
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

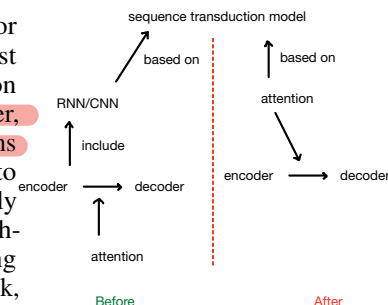
Aidan N. Gomez*[†]
University of Toronto
aidan@cs.toronto.edu

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



1 Introduction

1.1 RNN, LSTM, GRNN

Recurrent neural networks, long short-term memory [12] 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 [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

*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.

- 1.2 RNN劣势
1. 并行性差
 2. 在长序列情况下丢信息

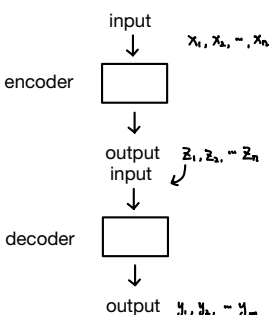
- 1.3 attention in RNN
- 主要用于如何将encoder的内容高效地传递给decoder

- 2.1 针对RNN并行性差
- 使用CNN替换RNN，利用其有多个input和output的通道，减少时序计算

- 2.2 针对RNN长序列丢信息
- 但CNN也存在对长序列input难以建模的缺陷

- 2.3
- 而transformer可以解决这个长序列问题，除此之外，通过多头注意力机制（multi-headed attention）实现多通道的output以解决effective resolution减少的问题

3.0.1



注意

1. input=n, output=m, 两者不一定相等
2. y是一个一个生成的，就像翻译的时候结果一个字一个字往外蹦

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 [18] and conditional computation [26], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

$$h_t = h_{t-1} + \tau$$

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, 16]. In all but a few cases [22], 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 [20], ByteNet [15] and ConvS2S [8], 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 [11]. 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, 22, 23, 19].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [28].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [14, 15] and [8].

3 Model Architecture an encoder-decoder structure + 自回归 (auto-regressive)

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 29]. Here, the encoder maps an input sequence of symbol representations (x_1, \dots, x_n) to a sequence of continuous representations $\mathbf{z} = (z_1, \dots, z_n)$. Given \mathbf{z} , the decoder then generates an output sequence (y_1, \dots, y_m) of symbols one element at a time. At each step the model is auto-regressive [9], consuming the previously generated symbols as additional input when generating the next.

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-

3.0.2 自回归

模型使用自身的过去值作为预测未来值的证据

$$x_1, x_2, \dots, x_{t-1}$$

3.1.1 encoder架构

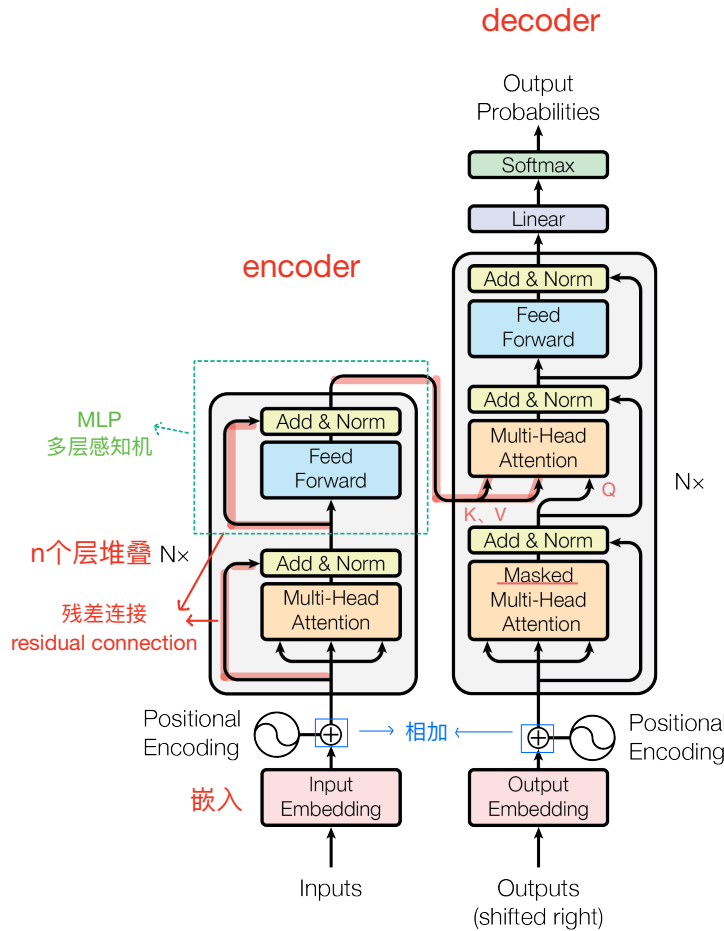
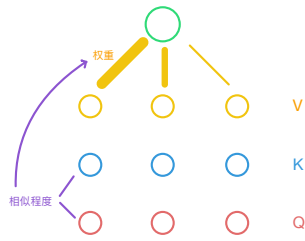


Figure 1: The Transformer - model architecture.

MLP
多层感知机

wise fully connected feed-forward network. We employ a residual connection [10] around each of the two sub-layers, followed by layer normalization [11]. (That is, the output of each sub-layer is $\text{LayerNorm}(x + \text{Sublayer}(x))$, where $\text{Sublayer}(x)$ is the function implemented by the sub-layer itself.) To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

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

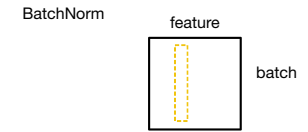
3.2.1 Scaled Dot-Product Attention

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

3.1.2 补充内容

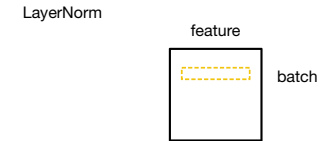
layer normalization VS batch normalization

情况一：当输入是二维时（最简单的情况）



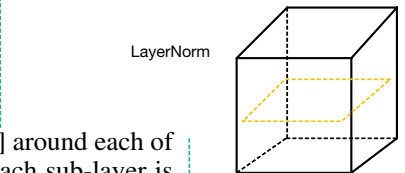
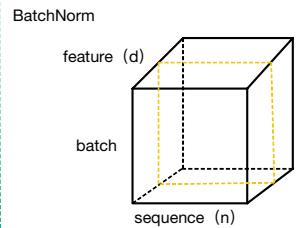
在训练时，每次取一列（特征），使其均值=0，方差=1
[how: (矩阵-均值)/方差]

在训练时，可做小批量均值和方差的计算；
在预测时，则对全局进行计算



在训练时，每次取一行（样本），使其均值=0，方差=1

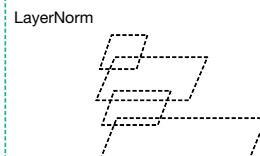
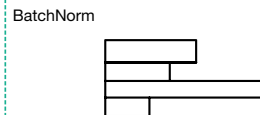
情况二：当输入是三维时



为什么LayerNorm用得更多？
因为在时序的序列模型中，样本长度会发生变化，造成以下问题：

问题一：当样本长度比较大时，每次做小批量时，BatchNorm方案均值与方差的抖动相对较大，而LayerNorm方案只在单个样本内计算均值与方差，不需要存储全局的均值方差，所以比较稳定

问题二：当碰到新的特别长的预测样本时，会导致BatchNorm方案中之前训练计算的全局的均值和方差没那么好用



补充内容

3.2.1.3 mask

将t+1及之后的时刻的key换成极小数，如-1e10，那么通过softmax之后，其值降为0，意味着最后得到的权重weight也为0

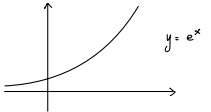
3.2.1.4 softmax

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}$$

原理：指数化 + 归一化

特性：

- 可作为概率分布解释
- 梯度相对平滑，有利于训练稳定性



3.2.1.5 内积如何表征相似度

内积的具体方式之一，是通过计算余弦相似度，来评估两个向量的相似度

主流的两种attention机制

（两者大差不差，只是内积用电脑计算更快更方便）

3.2.1.2 why $\frac{1}{\sqrt{d_k}}$

当dk不是很大时，缩放与否都行；当dk很大时，两个向量的内积可能会很大，值与值之间的相对距离可能会比较大，在经过softmax之后会更加接近0和1的两端，此时计算机梯度比较小，不利于收敛

3.2.2.1 整体流程

详见figure 2

3.2.2.2 Why多头注意力机制

根源因为embedding包含很多逻辑，如

- 语义逻辑(dog, cat, rabbit)
- 语法逻辑(walking, walked和swimming, swim)
- 上下文逻辑(cat和tree)

拆分head做attention，是在细分出不同的语义子空间，即不同类型的细分语义逻辑，使attention机制运行起来更细腻精准，更有针对性

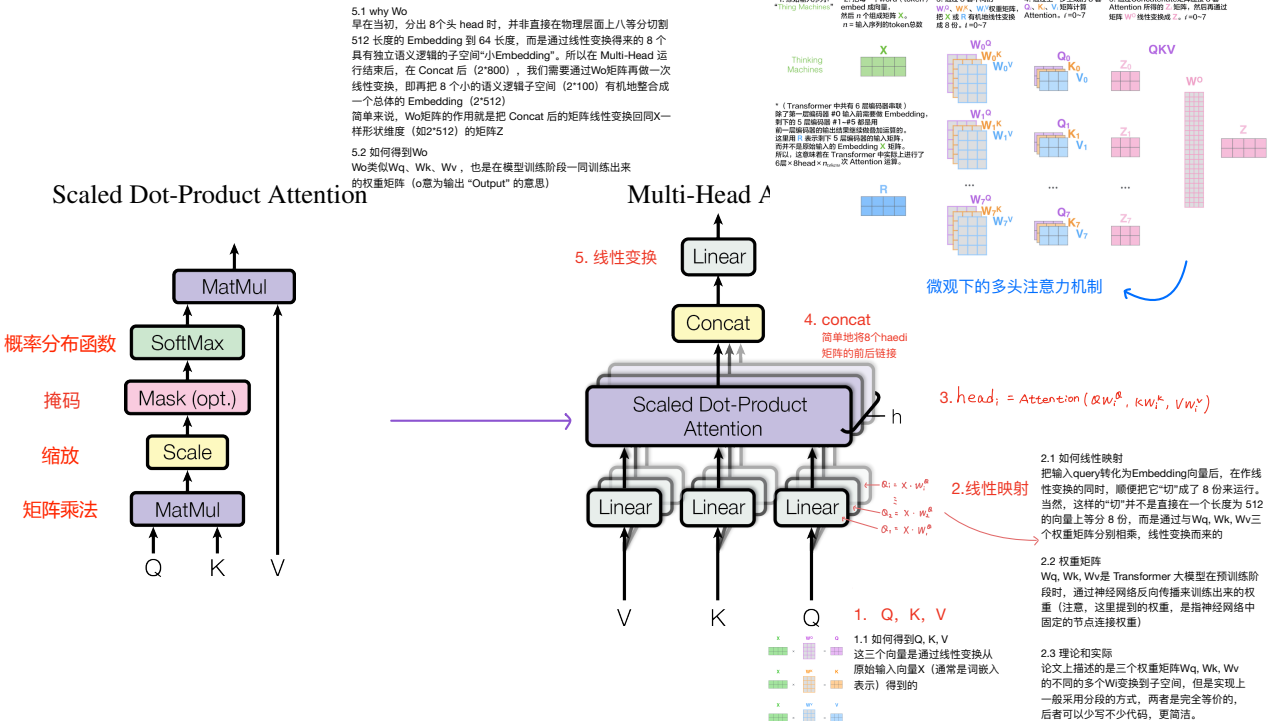


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

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:

3.2.1.1 计算方法

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

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

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

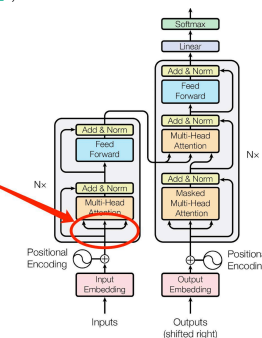
Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

⁴To 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 .

注1: \in 为“属于”符号, \mathbb{R} 为“实数集”, d_{model} 为整个 Transformer 模型中所有子层和 Embedding 层的输出维度, $d_{model} = 512$, $d_k = d_v = d_{model}/h = 64$, d_k 与 d_v 在模型的执行阶段不用必须相等, 可以取不同的值, h 为 head 数, h 默认为 8。

注2: 以上公式有一处需要解释一下, 针对 $head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ 这个公式很多朋友都问到了一个问题, 那就是为什么用 Q 、 K 、 V 而不是用 X 来分别与 W_i^Q 、 W_i^K 、 W_i^V 相乘做线性变换, Q 、 K 、 V 应该是 X 分别与 W_i^Q 、 W_i^K 、 W_i^V 做线性变换所得到的结果啊, 这里会不会是原文的笔误。解释: 在 Multi-Head Attention 中, 这个公式里的 Q 、 K 、 V 实际上是一种名义上的称呼, 是说他们要执行各自对应的使命, 而本质上他们都是 X 本身。注意论文中的这个图 (下图) 中红框圈出的部分, 进程走到这里简单地分成了四个分支, 其中1支直接去往后端的 Add & Norm 了, 另外3支输入到 Multi-Head Attention 中, 这3支就是 Q 、 K 、 V 。如果这里需要线性变换的话, 图中会给出对应的 Linear 变换标注的, 而实际上并没有。可以将这个公式里的 Q 、 K 、 V 等理解为输入矩阵 X 即可, 即

$head_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V)$



3.2.2.3 公式

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

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

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

3.2.2.4 h = 8

why h=8
根据实验结果的综合评分 (PPL、BLEU、参数规模、d-model等) 得出当 h=8 时效果最好

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{model}/h = 64$. 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 [31] [2] [8].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- 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

有效地将编码器 (encoder) 中的输出, 也就是key和value, 根据我想要的东西 (query) 给他拎出来

Q = K = V = X

如何掩码 (详见3.2.1.3)

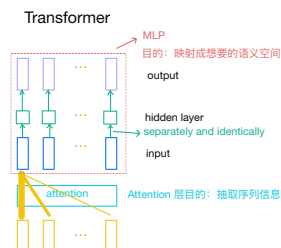
第三个自注意力层

第一个自注意力层

第二个自注意力层

Masked Multi-Head Attention

3.3.1



3.3 Position-wise Feed-Forward Networks 即单隐藏层的MLP

3.3 补充内容

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. position-wise, 即把一个MLP对每一个词作用一次

MLP是多层神经网络的一种, 具体来说: MLP是一种特殊的多层神经网络, 它由全连接层组成; 而多层神经网络中, 除了全连接层外, 还可以包含卷积层、池化层等其他类型的层

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

隐藏层中, w1把512维投影成2048维, 因为最后有残差连接, 所以w2将2048维再次投影为512维, 使得output为512维

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 $d_{model} = 512$, 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 [24]. In the embedding layers, we multiply those weights by $\sqrt{d_{model}}$.

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

将权重矩阵乘以 $\sqrt{d_{model}}$

Why: 在学习embedding时, 多多少少会把每一个向量的L2norm (欧式距离) 学成相对比较小的结果, 如1, 也就是说当矩阵维度变大的时候, 学习的一些权重值就会变小, 为了后续加上positional encoding时, 两者scale差不多, 所以乘以 $\sqrt{d_{model}}$

Transformer VS RNN

之如何有效地使用序列信息

相同的是, 都通过MLP做语义空间的转换; 不同的是如何传递序列信息: RNN是将 t-1 时刻的output, 和 t 时刻的input 一起组成新的input作为输入; Transformer则是通过attention层抽取序列信息, 然后通过MLP做语义转换

3.5.1 Why attention没有时序信息, 即一句话 (query) 顺序打乱之后, attention结果都是一样的 (都是value的加权和), 这是不合理的

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

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 [8].

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 [8] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

经实验发现，positional encoding的计算公式是固定的还是可被学习的，对结果影响不大
使用正弦函数的版本是因为它可以允许模型推理到比训练期间遇到的序列长度更长的序列

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, \dots, x_n) to another sequence of equal length (z_1, \dots, 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 [11]. 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 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 [31] and byte-pair [25] 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

3.5.2 How

- RNN如何解决时序性问题：t时刻的input = t-1时刻的输出 + t时刻的input
- Transformer如何解决时序性问题：用一个512维的向量来表示一个数字，一个位置（即时序信息），然后与embedding相加，即完成了在input中加入时序信息

3.5.3 如何生成位置向量

4.1 比较维度

- 计算复杂度
- 并行计算量
- 路径长度（网络中长距离依赖的之间的路径长度）

4.2 补充内容

在实践中，transformer对整个模型的假设更多，导致需要更多数据和更大模型才能训练出和RNN，CNN同样的效果，所以导致基于transformer的模型比较贵

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 [15], increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of k . Separable convolutions [6], however, decrease the complexity considerably, to $O(k \cdot n \cdot d + n \cdot d^2)$. 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

批处理 (Batch Processing) 是指将训练数据分成多个较小的批次 (batches) 进行处理的过程

提取词根, 以应对词的不同形态的变化, 如 +es, +ing, +ed, 使得整个字典比较小

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [3], which has a shared source-target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [31]. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

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

学习率的计算

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

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 正则化

We employ three types of regularization during training:

Residual Dropout We apply dropout [27] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of $P_{drop} = 0.1$.

在标准的dropout中, 我们随机丢弃 (置零) 网络中的一些神经元, 以减少过拟合。然而, 在残差网络中, 简单地每个残差块中应用dropout可能会导致信息流受阻, 因为dropout可能会阻断从输入直接流向输出的路径。为了解决这个问题, 残差dropout被提出, 它只在残差连接的分支上应用dropout, 而不是在恒等连接 (Identity Connection) 上应用

1.核心思想
将传统的one-hot编码硬标签 (hard labels) 转换为软标签 (soft labels), 以减少模型在训练过程中对标签的过度自信, 从而提高模型的泛化能力

2.具体原理
在标签平滑中, 对于一个n类分类问题, 如果真实标签是第i类, 那么在one-hot编码中, 第i类的位置会被标记为1, 其余位置为0。标签平滑会将这个硬标签转换为一个更加平滑的概率分布, 具体做法是将第i类的概率从1降低到1-e (e为平滑因子, 通常设置为一个较小的值, 如0.1), 然后将剩余的e概率均匀地分配给其他n-1个类别

3.作用机制
标签平滑通过减少模型对标签的绝对信心来工作, 避免模型在训练过程中过于自信地预测标签。这样做可以减轻模型对训练数据细节的过度依赖, 提升模型的泛化能力

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 [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.0	$2.3 \cdot 10^{19}$	

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

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 $\alpha = 0.6$ [31]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [31].

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 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.

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

6.2.3 补充内容之ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

1.ROUGE与BLEU
两者类似, ROUGE基于召回率 (recall), 而BLEU基于准确率 (precision); ROUGE往往用于评估文本摘要任务, 而BLEU用于评估机器翻译

2.常见ROUGE
• ROUGE-N: 在 N-gram 上计算召回率
• ROUGE-L: 考虑了机器译文和参考译文之间的最长公共子序列
• ROUGE-W: 改进了 ROUGE-L, 用加权的方法计算最长公共子序列
• ROUGE-S: 基于Skip-bigram的指标, 允许在匹配bigram时跳过一个单词

3.ROUGE-N计算方式

$$ROUGE-N = \frac{\sum_{s \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count_{max}(gram_n)}{\sum_{s \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

其中, 分母表示在参考译文中 N-gram 的个数, 分子表示参考译文与机器译文共有的 N-gram 个数

6.2.4 补充内容之 precision, recall

1.定义
• 准确度 (precision)
• 召回率 (recall): 漏网之鱼敏感度

2.计算方式

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN}$$

1.定义

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.

困惑度是衡量语言模型性能的一种指标，它反映了模型对文本的预测能力。困惑度越低，表示模型对文本的理解和预测能力越强。

2. 计算方式

对于一个给定的文本序列，困惑度是通过计算模型对每个词的概率的负对数平均值来得到的

$$PPL = \exp \left(-\frac{1}{N} \sum \log p(w_i | w_{1:i-1}) \right)$$

其中, N 是文本序列的长度,
 W_i 是序列中的第 i 个词, $p(w_i | w_{1:i-1})$
是给定前面词的情况下第 i 个
词的概率。

$$\text{BLUE} = \text{BP} \cdot \exp\left(\frac{1}{N} \sum_{i=1}^N \log p_i\right)$$

其中, BP为简短惩罚因子

$$BP = \begin{cases} 1 & c > \text{reference} \\ e^{-(1-\frac{r}{c})} & c \leq r \end{cases}$$

P_i 为 modified precision

$$P_n = \frac{\sum_i \sum_k \min \{ h_k(C_i), \max h_k(S_j) \}}{\sum_i \sum_k \min (h_k(C_i))}$$

$h_k(c_i)$ 为机器翻译中n-gram出现的次数

$h_k(s_{ij})$ 为标准答案中n-gram出现的次数

min的意义为排除机器翻译内容为
the the the the the the的特殊情况

可调超参	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

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.

