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

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.

arXiv:1706.03762v7 [cs.CL] 2 Aug 2023

在注明适当出处的情况下，谷歌特此授权复制本文中的表格和插图，仅限于用于新闻报道或学术作品。

注意力机制足矣

Ashish Vaswani*
谷歌大脑
avaswani@google.com

Noam Shazeer*
Google 大脑
noam@google.com

Niki Parmar*
Google 研究
nikip@google.com

雅各布·乌斯科尔赖特*
谷歌研究 usz@google.com

Llion Jones* 谷歌研究 llion@google.com

Aidan N. Gomez*
多伦多大学
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin*
illia.polosukhin@gmail.com

摘要

主导的序列转导模型基于包含编码器和解码器的复杂循环或卷积神经网络。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，即仅基于注意力机制的 Transformer，完全摒弃了循环和卷积。在两个机器翻译任务上的实验表明，这些模型在质量上更优，同时具有更好的并行性，且训练所需时间显著减少。我们的模型在 WMT 2014 英德翻译任务上实现了 28.4 的 BLEU 值，超越了包括集成方法在内的现有最佳结果，提高了 2 个 BLEU 以上。在 WMT 2014 英法翻译任务上，我们的模型在八块 GPU 上训练 3.5 天后，建立了 41.8 的新单模型最先进 BLEU 分数，这仅是文献中最佳模型训练成本的一小部分。我们展示了 Transformer 对其他任务的泛化能力，成功将其应用于英语成分分析，无论是大数据集还是有限训练数据均取得了良好效果。

*同等贡献。列出顺序为随机。雅各布（Jakob）提出了用自注意力机制替代循环神经网络（RNNs）的构想，并启动了对该想法的评估工作。阿希什（Ashish）与伊利亚（Illia）共同设计并实现了首批Transformer模型，且在该工作的各个方面均发挥了关键作用。诺亚（Noam）提出了缩放点积注意力、多头注意力及无参数位置表示，并参与了几乎每一个细节的工作。尼基（Niki）在我们原始代码库中设计、实现、调优并评估了无数模型变体以及tensor2tensor。利昂（Llion）同样尝试了多种新型模型变体，负责了我们最初的代码库，以及高效的推理和可视化功能。卢卡什（Lukasz）和艾丹（Aidan）度过了无数个漫长的日夜，设计并实现了tensor2tensor，取代了我们早期的代码库，极大地提升了结果性能并显著加速了我们的研究进程。

该工作是在谷歌大脑（Google Brain）期间完成的
该工作完成于谷歌研究院（Google Research）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

arXiv:1706.03762v7 [cs.CL] 2 Aug 2023

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]，已在序列建模和转换问题（如语言建模和机器翻译 [35, 2, 5]）中确立为最先进的方法。此后，众多研究持续推动循环语言模型和编码器-解码器架构的发展 [38, 24, 15]。

循环模型通常沿着输入和输出序列的符号位置对计算进行分解。通过将位置与计算时间步骤对齐，它们生成一系列隐藏状态 h_t，该状态是前一隐藏状态 h_{t-1} 和位置 t 处输入的函数。这种固有的顺序性排除了训练样本内部的并行化，而在序列长度较长时，由于内存限制限制了跨样本的批处理，这一点变得尤为关键。近期研究通过因式分解技巧 [21] 和条件计算 [32] 在计算效率方面取得了显著改进，后者同时也提升了模型性能。然而，顺序计算的基本约束依然存在。

注意机制已成为各种任务中 compelling 序列建模和转换模型不可或缺的一部分，使得在建模依赖关系时不必考虑其在输入或输出序列中的距离 [2, 19]。然而，除少数情况 [27] 外，此类注意机制通常与循环网络结合使用。

在本文中，我们提出了 Transformer 模型，该架构摒弃了循环结构，完全依赖注意力机制来建立输入与输出之间的全局依赖关系。Transformer 实现了显著更高的并行化程度，且在八个 P100 GPU 上仅训练十二小时后，即可在翻译质量方面达到新的先进水平。

2 背景

减少顺序计算的目标也是扩展神经GPU [16]、ByteNet [18] 和 ConvS2S [9] 的基础，这些模型均使用卷积神经网络作为基本构建模块，并行计算所有输入和输出位置的隐藏表示。在这些模型中，关联任意两个输入或输出位置信号所需的运算量随位置间距离的增加而增加，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加困难 [12]。在 Transformer 中，这被减少为常数数量的运算，尽管由于平均注意力加权位置导致有效分辨率降低，我们如第 3.2 节所述，通过多头注意力机制来抵消这种影响。

自注意力（self-attention），有时也称为片内注意力（intra-attention），是一种关注机制，用于关联单个序列中的不同位置，以计算该序列的表示。自注意力已成功应用于多种任务，包括阅读理解、抽取式摘要、文本蕴含以及学习与任务无关的句子表示 [4, 27, 28, 22]。

端到端记忆网络基于循环注意机制，而非序列对齐的循环结构，研究表明其在简单语言问答和语言建模任务中表现良好 [34]。

然而，据我们所知，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导模型，而未使用序列对齐的循环神经网络或卷积。在随后的章节中，我们将描述 Transformer，阐述自注意力的动机，并讨论其相对于 [17, 18] 和 [9] 等模型的优势。

3 模型架构

大多数具有竞争力的神经序列转导模型都采用编码器-解码器结构 [5, 2, 35]。在此，编码器将符号表示的输入序列 $(x_1, \ldots, x_n)$ 映射为连续表示序列 $\mathbf{z} = (z_1, \ldots, z_n)$。给定 $\mathbf{z}$ 后，解码器随后生成符号输出序列 $(y_1, \ldots, y_m)$，每次生成一个元素。在每一步中，该模型均为自回归式 [10]，即在生成下一个符号时利用先前已生成的符号作为额外输入。

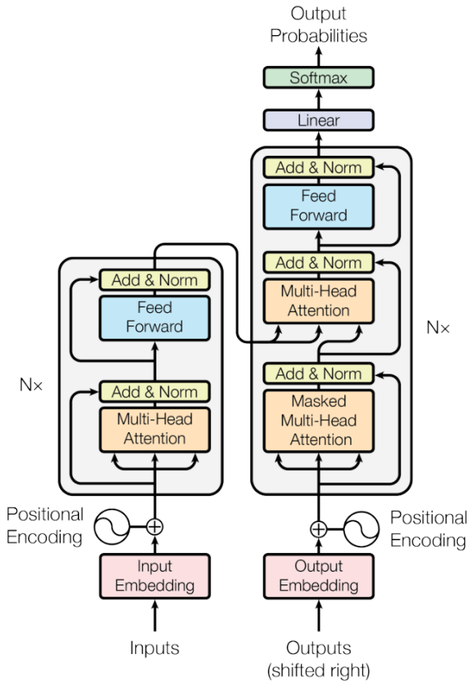


Figure 1: The Transformer - model architecture.

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

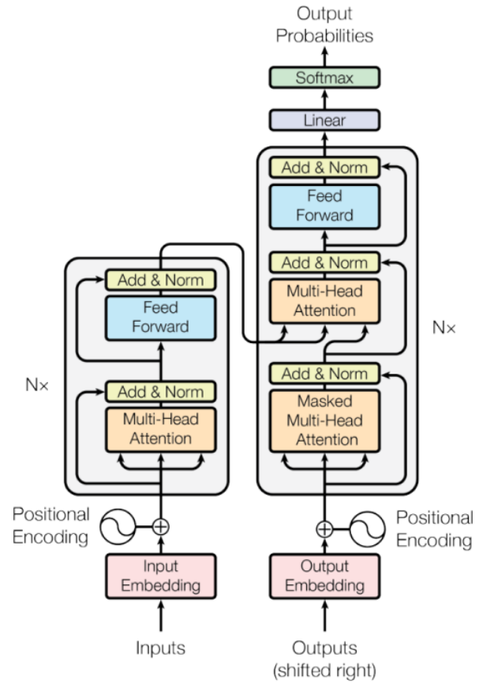


图 1：Transformer 模型架构。

Transformer 遵循这一整体架构，在编码器和解码器中均采用了堆叠的自注意力机制和逐点全连接层，如图 1 的左半部分和右半部分所示。

3.1 编码器与解码器堆叠

编码器：该编码器由 $N=6$ 个相同的层堆叠而成。每一层包含两个子层。第一个子层是多头自注意力机制，第二个子层是简单的位置前馈全连接网络。我们在每个子层周围采用残差连接 [11]，并随后进行层归一化 [1]。即，每个子层的输出为 LayerNorm(x + Sublayer(x))，其中 Sublayer(x) 是子层本身实现的功能。为了便于这些残差连接，模型中的所有子层以及嵌入层均产生维度为 $d_{\text{model}}=512$ 的输出。

解码器：解码器同样由 $N=6$ 个相同的层堆叠而成。除了编码器中每个层的两个子层外，解码器还插入了第三个子层，该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，并随后进行层归一化。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。这种掩码处理，结合输出嵌入偏移一个位置的事实，确保了位置 i 的预测仅依赖于位置小于 i 处的已知输出。

3.2 注意力

注意力函数可描述为将查询和一组键值对映射到输出的过程，其中查询、键、值和输出均为向量。该输出计算为加权和

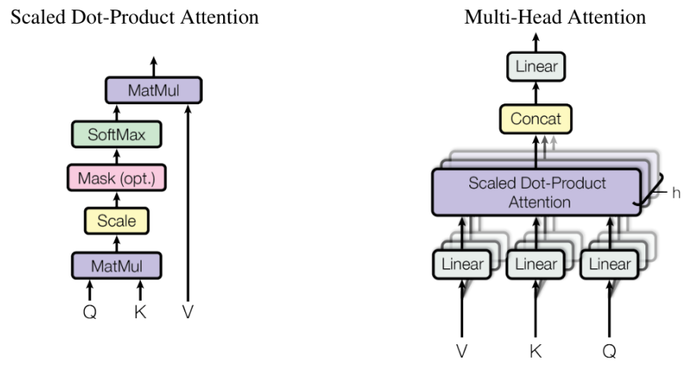


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

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.

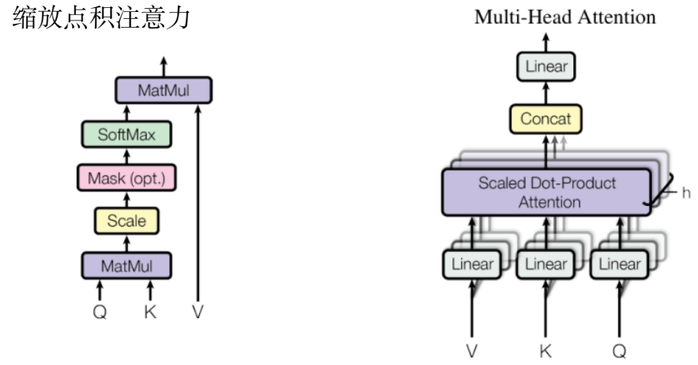


图 2：（左）缩放点积注意力。（右）多头注意力由多个并行运行的注意力层组成。

这些值的权重由查询与对应键之间的兼容性函数计算得出。

3.2.1 缩放点积注意力

我们将这种特定的注意力机制称为“缩放点积注意力”（图 2）。输入由维度为 d_k 的查询（queries）和键（keys），以及维度为 d_v 的值（values）组成。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 函数以获得值上的权重。

在实践中，我们同时在一组查询上计算注意力函数，并将它们打包成一个矩阵 Q。键和值也被打包成矩阵 K 和 V。我们计算输出矩阵如下：

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

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。除缩放因子 $\frac{1}{\sqrt{d_k}}$ 外，点积注意力与我们的算法相同。加法注意力使用具有单个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更节省空间，因为它可以使用高度优化的矩阵乘法代码来实现。

尽管当 d_k 取值较小时两种机制表现相似，但当 d_k 较大时，加性注意力（additive attention）的表现优于未进行缩放的点积注意力 [3]。我们推测，当 d_k 取值较大时，点积的数值会显著增大，导致 softmax 函数进入其梯度极小的区域 4。为抵消这种影响，我们将点积结果缩放 $\frac{1}{\sqrt{d_k}}$。

3.2.2 多头注意力

与使用 d_{model} 维度的键、值和查询执行单次注意力函数不同，我们发现将查询、键和值分别通过不同的、已学习的线性投影线性投影 h 次至 d_k、d_k 和 d_v 维度是有益的。然后在这些投影后的查询、键和值的每一个版本上并行执行注意力函数，从而产生 d_v 维度的

[4]为了说明点积为何会变大，假设dq 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 $q \cdot k = \sum_{i=1} q_i k_i$ 的均值为 0，方差为 d_k。

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。这些值被拼接并再次进行投影，从而得到最终值，如图2所示。

多头注意力机制使模型能够在不同位置联合关注来自不同表示子空间的信息。若仅使用单个注意力头，平均化操作会抑制这一能力。

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

其中投影矩阵为 $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$，$W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$，$W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ 以及 $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$。

在本研究中，我们采用 $h=8$ 个并行注意力层（或头）。对于每个头，我们使用 $d_k = d_v = d_\text{model}/h = 64$。由于每个头的维度降低，总计算成本与具有完整维度的单头注意力相当。

3.2.3 注意力机制在我们模型中的应用

Transformer 以三种不同的方式使用多头注意力机制：

在“编码器-解码器注意力”层中，查询（queries）来自前一个解码器层，而记忆键（memory keys）和值（values）则来自编码器的输出。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。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，如文献 [38, 2, 9] 所述。

编码器包含自注意力层。在自注意力层中，所有的键（keys）、值（values）和查询（queries）均源自同一位置，在本例中即为编码器中前一层的输出。编码器中的每个位置均可关注编码器前一层中的所有位置。

同样地，解码器中的自注意力层允许解码器中的每个位置关注到该位置及之前（含该位置）的所有位置。为了保持自回归特性，我们需要阻止解码器中的向左信息流。我们在缩放点积注意力内部通过屏蔽（设置为 $-\infty$）输入中对应于非法连接的所有 softmax 值来实现这一点。参见图 2。

3.3 逐位前馈网络

除了注意力子层之外，编码器与解码器中的每一层还包含一个全连接前馈网络，该网络对每个位置分别且以相同的方式应用。此网络由两个线性变换组成，中间夹有一个 ReLU 激活函数。

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

尽管线性变换在不同位置是相同的，但各层使用的参数不同。另一种描述方式是将其视为两次卷积核大小为 1 的卷积。输入和输出的维度为 $d_\text{model}=512$，而内部层的维度为 $d_{ff}=2048$。

3.4 嵌入与 Softmax

与其他序列转导模型类似，我们使用学习到的嵌入层将输入 token 和输出 token 转换为维度为 d_model 的向量。我们还使用常规的学习线性变换和 softmax 函数将解码器输出转换为预测的下一个 token 概率。在我们的模型中，两个嵌入层与预 softmax 线性变换之间共享相同的权重矩阵，这与文献 [30] 的做法相似。在嵌入层中，我们将这些权重乘以 $\sqrt{d_\text{model}}$。

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、逐层复杂度和顺序操作的最小数量。n 为序列长度，d 为表示维度，k 为卷积的核大小，r 为受限自注意力中的邻域大小。

层类型	每层复杂度	序列式运算量	最大路径长度
自注意力机制	O(n · d)	O(1)	O(1)
循环式	O(n · d)	O()	O(n)
卷积式	O(k · n · d)	O(1)	O(log (n))
自注意力机制（受限）	O(r · n · d)	O(1)	O(n/r)

3.5 位置编码

由于我们的模型不包含循环结构和卷积结构，为了使模型能够利用序列的顺序信息，我们必须注入关于序列中令牌相对或绝对位置的信息。为此，我们在编码器堆栈和解码器堆栈底部的输入嵌入处添加了“位置编码”。位置编码的维度与嵌入的维度 d_{model} 相同，以便两者可以进行相加。位置编码有许多选择，可以是学习得到的，也可以是固定的 [9]。

在本工作中，我们使用不同频率的正弦和余弦函数：

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

其中 pos 表示位置，i 表示维度。也就是说，位置编码的每个维度对应一个正弦波。波长从 2π 到 $10000 \cdot 2\pi$ 呈几何级数排列。我们选择这个函数是因为我们假设它能使模型容易地学习关注相对位置，因为对于任何固定的偏移量 k，PE_{pos+k} 都可以表示为 PE_{pos} 的线性函数。

我们还尝试使用了学习得到的位置嵌入 [9] 进行实验，发现两种版本产生的结果几乎相同（见表3第(E)行）。

我们选择了正弦形式，因为它可能使模型能够外推至比训练过程中遇到的更长的序列长度。

4 为什么使用自注意力机制

在本节中，我们将自注意力层与通常用于将符号表示的变长序列 $(x_1, \ldots, x_n)$ 映射到等长序列 $(z_1, \ldots, z_n)$ 的循环和卷积层进行了比较，其中 $x_i, z_i \in \mathbb{R}^d$，例如典型序列转导编码器或解码器中的隐藏层。为了说明我们使用自注意力的动机，我们考虑了三个需求。

一是每层的总计算复杂度。二是可并行化的计算量，其度量标准为所需的最少顺序操作数。

第三种是网络中长程依赖之间的路径长度。学习长程依赖是许多序列转导任务中的关键挑战。影响学习此类依赖能力的一个关键因素是前向和反向信号在网络中必须穿过的路径长度。输入和输出序列中任意位置组合之间的这些路径越短，学习长程依赖就越容易 [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。

如表1所示，自注意力层通过固定数量的顺序执行操作连接所有位置，而循环层需要 $O(n)$ 次顺序操作。就计算复杂度而言，当序列

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，在使用了机器翻译领域最先进模型的句子表示中，这通常是一种常见情况，例如子词 [38] 和字节对 [31] 表示。为了提高涉及极长序列任务的计算性能，可以将自注意力机制限制为仅考虑以相应输出位置为中心、大小为 r 的输入序列邻域。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。

卷积核宽度 $k<n$ 的单个卷积层无法连接所有的输入和输出位置对。要实现这一点，在连续卷积核的情况下需要堆叠 $O(n/k)$ 个卷积层，而在空洞卷积 [18] 的情况下则需要 $O(\log_k(n))$ 个，这增加了网络中任意两个位置之间最长路径的长度。通常情况下，卷积层的计算代价是循环层的 k 倍。然而，可分离卷积 [6] 能显著降低复杂度至 $O(k\cdot n\cdot d+n\cdot d^2)$。即便如此，当 $k=n$ 时，可分离卷积的复杂度仍等同于自注意力层与逐点前馈层的组合，这正是我们模型所采用的方法。

作为附加收益，自注意力机制能够生成更具可解释性的模型。我们检查了模型中的注意力分布，并在附录中展示和讨论了相关示例。不仅各个注意力头明显学会了执行不同的任务，而且许多注意力头似乎还表现出与句子的句法和语义结构相关的行为。

5 训练

本节介绍了我们模型的训练方案。

5.1 训练数据与批次处理

我们在标准的 WMT 2014 英德数据集上进行训练，该数据集包含约 450 万句对。句子采用字节对编码（byte-pair encoding）[3] 进行编码，其共享源语言-目标语言词汇表约为 37000 个词元。对于英法翻译任务，我们使用了规模更大的 WMT 2014 英法数据集，包含 3600 万句，并将词元切分为 32000 个 word-piece 词汇表 [38]。句子对按近似序列长度进行分组批量。每个训练批次包含一组句子对，其中分别包含约 25000 个源语言词元和 25000 个目标语言词元。

5.2 硬件与调度

我们在配备 8 块 NVIDIA P100 GPU 的单机上训练模型。对于使用文中所述超参数的基础模型，每个训练步骤耗时约 0.4 秒。基础模型总共训练了 100,000 步，历时 12 小时。对于大型模型（见表 3 最后一行描述），单步耗时为 1.0 秒。大型模型总共训练了 300,000 步（3.5 天）。

5.3 优化器

我们使用了 Adam 优化器 [20]，其中 $\beta_1=0.9$，$\beta_2=0.98$，$\varepsilon=10^{-9}$。在训练过程中，我们根据以下公式调整学习率：

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5}) \quad (3)$$

这对应于在前 warmup_steps 训练步数中线性增加学习率，随后按步数倒平方根的比例减小学习率。我们采用 warmup_steps = 4000。

5.4 正则化

我们在训练过程中采用三种类型的正则化：

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	$\mathbf{3.3 \cdot 10^{18}}$	
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：在英语到德语和英语到法语的 newstest2014 测试中，Transformer 以远低于以往最先进模型的训练成本，实现了更优的 BLEU 分数。

模型	BLEU		训练成本（FLOPs）	
	英–德	英–法	英–德	英–法
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 集成 [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL 集成 [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S 集成 [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer（基础模型）	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer（大模型）	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

残差丢弃（Residual Dropout） 我们对每个子层的输出应用丢弃法 [33]，在将其与子层输入相加并进行归一化之前执行。此外，我们在编码器堆栈和解码器堆栈中均对嵌入（embeddings）与位置编码（positional encodings）之和应用丢弃法。对于基础模型，我们采用的丢弃率为 $P_{drop} = 0.1$。

标签平滑 在训练过程中，我们采用了值为 $\varepsilon_{ls} = 0.1$ 的标签平滑策略 [36]。这会增加困惑度（perplexity），因为模型学会了更加不确定，但同时也提高了准确率和 BLEU 分数。

6 结果

6.1 机器翻译

在 WMT 2014 英语到德语翻译任务中，大型 Transformer 模型（表 2 中的 Transformer (big)）的表现超越了此前报道的所有最佳模型（包括集成模型），BLEU 值高出 2.0 以上，确立了 28.4 的新最先进 BLEU 分数。该模型的配置列于表 3 的最后一行。在 8 块 P100 GPU 上训练耗时 3.5 天。即便是我们的基础模型，其性能也超越了所有已发表的模型和集成模型，且训练成本仅为其他竞争模型的零头。

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，超越了此前所有已发表的单一模型，且训练成本仅为前代最先进模型的不到四分之一。用于英法翻译的 Transformer（大）模型采用了 $P_{drop} = 0.1$ 的 dropout 率，而非 0.3。

对于基础模型，我们使用了通过平均最后 5 个检查点（每 10 分钟保存一次）获得的单个模型。对于大型模型，我们平均了最后 20 个检查点。我们使用束搜索，束宽为 4，长度惩罚 $\alpha = 0.6$ [38]。这些超参数是在开发集上实验后选定的。我们将推理时的最大输出长度设置为输入长度加 50，但在可能时提前终止 [38]。

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持续单精度浮点能力的估计值相乘，来估算训练模型所使用的浮点运算次数 5。

6.2 模型变体

为了评估 Transformer 不同组件的重要性，我们以不同方式调整了基础模型，并测量了在英语到德语翻译任务上性能的变化

[5]我们分别采用 2.8、3.7、6.0 和 9.5 TFLOPS 作为 K80、K40、M40 和 P100 的数值。

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。所列困惑度为每个wordpiece的值，依据我们的字节对编码（byte-pair encoding）计算，不应与每个单词的困惑度进行比较。

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	位置嵌入而非正弦函数									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

开发集，newstest2013。我们使用了上一节所述的束搜索（beam search），但未采用检查点平均（checkpoint averaging）。我们在表 3 中展示了这些结果。

在表3行(A)中，我们调整了注意力头的数量以及注意力键和值的维度，同时保持计算量恒定，如第3.2.2节所述。虽然单头注意力的BLEU值比最佳设置低0.9，但头数过多也会导致质量下降。

在表3的第(B)行中，我们观察到减小注意力键尺寸 d_k 会降低模型质量。这表明确定兼容性并不容易，且使用比点积更复杂的兼容性函数可能有益。我们在第(C)和(D)行中进一步观察到，正如预期那样，更大的模型效果更好，而Dropout在避免过拟合方面非常有效。在第(E)行中，我们用可学习的嵌入位置编码 [9] 替代了我们的正弦位置编码，并观察到其结果与基础模型几乎一致。

6.3 英语成分分析

为了评估Transformer能否推广到其他任务，我们在英语成分分析上进行了实验。该任务具有特定的挑战：输出受到严格的结构约束，且显著长于输入。此外，RNN序列到序列模型在小数据条件下尚未能达到最先进结果[37]。

我们在宾夕法尼亚树库（Penn Treebank）[25] 的《华尔街日报》（WSJ）部分上训练了一个模型维度 d_{model}=1024 的 4 层 Transformer，包含约 40,000 个训练句子。我们还使用了包含约 1700 万句子的更大规模的高置信度语料和 BerkleyParser 语料，在半监督设置下对其进行了训练 [37]。对于仅使用 WSJ 的设置，我们采用了 16K 词表的词汇量；对于半监督设置，则采用了 32K 词表的词汇量。

我们仅进行少量实验，在第22节开发集上筛选了丢弃率、注意力机制与残差连接（第5.4节）、学习率及束搜索宽度；其余所有参数均保持与英译德基础翻译模型一致。在推理过程中，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在英语成分分析中表现出良好的泛化能力（结果来自WSJ第23部分）

解析器	训练	WSJ 23 F1值
Vinyals & Kaiser 等 (2014) [37]	仅使用 WSJ，判别式	88.3
Petrov 等 (2006) [29]	仅使用 WSJ，判别式	90.4
Zhu 等 (2013) [40]	仅使用 WSJ，判别式	90.4
Dyer 等 (2016) [8]	仅使用 WSJ，判别式	91.7
Transformer（4层）	仅使用 WSJ，判别式	91.3
Zhu 等 (2013) [40]	半监督	91.3
Huang & Harper (2009) [14]	半监督	91.3
McClosky 等 (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 (2015) [23]	多任务	93.0
Dyer 等 (2016) [8]	生成式	93.3

将最大输出长度增加至输入长度加 300。对于仅使用《华尔街日报》（WSJ）和半监督设置，我们均采用 21 的束搜索宽度及 α=0.3 的参数。

表4中的结果表明，尽管缺乏针对特定任务的微调，我们的模型仍表现出惊人的良好性能，除循环神经网络语法 [8] 外，其结果优于所有先前报告的模型。

与 RNN 序列到序列模型 [37] 不同，Transformer 甚至在仅使用包含 40,000 个句子的 WSJ 训练集进行训练时，其性能也优于 Berkeley-Parser [29]。

7 结论

在本文中，我们提出了Transformer，这是一个完全基于注意力的序列转导模型，用多头自注意力机制替换了编码器-解码器架构中最常用的循环层。

在翻译任务中，Transformer 的训练速度显著快于基于循环或卷积层的架构。在 WMT 2014 英德翻译和 WMT 2014 英法翻译任务上，我们均取得了新的最先进水平。在前一项任务中，我们的最佳模型甚至优于此前报道的所有集成模型。

我们对基于注意力模型的未来充满期待，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，以高效处理图像、音频和视频等大规模输入和输出。使生成过程减少序列依赖性也是我们另一项研究目标。

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

致谢 我们感谢 Nal Kalchbrenner 和 Stephan Gouws 提出的富有成效的评论、修正及启发。

参考文献

[1] Jimmy Lei Ba, Jamie Ryan Kiros, 和 Geoffrey E Hinton。层归一化。arXiv预印本 arXiv:1607.06450, 2016年。

[2] Dzmitry Bahdanau, Kyunghyun Cho, 和 Yoshua Bengio。通过联合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, 和 Quoc V. Le。神经机器翻译架构的大规模探索。CoRR, abs/1703.03906, 2017。

[4] 程建鹏, 董黎, 和 Mirella Lapata. 用于机器阅读的长短期记忆网络. arXiv预印本 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.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk 和 Yoshua Bengio。使用循环神经网络编码器-解码器进行统计机器翻译的短语表示学习。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, Çağlar Gülçehre, Kyunghyun Cho, 和 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] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros 和 Noah A. Smith。循环神经网络语法。载于 Proc. of NAACL，2016年。

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, 和 Yann N. Dauphin。卷积式序列到序列学习。arXiv 预印本 arXiv:1705.03122v2, 2017。

[10] Alex Graves。使用循环神经网络生成序列。arXiv 预印本集, arXiv:1308.0850, 2013.

[11] 何恺明, 张翔宇, 任少卿, 和 Jian Sun。用于图像识别的深度残差学习。见：IEEE 计算机视觉与模式识别会议论文集, 第770–778页, 2016年。

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, 和 Jürgen Schmidhuber. 循环网络中的梯度流：学习长期依赖性的困难, 2001.

[13] Sepp Hochreiter 和 Jürgen Schmidhuber。长短期记忆。Neural computation, 9(8):1735–1780, 1997。

[14] 黄中强和 Mary Harper。具有跨语言潜在标注的自训练 PCFG 语法。见：自然语言处理实证方法会议 2009 年论文集，第 832–841 页。ACL，2009 年 8 月。

[15] Rafal Jozefowicz, 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？见《神经信息处理系统进展》（NIPS），2016年。

[17] Łukasz Kaiser 和 Ilya Sutskever。神经 GPU 学习算法。载于 国际学习表征会议 (ICLR)，2016年。

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, 和 Koray Kavukcuoglu. 线性时间下的神经机器翻译. arXiv预印本 arXiv:1610.10099v2, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang 和 Alexander M. Rush。结构化注意力网络。载于《国际学习表征会议》，2017年。

[20] Diederik Kingma 和 Jimmy Ba。Adam：一种随机优化方法。载于 ICLR，2015年。

[21] Oleksii Kuchaiev 和 Boris Ginsburg。LSTM 网络的因子分解技巧。arXiv 预印本 arXiv:1703.10722, 2017年。

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou 和 Yoshua Bengio。结构化自注意句子嵌入。arXiv预印本，arXiv:1703.03130，2017年。

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, 和 Lukasz Kaiser. 多任务序列到序列学习. arXiv 预印本 arXiv:1511.06114, 2015.

[24] Minh-Thang Luong, Hieu Pham, 和 Christopher D Manning。基于注意力的神经机器翻译的有效方法。arXiv预印本 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.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz 和 Beatrice Santorini. 构建大型英语标注语料库：宾夕法尼亚树库。计算语言学, 19(2):313 – 330, 1993.

[26] David McClosky, Eugene Charniak 和 Mark Johnson. 用于句法分析的有效自训练方法. 见: NAACL 人类语言技术会议论文集, 主会议卷, 第152 – 159页. ACL, 2006年6月.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das 和 Jakob Uszkoreit。可分解注意力模型。载于《自然语言处理的实证方法》，2016年。

[28] Romain Paulus, Caiming Xiong 和 Richard Socher. 用于摘要生成的深度强化模型. arXiv预印本 arXiv:1705.04304, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux 和 Dan Klein。学习精确、紧凑且可解释的树标注。见：第21届计算语言学国际会议暨第44届ACL年会论文集，第433 – 440页。ACL，2006年7月。

[30] Ofir Press 和 Lior Wolf。利用输出嵌入改进语言模型。arXiv 预印本 arXiv:1608.05859, 2016。

[31] Rico Sennrich, Barry Haddow 和 Alexandra Birch. 使用子词单元进行罕见词的神经机器翻译. arXiv预印本 arXiv:1508.07909, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton 和 Jeff Dean。极其庞大的神经网络：稀疏门控专家混合层。arXiv预印本 arXiv:1701.06538, 2017。

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever 和 Ruslan Salakhutdinov。Dropout：一种防止神经网络过拟合的简单方法。机器学习研究杂志, 15(1):1929 – 1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston 和 Rob Fergus。端到端记忆网络。见 C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama 和 R. Garnett 主编，《神经信息处理系统进展》第28卷，第2440 – 2448页。Curran Associates, Inc.，2015年。

[35] Ilya Sutskever, Oriol Vinyals, 和 Quoc VV Le. 基于神经网络的序列到序列学习. 见 Neural Information Processing Systems 进展, 第3104 – 3112页, 2014年.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens 和 Zbigniew Wojna。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 和 Hinton。作为外语的语法。见：神经信息处理系统进展，2015年。

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, 等. 谷歌神经机器翻译系统：弥合人工翻译与机器翻译之间的差距。arXiv预印本 arXiv:1609.08144, 2016.

[39] 周杰, 曹颖, 王旭光, 李鹏, 徐伟. 用于神经机器翻译的具有前馈连接深度的循环模型. CoRR, abs/1606.04199, 2016.

[40] 朱慕华, 张跃, 陈文亮, 张敏, 和朱景波. 快速且精确的移进-归约成分句法分析. 见《ACL 2013年第51届年会论文集（第一卷：长论文）》，第434 – 443页. ACL, 2013年8月.

Attention Visualizations

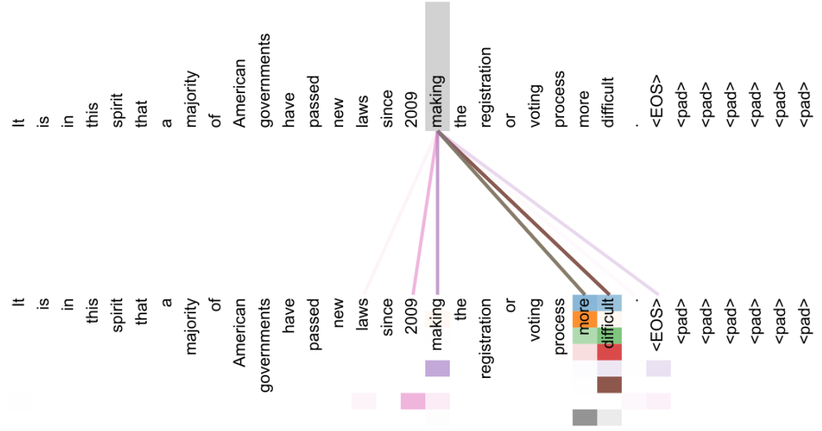


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word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

注意力可视化

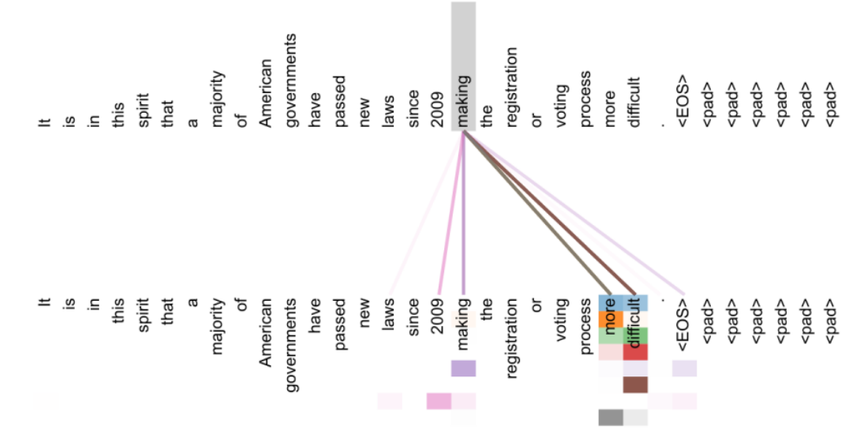


图 3：编码器自注意力机制在 6 层中的第 5 层捕捉长距离依赖关系的一个示例。许多注意力头关注动词‘making’的远距离依赖，以补全短语‘making...more difficult’。此处仅展示与单词‘making’相关的注意力。不同颜色代表不同的注意力头。建议以彩色模式查看。

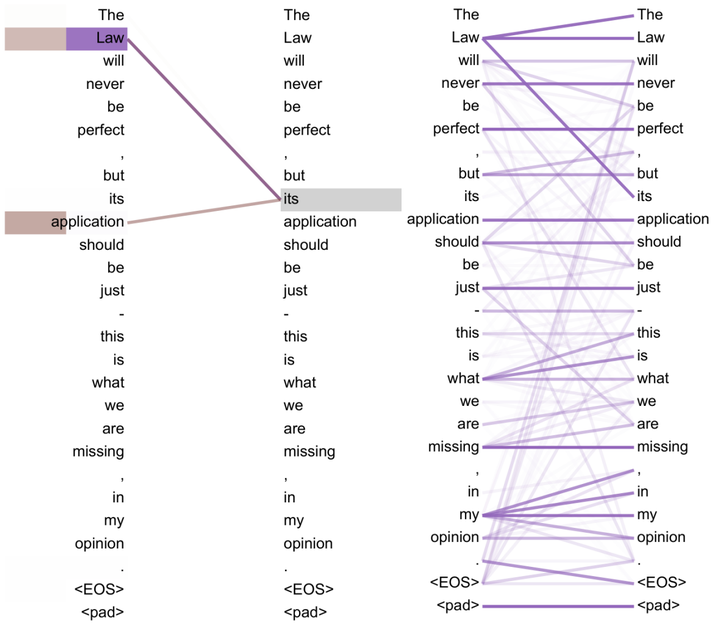
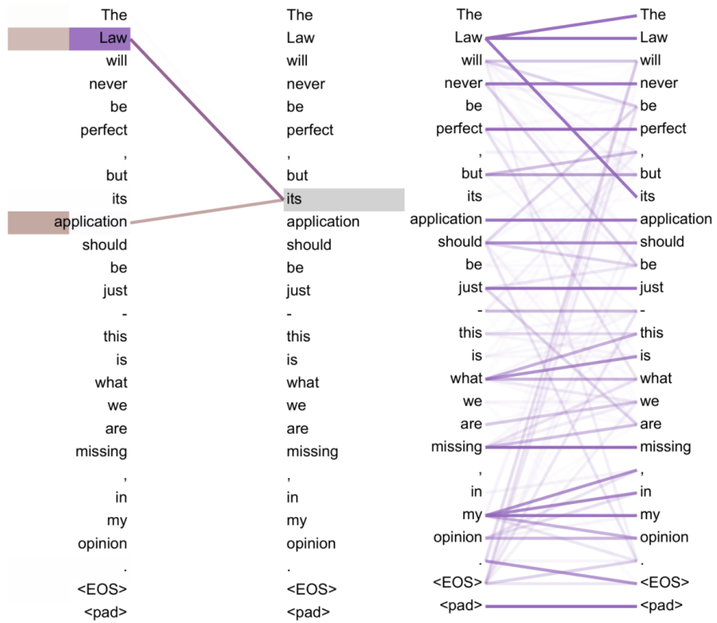


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

图 4：两个注意力头，同样参与了第 5 层到第 6 层的指代消解。顶部：针对第 5 个注意力头的完整注意力分布。底部：仅针对单词“its”从第 5 个和第 6 个注意力头提取的孤立注意力分布。注意该词的注意力分布非常集中（锐利）。



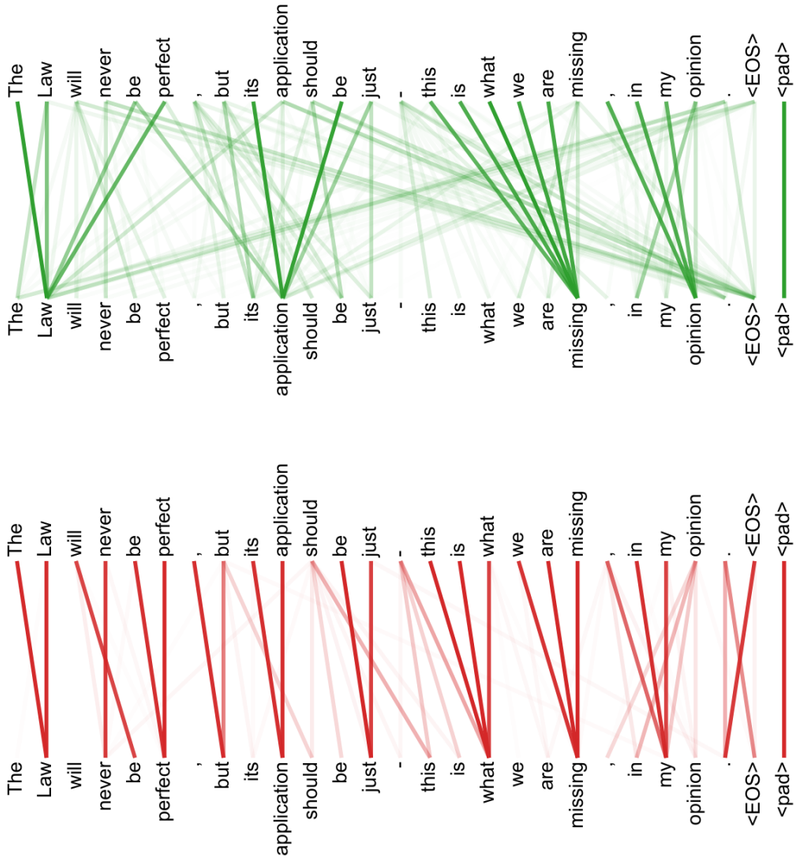


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

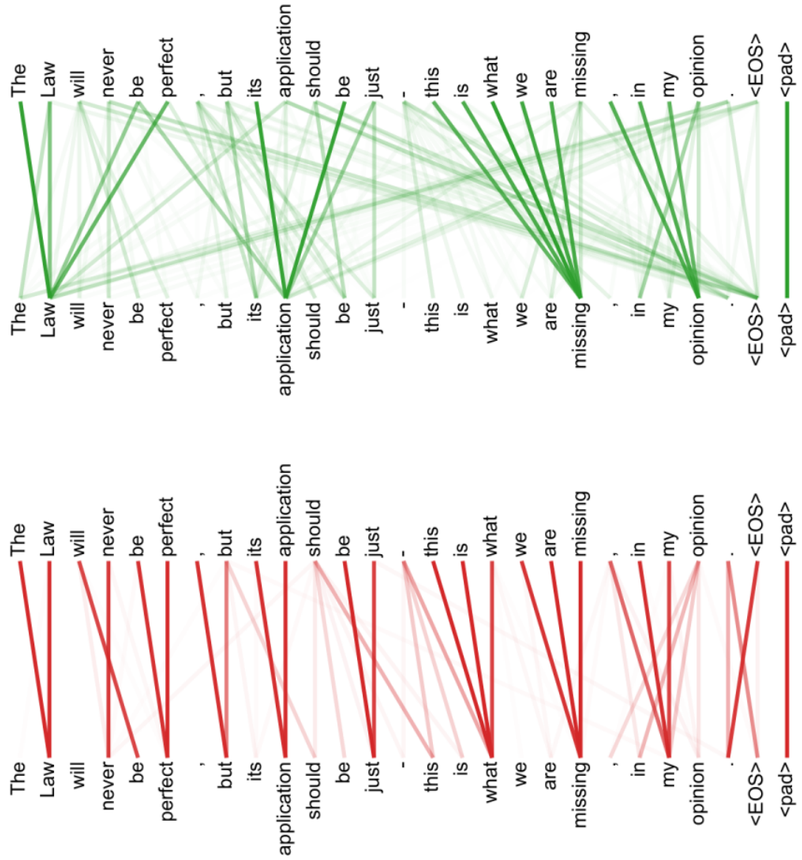


图 5：许多注意力头表现出与句子结构相关的行为。我们在上方给出了两个这样的例子，分别来自 6 层编码器自注意力机制中的第 5 层的两个不同注意力头。这些注意力头显然学会了执行不同的任务。