Attention Is All You Need 翻译

Content

Abstract：

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

摘要：

当前主流的序列转换模型都是基于复杂的循环神经网络（recurrent neural networks）或者卷积神经网络（convolutional neural networks），这些网络通常包括一个编码器和一个解码器。表现最好的模型还会通过注意力机制来连接编码器和解码器。我们提出了一个新的简单的网络架构——Transformer，仅基于自注意力机制，完全放弃了循环和卷积。两项机器翻译任务实验表明，这种新型模型在质量上优于现有最佳结果，同时具备更高的并行化能力，并且需要花费显著减少的时间来训练。在 WMT 2014 的英德翻译任务上，我们提出的 Transformer 模型实现了 28.4 的 BLEU 分数，这是迄今为止所有单一模型和集成模型在此项任务中最好的成绩，超越现有最佳结果整整 2 个 BLEU 分。在 WMT 2014 的英法翻译任务上，我们提出的 Transformer 模型创造了一个全新的单一模型最高 BLEU 分数：41.8，而这一切，只需要花费 3.5 天时间在八块 GPU 上训练，远低于现有最佳结果的成本。我们证明了 Transformer 可以很好地推广到其他 NLP 任务中。在依存句法解析任务（constituency parsing）上，

无论是使用大量还是有限的训练数据，我们提出的 Transformer 模型都取得了不错的成绩。

1 Introduction

Recurrent neural networks, long short-term memory and gated recurrent neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures.

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks and conditional computation, while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences. In all but a few cases, however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

循环神经网络，特别是长短期记忆（Long Short-Term Memory）和门控循环神经网络（Gated Recurrent Neural Networks），已经被公认为是在序列建模和转换问题中，如语言模型和机器翻译等领域中的最先进方法。之后，许多工作继续推动了循环语言模型和编码解码架构的界限。

循环模型通常沿着输入和输出序列的符号位置进行计算因子分解。通过将这些位置与计算时间上的步骤对齐，它们生成一个由前一步的隐藏状态 h_{t-1} 和位置 t 处的输入共同决定的隐藏状态 h_t 序列。这种本质上顺序的性质使得在训练样例内进行并行化成为不可能，尤其是在序列长度较长时，这种情况下，记忆限制会限制批处理跨越多个样例。最近的工作通过因子分解技巧和条件计算实现了显著的计算效率改进，同时在后一种情况下还改进了模型性能。但是，顺序计算的基本约束仍然存在。

注意力机制已经成为各种任务中令人信服的序列建模和转换模型的一个不可缺少部分，使得可以对输入或输出序列中的任意依赖项进行建模，无论其距离如何。然而，在几乎所有情况下，这些注意力机制都是与循环神经网络一起使用的。

在这篇文章中，我们提出了 Transformer 模型架构，该架构放弃了循环结构，取而代之的是完全依靠注意力机制来绘制输入和输出之间的全局依赖关系。Transformer 允许进行更大程度的并行化，在 8 个 P100 GPU 上训练 12 小时后，可以达到新的翻译质量最先进水平。

2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU, ByteNet and ConvS2S, all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations.

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks.

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17,18]and [9].

降低序列计算的目标也是扩展神经网络 GPU（Extended Neural GPU）、ByteNet 和 ConvS2S 的基础，它们都使用卷积神经网络作为基本构建块，为所有输入和输出位置并行计算隐藏表示。在这些模型中，关联两个任意输入或输出位置上的信号所需的操作数量会随着位置之间距离的增加而线性增长（ConvS2S）或对数增长（ByteNet）。这使得学习远处位置之间的依赖关系更加困难。在 Transformer 中，这被降低到一个常数的操作数量，但代价是由于平均加权注意力位置而减少的有效分辨率，我们通过多头注意力机制来对抗这种效果，如 3.2 部分所述。

自注意力（Self-Attention），也可被称为内部注意力，是一种关联单个序列不同位置的注意力机制，以计算该序列的表示。自注意力已成功应用于各种任务，包括阅读理解、抽象总结、文本蕴涵和学习与任务无关的句子表征。

端到端记忆网络（end-to-end memory networks）基于一个递归注意力机制，而不是序列对齐的递归结构，并且在简单语言的问答和语言建模任务中表现良好。

据我们所知，Transformer 是第一个完全依赖自注意力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。在接下来的章节中，我们将描述 Transformer，解释自注意力的动机，并讨论它相对于诸如[17, 18]和[9]等模型的优势。

3 Model Architecture

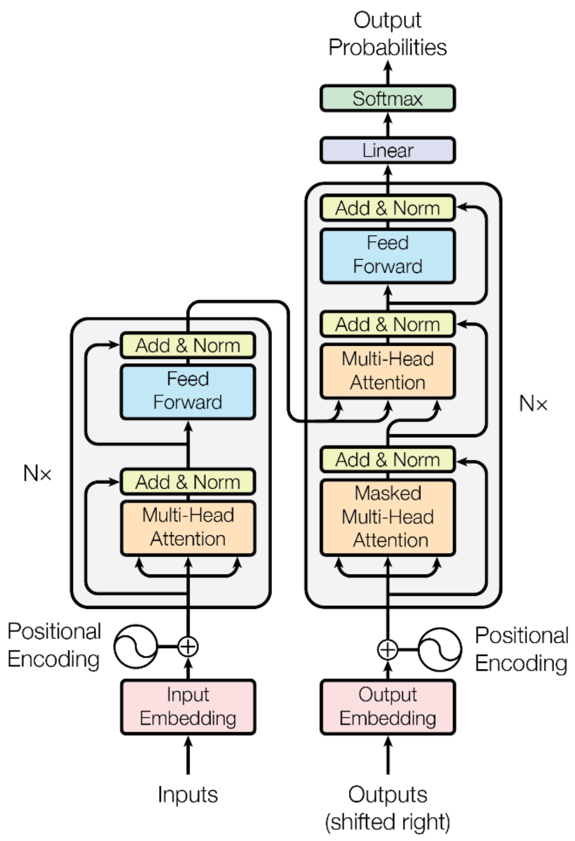


Figure 1: The Transformer - model architecture

Most competitive neural sequence transduction models have an encoder-decoder structure. Here, the encoder maps an input sequence of symbol representations **$(x_1,\ldots,x_n)$** to a sequence of continuous representations **$z=(z_1,\ldots,z_n)$**. Given **z**, the decoder then generates an output sequence **$(y_1,\ldots,y_m)$** of symbols one element at a time. At each step the model is auto-regressive, consuming the previously generated symbols as additional input when generating the next.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

大多数具有竞争力的神经序列转换模型都有一个编码器-解码器结构。在这里，编码器将输入符号表示的序列**$(x_1,\ldots,x_n)$**映射到连续表示的序列**$z=(z_1,\ldots,z_n)$**。给定**z**，解码器然后根据元素顺序生成输出符号序列**$(y_1,\ldots,y_m)$**，一次一个。每一步时，该模型都是自回归的，在生成下一个符号之前，它会将先前生成的符号作为额外输入来使用。

Transformer 遵循这种整体架构，使用堆叠的自注意力和逐点、全连接层来

实现编码器和解码器，如图 1 左半部分和右半部分所示。

3.1 Encoder and Decoder Stacks

Encoder:

The encoder is composed of a stack of **N=6** identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is **LayerNorm(x+Sublayer(x))**, where **Sublayer(x)** is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $\boldsymbol{d}_{\mathbf{model}}\mathbf{=512}$.

Decoder:

The decoder is also composed of a stack of **N=6** identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position $\boldsymbol{i}$ can depend only on the known outputs at positions less than $\boldsymbol{i}$.

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、位置敏感的全连接前馈网络。我们在每个子层周围使用残差连接，并且在其后面使用层归一化。这意味着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，都产生维度为 $\boldsymbol{d}_{\mathbf{model}}\mathbf{=512}$ 的输出。

解码器：

解码器也是由 N=6 个相同的层组成的。在编码器每个层中的两个子层之外，解码器插入了第三个子层，该子层对编码器堆栈输出执行多头注意力。与编码器类似，我们在每个子层周围使用残差连接，并且在其后面使用层归一化。我们还修改了解码器堆栈中的自注意力子层，以防止位置关注到后续位置。这一掩蔽技术，与输出嵌入向右偏移一个位置的事实相结合，确保对位置 ***i*** 的预测仅依赖于小于 ***i*** 的已知输出。

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

注意力函数可以描述为将一个查询（query）和一组键值（key-value）对映射到输出，其中查询、键、值和输出都是向量。输出是通过对值（value）进行加权求和来计算的，每个值的权重由一个兼容性函数根据查询和相应的键来计算。

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

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

The two most commonly used attention functions are additive attention, and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $1/\sqrt{d_k}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients. To counteract this effect, we scale the dot products by $1/\sqrt{d_k}$.

我们称我们的特殊注意力为“Scaled Dot-Product Attention”（图 2）。输入由维度为 d_k 的查询（query）和键（key）以及维度为 d_v 的值（value）组成。我们计算查询（query）与所有键（key）的点积，然后将每个结果除以$\sqrt{d_k}$，应用 softmax 函数来获得值（value）上的权重。

在实际应用中，我们同时计算一组查询（query）的注意力函数，将它们打包到一个矩阵 Q 中。键和值也被打包到矩阵 K 和 V 中。我们计算输出矩阵如下：

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

两个最常用的注意力函数是加性注意力和点积（乘性）注意力。点积注意力与我们的算法相同，除了每个元素除以 $1/\sqrt{d_k}$ 之外。加性注意力使用一个带有单一隐藏层的前馈网络来计算兼容性函数。尽管这两个注意力在理论复杂度上相似，但点积注意力在实际应用中要快得多且更节省空间，因为它可以利用高度优化的矩阵乘法代码实现。

在 d_k 的较小值的情况下，两个机制表现相似，但是当 d_k 的值较大时，加性注意力优于未经缩放的点积注意力。我们怀疑，当 d_k 的值很大时，点积会变得非常大，从而将 softmax 函数推入梯度极其小的区域。为了对抗这种效果，我们通过 $1/\sqrt{d_k}$ 缩放点积。

3.2.2 Multi-Head Attention

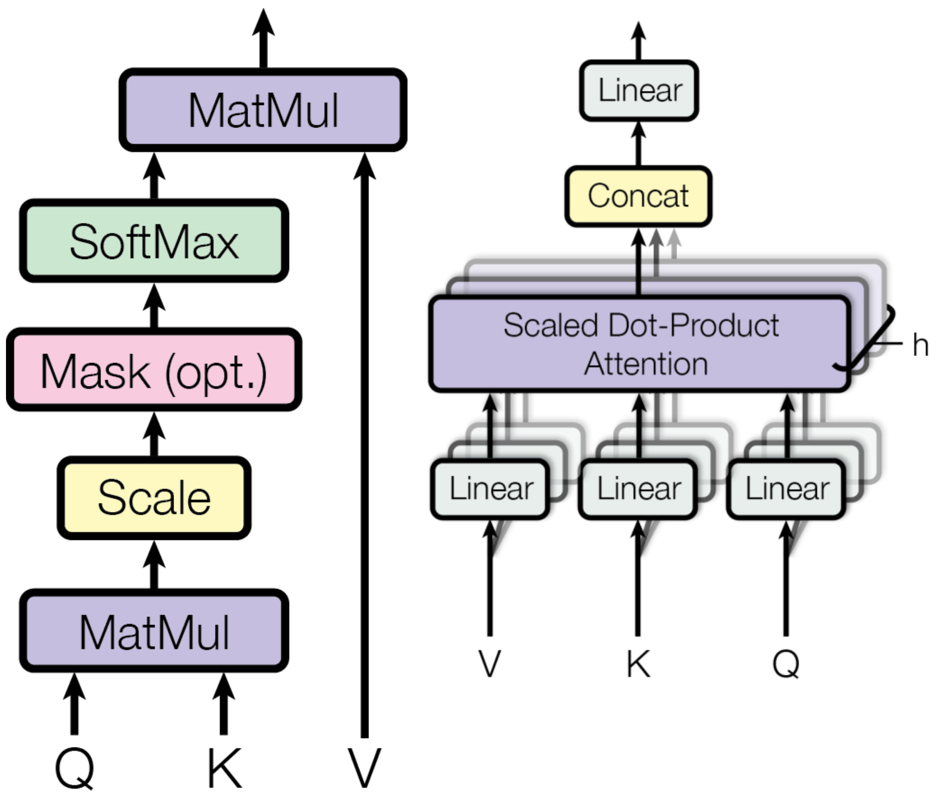


Figure 2 :(left) Scaled Dot-Product Attention.

(right) Multi-Head Attention consists of several attention layers running in parallel.

MatMul:矩阵乘法，Scale:缩放计算，Mask(opt.可选):掩码运算，

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure2.

我们没有直接执行一个单一的注意力函数（attention function）来处理维度为 dmodel 的键（keys）、值（values）和查询（queries）。相反，我们发现将这些键、值和查询线性投影到不同的维度空间，然后在每个维度空间上并行执行注意力函数是有益的。具体来说，我们将键、值和查询线性投影 h 次，每次使用不同的学习到的线性投影，将它们投影到 d_k、d_k 和 d_v 维度空间中。然后，在

这些投影版本的键、值和查询上，我们并行执行注意力函数，得到 d_v 维度的输出值。最后，这些输出值被连接在一起，然后再次被投影到最终值中，如图 2 所示。

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

多头注意力使模型能够同时关注来自不同表示子空间的信息以及位于不同位置的信息。使用单个注意力头时，平均化会抑制这种能力。

$$\begin{aligned}\mathrm{MultiHead}(Q,K,V) &= \mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W^O \\ \text{where } \mathrm{head_i} &= \mathrm{Attention}(QW_i^Q,KW_i^K,VW_i^V)\end{aligned}$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

其中投影矩阵是参数矩阵 $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ 和 $W^O \in \mathbb{R}^{hd_v \times d_{model}}$

In this work we employ h=8 parallel attention layers, or heads. For each of these we use $d_k=d_v=d_{model}/h=64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

在本工作中，我们使用了 h=8 个并行注意力层，也称为头（heads）。对于每一个头，我们都使用 $d_k=d_v=d_{model}/h=64$ 的维度。由于每个头的维度较低，因此总体计算成本与单头全维度注意力的计算成本相似。

3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].

The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.

Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

Transformer 在三个不同的地方使用多头注意力：

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。这类似于典型的编码器-解码器注意力机制，如[38, 2, 9]。

编码器包含自我注意力层。在一个自我注意力层中，所有的键、值和查询都来自同一个地方，在本例中，是前一层编码器的输出。编码器中的每个位置都可以关注到上一层编码器中的所有位置。

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到包括自身在内的所有位置。但是，我们需要阻止解码器中的左向信息流，以保留自回归属性。我们通过在缩放点积注意力中对 softmax 输入中的非法连接进行遮蔽（设置为 $-\infty$）来实现这一点。详见图 2。

3.3 Position-wise Feed-Forward Networks 位置感知前馈网络

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

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

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is d_{model}=512, and the inner-layer has dimensionality d_{ff}=2048.

除了注意力子层以外，我们的编码器和解码器中的每一层还包含一个全连接前馈网络（Fully Connected Feed-Forward Network），它会对每个位置进行独立且相同的处理。这个网络由两个线性变换组成，中间使用 ReLU 激活函数。

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

虽然线性变换在不同的位置上是相同的，但是它们使用了从层到层不同的参数。另一种描述方式是将其视为两个具有核大小 1 的卷积操作。在这种情况下，输入和输出的维度都是 d_{model}=512，而内部层的维度则是 d_{ff}=2048。

3.4 Embeddings and Softmax 嵌入（Embeddings）和 Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model}. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by $\sqrt{}d_{model}$.

与其他序列转换模型类似，我们使用学习到的嵌入（embeddings）将输入标记和输出标记转换为维度为 d_{model} 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下一个标记的概率。在我们的模型中，我们在两个嵌入层和 pre-softmax 线性变换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$\sqrt{}d_{model}$。

3.5 Positional Encoding 位置编码

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed.

In this work, we use sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = sin(pos / 10000^{2i / d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos / 10000^{2i / d_{\text{model}}})$$

Where ***pos*** is the position and ***i*** is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from **2*π*** to **10000·2*π***. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset ***k***, $\boldsymbol{PE_{pos+k}}$ can be represented as a linear function of ***PEpos***.

We also experimented with using learned positional embeddings instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

由于我们的模型不包含递归或卷积，因此为了让模型利用序列的顺序，我们必须将关于标记在序列中的相对或绝对位置信息注入到模型中。为此，我们将“位置编码”（positional encodings）添加到编码器和解码器栈底部的输入嵌入中。位置编码的维度与嵌入相同，即 d_{model}，因此它们可以相加。有很多位置编码选择，既可以学习，也可以固定。

在本研究中，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，每个位置编码的维度都对应一个正弦波。这些波长形成一个从 **2*π*** 到 **10000·2*π*** 的几何级数。我们选择了这个函数，因为我们假设它可以使模型轻松地学习相对位置之间的关系，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型位置嵌入（learned positional embeddings）而不是正弦和余弦函数的方法，并发现这两种版本几乎产生相同的结果（见表 3 行 E）。我们选择了正弦版本，因为它可能使模型能够扩展到比训练过程中遇到的序列长度更长的序列。

4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations $(x_1,\dots,x_n)$ to another sequence of equal length $(z_1,\dots,z_n)$, with $x_i,z_i \in \mathbb{R}^d$, such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

Table 1 Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. **n** is the sequence length, **d** is the representation dimension, **k** is the kernel size of convolutions and **r** the size of the neighborhood in restricted self-attention.

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n / r)$

As noted in Table1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires ***O(n)*** sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence length ***n*** is smaller than the representation dimensionality ***d***, which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece and byte-pair representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size $\boldsymbol{r}$ in the input sequence centered around the respective output position. This would increase the maximum path length to $\boldsymbol{O(n/r)}$. We plan to investigate this approach further in future work.

A single convolutional layer with kernel width $\boldsymbol{k<n}$ does not connect all pairs of input and output positions. Doing so requires a stack of $\boldsymbol{O(n/k)}$ convolutional layers in the case of contiguous kernels, or $\boldsymbol{O(logk(n))}$ in the case of dilated convolutions, increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of $\boldsymbol{k}$. Separable convolutions, however, decrease the complexity considerably, to $\boldsymbol{O(k\cdot n\cdot d+n\cdot d^2)}$. Even with $\boldsymbol{k=n}$, however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model.

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

在本节中，我们将自注意力层的各种方面与通常用于将一个可变长度符号表示序列（$x_1,\ldots,x_n$）映射到另一个等长序列（$z_1,\ldots,z_n$）的递归和卷积层进行比较，其中 $x_i,z_i\in\mathbb{R}^d$，例如在典型序列转换编码器或解码器中的隐藏层。激励我们使用自注意力的三个期望值。

第一个是每层的总计算复杂度（Total Computational Complexity Per Layer）。第二个是可并行化的计算量（Amount of Computation that Can be Parallelized）：这是指可以同时进行的计算量，而不是需要按顺序依次执行。

第三个是网络中长程依赖关系之间的路径长度（Path Length Between Long-Range Dependencies）。在许多序列转换任务中，学习长程依赖关系是一个关键

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。这些路径越短，则学习长程依赖关系就越容易。因此，我们还比较了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。通过这种方式，我们可以评估不同层类型对学习长程依赖关系的影响。

表 1 不同层类型的最大路径长度、每层复杂度和最小数量顺序操作。其中，**n** 是序列长度，**d** 是表示维度，**k** 是卷积核大小，而 **r** 是受限自注意力的邻域大小

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n / r)$

如表 1 所示，自注意力层（self-attention layer）可以通过常数数量的顺序操作连接所有位置，而递归层（recurrent layer）则需要 ***O(n)***个顺序操作。在计算复杂度方面，当序列长度 ***n*** 小于表示维度 ***d*** 时，自注意力层比递归层快。这在机器翻译中的状态艺术模型中很常见，例如使用 word-piece 和 byte-pair 表示的句子。为了改善长序列任务的计算性能，可以限制自注意力仅考虑输入序列中心化输出位置周围大小为 ***r*** 的邻域内的元素。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。

单个卷积层（convolutional layer）在 kernel 宽度 ***k<n*** 的情况下，并不能连接所有的输入和输出位置对。要实现这一点，需要一个栈中的 ***O(n/k)***个卷积层，在连续核（contiguous kernels）的情况下，或是 ***O(logk(n))***个卷积层，在扩张核（dilated convolutions）的情况下，这会增加网络中任意两个位置之间的最长路径长度。卷积层通常比递归层更昂贵，乘以一个因子 ***k***。然而，可分离卷积（separable convolutions）可以显著降低复杂度，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等于自注意力层和点式前馈网络（point-wise feed-forward layer）的组合，我们的模型采取了这种方法。

作为额外的优点，自注意力可能会产生更容易解释的模型。我们检查了来自我们模型的注意力分布，并在附录中呈现和讨论了一些例子。不仅仅是每个注意力头单独学习执行不同的任务，许多注意力头似乎展现出与句子的语法结

构和语义结构相关的行为。

5 Training

This section describes the training regime for our models.

本节描述了我们模型的训练制度。

5.1 Training Data and Batching 训练数据和批处理

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding, which has a shared source-target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

我们使用 WMT 2014 的标准英德数据集进行训练，包含约 4.5 百万个句子对。句子编码使用字节对编码（Byte Pair Encoding），共享源目标词汇表约 37000 个标记符。对于英法，我们使用了更大的 WMT 2014 英法数据集，包含 3600 万个句子，并将标记符分成 32000 个词元（word-piece）。句子对根据大致的序列长度进行批处理。每个训练批次包含一个句子对集合，约有 25000 个源标记符和 25000 个目标标记符。

5.2 Hardware and Schedule 硬件和时间表

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the bottom line of table3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

我们在一台机器上使用 8 个 NVIDIA P100 GPU 训练了我们的模型。对于使用本文描述的超参数的基本模型，每一步训练大约需要 0.4 秒。我们将基本模型训练了 100,000 步，总共 12 小时。对于我们的大型模型（在表 3 的底行中描述），每一步训练时间为 1.0 秒。大型模型被训练了 300,000 步（3.5 天）。

5.3 Optimizer 优化器

We used the Adam optimizer with $\boldsymbol{\beta_1}$**=0.9**, $\boldsymbol{\beta_2}$**=0.98** and $\boldsymbol{\epsilon}$**=10**$^{\boldsymbol{-9}}$. We varied the learning rate over the course of training, according to the formula:

我们使用了 Adam 优化器，设置参数为 $\boldsymbol{\beta_1}$**=0.9**, $\boldsymbol{\beta_2}$**=0.98** 和 $\boldsymbol{\epsilon}$**=10**$^{\boldsymbol{-9}}$。在训练过程中，我们根据以下公式调节学习率

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

This corresponds to increasing the learning rate linearly for the first ***warmup_steps*** training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used ***warmup_steps***=4000.

这对应于在训练的前 warmup_steps 步骤中线性增加学习率，然后在之后按照步骤数的逆平方根进行减少。我们使用了 warmup_steps=4000。

5.4 Regularization 规范化

We employ three types of regularization during training:

我们在训练过程中采用了三种正则化方法：

Residual Dropout

We apply dropout to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of $\mathbf{P}_{\boldsymbol{drop}}$**=0.1**.

我们将 dropout 应用于每个子层的输出上，在它们被添加到子层输入并规范化之前。此外，我们还将 dropout 应用于编码器和解码器栈中嵌入式表示和位置编码的总和。对于基本模型，我们使用了一个丢弃率 $\mathbf{P}_{\boldsymbol{drop}}$**=0.1**。

Label Smoothing

During training, we employed label smoothing of value ϵ_{ls}=0.1. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

在训练过程中，我们使用了标签平滑（label smoothing）技术，具体值为 ϵ_{ls}=0.1。这会降低模型的困惑度（perplexity），因为模型变得更加不确定，但是

却提高了准确率和 BLEU 得分。

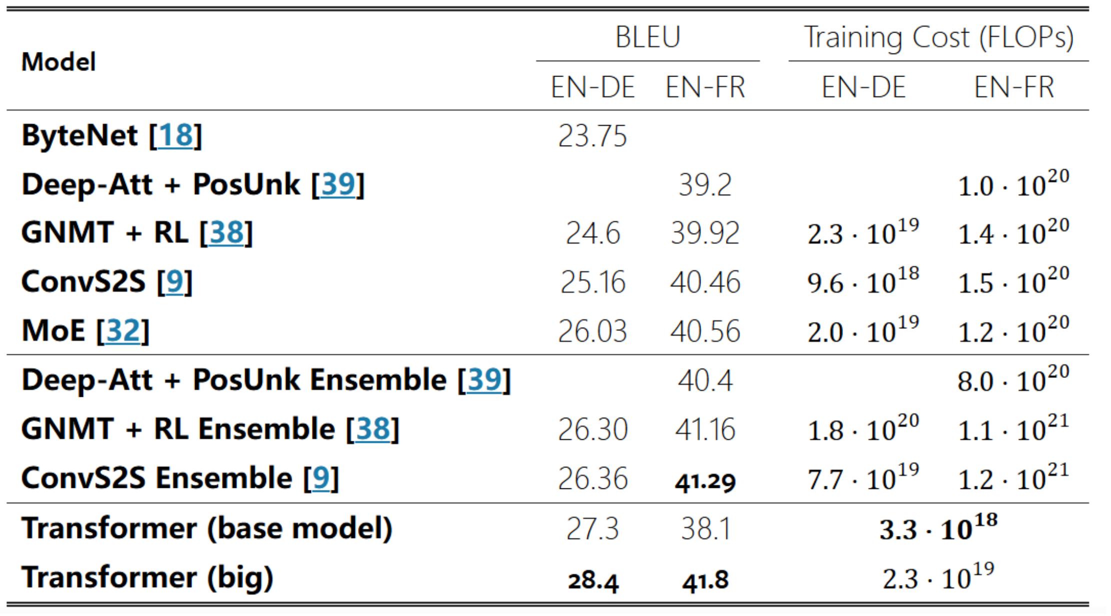
6 Results

6.1 Machine Translation

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Transformer

在新测试集 newstest2014 的英语-德语和英语-法语翻译任务上取得了比之前最优模型更好的 BLEU 评分，同时训练成本仅为原来的几分之一。



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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

在 WMT 2014 英德翻译任务中，大型 Transformer 模型（Transformer (big) 在表 2 中）比之前所有最佳报告的模型（包括集成模型）都要好，超过了 2.0 BLEU 得分，建立了新的最优 BLEU 得分 28.4。这个模型的配置在表 3 的底行列出。训练耗时 3.5 天，使用 8 个 P100 GPU。甚至我们的基础模型也超越了所

有之前发布的模型和集成模型，只需付出其中任何一个竞争模型的一小部分训练成本。

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of **41.0**, outperforming all of the previously published single models, at less than **1/4** the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate $\boldsymbol{P_{drop}}$=0.1, instead of 0.3.

在 WMT 2014 英法翻译任务中，我们的大型模型取得了 41.0 的 BLEU 得分，超越了所有之前发布的单个模型，在训练成本方面仅为之前最优模型的四分之一。用于英语到法语的 Transformer (big) 模型使用的 dropout 率是 $\boldsymbol{P_{drop}}$ =0.1，而不是 0.3。

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty α=0.6. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible.

对于基础模型，我们使用了单个模型，该模型是通过平均最后 5 个检查点获得的，这些检查点以 10 分钟间隔写入。对于大型模型，我们平均了最后 20 个检查点。我们使用了带有长度惩罚 α=0.6 的 beam search，beam 大小为 4。这些超参数是在开发集上实验后选定的。在推断过程中，我们将最大输出长度设置为输入长度+50，但在可能的情况下会提前终止。

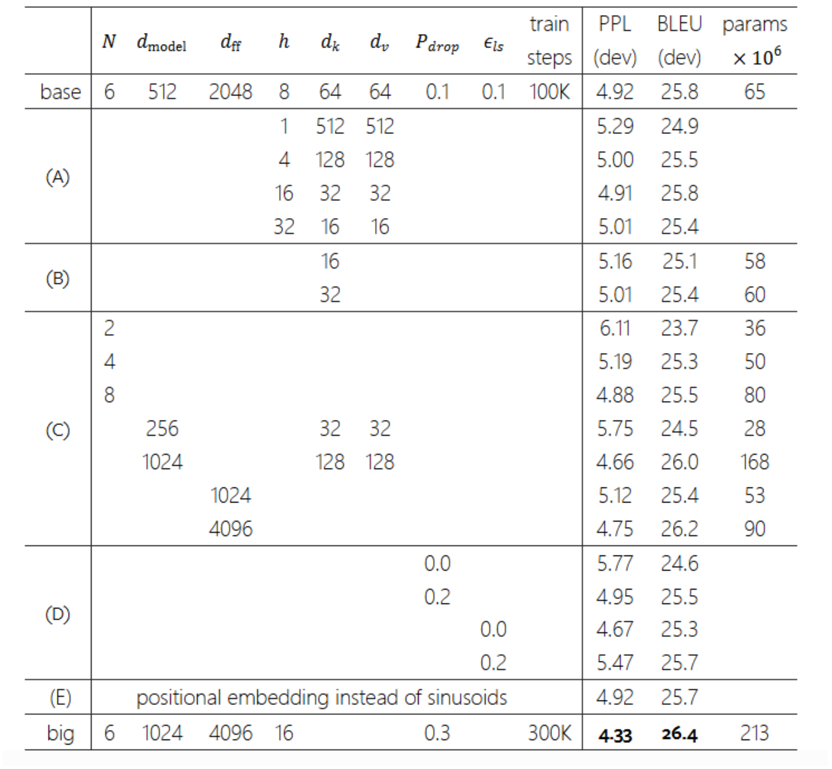
Table 2 summarizes our results and compares our translation quality and training costs to other model architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU.

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比较。我们通过将训练时间、使用的 GPU 数量以及每个 GPU 持续单精度浮点运算能力的估计值相乘来估计训练一个模型所需的浮点运算数。

6.2 Model Variations 模型变体

Table 3 Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

Transformer 架构的变体。未列出的值与基础模型的值相同。所有指标都在英语到德语翻译开发集 newstest2013 上。根据我们的字节对编码，列出的困惑是每个单词的，不应该与每个单词的困惑进行比较。



	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

To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3.

为了评估 Transformer 的不同组件的重要性，我们以不同的方式修改了我们的基础模型，测量了性能变化在英语-德语翻译任务上，在开发集 newstest2013 上。我们使用了前一节中描述的 beam search，但没有检查点平均。我们将这些结果呈现于表 3。

In Table 3 rows (A), we vary the number of attention heads and the attention key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

在表 3 的行（A）中，我们改变了注意力头数和注意力键值维度，保持计算量恒定，如第 3.2.2 节所述。虽然单头注意力比最佳设置差 0.9 BLEU，但质量也会随着太多头而降低。

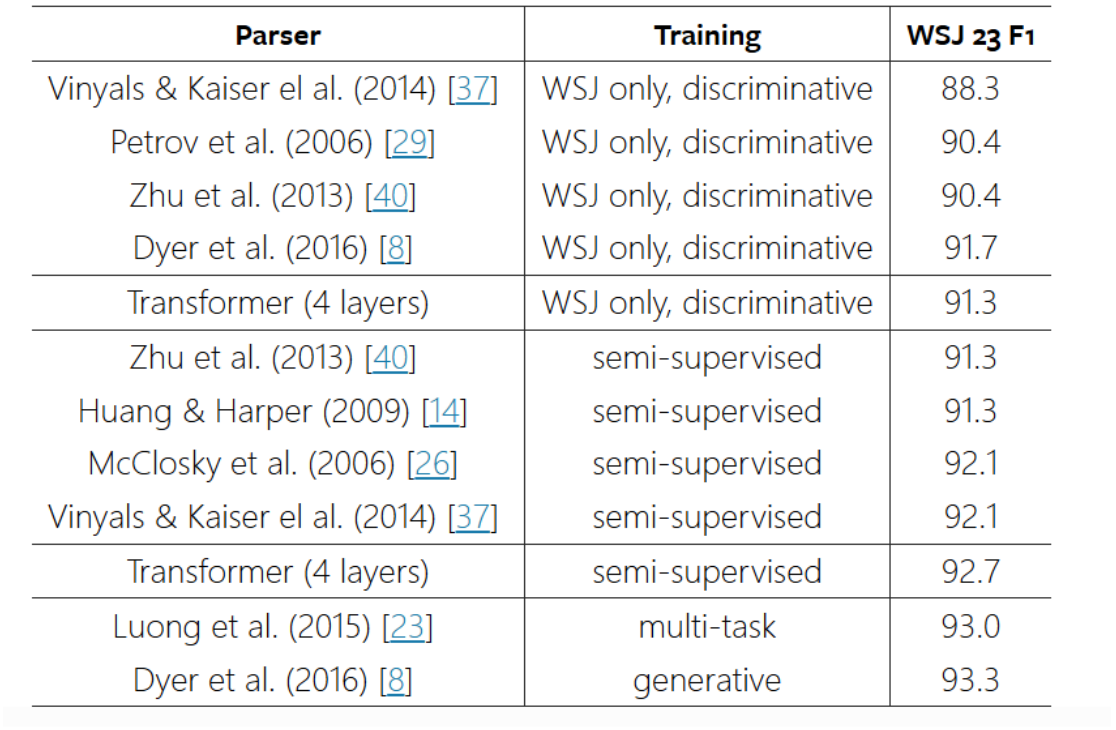
In Table 3 rows (B), we observe that reducing the attention key size $\boldsymbol{d_k}$ hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings, and observe nearly identical results to the base model.

在表 3 的行（B）中，我们观察到降低注意力键值大小 $\boldsymbol{d_k}$ 会损害模型质量。这表明确定兼容性不是很容易，并且可能需要一个比点积更复杂的兼容函数。在行（C）和（D）中，我们进一步观察到，正如预期的那样，更大的模型是更好的，而 dropout 在避免过拟合方面非常有帮助。在行（E）中，我们用学习到的位置嵌入替换了我们的正弦位置编码，并观察到与基础模型几乎相同的结果。

6.3 English Constituency Parsing 英语选区解析

Table 4The Transformer generalizes well to English constituency parsing (Results are on Section 23 of

WSJ) Transformer 在英语句法分析（WSJ 第 23 节的结果）方面表现良好



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

To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. This task presents specific challenges: the output is subject to strong structural constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes.

为了评估 Transformer 是否能推广到其他任务，我们在英语句法分析上进行了实验。这个任务带来了特定的挑战：输出受强结构约束，并且比输入要长得多。此外，在小数据量的情景下，RNN 序列-序列模型尚未能达到最优结果。

We trained a 4-layer transformer with d_{model}=1024 on the Wall Street Journal (WSJ) portion of the Penn Treebank, about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting.

我们在彭恩树库（Penn Treebank）的华尔街日报（WSJ）部分上训练了一个 4 层的 Transformer 模型，输入维度为 1024，约有 40K 个训练句子。我们还

在半监督设置下对其进行了训练，使用来自 BerkleyParser 的大型高置信度语料库和大约 1700 万个句子的语料库。对于仅 WSJ 设置，我们使用了一个 16K 的词汇表，而对于半监督设置，我们使用了一个 32K 的词汇表。

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. During inference, we increased the maximum output length to input length **+300**. We used a beam size of **21** and **α=0.3** for both WSJ only and the semi-supervised setting.

我们仅进行了少量的实验来选择 dropout、attention 和残差（第 5.4 节）、学习率和 beam 大小在第 22 节的开发集上，所有其他参数保持不变，从英文到德文基础翻译模型。在推理过程中，我们将最大输出长度增加到输入长度 + 300。 我们使用了 WSJ 仅有的 beam 大小为 21 和 α=0.3，以及半监督设置下的 beam 大小为 21 和 α=0.3。

Our results in Table4 show that despite the lack of task-specific tuning our model performs surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar.

表 4 中的结果显示，即使没有任务特定的调优，模型仍然表现出惊人的好成绩，比之前所有已发表的模型都有更好的结果，只有一个例外，那就是循环神经网络语法（Recurrent Neural Network Grammar）。

In contrast to RNN sequence-to-sequence models, the Transformer outperforms the BerkeleyParser even when training only on the WSJ training set of 40K sentences.

与 RNN 序列-序列模型相比，Transformer 甚至在仅训练于 WSJ 训练集的 40K 句子时，也超过了 BerkleyParser。

7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

在本工作中，我们提出了 Transformer，它是第一个完全基于注意力的序列

转换模型，通过多头自注意力替代了编码器-解码器架构中常用的循环层。

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

对于翻译任务，Transformer 可以比基于循环或卷积层的架构训练得更快。在 WMT 2014 英语-德语和 WMT 2014 英语-法语两个翻译任务上，我们取得了新的最优结果。 在前一个任务中，我们最好的模型甚至超过了所有以前报告过的集成。

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

我们对基于注意力的模型的未来感到兴奋，并计划将它们应用于其他任务。我们计划扩展 Transformer 到处理非文本输入和输出模态的问题，例如图像、音频和视频，并研究局部限制性注意力机制，以高效地处理大型输入和输出。使生成变得更不顺序是我们的另一个研究目标。

The code we used to train and evaluate our models is available at https://github.com/tensorflow/tensor2tensor.