Provided proper attribution is provided, Google hereby grants permission to reproduce the tables and figures in this paper solely for use in journalistic or scholarly works.

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

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

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

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

[cs.CL] 2 Aug 2023

arXiv:1706.03762v7

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.

*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.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

在提供适当署名的前提下，谷歌特此授权允许复制本文中的表格和图表，但仅限于新闻报道或学术研究用途。

注意力机制，尽在其中

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar*谷歌研究院 nikip@google.com

Jakob Uszkoreit*谷歌研究院 usz@google.com

Llion Jones* Google Research llion@google.com

艾丹·N·戈麦斯* †多伦多大学 aidan@cs.toronto.edu

Łukasz Kaiser*Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

[cs.CL] 2 Aug 2023

arXiv:1706.03762v7

摘要

主流序列转换模型基于包含编码器与解码器的复杂循环或卷积神经网络。顶尖模型还通过注意力机制连接编解码器。我们提出全新简化网络架构——Transformer，仅基于注意力机制，完全摒弃循环与卷积结构。 在两项机器翻译任务的实验中，该模型展现出更优的翻译质量，同时具备更强的并行处理能力，训练时间显著缩短。在WMT 2014英语-德语翻译任务中，本模型取得28.4的BLEU得分，较现有最佳结果（包括集成模型）提升超过2分。 在WMT 2014英语-法语翻译任务中，我们的模型仅用8块GPU训练3.5天便创下41.8的单模型BLEU新纪录，其训练成本仅为文献中顶尖模型的极小部分。通过在海量与有限训练数据下成功应用于英语构成性解析任务，我们证明Transformer模型具有良好的泛化能力。

*贡献均等。列举顺序随机。Jakob提出用自注意力替代循环神经网络，并启动了该理念的评估工作。Ashish与Illia共同设计并实现了首批Transformer模型，全程深度参与本项目各环节。Noam提出缩放点积注意力、多头注意力及无参数位置表示方案，成为另一位参与几乎所有细节的设计者。 Niki在原始代码库及tensor2tensor框架中设计、实现、调优并评估了无数模型变体。Llion同样探索了新型模型变体，负责初始代码库构建，并实现了高效推理与可视化功能。Lukasz和Aidan耗费无数个漫长日夜设计并实现tensor2tensor的各个组件，取代了早期代码库，显著提升了研究成果并极大加速了研究进程。

†本研究成果完成于谷歌大脑期间。

‡本研究成果完成于谷歌研究院任职期间。

第31届神经信息处理系统会议（NIPS 2017），美国加利福尼亚州长滩市。

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.

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.

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.

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 3.2.

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 [4, 27, 28, 22].

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 [34].

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].

3 Model Architecture

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 $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, 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.

1 引言

循环神经网络，特别是长短期记忆网络 [13] 和门控循环神经网络 [7] ，已在序列建模与转换问题（如语言建模和机器翻译 [35, 2, 5]）中确立为顶尖解决方案。此后众多研究持续拓展循环语言模型与编码器-解码器架构 [38, 24, 15]的边界。

循环模型通常将计算分解为输入输出序列的符号位置。通过将位置与计算时间步长对齐，它们生成隐状态序列 h_t，该序列是前一隐状态 h_{t-1} 与当前位置输入 t的函数。这种固有的序列化特性阻碍了训练样本内的并行化处理——当序列长度增加时，由于内存限制导致跨样本分批处理受限，该问题尤为关键。 近期研究通过因子分解技巧 [21]和条件计算 [32], 显著提升了计算效率，后者同时改善了模型性能。然而序列计算的根本限制依然存在。

注意力机制已成为各类任务中高效序列建模与转换模型的核心组件，能够在不考虑输入或输出序列中依赖关系距离的情况下进行建模 [2, 19]。然而除极少数情况外 [27], , 此类注意力机制通常与循环神经网络结合使用。

本研究提出的Transformer模型架构摒弃了递归机制，完全依赖注意力机制来建立输入与输出间的全局关联。该模型支持更高效的并行化处理，仅需在八块P100 GPU上训练十二小时，即可达到翻译质量的新标杆水平。

2 背景

减少序列计算的目标也构成了扩展神经GPU[16], ByteNet [18] 和ConvS2S [9], 的基础，这些模型均采用卷积神经网络作为基本构建模块，通过并行计算所有输入和输出位置的隐藏表示。 在这些模型中，关联任意输入或输出位置信号所需的运算量随位置间距增长：ConvS2S呈线性增长，ByteNet呈对数增长。 这使得学习远距离位置间的依赖关系变得更为困难 [12]。在Transformer模型中，该操作量被简化为常量级，但代价是因注意力加权位置的平均化导致有效分辨率降低——我们在第3.2节中通过多头注意力机制抵消了这一影响。

自注意力（亦称序列内注意力）是一种关联单一序列中不同位置的注意力机制，用于计算序列的表示。该机制已在阅读理解、摘要生成、文本蕴含判断及任务无关的句子表示学习等多种任务中成功应用 [4, 27, 28, 22]。

端到端记忆网络基于递归注意力机制而非序列对齐递归机制，已在简单语言问答和语言建模任务中展现出优异性能 [34]。

然而据我们所知，Transformer是首个完全依赖自注意力机制计算输入输出表示的转换模型，无需序列对齐的RNN或卷积操作。后续章节将阐述Transformer架构，阐明自注意力的设计动机，并探讨其相较于 [17, 18] 和 [9]等模型的优势。

3 模型架构

最具竞争力的神经序列转换模型均采用编码器-解码器结构 [5, 2, 35]。其中编码器将输入符号序列表示（$x_1, ..., x_n$）映射为连续表示序列$\mathbf{z} = (z_1, ..., z_n)$。 解码器接收 $\mathbf{z}$ 后，逐元素生成符号输出序列（$y_1, ..., y_m$）。模型在每个步骤均采用自回归机制[10], , 即生成下一个符号时会将先前生成的符号作为额外输入进行处理。

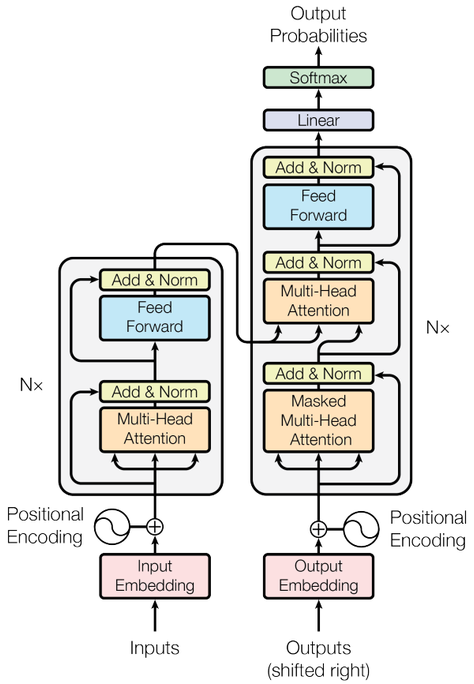

Figure 1: The Transformer - model architecture.

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.

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

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.

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

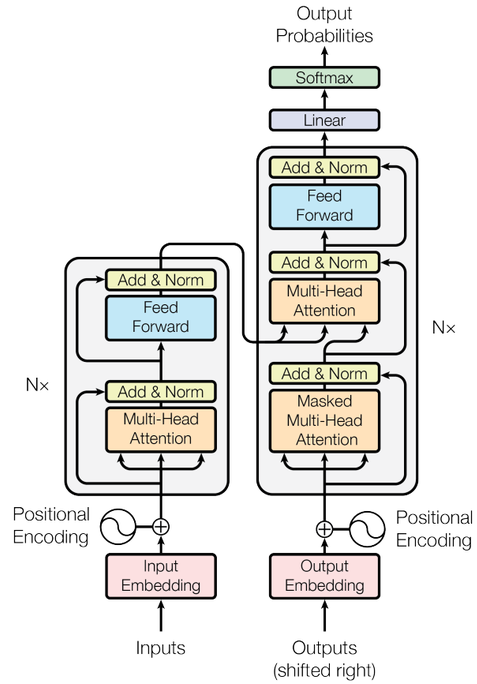

图1：Transformer模型架构示意图。

Transformer模型采用堆叠式自注意力机制与点积全连接层构建整体架构，编码器与解码器分别如图1左、右半部分所示。

3.1 编码器-解码器堆栈

编码器：编码器由堆叠的 $N = 6$ 相同层构成。每层包含两个子层：首层为多头自注意力机制，次层为简单的位置全连接前馈网络。我们在两个子层间均设置残差连接 [11]，随后进行层归一化 [1]。即每个子层的输出为LayerNorm(子层(子层))，其中子层(子层)是子层自身实现的函数。为便于这些残差连接，模型中所有子层及嵌入层的输出维度均为。

解码器：解码器同样由一组 $N = 6$ 个相同的层组成。除每个编码器层的两个子层外，解码器还插入第三个子层，该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采用残差连接，随后进行层归一化。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。这种掩码机制结合输出嵌入偏移一个位置的特性，确保对位置 i 的预测仅依赖于位置小于 i的已知输出。

3.2 注意力机制

注意力函数可描述为将查询与一组键值对映射至输出，其中查询、键、值及输出均为向量。输出通过加权求和计算得出。

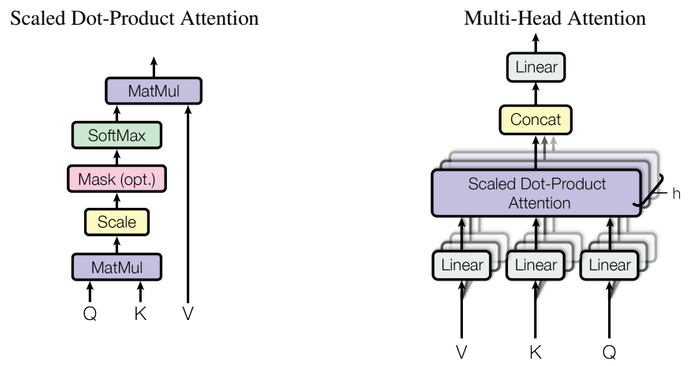

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

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.

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:

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

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.

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]. 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 [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

3.2.2 Multi-Head Attention

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

[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.

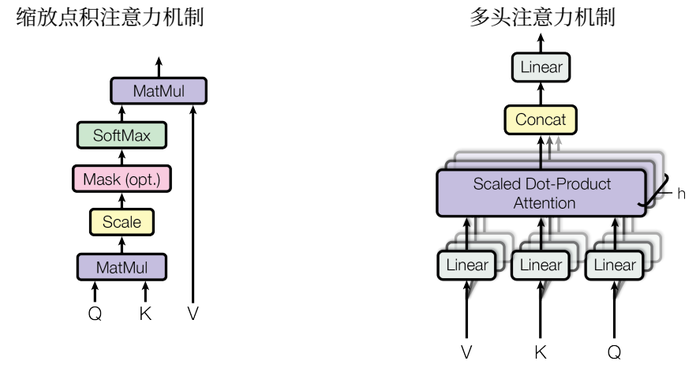

图2：（左）缩放点积注意力机制。（右）多头注意力由多个并行运行的注意力层构成。

其中权重由查询与对应键的兼容性函数计算得出。

3.2.1 缩放点积注意力机制

我们特别关注"缩放点积注意力"机制（图2）。输入包含维度为 d_k的查询与键，以及维度为 d_v的值。我们计算查询与所有键的点积，将每个结果除以 $\sqrt{d_k}$，并应用softmax函数以获得值的权重。

实际计算中，我们将一组查询同时打包为矩阵 Q，并计算其注意力函数。键值对同样打包为矩阵 K 和 V。输出矩阵的计算方式如下：

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

最常用的两种注意力函数是加法注意力 [2], 和点积（乘法）注意力。 点积注意力与我们的算法完全一致，仅在 $\frac{1}{\sqrt{d_k}}$ 的缩放因子上存在差异。加法注意力通过单隐藏层的前馈网络计算兼容性函数。尽管两者在理论复杂度上相似，但点积注意力在实践中速度更快且空间效率更高，因为它可通过高度优化的矩阵乘法代码实现。

当 d_k 取较小值时，两种机制表现相近；但当 d_k [3]增大时，加法注意力在无需缩放的情况下优于点积注意力。我们推测当d_k值较大时，点积数值会急剧增大，导致softmax函数进入梯度极小的区域 [4]。为抵消此效应，我们通过 $\frac{1}{\sqrt{d_k}}$ 对点积进行缩放处理。

3.2.2 多头注意力机制

我们发现，相较于仅对维度为 d_{model}的键、值和查询执行单次注意力计算，将查询、键和值分别通过不同的学习线性投影线性投射至维度为 d_k、 d_k 和 d_v 的维度 h 次，更为有效。随后在这些投影后的查询、键和值版本上并行执行注意力函数，最终生成维度为 d_v的

[4]为说明点积值增大的原因，假设 q 和 k 的分量均为均值为0、方差为1的独立随机变量。则其点积 $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$具有均值0、方差 d_k的特性。

output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

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.

$$\begin{aligned}\text{MultiHead}(Q,K,V) &= \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O \\ \text{where head}_\text{i} &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

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 $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$.

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.

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.

3.3 Position-wise Feed-Forward Networks

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.

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

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 $d_{ff} = 2048$.

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_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 $\sqrt{d_\text{model}}$.

输出值。这些值被连接后再次进行投影，最终得到如图2所示的结果。

多头注意力机制使模型能够同时关注不同位置、不同表示子空间的信息。单头注意力中的平均机制会抑制这种协同处理能力。

$$\begin{aligned}\text{MultiHead}(Q,K,V) &= \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O \\ \text{where head}_\text{i} &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

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

本研究采用 $h = 8$ 个并行注意力层（即头部）。每个头部使用$d_k = d_v = d_\text{model}/h = 64$个注意力单元。由于各头部维度缩减，总计算成本与全维度单头注意力模型相当。

3.2.3 注意力机制在我们模型中的应用

Transformer通过三种方式运用多头注意力机制：

在"编码器-解码器注意力"层中，查询来自前一解码层，而记忆键与值则源自编码器输出。这使得解码器中的每个位置都能关注输入序列中的所有位置，从而模拟了[38, 2, 9]等序列到序列模型中典型的编码器-解码器注意力机制。

编码器包含自注意力层。在自注意力层中，所有键、值和查询均来自同一位置——即编码器前一层的输出。编码器中每个位置都能关注编码器前一层的所有位置。

• 同样地，解码器中的自注意力层允许每个位置关注解码器中所有位置（包括该位置本身）。为保持自回归特性，需阻止解码器中的左向信息流动。我们通过缩放点积注意力机制实现此目标：将软最大化输入中所有非法连接对应的值屏蔽（设为 $-\infty$）。详见图 2。

3.3 位置级前馈网络

除注意力子层外，编码器与解码器的每个层级均包含全连接前馈网络，该网络对每个位置独立且完全相同地应用。其结构包含两个线性变换，中间夹有aReLU激活函数。

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

尽管不同位置的线性变换相同，但各层使用的参数各不相同。另一种描述方式是两个卷积操作，其卷积核尺寸均为1。输入与输出的维度均为 $d_\text{model} = 512$，而内部层的维度为 $d_{ff} = 2048$。

3.4 嵌入与Softmax

与其他序列转换模型类似，我们采用学习到的嵌入向量将输入/输出标记转换为维度为 d_model 的向量。同时使用常规的线性转换与softmax函数，将解码器输出转换为预测的下一个标记概率。 在我们的模型中，两个嵌入层与软最大化前线性变换共享相同的权重矩阵，类似于[30]。在嵌入层中，我们将这些权重与 $\sqrt{d_\text{model}}$相乘。

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

3.5 Positional Encoding

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 [9].

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 2π 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}.

We also experimented with using learned positional embeddings [9] 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.

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 $(x_1, ..., x_n)$ to another sequence of equal length $(z_1, ..., z_n)$, 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 [12]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

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

表1：不同层类型的最大路径长度、每层复杂度及最小序列操作次数。 n 表示序列长度， d 表示表示维度， k 表示卷积核尺寸， r 表示受限自注意力中的邻域尺寸。

层类型	每层复杂度	序列操作	最大路径长度
自注意力机制	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
循环	$O(n \cdot d^2)$	$O(n)$	$O(n)$
卷积	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
自注意力（受限）	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 位置编码

由于我们的模型不含递归和卷积结构，为使模型利用序列顺序信息，必须注入关于序列中词符相对或绝对位置的信息。为此，我们在编码器和解码器堆栈底部的输入嵌入中添加"位置编码"。 位置编码与嵌入向量具有相同维度 d_{model}，以便两者相加。位置编码存在多种选择方案，包括学习型和固定型 [9]。

本研究采用不同频率的正弦与余弦函数：

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

其中 pos 表示位置， i 表示维度。即位置编码的每个维度对应一条正弦曲线。 波长从 2π 到 $10000 \cdot 2\pi$呈等比递增。选择此函数是基于以下假设：当偏移量 k固定时， PE_{pos+k} 可表示为PE_{pos}的线性函数，这将使模型能轻松学习通过相对位置进行注意力分配。

我们还尝试使用学习到的位置嵌入 [9] ，发现两种版本产生的结果几乎相同（见表3第(E)行）。我们选择正弦版本是因为它可能使模型能够外推到比训练期间遇到的更长的序列长度。

4 为何采用自注意力机制

本节将从多个维度对比自注意力层与循环层、卷积层的特性——后两者常用于将变长符号序列表示（$x_1, ..., x_n$）映射至等长序列（$z_1, ..., z_n$） ，如典型序列转换编码器或解码器中的隐藏层。我们基于三个核心诉求采用自注意力机制：

其一是每层的总计算复杂度。其二是可并行化的计算量，以所需最小序列操作数衡量。

第三个关键因素是网络中长程依赖关系的路径长度。学习长程依赖关系是许多序列转换任务的核心挑战。 影响学习此类依赖关系的关键因素之一，是前向与后向信号在网络中必须穿越的路径长度。输入序列与输出序列中任意位置组合间的路径越短，学习长程依赖关系就越容易 [12]。因此我们还比较了由不同层类型组成的网络中，任意两个输入与输出位置间的最大路径长度。

如表1所示， 自注意力层通过恒定数量的顺序操作连接所有位置， 而循环层需执行 $O(n)$次顺序操作。在序列长度为时， 自注意力层的计算复杂度低于循环层。

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.

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 [18], 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 $O(k \cdot n \cdot d + n \cdot d^2)$. 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.

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.

5 Training

This section describes the training regime for our models.

5.1 Training Data and Batching

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 [38]. 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.

5.2 Hardware and Schedule

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).

5.3 Optimizer

We used the Adam optimizer [20] 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:

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

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 $warmup\_steps = 4000$.

5.4 Regularization

We employ three types of regularization during training:

长度 n 小于表示维度 d，这种情况在机器翻译前沿模型使用的句子表示中最为常见，例如词片[38] 和字节对 [31] 表示。 为提升超长序列任务的计算性能，可将自注意力机制限定为仅考虑以输出位置为中心的输入序列邻域，其大小为 r 。此举将使最大路径长度提升至 $O(n/r)$。我们计划在后续研究中深入探索该方法。

单个卷积层（其核宽度为 $k < n$ ）无法连接所有输入与输出位置的配对。 若要实现全连接，连续卷积核需堆叠 $O(n/k)$层卷积层，而膨胀卷积则需堆叠 $O(log_k(n))$层，这将增加网络中任意两位置间最长路径的长度。卷积层的计算成本通常比循环层高出 k倍。 然而分离卷积 [6], 能显著降低复杂度至 $O(k \cdot n \cdot d + n \cdot d^2)$。即便采用 $k = n$，分离卷积的复杂度仍等同于自注意力层与点前馈层的组合——这正是我们模型所采用的方法。

附带优势在于，自注意力机制能生成更可解释的模型。我们在附录中展示并讨论了模型注意力分布的实例分析。不仅各注意力头部能清晰执行不同任务，许多头部还呈现出与句子语法结构及语义特征相关的行为模式。

5 训练

本节描述了我们模型的训练方案。

5.1 训练数据与批处理

我们采用标准WMT 2014英德语数据集进行训练，该数据集包含约450万对句子。句子采用字节对编码 [3], 进行编码，其源目标词汇表共享约37000个词素。 针对英法翻译，我们采用了规模显著更大的WMT 2014英法数据集（含3600万个句子），并将词元拆分至32000词的词元词表 [38]。句子对按近似序列长度分批处理，每个训练批次包含约25000个源词元和25000个目标词元的句子对集合。

5.2 硬件与时间安排

我们在配备8块NVIDIA P100 GPU的单台机器上完成模型训练。 采用论文所述超参数的基础模型，每次训练步耗时约0.4秒。基础模型总训练步数为100,000步（12小时）。大型模型（见表3末行）的单步耗时为1.0秒，训练步数达300,000步（3.5天）。

5.3 优化器

我们采用Adam优化器 [20] 配合 $\beta_1 = 0.9$, $\beta_2 = 0.98$ 和 $\epsilon = 10^{-9}$参数。根据公式：

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

这相当于在前 $warmup\_steps$ 个训练步骤中线性增加学习率，之后按步骤数的平方根反比线性递减。我们采用了$warmup_steps = 4000$参数。

5.4 正则化

我们在训练过程中采用三类正则化：

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.

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

Residual Dropout We apply dropout [33] 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 $P_{drop} = 0.1$.

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

6 Results

6.1 Machine Translation

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.

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 $P_{drop} = 0.1$, instead of 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 $\alpha = 0.6$ [38]. 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 [38].

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 [5].

6.2 Model Variations

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

[5]We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

表2：Transformer模型在英语-德语及英语-法语新闻测试2014数据集上取得优于前沿模型的BLEU分数，且训练成本仅为前者的几分之一。

模型	BLEU		训练成本（浮点运算次数）	
	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（基础模型）	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

残差式Dropout我们在每个子层输出添加至子层输入并归一化前，对其应用Dropout [33] 。此外，在编码器和解码器堆栈中，我们对嵌入向量与位置编码的和应用Dropout。基础模型采用$P_{drop} = 0.1$的概率率。

标签平滑训练过程中采用值 $\epsilon_{ls} = 0.1$ [36]的标签平滑技术。该方法虽会降低困惑度（模型学会表现出更高不确定性），但能提升准确率与BLEU得分。

6 项结果

6.1 机器翻译

在WMT 2014英语-德语翻译任务中，大型Transformer模型（表2中的Transformer (big)）以28.4的BLEU得分刷新历史纪录，较此前最佳模型（含集成模型）提升超过2.0分。该模型配置详见表3末行。 在8块P100 GPU上训练耗时3.5天。即使基础模型也以远低于竞争模型的训练成本，超越了所有已发表模型及集成方案。

在WMT 2014英语-法语翻译任务中，我们的大型模型以41.0的BLEU得分超越所有已发表的单一模型，且训练成本低于 $1/4$ 前沿模型的 $P_{drop} = 0.1$倍。该英语-法语Transformer（大型）模型采用 $P_{drop} = 0.1$的dropout率替代0.3参数。

基础模型采用最后5个检查点的平均值（每10分钟保存一次），大型模型则采用最后20个检查点的平均值。采用束搜索算法，束宽为4，长度惩罚值为 $\alpha = 0.6$ [38]。这些超参数经开发集实验后确定。 推理阶段将最大输出长度设为输入长度 + 50，但尽可能提前终止 [38]。

表2总结了我们的研究成果，并将翻译质量与训练成本与文献中其他模型架构进行了对比。我们通过将训练时间、所用GPU数量以及每块GPU的持续单精度浮点运算能力估算值相乘，来估算模型训练所需的浮点运算次数 [5]。

6.2 模型变体

为评估Transformer各组件的重要性，我们通过多种方式调整基础模型，并测量其在英语到德语翻译任务上的性能变化。

[5]我们使用了2.8、3. K80、K40、M40及P100显卡分别提供7、6.0和9.5 TFLOPS浮点运算性能 , respectively.

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.

	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

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.

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 [9], and observe nearly identical results to the base model.

6.3 English Constituency Parsing

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].

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.

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

表3：Transformer架构变体。未列值与基础模型相同。所有指标均基于英语-德语翻译开发集newstest2013。所列困惑度采用基于字节对编码的词片单位计算，不可与基于单词的困惑度进行比较。

	N	$d_{\text{模型}}$	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	训练步骤	PPL (dev)	BLEU (dev)	参数 $\times 10^6$
基础模型	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)	位置嵌入替代正弦曲线									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

开发集为newstest2013。我们采用前文所述的束搜索算法，但未进行检查点平均处理。相关结果见表3。

在表3的(A)行中，我们改变注意力头数量及注意力键值维度，同时保持计算量恒定（详见第3.2.2节）。单头注意力模型比最佳设置低0.9 BLEU分，但过多头数也会导致质量下降。

在表3的(B)行中，我们观察到缩减注意力键尺寸 d_k 会损害模型质量。这表明兼容性判断并非易事，采用比点积更复杂的兼容性函数可能更有效。在(C)和(D)行中我们进一步发现：如预期所示，更大规模的模型表现更优，且dropout机制对避免过拟合极具帮助。在(E)行中，我们将正弦位置编码替换为学习型位置嵌入 [9]，结果与基础模型几乎完全一致。

6.3 英语构成成分解析

为评估Transformer模型能否推广至其他任务，我们针对英语构成性解析开展了实验。该任务存在特殊挑战：输出结果受强结构约束且长度远超输入。此外，在小数据场景下，RNN序列到序列模型尚未能达到最先进水平 [37]。

我们使用 $d_{model} = 1024$ 在宾夕法尼亚语法库 [25], 的《华尔街日报》部分（约4万训练句）上训练了4层Transformer模型。我们还采用半监督训练模式，使用规模更大的高置信度语料库和BerkleyParser语料库（约1700万句）进行训练[37]。WSJ专属训练采用1.6万词汇表，半监督训练则采用3.2万词汇表。

我们仅在第22节开发集上进行了少量实验来选择dropout（包括注意力与残差机制，详见第5.4节）、学习率及束大小，其余参数均沿用英语-德语基础翻译模型的设置。在推理阶段，我们

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

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.

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 Berkeley-Parser [29] even when training only on the WSJ training set of 40K sentences.

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.

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.

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.

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

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

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.

表4：Transformer模型在英语构成成分解析任务中表现出良好泛化能力（结果基于WSJ第23版）

解析器	训练	WSJ 23 F1
Vinyals & Kaiser 等 (2014) [37]	仅限WSJ数据集，判别式模型	88.3
Petrov等人（2006） [29]	仅限WSJ数据集，判别式模型	90.4
朱等（2013） [40]	仅限WSJ数据集，判别式模型	90.4
Dyer等人（2016） [8]	仅限WSJ数据集，判别式模型	91.7
Transformer（4层）	仅限WSJ数据集，判别式模型	91.3
朱等（2013） [40]	半监督	91.3
Huang & Harper (2009) [14]	半监督	91.3
McClosky等人（2006） [26]	半监督	92.1
Vinyals & Kaiser 等 (2014) [37]	半监督	92.1
Transformer（4层）	半监督	92.7
Luong等人（2015） [23]	多任务	93.0
Dyer等人（2016） [8]	生成式	93.3

incr 将最大输出长度放宽至输入长度 + 300。我们采用21的束大小和 $\alpha = 0.$ 3
for 无论仅使用WSJ数据集还是半监督设置。

表4结果表明，尽管未进行任务特定调优，本模型仍表现出惊人优势，除循环神经网络语法[8]外，其性能超越所有已报道模型。"

相较于循环神经网络序列到序列模型 [37]，Transformer模型在仅使用4万句WSJ训练集训练时 [29]，其性能仍超越伯克利解析器。

7 结论

本研究提出Transformer模型——首个完全基于注意力机制的序列转换模型，通过多头自注意力机制替代编码器-解码器架构中最常用的循环层。

对于翻译任务，Transformer的训练速度远快于基于循环或卷积层的架构。在WMT 2014英语-德语及英语-法语翻译任务中，我们均创下新纪录。在前者任务中，我们的最佳模型甚至超越了所有已报道的集成模型。

我们对基于注意力机制的模型前景充满期待，计划将其应用于其他任务领域。未来将探索将Transformer模型扩展至文本以外的输入输出模态，并研究局部限定注意力机制以高效处理图像、音频、视频等大规模输入输出数据。实现非序列化生成亦是我们的重要研究目标。

我们用于模型训练与评估的代码已开源，详见：https://github.com/tensorflow/tensor2tensor。

Ac 鸣谢我们衷心感谢Nal Kalchbrenner与Stephan Gouws所作的卓有成效的贡献。 1
co 评论、修正与灵感。

参考文献

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 层归一化. arXiv预印本 arXiv:1607.06450, 2016.[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 通过协同学习对齐与翻译实现神经机器翻译。CoRR, abs/1409.0473, 2014.[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. 神经机器翻译架构的大规模探索。 CoRR，abs/1703.03906，2017年。[4] 程建鹏、李东、米雷拉·拉帕塔。用于机器阅读的长短期记忆网络。arXiv预印本arXiv:1601.06733，2016年。

[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.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 基于RNN编码器-解码器的短语表示学习在统计机器翻译中的应用。CoRR, abs/1406.1078, 2014.[6] Francois Chollet. Xception：基于深度可分离卷积的深度学习。arXiv预印本arXiv:1610.02357，2016年。[7] Junyoung Chung、Çaglar Gülçehre、Kyunghyun Cho与Yoshua Bengio。门控循环神经网络在序列建模中的实证评估。 CoRR，abs/1412.3555，2014。[8] 克里斯·戴尔、阿迪古纳·昆科罗、米格尔·巴列斯特罗斯和诺亚·A·史密斯。循环神经网络语法。 NAACL会议论文集，2016年。[9] Jonas Gehring、Michael Auli、David Grangier、Denis Yarats和Yann N. Dauphin。卷积序列到序列学习。arXiv预印本arXiv:1705.03122v2，2017年。[10]Alex Graves。 基于递归神经网络的序列生成。arXiv预印本arXiv:1308.0850，2013年。[11] 何开明、张向宇、任绍清、孙健。图像识别的深度残差学习。 收录于IEEE计算机视觉与模式识别会议论文集，第770–778页，2016年。[12] 塞普·霍赫赖特、约舒亚·本吉奥、保罗·弗拉斯科尼与于尔根·施密德胡伯。循环网络中的梯度流：学习长期依赖的困难，2001年。[13] 塞普·霍赫赖特与于尔根·施密德胡伯。长短期记忆网络。 神经计算，9(8):1735–1780，1997年。[14] 黄中强与玛丽·哈珀。跨语言隐含标注的自训练PCFG语法。载于2009年自然语言处理经验方法会议论文集，第832–841页。 ACL, 2009年8月。[15] Rafal J ozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, 和 Yonghui Wu. 探索语言建模的极限。arXiv预印本arXiv:1602.02410, 2016。[16] Łukasz Kaiser 和 Samy Bengio. 主动记忆能否替代注意力机制？ 载于《神经信息处理系统进展》(NIPS), 2016年。[17] Łukasz Kaiser与Ilya Sutskever. 神经GPU学习算法. 载于国际学习表示会议(ICLR), 2016年。[18] 纳尔·卡尔布伦纳、拉斯·埃斯佩霍尔特、卡伦·西蒙扬、亚伦·范登奥德、亚历克斯·格雷夫斯与科雷·卡武库库奥卢。线性时间神经机器翻译。arXiv预印本arXiv:1610.10099v2，2017年。[19] 尹·金、卡尔·丹顿、梁黄与亚历山大·M·拉什。 结构化注意力网络。国际学习表示会议论文集，2017年。[20] 迪德里克·金马与吉米·巴。Adam：一种随机优化方法。ICLR会议论文集，2015年。[21] 奥列克西·库恰耶夫与鲍里斯·金斯堡。LSTM 网络的因子分解技巧。arXiv 预印本 arXiv:1703.10722，2017 年。[22] 林周翰、冯敏伟、西塞罗·诺盖拉·多斯·桑托斯、于莫、向冰、周博文、约书亚·本吉奥。 结构化自注意力句子嵌入。arXiv预印本arXiv:1703.03130，2017年。[23] 明-桑·吕昂、国·V·黎、伊利亚·苏茨克弗、奥里奥尔·维尼亚尔斯、卢卡斯·凯泽。 多任务序列到序列学习。arXiv预印本arXiv:1511.06114，2015年。[24] 明-桑·吕昂、休·范、克里斯托弗·D·曼宁。基于注意机制的神经机器翻译有效方法。arXiv预印本arXiv:1508.04025，2015年。

[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.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. Computational li nguistics, 19(2):313–330, 1993.[26] David McClosky, Eugene Charniak, and Mark Johnson. 高效自训练语法分析。载于北美计算语言学会议人类语言技术大会主会场论文集, 第152–159页。ACL，2006年6月。[27] 安库尔·帕里克、奥斯卡·泰克斯特伦、迪潘詹·达斯与雅各布·乌兹科雷特。可分解注意力模型。载于《自然语言处理经验方法》，2016年。[28] Romain Paulus, Caiming Xiong, and Richard Socher. 基于深度强化学习的抽象化摘要模型. arXiv预印本 arXiv:1705.04304, 2017.[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 学习精确、紧凑且可解释的树注释. 收录于《第21届计算语言学国际会议暨ACL第44届年会论文集》，第433–440页。ACL，2006年7月。[30] Ofir Press与Lior Wolf. 利用输出嵌入改进语言模型. arXiv预印本arXiv:1608.05859, 2016.[31] 里科·森里奇、巴里·哈多、亚历山德拉·伯奇。基于亚词单元的罕见词神经机器翻译。arXiv预印本arXiv:1508.07909，2015年。[32] 诺姆·沙泽尔、阿扎利亚·米尔霍塞尼、克日什托夫·马齐亚尔兹、安迪·戴维斯、郭乐、杰弗里·辛顿、杰夫·迪恩。 超大规模神经网络：稀疏门控专家混合层。arXiv预印本arXiv:1701.06538，2017年。[33] Nitish Srivastava、Geoffrey E Hinton、Alex Krizhevsky、Ilya Sutskever和Ruslan Salakhutdinov。 Dropout：防止神经网络过拟合的简易方法。《机器学习研究期刊》，15(1):1929–1958, 2014.[34] 赛因巴亚尔·苏赫巴塔尔、亚瑟·斯拉姆、杰森·韦斯顿与罗伯·弗格斯。端到端记忆网络。 载于C. Cortes、N. D. Lawrence、D. D. Lee、M. Sugiyama和R. Garnett编著的《神经信息处理系统进展28》，第2440–2448页。Curran Associates公司，2015年。[35] 伊利亚·苏茨克维尔、奥里奥尔·维尼亚尔斯和Quoc VV Le。 基于神经网络的序列到序列学习。《神经信息处理系统进展》，第3104–3112页，2014年。[36] 克里斯蒂安·塞格迪、文森特·范霍克、谢尔盖·约菲、乔纳森·施伦斯与兹比格涅夫·沃伊纳。 重新思考计算机视觉中的Inception架构。CoRR, abs/1512.00567, 2015.[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. 作为外语的语法。载于《神经信息处理系统进展》，2015.[38] 吴永辉、Mike Schuster、陈志锋、Quoc V Le、Mohammad Norouzi、Wolfgang Macherey、Maxim Krikun、曹源、高秦、Klaus Macherey等。谷歌神经机器翻译系统：弥合人类与机器翻译的鸿沟。arXiv预印本arXiv:1609.08144, 2016年。[39] 周杰、曹颖、王旭光、李鹏、徐伟。用于神经机器翻译的带快速前向连接的深度递归模型。CoRR, abs/1606.04199, 2016.[40] 朱慕华、张悦、陈文亮、张敏、朱景波。快速精确的移位-缩减成分解析法。 收录于第51届ACL年会论文集（第1卷：长篇论文），第434–443页。ACL，2013年8月。

Attention Visualizations

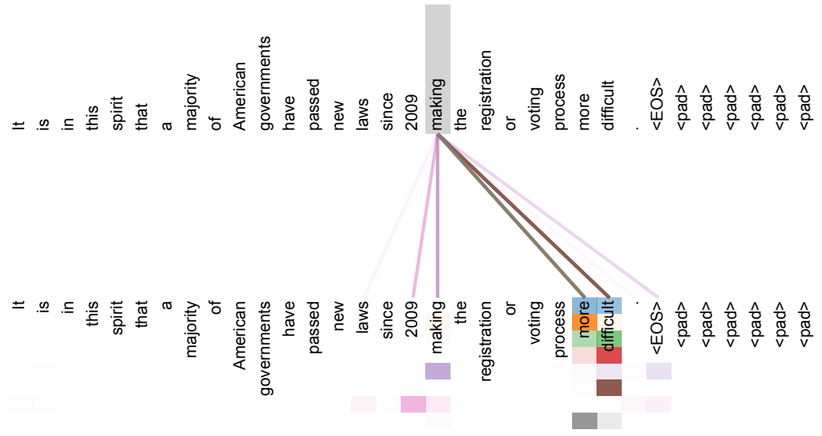

Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color.

注意力可视化

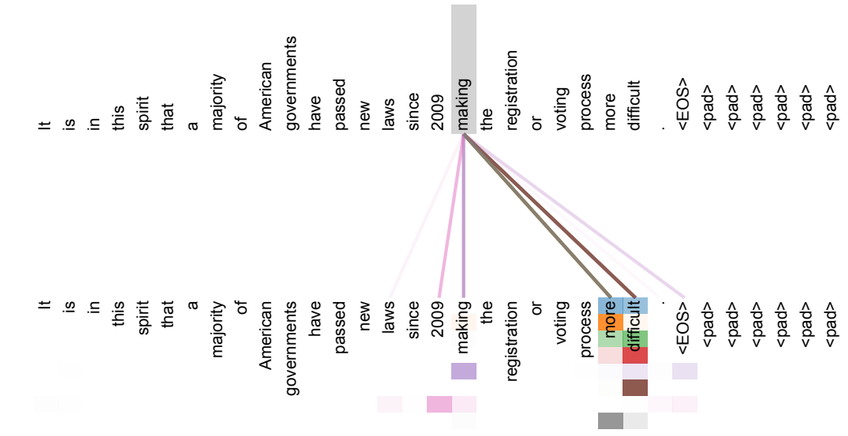

图3：编码器自注意力机制中追踪长距离依赖关系的示例（第5层/共6层）。多个注意力头关注动词"making"的远距离依赖关系，完成"making...more difficult"的短语结构。图中仅展示"making"一词的注意力分布，不同颜色代表不同注意力头。建议彩色显示效果最佳。

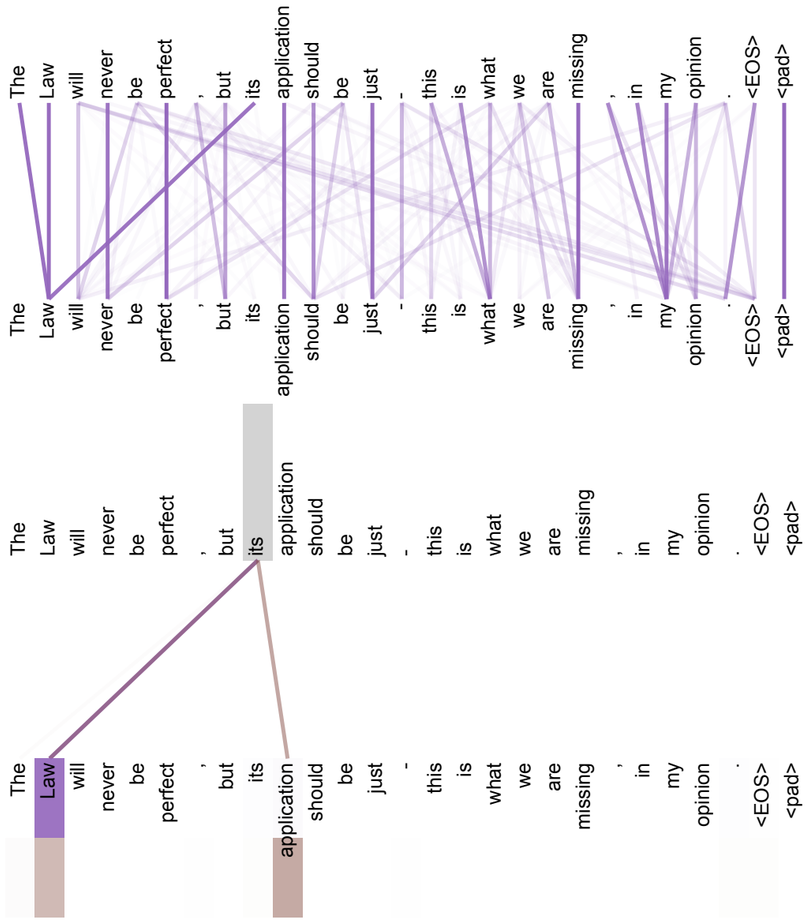

Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

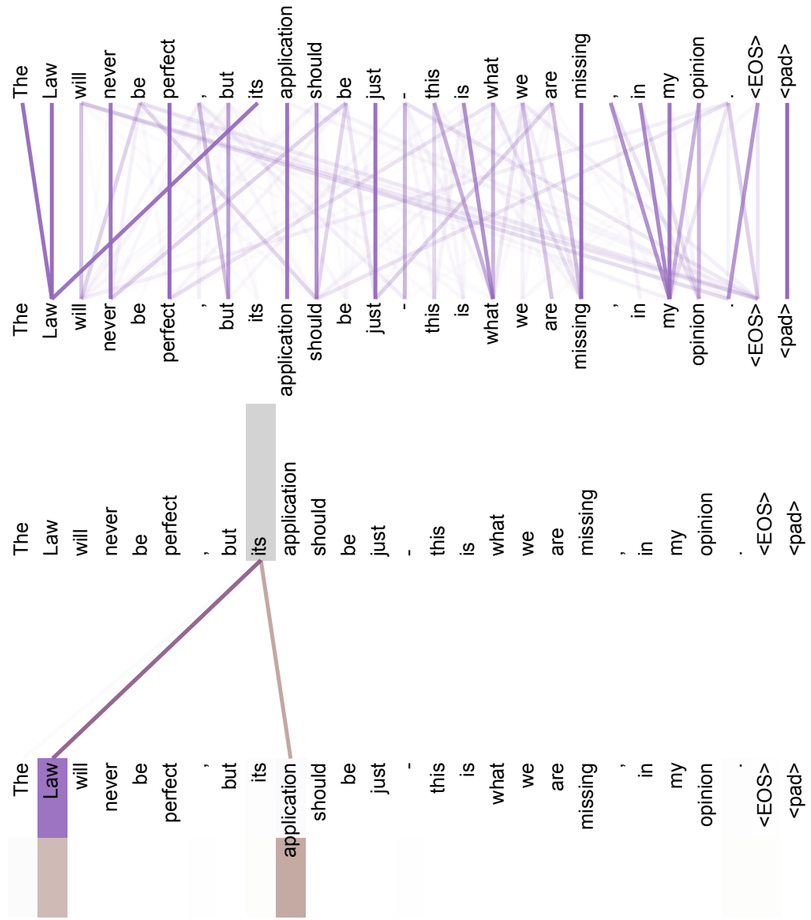

图4：两个注意力头（位于6层中的第5层）显然参与了指代消解。上图：第5层的完整注意力图。下图：仅提取"its"一词在第5、6层注意力头中的局部注意力图。注意该词的注意力分布呈现极度尖锐的特征。

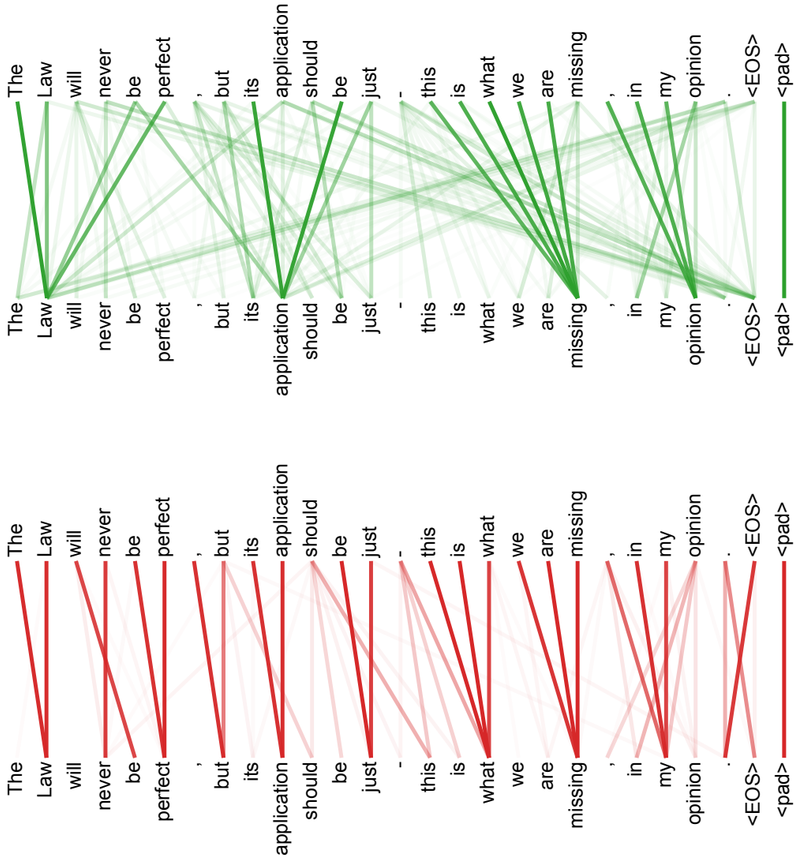

Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

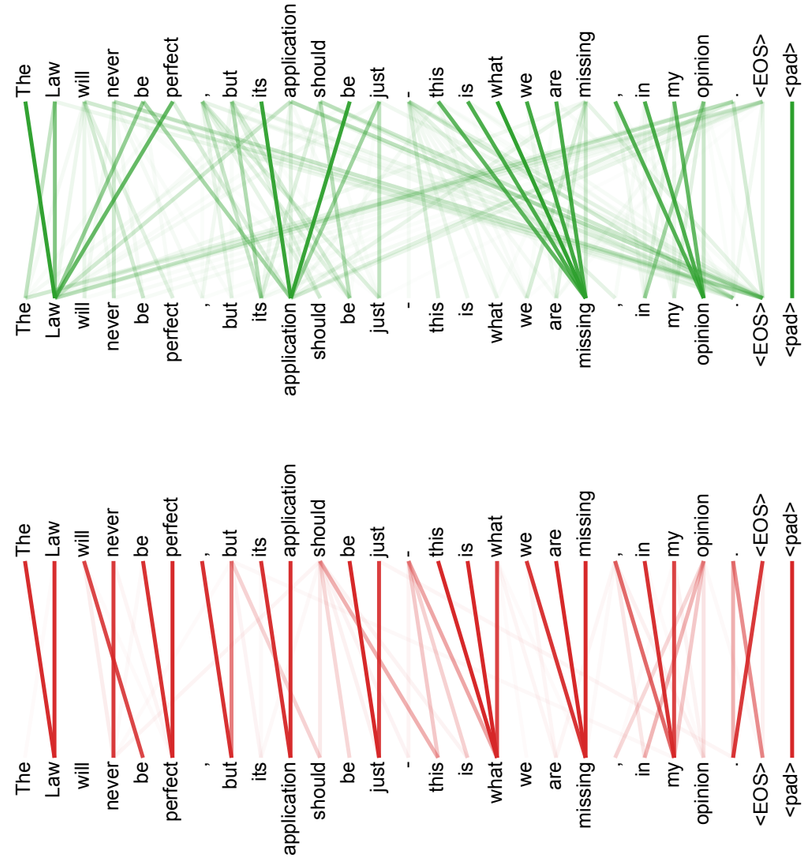

图5：众多注意力头部展现出与句子结构相关的行为特征。上文展示了来自编码器自注意力层（6层中的第5层）两个不同头部的典型案例。这些头部已明确习得了执行不同任务的能力。