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

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

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.

在提供适当署名的情况下，谷歌特此授权复制本文中的表格和图表，仅限于用于新闻或学术作品。

Attention Is All You Need

Ashish Vaswani* Google Brain GoogleBrain avaswani@google.com
Noam Shazeer* noam@google.com
Niki Parmar* Google Research Google Research nikip@google.com
Jakob Uszkoreit* usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* †多伦多大学 aidan@cs.toronto.edu
Łukasz Kaiser*Google Brain lukaszkaiser@google.com
伊利亚·波洛苏金* ‡ illia.polosukhin@gmail.com

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

摘要

目前主流的序列转导模型基于复杂的循环或卷积神经网络，包含编码器和解码器。表现最佳的模型还通过注意力机制连接编码器和解码器。我们提出了一种新的简单网络架构——Transformer，完全基于注意力机制，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，且更易于并行化，训练时间也显著减少。我们的模型在WMT 2014英语-德语翻译任务上取得了28.4的BLEU分数，比现有的最佳结果（包括集成模型）提高了超过2个BLEU。在WMT 2014英语-法语翻译任务上，我们的模型在八块GPU上训练3.5天后，达到了41.8的单模型SOTA BLEU分数，其训练成本仅为文献中最优模型的极小部分。我们通过将Transformer成功应用于英语依存句法分析任务（无论训练数据量大或量小），证明了其具有良好的泛化能力。

*平等贡献。列表顺序是随机的。Jakob 提议用自注意力机制替换 RNN，并开始评估这个想法的努力。Ashish 与 Illia 设计并实现了首批 Transformer 模型，并在这项工作的各个方面都发挥了关键作用。Noam 提议了缩放点积注意力、多头注意力和无参数的位置表示，并成为参与几乎所有细节的另一位关键人物。Niki 设计、实现、调整和评估了我们原始代码库和 tensor2tensor 中的无数模型变体。Llion 也实验了新的模型变体，负责我们最初的代码库以及高效的推理和可视化。Lukasz 和 Aidan 花费了无数漫长的日子设计和实现 tensor2tensor 的各个部分，替换了我们早期的代码库，极大地提高了结果，并极大地加速了我们的研究。

†在 Google Brain 期间完成的工作。

‡在 Google Research 期间完成的工作。

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

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 简介

循环神经网络，特别是长短期记忆 [13] 和门控循环 [7] 神经网络，已被牢固确立为序列建模和转换问题（如语言建模和机器翻译 [35, 2, 5]）中的最先进方法。此后，人们持续投入大量努力，以扩展循环语言模型和编码器-解码器架构 [38, 24, 15]的边界。

循环模型通常沿着输入和输出序列的符号位置进行计算。将位置与计算时间步对齐，它们生成一个隐藏状态序列 h_t，该序列是先前隐藏状态 h_{t-1} 和位置输入 t的函数。这种固有的顺序性质使得在训练样本内部无法并行化，这在较长的序列长度下变得至关重要，因为内存限制限制了跨样本的批处理。近期研究通过分解技巧 [21]和条件计算 [32], 在计算效率方面取得了显著改进，同时也在后者的情况下提升了模型性能。然而，序列计算的 fundamental constraint 仍然存在。

注意力机制已成为各种任务中引人入胜的序列建模和转导模型的重要组成部分，允许在不考虑输入或输出序列中距离的情况下建模依赖关系 [2, 19]。但在绝大多数情况下 [27], ，这种注意力机制是与循环网络一起使用的。

在这项工作中，我们提出了Transformer，一种避免递归、完全依赖注意力机制来建立输入和输出之间全局依赖关系的模型架构。Transformer允许显著更多的并行化，并且在经过在8块P100 GPU上训练仅12小时后，可以达到翻译质量的新水平。

2 背景

减少顺序计算的目标也构成了Extended Neural GPU[16], ByteNet [18] 和ConvS2S [9], 的基础，所有这些都使用卷积神经网络作为基本构建块，并行地为所有输入和输出位置计算隐藏表示。在这些模型中，将两个任意输入或输出位置信号关联所需的操作数量随位置之间的距离增长，对于ConvS2S呈线性增长，对于ByteNet呈对数增长。这使得学习远处位置之间的依赖关系更加困难 [12]。在Transformer中，这减少为常数数量的操作，尽管这导致了由于对注意力加权位置进行平均而导致的有效分辨率降低，我们通过第3.2节中描述的多头注意力机制来抵消这种效应。

自注意力机制，有时也称为序列内注意力，是一种关联单个序列不同位置的注意力机制，用于计算序列的表示。自注意力机制已成功应用于多种任务，包括阅读理解、抽象摘要、文本蕴涵以及学习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力机制，而非序列对齐的递归，已被证明在简单语言问答和语言建模任务上表现良好 [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其输入和输出表示的转换模型，而无需使用序列对齐的RNN或卷积。在接下来的部分中，我们将描述 Transformer，阐述自注意力机制，并讨论其相对于 [17, 18] 和 [9] 等模型的优点。

3 模型架构

大多数具有竞争力的神经序列转换模型都采用编码器-解码器结构 [5, 2, 35]。在此，编码器将符号表示的输入序列 $(x_1, ..., x_n)$ 映射为连续表示的序列 $\mathbf{z} = (z_1, ..., z_n)$。给定 z 后，解码器会逐个元素地生成符号组成的输出序列 $(y_1, ..., y_m)$。在每一步，模型都是自回归的 [10], ，在生成下一个元素时会将先前生成的符号作为附加输入进行消耗。

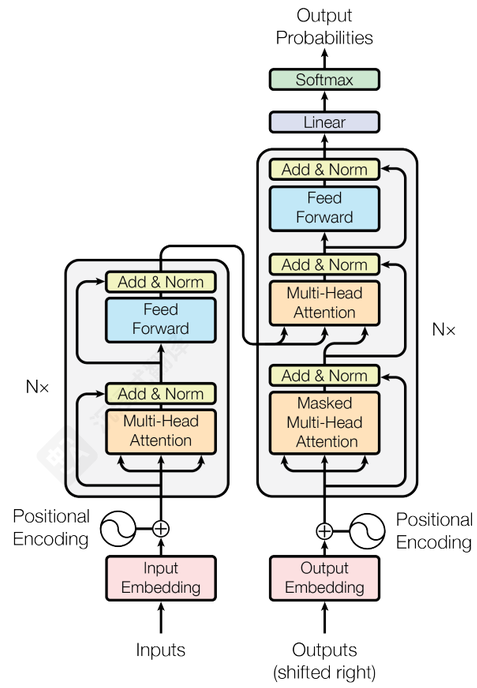

Figure 1: The Transformer - model architecture.

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

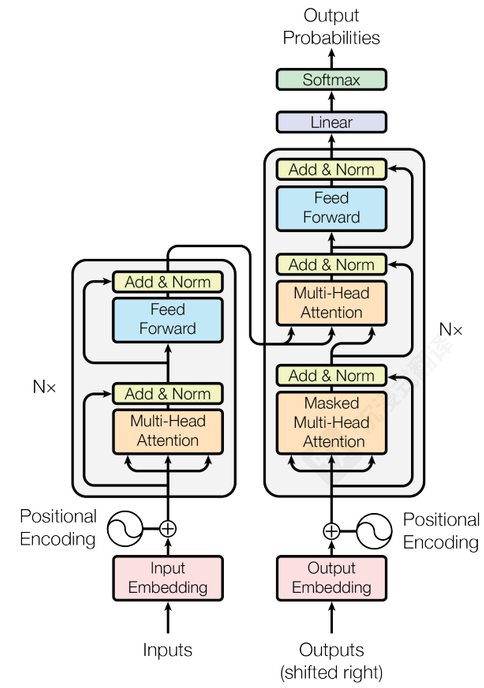

图1：Transformer - 模型架构。

Transformer 按照这种整体架构工作，对编码器和解码器都使用堆叠的自注意力和逐点全连接层，分别如图 1 的左半部和右半部所示。

3.1 编码器和解码器堆栈

编码器：编码器由 $N = 6$ 个相同的层堆栈组成。每一层有两个子层。第一个是多头自注意力机制，第二个是一个简单、位置感知的全连接前馈网络。我们在每个子层周围使用残差连接 [11]，然后进行层归一化 [1]。也就是说，每个子层的输出是 $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$，其中 $\mathrm{Sublayer}(x)$ 是子层本身实现的函数。为了方便这些残差连接，模型中的所有子层以及嵌入层都产生维度为 $d_{\text{model}} = 512$ 的输出。

解码器：解码器也由 $N = 6$ 个相同的层堆栈组成。除了每个编码器层中的两个子层外，解码器还插入一个第三个子层，该子层对编码器堆栈的输出执行多头注意力。与编码器类似，我们在每个子层周围使用残差连接，然后进行层归一化。我们还修改了解码器堆栈中的自注意力子层，以防止位置关注后续位置。这种掩码，加上输出嵌入偏移一个位置的事实，确保位置 i 的预测只能依赖于位置小于 i 的已知输出。

3.2 注意

注意力函数可以被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、值和输出都是向量。输出计算为值的加权求和

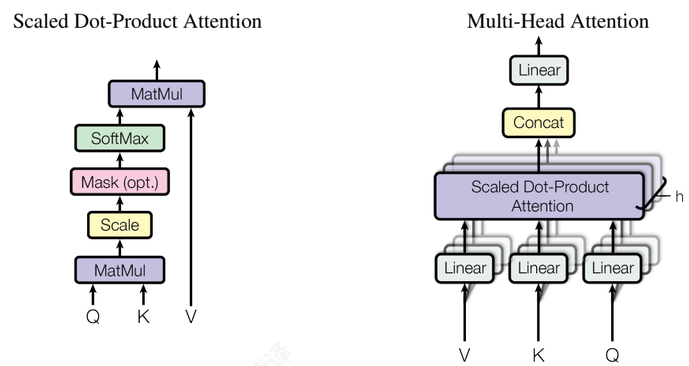

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

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:

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

The two most commonly used attention functions are additive attention [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.

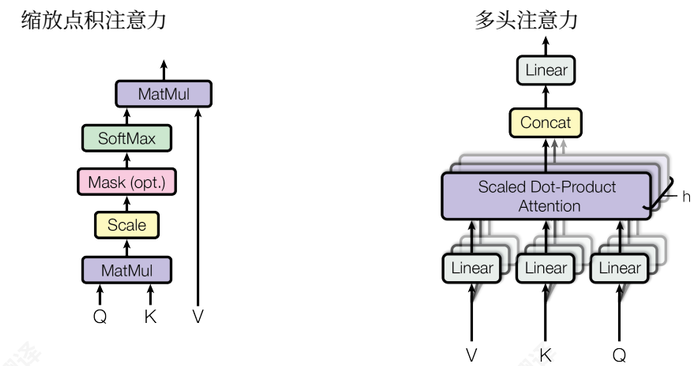

图2：(左)缩放点积注意力。(右)多头注意力由多个并行运行的注意力层组成。

其中分配给每个值的权重由查询与相应键的兼容性函数计算得出。

3.2.1 缩放点积注意力

我们将这种注意力称为"缩放点积注意力"（图2）。输入由维度为 d_k的查询和键，以及维度为 d_v的值组成。我们将查询与所有键的点积计算出来，除以 $\sqrt{d_k}$，并对值应用softmax函数以获得权重。

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

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

最常用的两种注意力函数是加性注意力 [2], 和点积（乘性）注意力。点积注意力与我们的算法相同，除了缩放因子 $\frac{1}{\sqrt{d_k}}$。加性注意力使用单隐藏层的前馈网络来计算兼容性函数。虽然两者在理论复杂度上相似，但在实践中，点积注意力速度更快、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。

虽然对于较小的 d_k 值，这两种机制表现相似，但对于较大的 d_k [3]值，加性注意力在无缩放的点积注意力中表现更优。我们推测，对于较大的d_k值，点积的幅度会变得很大，将softmax函数推入梯度极小的区域 [4]。为了抵消这种效应，我们对点积进行缩放 $\frac{1}{\sqrt{d_k}}$。

3.2.2 多头注意力

与其使用具有 d_{model}维键、值和查询的单个注意力函数，我们发现将查询、键和值线性投影 h 次，分别映射到 d_k、 d_k 和 d_v 维，效果更佳。然后，我们对这些投影后的查询、键和值并行执行注意力函数，得到 d_v维

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

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。这些值被连接起来，再次投影后，得到最终值，如图2所示。

多头注意力允许模型在不同位置同时关注来自不同表示子空间的信息。使用单个注意力头时，平均操作会抑制这种能力。

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

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

在这项工作中，我们采用 $h = 8$ 个并行注意力层，或称为头。对于每一个，我们使用 $d_k = d_v = d_\text{model}/h = 64$。由于每个头的维度降低，总计算成本与全维度单头注意力的成本相似。

3.2.3 注意力在我们模型中的应用

Transformer 在三种不同方式中使用多头注意力：

- 在 "编码器-解码器注意力" 层中，查询来自前一个解码器层，而记忆键和值来自编码器的输出。这允许解码器中的每个位置关注输入序列中的所有位置。这模拟了序列到序列模型（如 [38, 2, 9]）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，所有的键、值和查询都来自同一位置，在这种情况下，编码器前一层的输出。编码器中的每个位置可以关注编码器前一层中的所有位置。

- 类似地，解码器中的自注意力层允许解码器中的每个位置关注解码器中所有位置，包括该位置本身。我们需要防止解码器中的左向信息流动，以保持自回归特性。我们在缩放点积注意力内部通过屏蔽（设置为 $-\infty$）softmax 输入中对应非法连接的所有值来实现这一点。见图 2。

3.3 位置式前馈网络

除了注意力子层之外，我们编码器和解码器中的每一层都包含一个全连接前馈网络，该网络针对每个位置单独且相同地应用。这由两个线性变换和中间的 ReLU 激活组成。

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

虽然线性变换在不同位置上是相同的，但它们在层与层之间使用不同的参数。另一种描述方式是将其视为两个内核大小为1的卷积。输入和输出的维度是 $d_\text{model} = 512$，内层的维度是$d_{ff} = 2048$。

3.4 嵌入和Softmax

与其他序列转导模型类似，我们使用学习到的嵌入将输入标记和输出标记转换为维度为 d_model的向量。我们还使用通常的学习到的线性变换和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，我们在两个嵌入层和预softmax线性变换之间共享相同的权重矩阵，类似于 [30]。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、每层复杂度和最小顺序操作数。 n 是序列长度， d 是表示维度， k 是卷积的核大小， r 是受限自注意力中的邻域大小。

层类型	每层复杂度	顺序操作	最大路径长度
自注意力	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
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

由于我们的模型不包含递归和卷积，为了让模型利用序列的顺序，我们必须注入一些关于序列中标记的相对或绝对位置的信息。为此，我们在编码器和解码器堆栈的底部向输入嵌入添加“位置编码”。位置编码的维度与嵌入相同，以便两者可以相加。位置编码有很多选择，包括学习和固定的 [9]。

在本工作中，我们使用不同频率的正弦和余弦函数：

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

pos 在哪里是位置， i 在哪里是维度。也就是说，位置编码的每个维度都对应一个正弦波。波长从 2π 到 $10000 \cdot 2\pi$ 形成一个几何级数。我们选择这个函数是因为我们假设它可以让模型轻松地学习通过相对位置来关注，因为对于任何固定的偏移量 k， PE_{pos+k} 可以表示为 PE_{pos} 的线性函数。

我们还尝试使用学习过的位置嵌入 [9]，但发现两个版本产生的结果几乎相同（见表3行(E)）。我们选择正弦版本是因为它可能允许模型推广到比训练中遇到的更长的序列长度。

4 为什么是自注意力

在本节中，我们将自注意力层与通常用于将一个符号表示的变量长度序列 $(x_1, ..., x_n)$ 映射到另一个等长序列 $(z_1, ..., z_n)$ 的循环和卷积层进行比较，例如典型序列转换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，我们考虑了三个期望。

一个是每层的总计算复杂度。另一个是可以并行化的计算量，以所需的最少顺序操作数来衡量。

第三个是网络中长期依赖之间的路径长度。学习长期依赖是许多序列转换任务中的一个关键挑战。影响学习这种依赖能力的一个关键因素是前向和后向信号在网络中必须穿越的路径长度。输入和输出序列中任何位置组合之间的路径越短，学习长期依赖就越容易 [12]。因此，我们还比较了由不同层类型组成的网络中，任何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。在计算复杂度方面，当序列长度 O小于表示维度n时，自注意力层比循环层更快，这在机器翻译中最先进的模型（如word-piece和byte-pair 表示）使用的句子表示中通常是这种情况。为了提高涉及非常长序列的任务的计算性能，自注意力可以限制为仅考虑输入序列中围绕相应输出位置的大小为 的邻域。这将使最大路径长度增加到 ()。我们计划在未来的工作中进一步研究这种方法。

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:

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

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

5.4 Regularization

We employ three types of regularization during training:

长度 n 小于表示维度 d，这通常是机器翻译中最先进的模型（如word-piece[38] 和 byte-pair [31] 表示）使用的句子表示的情况。为了提高涉及非常长序列的任务的计算性能，自注意力可以限制为仅考虑输入序列中围绕相应输出位置的大小为 r 的邻域。这将使最大路径长度增加到 $O(n/r)$。我们计划在未来的工作中进一步研究这种方法。

一个卷积核宽度为 $k < n$ 的单层卷积层并不能连接所有输入和输出位置。要实现这一点，在连续卷积核的情况下需要堆叠 $O(n/k)$ 层卷积层，而在扩张卷积的情况下则需要 $O(log_k(n))$ 层卷积层 [18], ，从而增加网络中任意两个位置之间最长路径的长度。卷积层通常比循环层更昂贵，成本高出 k 倍。然而，可分离卷积 [6]，可以显著降低复杂度，至 $O(k \cdot n \cdot d + n \cdot d^2)$。即便如此，可分离卷积的复杂度仍等于自注意力层和逐点前馈层的组合，这也是我们模型采用的方法。

作为一项附加好处，自注意力机制可以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，并在附录中呈现和讨论了相关示例。不仅单个注意力头能清晰地学习执行不同任务，许多注意力头似乎还表现出与句子句法和语义结构相关的行为。

5 训练

本节描述了我们模型的训练流程。

5.1 训练数据和批处理

我们在标准 WMT 2014 英德数据集上进行了训练，该数据集包含约 450 万个句子对。句子使用字节对编码 [3], 进行编码，其共享的源-目标词汇表约有 37000 个词元。对于英法语，我们使用了规模显著更大的 WMT 2014 英法语数据集，包含 3600 万个句子，并将词元分割为 32000 个词元词汇 [38]。句子对按近似序列长度进行批处理。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 硬件和计划

我们在一台配备 8 块 NVIDIA P100 GPU 的机器上训练我们的模型。对于本文中描述的超参数所使用的基模型，每个训练步骤约耗时 0.4 秒。我们训练基模型共 100,000 步或 12 小时。对于我们的大模型（见表 3 最下面一行所述），步骤耗时为 1.0 秒。大模型训练了 300,000 步（3.5 天）。

5.3 优化器

我们使用了 Adam 优化器 [20]，结合 $\beta_1 = 0.9$、$\beta_2 = 0.98$ 和 $\epsilon = 10^{-9}$。我们在训练过程中调整学习率，具体公式如下：

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

这对应于在前 $warmup_steps$ 个训练步骤中线性增加学习率，之后则按步骤数的平方根倒数成比例减小。我们使用了 $warmup_steps = 4000$。

5.4 正则化

我们在训练过程中采用三种正则化方法：

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：Transformer在英语到德语和英语到法语的NewsTest2014测试中，以更低的训练成本实现了比先前最先进模型更好的BLEU分数。

模型	BLEU		训练成本（FLOPs）	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + 位置未知 [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 (大型)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

残差Dropout 我们在子层的输出上应用dropout [33]，然后将其添加到子层输入并进行归一化。此外，我们还在编码器和解码器堆栈中的嵌入和位置编码的和中应用dropout。对于基础模型，我们使用$P_{drop} = 0.1$的比率。

标签平滑 在训练过程中，我们采用了值为 $\epsilon_{ls} = 0.1$ [36]的标签平滑。这会降低困惑度，因为模型会变得不那么确定，但能提高准确率和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)）比之前报道的最佳模型（包括集成模型）的BLEU分数高出2.0以上，创下了新的BLEU分数纪录28.4。该模型的配置列于表3的最底行。在8块P100 GPU上训练耗时3.5天。即使我们的基础模型也超越了所有之前发表的模型和集成模型，且训练成本远低于任何竞争模型。

在WMT 2014英语到法语翻译任务中，我们的大型模型取得了41.0的BLEU分数，超越了所有之前发表的单一模型，且训练成本不到前一代最佳模型的 $1/4$ 。用于英语到法语训练的Transformer (big) 模型使用了dropout率 $P_{drop} = 0.1$，而不是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分钟记录一次）得到的单个模型。对于大型模型，我们平均了最后20个检查点。我们使用了beam search，beam大小为4，并应用了长度惩罚 $\alpha = 0.6$ [38]。这些超参数是在开发集上进行实验后选择的。我们在推理时将最大输出长度设置为输入长度 + 50，但在可能的情况下提前终止 [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其他文献中的模型架构进行了比较。我们通过将训练时间、使用的GPU数量以及每个GPU的持续单精度浮点性能估计值[5]相乘，来估计训练模型所需的浮点运算次数。

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开发集（newstest2013）上的性能变化。

[5] 我们分别为K80、K40、M40和P100使用了2.8、3.7、6.0和9.5 TFLOPS的值。

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。所有指标均在英语到德语的翻译开发集newstest2013上。列出的困惑度是按我们的字节对编码的词片级别，不应与词级别困惑度进行比较。

	N	$d_{\text{模型}}$	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	训练步数	PPL (开发)	BLEU (开发)	参数 $\times 10^6$
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)	位置编码而非正弦函数									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

我们使用了上一节中描述的集束搜索（beam search），但没有进行检查点平均（checkpoint averaging）。这些结果展示在表 3 中。
在表3的行(A)中，我们变化了注意力头的数量以及注意力键和值的维度，同时保持计算量不变，如第3.2.2节所述。虽然单头注意力比最佳设置差0.9个BLEU，但过多的头数也会导致质量下降。

在表3的行(B)中，我们观察到减少注意力键大小 d_k 会降低模型质量。这表明确定兼容性并不容易，并且可能需要比点积更复杂的兼容性函数。我们进一步观察到，在行(C)和(D)中，正如预期的那样，更大的模型效果更好，而dropout在避免过拟合方面非常有帮助。在行(E)中，我们用学习的位置编码 [9],替换了我们的正弦位置编码，并观察到与基础模型几乎相同的结果。

6.3 英语依存句法分析

为了评估Transformer是否可以泛化到其他任务，我们在英语依存句法分析上进行了实验。这项任务提出了特定的挑战：输出受强结构约束，并且显著长于输入。此外，RNN序列到序列模型在小数据环境下尚未达到最先进的结果 [37]。

我们在华尔街日报（WSJ）部分的宾夕法尼亚树库（Penn Treebank）上训练了一个4层Transformer，使用了约40K个训练句子。我们还以半监督的方式训练了它，使用了来自更大规模的、高置信度的BerkleyParser语料库，大约有17M个句子[37]。对于WSJ单独的设置，我们使用了16K个token的词汇表；对于半监督的设置，我们使用了32K个token的词汇表。

我们仅对 Section 22 开发集上选择 dropout、注意力机制和残差（第 5.4 节）的学习率及束大小进行了一小部分实验，其余所有参数均保持与英语到德语的基准翻译模型相同。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在英语依存句法分析上表现良好（结果见WSJ第23节）

解析器	训练	WSJ 23 F1
Vinyals & Kaiser等（2014） [37]	仅WSJ，判别式	88.3
Petrov 等人 (2006) [29]	仅 WSJ，区分性	90.4
Zhu 等人 (2013) [40]	仅 WSJ，区分性	90.4
Dyer 等人 (2016) [8]	仅 WSJ，区分性	91.7
Transformer (4层)	仅WSJ，判别性	91.3
Zhu等人 (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 与 Kaiser 等人 (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

将最大输出长度增加到输入长度 + 300。我们使用了束大小 21 和 $\alpha = 0.3$，分别用于仅 WSJ 和半监督设置。

表4中的结果表明，尽管缺乏针对特定任务的微调，我们的模型表现出了惊人的性能，其结果优于所有先前报道的模型，除了循环神经网络语法 [8]。

与 RNN 序列到序列模型 [37],相比，Transformer 甚至在仅使用 40K 个句子的 WSJ 训练集上进行训练时，也优于 Berkeley 解析器 [29] 。

7 结论

在这项工作中，我们提出了Transformer，这是第一个完全基于注意力机制的序列转换模型，用多头自注意力机制取代了编码器-解码器架构中最常用的循环层。

对于翻译任务，Transformer的训练速度比基于循环层或卷积层的架构快得多。在WMT 2014英语到德语和WMT 2014英语到法语翻译任务中，我们取得了新的技术突破。在前者中，我们的最佳模型甚至超越了所有先前报告的集成模型。

我们对基于注意力的模型的未来充满期待，并计划将它们应用于其他任务。我们计划将Transformer扩展到涉及文本以外的输入和输出模态的问题，并研究局部、受限的注意力机制，以高效处理图像、音频和视频等大型输入和输出。使生成过程不那么序列化也是我们另一个研究目标。

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

致谢 我们感谢 Nal Kalchbrenner 和 Stephan Gouws 的富有成效的评论、纠正和启发。

参考文献

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[8] 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. 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.

11

[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 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.

12

Attention Visualizations

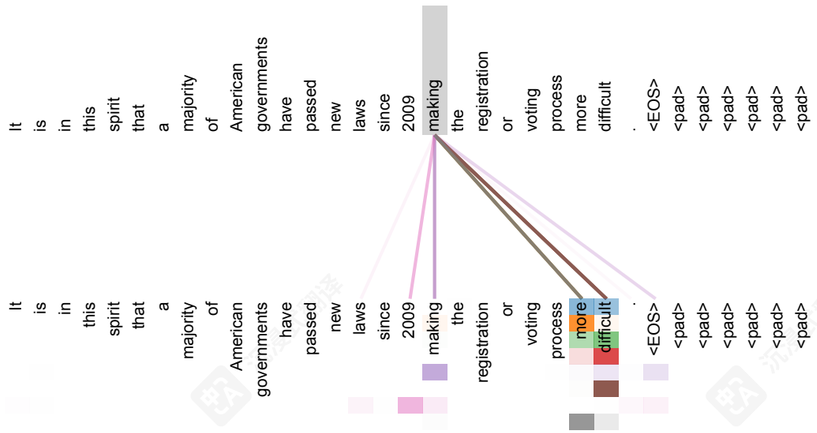

Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word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

注意力可视化

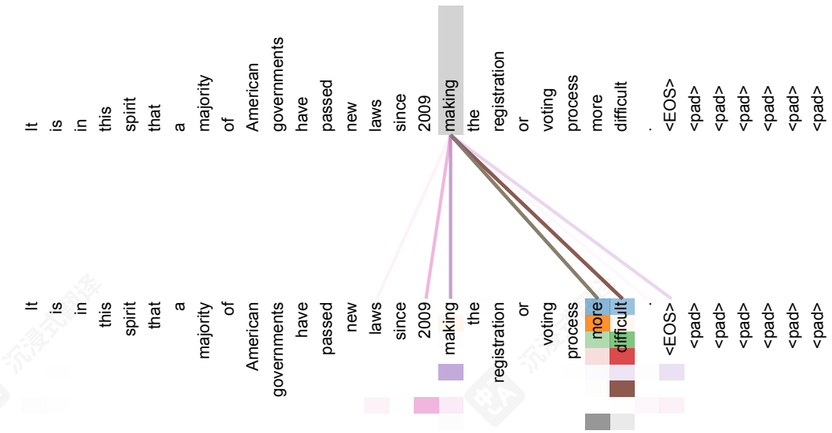

图3：一个示例，展示了第6层的第5层编码器自注意力机制中，长距离依赖的注意力机制。许多注意力头关注动词‘making’的远距离依赖，完成了短语‘making...更困难’。这里仅显示了单词‘making’的注意力。不同颜色代表不同的头。建议彩色查看。

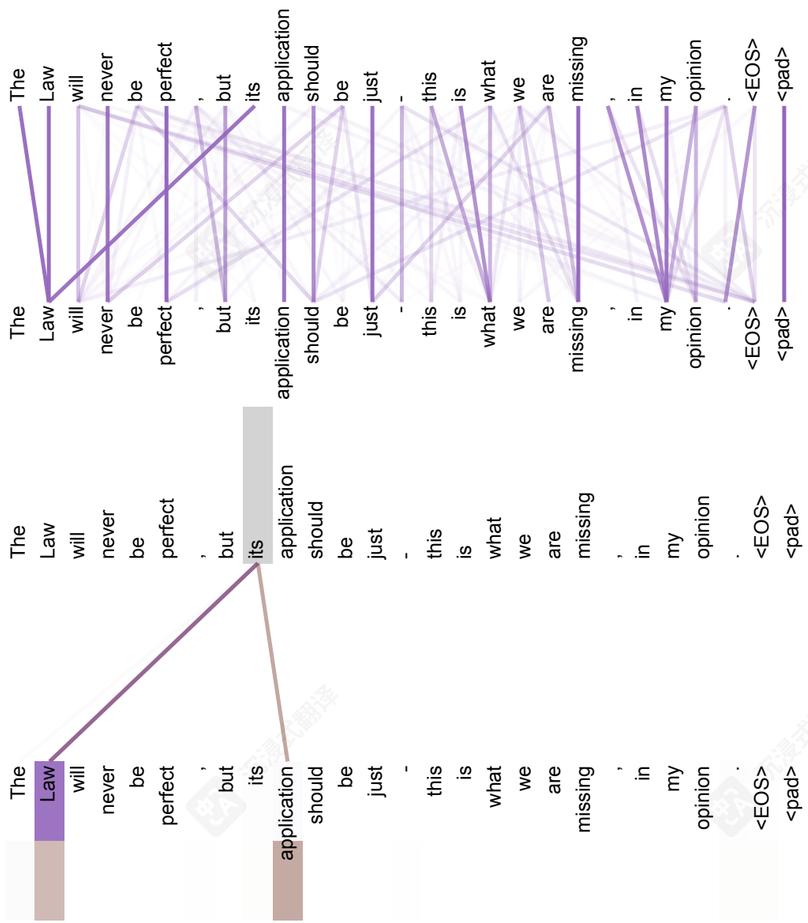

Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

图4：两个注意力头，显然参与了指代消解，位于第6层的第5层。顶部：头5的完整注意力。底部：仅从单词‘其’对头5和6的注意力。注意，这个单词的注意力非常集中。

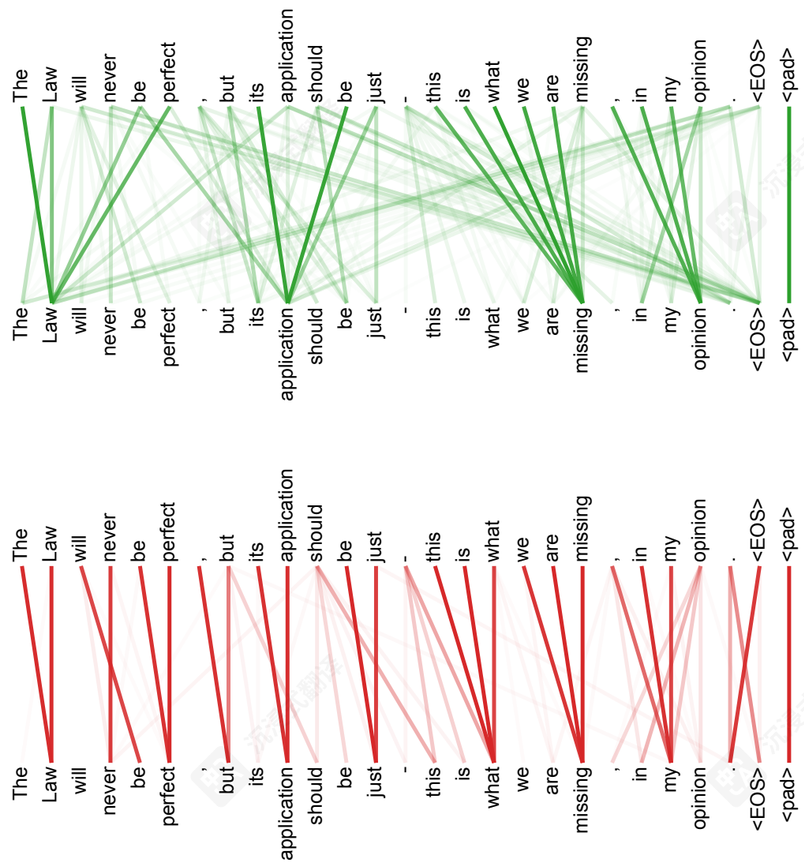

Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

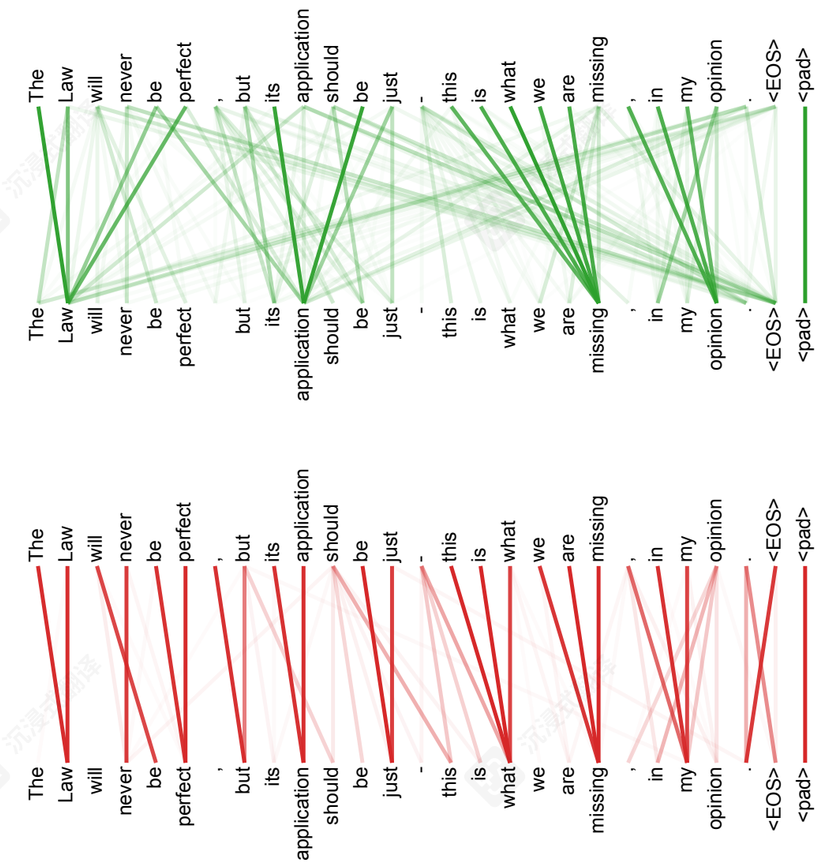

图5：许多注意力头表现出与句子结构相关的行为。我们在上面给出了两个这样的例子，来自编码器自注意力机制在第5层（共6层）的两个不同头。这些头显然学会了执行不同的任务。