



中国科学技术大学
University of Science and Technology of China

《人工智能数学原理与算法》
第5章 Transformer

5.2Transformer的编码与解码器

宋彦

songyan@ustc.edu.cn

01

Transformer的整体架构

02

Transformer的编码器

03

Transformer的解码器

本课重点

Transformer的提出

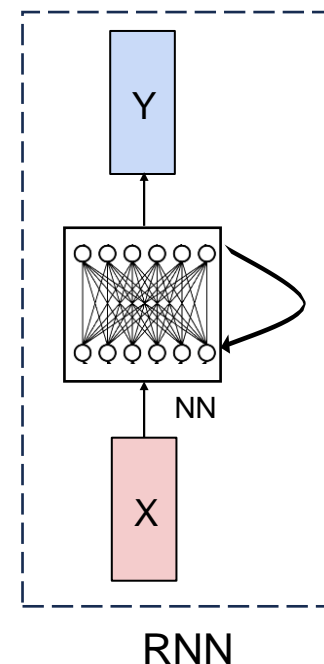
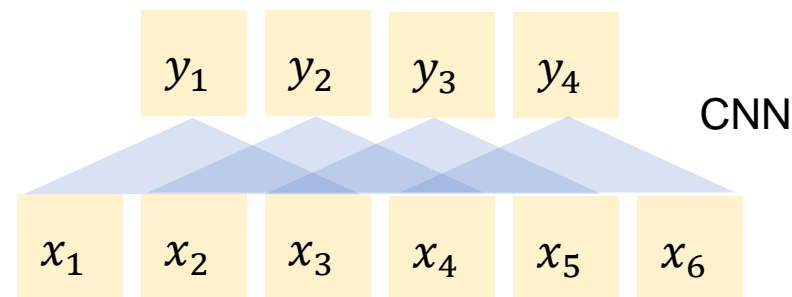
□ Transformer之前的主流文本处理架构

□ 卷积神经网络 (CNN)

- 利用卷积核抽取局部信息
- 卷积核参数共享
- 通过多层卷积和池化操作，逐步形成高级特征
- ...

□ 循环神经网络 (RNN)

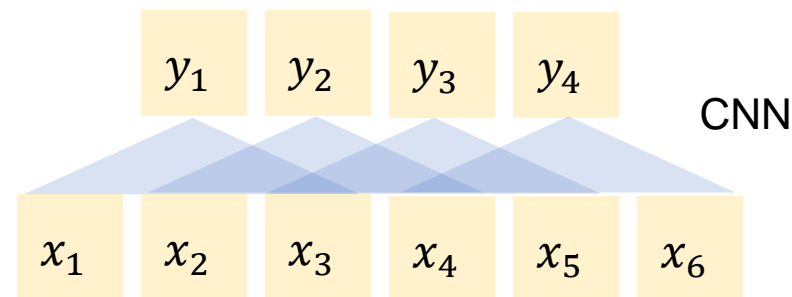
- 通过隐藏状态传递实现序列编码
- 每个循环网络单元共用一套参数
- 通过层数堆叠提升表征能力
- ...



Transformer的提出

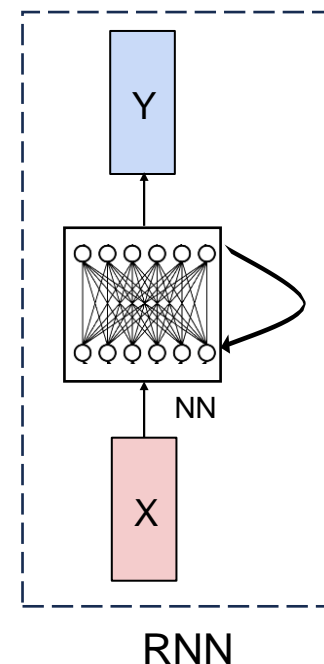
□ CNN的局限

- CNN仅仅编码局部信息，无法处理全局或者跳跃式的依赖
- 在处理序列时，难以针对序列长度做出动态调整



□ RNN的局限

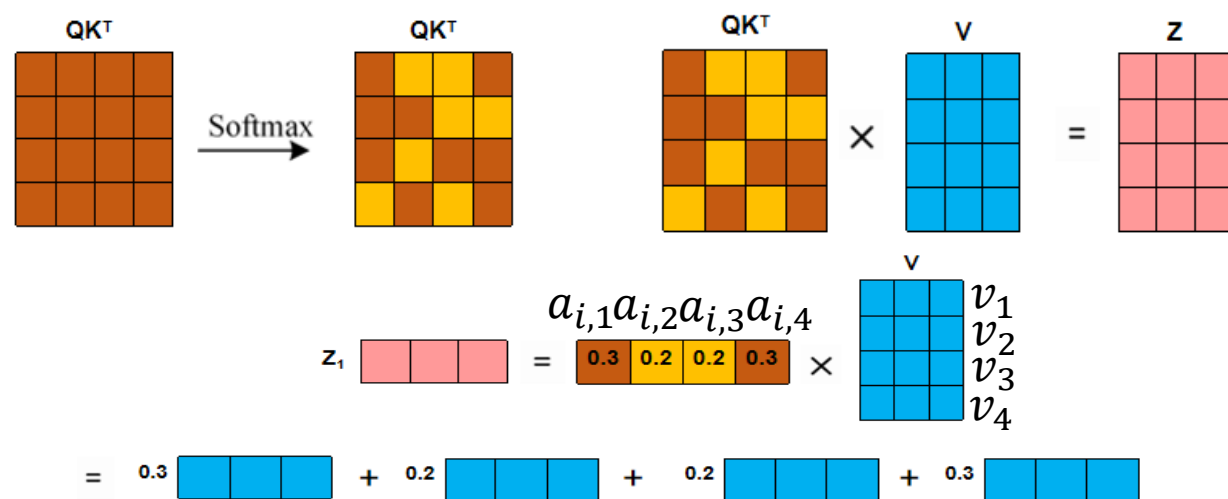
- RNN难以处理长依赖，存在遗忘问题
- 存在梯度消失或者梯度爆炸的问题



Transformer的提出

□ 回顾

- Self-attention计算每个输入单元对输出的贡献，因此能够在全局范围内对输入进行建模
- 然而，self-attention无法区分不同输入的位置信息
- Q、K、V仅从单一维度对不同输入单元的贡献进行计算



Transformer的提出

□ 模型架构需要满足以下条件

- 以Attention为基础，计算全局信息
- 用特定的方式提供输入单元的位置信息
- 能够计算不同方面的注意力，评估输入单元在不同方面的影响
- 高效可计算，可通过堆叠层数等方式快速拓展
- ...

□ Ashish Vaswani 等人于2017年提出Transformer架构



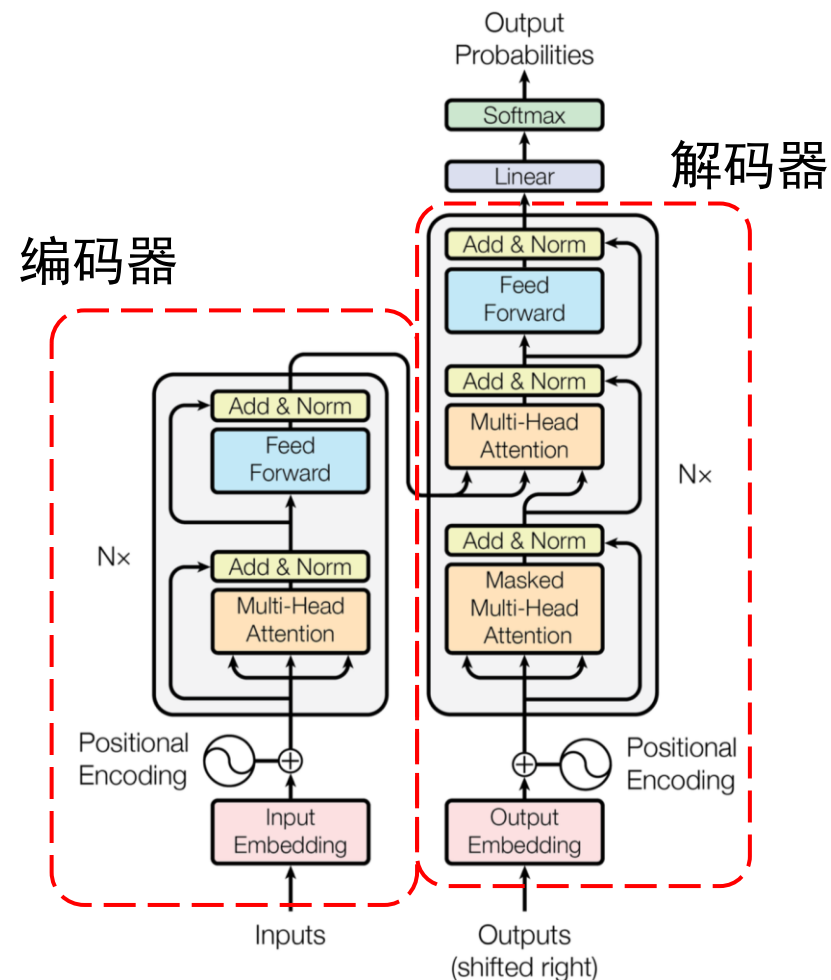
Transformer

□ 基于注意力机制的序列建模

- 结构类似CNN，但考虑了长距离依赖

□ 应用新技术以增强模型的表征能力

- 位置编码
- 多头（注意力）
- 类残差连接
- ...



01

Transformer的整体架构

02

Transformer的编码器

03

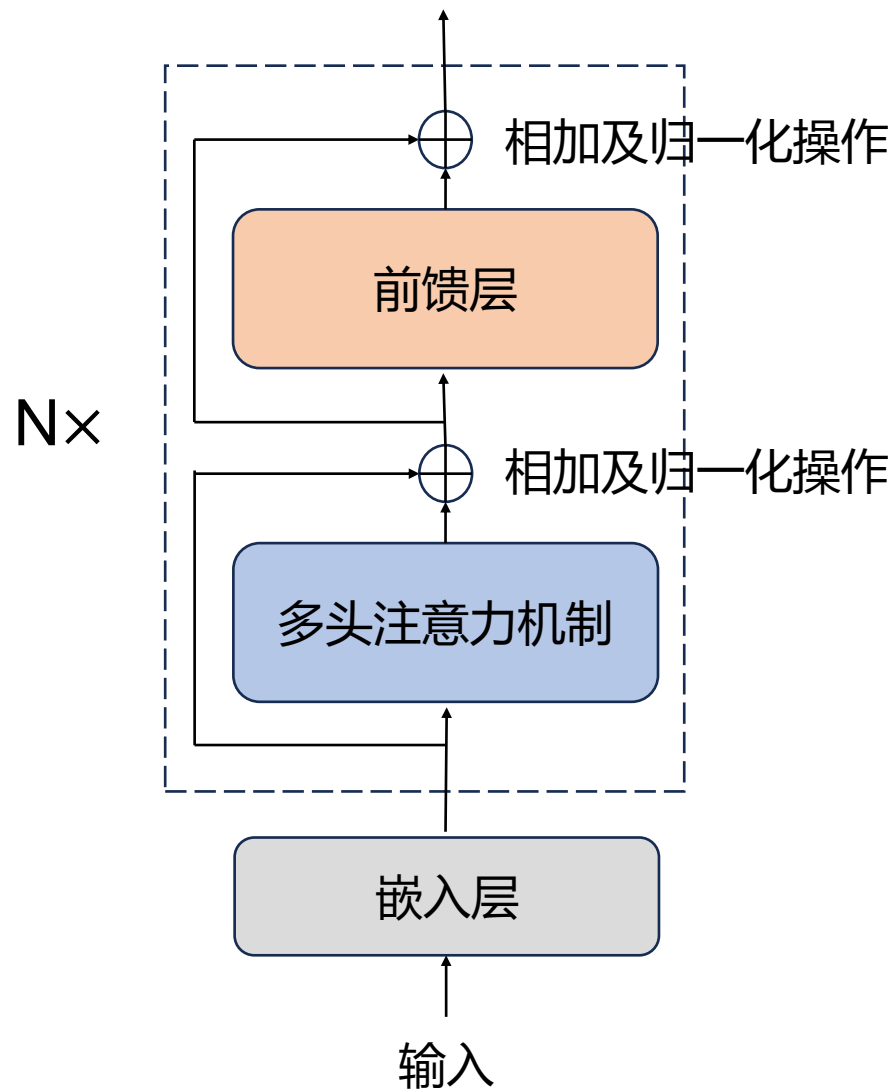
Transformer的解码器

本课重点

Transformer的编码器整体架构

□ Transformer 编码器的整体架构

- 嵌入层
 - 多头注意力机制
 - 前馈层
 - 相加及归一化
- ## □ Transformer可通过层数堆叠实现参数的增加



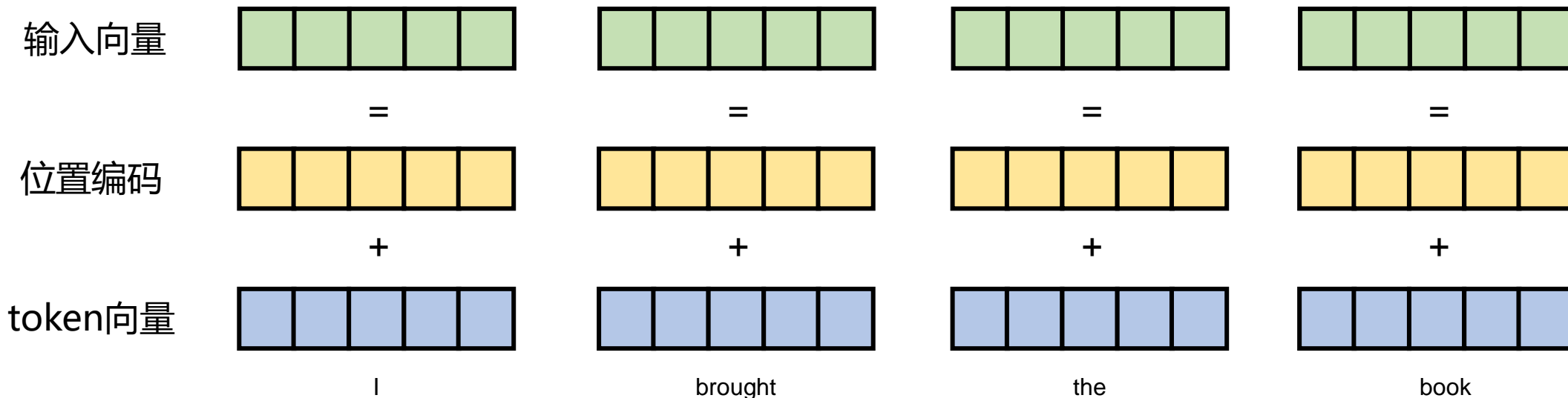
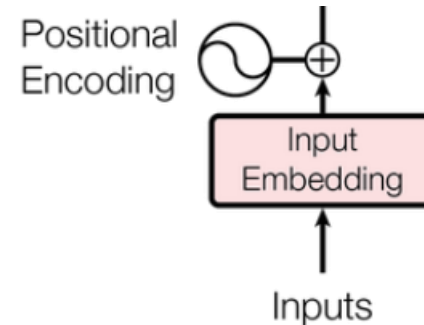
Transformer的嵌入层

□ Transformer有两部分输入

□ Token向量

□ 位置编码

□ 注意，Transformer采用的是一种类似CNN的建
模范式，模型本身无法理解单词之间的顺序关系。



- Token指的是文本处理中的基本单元，例如一个词、一个字、一个子词 (subword) 或符号。

“I bought the book that he read yesterday”

→ [“I”, “bought”, “the”, “book”, “that”, “he”, “read”, “yesterday”]

- Token向量

	<	—	d_{emb_dim}	—	>
<i>I</i>	0.31	-0.58	...	0.04	-0.32
<i>brought</i>	-0.41	0.97	...	-0.62	-0.79
<i>the</i>	-0.59	-0.09	...	0.64	0.05
<i>book</i>	0.26	0.32	...	-0.33	-0.02
...
<i>yesterday</i>	0.53	0.38	...	-0.08	0.80

通常是随机初始化的，也可以使用预训练的词嵌入 (embeddings) 。

□ 位置编码

□ 与 RNN 风格的模型不同，Transformer 中没有明确的方式使用“循环”机制对词序列进行编码

如何为不同的 token 设置可扩展的编码方式？

- 对于长序列必须合理
- 其值应可跟踪
- 能够记录单词之间的相对位置（为后续注意力做好准备）

□ 位置编码使用三角函数

$$\begin{array}{c} / \\ \text{brought} \\ \text{the} \\ \text{book} \\ \dots \\ \text{yesterday} \end{array} \begin{array}{c} < & - & d_{emb_dim} & - & > \\ \left(\begin{array}{c} \sin \left(\frac{0}{10000 \frac{0}{emb_dim}} \right) \\ \sin \left(\frac{1}{10000 \frac{0}{emb_dim}} \right) \\ \sin \left(\frac{2}{10000 \frac{0}{emb_dim}} \right) \\ \sin \left(\frac{3}{10000 \frac{0}{emb_dim}} \right) \\ \dots \\ \sin \left(\frac{pos}{10000 \frac{0}{d_{emb_dim}}} \right) \end{array} \right. & \left(\begin{array}{c} \cos \left(\frac{0}{10000 \frac{0}{emb_dim}} \right) \\ \cos \left(\frac{1}{10000 \frac{0}{emb_dim}} \right) \\ \cos \left(\frac{2}{10000 \frac{0}{emb_dim}} \right) \\ \cos \left(\frac{3}{10000 \frac{0}{emb_dim}} \right) \\ \dots \\ \cos \left(\frac{pos}{10000 \frac{0}{emb_dim}} \right) \end{array} \right) & \left(\begin{array}{c} \sin \left(\frac{0}{10000 \frac{2}{emb_dim}} \right) \\ \sin \left(\frac{1}{10000 \frac{2}{emb_dim}} \right) \\ \sin \left(\frac{2}{10000 \frac{2}{emb_dim}} \right) \\ \sin \left(\frac{3}{10000 \frac{2}{emb_dim}} \right) \\ \dots \\ \sin \left(\frac{pos}{10000 \frac{2}{emb_dim}} \right) \end{array} \right) & \left(\begin{array}{c} \cos \left(\frac{0}{10000 \frac{2}{emb_dim}} \right) \\ \cos \left(\frac{1}{10000 \frac{2}{emb_dim}} \right) \\ \cos \left(\frac{2}{10000 \frac{2}{emb_dim}} \right) \\ \cos \left(\frac{3}{10000 \frac{2}{emb_dim}} \right) \\ \dots \\ \cos \left(\frac{pos}{10000 \frac{2}{emb_dim}} \right) \end{array} \right) & \left(\begin{array}{c} \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \end{array} \right) \end{array}$$

- 为了添加位置信息（序列的顺序）

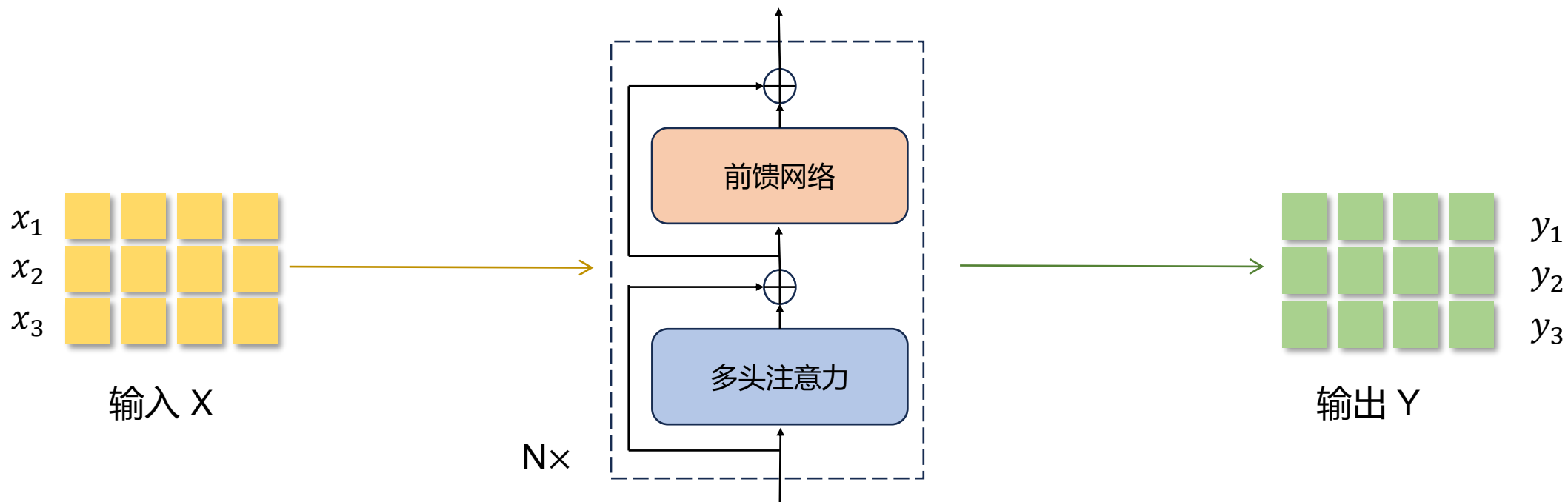
$$PE(pos, 2i) = \sin(pos/10000^{2i/d_{model}})$$

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

- 位置编码的每个维度都对应一个正弦曲线
- 对于任何固定偏移量 k , PE_{pos+k} 可以表示为 PE_{pos} 的线性变换, 这使得模型能够轻松地通过相对位置来学习注意力

多层多头注意力

□ Transformer包括多层多头注意力



回顾：自注意力

□ 每个输入被分解为三个部分

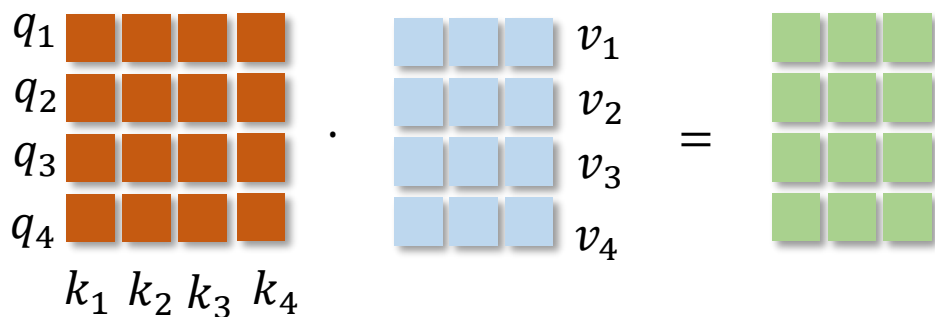
$$X_{Embedding} * W^Q = Q$$

$$X_{Embedding} * W^K = K$$

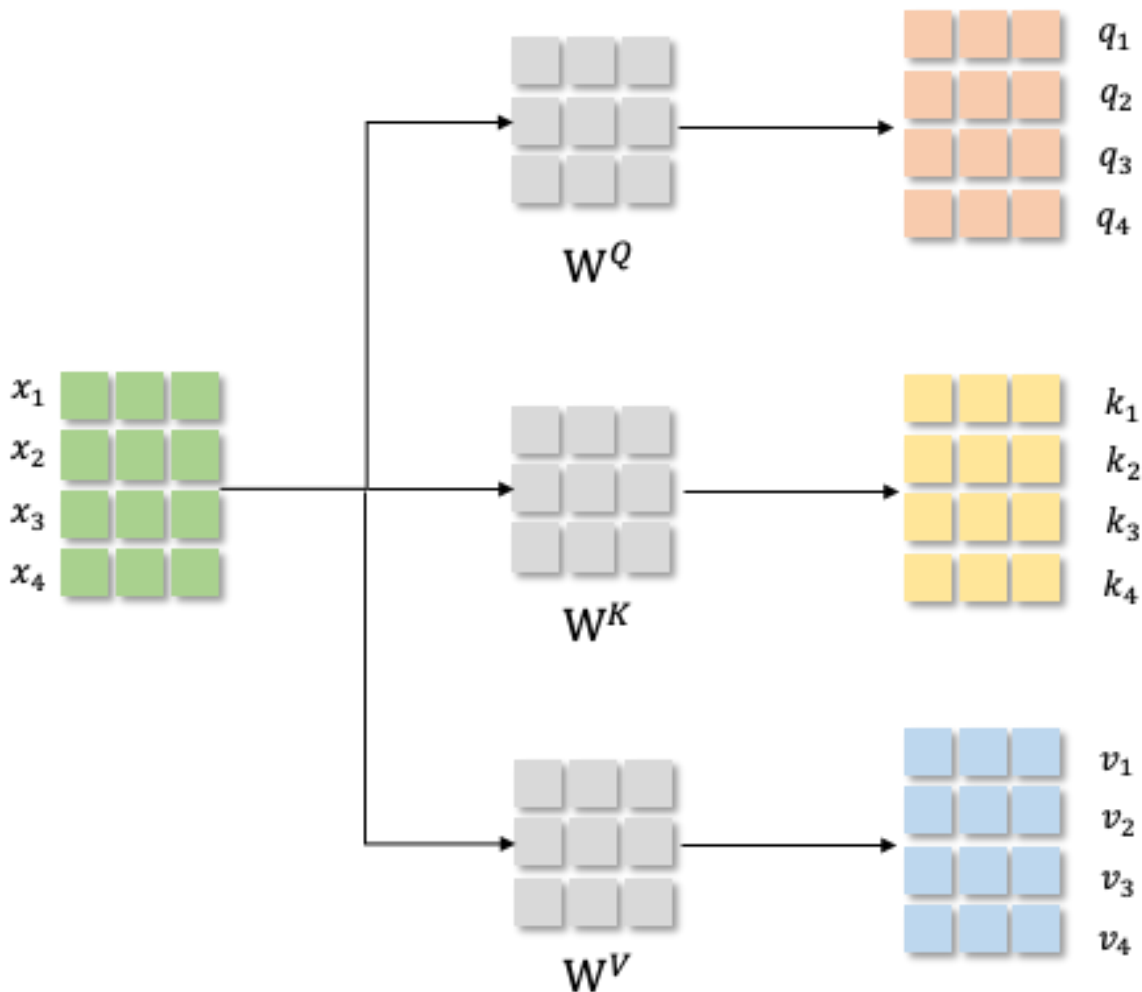
$$X_{Embedding} * W^V = V$$

□ QK^T 计算注意力，然后使用该注意力为V加权

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



The diagram illustrates the calculation of the attention matrix. On the left, a 4x4 matrix of orange squares represents the query matrix Q , with rows labeled q_1, q_2, q_3, q_4 and columns labeled k_1, k_2, k_3, k_4 . This is followed by a dot operator. Next is a 4x4 matrix of blue squares representing the key matrix K^T , with rows labeled v_1, v_2, v_3, v_4 and columns labeled k_1, k_2, k_3, k_4 . An equals sign follows, leading to a 4x4 matrix of green squares representing the resulting attention matrix.



多头机制的动机

- **需要处理的信息往往具有多个维度**

- 文本的多维性：不同内容的句法、语义功能维度
- 图像的多维性：整体特性、局部特性等
- ...

- **单独的self-attention信息视角单一，从而带来表示能力的瓶颈**

- **可以通过多头，实现不同维度信息的加权**

多头机制

- 对于每个单词，应该有多组 Q、K、V 可用
- 因此，多重注意适合从多角度处理这种情况

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

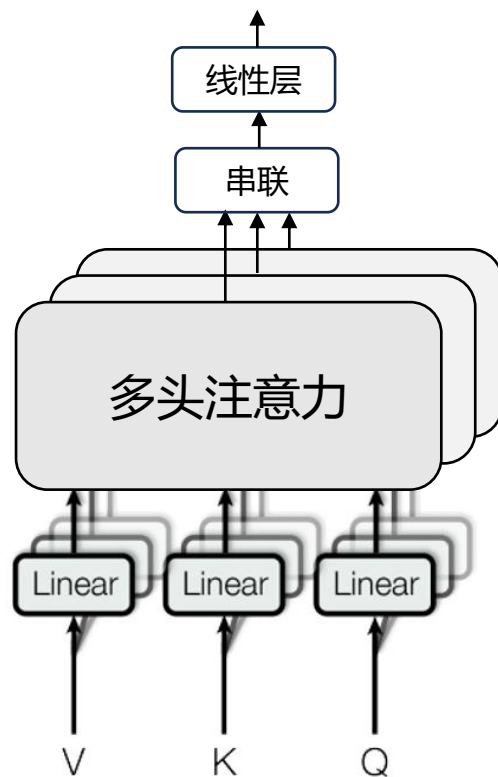
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1 \cdots, \text{head}_h)W^O$$

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

$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

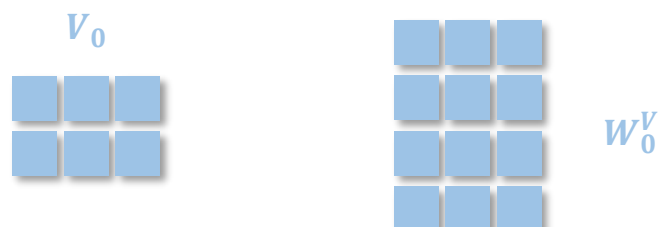
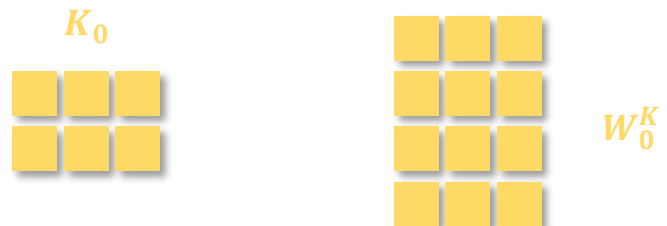
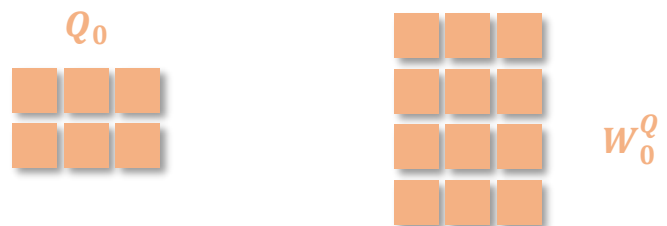
$$W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$

$$h = 8, d_k = d_v = \frac{d_{\text{model}}}{h} = 64$$

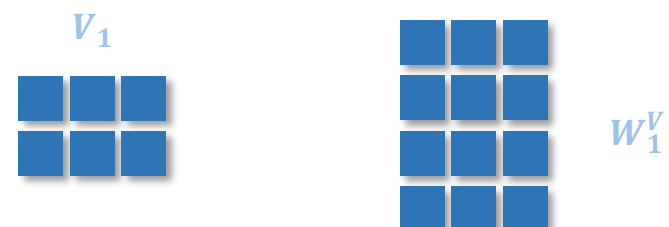
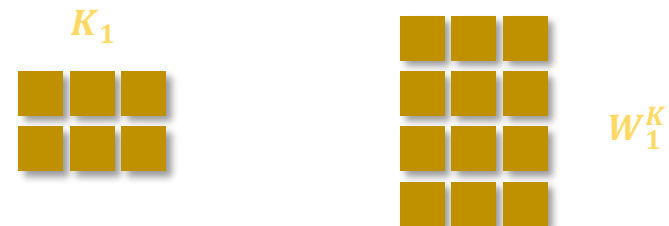


多头机制

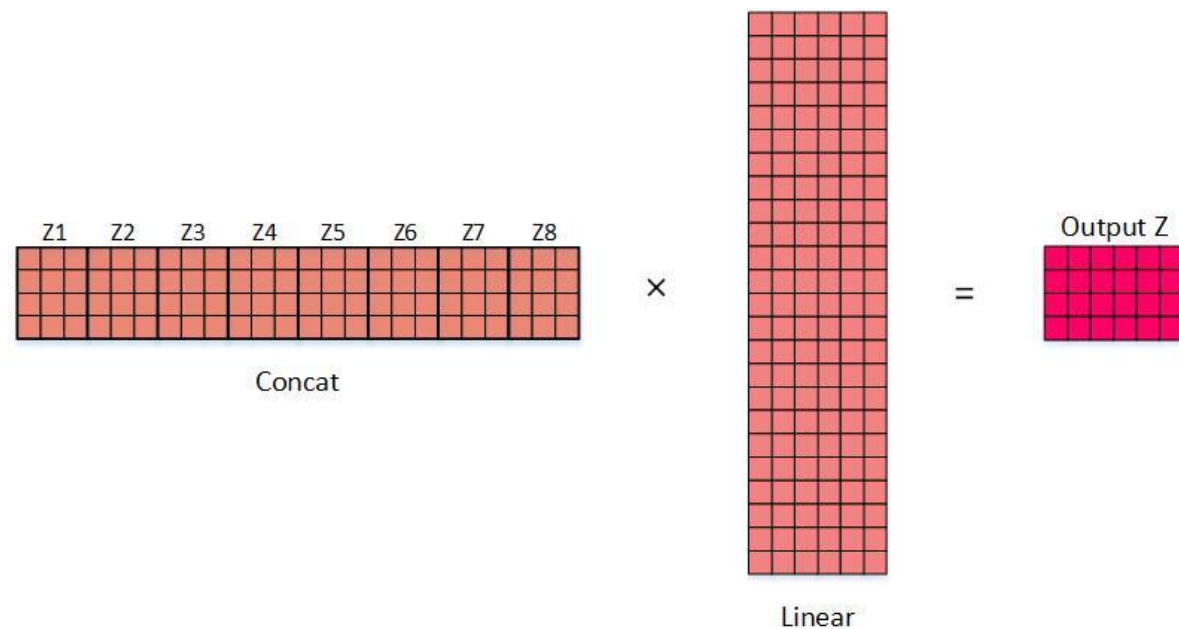
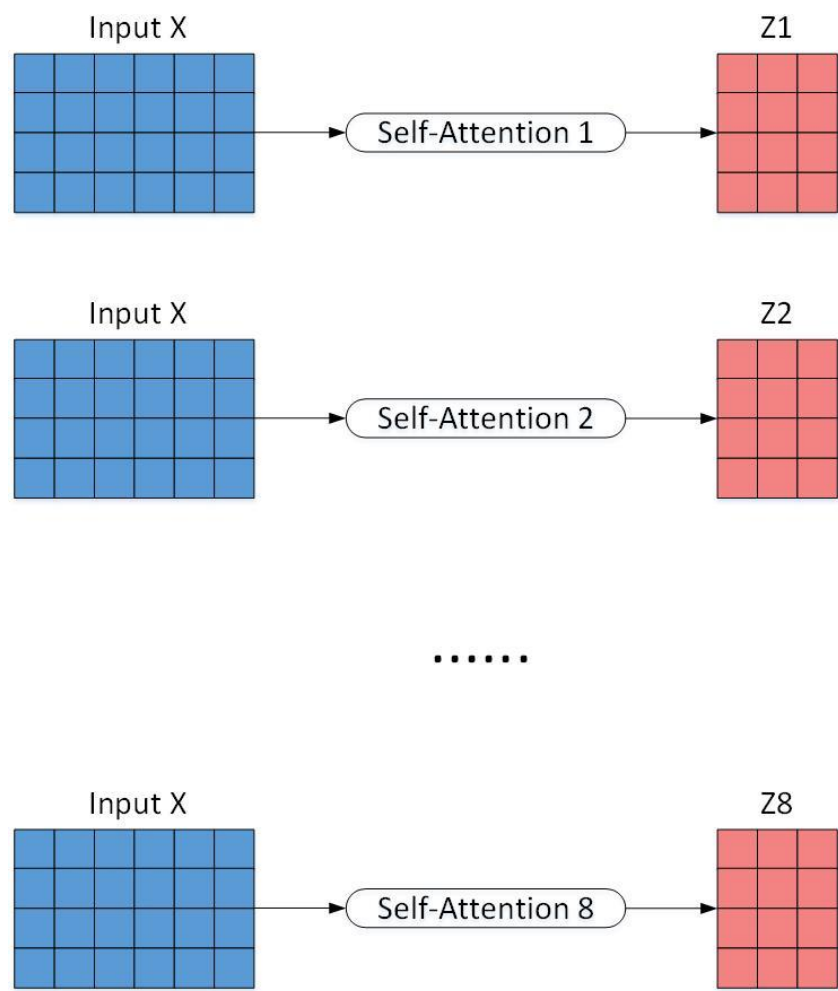
ATTENTION HEAD #0



ATTENTION HEAD #1

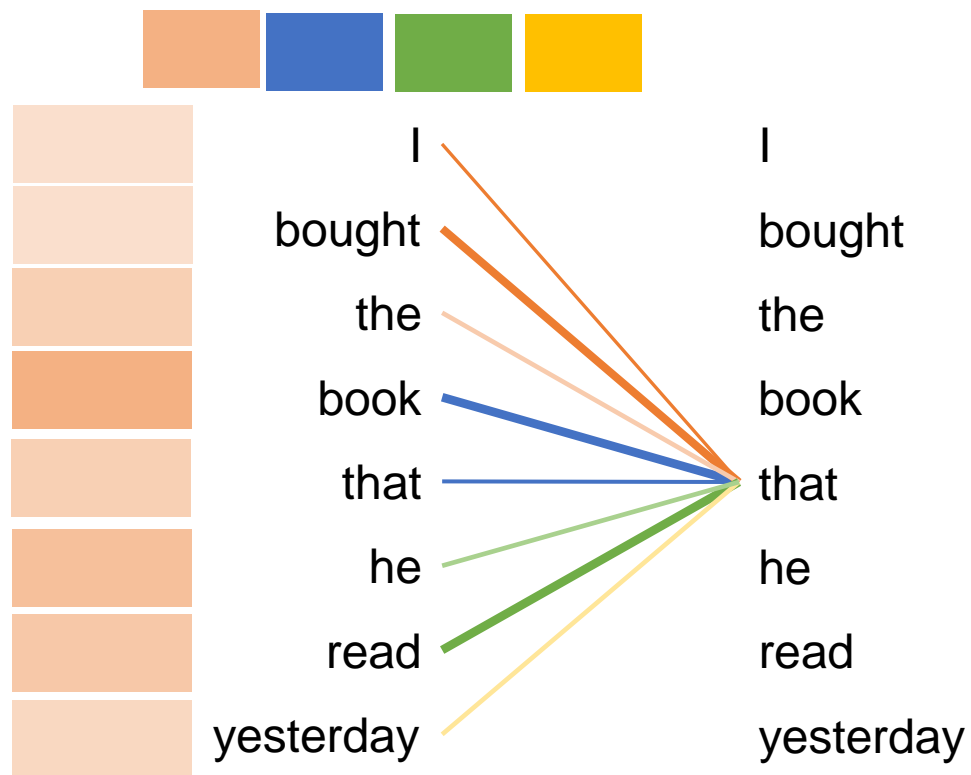


多头机制



通过线性变换，将多头向量与输出的维度对齐

□ 多头注意力的意义是什么？

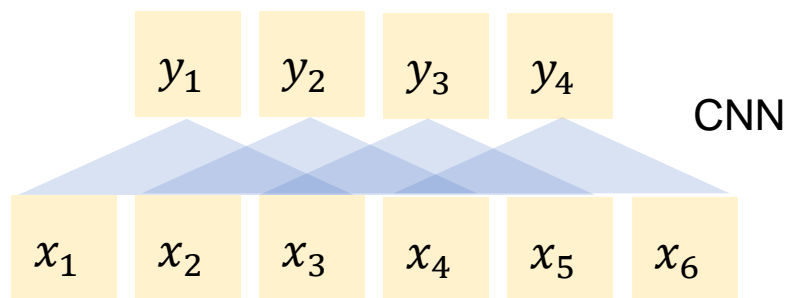


□ 不同“头”的查询向量从不同角度确定重要的输入

卷积层与自注意力层的对比

- 两者都通过计算上下文的加权和来更新表示
- 两者都可以并行执行
- 多头对应于 CNN 中的通道

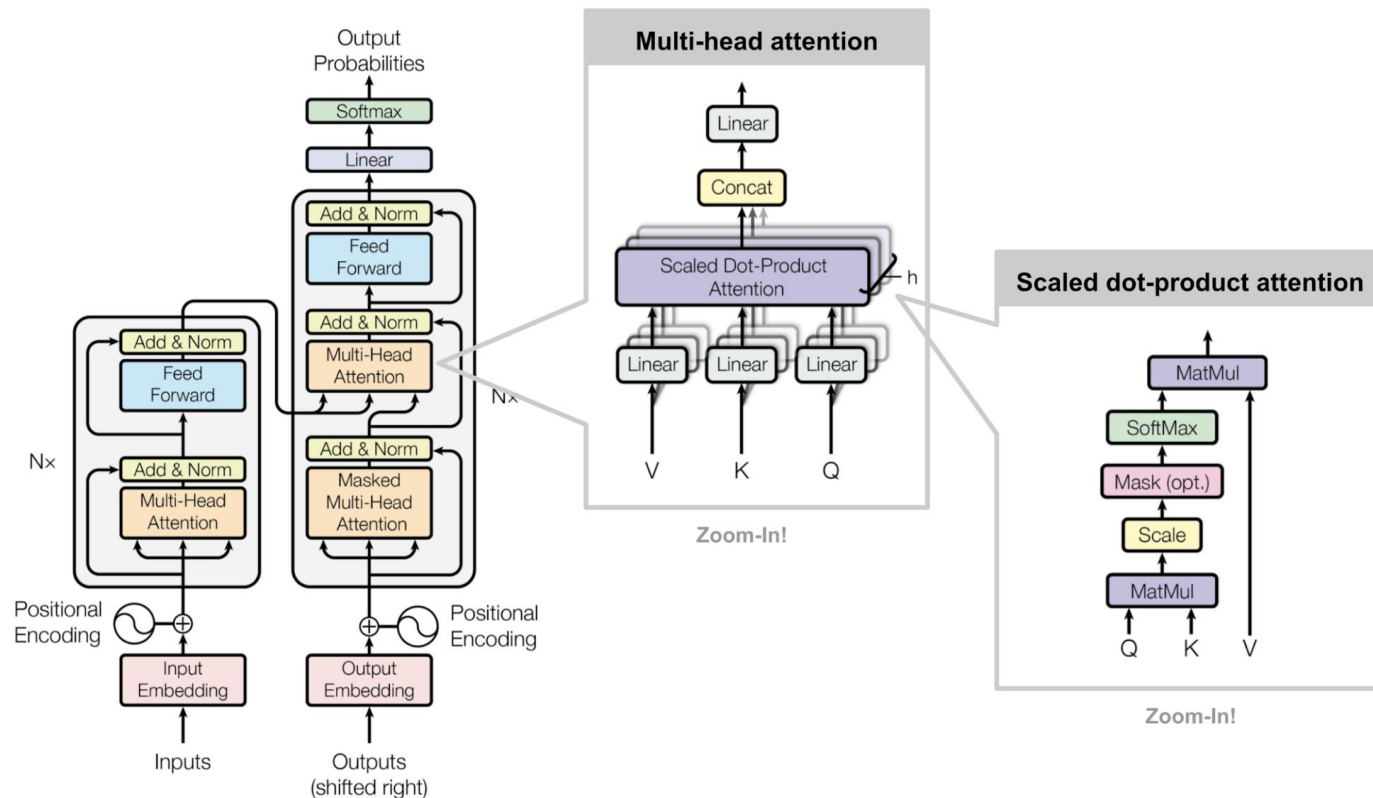
$$\begin{matrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{matrix} \begin{matrix} k_1 & k_2 & k_3 & k_4 \end{matrix} \cdot \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} = \begin{matrix} & & & \\ & & & \\ & & & \\ & & & \end{matrix}$$



$$z_1 \begin{matrix} & & \end{matrix} = \begin{matrix} 0.3 & 0.2 & 0.2 & 0.3 \end{matrix} \times \begin{matrix} v \\ & & & \\ & & & \\ & & & \\ & & & \end{matrix}$$
$$= 0.3 \begin{matrix} & & \end{matrix} + 0.2 \begin{matrix} & & \end{matrix} + 0.2 \begin{matrix} & & \end{matrix} + 0.3 \begin{matrix} & & \end{matrix}$$

多头机制的总结

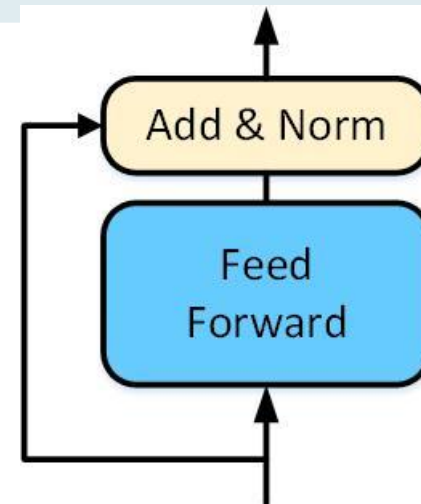
- Q、K、V 都是从输入中获取的
- Q 用于根据关键向量从特定角度确定重要的值向量
- “多头”用于从不同角度对重要信息进行建模



□ 前馈层

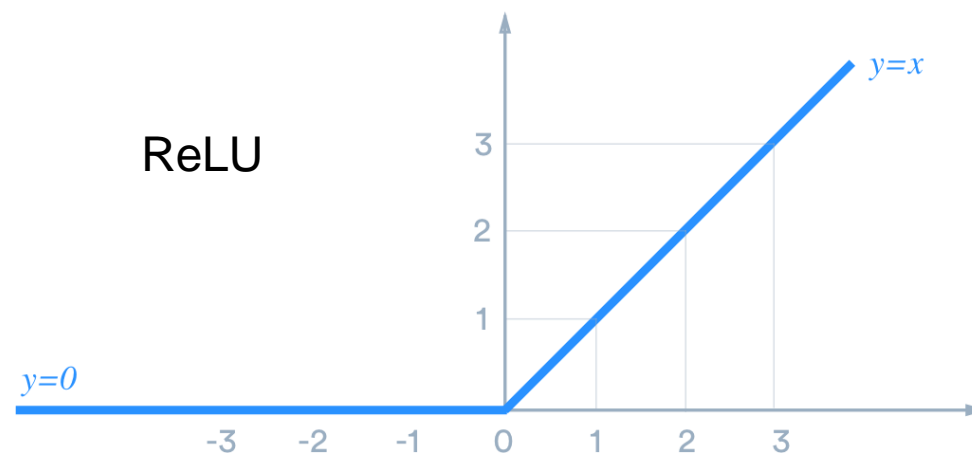
$$FFN(x) = \max(0, xW_1 + b_1) W_2 + b_2$$

$$W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}, W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$$



□ 线性变换用于将注意力映射到更大维度的表示上，然后再通过 RELU 引入非线性进行过滤，最后恢复到原始维度。

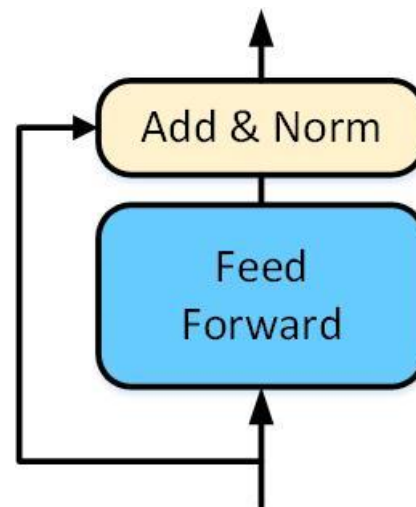
$$RelU(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



相加和归一化

□ 相加

- 与FFN结果的残差连接
- $F(x) + x$ 形式保持梯度的稳定性



□ 归一化

- 层归一化相当于对样本中的所有数据进行归一化
- 不受批次大小影响，具有与批次归一化类似的效果

$$\mu^l = \frac{1}{H} \sum_{i=1}^H \alpha_i^l$$

$$\sigma^l = \sqrt{\frac{1}{H} \sum_{i=1}^H (\alpha_i^l - \mu^l)^2}$$

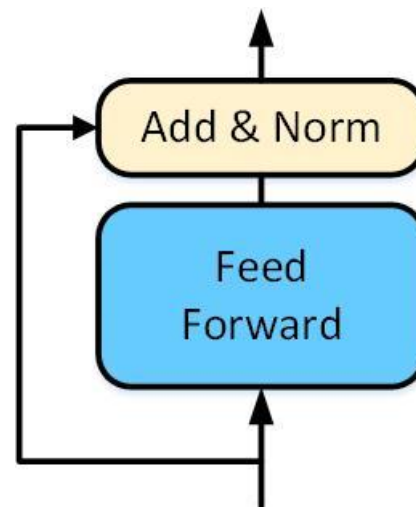
相加和归一化

□ 相加的动机

- 深层网络容易出现梯度消失/退化问题
- 增加网络深度未必带来性能提升

□ 相加的效果

- 保持梯度通道畅通，缓解梯度消失
- 让子层更关注“增量”学习 (learning residual)
- 支持上百层的稳定训练



$$output = Sublayer(x) + x$$

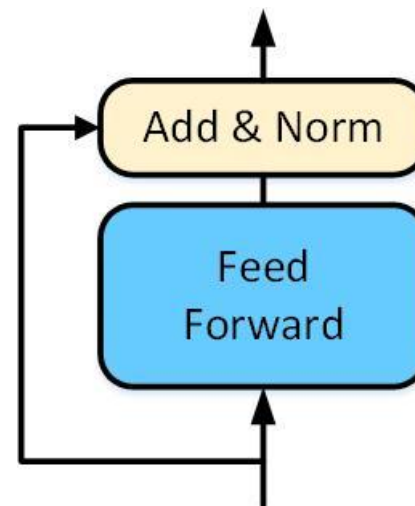
相加和归一化

□ 归一化的动机

- Transformer 中批次大小通常很小或变化大
- Batch Normalization 在此场景下不稳定

□ 归一化的效果

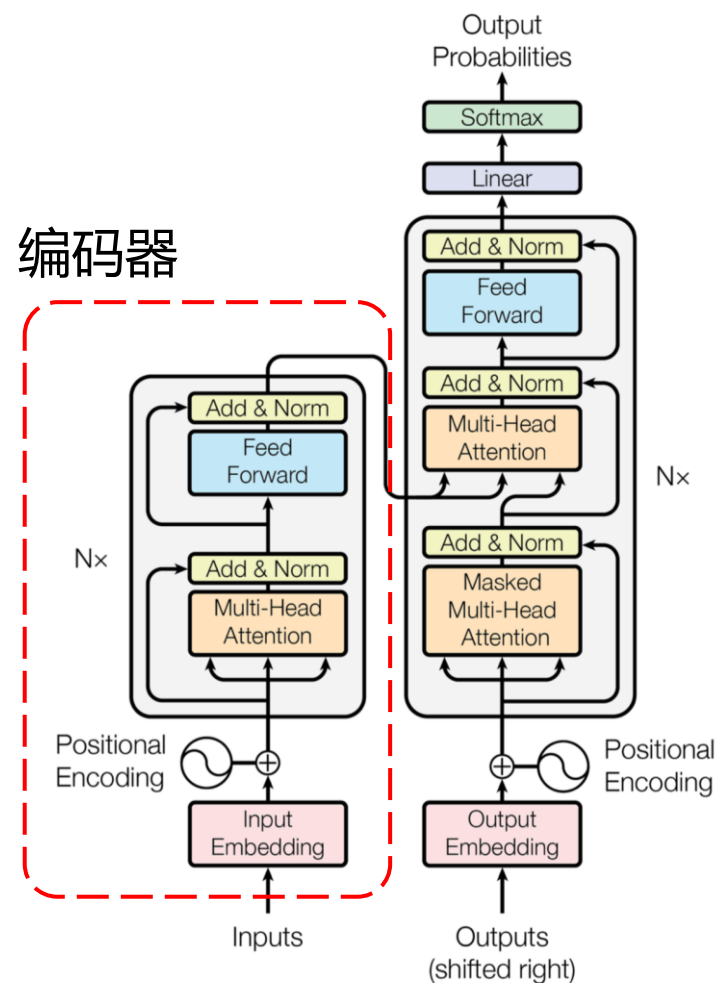
- 稳定中间激活分布，减少内部协变量偏移
- 缩短收敛时间，提高训练稳定性
- 结合残差连接，进一步提升性能



$$LN(x) = \frac{x - \mu}{\sigma} \odot \gamma + \beta$$

Transformer编码器的总结

- **输入处理**: Token Embedding 与位置编码相加
- **堆叠结构**: 通常多层相同的编码层, 并且每层输出维度与输入一致
- **编码层组件**: 多头自注意力 → 相加和归一化 → 前馈网络 → 相加和归一化
- **最终输出**: 每个位置的上下文表示, 可供解码或下游任务使用



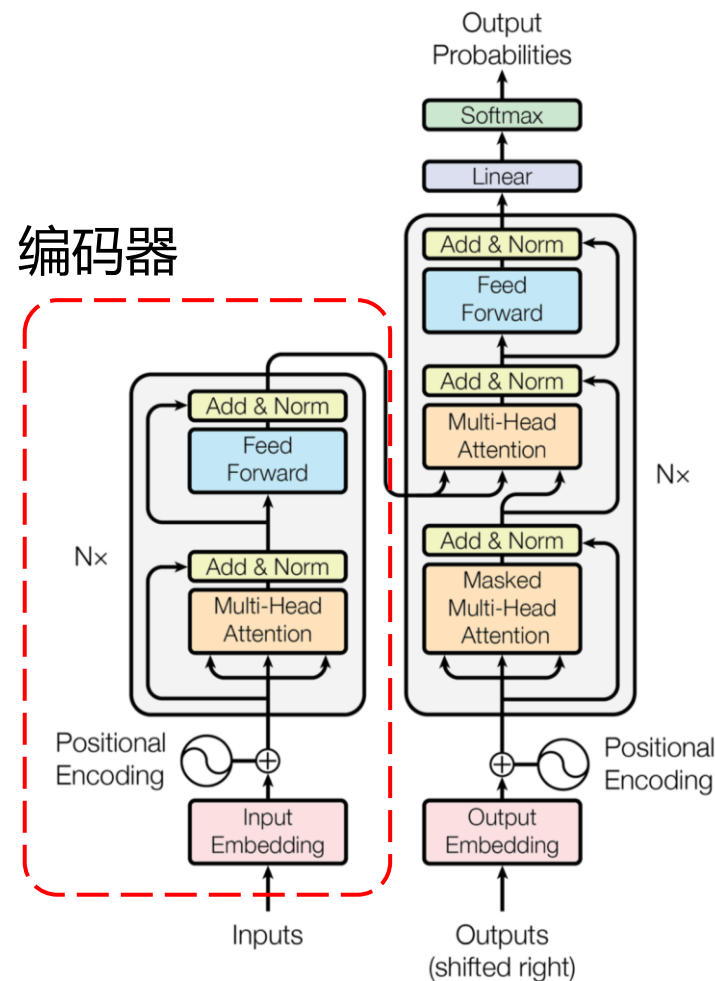
Transformer编码器的总结

□ 多头注意力机制是关键

- 自注意力机制用于集中注意力整合所有输入单元的重要信息
- 多头机制用于从不同角度对重要信息进行建模

□ 位置编码用于处理输入的顺序信息

- 所有输入单元均可见
- 正余弦函数编码或可学习向量，与词向量相加，保留序列的先后信息



Transformer编码器的总结

□ 位置前馈网络 (Feed-Forward Network)

- 两层全连接 + ReLU 激活 $FFN(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$
- 在每个位置独立作用

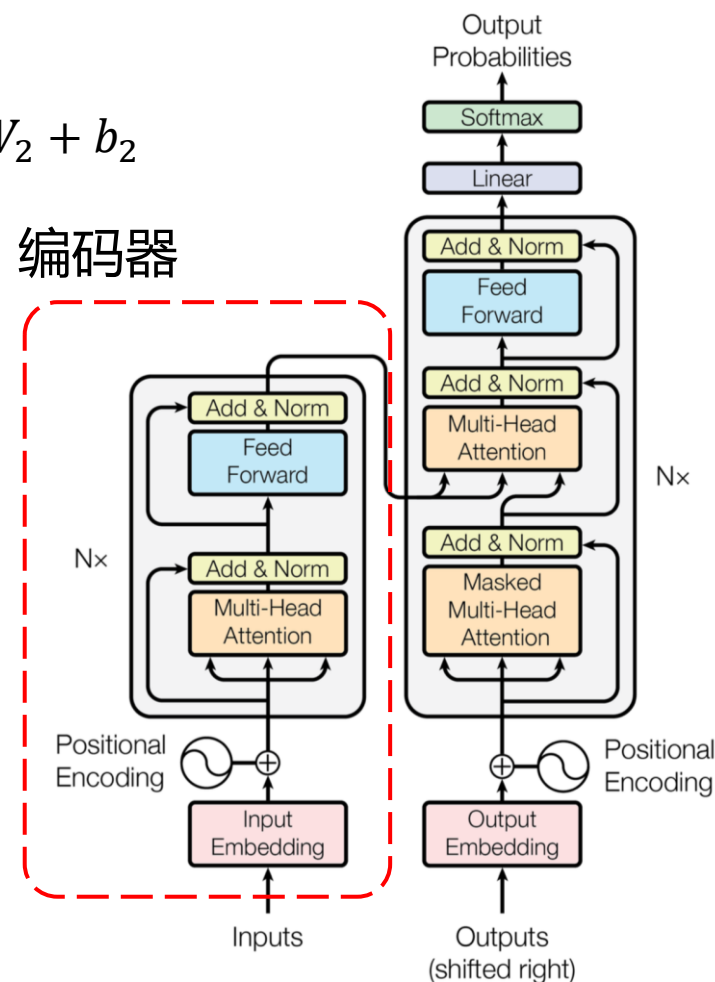
□ 残差连接 (Residual Connection)

- 输入 x 与子层输出相加 $output = \text{Sublayer}(x) + x$
- 保持梯度通道畅通, 子层专注学习增量

□ 层归一化 (Layer Normalization)

- 对单个样本的所有特征维度做归一化
- 稳定激活分布, 加快收敛并提升训练稳定性

$$LN(x) = \frac{x - \mu}{\sigma} \odot \gamma + \beta$$



01

Transformer的整体架构

02

Transformer的编码器

03

Transformer的解码器

本课重点

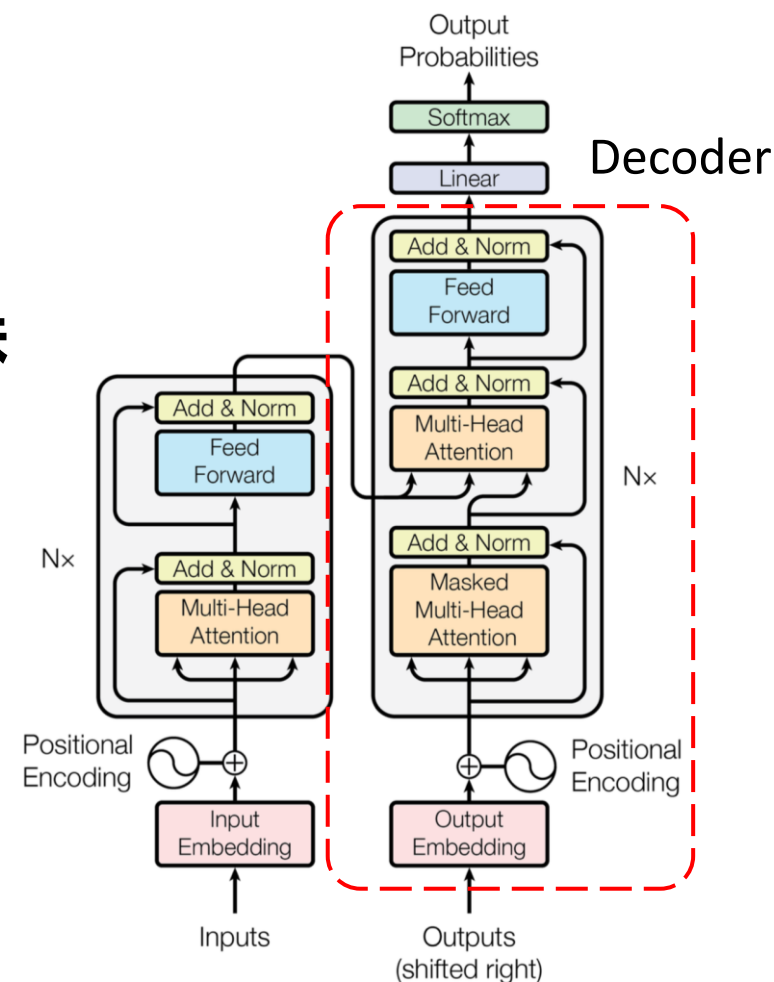
Transformer 解码器

□ 与 RNN 类似，Transformer 解码器在推理中在每个时间步骤预测一个token

□ 解码器的输入是任务的现有输出（右移），这意味着解码器只能看到当前时间步骤之前的内容

□ 如何对此进行建模？

□ 我们需要屏蔽当前时间步骤之后的输出：掩码多头注意力



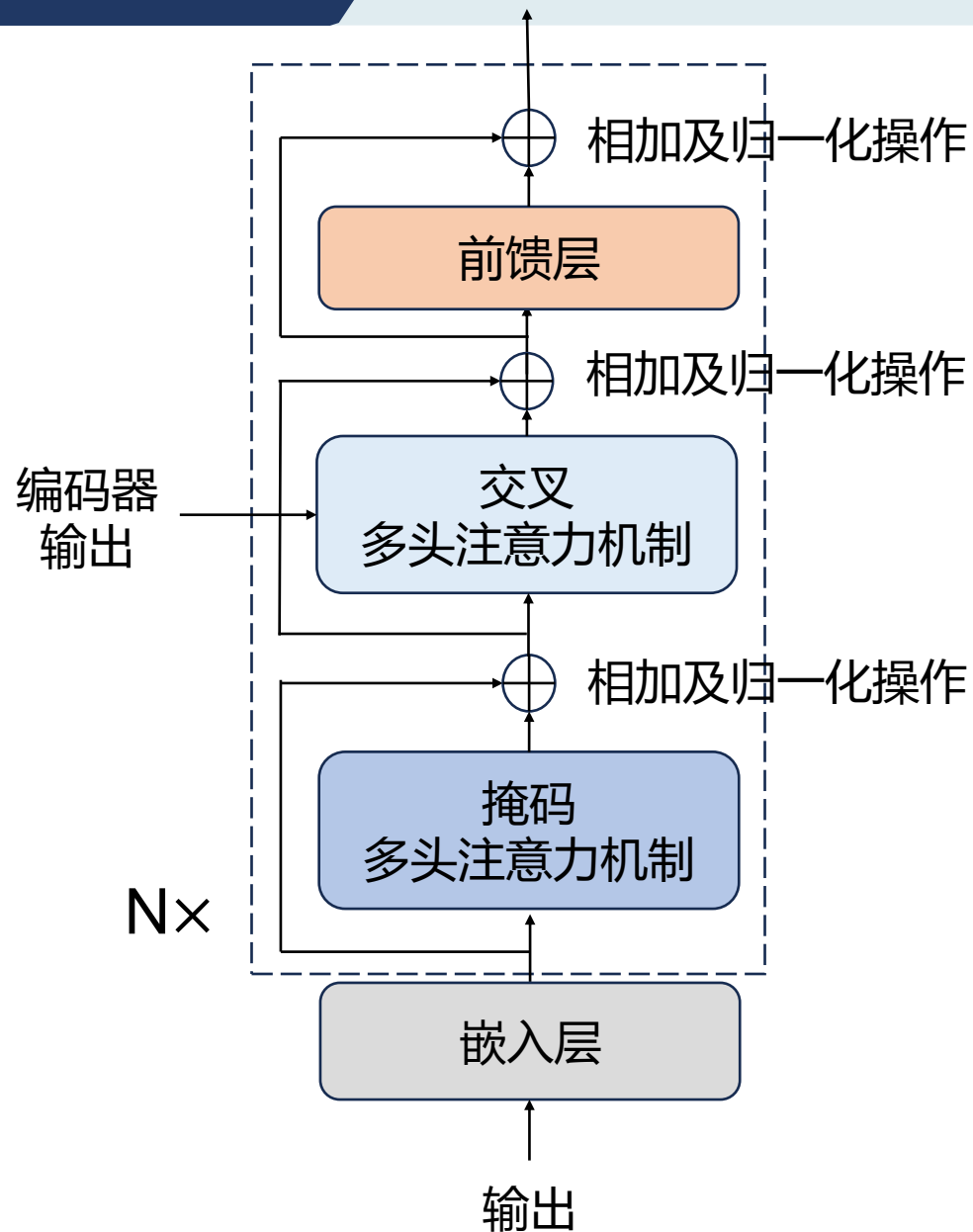
Transformer 解码器

□ Transformer解码器包括以下结构

- 嵌入层
- 掩码多头注意力机制
- 交叉多头注意力机制
- 相加及归一化操作

□ 与Transformer编码器的差别

- 掩码机制
- 交叉多头注意力

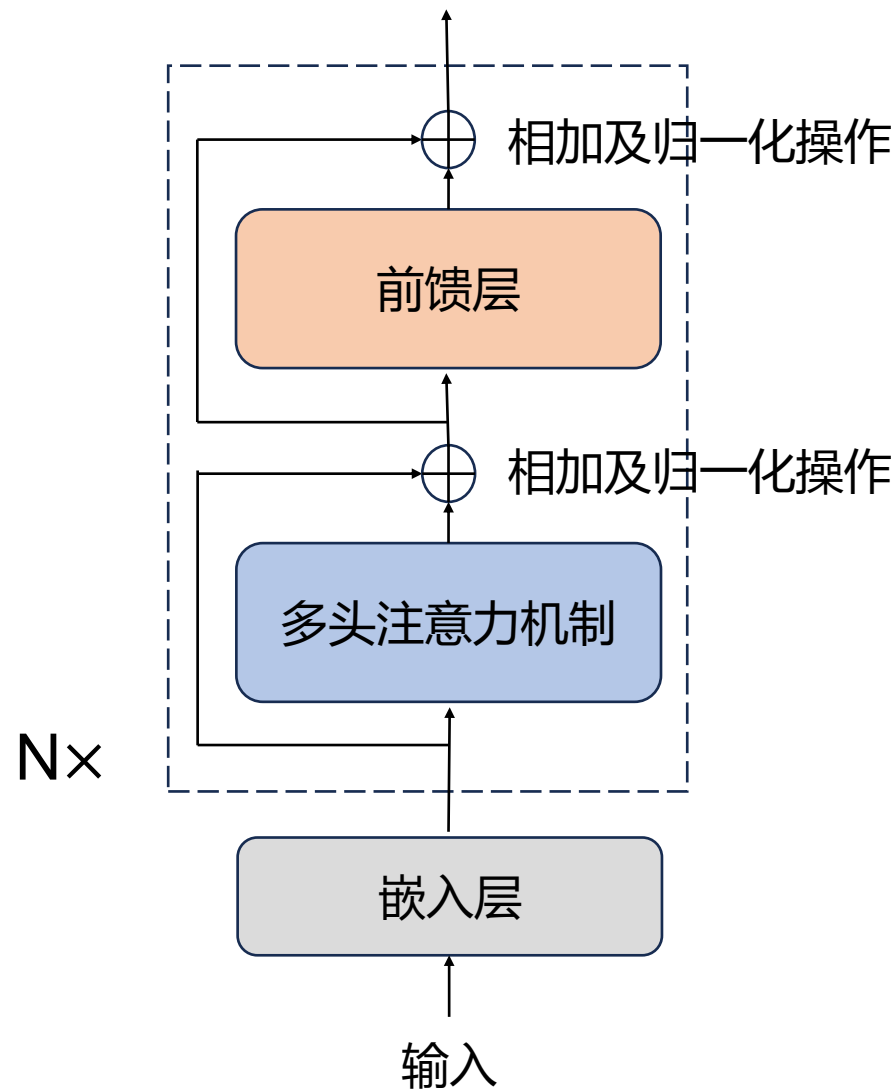


□ 掩码多头注意力的动机

- 在解码时，我们需要保证当前位置只能看到已生成的前序输出，避免“偷看”未来信息。
- 保持并行计算：虽然是序列生成，仍希望利用并行化优势，通过矩阵运算一次性完成多头计算。

□ 掩码多头注意力的实现思路

- 构造掩码矩阵 (Mask)：对于长度为 T 的序列，生成一个 $T \times T$ 的上三角掩码矩阵，掩盖 (i,j) 中 $j > i$ 的位置。



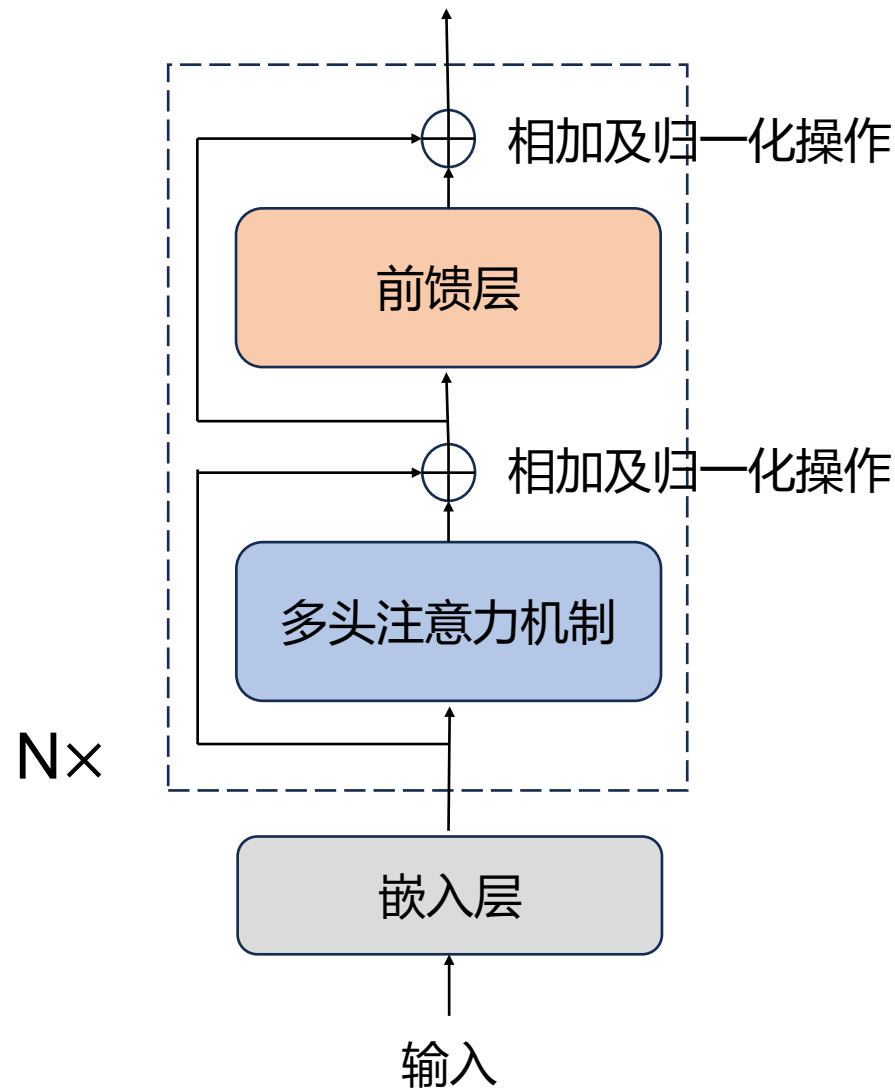
Transformer 解码器

□ 交叉多头注意力的动机

- 融合编码器信息：解码器在每一步生成时，需要参考整个源序列的上下文表示，保证生成内容与输入对齐。
- 多角度关注：不同注意力头可以在编码器各层次、不同特征子空间中捕捉多样化的源语义依赖。

□ 实现思路

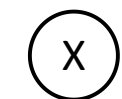
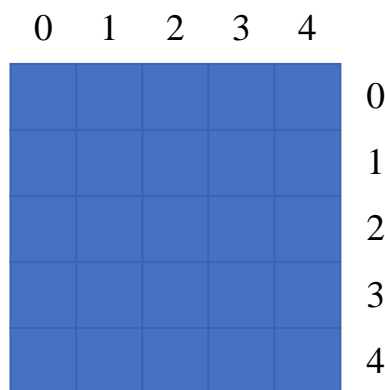
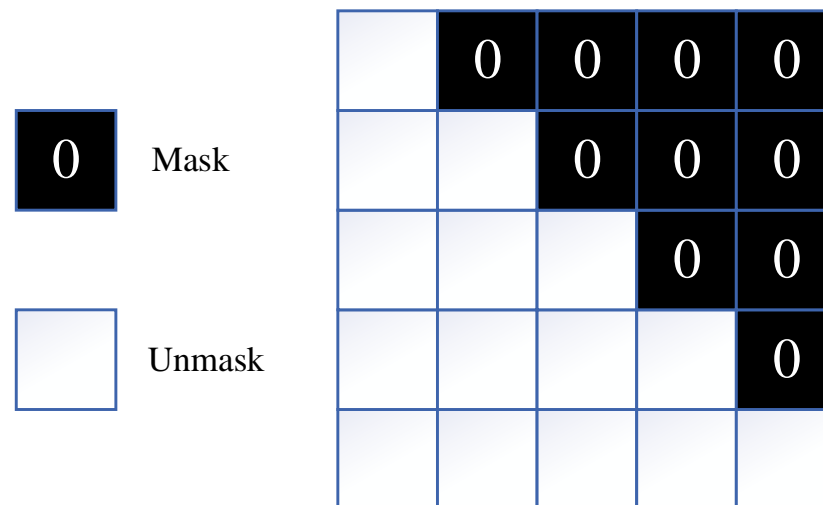
- Query 来自解码器前一层输出：Q = 解码器当前时刻的隐藏状态（包含已生成信息）。
- Key、Value 来自编码器输出：K, V = 编码器最后一层的全序列表示。



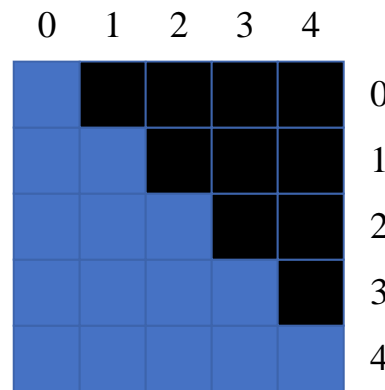
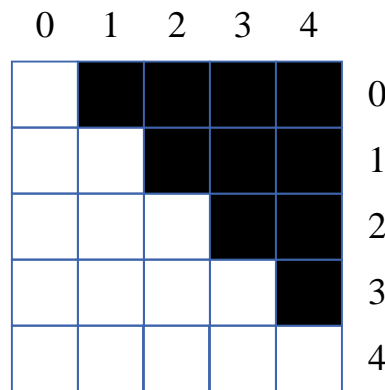
掩码多头注意力机制

□ 对于解码过程中看不到未来token的情况

□ 使用已知输出进行预测

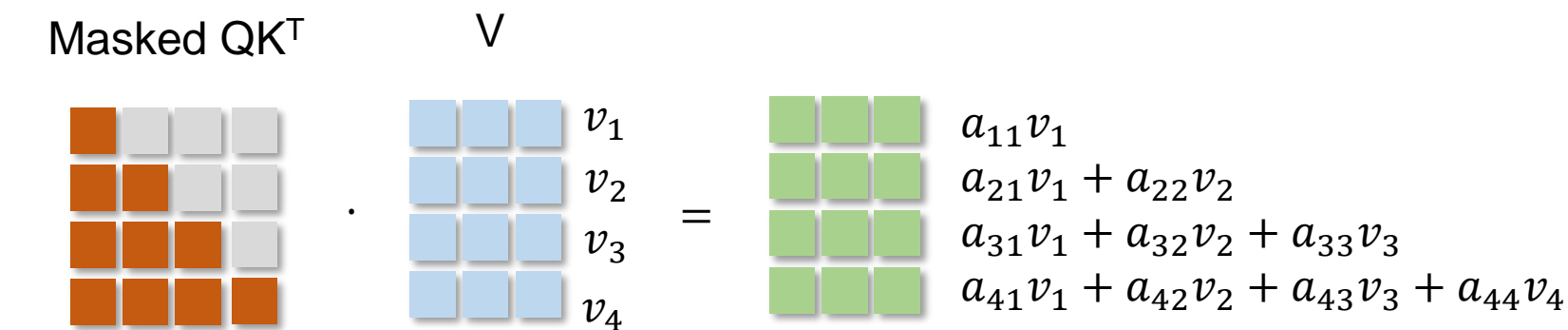
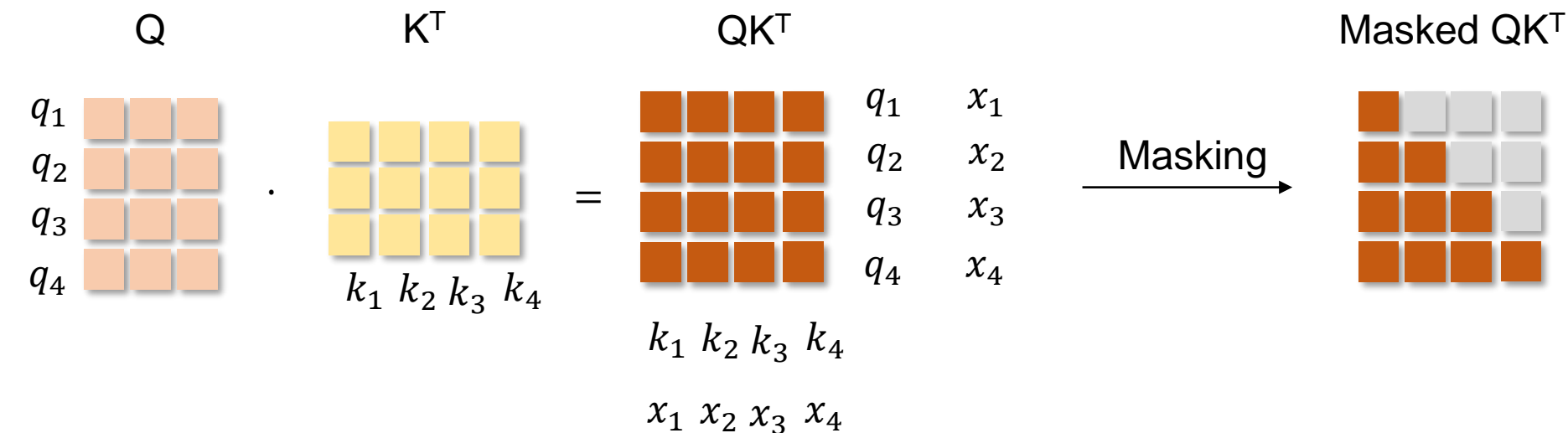


Dot-product



掩码多头注意力机制

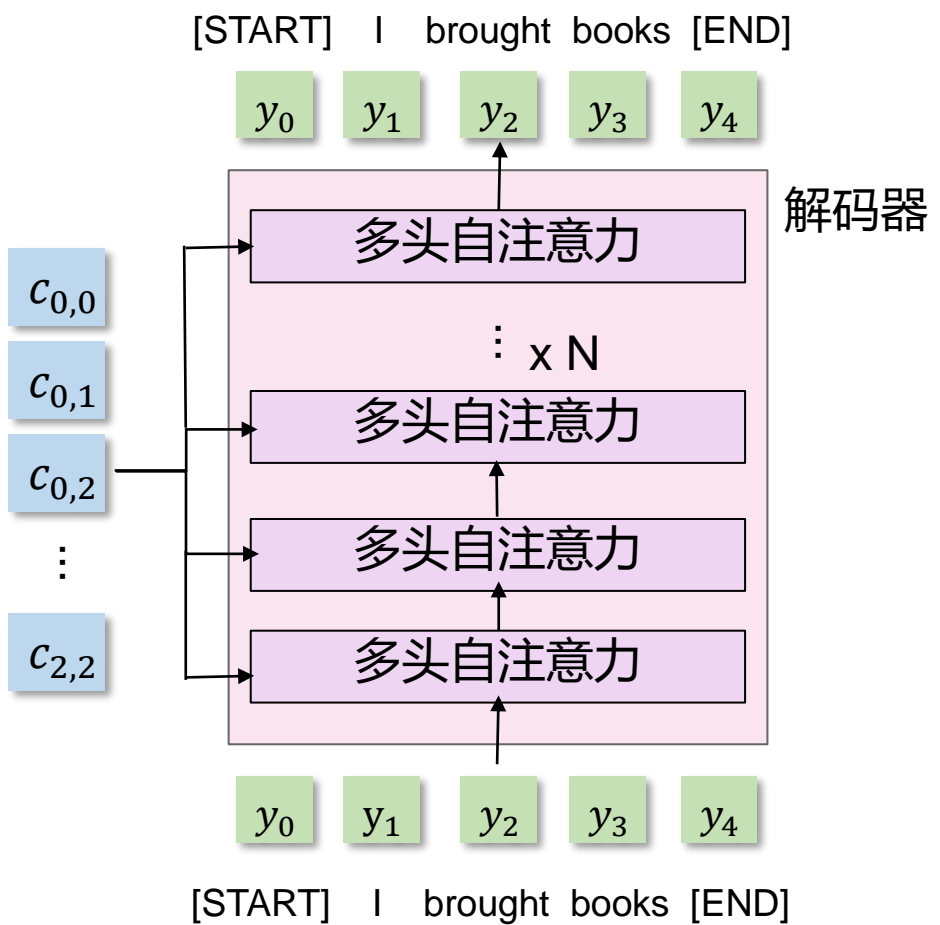
• 进一步理解



输出中每个向量，
不依赖其后面向量

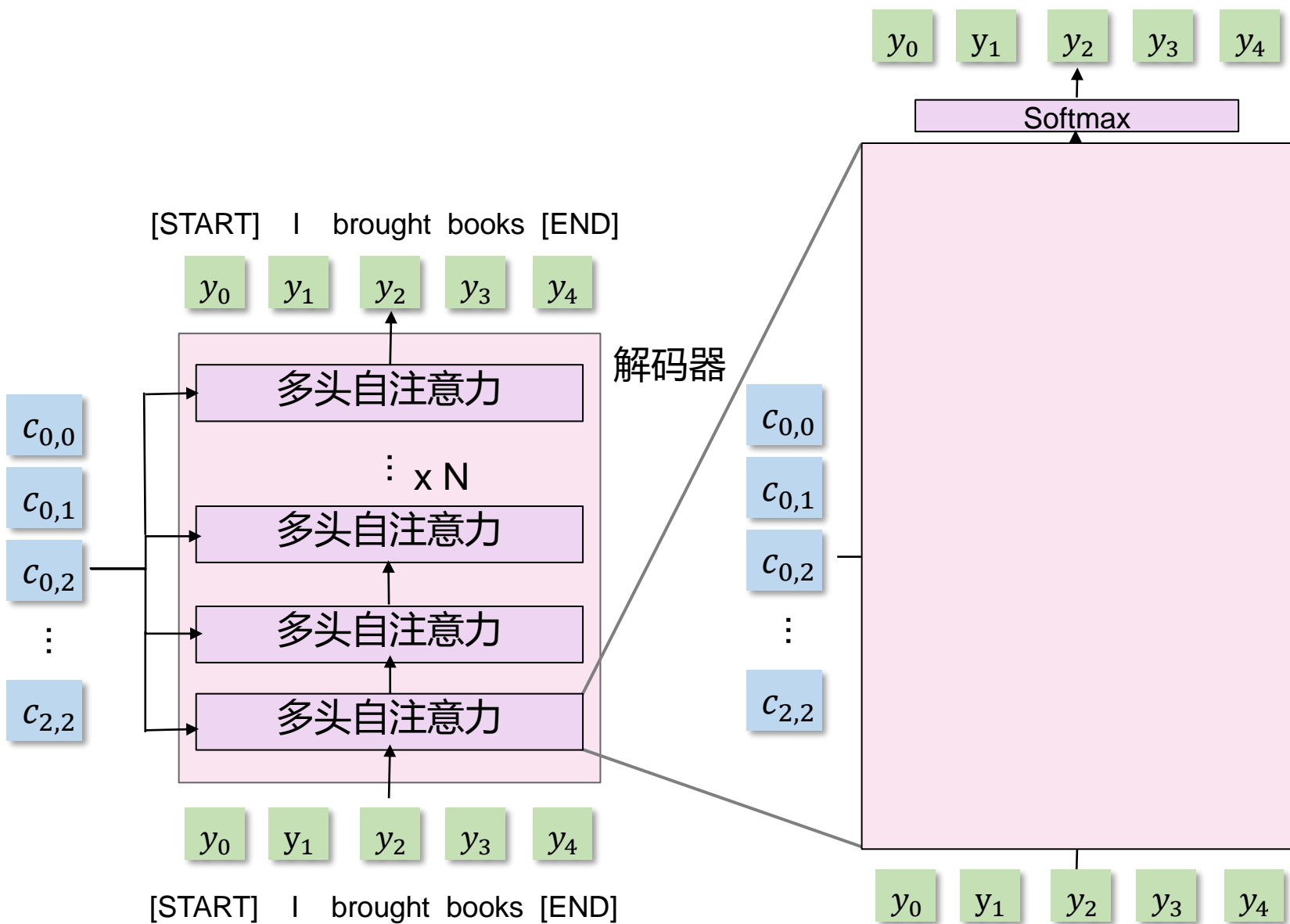
$$A = \{a_{i,j}\}$$
$$a_{i,j} = 0 \text{ if } j > i$$

解码器



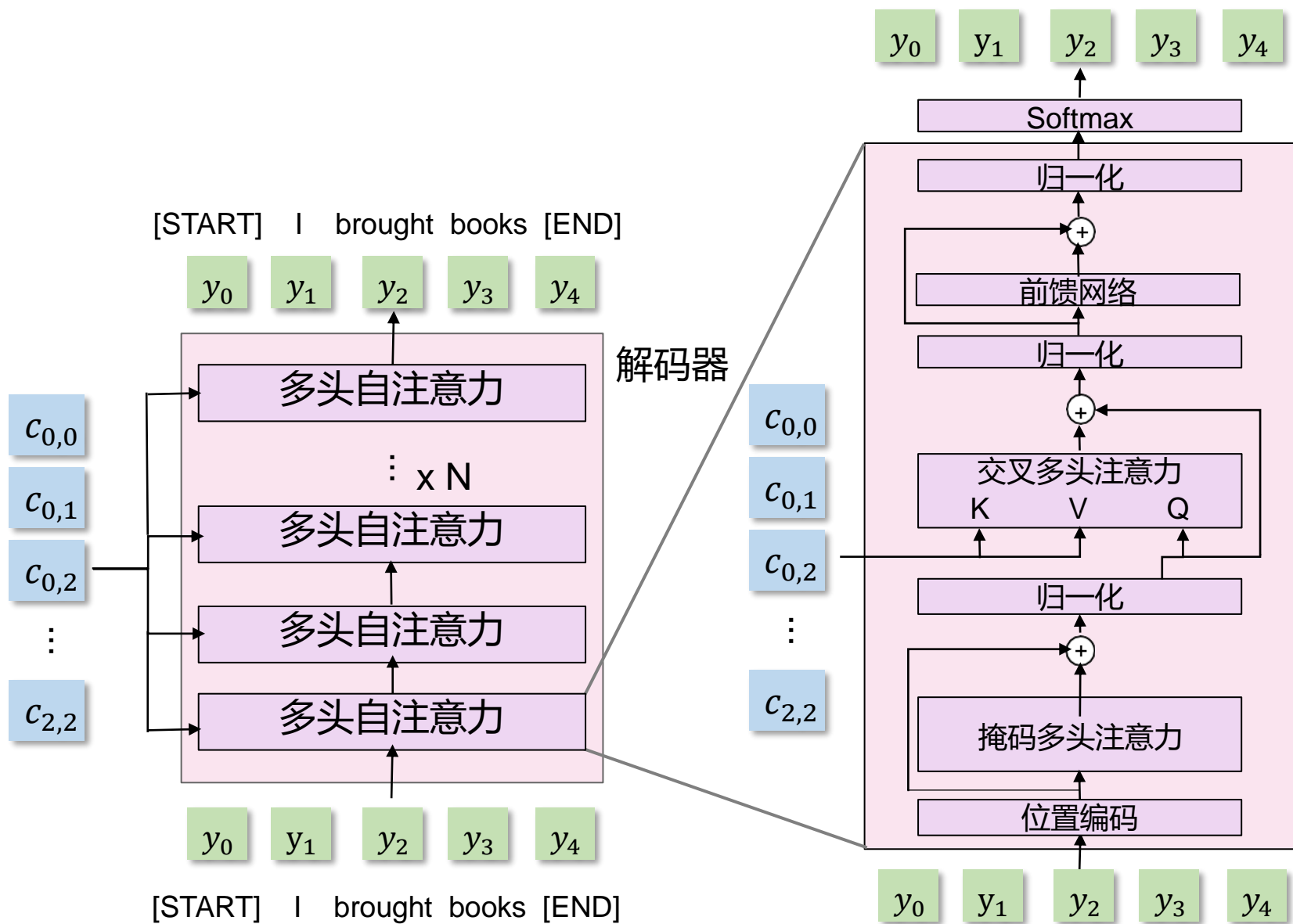
由N个解码器块组成。

解码器



让我们进入 Transformer
解码器模块

解码器



多头注意力模块关注
Transformer 编码器的输出。

为什么编码器的输入可以
作为K和V呢？

- 现有的输出（查询）决定了不同输入上下文向量 c_t 的重要性
- 上下文向量（值）的信息被整合以获得输出
- 来自编码器的信息被整合到解码过程中

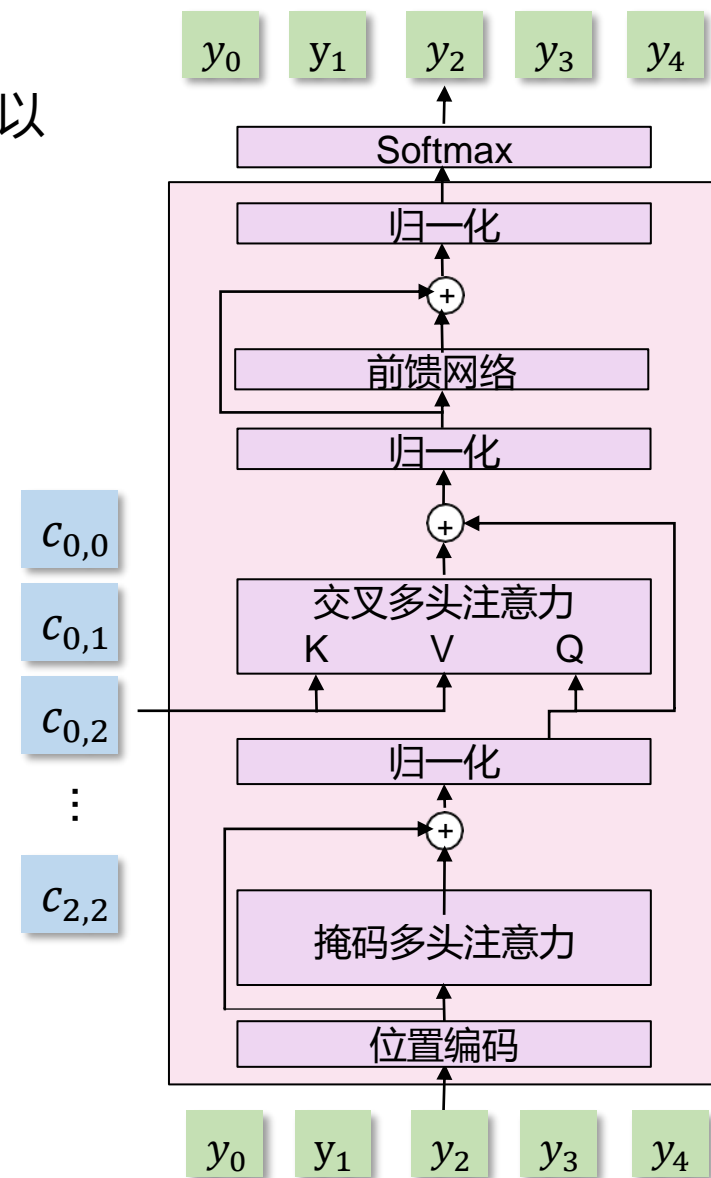
为什么编码器的输入可以
作为K和V呢？

$$\begin{aligned}q_i &= x_i \cdot W^Q \\k_t &= c_t \cdot W^K \\v_t &= c_t \cdot W^V\end{aligned}$$

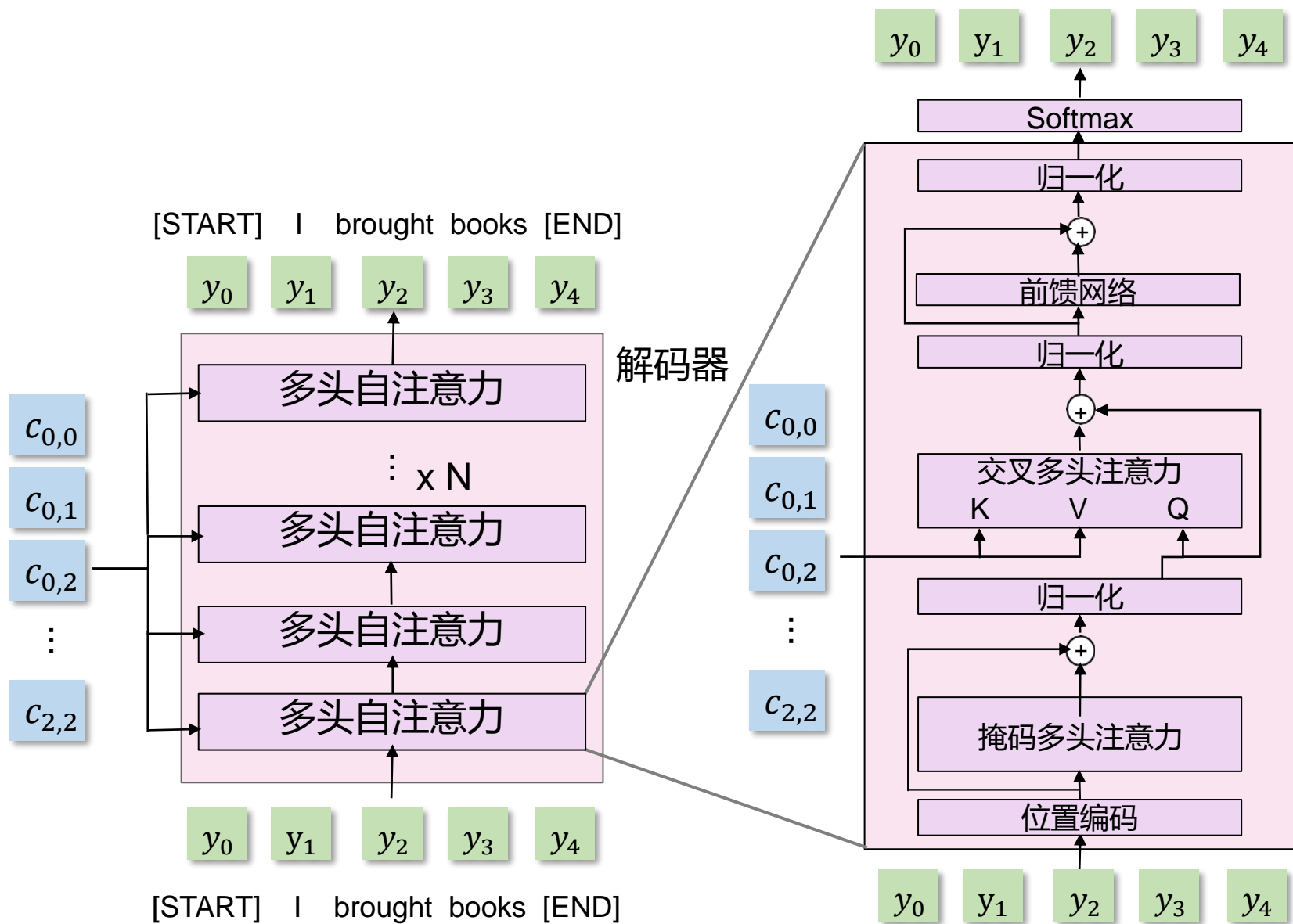
$$e_{i,t} = \sum_{j=1}^T q_{i,j} \cdot k_{j,t} = q_i \cdot k_t$$

$$a_{i,t} = \frac{\exp(e_{i,t})}{\sum_{t=1}^T \exp(e_{i,t})}$$

$$z_i = \sum_{t=1}^T a_{i,t} \cdot v_t$$



解码器



Transformer 解码器:

输入: 向量集 x 和上下文向量 c 。

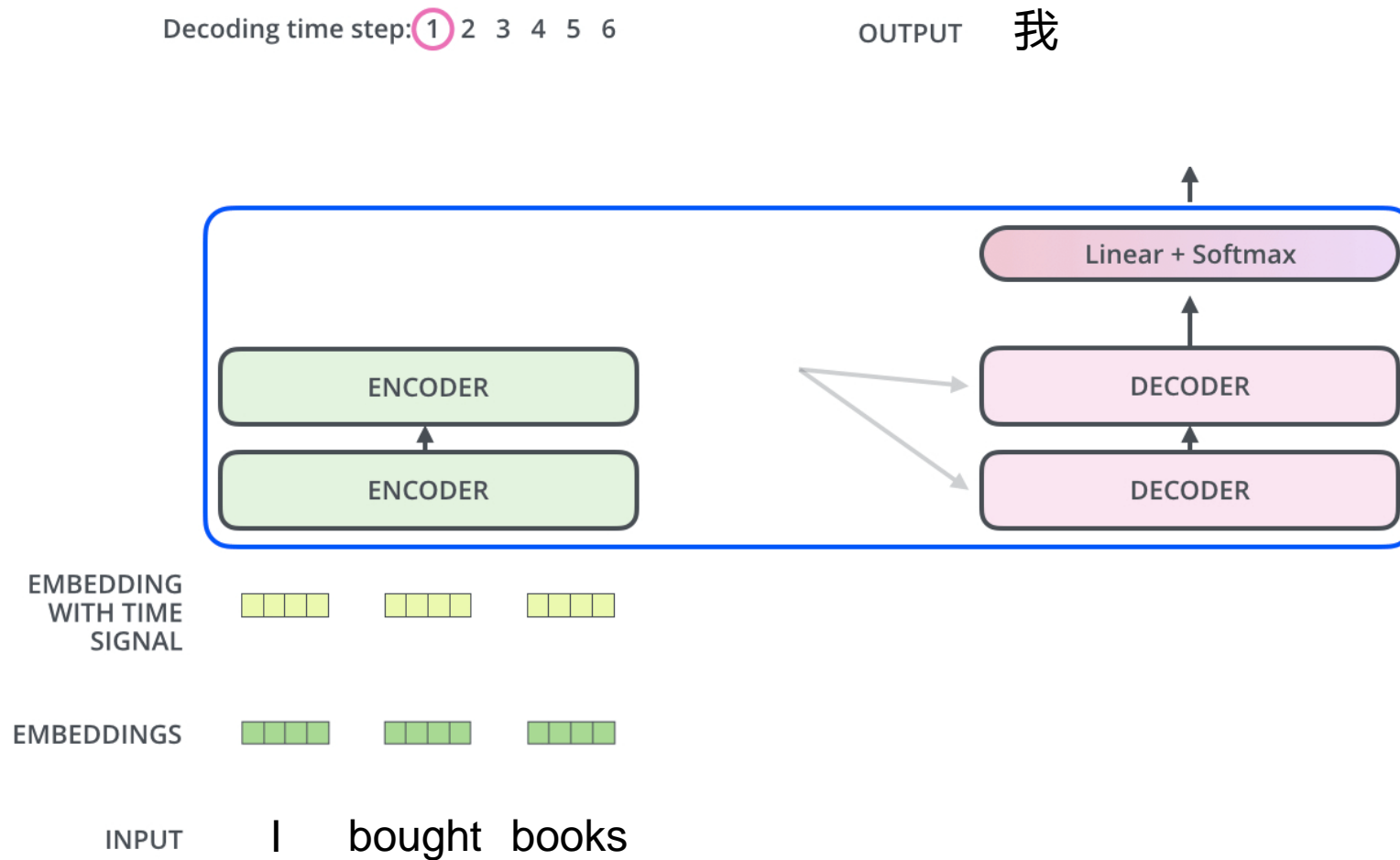
输出: 向量 y 。

掩码自注意力仅与过去的输入进行交互。

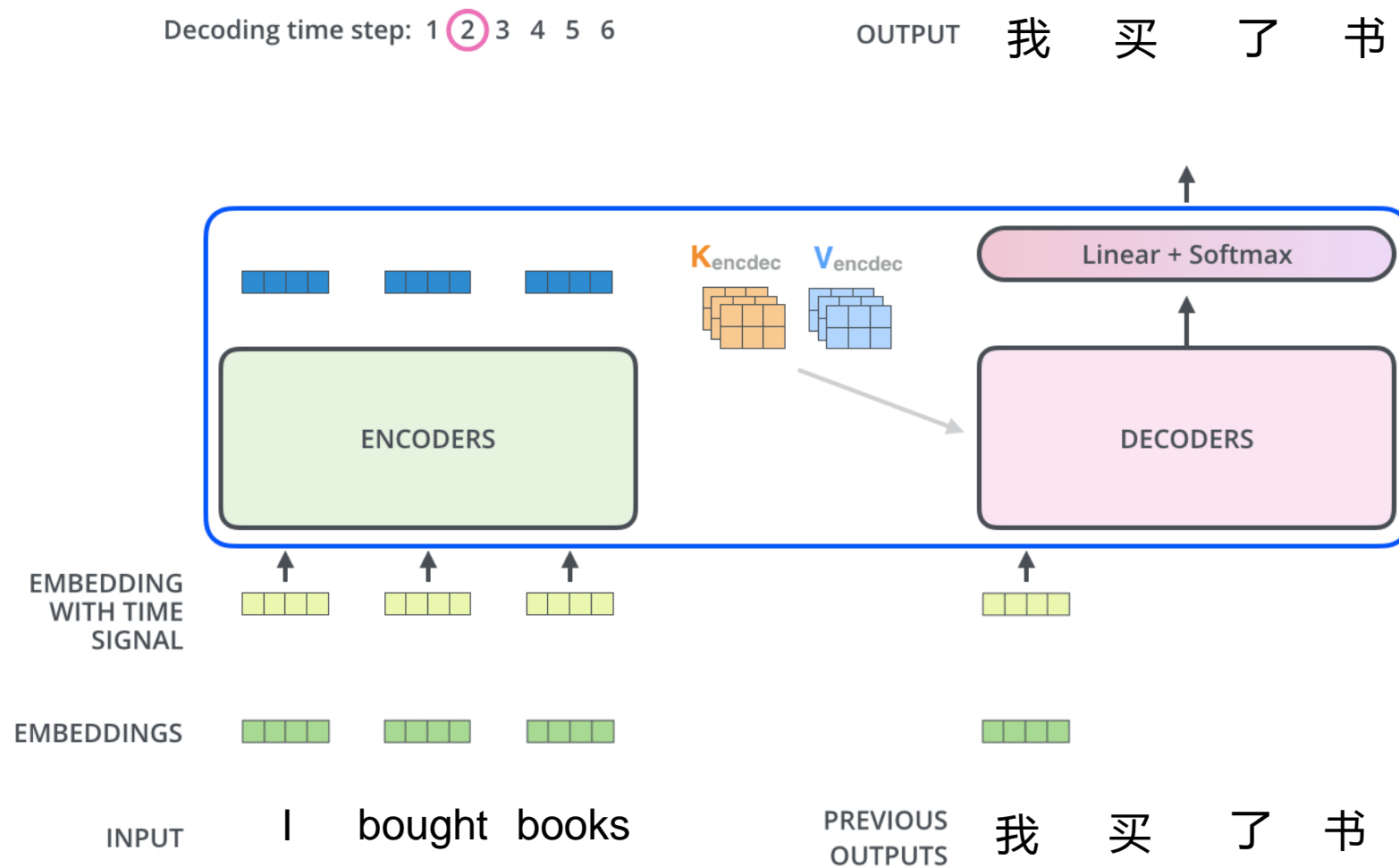
多头注意力模块不是自注意力模块。它关注编码器的输出。

高度可扩展、高度可并行，但内存使用率高。

解码器：逐层操作



解码器：逐层操作

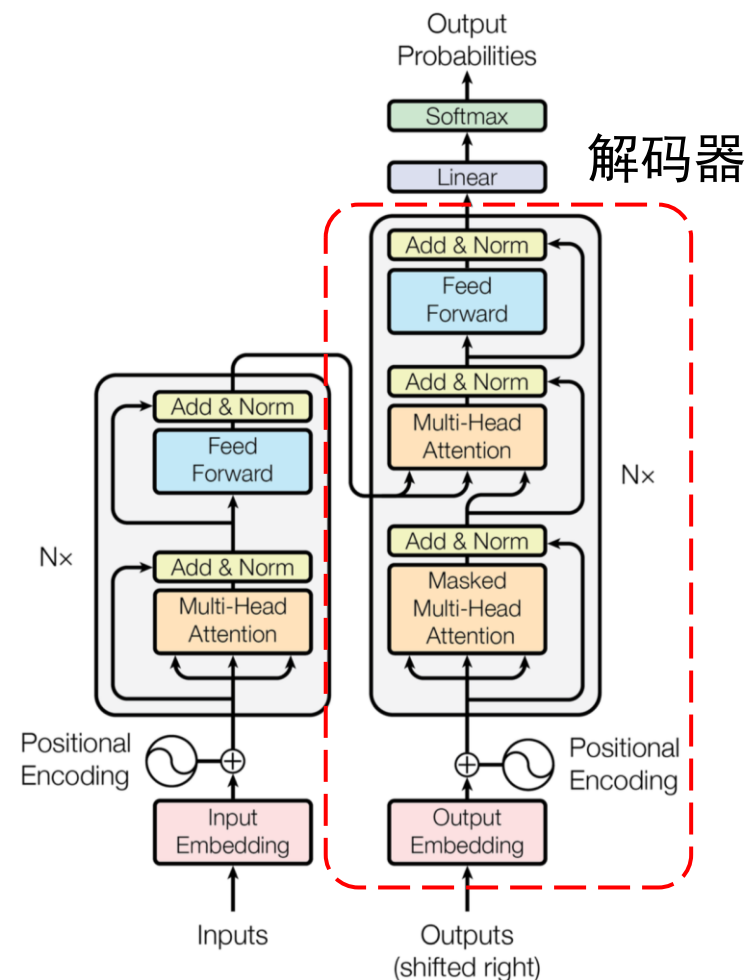


Transformer 解码器的总结

□ 整体上采用与Transformer编码器类似的架构

- 输入包括位置编码与词向量
- 基于多头注意力计算不同内容的权重
 - 使用掩码多头注意力计算上文内容的权重
 - 使用交叉注意力融合编码器信息
- 包括前馈层，相加和归一化机制
- 通过softmax层预测输出

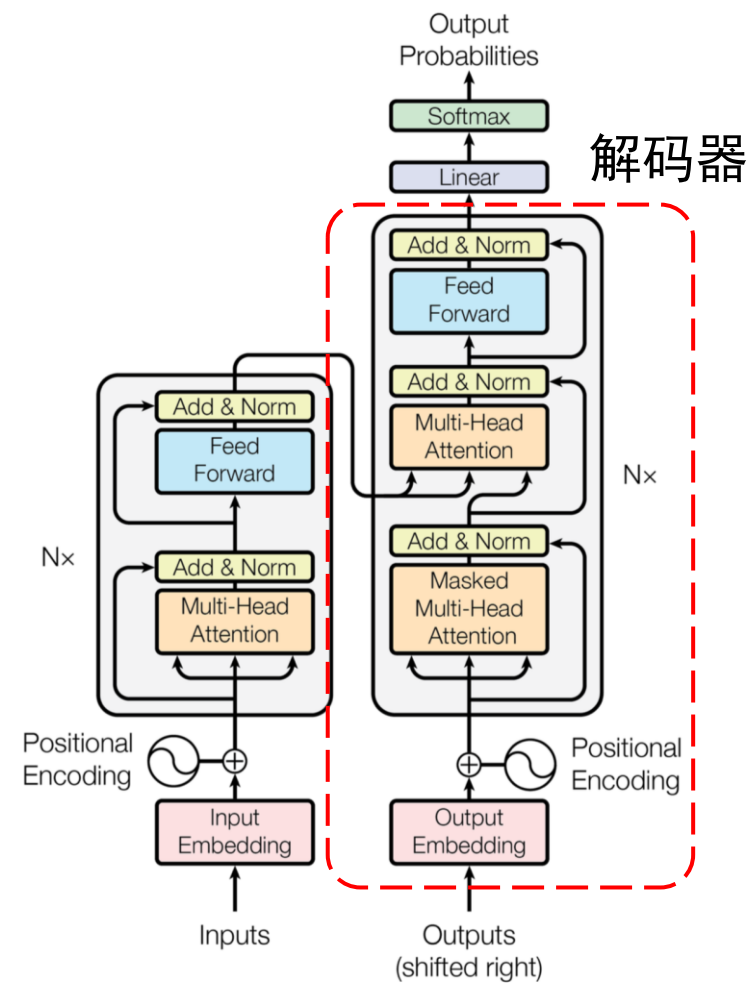
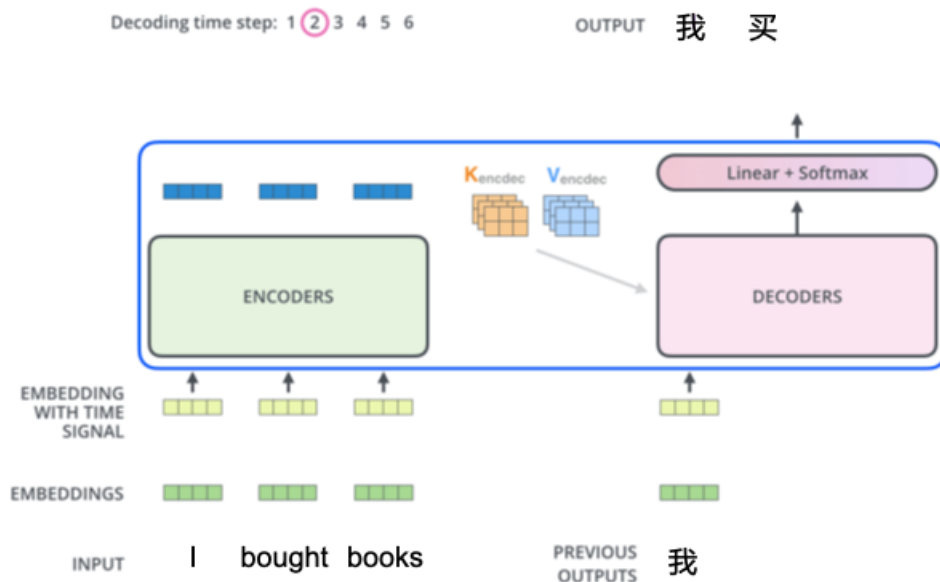
□ 通过堆叠实现层数的增加



Transformer 解码器的总结

Transformer 解码器的独特之处

- 使用掩码机制，实现单向信息编码
- 使用交叉注意力机制融合编码器信息
- 使用类似RNN的方法，自回归地逐词生成内容

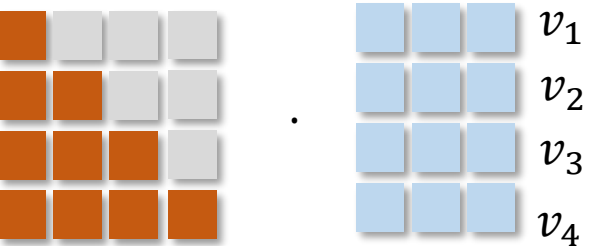


Transformer 解码器的总结

□ 掩码多头注意力机制

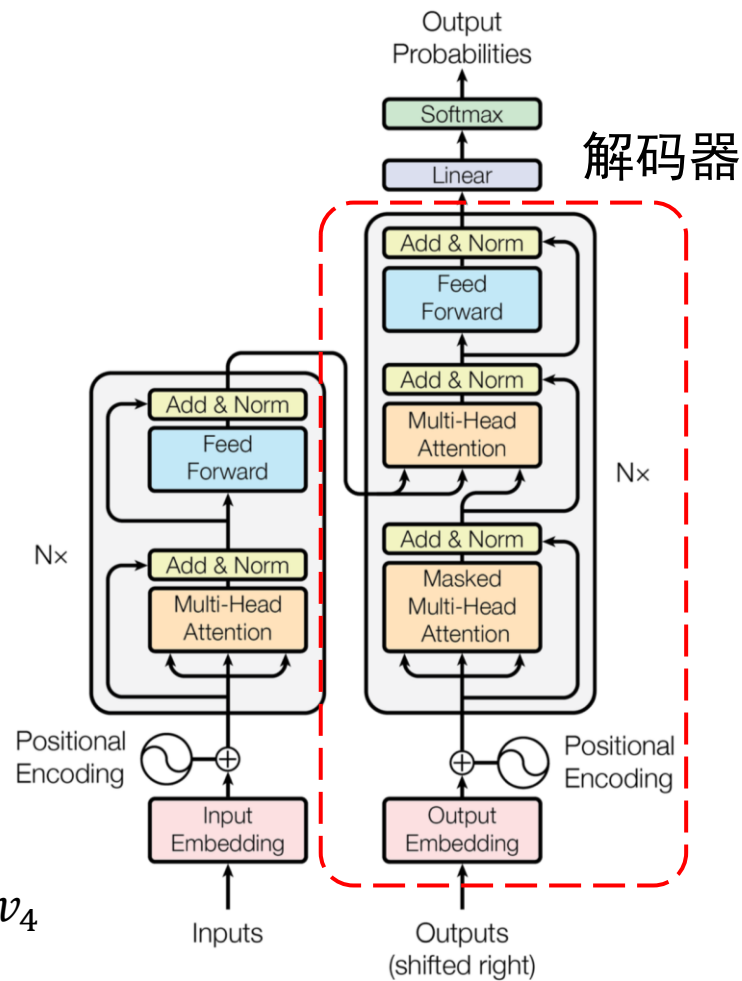
- 解码器一般用于生成任务，其任务特点决定了模型仅能获取单方向信息（即左边的上文信息）
- 通过掩码，模型无法访问未来的单词；同时，模型可以在训练中并行执行，增加训练效率

Masked QK^T V



$A = \{a_{i,j}\}$
 $a_{i,j} = 0 \text{ if } j > i$

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} a_{11}v_1 \\ a_{21}v_1 + a_{22}v_2 \\ a_{31}v_1 + a_{32}v_2 + a_{33}v_3 \\ a_{41}v_1 + a_{42}v_2 + a_{43}v_3 + a_{44}v_4 \end{bmatrix}$$

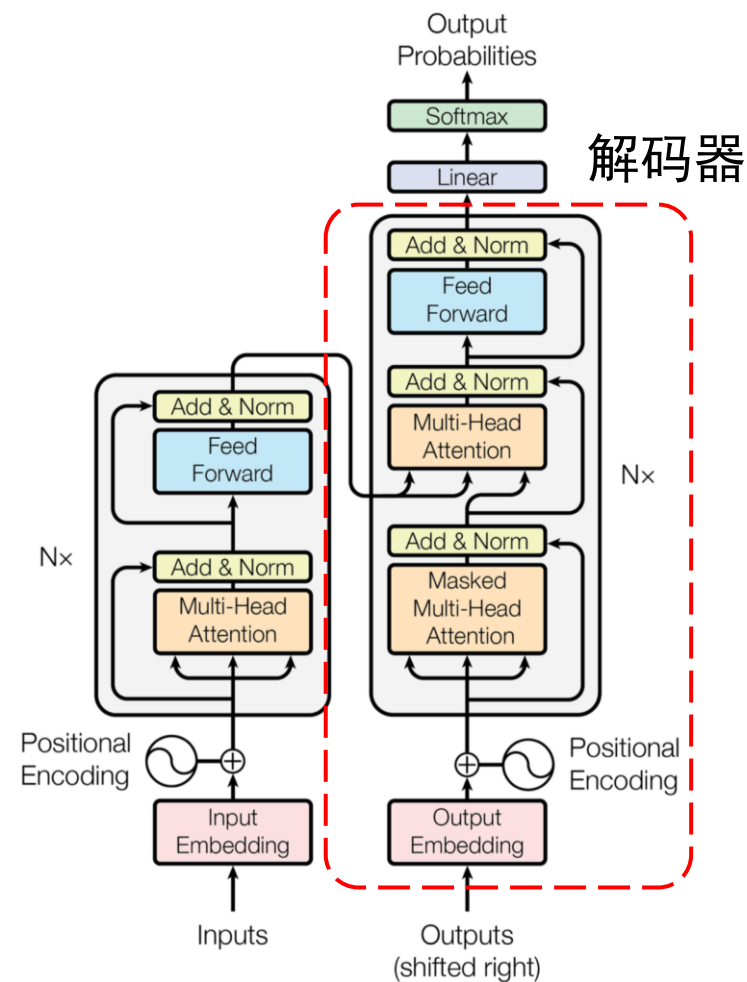


Transformer 解码器的总结

□ 使用交叉注意力融合编码器信息

- 交叉注意力出现在Transformer解码器中的每一层
- 来自编码器的信息用作 K 和 V，来自解码器的信息用作 Q
- 使用解码器中的信息定位编码器输出的重要向量

	自注意力	交叉注意力
Q来源	当前层自身	解码器自身
K、V来源	当前层自身	编码器输出



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 in an encoder-decoder configuration. 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. 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.

□ 机器翻译数据集

- WMT 2014 英→德: 约 4.5M 句对, BPE 词汇量 $\approx 37\ 000$
- WMT 2014 英→法: 约 36M 句对, word-piece 词汇量 $\approx 32\ 000$

□ 模型参数

- Transformer 架构: 纯注意力机制 (无循环 / 卷积)
- 编码器和解码器各 6 层, 隐藏维度 512, 多头注意力 $h=8$

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.8	$2.3 \cdot 10^{19}$	

	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	

- Dan Jurafsky and James H. Martin. Speech and Language Processing (3rd ed. draft); Chapter 9. 2025
 - 系统介绍了Transformer的原理、优势以及局限

- Jesse Vig. “A Multiscale Visualization of Attention in the Transformer Model.” 2019.
 - 开发多尺度可视化工具，从不同层次和粒度解读 Transformer 的 attention 分布；
 - 演示如何用该工具检测模型偏差、识别关键注意力头并关联到模型行为，大幅提升自注意力的可解释性。

- Jay Alammar. The Illustrated Transformer.
 - 通过大量图示与动画直观解读Transformer各层的运作方式
 - 逐步展示了Query、Key、Value以及多头注意力的计算流程；
 - 通过可交互示例帮助理解位置编码和残差连接的设计意义；
 - 配套丰富的多语种翻译版本，方便不同背景的人群阅读
 - <https://jalammar.github.io/illustrated-transformer/>

- Transformer 的编码器和解码器各自发挥了怎样的作用？与BERT和GPT有什么关联？
- 在交叉注意力中，Q，K，V的来源一定要是Q来自解码器，K和V来自编码器吗？在其他场景下，是否可以有不同的设计？