# Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer∗
Google Brain
noam@google.com

Niki Parmar∗
Google Research
nikip@google.com

Łukasz Kaiser∗
Google Brain
lukaszkaiser@google.com

Llion Jones∗
Google Research
llion@google.com

Aidan N. Gomez∗
University of Toronto
aidan@cs.toronto.edu

Jakob Uszkoreit∗
Google Research
usz@google.com

Illia Polosukhin∗
Google Research
illia.polosukhin@gmail.com

## 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 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

主流的序列转导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。表现最佳的模型还通过注意力机制连接编码器和解码器。我们提出了一种新的简单网络架构，Transformer，完全基于注意力机制，完全不使用递归和卷积。在两个机器翻译任务上的实验表明，这些模型在质量上优于其他模型，同时更具并行性，并且训练所需时间显著减少。我们的模型在 WMT 2014 英语到德语翻译任务上达到了 28.4 BLEU，超过了现有最佳结果，包括集成模型，提升了超过 2 BLEU。在 WMT 2014 英语到法语翻译任务上，我们的模型在八个 GPU 上训练 3.5 天后，创造了新的单模型最先进的 BLEU 分数 41.0，训练成本仅为文献中最佳模型的一小部分。

## 1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] 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 [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

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 [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

循环神经网络，尤其是长短期记忆 [13] 和门控循环 [7] 神经网络，已牢固确立为序列建模和传导问题（例如语言建模和机器翻译 [35, 2, 5]）的先进方法。此后，人们不断努力突破循环语言模型和编码器-解码器架构的界限 [38, 24, 15]。
循环模型通常根据输入和输出序列的符号位置分解计算。将位置与计算时间的步骤对齐，它们会生成隐藏状态 $h_t$ 序列，作为前一个隐藏状态 $h_{t-1}$ 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。最近的研究通过分解技巧 [21] 和条件计算 [32] 显著提高了计算效率，同时在后者的情况下也提高了模型性能。然而，顺序计算的基本约束仍然存在。

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 [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

注意力机制已成为各种任务中引人注目的序列建模和传导模型不可或缺的一部分，允许对依赖关系进行建模，而无需考虑它们在输入或输出序列中的距离 [2, 19]。然而，除了少数情况 [27] 外，此类注意力机制都与循环网络结合使用。

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.

在这项研究中，我们提出了 Transformer，这是一种避免循环 (recurrence) 结构的模型架构，它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 可以实现更高的并行化，在 8 个 P100 GPU 上训练 12 小时后，就可以达到翻译质量的新水平。

## 2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], 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 [12]. 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 section .

减少顺序计算的目标也构成了扩展神经 GPU [16]、ByteNet [18] 和 ConvS2S [9] 的基础，这些模型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的信号所需的操作数量随着位置之间的距离而增长，对于 ConvS2S 是线性增长，对于 ByteNet 是对数增长。这使得学习远距离位置之间的依赖关系变得更加困难 [12]。在 Transformer 中，这被减少为一个恒定的操作数量，尽管由于平均注意加权位置而导致有效分辨率降低，这是我们在第 节中通过多头注意力来抵消的效果。

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 [4, 27, 28, 22].

自注意力，有时称为内部注意力，是一种将单个序列的不同位置关联起来以计算序列表示的注意机制。自注意力已成功应用于多种任务，包括阅读理解、抽象摘要、文本蕴涵和学习任务无关的句子表示 [4, 27, 28, 22]。

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

* 平等贡献。列出顺序是随机的。Jakob 提出了用自注意力替代 RNN，并开始评估这一想法的工作。Ashish 与 Illia 设计并实现了第一个 Transformer 模型，并在这项工作中的每个方面都发挥了关键作用。Noam 提出了缩放点积注意力、多头注意力和无参数位置表示，并成为几乎每个细节的另一位参与者。Niki 在我们的原始代码库和 tensor2tensor 中设计、实现、调整和评估了无数模型变体。Llion 还实验了新颖的模型变体，负责我们的初始代码库，以及高效推理和可视化。Lukasz 和 Aidan 花费了无数个漫长的日子设计和实现 tensor2tensor 的各个部分，替换了我们早期的代码库，大大改善了结果并大幅加速了我们的研究。

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 [28].

端到端记忆网络基于递归注意机制，而不是序列对齐的递归，并且已被证明在简单语言问答和语言建模任务中表现良好 [28]。

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 [14, 15] and [8].

然而，尽我们所知，Transformer 是第一个完全依赖自注意力机制来计算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。在接下来的部分中，我们将描述 Transformer，阐明自注意力机制，并讨论其相对于模型 [14, 15] 和 [8] 的优势。

## 3 Model Architecture **模型架构**

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations z = $(z_1, ..., z_n)$. Given z, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next.

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 $(x_1 ... x_n)$ 映射到连续表示序列 z = $(z_1, ..., z_n)$。给定 z，解码器然后一次一个元素地生成符号的输出序列 $(y_1 ... y_m)$。在每个步骤中，模型都是自回归的 [10]，在生成下一个符号时使用先前生成的符号作为附加输入。

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.

Transformer 遵循这种整体架构，使用堆叠的自注意力机制和逐点的全连接层，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 Encoder and Decoder Stacks **编码器和解码器堆栈**

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 [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\text{LayerNorm}(x + \text{Sublayer}(x))$, where $\text{Sublayer}(x)$ 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 $d_{\text{model}} = 512$.

编码器: 编码器由一组 $N = 6$ 相同的层组成。每层有两个子层。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。我们在每个子层周围采用残差连接 [10]，然后进行层归一化 [1]。也就是说，每个子层的输出是 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 $d_{\text{model}} = 512$ 的输出。

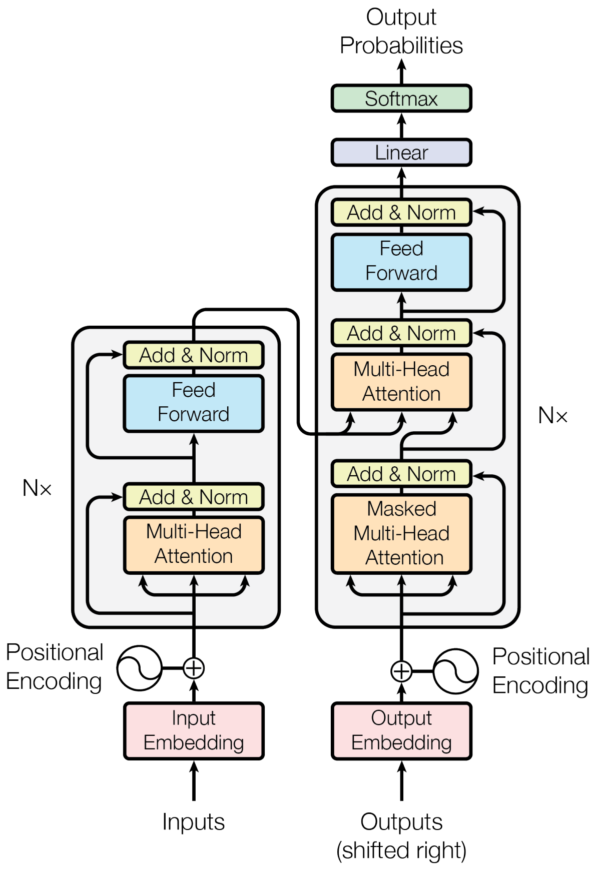

Figure 1: The Transformer - model architecture.

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 $i$ can depend only on the known outputs at positions less than $i$.

解码器: 解码器也由一组 $N = 6$ 相同的层组成。除了每个编码器层中的两个子层外，解码器还插入了第三个子层，该子层对编码器堆栈的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，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，以防止位置关注后续位置。这种掩蔽，加上输出嵌入偏移一个位置，确保位置 $i$ 的预测只能依赖于位置小于 $i$ 的已知输出。

## 3.2 Attention **注意力**

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查询和一组键值对映射到输出，其中查询、键、值和输出都是向量。输出作为值的加权和计算，其中分配给每个值的权重由查询与相应键的兼容性函数计算得出。

### 3.2.1 Scaled Dot-Product Attention **缩放点积注意力**

We call our particular attention ”Scaled Dot-Product Attention” (Figure 2). 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.

我们将我们的特定注意力称为 “缩放点积注意力” (图 2)。输入由维度为 $d_k$ 的查询和键，以及维度为 $d_v$ 的值组成。我们计算查询与所有键的点积，将每个点积除以 $\sqrt{d_k}$ ，并应用 softmax 函数以获得值的权重。

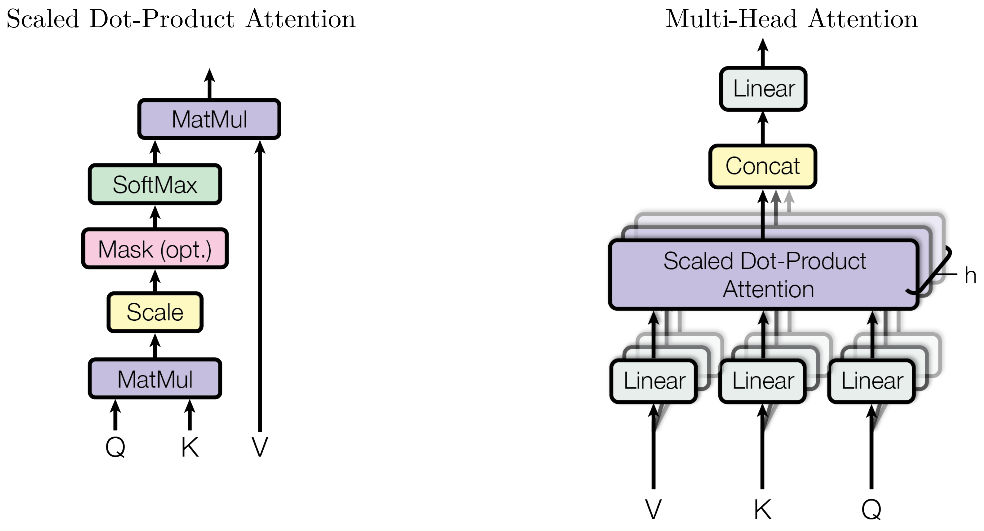

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.
(左) 缩放点积注意力. (右) 多头注意力由多个并行运行的注意力层组成.

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:

在实践中，我们同时对一组查询计算注意力函数，这些查询被打包成一个矩阵 $Q$ 。键和值也被打包成矩阵 $K$ 和 $V$ 。我们计算输出矩阵如下:

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

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$ . 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.

最常用的两种注意力函数是加性注意力 [2] 和点积 (乘法) 注意力。点积注意力与我们的算法相同，除了缩放因子 $\frac{1}{\sqrt{d_k}}$ 。加性注意力使用具有单个隐藏层的前馈网络计算兼容性函数。虽然这两者在理论复杂性上相似，但在实践中，点积注意力要快得多且更节省空间，因为它可以使用高度优化的矩阵乘法代码实现。

While for small values of $d_k$ the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of $d_k$ [3].

虽然对于小值 $d_k$ 两种机制的表现相似，但加性注意力在较大值 $d_k$ 时优于未缩放的点积注意力 [3]。

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 $\frac{1}{\sqrt{d_k}}$ .

我们怀疑对于大值 $d_k$ ，点积的大小会增大，使得 softmax 函数进入梯度极小的区域。为了抵消这种影响，我们将点积缩放为 $\frac{1}{\sqrt{d_k}}$ 。

### 3.2.2 Multi-Head Attention **多头注意力**

Instead of performing a single attention function with $d_{\text{model}}$ -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 Figure 2.

我们发现，与其使用 $d_{\text{model}}$ 维的键、值和查询执行单一的注意力函数，不如将查询、键和值线性投影 $h$ 次，使用不同的学习线性投影到 $d_k, d_k$ 和 $d_v$ 维度。在这些投影版本的查询、键和值上，我们并行执行注意力函数，产生 $d_v$ 维的输出值。这些值被连接并再次投影，最终得到如图 2 所示的值。

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在不同位置共同关注来自不同表示子空间的信息。使用单个注意力头时，平均会抑制这一点。

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \\ \text{where head}_\text{i} &= \text{Attention}(QW_{Q_i}^{d_{\text{model}} \times d_q}, KW_{K_i}^{d_{\text{model}} \times d_k}, VW_i^{V\, d_{\text{model}} \times d_v})\end{aligned}$$

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

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

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

在这项工作中，我们采用 $h = 8$ 并行注意力层或头。对于每个头，我们使用 $d_k = d_v = d_{\text{model}}/h = 64$ 由于每个头的维度减少，总的计算成本与具有完整维度的单头注意力相似。

[4] To illustrate why the dot products get large, assume that the components of $q$ and $k$ are independent random variables with mean 0 and variance 1 . Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$ , has mean 0 and variance $d_k$ .

为了说明为什么点积会变大，假设 $q$ 和 $k$ 的分量是均值为 0、方差为 1 的独立随机变量。那么它们的点积 $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$ 的均值为 0，方差为 $d_k$ 。

### 3.2.3 Applications of Attention in our Model **我们模型中注意力的应用**

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

Transformer 以三种不同的方式使用多头注意力:

- 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 [31, 2, 8].

- 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典型的编码器-解码器注意力机制，例如 [31, 2, 8]。

- 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.

- 编码器包含自注意力层。在自注意力层中，所有键、值和查询都来自同一位置，在本例中，即编码器中上一层的输出。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。

- 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.

- 类似地，解码器中的自注意力层允许解码器中的每个位置关注解码器中该位置及之前的所有位置。我们需要防止解码器中的左向信息流动，以保持自回归特性。我们通过在缩放点积注意力中屏蔽 (设置为 $-\infty$ ) 所有与非法连接对应的 softmax 输入值来实现这一点。见图 2。

## 3.3 Position-wise Feed-Forward Networks **位置-wise 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.

除了注意力子层外，我们的编码器和解码器中的每一层都包含一个完全连接的前馈网络，该网络分别且相同地应用于每个位置。这由两个线性变换和中间的 ReLU 激活组成。

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

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 $d_{\text{model}} = 512$ , and the inner-layer has dimensionality $d_{ff} = 2048$ .

虽然不同位置的线性变换是相同的，但它们在层与层之间使用不同的参数。另一种描述方式是将其视为两个卷积，卷积核大小为 1。输入和输出的维度为 $d_{\text{model}} = 512$ ，而内层的维度为 $d_{ff} = 2048$

## 3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension $d_{\text{model}}$ . 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 [24]. In the embedding layers, we multiply those weights by $\sqrt{d_{\text{model}}}$ .

与其他序列转导模型类似，我们使用学习到的嵌入将输入 tokens 和输出 tokens 转换为维度为 $d_{\text{model}}$ 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的下一个 tokens 概率。在我们的模型中，我们在两个嵌入层和预 softmax 线性变换之间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 $\sqrt{d_{\text{model}}}$ 。

## 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 $d_{\text{model}}$ as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [8].

由于我们的模型不包含递归和卷积，为了使模型能够利用序列的顺序，我们必须注入一些关于序列中 tokens 的相对或绝对位置的信息。为此，我们在编码器和解码器堆栈的底部向输入嵌入添加“位置编码”。位置编码与嵌入具有相同的维度 $d_{\text{model}}$ ，以便可以相加。位置编码有许多选择，包括学习到的和固定的 [8]。

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

在这项工作中，我们使用不同频率的正弦和余弦函数:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right)$$

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

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 $2\pi$ to $10000 \cdot 2\pi$ . 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 $k, PE_{pos+k}$ can be represented as a linear function of $PE_{pos}$ .

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波。波长从 $2\pi$ 到 $10000 \cdot 2\pi$ 形成几何级数。我们选择这个函数是因为我们假设它可以让模型更容易地通过相对位置进行注意，因为对于任何固定的偏移量，$k, PE_{pos+k}$ 可以表示为 $PE_{pos}$ 的线性函数。

We also experimented with using learned positional embeddings [8] 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.

我们还尝试使用学习到的位置嵌入 [8]，发现这两种版本产生了几乎相同的结果 (见表 3 行 (E))。我们选择正弦版本是因为它可能允许模型推断出比训练期间遇到的序列长度更长的序列。

## 4 Why Self-Attention 为什么自注意力

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 $(x_1, \ldots, x_n)$ to another sequence of equal length $(z_1, \ldots, z_n)$ , 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.

在本节中，我们比较自注意力层与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 $(z_1, \ldots, z_n)$ 的递归和卷积层的各个方面，具有 $x_i, z_i \in \mathbb{R}^d$ ，例如典型序列转导编码器或解码器中的隐藏层。为了激励我们使用自注意力，我们考虑三个期望。

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.

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，以所需的最小顺序操作数来衡量。

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 [12]. Hence we compare the maximum path length between any two input and output positions in networks composed of the different layer types.

第三个是网络中长距离依赖的路径长度。学习长距离依赖是许多序列转导任务中的一个关键挑战。影响学习这种依赖能力的一个关键因素是前向和后向信号在网络中必须遍历的路径长度。输入和输出序列中任何位置组合之间的路径越短，学习长距离依赖就越容易 [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。

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.
不同层类型的最大路径长度、每层复杂度和最小连续操作数。$n$ 是序列长度，$d$ 是表示维度，$k$ 是卷积的核大小，$r$ 是受限自注意力中的邻域大小。

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

As noted in Table 1, 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 [38] and byte-pair [31] representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size $r$ in the input sequence centered around the respective output position. This would increase the maximum path length to $O(n/r)$. We plan to investigate this approach further in future work.

如表 1 所示，自注意力层以恒定数量的顺序执行操作连接所有位置，而递归层则需要 $O(n)$ 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，自注意力层比递归层更快，这在机器翻译的最先进模型中使用的句子表示中通常是这种情况，例如 word-piece [38] 和 byte-pair [31] 表示。为了提高处理非常长序列的任务的计算性能，自注意力可以限制为仅考虑围绕相应输出位置的输入序列中的大小为 $r$ 的邻域。这将使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研究这种方法。

A single convolutional layer with kernel width $k < n$ does not connect all pairs of input and output positions. Doing so requires a stack of $O(n/k)$ convolutional layers in the case of contiguous kernels, or $O(\log_k(n))$ in the case of dilated convolutions [15], 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 $k$ . Separable convolutions [6], however, decrease the complexity considerably, to $O\left(k \cdot n \cdot d + n \cdot d^2\right)$ . Even with $k = n$ , 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.

单个卷积层的卷积核宽度 $k < n$ 并不连接所有输入和输出位置对。这样做需要在连续卷积核的情况下堆叠 $O(n/k)$ 个卷积层，或者在扩张卷积的情况下 $O(\log_k(n))$ ，从而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，增加了 $k$ 的倍数。然而，分离卷积 [6] 显著降低了复杂性，降至 $O\left(k \cdot n \cdot d + n \cdot d^2\right)$ 。即使在 $k = n$ 的情况下，分离卷积的复杂性仍然等于自注意力层和逐点前馈层的组合，这是我们模型中采用的方法。

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.

作为附带好处，自注意力可能产生更具可解释性的模型。我们检查了模型的注意力分布，并在附录中展示和讨论了示例。单个注意力头不仅明显学习执行不同的任务，许多似乎还表现出与句子的句法和语义结构相关的行为。

## 5 Training **训练**

This section describes the training regime for our models.

本节描述了我们模型的训练方案。

### 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 [3], 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 [31]. 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.

我们在标准的 WMT 2014 英德数据集上进行了训练，该数据集包含约 450 万对句子。句子使用字节对编码 [3] 进行编码，该编码具有约 37000 个 tokens 的共享源-目标词汇表。对于英法数据集，我们使用了显著更大的 WMT 2014 英法数据集，包含 3600 万个句子，并将 tokens 拆分为 32000 个词片词汇表 [31]。句子对根据近似序列长度进行批处理。每个训练批次包含一组句子对，约包含 25000 个源 tokens 和 25000 个目标 tokens。

### 5.2 Hardware and Schedule **硬件和调度**

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 table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

我们在一台配备 8 个 NVIDIA P100 GPU 的机器上训练我们的模型。对于我们使用的基础模型，采用了本文中描述的超参数，每个训练步骤大约需要 0.4 秒。我们对基础模型进行了总计 100,000 步或 12 小时的训练。对于我们的大型模型 (在表 3 的底行中描述)，每步时间为 1.0 秒。大型模型训练了 300,000 步 (3.5 天)。

### 5.3 Optimizer **优化器**

We used the Adam optimizer [17] with $\beta_1 = 0.9, \beta_2 = 0.98$ and $\epsilon = 10^{-9}$ . We varied the learning rate over the course of training, according to the formula:

> 我们使用了 Adam 优化器 [17]，并使用了 $\beta_1 = 0.9, \beta_2 = 0.98$ 和 $\epsilon = 10^{-9}$ 。我们在训练过程中根据以下公式变化学习率:

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

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 .

> 这对应于在前 warmup_steps 训练步骤中线性增加学习率，并在此之后按步骤编号的倒数平方根成比例地降低学习率。我们使用了 warmup_steps = 4000 。

### 5.4 Regularization **正则化**

We employ three types of regularization during training:

> 我们在训练过程中采用三种类型的正则化:

Residual Dropout We apply dropout [27] 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 $P_{\text{drop}} = 0.1$ .

> 残差丢弃我们在每个子层的输出上应用丢弃 [27]，在其被添加到子层输入并进行归一化之前。此外，我们在编码器和解码器堆栈中对嵌入和位置编码的和应用丢弃。对于基础模型，我们使用的比率为 $P_{\text{drop}} = 0.1$ 。

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [30]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

> 在训练过程中，我们采用了值为 $\epsilon_{ls} = 0.1$ 的标签平滑。这会影响困惑度，因为模型学习变得更加不确定，但提高了准确性和 BLEU 分数。

## 6 Results

### 6.1 Machine Translation **机器翻译**

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（表 2 中的 Transformer (big)）的表现比之前报告的最佳模型（包括集成）高出 2.0 BLEU，创下了 28.4 的最新 BLEU 得分。此模型的配置列在表 3 的底部。在 8 个 P100 GPU 上训练耗时 3.5 天。即使是我们的基础模型也超越了所有之前发布的模型和集成，而训练成本仅为任何竞争模型的一小部分。

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 $P_{\text{drop}} = 0.1$ , instead of 0.3 .

在 WMT 2014 英语到德语翻译任务中，大型 transformer 模型 (表 2 中的 Transformer (big)) 比之前报告的最佳模型 (包括集成模型) 高出超过 2.0 BLEU，建立了 28.4 的新的最先进 BLEU 分数。该模型的配置列在表 3 的底行。训练在 8 个 P100 GPU 上进行了 3.5 天。即使是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，训练成本仅为任何竞争模型的一小部分。
在 WMT 2014 英法翻译任务中，我们的大模型取得了 41.0 的 BLEU 分数，优于之前发布的所有单一模型，而训练成本不到之前最先进模型的 1/4。针对英法翻译进行训练的 Transformer (big) 模型使用的 dropout 率为 $P_{\text{drop}} = 0.1$ ，而不是 0.3 。

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 在英语到德语和英语到法语的 newstest2014 测试中，取得了比以前的最先进模型更好的 BLEU 分数，同时训练成本却低得多。

| 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 | $2.3 \cdot 10^{19}$ | |

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 $\alpha = 0.6$ [31]. 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 [31].

对于基础模型，我们使用了通过平均最后 5 个检查点获得的单一模型，这些检查点是以 10 分钟的间隔写入的。对于大型模型，我们平均了最后 20 个检查点。我们使用了束搜索，束大小为 4，长度惩罚为 $\alpha = 0.6$ [31]。这些超参数是在开发集上实验后选择的。我们在推理期间将最大输出长度设置为输入长度 +50，但在可能的情况下提前终止 [31]。

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 的持续单精度浮点能力的估计相乘，来估算训练模型所使用的浮点运算次数. [5]

[5] We used values of 2.8,3.7,6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.
我们对 K80、K40、M40 和 P100 使用了 2.8、3.7、6.0 和 9.5 TFLOPS 的值。

## 6.2 Model Variations **模型变体**

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.
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.

为了评估 Transformer 不同组件的重要性，我们以不同方式改变了基础模型，测量在开发集 newstest2013 上英德翻译性能的变化。我们使用了前一节中描述的束搜索，但没有检查点平均。我们在表 3 中展示了这些结果。
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，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低 0.9 BLEU，但过多的头也会导致质量下降。

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相同。所有指标均基于英德翻译开发集 newstest2013。列出的困惑度是按字词片段计算的，依据我们的字节对编码，不应与按字计算的困惑度进行比较。

| | $N$ | $d_{\text{model}}$ | | $h$ | $d_k$ | $d_v$ | $P_{drop}$ | $\epsilon_{ls}$ | 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 |

In Table 3 rows (B), we observe that reducing the attention key size $d_k$ 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 [8], and observe nearly identical results to the base model.

在表 3 的行 (B) 中，我们观察到减少注意力键大小 $d_k$ 会损害模型质量。这表明确定兼容性并不容易，可能需要比点积更复杂的兼容性函数。我们进一步观察到在行 (C) 和 (D) 中，正如预期的那样，更大的模型表现更好，而 dropout 在避免过拟合方面非常有帮助。在行 (E) 中，我们用学习的位置信息嵌入 [8] 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 **英语成分分析**

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

| 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 [37].

为了评估 Transformer 是否可以推广到其他任务，我们对英语成分分析进行了实验。这项任务提出了特定的挑战：输出受到严格的结构约束，并且比输入长得多。此外，RNN 序列到序列模型在小数据范围内无法获得最先进的结果 [37]。

We trained a 4-layer transformer with $d_{model} = 1024$ on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], 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 [37]. 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 Penn Treebank [25] 的华尔街日报 (WSJ) 部分训练了一个 4 层转换器，其中 $d_{model} = 1024$，大约有 40K 个训练句子。我们还在半监督设置中对其进行了训练，使用更大的高置信度和 BerkleyParser 语料库，其中大约有 17M 个句子 [37]。我们在 WSJ 专用设置中使用了 16K 个 tokens 的词汇表，在半监督设置中使用了 32K 个 tokens 的词汇表。

We performed only a small number of experiments to select the dropout, both attention and residual (section ??), 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 $\alpha = 0.3$ for both WSJ only and the semi-supervised setting.

我们仅进行了少量实验来选择第 22 节开发集上的 dropout、注意力和残差 (section ??)、学习率和波束大小，所有其他参数与英语到德语基础翻译模型保持不变。在推理过程中，我们将最大输出长度增加到输入长度 + 300。我们对 WSJ 和半监督设置都使用了 21 的波束大小和 $\alpha = 0.3$。

Our results in Table 4 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 [8].

In contrast to RNN sequence-to-sequence models [37], the Transformer outperforms the BerkeleyParser [29] even when training only on the WSJ training set of 40K sentences.

表 4 中的结果表明，尽管缺乏针对特定任务的调整，但我们的模型表现得出奇的好，除循环神经网络语法 [8] 外，其结果比所有以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 个句子的 WSJ 训练集上进行训练，Transformer 的表现也优于 BerkeleyParser [29]。

## 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递归层。

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 的训练速度显著快于基于递归或卷积层的架构。在 WMT 2014 英语到德语和 WMT 2014 英语到法语的翻译任务中，我们达到了新的最先进水平。在前一个任务中，我们的最佳模型甚至超越了所有之前报告的集成模型。

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、受限的注意力机制，以高效处理大型输入和输出，如图像、音频和视频。减少生成的顺序性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

我们用于训练和评估模型的代码可在 https://github.com/tensorflow/tensor2tensor 获取。

Acknowledgements We are grateful to Nal Kalchbrenner and Stephan Gouws for their fruitful comments, corrections and inspiration.

**致谢** 我们感谢 Nal Kalchbrenner 和 Stephan Gouws 的宝贵意见、修正和启发。

## References

[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.