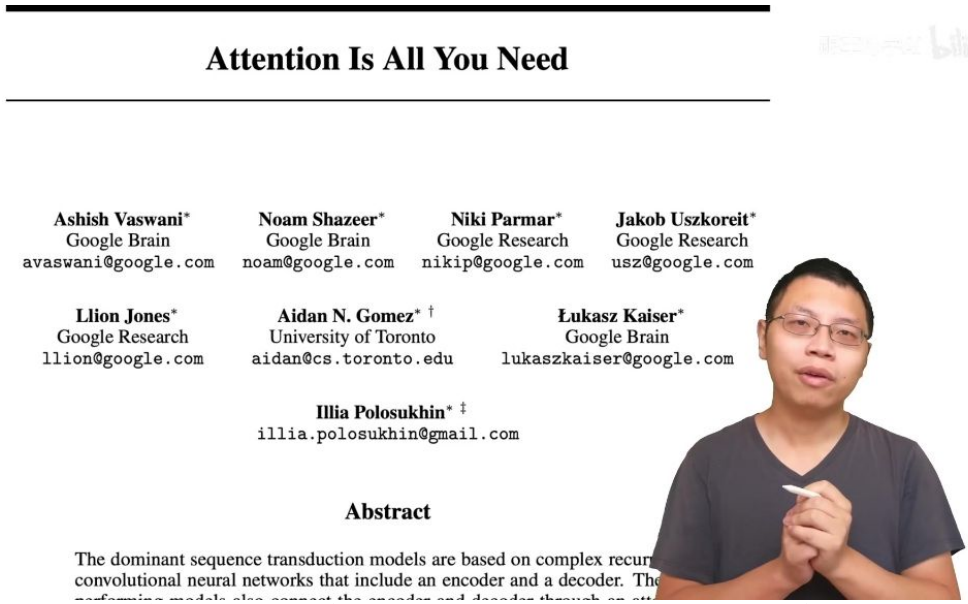
一、作者介绍 01:07



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 recur
convolutional neural networks that include an encoder and a decoder. Th

- **作者构成**: 共8位作者，其中6位来自Google Research/Google Brain，2位来自多伦多大学（标注为实习期间完成工作）
- **署名特点**: 每位作者姓名后均标注星号(*)，表示同等贡献
- **行业惯例**:
  - 机器学习领域通常按贡献大小排序作者
  - 第一作者通常贡献80%工作
  - 前2-3位同等贡献较常见，但8位全部同等贡献非常罕见

二、同样贡献 01:45

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.

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 an

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and sta
the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models
has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-h
attention and the parameter-free position representation and became the other person involved in nearly eve
detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase an
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
implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively
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,

- **特殊标注**: 论文末尾注明"*Equal contribution. Listing order is random."
- **贡献分配**:
  - Jakob: 提出用self-attention替代RNN的核心理念
  - Ashish和Illia: 设计并实现首个Transformer模型
  - Noam: 提出scaled dot-product attention、multi-head attention和无参数位置编码
  - Niki: 完成大量模型变体的设计实现与调优

- Llion: 负责初始代码库、高效推理和可视化
- Lukasz和Aidan: 重构tensor2tensor代码库显著提升性能

- **学术规范意义**:
  - 明确记录每位作者的具体贡献
  - 避免随意挂名，确保署名与实质贡献匹配
  - 为多作者论文的贡献说明提供示范

三、摘要 03:22

1. 主流序列转录模型 03:24

- **基本概念**: 序列转录模型指输入一个序列生成另一个序列的模型，如机器翻译中输入英文输出中文
- **传统架构**: 基于复杂的循环神经网络(RNN)或卷积神经网络(CNN)，采用encoder-decoder结构
- **性能优化**: 最佳模型在encoder和decoder之间加入了注意力机制(attention mechanism)

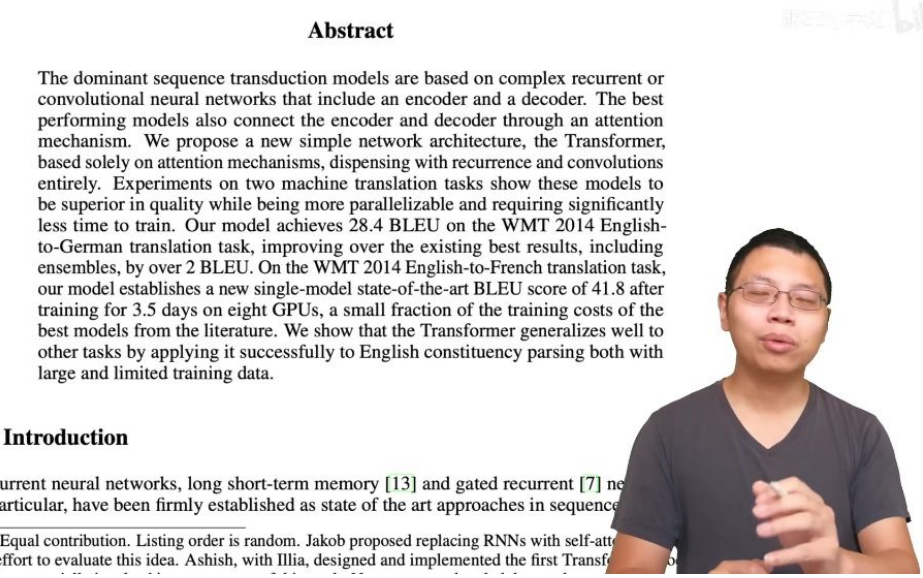
2. Transformer模型介绍 04:54

- **架构创新**: 完全基于注意力机制，摒弃了传统的循环和卷积结构
- **命名特点**: "Transformer"中文译为"变形金刚"，采用大众熟知名词便于记忆传播
- **研究趋势**: 简单架构(simple)成为褒义词，只要效果好（如ResNet的成功案例）

3. Transformer模型优势 06:16

- **训练效率**: 显著提高并行性，训练时间大幅缩短（8个GPU上仅需3.5天）
- **性能表现**:
  - WMT2014英德翻译：28.4 BLEU，超越之前最佳结果2个BLEU
  - WMT2014英法翻译：41.8 BLEU，创下单模型新纪录
- **计算成本**: 训练开销仅为文献中最佳模型的一小部分

4. Transformer模型应用泛化能力 07:07



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.

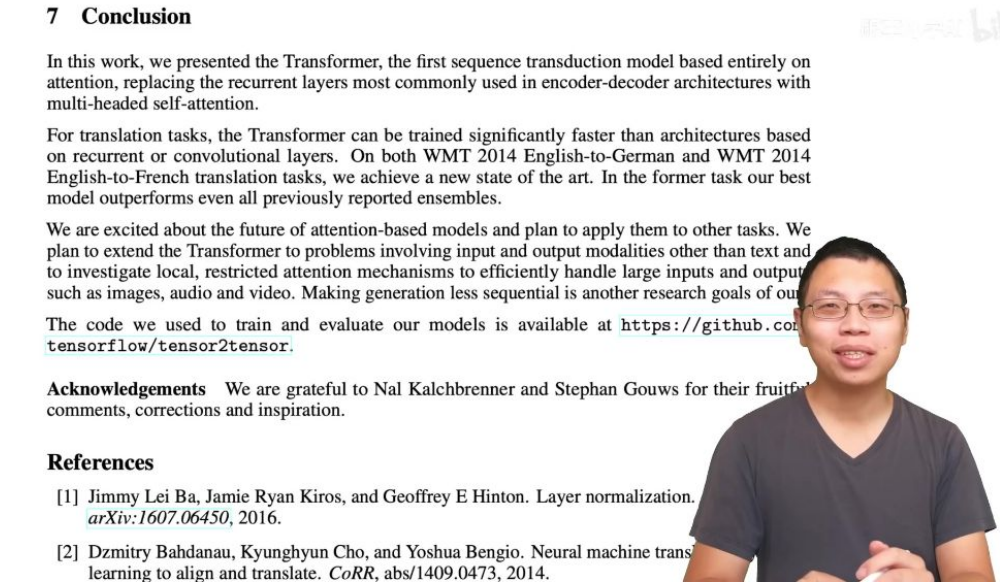
1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] ne
in particular, have been firmly established as state of the art approaches in sequence

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-att
the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transf

- **初始应用**: 专注于机器翻译领域（相对小众的应用场景）
- **扩展应用**: 成功应用于英语选区解析任务，无论训练数据量大小
- **后续发展**: 被BERT、GPT等模型扩展到更广泛的NLP任务，最终"出圈"应用于图像、视频等多模态领域

5. Transformer模型结论与未来方向 08:15



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 our

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

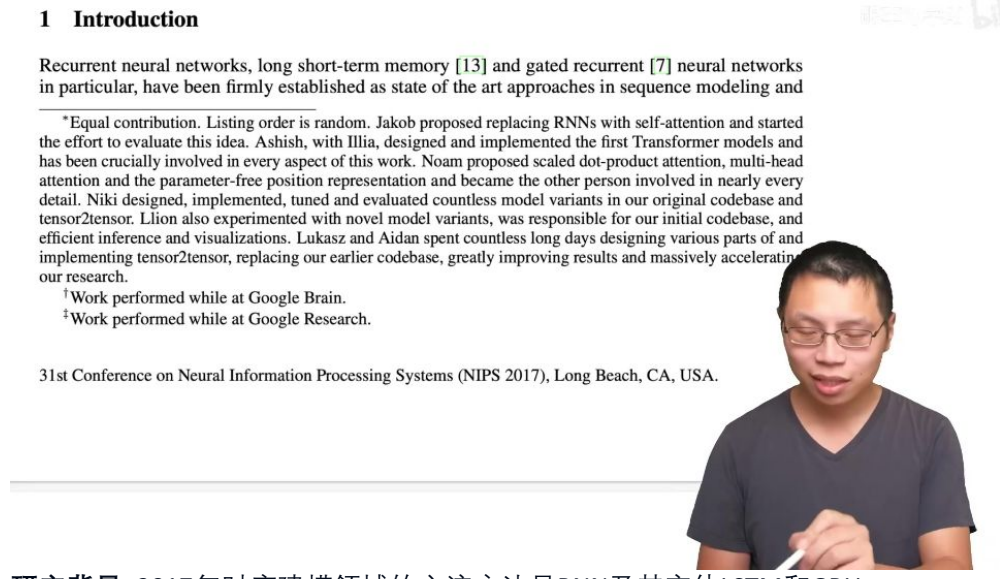
[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

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

- **技术突破**：首次完全基于注意力的序列转录模型，用multi-headed self-attention取代循环层
- **实践验证**: 在翻译任务上训练速度更快，质量更高
- **未来方向**:
  - 扩展到文本以外的模态（图像、音频、视频）
  - 研究局部受限注意力机制处理大规模输入输出
  - 减少生成过程的序列依赖性
- **开源支持**: 代码公开在tensor2tensor库，促进研究复现和应用扩展

四、导言 10:05

1. 导言概述 10:08



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

*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.

- **研究背景**: 2017年时序建模领域的主流方法是RNN及其变体LSTM和GRU
- **写作特点**: 导言部分较短，可视为摘要前半部分的扩展版本

2. 时序模型中的主流模型 10:34

- **两大架构**:
  - **语言模型**: 主要用于序列生成任务

- **编码器-解码器**: 适用于输出结构化信息较多的场景

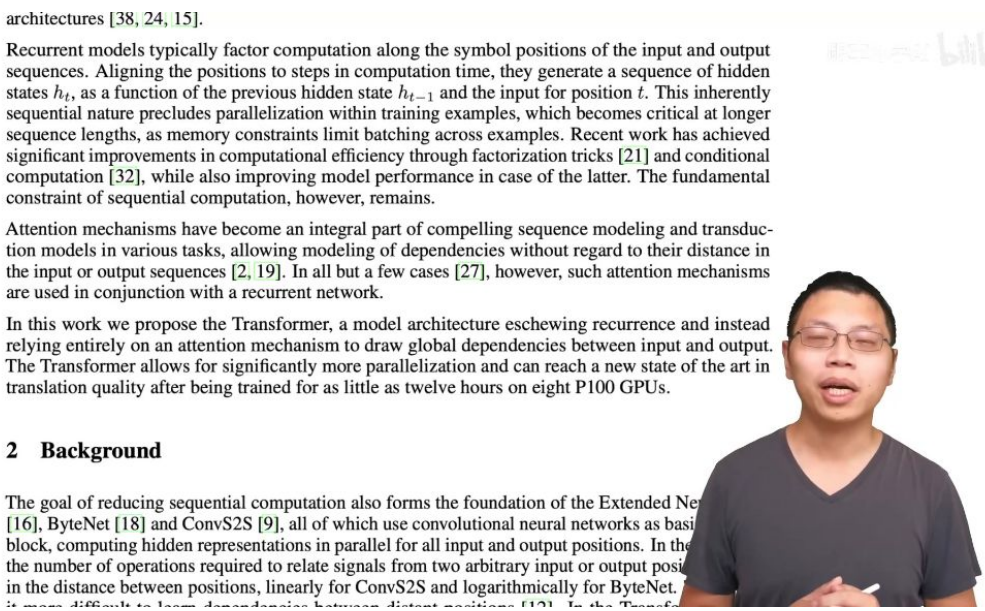
3. RNN的特点与缺点 10:45

- **计算机制**:
  - 序列从左到右逐步计算，每个时间步t生成隐藏状态h_t
  - h_t由前一个隐藏状态h_{t-1}和当前输入共同决定：$h_t = f(h_{t-1}, x_t)$
- **核心优势**:
  - 通过隐藏状态传递历史信息，有效处理时序数据
- **主要缺陷**:
  - **并行性差**: 必须串行计算，100个词的句子需要100步计算
  - **长程依赖**: 早期时序信息在长序列中容易丢失
  - **内存开销**: 增大h_t维度可缓解信息丢失但增加存储压力
- **改进尝试**:
  - 通过分解技巧和条件计算提升效率
  - 但序列计算的本质限制仍未解决

4. Attention在RNN上的应用 13:00

- **应用场景**:
  - 主要用于编码器-解码器架构中的信息传递
  - 与RNN配合使用，增强远距离依赖建模能力
- **功能特点**:
  - 可建模任意距离的依赖关系

5. Transformer模型概述 13:20



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 Ne
[16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basi
block, computing hidden representations in parallel for all input and output positions. In th
the number of operations required to relate signals from two arbitrary input or output posi
in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet.

- **架构创新**:
  - 完全摒弃循环结构，纯基于注意力机制
  - 通过自注意力建立输入输出的全局依赖
- **性能优势**:
  - 支持高度并行计算
  - 在8块P100 GPU上训练12小时即可达到SOTA翻译质量

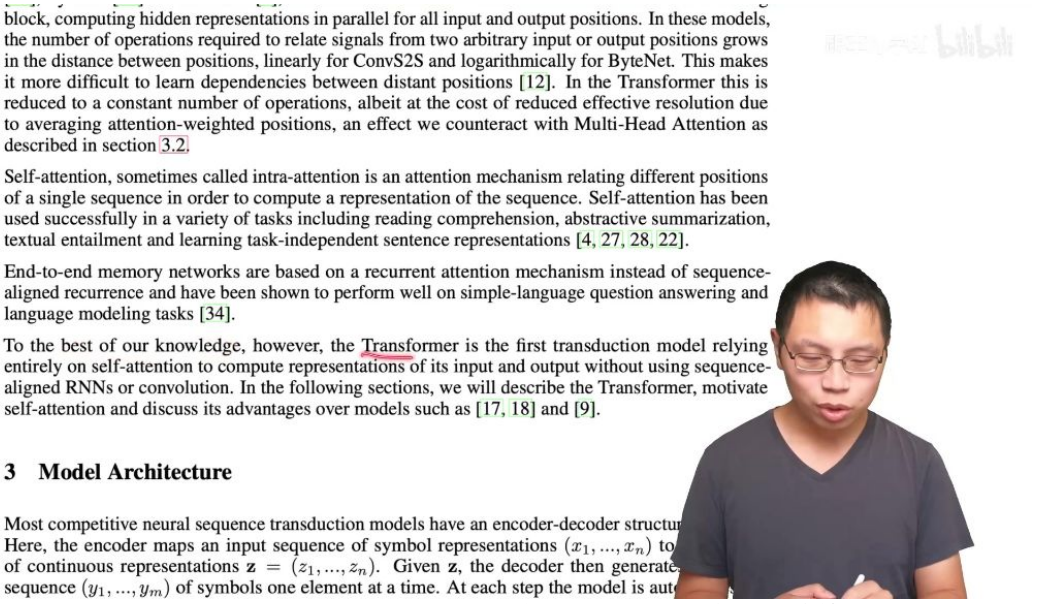
6. 导言总结 13:59

- **写作特点**:
  - 篇幅精简，突出核心创新点

- 受NIPS会议8页篇幅限制影响
- **研究意义**:
  - 提出全新网络架构，包含多项创新设计

五、相关工作 14:36

- **时序计算替代方案**：Extended Neural GPU、ByteNet和ConvS2S使用卷积神经网络替代循环神经网络，实现并行计算所有输入输出位置的隐藏表示。
- **远程依赖问题**：在CNN模型中，两个任意位置信号关联所需操作数随距离增长（ConvS2S线性增长，ByteNet对数增长），导致远距离位置依赖关系难以学习。
- **Transformer改进**：将操作数减少为常数级别，但会因注意力权重位置平均化导致有效分辨率降低，通过多头注意力机制（Multi-Head Attention）进行补偿。


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].

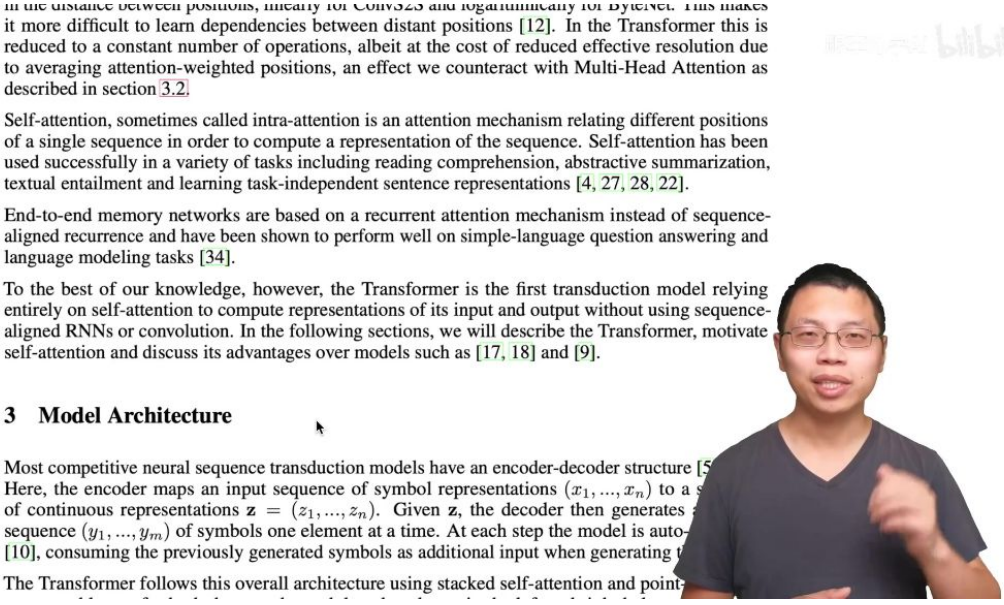
End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

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 structur
Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to
of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generate
sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is aut

- **自注意力机制**：又称内部注意力，通过关联单个序列的不同位置来计算序列表示，已成功应用于阅读理解、抽象摘要、文本蕴含和句子表示学习等任务。
- **多头注意力优势**：模拟CNN的多输出通道效果，通过不同注意力头捕获多样化模式。
- **记忆网络对比**：端到端记忆网络基于循环注意力机制，在简单语言问答和语言建模任务表现良好，但与Transformer的纯自注意力架构有本质区别。


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
Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a s
of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates
sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-
[10], consuming the previously generated symbols as additional input when generating t

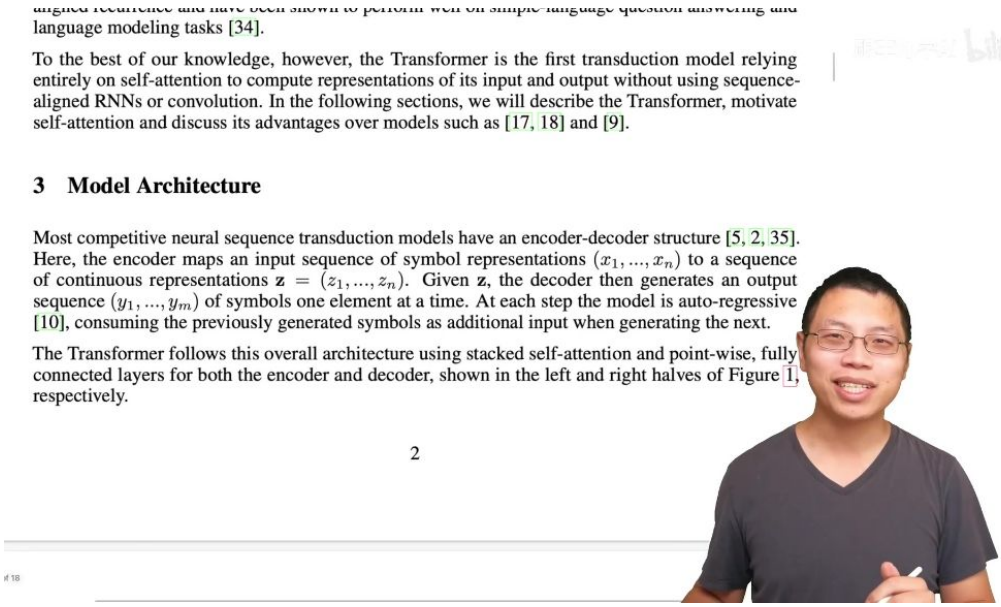
- The Transformer follows this overall architecture using stacked self-attention and point-

- **架构创新**：首个完全依赖自注意力计算输入输出表示的编解码模型，无需序列对齐的RNN或卷积操作。
- **关键区别**：
  - 与CNN模型相比：突破位置距离对计算复杂度的限制
  - 与RNN模型相比：避免序列顺序计算的局限性
  - 与记忆网络相比：不依赖循环注意力机制

六、模型架构 16:35

1. 编码器和解码器堆栈 16:44

1）编码器 22:39



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].

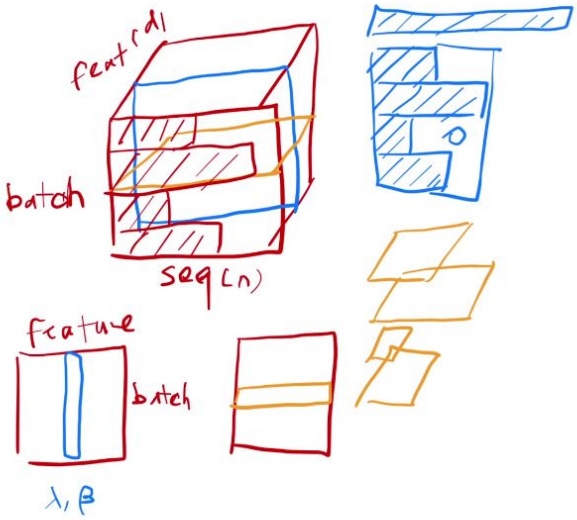
3 Model Architecture

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.

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.

2

- **核心结构**：由$N = 6$个相同层堆叠而成，每层包含两个子层：
  - 多头自注意力机制（Multi-head self-attention）
  - 前馈神经网络（Position-wise fully connected feed-forward network）
- **残差连接**：每个子层采用$x + Sublayer(x)$形式，后接层归一化（LayerNorm）
- **维度统一**：所有子层及嵌入层输出维度固定为$d_{model} = 512$，简化模型设计
- **层归一化特点**：
  - 对每个样本独立计算均值和方差，解决序列长度变化问题
  - 相比批归一化（BatchNorm）更稳定，尤其适用于变长序列
  - 计算方式：样本内特征归一化而非特征跨样本归一化



-
- **三维输入处理**：当输入为(batch, sequence, feature)时：
  - BatchNorm：沿(batch, sequence)维度切分计算统计量
  - LayerNorm：沿(feature)维度计算统计量
- **稳定性优势**：序列长度变化时，LayerNorm的统计量计算不受padding影响

2）解码器 32:06

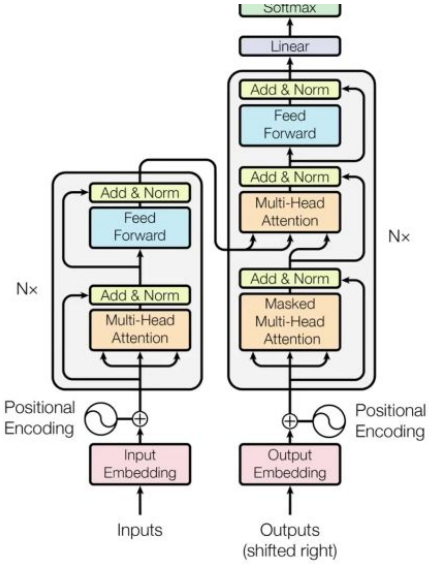


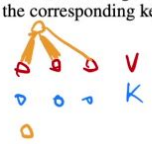
Figure 1: The Transformer - model architecture.

-
- **核心差异**：在编码器两个子层基础上增加第三子层：
  - 编码器-解码器注意力层（Encoder-decoder attention）
- **自回归特性**：
  - 输出序列逐个生成，当前输出依赖先前输出
  - 通过右移（shifted right）实现输出嵌入的位置偏移
- **掩码机制**：
  - 防止解码时访问后续位置信息（防止信息泄露）
  - 保证预测时只能看到当前位置及之前的信息
  - 实现方式：带掩码的多头注意力（Masked Multi-Head Attention）

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.



- **基本概念**：将查询(query)映射到键值对(key-value)的输出
- **计算过程**：
  - 输出是值的加权和：$output = \sum(compatibility(query, key) \cdot value)$
  - 权重由query与key的相似度（兼容函数）决定
- **动态特性**：相同key-value下，不同query会产生不同权重分布和输出

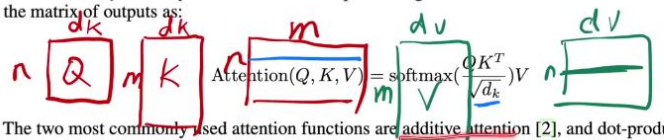
2. 注意力机制 33:30

1）规模化点乘注意力 36:05

- Scaled Dot-Product Attention计算方式

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:



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

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

- **输入组成**：由维度为d_k的queries和keys，以及维度为d_v的values组成
- **计算过程**：
  - **相似度计算**：每个query与所有keys进行点积运算，点积值越大表示向量相似度越高（余弦值越大），为零则表示正交无相似度
  - **缩放处理**：将点积结果除以$\sqrt{d_k}$进行缩放
  - **权重生成**：通过softmax函数将缩放后的值转换为非负且总和为1的权重
  - **输出计算**：将权重作用于values得到最终输出
- **矩阵化实现**：

- **批量处理**：将多个queries打包成矩阵Q（n×d_k），keys和values分别打包为矩阵K（m×d_k）和V（m×d_v）
- **高效计算**：通过两次矩阵乘法（QK^T和权重矩阵×V）实现整个注意力计算，输出为n×d_v矩阵

- Scaled Dot-Product Attention计算示例 37:40
  - **具体示例**：
    - 给定n个queries和m个key-value pairs
    - 每个query与所有keys计算点积得到n×m矩阵
    - 对矩阵每行进行softmax得到注意力权重
    - 权重矩阵与V相乘得到n×d_v输出矩阵
  - **并行优势**：矩阵乘法可高度并行化，特别适合处理序列数据
- 常见的注意力机制：加性注意力机制 39:37

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

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

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

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 regi

  - **主要类型对比**：
    - **加性注意力**：
      - 使用单隐藏层前馈网络计算兼容性函数
      - 可处理query和key不等长的情况
      - 在大d_k值时表现优于未缩放的点积注意力
    - **点积注意力**：
      - 与Scaled Dot-Product基本相同，仅缺少$\frac{1}{\sqrt{d_k}}$缩放因子
      - 计算更快且更节省空间（可利用优化矩阵乘法）
  - **缩放必要性**：
    - 当d_k较大时，点积值会变得很大，导致softmax进入梯度极小区域
    - 通过$\frac{1}{\sqrt{d_k}}$缩放可抵消这种效应
    - 小d_k时两种机制表现相似

2）多头注意力 40:06

- 两种常用的注意力函数

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

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

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

-
- **点积注意力**:
  - **计算方式**: 计算query和key的点积，除以$\frac{1}{\sqrt{d_k}}$后做softmax
  - **优势**: 实现简单高效，只需两次矩阵乘法
  - **公式**:$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$
- **加性注意力**:
  - **计算方式**: 使用带单隐藏层的前馈网络计算兼容性函数
  - **对比**: 理论复杂度相似，但点积注意力实际更快更省空间

- 为什么要缩放点积结果 40:24

query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

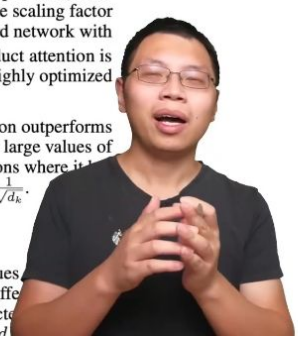
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

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



-
- **小维度情况**: 当d_k较小时，两种注意力表现相似
- **大维度问题**:
  - **现象**:d_k较大时点积结果绝对值增大，使softmax进入梯度极小区域
  - **数学解释**: 假设q和k分量是均值为0、方差为1的独立随机变量，点积q·k的方差为d_k
  - **解决方案**: 缩放因子$\frac{1}{\sqrt{d_k}}$将方差控制为1

- **实际应用**: Transformer中d_k = 512，缩放是必要选择
- Multi-Head Attention计算方式 40:34

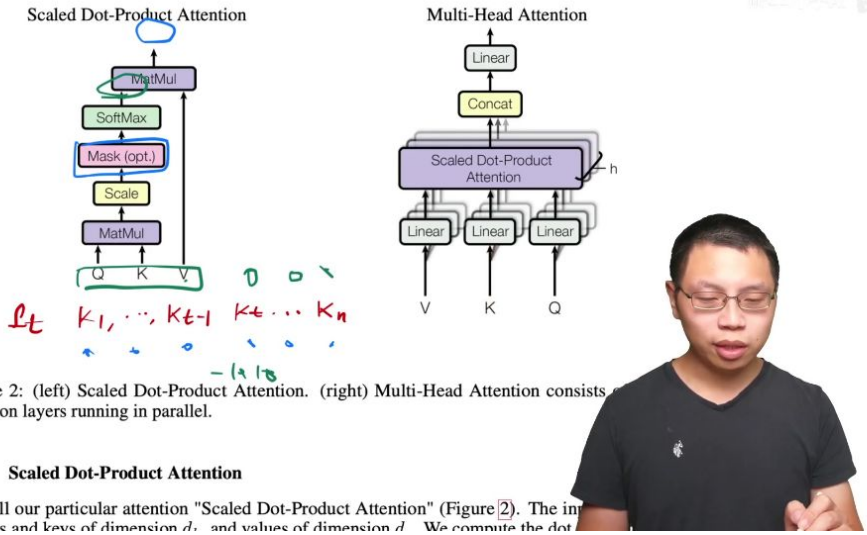


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists attention layers running in parallel.

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The in queries and keys of dimension d_k, and values of dimension d. We compute the dot

  - **基本流程**:
    - 将Q、K、V通过可学习的线性投影h次
    - 在每个投影空间并行计算注意力函数
    - 拼接所有头的结果再做最终投影
  - **参数设置**:
    - 常用头数h=8
    - 每个头的维度$d_k = d_v = d_{model} / h = 64$
  - **计算优势**: 因降维处理，总计算成本与单头全维度注意力相当
- 为什么要做mask 41:59
  - **核心目的**: 防止当前时间步看到未来信息
  - **序列建模**: 假设query和key等长(n)，时间对齐
  - **合理限制**: 时刻t的queryQ_t应只关注K_1到K_{t-1}
- mask的实现方式 43:10

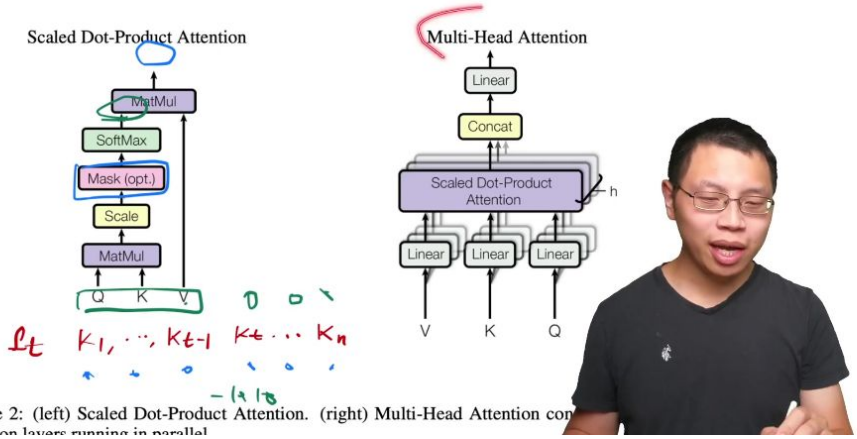


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention con attention layers running in parallel.

  - **技术方案**:
    - 计算所有注意力分数
    - 将未来位置(K_t及之后)替换为极大负值(如 - $1e^{10}$)

- **softmax效应**: 大负数经指数运算后接近0
- **实际效果**: 输出时只使用V_1到V_{t-1}的信息

- Multi-Head Attention的实现方式 44:11
  - **投影操作**: 通过可学习的W_i^Q, W_i^K, W_i^V将输入降至低维
  - **并行计算**: h个注意力头同时计算
  - **输出整合**: 使用W^O将拼接结果投影回原维度
- 为什么要做多头注意力机制 45:15

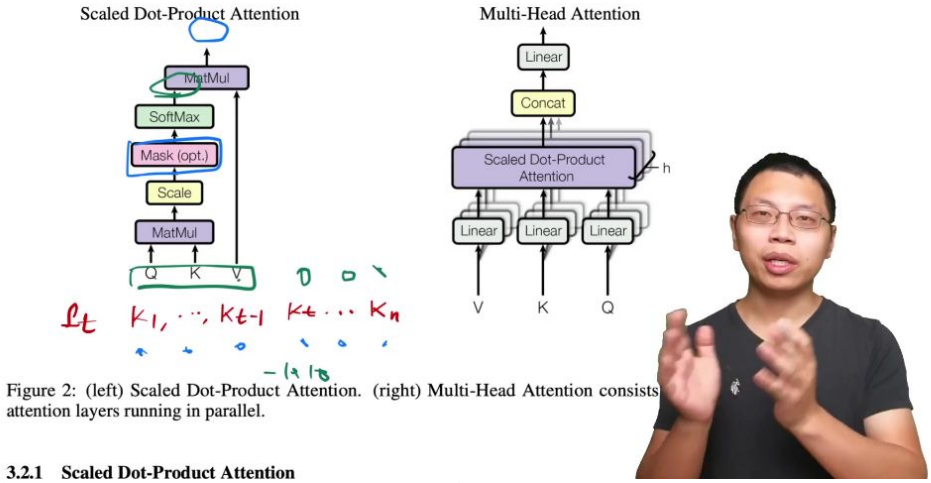


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists attention layers running in parallel.

  - 3.2.1 Scaled Dot-Product Attention
  - **单头限制**: 原始点积注意力缺乏可学习参数
  - **类比卷积**: 类似多输出通道捕捉不同模式
  - **核心思想**: 通过不同投影学习多样化的相似性度量
  - **优势**: 能同时关注不同表示子空间的信息
- Multi-Head Attention的计算公式 46:15

整体公式:
  - $MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O$

单头计算:
  - $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$
  - **参数维度**:
    - $W_i^Q, W_i^K \in R^{d_{model} \times d_k}$
    - $W_i^V \in R^{d_{model} \times d_v}$
    - $W^O \in R^{hd_v \times d_{model}}$
- Multi-Head Attention的实际应用 46:51
  - **典型配置**:
    - 头数h=8
    - 各头维度64($d_k = d_v = d_{model} / h$)
  - **实现技巧**:
    - 可通过单次矩阵乘法实现所有头的并行计算
    - 残差连接保持输入输出维度一致
  - **应用场景**: 主要用于编码器-解码器注意力机制

3）应用案例 47:45

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 left information flow in the decoder to preserve the auto-regressive property. We implement inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in 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 c

- **多头注意力参数**：使用$h = 8$个并行注意力头，每个头的维度$d_k = d_v = d_{model} / h = 64$，总计算成本与全维度的单头注意力相当
- **注意力应用场景**：Transformer在三个不同场景使用多头注意力机制
- Transformer中三种注意力使用方式概述 47:48
  - **编码器-解码器注意力**：
    - **query来源**：前一解码器层
    - **key/value来源**：编码器输出
    - **功能**：允许解码器每个位置关注输入序列所有位置，模仿传统序列到序列模型的注意力机制
  - **编码器自注意力**：
    - **输入同源性**：key、value和query均来自编码器前一层的输出
    - **关注范围**：每个位置可关注编码器前一层的所有位置
  - **解码器自注意力**：
    - **自回归限制**：使用mask防止当前位置关注后续位置，保持自回归特性
    - **实现方式**：通过将非法连接的softmax值设为-∞来实现mask
- 编码器自注意力机制：输入与输出 48:19
  - **输入结构**：对于长度为n的句子，输入为n个维度为d的向量
  - **三路输入**：同一输入复制为query、key和value三个相同副本
  - **输出特性**：
    - **维度保持**：输出保持n×d维度，与输入尺寸相同
    - **加权本质**：输出是输入的加权和，权重由query与各key的相似度决定
    - **自相似特性**：向量与自身的相似度最高，对应权重最大
  - **多头扩展**：通过h个不同投影空间学习多样化相似度度量
- 解码器自注意力机制：mask的作用 50:56
  - **结构相似性**：与编码器自注意力结构相同，输入为m个维度为d的向量
  - **核心差异**：
    - **mask实现**：计算当前query输出时，屏蔽后续位置的注意力权重（设为0）
    - **视觉表示**：使用黄色线表示被mask的权重连接
  - **自回归保障**：确保解码时每个位置只能基于已生成内容计算，不能"偷看"未来信息
- 编码器-解码器注意力机制：query、key和value的来源 51:38
  - **输入来源**：
    - **key/value**：编码器最后一层的n个d维输出

        - **query**：解码器下一层的m个d维输出
    - **工作机制**：
        - **动态聚焦**：根据解码器当前query，从编码器输出中提取相关信息
        - **权重可视化**：蓝色连接线粗细表示query与编码器各位置的相似度
    - **实例说明**：
        - **翻译案例**：解码"你好"时会给编码器的"hello"较大权重，解码"世界"时会给"world"较大权重
        - **信息筛选**：有效实现编码器信息到解码器的条件传递，忽略不相关部分

3. 位置聚合的馈送网络 54:54

1）Position-wise Feed-Forward Networks概述 55:00

- **本质**：实际上是一个多层感知机(MLP)，但具有位置独立性特点
- **位置处理**：对输入序列中的每个词向量(position)独立且相同地应用相同的MLP结构
- **术语解释**："point-wise"指在每个512维的词向量(position)上独立操作

2）Position-wise Feed-Forward Networks的计算过程 55:36

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.

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

While the linear transformations are the same across different positions, they use different param from layer to layer. Another way of describing this is as two convolutions with kernel siz The dimensionality of input and output is $d_{\text{model}} = 512$, and the inner-layer has dimensiona $d_{ff} = 2048$.

3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to co tokens and output tokens to vectors of dimension d_{model}. We also use the usual learne mation and softmax function to convert the decoder output to predicted next-toker our model, we share the same weight matrix between the two embedding layers a linear transformation, similar to [30]. In the embedding layers, we multiply those

- **数学表达**：$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$
- **维度变换**：
    - 输入维度：$d_{model} = 512$
    - 隐藏层维度：$d_{ff} = 2048$（扩大4倍）
    - 输出维度：还原为512维
- **结构组成**：
    - 第一线性层：512→2048维投影
    - ReLU激活函数
    - 第二线性层：2048→512维投影
- **实现特点**：在PyTorch中可直接用两个线性层实现，自动处理3D输入张量

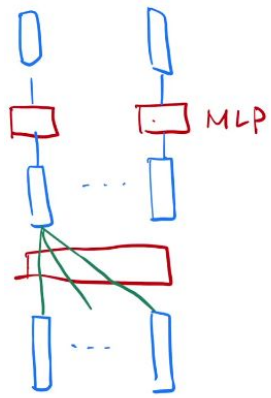
3）对比：Transformer与RNN在处理序列信息上的差异 58:42

- **Transformer机制**：
    - 通过Attention层全局聚合序列信息
    - MLP独立处理每个已包含全局信息的position
    - 信息流：序列→Attention汇聚→独立MLP处理
- **RNN机制**：

- 线性层处理当前输入
- 通过绿色连接传递上一时刻输出作为历史信息
- 信息流：当前输入+历史信息→线性层→输出

- **核心差异**：
  - Transformer：并行全局信息聚合
  - RNN：串行局部信息传递
  - 相同点：都使用线性变换进行语义空间转换

4. 嵌入和softmax 01:00:28



-
- **组成模块**：
  - 编码器输入Embedding
  - 解码器输入Embedding
  - softmax前线性变换
- **参数共享**：三个模块共享相同权重矩阵以简化训练
- **数值处理**：Embedding权重乘以$\sqrt{d}$（d=512）
  - **原因**：防止Embedding的L2范数随维度增大而减小
  - **效果**：使Embedding与后续相加操作保持相同数量级
- **位置编码必要性**：
  - Attention本身不具备时序感知能力
  - 词序打乱会导致相同Attention输出
  - 需显式加入位置信息以保持序列语义

5. 位置编码 01:02:53

Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

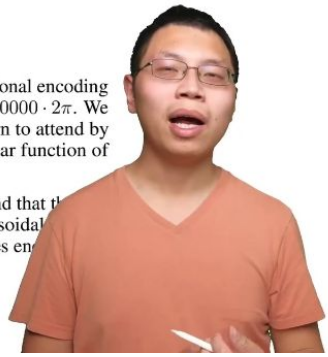
tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned 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 th versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal because it may allow the model to extrapolate to sequence lengths longer than the ones en during training.



4 Why Self-Attention

-

1）基本概念

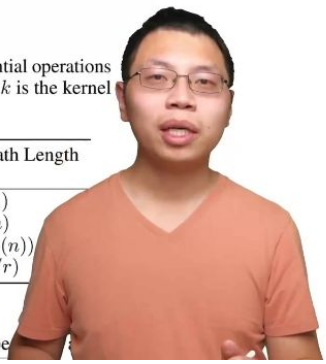
- **作用机制**：在Transformer模型中，通过将位置信息直接编码到输入向量中来解决序列顺序问题。每个词的位置i用数字表示(如1,2,3...)，然后将该位置信息加入词向量。
- **实现方式**：使用长度为512的向量表示位置信息，与词嵌入向量维度d_{model} = 512相同，可以直接相加。
- **数值范围**：通过乘以$\sqrt{d_{model}}$使位置编码值保持在-1到1之间，与词嵌入值范围匹配。

2）编码实现

5

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



- tokens in the sequence. To this end, we add "positional encodings" to the input embe
- **向量表示**：类似计算机用32位二进制表示数字，位置编码用512维向量表示位置序号。
- **函数选择**：采用不同频率的正弦和余弦函数组合计算各维度值：
  - 公式：$PE_{(pos,2i)} = sin(pos\ /\ 10000^{2i\ /\ d_{model}})$
  - 公式：$PE_{(pos,2i+1)} = cos(pos\ /\ 10000^{2i\ /\ d_{model}})$
- **几何特性**：波长形成从2π到10000·2π的几何级数，使模型能学习相对位置关系。

3）架构整合

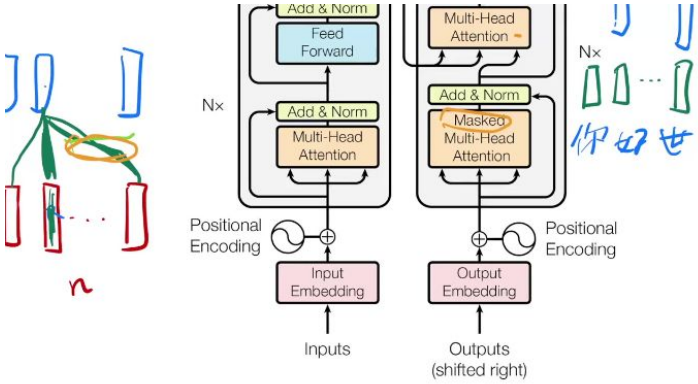


Figure 1: The Transformer - model architecture.

3.1 Encoder and Decoder Stacks

Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each
sub-layers. The first is a multi-head self-attention mechanism, and the second is a s
wise fully connected feed-forward network. We employ a residual connection [11]
the two sub-layers, followed by layer normalization [11]. That is, the output of ea

-
- **相加操作**：词嵌入向量与位置编码向量直接相加，完成时序信息注入。
- **顺序不变性**：模型输出仅与输入值的顺序相关，打乱输入顺序只会对应改变输出顺序。
- **实践优势**：相比可学习的位置嵌入，正弦编码能更好地处理训练时未见过的序列长度。

4）设计原理

- **线性关系**：对任意固定偏移量k，PE_{pos+k}可以表示为PE_{pos}的线性函数，便于模型学习相对位置。
- **实验验证**：与可学习位置嵌入相比效果相近，但正弦编码在长序列外推方面表现更好。
- **维度匹配**：位置编码维度d_{model} = 512与内层维度d_{ff} = 2048形成4倍扩展关系。

七、自注意力优点 01:05:46

1. 模型选择：正弦函数版本的位置编码

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
different layer types.

As noted in Table 1, a self-attention layer connects all positions with a constant number of se
executed operations, whereas a recurrent layer requires $O(n)$ sequential operations. I
computational complexity, self-attention layers are faster than recurrent layers when th
length n is smaller than the representation dimensionality d, which is most often th
sentence representations used by state-of-the-art models in machine translations, such as

-
- **选择依据**：实验比较了学习得到的位置嵌入和正弦函数版本，发现两者效果几乎相同（见表3行E）

- **最终选择**：采用正弦函数版本，因为该版本能让模型外推到比训练时更长的序列长度

实现方式：使用不同频率的正弦和余弦函数计算位置编码：

$$PE_{(pos,2i)} = sin(pos / 10000^{2i / d_{model}})$$

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

2. 自注意力层与循环层、卷积层的比较 01:06:25

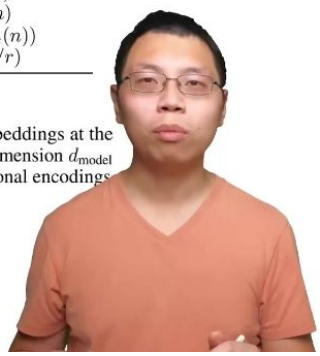
Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

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

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:

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



-
- **比较维度**：主要从三个关键指标进行比较：
  - 每层计算复杂度
  - 可并行化的计算量（用所需顺序操作的最小数量衡量）
  - 网络中长距离依赖的路径长度

3. 自注意力层：计算复杂度、顺序操作、最大路径长度 01:07:10

- **计算复杂度**：$O(n^2 \cdot d)$，因为需要计算query矩阵（n×d）和key矩阵（n×d）的乘积
- **顺序操作**：$O(1)$，矩阵乘法并行度高
- **最大路径长度**：$O(1)$，任意query与任意key-value pair只需一次运算即可关联
- **优势**：特别适合处理长序列数据，信息传递一步到位

4. 循环层：计算复杂度、顺序操作、最大路径长度 01:08:26

- **计算复杂度**：$O(n \cdot d^2)$，每个时间步需要进行d×d矩阵运算，共n步
- **顺序操作**：O(n)，必须按时间步顺序计算
- **最大路径长度**：O(n)，最初点的信息需要经过n步才能传递到最后
- **缺点**：对长序列处理不佳，信息容易在传递过程中丢失

5. 卷积层：计算复杂度、顺序操作、最大路径长度 01:10:05

- **计算复杂度**：$O(k \cdot n \cdot d^2)$，k为卷积核大小（通常为3或5）
- **顺序操作**：$O(1)$，卷积操作并行度高
- **最大路径长度**：$O(log_k(n))$，信息需要通过多层卷积才能传递到远处
- **特点**：实际计算复杂度与RNN相近，但通常比RNN更快

6. 受限的自注意力：计算复杂度、顺序操作、最大路径长度 01:11:13

- **计算复杂度**：O(r·n·d)，r为受限邻域大小
- **顺序操作**：$O(1)$
- **最大路径长度**：$O(n / r)$，远距离信息需要多步传递

- **应用情况**：实际使用较少，因为牺牲了处理长距离依赖的优势

八、实验 01:12:49

1. 训练数据集和batching方法 01:12:57

- **数据集**：
  - 英语-德语：WMT 2014标准数据集，4.5万句对
  - 英语-法语：300万句对
- **分词方法**：使用Byte Pair Encoding(BPE)
  - **优点**：提取词根，降低字典大小（3.7万token）
  - **特点**：英德共享字典，编码器解码器可共享embedding权重

2. 硬件和Schedule 01:14:14

- **硬件配置**：8个NVIDIA P100 GPU
- **训练时间**：
  - 基础模型：10万步，12小时（每步0.4秒）
  - 大模型：30万步，3.5天（每步1.0秒）

3. 优化器 01:15:32

- **优化算法**：Adam优化器
- **参数设置**：
  - $\beta_1 = 0.9$
  - $\beta_2 = 0.98$
  - $\epsilon = 10^{-9}$

学习率调度：

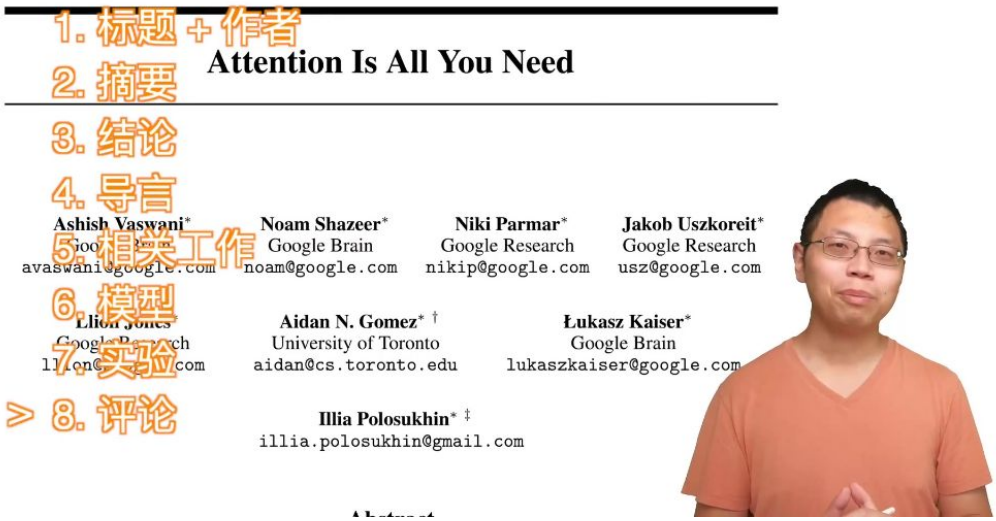
- $lrate = d_{model}^{-0.5} \cdot min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$
  - warmup_steps=4000
  - 模型越宽（d_{model}越大），学习率越低

4. 正则化 01:16:33

- **残差Dropout**：
  - 在每个子层（多头注意力层和MLP）输出上应用
  - 在进入残差连接和层归一化前执行
  - dropout率=0.1，剩余值乘以1.1
- **输入Dropout**：
  - 在词嵌入和位置编码相加后应用
  - 同样使用0.1的dropout率

九、Transformer 01:17:30

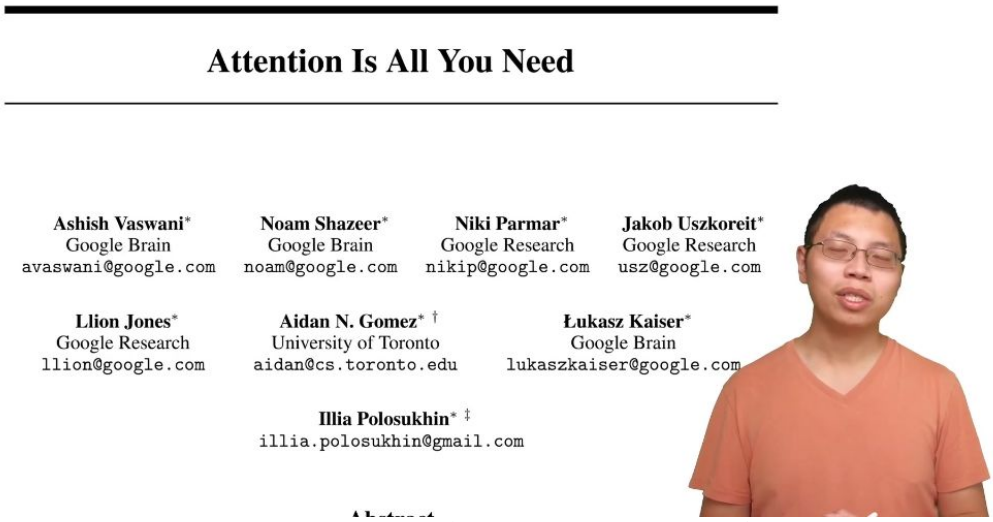
1. 写作评价 01:21:49



- **简洁性**: 文章写作非常简洁，每句话只讲一件事情，全文可在1.5小时内读完
- **直白风格**: 基本采用"提出模型→描述结构→展示结果"的直线式写作，缺乏故事性叙述
- **篇幅限制**: 为容纳大量内容牺牲了深度讨论，建议将次要内容移至附录以保留正文叙事空间

2. 模型评价 01:22:59

1）影响力



- **领域扩展**: 从机器翻译扩展到几乎所有NLP任务(BERT/GPT)，并延伸至图像、语音、视频等多模态领域
- **范式统一**: 类似CNN对CV的影响，提供统一框架替代任务专用架构，降低领域知识门槛
- **技术融合**: 多模态特征可共享语义空间，促进跨模态模型发展

2）技术特性

(3.5 days).

5.3 Optimizer

We used the Adam optimizer [20] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula:

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

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

5.4 Regularization

We employ three types of regularization during training:

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 positional encodings in both the encoder and decoder stacks. For the base model, we use $P_{drop} = 0.1$.

-
- **正则化组合**:
  - **残差Dropout**: 在每子层输出、嵌入求和、位置编码处应用，基础模型P_{drop} = 0.1
  - **标签平滑**: 使用ϵ_{ls} = 0.1，虽增加困惑度但提升BLEU分数
- **优化策略**:
  - Adam优化器参数β_1 = 0.9,β_2 = 0.98,ϵ = 10^{-9}
  - 学习率公式：$lrate = d_{\text{modd}}^{-0.5} \cdot min\left(step^{-0.5}, step \cdot warmup\_steps^{-1.5}\right)$
  - 4000步线性warmup

3）局限与启示

Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{2}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

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 in Table 2) outperforms the best previously reported models (including ensembles) by more BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our ba surpasses all previously published models and ensembles, at a fraction of the training cos the competitive models.

[19] On the WMT 2014 English-to-French translation task, our big model achieves a BLEU s

-
- **架构误解**:
  - 标题"Attention is All You Need"存在误导，MLP和残差连接同等重要
  - Attention主要作用为序列信息聚合，不直接建模顺序关系
- **数据需求**: 因假设更一般化，需要更大数据量和模型规模补偿信息提取效率
- **研究启示**: 证明CNN/RNN之外存在更优架构，激励MLP等简单架构的探索

	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**	

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

- **参数设计**:
  - 基础模型：6层，$d_{model} = 512$，8头($h = 8$)，$d_k = d_v = 64$
  - 大模型：宽度加倍，$d_{model} = 1024$，$h = 16$，$P_{drop} = 0.3$，训练30万步
  - 关键参数：层数(N)、模型宽度(d)、头数(h)按比例缩放，保持$h \times d_k = d_{model}$

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
		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

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

- **泛化能力**：在英语成分句法分析任务(WSJ23)达到91.3 F1，半监督设置达92.7 F1，验证架构普适性

十、知识小结

知识点	核心内容	关键创新点	应用领域
模型架构	纯基于注意力机制的编码器-解码器结构	完全摒弃RNN/CNN，首创self-attention机制	NLP/计算机视觉/语音处理
多头注意力	并行计算h个注意力头模拟CNN多通道效果	Multi-head设计实现不同语义空间的特征提取	机器翻译/文本生成
位置编码	正弦函数生成的位置向量与词嵌入相加	**解决时序信息缺失问题**的非参数化方案	序列建模任务

层标准化	LayerNorm替代BatchNorm处理变长序列	稳定训练过程，适应不同长度输入	深度神经网络训练
残差连接	每个子层输出与输入直接相加	缓解梯度消失，支持超深网络构建	模型优化技术
训练效率	8GPU训练3.5天达到SOTA	并行计算优势比RNN快**4-10倍**	大规模模型训练
迁移能力	单一架构处理多模态数据	开创"基础模型"(Foundation Model)范式	跨模态学习
对比维度	Transformer	RNN	CNN
---	---	---	---
计算复杂度	$O(n^2 \cdot d)$	$O(n \cdot d^2)$	$O(k \cdot n \cdot d^2)$
并行度	完全并行	序列依赖	局部并行
长程依赖	一步直达	需要n步传递	需要$\log_k n$层
参数量	较大(需预训练)	中等	较小
实验数据	基座模型	大模型	
---	---	---	
参数量	65M	213M	
训练时间	12小时	3.5天	
BLEU得分	28.4	**41.8**	
硬件配置	8×P100 GPU	8×P100 GPU	
关键数字	数值	意义	
---	---	---	
隐藏层维度	512	基准模型宽度	
注意力头数	8	多头设计参数	
前馈层维度	2048	MLP中间层扩展	
Dropout率	0.1	正则化强度	