在注明出处的前提下，Google 特此授予仅为新闻报道或学术研究之目的复制本文图表的许可。

注意力即一切

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*†
多伦多大学
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin*‡
illia.polosukhin@gmail.com

摘　要

主流的序列转导模型大多基于包含编码器与解码器的复杂循环或卷积神经网络。表现最好的模型还会通过注意力机制将编码器与解码器连接起来。我们提出一种全新的简单网络架构——Transformer，它完全基于注意力机制，彻底摒弃了循环与卷积。在两个机器翻译任务上的实验表明，该模型在质量上更优，同时具有更强的并行性，所需训练时间也大幅缩短。我们的模型在 WMT 2014 英译德任务上取得了 28.4 BLEU，比包括集成模型在内的现有最佳结果高出 2 BLEU 以上。在 WMT 2014 英译法任务上，我们的模型在 8 块 GPU 上训练 3.5 天后，取得了 41.8 BLEU 的单模型新纪录，而其训练成本仅为文献中最佳模型的一小部分。我们还将 Transformer 成功应用于训练数据充足与训练数据有限两种情形下的英文成分句法分析，证明它能够很好地泛化到其他任务。

*同等贡献。作者排列顺序随机。Jakob 提出用自注意力取代 RNN，并启动了对这一想法的评估。Ashish 与 Illia 设计并实现了最初的 Transformer 模型，并在本工作的各个方面都发挥了关键作用。Noam 提出了缩放点积注意力、多头注意力以及无参数的位置表示，是另一位几乎参与了每一处细节的人。Niki 在我们最初的代码库和 tensor2tensor 中设计、实现、调优并评估了无数模型变体。Llion 也试验了新颖的模型变体，负责最初的代码库以及高效推理与可视化。Łukasz 与 Aidan 耗费了无数漫长的时日设计并实现 tensor2tensor 的各个部分，取代了我们早期的代码库，显著改善了结果并极大地加速了研究。

†在 Google Brain 工作期间完成。

‡在 Google Research 工作期间完成。

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。

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

1 引言

循环神经网络，尤其是长短期记忆网络[13]和门控循环神经网络[7]，已被牢固确立为语言建模、机器翻译等序列建模与转导问题的最先进方法[35, 2, 5]。此后，大量工作继续拓展循环语言模型与编码器-解码器架构的边界[38, 24, 15]。

循环模型通常沿输入序列与输出序列的符号位置来分解计算。它们把位置与计算时间上的步骤对齐，生成一系列隐藏状态 h_t，作为前一个隐藏状态 h_{t-1} 与位置 t 处输入的函数。这种固有的顺序特性使训练样本内部无法并行化；当序列长度较长时，这一问题变得尤为关键，因为内存限制了跨样本的批处理。近期的工作通过分解

技巧[21]和条件计算[32]在计算效率上取得了显著提升，后者还同时改善了模型性能。然而，顺序计算这一根本约束依然存在。

在各类任务中，注意力机制已成为性能出色的序列建模与转导模型不可或缺的组成部分，它能够在不考虑依赖在输入或输出序列中距离远近的情况下对依赖进行建模[2, 19]。然而，除少数情况外[27]，这类注意力机制都是与循环网络结合使用的。

在本工作中，我们提出了 Transformer，这是一种摒弃循环、完全依赖注意力机制来刻画输入与输出之间全局依赖的模型架构。Transformer 具有强得多的并行性，在 8 块 P100 GPU 上仅训练十二小时，便能达到翻译质量的新的最先进水平。

2 背景

减少顺序计算这一目标同样构成了 Extended Neural GPU[16]、ByteNet[18] 和 ConvS2S[9] 的基础，它们都以卷积神经网络作为基本构件，为所有输入和输出位置并行计算隐藏表示。在这些模型中，关联任意两个输入或输出位置的信号所需的操作数会随位置之间的距离而增长——ConvS2S 中为线性增长，ByteNet 中为对数增长。这使得学习远距离位置之间的依赖变得更加困难[12]。在 Transformer 中，这一操作数被减少为常数，尽管对注意力加权位置取平均会降低有效分辨率；我们用第 3.2 节所述的多头注意力来抵消这一影响。

自注意力（有时称为内部注意力）是一种将单个序列的不同位置关联起来、以计算该序列表示的注意力机制。自注意力已成功应用于多种任务，包括阅读理解、生成式摘要、文本蕴涵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力机制而非与序列对齐的循环，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 是第一个完全依靠自注意力来计算其输入与输出表示、而不使用与序列对齐的 RNN 或卷积的转导模型。在接下来的各节中，我们将介绍 Transformer，阐明自注意力的动机，并讨论它相较 [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]。其中，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 $z = (z_1, \ldots, z_n)$。给定 z，解码器随后逐元素生成由符号组成的输出序列 $(y_1, \ldots, y_m)$。在每一步，模型都是自回归的[10]，即在生成下一个符号时，会把先前生成的符号作为额外输入。

Transformer 在编码器和解码器中都采用堆叠的自注意力层与逐点全连接层来实现这一整体架构，分别如图 1 的左半部分与右半部分所示。

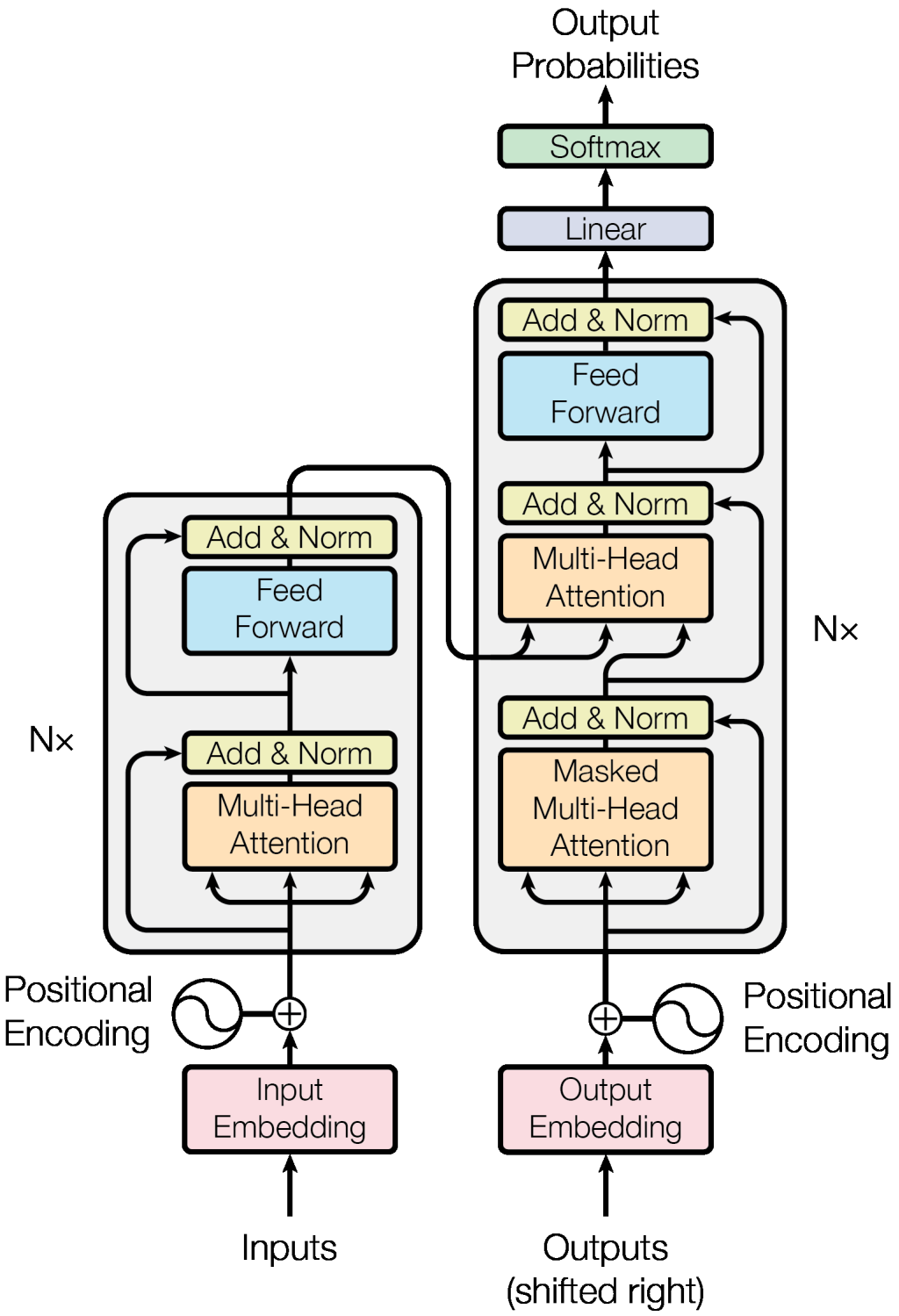


图 1 Transformer —— 模型架构。（图中标签为原图英文）

3.1 编码器与解码器堆叠

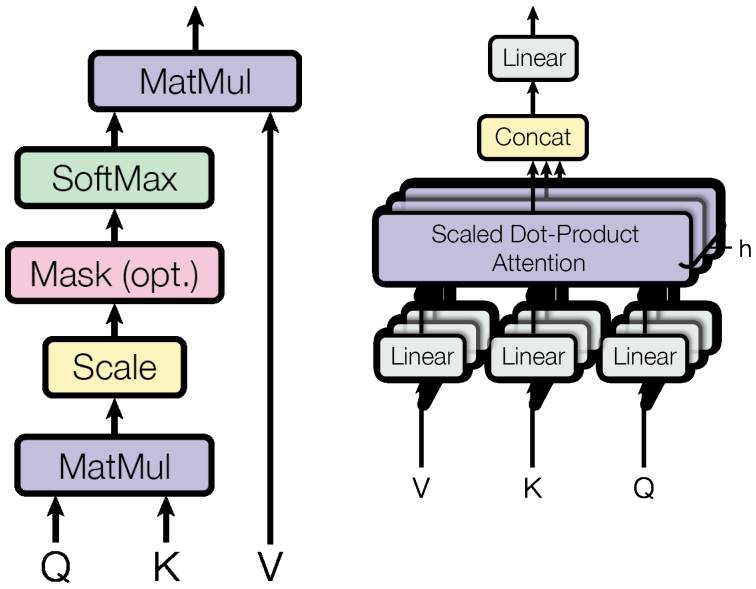
编码器：编码器由 $N = 6$ 个相同的层堆叠而成。每个层包含两个子层。第一个是多头自注意力机制，第二个是简单的、逐位置的全连接前馈网络。我们在每个子层周围采用残差连接[11]，随后进行层归一化[1]。也就是说，每个子层的输出为 $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$，其中 $\mathrm{Sublayer}(x)$ 是该子层自身实现的函数。为便于实现这些残差连接，模型中所有子层以及嵌入层都输出维度为 $d_{\text{model}} = 512$ 的结果。

解码器：解码器同样由 $N = 6$ 个相同的层堆叠而成。除每个编码器层中的两个子层外，解码器还插入第三个子层，该子层对编码器堆叠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采用残差连接，随后

进行层归一化。我们还对解码器堆叠中的自注意力子层加以修改，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做法，确保了对位置 i 的预测只能依赖于位置 i 之前的已知输出。

3.2 注意力

注意力函数可以描述为：将一个查询和一组键-值对映射为一个输出，其中查询、键、值和输出均为向量。输出为这些值的加权和，而赋予每个值的权重由查询与对应键的兼容性函数计算得到。



缩放点积注意力 **多头注意力**

图 2 （左）缩放点积注意力。（右）多头注意力由若干并行运行的注意力层组成。（图中标签为原图英文）

3.2.1 缩放点积注意力

我们将自己所用的特定注意力称为“缩放点积注意力”（图 2）。其输入由维度为 d_k 的查询和键、以及维度为 d_v 的值组成。我们计算查询与所有键的点积，将每个点积除以 $\sqrt{d_k}$，然后应用 softmax 函数以得到这些值上的权重。

在实际中，我们对一组查询同时计算注意力函数，将它们打包成矩阵 Q。键和值也分别打包成矩阵 K 和 V。输出矩阵按下式计算：

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

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。点积注意力除了缩放因子 $\frac{1}{\sqrt{d_k}}$ 之外，与我们的算法完全相同。加性注意力使用带单个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近，但点积注意力在实际中要快得多、也更节省空间，因为它可以借助高度优化的矩阵乘法代码来实现。

当 d_k 取值较小时，两种机制表现相近；但当 d_k 较大时，不加缩放的点积注意力不如加性注意力[3]。我们推测，对于较大的 d_k，点积的数值会变得很大，将 softmax 函数推入梯度极小的区域[4]。为抵消这一影响，我们用 $\frac{1}{\sqrt{d_k}}$ 对点积进行缩放。

4为说明点积为何会变大，假设 q 和 k 的各分量是均值为 0、方差为 1 的独立随机变量，则它们的点积 $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$ 的均值为 0、方差为 d_k。

3.2.2 多头注意力

我们发现，与其用 d_{model} 维的键、值和查询执行单一的注意力函数，不如用不同的、学习得到的线性投影，将查询、键和值分别线性投影 h 次到 d_k、d_k 和 d_v 维，这样做是有益的。随后，在这些投影后的查询、键和值上，我们并行地执行注意力函数，得到 d_v 维的输出值。将这些输出拼接后再次投影，便得到最终的值，如图 2 所示。

多头注意力使模型能够联合关注不同位置上来自不同表示子空间的信息。若只有单个注意力头，取平均会妨碍这种能力。

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O$$

$$\text{其中} \quad \text{head}_i = \text{Attention}(Q W_i^Q, K W_i^K, V W_i^V)$$

这些投影为参数矩阵 $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$、$W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$、$W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$，以及 $W^O \in \mathbb{R}^{h d_v \times d_{\text{model}}}$。

在本工作中，我们采用 $h = 8$ 个并行的注意力层（即头）。对每个头，我们使用 $d_k = d_v = d_{\text{model}} / h = 64$。由于每个头的维度有所降低，总的计算代价与使用完整维度的单头注意力相近。

3.2.3 注意力在本模型中的应用

Transformer 以三种不同方式使用多头注意力：

- 在“编码器-解码器注意力”层中，查询来自前一个解码器层，而记忆中的键和值来自编码器的输出。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，效仿了序列到序列模型（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处，即编码器中前一层的输出。编码器中的每个位置都能关注编码器前一层的所有位置。
- 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截至并包括该位置在内的所有位置。为保持自回归特性，我们需要阻止解码器中向左的信息流。我们在缩放点积注意力内部通过掩码来实现这一点——将 softmax 输入中对应于非法连接的所有值置为 $-\infty$。见图 2。

3.3 逐位置前馈网络

除注意力子层外，编码器和解码器中的每个层都包含一个全连接前馈网络，它被分别且相同地应用于每个位置。该网络由两个线性变换以及夹在其间的 ReLU 激活组成。

$$\text{FFN}(x) = \max(0, x W_1 + b_1) W_2 + b_2 \tag{2}$$

尽管这些线性变换在不同位置上相同，但它们在不同层之间使用不同参数。另一种描述方式是把它看作两个卷积核大小为 1 的卷积。输入与输出的维度为 $d_{\text{model}} = 512$，内层的维度为 $d_{ff} = 2048$。

3.4 嵌入与 Softmax

与其他序列转导模型类似，我们使用学习得到的嵌入将输入词元和输出词元转换为维度为 d_{model} 的向量。我们还使用常规的、学习得到的线性变换和 softmax 函数，将解码器输出转换为对下一个词元的预测概率。在本模型中，我们在两个嵌入层与 softmax 之前的线性变换之间共享同一权重矩阵，这与[30]类似。在嵌入层中，我们将这些权重乘以 $\sqrt{d_{\text{model}}}$。

表 1 不同层类型的最大路径长度、每层复杂度与最少顺序操作数。其中 n 为序列长度，d 为表示维度，k 为卷积的卷积核大小，r 为受限自注意力中邻域的大小。

层类型	每层复杂度	顺序操作数	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循环和卷积，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词元相对或绝对位置的信息。为此，我们在编码器和解码器堆叠底部的输入嵌入上加入“位置编码”。位置编码与嵌入具有相同的维度 d_{model}，从而两者可以相加。位置编码有多种选择，包括学习得到的和固定的[9]。

在本工作中，我们使用不同频率的正弦和余弦函数：

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \sin\Big(\mathrm{pos}/10000^{2i/d_{\text{model}}}\Big)$$

$$\mathrm{PE}_{(\mathrm{pos},2i+1)} = \cos\Big(\mathrm{pos}/10000^{2i/d_{\text{model}}}\Big)$$

其中 pos 为位置，i 为维度。也就是说，位置编码的每个维度对应一条正弦曲线，其波长构成从 2π 到 $10000 \cdot 2\pi$ 的等比数列。我们选择这一函数，是因为我们假设它能让模型容易地学会按相对位置进行关注：对于任意固定偏移 k，$\mathrm{PE}_{\mathrm{pos}+k}$ 都可以表示为 $\mathrm{PE}_{\mathrm{pos}}$ 的线性函数。

我们还试验了改用学习得到的位置嵌入[9]，发现两个版本产生了几乎相同的结果（见表 3 第 (E) 行）。我们选择正弦版本，是因为它或许能让模型外推到比训练期间所遇到的更长的序列长度。

4 为何选择自注意力

在本节中，我们将自注意力层与常用于把一个变长的符号表示序列 $(x_1, \ldots, x_n)$ 映射为等长的另一个序列 $(z_1, \ldots, z_n)$（其中 $x_i, z_i \in \mathbb{R}^d$）的循环层和卷积层进行多方面比较，例如典型序列转导编码器或解码器中的某个隐藏层。出于采用自注意力的动机，我们考虑三个方面的要求。

其一是每层的总计算复杂度；其二是可并行的计算量，以所需的最少顺序操作数来衡量。

其三是网络中长距离依赖之间的路径长度。学习长距离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，是前向与后向信号在网络中必须经过的路径长度。输入序列与输出序列中任意位置组合之间的这些路径越短，就越容易学习长距离依赖[12]。因此，我们还比较了由不同层类型构成的网络中任意两个输入与输出位置之间的最大路径长度。

如表 1 所示，自注意力层以常数个顺序执行的操作连接所有位置，而循环层需要 $O(n)$ 个顺序操作。在计算复杂度方面，当序列长度 n 小于表示维度 d 时，自注意力层比循环层更快；机器翻译中最先进模型所使用的句子表示（如词块[38]和字节对[31]表示）通常正是这种情形。为提升涉及超长序列任务的计算性能，可以将自注意力限制为：在输入序列中，只考虑以相应输出位置为中心、大小为 r 的邻域。这会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一方法。

卷积核宽度 $k < n$ 的单个卷积层无法连接所有输入与输出位置对。要做到这一点，在连续卷积核的情形下需要堆叠 $O(n/k)$ 个卷积层，在空洞卷积的情形下则需要 $O(\log_k(n))$ 个[18]，这会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贵 k 倍。然而，可分离卷积[6]将复杂度大幅降低至 $O(k \cdot n \cdot d + n \cdot d^2)$。即便取 $k = n$，可分离卷积的复杂度也等于一个自注意力层与一个逐点前馈层的组合，而这正是我们在模型中采用的做法。

作为附带的好处，自注意力还能带来更具可解释性的模型。我们检查了模型的注意力分布，并在下面给出并讨论了一些示例。不仅各个注意力头明显学会了执行不同的任务，许多注意力头还表现出与句子句法和语义结构相关的行为。

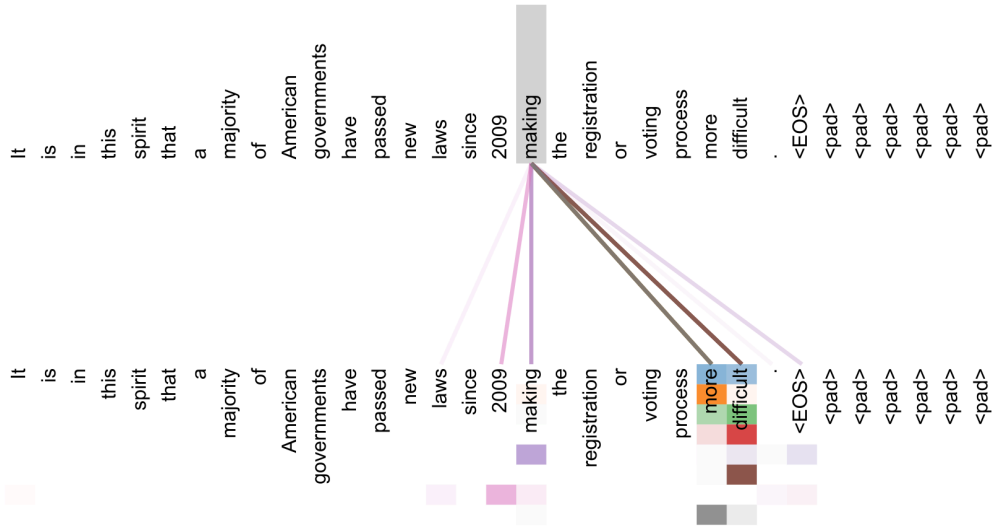


图 3 编码器第 5 层（共 6 层）自注意力中，注意力机制跟随长距离依赖的一个示例。许多注意力头关注动词“making”的一个远距离依赖，从而补全短语“making…more difficult”。此处仅显示针对单词“making”的注意力；不同颜色代表不同的头，建议以彩色查看。（图中标签为原图英文）

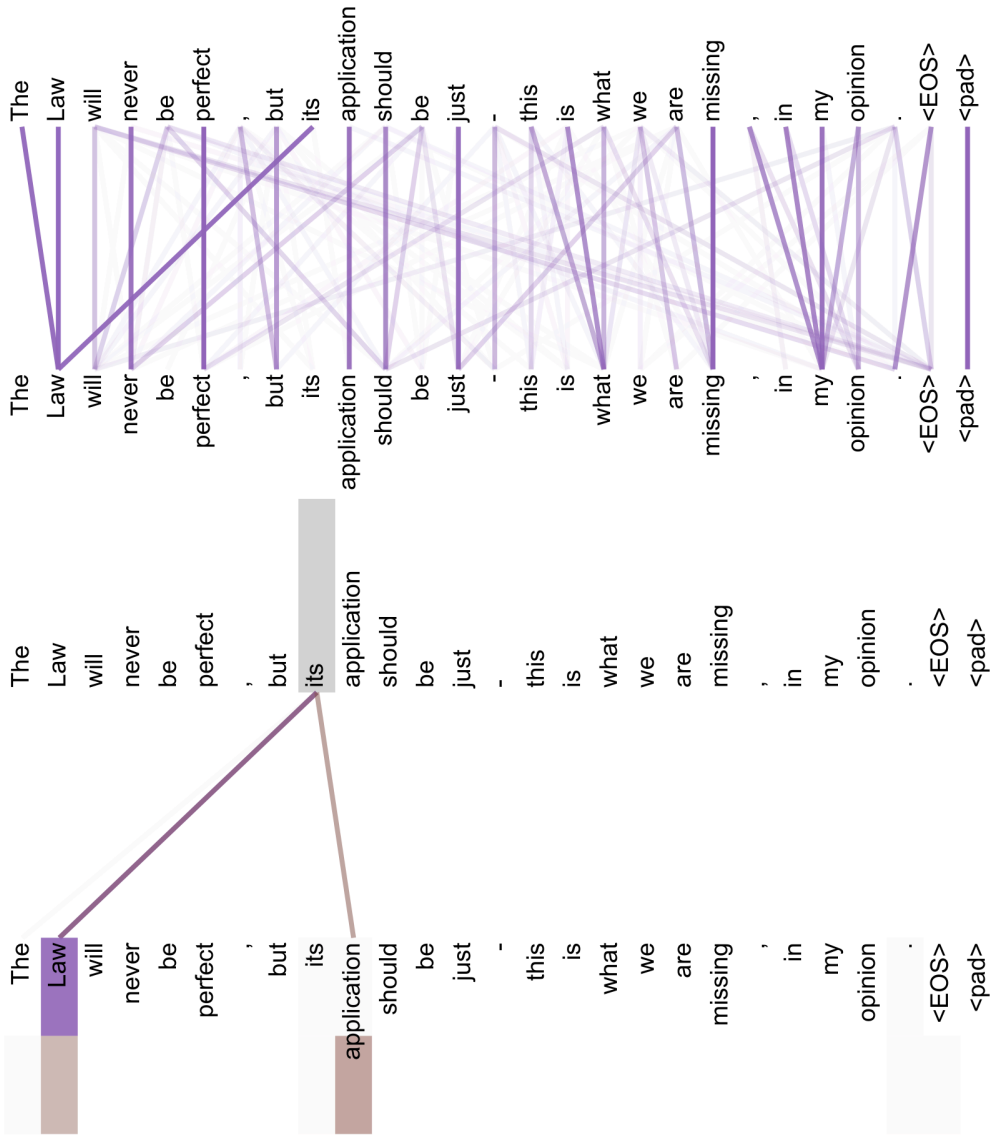


图 4 同样在第 5 层（共 6 层）中，两个显然参与了指代消解的注意力头。上：第 5 号头的完整注意力；下：第 5 号和第 6 号头仅针对单词“its”的孤立注意力。请注意，对该词的注意力非常集中。（图中标签为原图英文）

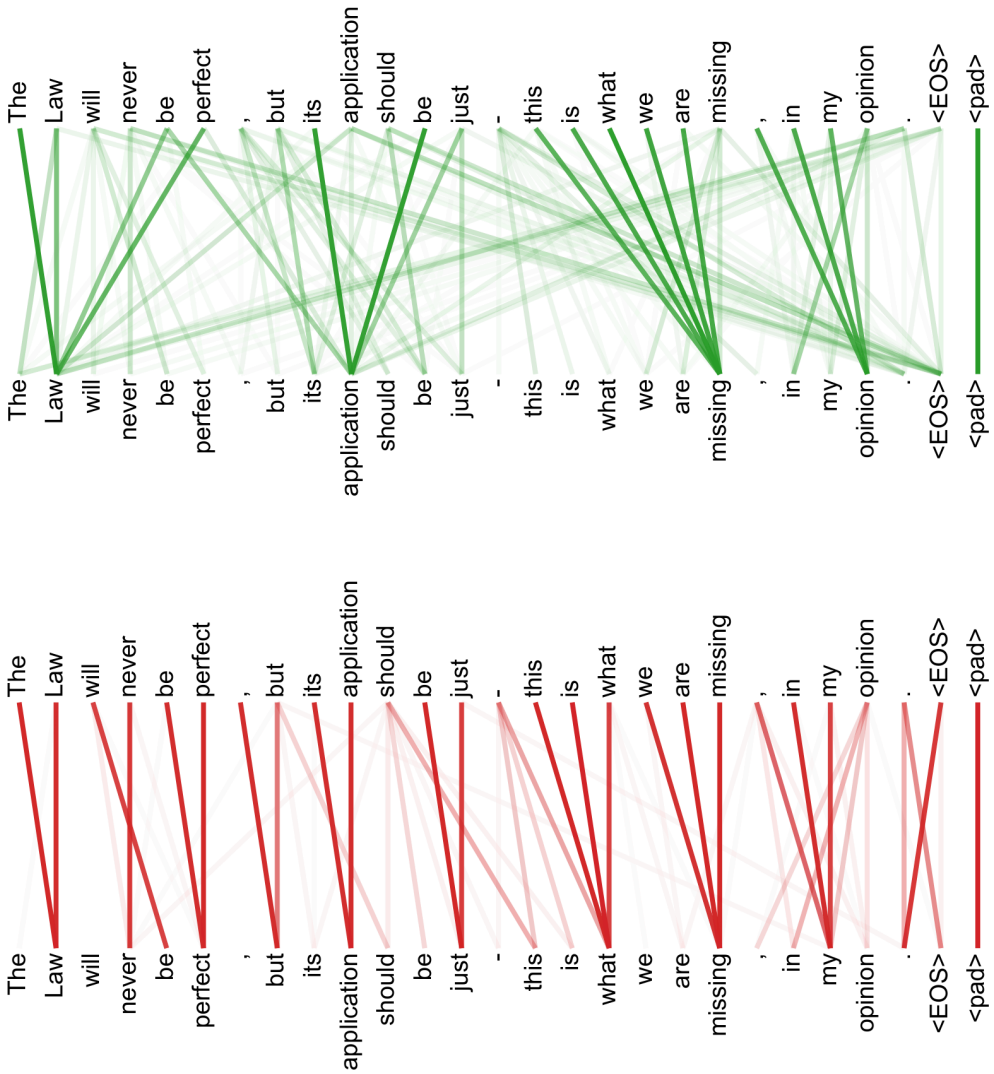


图 5 许多注意力头表现出似乎与句子结构相关的行为。上面给出两个这样的例子，来自编码器第 5 层（共 6 层）自注意力的两个不同的头；这些头显然学会了执行不同的任务。（图中标签为原图英文）

5 训练

本节描述我们模型的训练方案。

5.1 训练数据与分批

我们在标准的 WMT 2014 英德数据集上训练，该数据集包含约 450 万个句子对。句子采用字节对编码[3]进行编码，其源端与目标端共享约 37000 个词元的词表。对于英法，我们使用大得多的 WMT 2014 英法数据集，包含 3600 万个句子，并将词元切分为 32000 个词块的词表[38]。句子对按大致的序列长度分批，每个训练批次包含一组句子对，约含 25000 个源词元和 25000 个目标词元。

5.2 硬件与训练计划

我们在一台配备 8 块 NVIDIA P100 GPU 的机器上训练模型。对于采用全文所述超参数的 base 模型，每个训练步骤约需 0.4 秒。base 模型总共训练 100,000 步，即 12 小时。对于 big 模型（见表 3 最后一行），每步时间为 1.0 秒，共训练 300,000 步（3.5 天）。

5.3 优化器

我们使用 Adam 优化器[20]，其中 $\beta_1 = 0.9$、$\beta_2 = 0.98$、$\epsilon = 10^{-9}$。我们在训练过程中按下式调整学习率：

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

这对应于：在前 warmup_steps 个训练步骤中线性提高学习率，此后按步骤数的平方根反比降低学习率。我们取 warmup_steps = 4000。

5.4 正则化

我们在训练中采用三种正则化手段：

表 2 在 newstest2014 的英译德与英译法测试上，Transformer 以仅为一小部分的训练成本取得了比以往最先进模型更好的 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 + 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 模型）	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer（big）	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

残差 dropout：在每个子层的输出被加到子层输入并归一化之前，我们对其施加 dropout[33]。此外，我们对编码器和解码器堆叠中嵌入与位置编码之和施加 dropout。对于 base 模型，我们使用 $P_{\text{drop}} = 0.1$ 的比率。

标签平滑：在训练期间，我们采用取值 $\epsilon_{\text{ls}} = 0.1$ 的标签平滑[36]。这会损害困惑度，因为模型学会变得更加不确定，但它会提升准确率和 BLEU 分数。

6 结果

6.1 机器翻译

在 WMT 2014 英译德任务上，big Transformer 模型（表 2 中的 Transformer (big)）比包括集成模型在内的以往最佳模型高出 2.0 BLEU 以上，取得了 28.4 BLEU 的新的最先进成绩。该模型的配置列于表 3 最后一行，在 8 块 P100 GPU 上训练共 3.5 天。即便是我们的 base 模型，也以仅为任何竞争模型一小部分的训练成本，超越了所有先前发表的模型与集成模型。

在 WMT 2014 英译法任务上，我们的 big 模型取得了 41.0 BLEU，以不到先前最先进模型 1/4 的训练成本，超越了所有先前发表的单模型。为英译法训练的 Transformer（big）模型使用 $P_{\text{drop}} = 0.1$ 的 dropout 比率，而非 0.3。

对于 base 模型，我们使用单个模型，该模型由最后 5 个检查点（每隔 10 分钟写入一次）平均得到；对于 big 模型，我们平均最后 20 个检查点。我们采用束宽为 4、长度惩罚 $\alpha = 0.6$ 的束搜索[38]。这些超参数是在开发集上实验后选定的。我们将推理期间的最大输出长度设为输入长度 + 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与训练成本同文献中的其他模型架构进行比较。我们通过将训练时间、所用 GPU 数量以及每块 GPU 持续单精度浮点算力的估计值三者相乘，来估算训练一个模型所用的浮点运算次数[5]。

5我们对 K80、K40、M40 和 P100 分别采用 2.8、3.7、6.0 和 9.5 TFLOPS 的数值。

6.2 模型变体

为评估 Transformer 不同组成部分的重要性，我们以不同方式改变 base 模型，在开发集 newstest2013 上测量英译德翻译性能的变化。我们使用上一节所述的束搜索，但不做检查点平均。结果在表 3 中给出。

表 3 Transformer 架构的变体。未列出的取值与 base 模型相同。所有指标均在英译德翻译开发集 newstest2013 上；所列困惑度按我们的字节对编码、以词块为单位，不应与以词为单位的困惑度相比较。

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	训练步数	PPL (dev)	BLEU (dev)	参数量 $\times 10^6$
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

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	训练步数	PPL (dev)	BLEU (dev)	参数量 $\times 10^6$
(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)	以位置嵌入代替正弦位置编码									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

在表 3 的 (A) 行中，如第 3.2.2 节所述，我们在保持计算量不变的情况下，改变注意力头的数量以及注意力键和值的维度。单头注意力比最佳设置差 0.9 BLEU，而头数过多时质量也会下降。

在表 3 的 (B) 行中，我们观察到减小注意力键的大小 d_k 会损害模型质量。这表明确定兼容性并不容易，且比点积更复杂的兼容性函数或许是有益的。我们在 (C) 和 (D) 行进一步观察到，正如预期，更大的模型更好，而 dropout 对避免过拟合非常有帮助。在 (E) 行，我们用学习得到的位置嵌入[9]取代正弦位置编码，观察到与 base 模型几乎相同的结果。

6.3 英文成分句法分析

为评估 Transformer 能否泛化到其他任务，我们在英文成分句法分析上进行了实验。该任务呈现出一些特殊挑战：输出受到很强的结构约束，且明显比输入更长。此外，RNN 序列到序列模型一直未能在小数据情形下取得最先进的结果[37]。

我们在 Penn Treebank[25] 的《华尔街日报》（WSJ）部分上训练了一个 4 层、$d_{\text{model}} = 1024$ 的 Transformer，约含 4 万个训练句子。我们还在半监督设置下训练它，使用约含 1700 万个句子的、规模更大的高置信度语料与 BerkeleyParser 语料[37]。仅 WSJ 设置下我们使用 16K 词元的词表，半监督设置下使用 32K 词元的词表。

我们只进行了少量实验，在第 22 节开发集上选择 dropout（注意力 dropout 与残差 dropout，见 5.4 节）、学习率和束宽，其他所有参数都与英译德 base 翻译模型保持不变。在推理期间，我们将最大输出长度提高到输入长度 + 300；仅 WSJ 与半监督这两种设置都使用束宽 21 和 $\alpha = 0.3$。

表 4 Transformer 能很好地泛化到英文成分句法分析（结果在 WSJ 第 23 节上）。

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	仅 WSJ，判别式	88.3
Petrov et al. (2006) [29]	仅 WSJ，判别式	90.4
Zhu et al. (2013) [40]	仅 WSJ，判别式	90.4
Dyer et al. (2016) [8]	仅 WSJ，判别式	91.7
Transformer（4 层）	**仅 WSJ，判别式**	**91.3**
Zhu et al. (2013) [40]	半监督	91.3
Huang & Harper (2009) [14]	半监督	91.3
McClosky et al. (2006) [26]	半监督	92.1
Vinyals & Kaiser el al. (2014) [37]	半监督	92.1
Transformer（4 层）	**半监督**	**92.7**
Luong et al. (2015) [23]	多任务	93.0
Dyer et al. (2016) [8]	生成式	93.3

我们在表 4 中的结果表明，尽管缺乏针对任务的调优，本模型的表现却出奇地好，除循环神经网络语法（Recurrent Neural Network Grammar）[8]之外，比所有先前报告的模型都取得了更好的结果。

与 RNN 序列到序列模型[37]不同，即便仅在含 4 万个句子的 WSJ 训练集上训练，Transformer 也优于 BerkeleyParser[29]。

7 结论

在本工作中，我们提出了 Transformer，这是第一个完全基于注意力的序列转导模型，用多头自注意力取代了编码器-解码器架构中最常用的循环层。

对于翻译任务，Transformer 的训练速度能够显著快于基于循环或卷积层的架构。在 WMT 2014 英译德和 WMT 2014 英译法两个翻译任务上，我们都取得了新的最先进成绩。在前一个任务上，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感到兴奋，并计划将它们应用到其他任务。我们计划把 Transformer 扩展到文本之外、涉及其他输入与输出模态的问题，并研究局部的、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与输出。让生成过程不那么顺序化，是我们的另一个研究目标。

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 提供的富有成效的意见、更正与启发。

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.

[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 V 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.