****

**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，包括编码器和解码器。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，即Transformer，它完全基于注意力机制，完全省去了递归和卷积。在两个机器翻译任务上的实验表明，这些模型在质量上更优越，同时更具并行性，并且需要更少的训练时间。我们的模型在WMT 2014英德文翻译任务中实现了28.4 BLEU，比现有的最佳结果（包括模型融合）提高了2个BLEU以上。在WMT2014英语-德语的翻译任务上,我们的模型在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），创造了单模型的SOTA结果，BLEU分数为41.8。

**1引言**

递归神经网络，特别是长短期记忆[12]和门控递归[7]神经网络，已被牢固地确立为序列建模和转换问题（如语言建模和机器翻译）的最新方法[29，2，5]。此后，许多努力不断突破递归语言模型和编码器-解码器架构的界限[31，21，13]。

递归模型通常沿着输入和输出序列的符号位置来考虑计算。将位置与计算时间中的步长对齐，它们生成一系列隐藏状态ht，作为先前隐藏状态ht−1和位置t的输入的函数。这种固有的顺序性排除了训练示例中的并行化，这在较长的序列长度下变得至关重要，因为内存约束限制了示例之间的批处理。最近的工作通过因子分解技巧[18]和条件计算[26]显著提高了计算效率，同时也提高了后者的模型性能。然而，顺序计算的基本约束仍然存在。

注意机制已成为各种任务中令人信服的序列建模和转换模型的组成部分，允许对依赖性进行建模，而不考虑它们在输入或输出序列中的距离[2，16]。然而，在除少数情况外的所有情况下[22]，这种注意力机制都与递归网络结合使用。

在这项工作中，我们提出了Transformer，这是一种避免重复的模型架构，而是完全依赖于注意力机制来绘制输入和输出之间的全局依赖关系。

Transformer极大地允许了更多的并行化，并且在八个P100 GPU上训练了十二个小时后，可以在翻译质量方面达到最新的水平。

1. **背景**

减少序列计算的目标也构成了扩展神经GPU[20]、ByteNet[15]和ConvS2S[8]的基础，所有这些都使用卷积神经网络作为基础模块，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关联起来所需的操作数，随位置间的距离而增长，ConvS2S为线性增长，ByteNet为对数增长。这使得学习远距离位置之间的依赖关系变得更加困难[11]。在Transformer中，这被减少到恒定数量的操作，虽然代价是由于平均 注意力加权位置信息降低了有效分辨率，如第3.2节所述，我们用多头注意力抵消了这种影响。

自注意力机制，有时称为内部注意，是一种将单个序列的不同位置联系起来以计算序列的表示的注意机制。自注意力机制已成功地用于各种任务，包括阅读理解、抽象摘要、文本隐含和学习任务无关的句子表征[4，22，23，19]。

端到端的记忆网络使用循环注意力机制替代序列对齐的循环，已被证明在简单的语言问答和语言建模任务中表现良好[28]。

然而，据我们所知，Transformer是第一个完全依靠自我注意来计算其输入和输出表示的转换模型，而不使用序列对齐的RNN或卷积的转换模型。在接下来的部分中，我们将描述Transformer,motivate ,self-attention，并讨论其相对于[14，15]和[8]等模型的优势。

**3模型结构**

架构大多数竞争性神经序列转换模型都有编码器-解码器结构[5，2，29]。这里，编码器将符号表示的输入序列映射到连续表示的序列。给定z，解码器以一次生成一个字符的方式生成输出序列。在每一步，模型都是自回归的[9]，在生成下一步时，将先前生成的符号作为附加输入。

Transformer遵循这种整体架构，使用堆叠的self-attention层、point-wise和全连接层，分别用于编码器和解码器，如图1的左半部分和右半部分所示。

**3.1编码器栈和解码器栈**

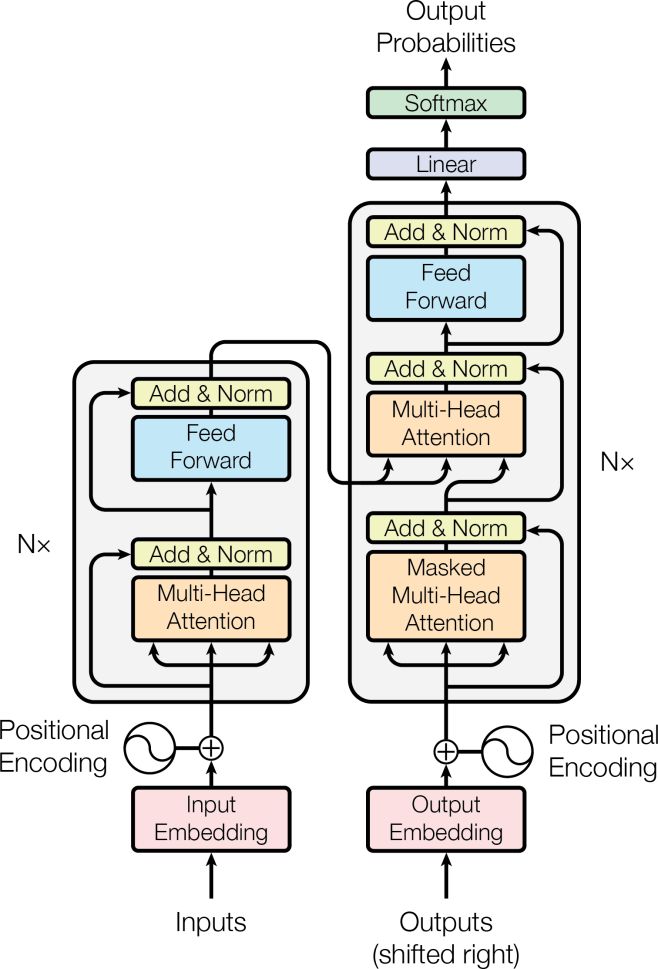
编码器：编码器由个相同层的堆栈组成。每层有两个子层。第一种是多头自我注意机制，第二种是简单的，位置全连接的前馈神经网络。

图1：Tansfomer – 模型架构

我们在两个子层中的每一个子层周围使用残差连接[10]，然后进行层归一化[1]。也就是说，每个子层的输出是，其中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都会产生维度为的输出。

解码器：解码器也是由N=6个相同层的堆栈组成的。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，该第三子层对编码器堆栈的输出执行多头注意力机制。与编码器类似，我们在每个子层周围使用残差连接，然后进行层归一化。我们还修改了解码器堆栈中的自注意子层，以防止当前位置信息中被添加进后续的位置信息。这种掩码与偏移一个位置的输出embedding相结合， 确保对第i个位置的预测 只能依赖小于i的已知输出。

**3.2注意力机制**

注意力机制可以描述为将一个query和一组key-value对映射到一个输出，其中query，keys，values和输出均是向量。输出是values的加权求和，其中每个value的权重 通过query与相应key的兼容函数来计算。

**3.2.1****缩放的点积注意力机制**

我们将我们的特别注意力称为“缩放的点积注意力机制”（图2）。输入由维度dk的query和key以及维度的values组成。我们计算query和所有key的

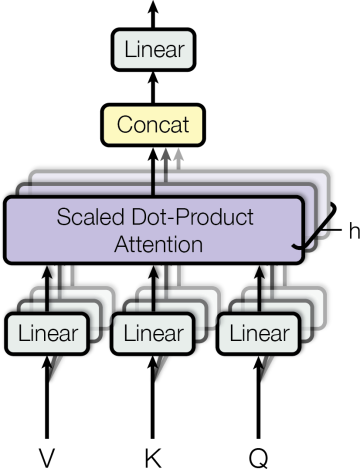
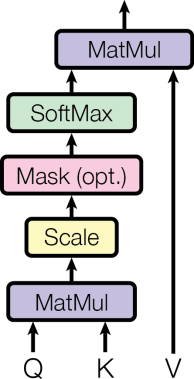
缩放的点积注意力机制 多头注意力机制

图2：(左)缩放点积注意力，(右)多头注意力由多个并行运行的注意力层组成。

点积，再将每个键除以，并应用softmax函数来获得values的权重。

在实践中，我们同时计算一组查询的注意力函数，将其打包成矩阵Q。键和值也打包成矩阵K和V。我们将输出矩阵计算为:

(1)

两个最常用的注意力函数是加性注意力[2]和点积注意力。点积注意力与我们的算法相同，只是比例因子为。加性注意力使用具有单个隐藏层的前馈网络来计算兼容性函数。虽然两者在理论复杂性上相似，但点积注意力在实践中要快得多，空间效率也更高，因为它可以使用高度优化的矩阵乘法代码来实现。

虽然对于较小的值，这两种机制的表现相似，但在不缩放较大的[3]值的情况下，加性注意力优于点积注意力。我们怀疑，对于较大的dk值，点积的大小会变大，从而将softmax函数推向具有极小梯度4的区域。为了抵消这种影响，我们将点积按进行缩放。

**3.2.2多头注意力机制**

我们发现，将查询、键和值分别线性投影到、和维度，而不是用维度的键、值和查询执行单个注意力功能，这是有益的。然后，在这些查询、键和值的投影版本中的每一个上，我们并行执行注意力函数，生成dv维输出值。这些被连接起来并再次投影，得到最终的值，如图2所示。

多头注意力允许模型把不同位置子序列的表示都整合到一个信息中。如果只有一个注意力头部，它的平均值会削弱这个信息。

（横线以下的东西）4为了说明为什么点积变大，假设q和k的分量是独立的随机变量，平均值为0，方差为1。那么他们的点积，，具有均值0和方差dk。



其中投影是参数矩阵

在这项工作中，我们使用了h=8个平行的注意力层或头部。对于其中的每一个，我们使用。由于每个头部的维度减少，总计算成本与全维度的单头部注意力相似。

**3.2.3注意力机制在我们的模型中的应用**

Transformer以三种不同的方式使用多头注意力：

•在“编码器-解码器注意力”层中，query来自前一个解码器层，键和值来自编码器的输出。这允许解码器中的每个位置关注输入序列中的所有位置。这模仿了序列到序列模型中的典型编码器-解码器注意力机制，例如[31，2，8]。

•编码器包含自我注意力层。在自我注意力层中，所有的键、值和query都来自同一个地方，在这种情况下，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的前一层中的所有位置。

•类似地，解码器中的自我注意力层允许解码器中的每个位置关注解码器中的所有位置，直到包括该位置。为了保持解码器的自回归特性，需要防止解码器中的信息向左流动。我们在缩放的点积注意力机制的内部 ，通过屏蔽softmax输入中所有的非法连接值（设置为 ）实现了这一点。见图2。

**3.3基于位置的前馈神经网络**

除了注意子层之外，我们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，该网络单独且相同地应用于每个位置。这包括两个线性变换，其间有一个ReLU激活。



虽然线性变换在不同的位置上是相同的，但它们在不同的层之间使用不同的参数。另一种描述方式是将其描述为核大小为1的两个卷积。

输入和输出的维度为，内层的维度为。

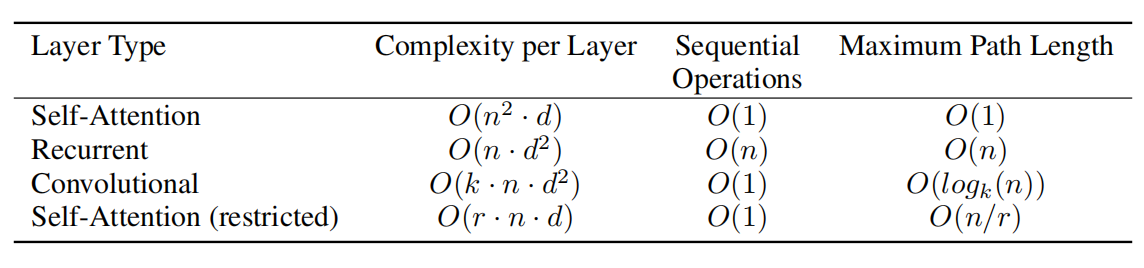
**3.4词嵌入和 Softmax**

与其他序列转换模型类似，我们使用学习到的嵌入词向量 将输入字符和输出字符转换为维度为​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[24]。 在嵌入层中，我们将这些权重乘以。

**3.5位置编码**

由于我们的模型不包含递归和卷积，为了让模型利用序列的顺序，我们必须加入序列中关于字符相对或者绝对位置的一些信息。为此，我们在encoder和decoder堆栈底部的输入嵌入中添加“位置编码”。 位置编码和嵌入的维度​相同，所以它们两个可以相加。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[8]。

表1：不同层类型的最大路径长度、每层复杂性和最小顺序操作次数。n是序列长度，d是表示维数，k是卷积的核大小，r是限制自注意中邻域的大小。



在这项工作中，我们使用了不同频率的正弦和余弦函数：



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。波长形成了从2π到10000·2π的几何数列。我们之所以选择这个函数，是因为我们假设它可以让模型很容易地通过相对位置来学习,因为对任意确定的偏移k,可以表示为的线性函数。

我们还尝试使用预先学习的positional embeddings[9]来代替正弦波，发现这两个版本产生了几乎相同的结果 (see Table 3 row (E))。我们之所以选择正弦曲线，是因为它允许模型扩展到比训练中遇到的序列长度更长的序列。

**4为什么选择自注意**

在本节中，我们将自我注意层的各个方面与递归层和卷积层进行了比较，比较的方式是将一个可变长度的符号表示序列映射到另一个等长序列，*xi , zi ∈* R *d* ,例如典型序列转换编码器或解码器中的隐藏层。我们考虑了三个方面，最后促使我们使用自注意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，通过所需的最小顺序操作数来衡量。

第三个是网络中长程依赖项之间的路径长度。学习长程依赖性是许多序列转换任务中的一个关键挑战。影响学习这种依赖性的能力的一个关键因素是前向和后向信号在网络中必须经过的路径的长度。输入和输出序列中任何位置组合之间的这些路径越短，就越容易学习长程依赖性[11]。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，自注意层用恒定数量的顺序执行的操作连接所有位置，而递归层需要个顺序操作。就计算复杂度而言，当序列长度n小于表示维度d时，自注意层比递归层更快，最常见的情况是机器翻译中最先进的模型使用的句子表示，如分词[31]和字节对[25]表示。

为了提高涉及很长序列的任务的计算性能，可以将自注意限制为仅考虑输入序列中r的尺寸以相应的输出位置为中心。这将使最大路径长度增加到。我们计划在未来的工作中进一步研究这种方法。

核宽度的单个卷积层并不连接所有输入和输出位置对。在连续核的情况下，这样做需要卷积层的堆栈，或者在扩张卷积的情况下需要[15]，从而增加网络中任何两个位置之间最长路径的长度。卷积层通常比递归层贵k倍。然而，可分离卷积[6]将复杂性显著降低到。然而，即使，可分离卷积的复杂性也等于自注意层和逐点前馈层的组合，这是我们在模型中采用的方法。

作为副作用，自关注力可以产生更多可解释的模型。我们从我们的模型中检查注意力分布，并在附录中给出和讨论示例。每个attention head不仅清楚地学习到执行不同的任务，还表现出了许多和句子的句法和语义结构相关的行为。

**5.训练**

本节介绍了我们模型的训练方法。

**5.1训练数据和批处理**

我们在标准WMT 2014英语-德语数据集上进行训练，该数据集由约450万个句子对组成。句子使用字节对编码[3]进行编码，该编码具有约37000个标记的共享源目标词汇表。对于英语-法语，我们使用了更大的WMT 2014英语-法语数据集，该数据集由3600万个句子组成，并将标记拆分为32000个单词的词汇[31]。句子对按近似的序列长度分批排列在一起。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**

我们在一台带有8个NVIDIA P100 GPU的机器上训练了我们的模型。对于我们使用论文中描述的超参数的基本模型，每个训练步骤大约需要0.4秒。我们对基础模型进行了总共100000步或12小时的训练。对于我们的大型模型（如表3的底线所述），步长为1.0秒。大型模型接受了300000步（3.5天）的训练。

**5.3优化器**

我们使用了Adam优化器[17]，其中, 和。我们在训练过程中改变了学习率，根据公式：



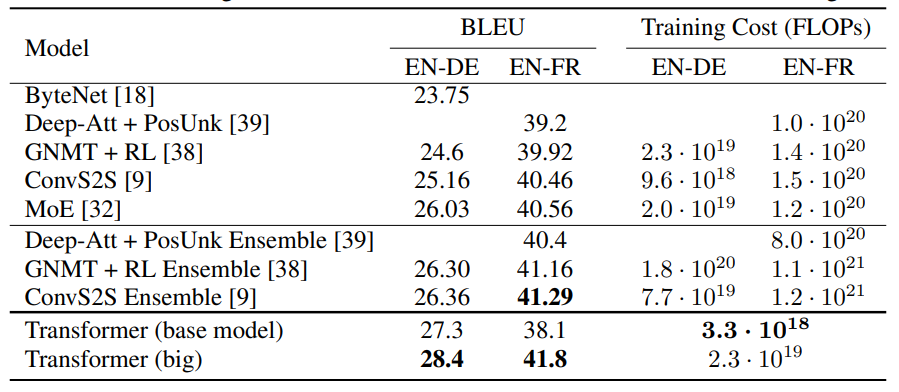
这对应于在第一次 步骤中线性地增加学习速率，并且随后将其与步骤数的平方根成比例地减小。我们使用了=4000。

**5.4正则化**

我们在训练期间使用三种类型的正则化：

残差丢弃 我们将丢弃[27]应用于每个子层的输出，然后将其添加到子层输入并进行归一化。此外，我们将丢弃应用于编码器和解码器堆栈中的嵌入和位置编码的和。对于基本模型，我们使用的速率。

表2:Transformer在2014年英语到德语和英语到法语新闻测试中获得了比以前最先进的模型更好的BLEU分数，只需少量的培训成本。



标签平滑在训练过程中，我们使用了值为的标签平滑[30]。这伤害了ppl，因为模型学会了更加不确定，但提高了准确性和BLEU分数。

**6.结果**

**6.1机器翻译**

在WMT 2014英语到德语翻译任务中，大Transformer模型（表2中的变压器（大））比以前报道的最好的模型（包括组合）高出2.0以上的BLEU，建立了28.4的新的最先进的BLEU分数。该型号的配置列于表3的最后一行。8台P100 GPU的培训耗时3.5天。即使是我们的基本模型也超过了之前发布的所有模型和集合，其训练成本仅为其他竞争模型的一小部分。

在WMT 2014英法翻译任务中，我们的大模型的BLEU得分为41.0，优于之前发布的所有单个模型，训练成本不到之前最先进模型的1/4。为英语到法语训练的Transformer（大）模型使用的丢弃率，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checkpoints的平均值，这些checkpoints每10分钟保存一次。 对于大型模型，我们对最后20个checkpoints进行了平均。 我们使用beam search，beam大小为4 ，长度惩罚α = 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，但在条件允许时会尽早终止[31]。

表2总结了我们的结果，并将我们的翻译质量和培训成本与文献中的其他模型架构进行了比较。我们通过乘以训练时间、使用的GPU数量和每个GPU 5的持续单精度浮点容量的估计来估计用于训练模型的浮点运算的数量。

**6.2模型变体**

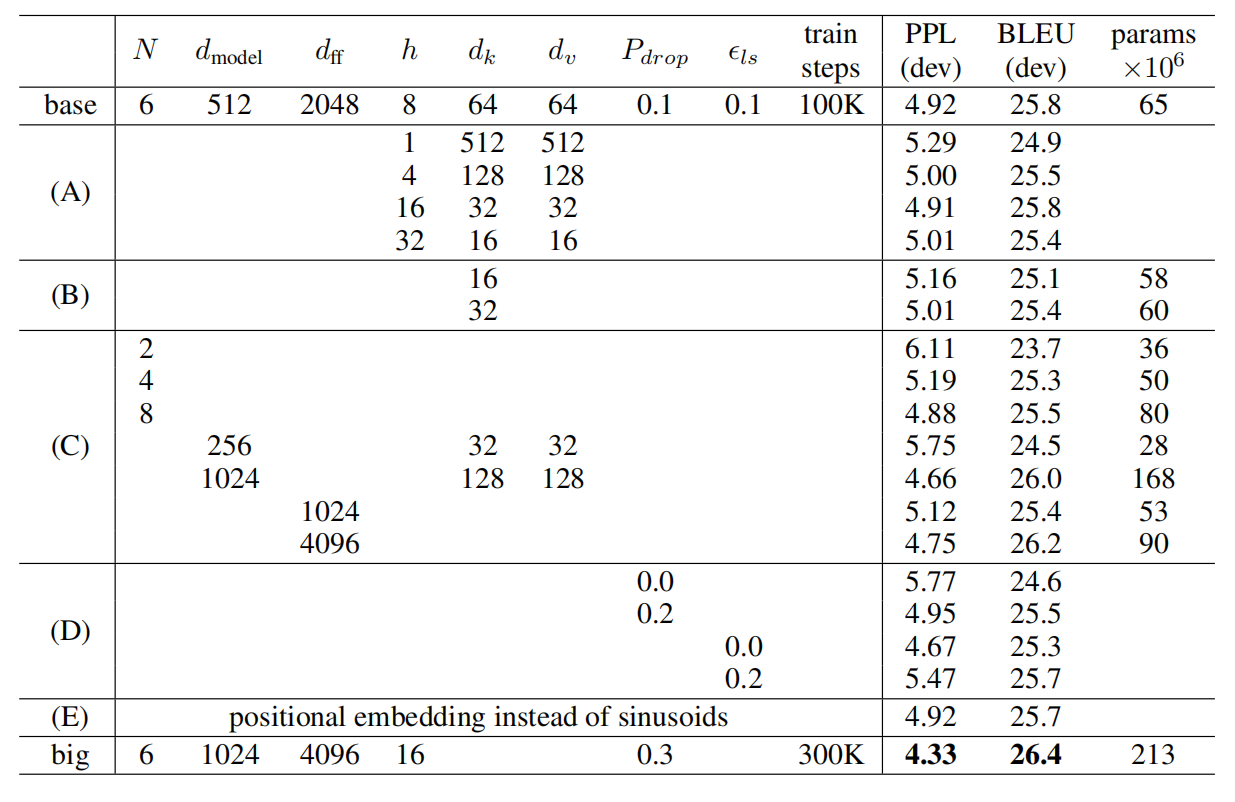
为了评估变压器不同组件的重要性，我们以不同的方式改变了我们的基本模型，测量了开发集英语到德语翻译的性能变化，newstest2013。我们使用了波束搜索，如前一节所述，但没有检查点平均。我们在表3中给出了这些结果。

在表3的行（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，保持计算量不变，如第3.2.2节所述。虽然单头关注度比最佳设置差0.9 BLEU，但如果头太多，质量也会下降。

（横线以下的内容）

5我们对K80、K40、M40和P100分别使用了2.8、3.7、6.0和9.5 TFLOPS的值。

表3：Transformer架构的变化。未列出的值与基本模型的值相同。所有指标都在英语到德语翻译开发集合中，newstest2013。根据我们的字节对编码，列出的ppl是每个单词的，不应该与每个单词的ppl进行比较。



在表3的行（B）中，我们观察到减少注意力键大小dk会损害模型质量。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。我们在第（C）行和第（D）行中进一步观察到，正如预期的那样，更大的模型更好，而丢弃非常有助于避免过度拟合。在第（E）行中，我们将正弦位置编码替换为学习的位置嵌入[8]，并观察到与基本模型几乎相同的结果。

**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，用多头自注意取代了编码器-解码器架构中最常用的递归层。

对于翻译任务，Transformer可以比基于递归或卷积层的架构更快地进行训练。在WMT 2014英语到德语和WMT 2014英文到法语的翻译任务中，我们都达到了最好的水平。在前一项任务中，最佳模型甚至优于之前报道的所有整合模型。

我们对基于注意力的模型的未来感到兴奋，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模态的问题，并研究局部、限制注意力机制，以有效处理图像、音频和视频等大型输入和输出。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在https://github.com/ tensorflow/tensor2tensor 。

**鸣谢**

我们感谢Nal Kalchbrenner和Stephan Gows富有成效的评论、更正和启发。

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] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122v2, 2017.

[9] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[10] 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.

[11] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[13] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410, 2016.

[14] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In International Conference on Learning Representations (ICLR), 2016.

[15] 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.

[16] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In International Conference on Learning Representations, 2017.

[17] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[18] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. arXiv preprint arXiv:1703.10722, 2017.

[19] 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.

[20] Samy Bengio Łukasz Kaiser. Can active memory replace attention? In Advances in Neural Information Processing Systems, (NIPS), 2016.

[21] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attentionbased neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

[22] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In Empirical Methods in Natural Language Processing, 2016.

[23] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304, 2017.

[24] Ofir Press and Lior Wolf. Using the output embedding to improve language models. arXiv preprint arXiv:1608.05859, 2016.

[25] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909, 2015.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. CoRR, abs/1512.00567, 2015.

[31] 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.

[32] 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.