# Attention is all you need

**摘要** 在过去序列模型主要由循环神经网络(下面简称rnn)或者是卷积神经网络(下面简称cnn)加上编码器(下面简称encoder)和解码器(下面简称decoder)搭建而成， 某些性能比较好的模型也会在原来基础上将编码器和解码器与注意力(下面简称attention)机制联系起来。在这里，我们提出了一种全新的网络架构---Transformer，它完全摒弃了传统的cnn和rnn，完全是只基于attention机制。在两项机器翻译任务的实验中，Transformer模型比传统模型在两个方面表现出更好的性能，一是它更容易实现并行化，二是在所需的训练时间较短。Transformer在 WMT 2014 Englishto-German机器翻译任务上BLEU得分为28.4，打破了原来的记录，即使原有的最好模型是由多个模型融合而成，BELU得分也比Transformer低了2分。 在WMT 2014 English-to-French 机器翻译任务上，在八块GPU上训练3.5天之后，Transformer建立了一个新的单模型BELU最高得分41.8, 训练代价远远小于原来文献中的最好模型。在英文成分句法任务上，不管是大量训练集还是小量训练集，Transformer都能够表现的很好，这表明Transformer具有良好的泛化能力。

**1. 引入**

近些年，rnn，尤其是长期记忆网络（简称lstm)和门控循环单元（简称gru)，在序列模型比如语言模型，语言转换问题比如机器翻译方面发挥着巨大作用，大量的精力被投入其中，以进一步促进rnn以及encoder-decoder模型的发展。

rnn通常考虑输入输出符号序列的因子计算，在计算的时候，将符号的位置与计算步骤对齐，rnn就会产生一系列的中间状态ht，即



这种固有的顺序模式排除了在训练中进行并行化的可能性，如果训练集是较长的序列，那么这一缺陷长将进一步恶化，因为内存容量将限制批处理的规模（训练每一个数据集集将消耗更多的内存，因此batch size也将减少，从而延长训练时间）。最近有大量关于这方面研究，通过一些分解的技巧以及条件计算，在计算效率上取得一定的突破，使得模型性能得到了提升。但是，序列计算最最根本的限制还是无法避免。Attention成为了各项nlp任务中序列模型以及语言转换模型的重要组成部分，它在无需考虑输入输出序列距离远近的情况下实现了模型中各个序列之间的依赖性。但是在大多数情况下，Attention只是rnn一个组成部分，rnn并没有被抛弃。在这篇论文中，我们提出了Transformer，它避免使用了循环，以attention完全代替它，来实现输入与输出之间的全局依赖性。Transformer更容易实现高度的并行化，在八台P100 GPUs 上面短短训练12小时之后，机器翻译质量就达到了一个新的高度。

个人注解：谷歌提出的Transformer很吸引眼球的地方就是它完全摒弃了传统的rnn结构，仅仅用attention来构建起整个网络。

**2. 背景**

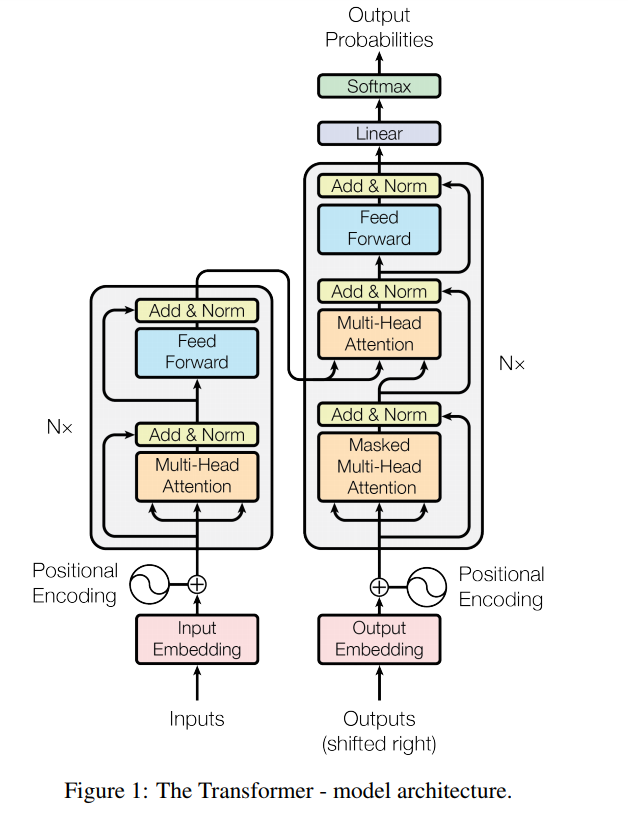
Extented Neural GPU， ByteNet以及ConvS2S出现的背景也是为了减少序列计算的时间，它们都使用cnn来作为最基本的模块，对所有的输入输出位置并行地计算隐层表示权重。在这些模型中，将来自任意输入或输出位置的信号联系起来所需要的操作数量随着位置之间距离的增加而增加，对于ConvS2S来说是线性的，对于ByteNet来说是log的。这使得学习远距离之间的依赖关系变得更加困难。但是在Transformer中，这些操作数量降低至常数级别，尽管代价是降低了有效分辨率(由于对attention位置权重进行了平均)，如3.2节所示，我们用multi-head attention来抵消这种效应。Self-attention是一种特殊的attention，它将单个序列的不同位置关联起来，以这个来表示序列。Self-attention在多个任务中取得了巨大的突破，比如阅读理解，文章摘要，文本蕴含以及学习文本独立的句子表示。端对端的记忆网络基与循环attention机制，而不是顺序对齐的循环（rnn），它在单语言问答，语言模型的构建等任务上表现出色。就我们目前所知道的，Transformer是第一个完全依赖self-attention来计算输入输出表示的模型，它完全没有使用到任何的rnn和cnn。在接下来几个部分中，我们将会介绍Transformer，self-attentiion。

个人注解：近期很多模型都在降低计算成本方面做了许多优化，相比之下，transformer迈出了更大的一步，它将计算不同位置的依赖的代价降低到了常数级别，这是一个很大的突破。

**3. 模型架构**

大多数有竞争力的神经序列转导模型都有一个encoder-decoder结构。其中，encoder将符号表示的输入序列(x1,...,xn)映射称为一个连续的输出序列z=(z1,...,zn)。给定一个z，decoder然年生成将一个序列(y1,...ym)，每次仅输出一个。在每一步这个encoder-decoder是自回归的，在生成下一个符号时，使用前面生成的符号作为附加输入。Transformer也遵循这样的架构，它的encoder和decoder都是由堆叠起来的self-attention和全连接层构成，encoder是Figure1的左边，decoder是Figure2的右边。

个人注解：Transformer是一个以encoder-decoder为架构的模型。



**3.1 Encoder和decoder 的堆叠**

Encoder：Encoder是由六层相同的层堆叠起来。每一层又分为两个子层，第一个子层是一个multi-head attention，第二个子层是一个全连接前馈网络。我们在每个子层上又加了一个残差连接，之后跟了一个正规化层。也就是说，每一个子层的输出是



其中

出于使用残差连接的考虑，模型中所有的子层，包括嵌入层，输出的维数都是512。

个人注解：Figure1中左边的那一部分就是对应一个encoder的layer，一个layer由两小部分构成，下半部分为attention层，上半部分为feed forward层，两层都有residual连接，对应CV中的residule net，作用是降低神经网络训练的难度，当然，加入norm也是为了降低训练难度。

Decoder：Decoder也是由六层相同的层堆叠起来。每一层有三个子层构成，其中两个是跟Encoder的子层一模一样，在这基础上又嵌入了一个新的子层，它对encoder中的输出进行multi-head attention。跟encoder类似，我们在每一个子层上面加了一个残差连接，之后跟了一个正规化层。我们将最底下那个multi-head attention子层进行了略微的修改，防止当前的位置注意到后面的位置。由于嵌入层的输出每次都偏移了一个位置，这个带屏蔽作用的multi-head attention，确保了对位置i的预测只能依赖于小于i的已知输出。

个人注解：Figure1中右边的那一部分就对应一个decoder的layer，大部分都和前面的encoder很接近。有三个地方需要注意：首先它的嵌入层是由position encoding和output encoding的加和，其次它的第二个multi-attention的两个输入是encoder，另外一个输入时第一个multi-attention的输出，最后一个区别就是它第一个multi-attention加了个mask机制，它主要作用是让对position i的预测仅仅依赖于i前面的position，即将注意力限制在前面几个position，mask掉后面几个position。

**3.2 Attention**

Attention函数是这样定义的

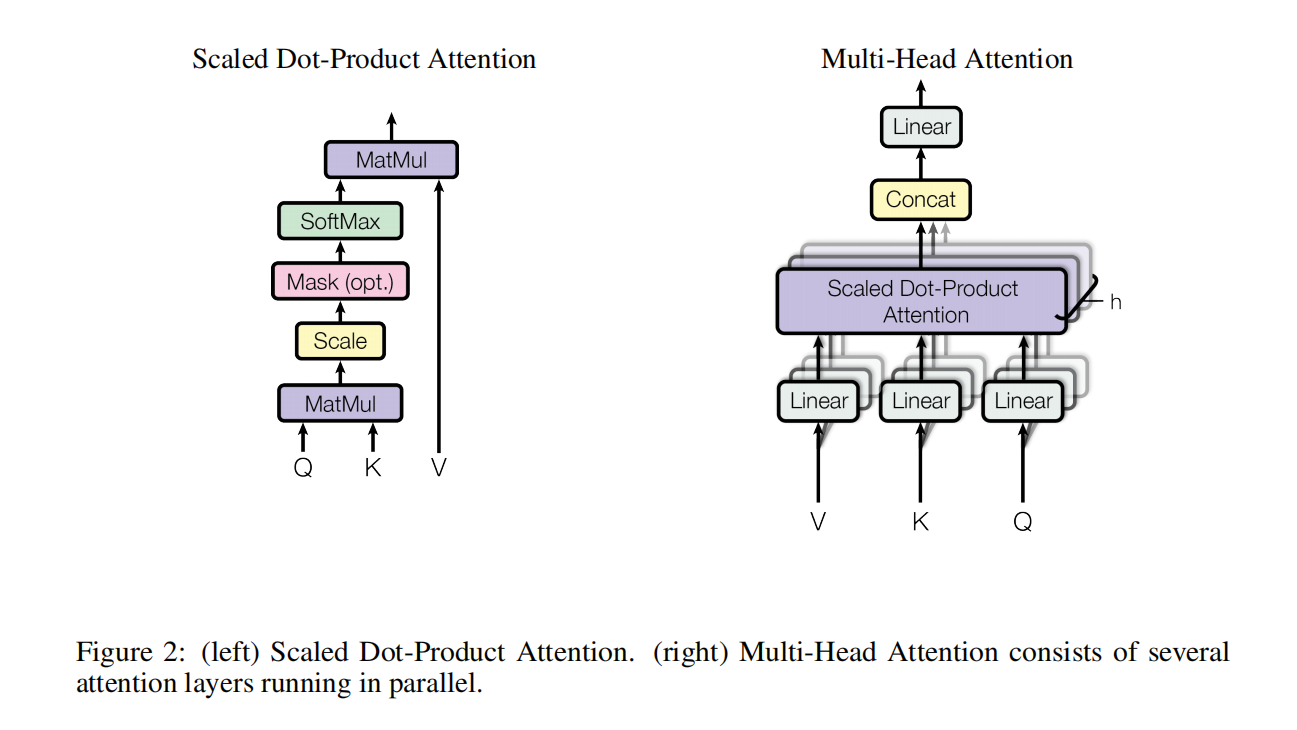


其中query，keys，values以及output都是向量，并且



这里的weight是由下面公式计算





**3.2.1 Scaled Dot-Product Attention**

我们将这个特别的attention称为Scaled Dot-Product Attention。它的输入包括dk维的query，key以及dv维的value。我们通过下列公式计算这个attention的输出



这里Q,K,V都进行了矩阵化，方便并行计算。

两个最最常用的attention函数是additive attention以及dot-product attention。Dot-product attention跟我们这里提出的Sacled Dot-product attention算法类似，只是我们的在计算时乘以倍，additive attention计算时使用了一个带有单个隐藏层的前馈神经网络。尽管这两个attention在理论计算复杂度上几乎相同，dot-product attention在实际应用中计算更快并且空间使用上更少。因为它在实现的时候使用了高度优化的矩阵乘法代码。



当dk值较小时，这两个attention作用几何是一样的，但是如果dk值比较大，additive attention比dot-product attention性能更加出色。我们怀疑，对于比较大的dk值，dot-product attention的输出将会大幅度的增加，之后它被送入softmax函数，由于softmax在输入值较大时，梯度会很小，导致出现梯度消失的情况。为了抵消这种效应，我们在dot-product attention的基础上乘 倍构成scaled dot-product attention。



个人注解：原本attention函数中最最常用的是additive attention和dot-product attention，dot-product attention速度更快，空间成本更小，但是它存在梯度消失的问题，作者将它适当的进行改进，乘以了某个常数，构成了scaled dot-product attention，解决了梯度消失问题。

**3.2.2 Multi-Head Attention**

除了仅仅简单的在 维的key,value,query上进行单层的attention操作，我们发现在它的基础上稍加调整会带来更大的好处，即首先将query，key，value分别线性映射到dk，dk，dv维空间，将这个过程并行运行h次，我们得到h个dk，dk，dv向量。之后我们将它们并行通过h个attention函数，得到h个dv维的输出。这些输出被连接起来，并且又一次被映射，得到最终的结果，整个过程在Figure2的右半部分。multi-head attention使得模型同时关注来自不同表示不同位置的子空间。如果只有一个attention，就达不到这个效果。







Where the projections are parameter matrices ，，，。

在这里，我们使用了h=8个并行attention层。每一个attention层中，我们让dk = dv = dmodel/h = 64。由于对每个attention都进行了降维操作，整体的计算代价跟单个attention（dmodel = 8\*64=512）的计算代价相同。

个人注解：神经网络的核心就是representation，网络的representation越多，这个网络就越强大。对于自然语言，好比原来只能学到一个顺序的representation，现在还能学到一个逆序的representation，甚至乱序的representation，这许许多多不同的representation合起来就是我们人类对于语言的理解。在这里，作者原来的attention在某种意义上只能学到单类的representation，但是进行多次投影变换之后，multi-attention能学到多种representation。

**3.2.3 attention在我们模型中的应用**

Transformer在三个不同的方面使用了multi-head attention

1. 在encoder-decoder attention层，query来自前面的decoder层，key和value来自于encoder的输出。这使得decoder中每一个位置都能注意到输入序列中的所有位置。这模仿了seq2seq模型中的encoder-decoder attention机制。

2. encoder包含了self-attention层。在self-attention层中所有的key，value，query来自于同一个地方，即encoder前一层的输出。Encoder中的每个位置都能注意到前一层中的所有位置。

3. 相同的，decoder中的self-attention层使得decoder中的每个位置能够注意到前一层中的某些位置（这里并不是所有位置，由于有mask的存在，这里只能注意到当前位置的前面几个位置)，我们需要维持decoder中的左侧信息流，这样decoder才有自回归特性。我们在scaled-dot attention中加了一个mask的操作（设置为负无穷），这个操作保证softmax操作之后不会将非法的values连接到attention之中。

**3.3 全连接层**

除了attention子层，我们的encoder和decoder还包括一个全连接层，它分别被相同地作用于各个位置。这个全连接层包括两层线性变换，它们之间夹了一层relu激活层



同一层不同位置的线性表换参数是相同的，只是不同层之间，参数可能有所变化。

另一个描述这个的全连接层的方式是使用两个卷积核，它们的核大小是1。输入输出的维数是512，隐藏层的维度是2048。

个人注解：这里的全连接层没有什么大的创新，但是作者从cnn的视角来看全连接层很新鲜。

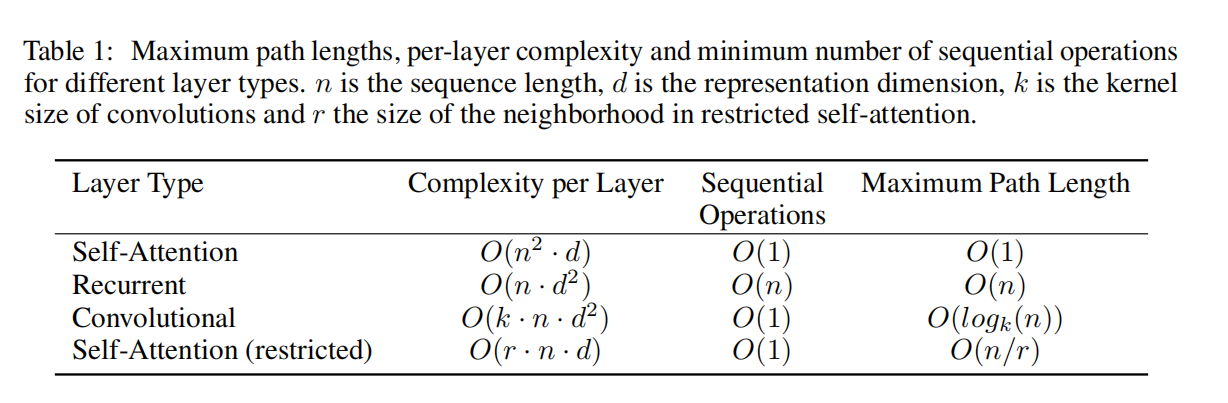
**3.4 嵌入层和softmax**

跟其他序列传导模型类似，我们使用预训练的嵌入层将输入和输出映射成为dmodel维向量。我们还使用线性表换和softmax函数将decoder的输出转变成下一字符的概率向量。在我们的模型中，我们在两个嵌入层和pre-softmax线性变换中共享了权重矩阵。在嵌入层中，我们将这些权重乘以了



**3.5 位置编码**

由于我们的模型没有rnn和cnn，为了使得模型能够利用序列的顺序，我们必须注入一些关于相对位置或者绝对位置的信息



因此，我们在输入的嵌入层中添加了位置编码，它们位于encoder和decoder的底部。 位置编码的维数跟嵌入层相同，这样这两个可以被相加。我们有许多位置编码的方式。

在这里，我们使用了不同频率的正弦和余弦函数:





其中，pos表示位置，i表示维数。也就是说，每一维的位置编码跟一个正弦曲线相关。波长呈现出一个从2π到10000 · 2π几何级数分布。我们选择这个函数是因为我们假设它可以让模型很容易地学习到相对位置，因为对于一个固定地偏差k， 可以表示成一个关于 的线性函数。



我们也实验了使用预先训练的位置编码的效果(see Table 3 row (E))，实验表明使用公式计算出来的位置编码的效果和使用预先训练的位置编码的效果几乎一样。我们选择使用公式计算出来的位置编码因为它能让模型预测的序列长度长于它在训练中遇到的序列长度。

个人注解：位置编码之前也在其他模型中有所出现，比如FB的Conv Seq2Seq，但以前的位置编码都是根据任务训练而得，谷歌直接给出了一个构造位置编码的公式，而且他们提到这个构造出来的位置编码在效果上跟训练出来的一样，并且在某些方面还更胜一筹。

**4 为什么使用Self-Attention**

在这部分中我们比较了self-attention和rnn以及cnn的不同方面，rnn和cnn通常被用在将一个可变长度的序列(x1,...,xn)映射成另一个等长的序列（z1,...zn),其中作为一个传统的序列转换encoder或者decoder的隐藏层。我们使用self-attention的动机主要有三个。



第一就是每一层的总体计算复杂度，第二个就是可以被并行的计算数目，这个指标由所需的最小序列操作数目来评测。

第三个就是网络中远程依赖的路径长度，远程依赖学习在许多序列转换任务中是一个关键难点。一个影响学习远程依赖能力的关键因素是网络中正向和反向信号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，就越容易学习到远程依赖关系。 因此我们也比较了由不同层构成的网络中任意两个输入输出位置之间的最大路径长度。

正如在表一中指出，self-attention层在常数量级的序列操作内连接了所有的位置，与之相比，rnn需要O(n)个序列操作。在计算复杂度方面，当序列长度n比表示维度d小的时候，self-attention层比rnn要快很多。要知道，在最近的机器翻译领域中，我们使用word-piece和byte-pair来表示句子，它们通常的序列长度n比表示维数d小。为了提高在长序列任务上的计算性能，self-attention可以被限制到仅仅考虑在输入序列中以相应输出位置为中心的大小为r的领域。但是这样做会使得最大路径长度增加到O(n/r)。我们打算今后进一步调查这个方法。单个卷积核为k(k<n)的卷积层不能够连接任意输入输出位置的组合。如果想要连接任意输入输出位置的组合，在邻近核的情况下，我们需要一共O(n/k)层卷积层，在扩展卷积的情况下，我们需要一共层卷积层。增加了网络中任意两个位置之间的最长路径。Cnn层的计算代价通常要比rnn大，与因子k有关。然而，可分卷积能够大幅减少复杂度到。然而即使是k=n，一个可分卷积的复杂度等于self-attention和前向全连接层时间复杂度的加和，即我们模型中采用的方法。



同时，self-attention还有一个优点，它更容易解释。从我们的模型中研究attention的分布并且在附录中讨论了一些样例中可以发现，每个单独的attention能够清晰的学习到如何执行不同的任务，它们中的许多似乎展现与句子的语法和语义结构的行为。

个人注解：为什么使用self-attention，作者在三个指标上与rnn跟cnn作出对比，同时最后指出self-attention学到的东西更容易解释。

**5 训练**

本节我们介绍模型的训练方法。

**5.1 训练数据和训练批次**

我们在WMT 2014 English-German dataset上训练，这个数据集包括560万句子对。这些句子对使用了字节编码，源语句和目标语句共享大约37000个词符的词汇表。对于英法翻译，我们采用更大的数据集--2014 English-French dataset，它包括了3600万个句子对，并将 词符分成32000个word-piece词汇表。序列长度相近的句子一起进行批处理训练，每一个训练批次的句子对包括大约25000个源词符和25000个目标词符。

**5.2 硬件和时间**

我们在一台由8个 P100 GPU的机器上训练我们的模型，我们使用这篇论文描述的超参数训练我们的基础模型，每一个训练步骤耗时大约0.4s。我们的基础模型共训练了10万步，共12个小时。对于较大的模型（在table3底部描述的），每一个训练步骤耗时大约1s。总共训练了30万步，共3.5天。

**5.3 优化**

我们使用了adam优化，其中，我们根据一下公式在训练过程中改变学习率：

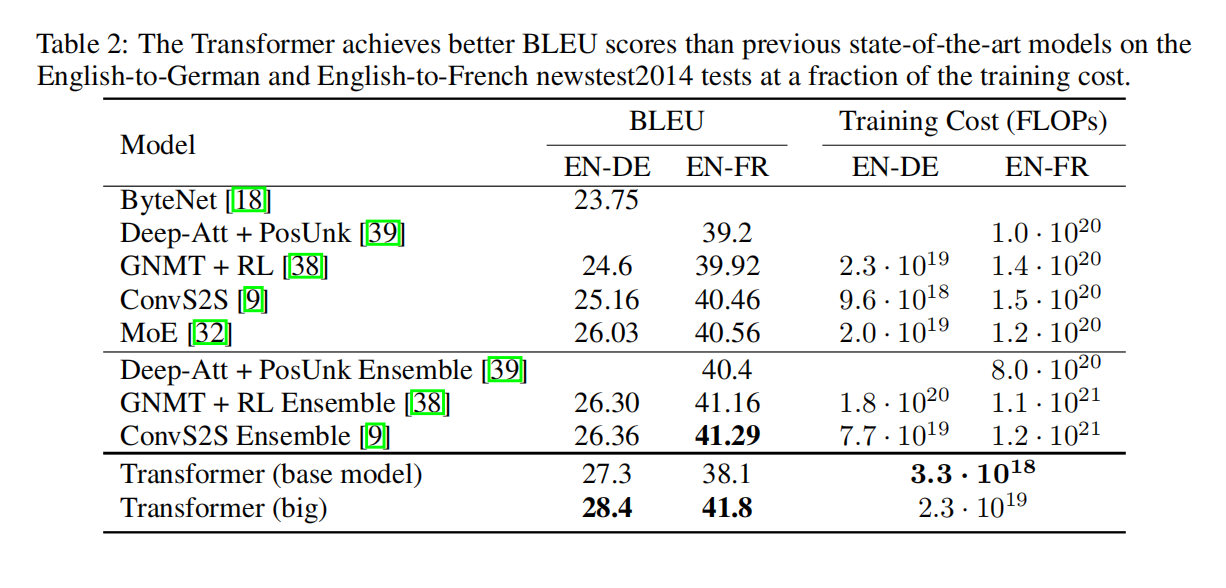


这对应于在warmup\_steps步骤中线性的增加学习速率，并且随后将其与步骤数的平方根成比例地减小。我们使用warmup\_steps = 4000。

**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encoder和decoder的堆栈中，我们将丢弃应用到嵌入和位置编码的和。对于基本模型，我们的丢弃率为0.1。



**标签平滑** 在训练过程中，我们使用的标签平滑率为0.1，这让模型不易理解，因为模型学得更加不确定，但提高了准确率和BLEU得分。

**6 结果**

**6.1 机器翻译**

在WMT 2014 English-to-German translation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。在8个P100 GPU上训练，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 English-to-French translation任务中，我们的大型transformer模型的BLEU得分为41*.*0，超过了之前发布的所有单一模型，训练成本为先前最先进模型的四分之一。这个为英语法语翻译任务训练的Transformer的丢弃率为0.1，而不是0.3。

对于基础模型，它是由最后5个检查点平均而成，这些检查点每10分钟写一次。对于大型模型，它是由最后20个检查点平均而成。我们使用beam search，beam大小为4，长度惩罚*α*=0*.*6。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。

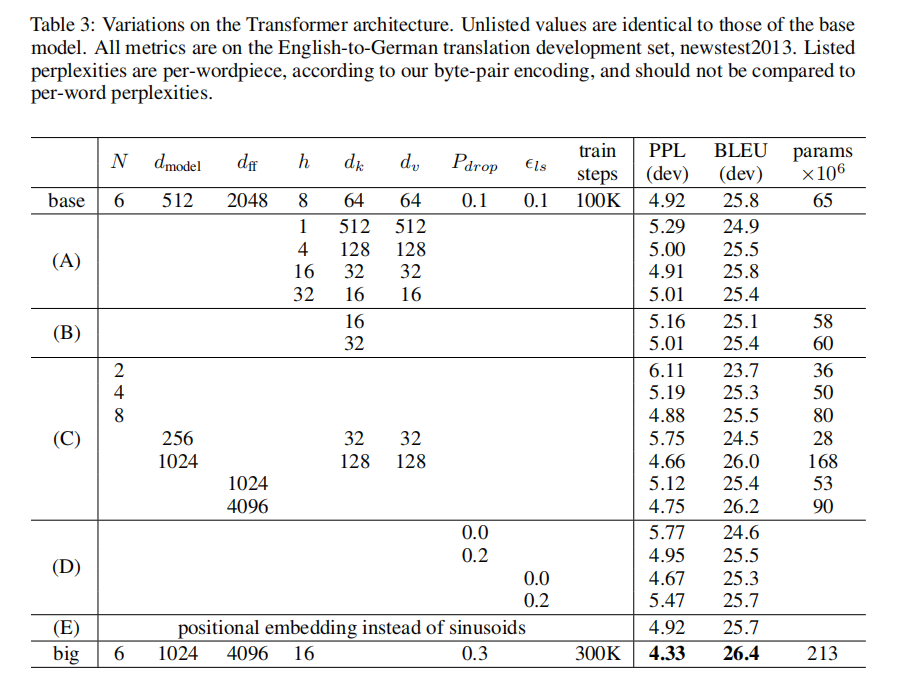
表2总结了我们的结果，并将我们的翻译质量和训练成本与文献中的其他模型体系结构进行了比较。我们通过将训练时间、所使用的GPU的数量以及每个GPU的持续单精度浮点能力的估计相乘来估计用于训练模型的浮点运算的数量。

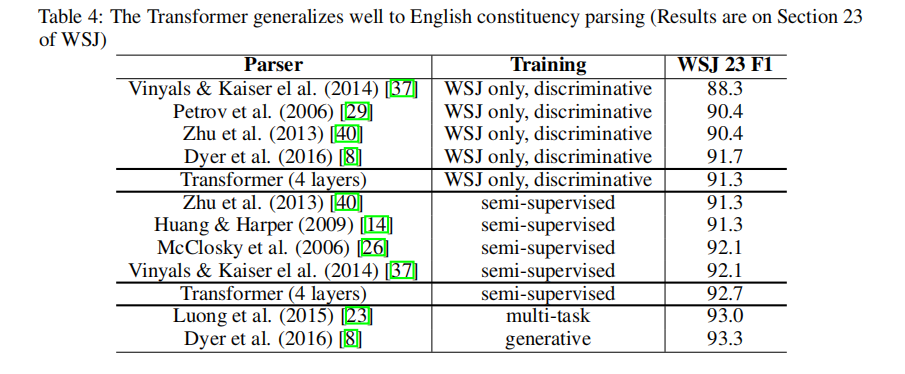
**6.2 模型的变体**

为了评估Transformer不同组件的重要性，我们以不同的方式改变我们的基础模型，在开发集newstest2013上测试英文-德文翻译性能的变化。 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3中列出这些结果。

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，但是head attention数太多性能也不太好。

在表3行（B）中，我们观察到减小key的大小*dk*会降低模型的性能。这表明确定相容性并不容易，并且比点积更复杂的相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，我们用预训练位置编码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 英文成分句法**

为了评估Transformer是否可以泛化到其他任务，我们在英文成分句法任务上做了实验。 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中获得最好的结果。

我们在Penn Treebank的Wall Street Journal（WSJ）部分训练了一个4层*dmodel* 为 1024的 transformer，用了约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，用了大约17M个句子。 对于WSJ, 我们使用了一个16K词符的词汇表，对于半监督学习，我们使用了一个32K词符的词汇表。

在开发集上，我们只做了少量实验来选择dropout，attention，残差, 学习率以及beam大小，所有其他的参数都与英语德语翻译模型相同。在推断过程中，我们将最大输出长度增加到输入长度加上300。 对于WSJ和半监督，我们都使用beam大小21 和*α*= 0*.*3 。

表4中我们的结果表明，尽管没有对特定任务进行调优，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之外的所有模型都好。

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

**7 结论**

在这项工作中，我们提出了Transformer，第一个完全基于attention的序列转换模型，用multi-headed self-attention取代了encoder-decoder架构中最常用的循环层。

对于翻译任务，Transformer可以比基于rnn或cnn的体系的模型训练更快。 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，并研究局部的、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。让生成具有具有更少的顺序性是我们的另一个研究目标。