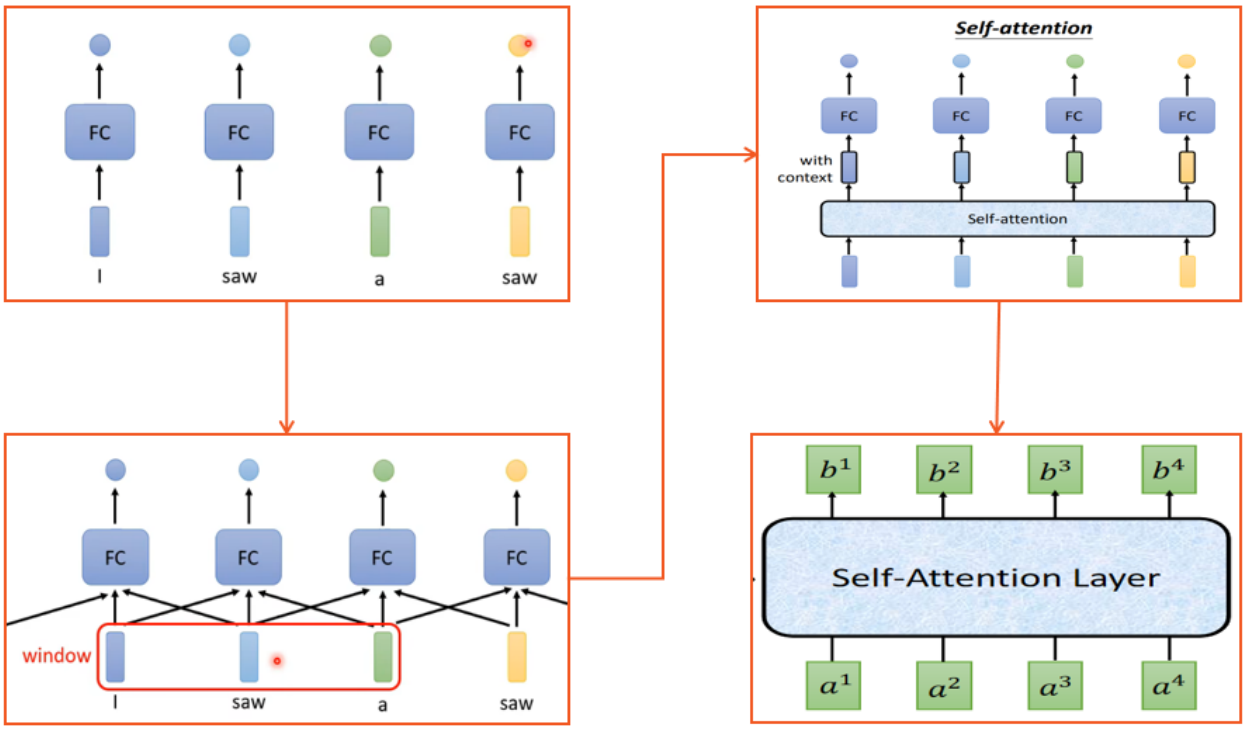
在Transformer中，自注意力机制被应用在两个方面：编码器和解码器。编码器用于将输入序列映射到一个隐层表示，解码器用于将这个隐层表示映射到输出序列。在编码器中，自注意力机制被用于对输入序列进行编码，每个位置的向量表示包含了输入序列中所有位置的信息。在解码器中，自注意力机制被用于对编码器的隐层表示和输出序列中已生成的部分进行解码，从而生成下一个标记。

除了自注意力机制，Transformer中还有两个其他重要的组成部分：多头注意力机制和前向神经网络。多头注意力机制可以在不同的表示子空间中进行自注意力计算，这有助于模型在多个粒度上对输入序列进行建模。前向神经网络是一种全连接的神经网络，用于对每个位置的向量表示进行非线性变换和映射。

总之，Transformer是一种基于神经网络的自然语言处理模型，它使用自注意力机制对输入序列进行建模，并通过多头注意力机制和前向神经网络进行计算。Transformer是目前在各种NLP任务中最先进的模型之一，它的出现彻底改变了自然语言处理领域的面貌。

二. 自注意力

1）self-attention可以兼顾前后的内容，同时计算效率较高，性能较好



2）输入x为one-hot向量，首先经过W矩阵映射到q、k和v，其中v是信息，q和k是用来计算当前的token和其它token的相似度的。

3）拿q和每个k做可伸缩的内积，就是内积后再除以q向量的长度开根号，除以√d主要是防止后面的softmax函数在反向传播时梯度较快达到较小的值。

4）将上一步的可伸缩内积进行softmax函数变换。

5）将所有的信息v加权求和，至此，完成了某一个向量的蕴含上下文的操作变换。

6）对每一个输入做attention操作。

7）如下所示，attention操作是可以并行化。

8）以上是单个展示输入向量的attention过程，实际是三个统一的权重矩阵Wq、Wk、Wv对输入一起做变换，生成Q、K和V矩阵，然后批量做可伸缩内积、softmax、以及加权和。

9）批量做可伸缩内积、softmax、以及加权和

10）多头注意力，在输入向量首先映射到q、k、v之后，再次通过矩阵映射到q1、q2，k1、k2，v1、v2。然后分别用q1、k1和v1进行单头注意力计算，

三. 网络架构

下图展示了Transformer的网络架构，和一般的seq2seq模型一样，它包含编码器和解码器两大部件，左边是序列模型的输入和编码器，右边是序列模型上一次的输出加解码器和最后的输出概率。

编码器部分：以语音识别为例，输入“机器学习”这个语音的四个向量后，首先对输入向量进行embedding变换，然后加上位置embedding，得到一个新的向量b，这个向量作为输入经过多头自注意力后，得到向量a，a向量和b向量相加，并且将向量和进行标准化，标准化后的向量经过前馈神经网络并进一步执行残差和标准化操作，得到编码器的输出。

解码器部分：首先输入一个特殊的开始符，进行embedding和位置编码，并加和，加和后的向量先执行一遍带掩码的多头自注意力层，所谓掩码是指某向量只考虑之前的所有向量进行关联，然后执行一个交叉注意力操作，所谓交叉是指，输入的q来自编码器上一层的输出，而k和v来自编码器的输出，然后执行前馈神经网络层，最后进行线性操作和softmax操作，输出一个概率向量，向量长度为词的长度，其中词库除了汉字和标点符号外，还有一个结束符END。第一个开始字符预测出一个字后，这个字作为解码器的下一个输入，循环往复。

以上是以预测为例说明了Transformer的编码器和解码器执行过程，但在训练时稍有不同，即在解码器的输入端每次是给一个真实的token，用这个token得到下一个token的概率，也就是说在训练时是偷看了真实答案。

以上讲解了Transformer的主要架构，以及里面的细节，更加的详细的内容参见论文attenion is all you need，地址为：https://arxiv.org/abs/1706.03762，以下为论文的中文版本。

Attention is all you need:

摘要：

主流的seq2seq模型是基于包含了编码器和解码器的复杂循环神网络或者卷积神经网络来实现的。表现最好的模型还通过注意力机制将编码器和解码器相连接。我们提出了一种新的简单的网络架构-Transformer，仅基于注意力机制，完全摒弃了循环和卷积。在两个机器翻译任务上的实验表明，这些模型在质量上表现更优秀，同时更易于并行化，且训练时间明显缩短。我们的模型在WMT2014英德翻译任务上取得了28.4BLEU的成绩，相比现有最好的结果（包括集成模型），提高了2个BLEU。在WMT2014英法翻译任务上，我们的模型仅在8个GPU上训练了3.5天，在训练成本远低于文献中最佳模型的情部下，实现了新的单模型BLEU分数最高的记录，达到41.8。我们证明了模型适用于其他任务，并成功将其应用于英语成分分析，包括大量和有限的训练数据。

1 引言

循环神经网络（Recurrent neural networks），长短时记忆网络（Long shot-term memory）和门控循环网络（Gated recurrent neural networks）已经被确认为序列建模和转导问题（例如语言建模和机器翻译）中的最先进方法。自那时以来，已经进行了大量工作来推动循环语言模型和编码器-解码器架构的发展。

循环模型通常根据输入和输出序列中的符号位置分解计算。将位置与计算时间齐步，它们将生成一系列隐藏状ht，作为先前隐藏状态ht-1和位置t的输入函数。这种固有的顺序性质使得在训练时不能并行化，而并行化在句子较长时是非常重要的。最近的工作能过分解技巧和条件计算实现了显著的计算效率提高，同时在后一种情况下提高了模型性能。但是，顺序计算的基本限制仍然存在。

在各种任务中，注意力机制已经成为引人入胜的序列和转导模型中重要的组成部分，允许在输入和输出中不考虑它们这间的距离来建模。极少数情况以下除外，少数情况下，这种注意力机制通常与循环神经网络结合使用。

在这项工作中，我们提出了Transformer模型架构，它避免使用循环，而是完全注意力机制来描述输入和输出的全局依赖关系。Transformer允许更多的并行化，并且在使用八个P100GPU进行为期仅十二个小时的训练后，可以达到翻译质量的新水平。

2 背景介绍：

减少顺序计算的目标也是Extended Neural 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，在所有输入和输出位置上并行计算隐藏表示。在这些模型中，将来自两个任意输入或输出位置的信号相关联所需的操作数量随着位置之间的距离而增加，ConvS2S的增长是线性的，ByteNet的增长是对数的。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，这被减少为一定数量的操作，尽管由于平均注意加权位置而导致有效分辨率降低，我们通过多头注意力在第3.2节中描述的方法来抵消这种效应。

“自注意力”（有时也称为“内部注意力”）是深度学习模型中的一种技术，它能够关注序列中不同位置，以计算序列的表示。这与传统的注意力机制不同，后者通常涉及关注不同的上下文部分，而自注意力则可以关注序列本身的不同部分。自注意力机制已经成功地应用于许多任务，包括阅读 理解、抽象汇总、文本蕴含和学习无关的句子表示。

端到端记已网络是基于递归注意力机制而不是序列对齐递归，并已被证明在简单语言问答和语言建模任务中表现良好。

据我们所知，Transformer是第一个完全依靠自注意力机制计算输入和输出表示而不使用序列对齐RNN或卷积的传导模型。在接下来的章节中，我们将描述Transforer，阐述自注意力的动机，并讨论它相对于[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传导模型都具有编码器-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=（x1, ..., xn）映射到连续表示的z=（z1, .., zn）。给定z，解码器然后一次生成一个符号输出序列（y1, ..., ym）。在每一步，模型都是自回归的，在生成下一个符号时，使用先前生成的符号作为额外的输入。

Transformer遵循这种整体架构，使用叠加的自注意力和逐点全连接层用于编码器和解码器，分别显示在图1的左半部分和右半部分。

图3.1：Transformer的技术架构

3.1 编码器和解码器堆栈

编码器：编码器由N=6个相同的层组成。每个层有两个子层。第一个是多头自注意力机制，第二个是简单的逐位置全连接前馈网络。我们在两个子层周围采用残差连接【11】，然后进行归一化【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，模型中的所有以及嵌入层都产生dmodel=512维的输出。

解码器：解码器也是由N=6相同的层组成。除了每个编码器层中的两个子支之外，解码器还插入了第三个子层，该子层在编码器堆栈的输出上执行多头注意力。与编码器类似，我们在每个子层周围采用残差连接，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，以防止关注后续位置。这种掩码，加上输出嵌入偏移一个位置，确保位置i的预测只依赖于小于i位置的已知输出。

3.2 注意力

一个注意力函数可以被描述为将一个查询和组键-值对映到一个输出，其中查询、键、值和输出都是向量。蜗出被计算为值的加权和，其中分配给每个值的权重是由查询与相 应键的兼容性函数计算得出的。

3.2.1 缩放点注意力

我们称这种特定的注意力为“缩放点注意力”（图2）。输入由维度为dk的查询和键以及维度为dv的值组成。我们计算查询和所有键的点积，将每个点积除以√dk，并应用softmax函数以获得值的权重。

图3.2 （左边）可缩放点积注意力。 （右边）由几个注意力层并行组成的多头注意力

实际上，我们同时在一组查询上计算注意力函数，将它们打包成矩阵Q。键和值也被打包成矩阻K和V。我们计算输出矩阵如下：

两种最常用的注意力函数是加和注意力【2】和点积（乘法）注意力。点积注意力与我们的算法相同，只是缩放因子为1/√dk。加和注意力使用单隐藏层的前馈网络计算兼容函数。虽然两者理论复杂度上相似，但在实践中，点积注意力更快速和更节省空间，因为它可以使用高度优化的矩阵乘法代码来实现。

当dk的值较小的时候，这两种机制的性能相似，但是在较大的dk值下，加法注意力优于没有缩放的点积注意力【3】。我们怀疑在dk值较大的情况下，点积的结果变得非常大，将softmax函数推到其梯度极小的区域。为了抵消这种影响，我们将点积结果缩放1/√dk。

3.2.2 多头注意力

我们发现，与其使用一个dmodel维度的键、值和查询来执行单个注意力函数，通过将查询、键和值分别线性投影到dk、dk和dv维度，会更加有益。对每一组线性投影的查询、键和值，我们进行并行的注意力函数计算，生成dv维度的输出。这些值被连接在一起，并再次进行投影，最终生成最终值，如果3.2所示。

多头注意力机制允许模型在不同的位置同时关注来自不同表示子空间的信息。使用单个注意力头，平均化会抑制这种特性。

这里投影是参数矩阵：

在这项工作中，我们使用h=8个并行的注意力层或头部。对于每个头部，我们使用dk=dv=dmodel/h=64。由于每个头部的降维，总的计算成本类似于完整维度的单头注意力机制。

3.2.3 模型中的注意力应用

Transformer模型使用多头注意力机制有以下三种不同的方式：

- 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，比如【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编码器前一层中的所有位置。
- 类似地，解码器中的自注力层允许解码器中的每个位置关注到该位置以及之前所有位置的信息。我们需要防止解码器中的信息向左流动以保持自回归性质。我们通过在缩放的点积注意力机制内部实现这一点，即通过将对应于非法连接的输入softmax值屏蔽掉（设置为-∞）来实现。请参见图2。

3.3 位置编码前馈神经网络

除了注意力子层之外，我们编码器和解码器中的每一层还包含一个全连接前馈网络，分别对每个位置进行相同的独立转换。它由两个线性变换组成，中间有ReLU激活函数。

尽管线性变换在不同的位置上是相同的，但是它们在不同的层之间使用不同的参数。另一种描述方法是，它们是两个内核大小为1的卷积。输入和输出是dmodel=512，内层的维度是dff=2048。

3.4 Embeddings and Softmax

与其它序列模型类似，我们使用学习到的嵌入（embeddings）将输入和输出单位转换为维度为dmodel的向量。我们还使用通常的线性变换和softmax函数将解码器的输出转换为预测下一个单位的概率。在我们的模型中，我们在两个嵌入层和softmax层之间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√dmodel。

3.5 位置编码

由于我们的模型不含递归和卷积，为了使模型利用序列的顺序，我们必须注入一些有关序列中相对或绝对位置的信息。为此，在编码器和解码器堆栈底部的输入嵌入中添加”位置编码 “。位置编码具有与嵌入相同的dmodel维度，以便两者可以相加。有许多位置编码的选择，可以是可学习的，也可以是固定的【9】。

在这项工作中，我们使用不同频率的正弦和余弦函数来作为位置编码：

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一个正弦曲线。波长形成了一个几何级数，从2pi到10000pi。我们选择这个函数是因为我们假设它能够让模型容易学习到通过相对位置进行注意力计算，因为对于任何固定的偏移量k，PE(pos+k)可以被表示为PE(pos)的线性函数。

我们还尝试使用学习的位置嵌入【9】，并发现两个版本产生了几乎相同的结果（参见表3行（E））。我们选择了正弦版本，因为它可能允许模型外推到比训练中遇到的序列长度更长的序列长

度。

表3.1 表1：不同层类型的最大路径长度，每层的复杂度和最小连续操作数。其中，n为序列长度，d为表示维度，k为卷积核大小，r为限制自注意力中邻域的大小。

4 为什么是自注意力

在本节中，我们从自注意力层的各个部分和RNN、CNN进行比较，它们都是将一排不定长的序列x=（x1, x2, ..., xn）映射到等长的序列z=（z1, z2, ..., zn）， 其中xi，zi∈ Rd，就好像典型的序列转导模型中的编码器和解码器。驱动我们使用自注意力的因素主要是以下三个。

其中一个是每层的总计算复杂度。另一个是可以并行化的计算量，由所需要的最小顺序操作数来衡量。

第三个是网络中长程依赖之间的路径长度。在许多序列转导任务中，学习长程依赖是一个关键挑战。影响学习这种依赖关系能力的一个关键因素是正向和反向信号在网络中需要遍历的路径长度。这些路径在输入和输出序列中任意位置之间的长度越短，学习长程依赖就越容易 [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，仅需执行恒定数量的顺序操作，而递归层则需要 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，自注意力层比递归层更快，而在机器翻译中使用的句子表示中，这种情况最常见，例如 word-piece [38] 和 byte-pair [31] 表示法。为了改善涉及非常长序列的任务的计算性能，自注意力可以限制仅考虑以相应输出位置为中心的输入序列中大小为 r 的邻域。这将增加最大路径长度至 O(n/r)。我们计划在未来的工作中进一步研究这种方法。

如果卷积核宽度 k < n，则单个卷积层不会连接所有的输入和输出位置。在连续核的情况下，这需要 O(n/k) 个卷积层的堆栈，或在扩张卷积的情况下需要 O(logk(n)) 个卷积层 [18]，这将增加网络中任意两个位置之间的最长路径长度。相对于递归层，卷积层通常更加昂贵，因为需要乘以 k 的因子。然而，可分离卷积 [6] 可以大大降低复杂度，降至 O(k·n·d + n·d^2)。然而，即使在 k=n 的情况下，可分离卷积的复杂度也等于自注意力层和点积前馈层的组合，这是我们在模型中采用的方法。

自注意力还可以带来更可解释的模型。我们检查了我们模型的注意力分布，并在附录中展示和讨论了一些例子。不仅单个注意力头明显学习了执行不同的任务，许多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方案。

5.1 训练数据和批处理

我们在标准的WMT2014英德数据集上进行了训练，该数据集包含约450万个句子对。使用字对编码【3】对句子进行编码，其中源-目标词汇表共包含约37000个标记。对于英法语言，我们使用规模更大的WMT2014英法数据集，其中包含3600万个句子，并将标记拆分为32000个词片段词汇表【38】。句子对通过近似序列长度分批处理。每个训练批次包含一组约25000个源标记和25000个目标标记的句子对。

5.2 硬件和训练时间

我们在一台装有8个NVIDIA P100 GPU的机器上训练了我们的模型。对于使用本文所述超参的基础模型，每训练一轮大约需要0.4秒。我们总共训练了基础了10万步或12小时。对于我们的大型模型（表3底部的描述），每轮时间为1.0秒。大型模型总共训练了300000步（3.5天）。

5.3 优化器

我们使用Adam优化器，参数为β1 = 0.9，β2 = 0.98和ϵ = 10−9，我们根据以下公式在训练过程中改变学习率：

这相当于在前warmup_steps个训练步骤中线性增加学习率，之后按步骤数的倒数平方成比例地降低。我们使用warmup_steps=4000。

5.4 正则化

我们在训练时采用三种正则化方法：

Residual Dropout：我们在每个子层的输出加入残差连接之前，对其进行了dropout操作【33】并进行了标准化。此外，我们还对编码器和解码器中的嵌入和位置编码之和进行了dropout操作。在基础模型中，我们使用了Pdrop=0.1的比率。

Label Smoothing：在训练期间，我们采用了标签平滑方法，平滑值为ϵls = 0.1 [36]。这会使我们困惑，因为模型将会更加不确定，但会提高准确性和BLEU分数。

6 结论

6.1 机器翻译

在WMT 2014英德翻译任务中，大型Transformer模型(见表2中的Transformer (big))的性能优于以前报道的最佳模型(包括组合模型)超过2.0 BLEU，创造了28.4的新BLEU得分的最新技术水平。该模型的配置列在表3的最后一行。训练过程在8个P100 GPU上耗时3.5天。即使是我们的基础模型也超过了以前发布的所有模型和组合模型，而训练成本仅为任何竞争模型的一小部分。

WMT 2014英法翻译任务中，我们的大型模型取得了41.0的BLEU分数，在训练成本低于以前最先进模型的1/4的情况下，优于所有先前发布的单一模型。英法翻译的Transformer (big)模型使用了Pdrop = 0.1的丢失率，而不是0.3。

对于基础模型，我们使用了一个由最后5个检查点平均得到的单一模型，这些检查点是以10分钟的间隔写入的。对于大型模型，我们对最后20个检查点进行了平均。我们使用带有束搜索的方法进行推断，束大小为4，长度惩罚参数为α=0.6 [38]。这些超参数是在开发集上进行实验后选择的。在推断期间，我们将最大输出长度设置为输入长度+50，但尽可能早地终止[38]。

表格3.2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结构进行了比较。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的估计值相乘来估计训练模型使用的浮点运算次数。

表3.2 Transformer 在英语到德语和英语到法语 newstest2014 测试中取得了比以前最先进的模型更好的 BLEU 得分，但训练成本仅是之前的一小部分。

6.2 模型变体

为了评估 Transformer 不同组成部分的重要性，我们以不同的方式改变了基础模型，并在开发集 newstest2013 上测量英德翻译的性能变化。我们使用了上一节中描述的束搜索，但没有使用检查点平

均值。我们在表3中呈现了这些结果。

在表3中的(A)行，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，如第3.2.2节所述。虽然单头注意力比最佳设置差0.9 BLEU，但质量也会随着头数过多而下降。

表3.3 Transformer 架构的变体。未列出的值与基本模型相同。所有指标均基于英德翻译开发集 newstest2013。列出的困惑度是每个 wordpiece 的困惑度，根据我们的字节对编码，不应与每个单词的困惑度进行比较。

在表3的(B)行，我们观察到减少注意力键的尺寸 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是有益的。我们在(C)和(D)行中进一步观察到，如预期的那样，更大的模型更好，而丢弃可以帮助避免过度拟合。在(E)行中，我们将正弦位置编码替换为学习的位置嵌入[9]，并观察到与基本模型几乎相同的结果。

6.3 英语句子成分句法分析

为了评估 Transformer 是否能够推广到其他任务，我们在英语句子成分句法分析上进行了实验。这个任务面临着特定的挑战：输出受到强烈的结构约束，而且比输入明显更长。此外，循环神经网络序列到序列模型在小数据情况下无法达到最先进的结果[37]。

我们在Penn Treebank[25]的《华尔街日报》（WSJ）部分（约40K训练句子）上训练了一个4层Transformer，其中dmodel=1024。我们还在半监督的情况下进行了训练，使用了更大的高置信度和BerkleyParser语料库，大约包含1700万个句子[37]。对于仅使用WSJ的情况，我们使用了16K个令牌的词汇表；对于半监督情况，我们使用了32K个令牌的词汇表。

我们只进行了少量实验，以选择在第22节开发集上的dropout，注意力和残差（第5.4节），学习率和beam大小，所有其他参数均与英语到德语基本翻译模型相同。在推理期间，我们将最大输出长度增加到输入长度+300。我们在仅使用WSJ和半监督情况下都使用了beam大小为21和α=0.3。

我们在表4中的结果显示，尽管缺乏任务特定的调优，我们的模型表现出色，除了循环神经网络语法[8]外，其结果比所有先前报告的模型都要好。

与RNN序列到序列模型[37]相比，即使仅在40K句子的WSJ训练集上训练，Transformer的表现也优于Berkley Parser [29]。

表3.4 Transformer模型在英语成分句法分析方面具有很好的泛化能力（结果在WSJ的第23节上）。

7 结论

在这项工作中，我们提出了Transforer，这是第一个完全基于注意力机制的序列传导模型，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的训练速试比基于循环或卷积层的架构快得多。在WMT2014年英德翻译和英法翻译任务中，我们都取得了新的最先进的水平。在前一个任务中，我们的最佳模型甚至胜过以前报告的所有集合模型。

我们对基于注意力机制的未来感到兴奋，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形式的问题，并研究本地的、受限的注意力机制，以有效地处理像图像、音频和视频等大输入输出。让生成变得不那么顺序化是我们的另一个研究目标。

我们用来训练和评估模型的代码可以在 h t t p s : / / g i t h u b . c o m / t e n s o r f l o w / t e n s o r 2 t e n s o r找到。

致谢

我们感谢Nal Kalchbrenner和Stephan Gouws对我们的工作提出的有益的评论、纠正和启示。

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, 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 Ko-
ray 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 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. 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.