AI Box

Transformer模型

《大语言模型》编写团队：李军毅

大模型神经网络的奠基之作

AI Box

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

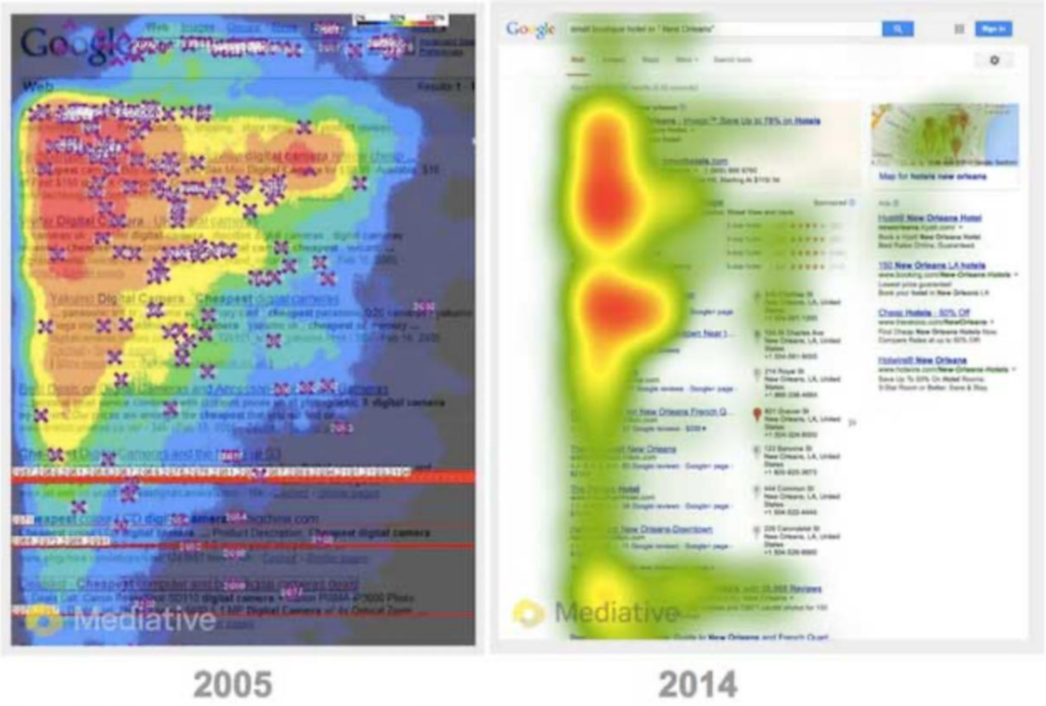
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Attention is all you need [PDF] neurips.cc

A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurips.cc

… Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and … Noam proposed scaled dot-product attention, multi-head …

Why Attention?



Source: The Evolution of Google Search Results Pages, Mediative, 2014

在搜索场景中，人们的目光往往会更加关注左上角的三角区域（即第一条搜索结果的位置）

A similar study conducted by Mediative in 2005 found that users tended to focus their gaze on the "Golden Triangle"—the top-left corner of a SERP where the first result was usually displayed.

The 2014 results were strikingly different from those a decade ago. The Golden Triangle has all but disappeared; instead, users tend scan more vertically and to vary their focus depending on what they are searching for.

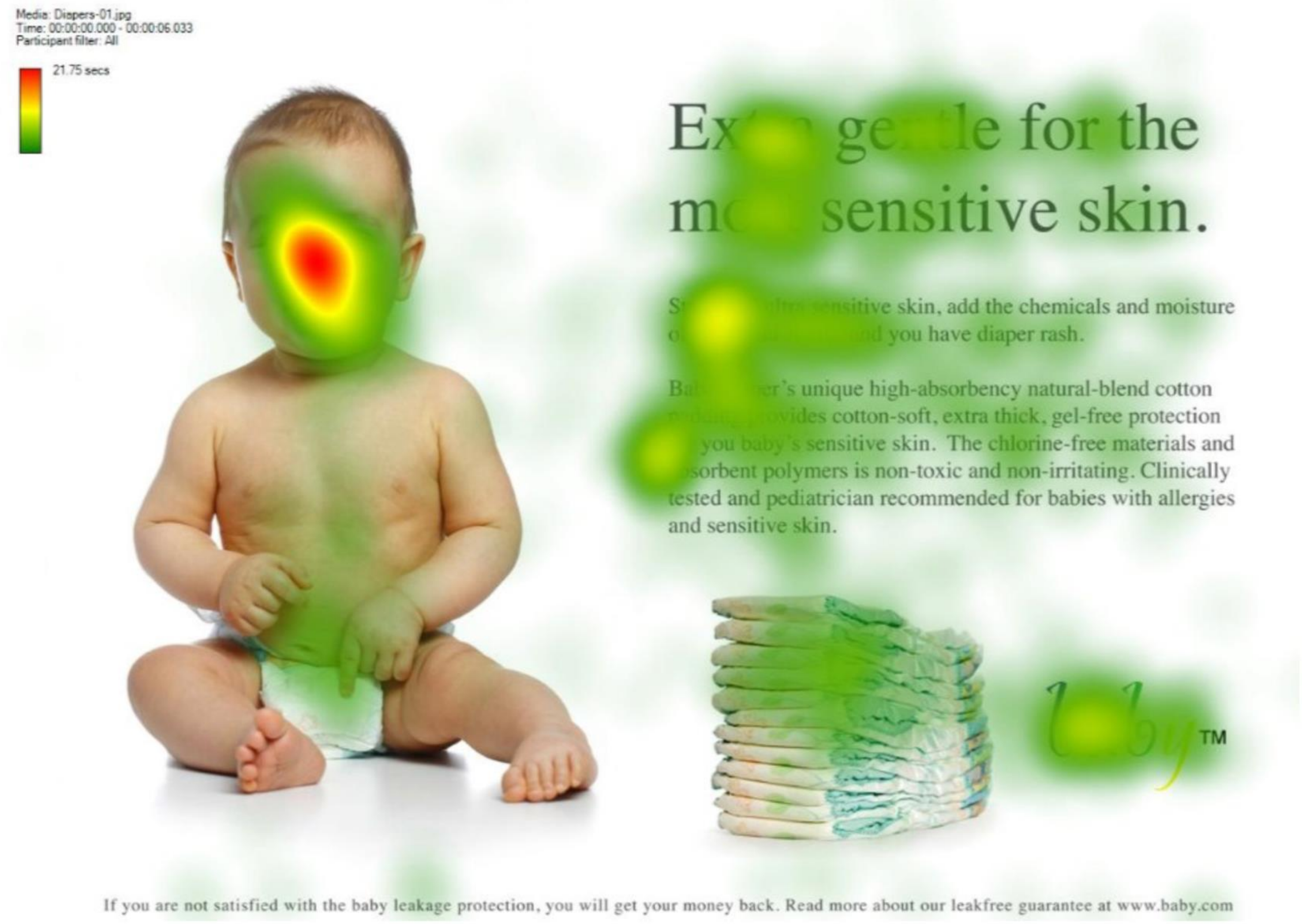
https://www.marketingprofs.com/charts/2014/26167/eye-tracking-study-how-users-view-google-search-result-pages

Why Attention?

在阅读过程中

https://commons.wikimedia.org/wiki/Category:Eye_tracking#/media/File:Heatmaps_und_AOIs.png

Why Attention?

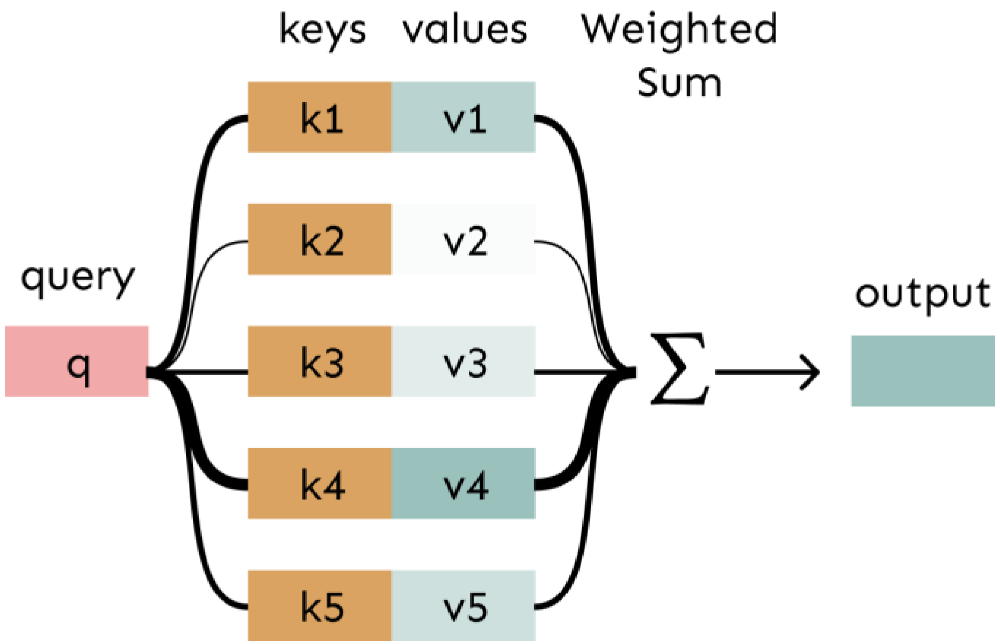


在**浏览图文信息**时

https://www.objectiveexperience.com/eye-tracking-ux-research/

注意力机制

➢ 可以视为一种基于相似度的查表



➢ 第一步：计算query与key相似度

$$e_{ij} = q_i^T k_j$$

➢ 第二步：相似度规范化

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

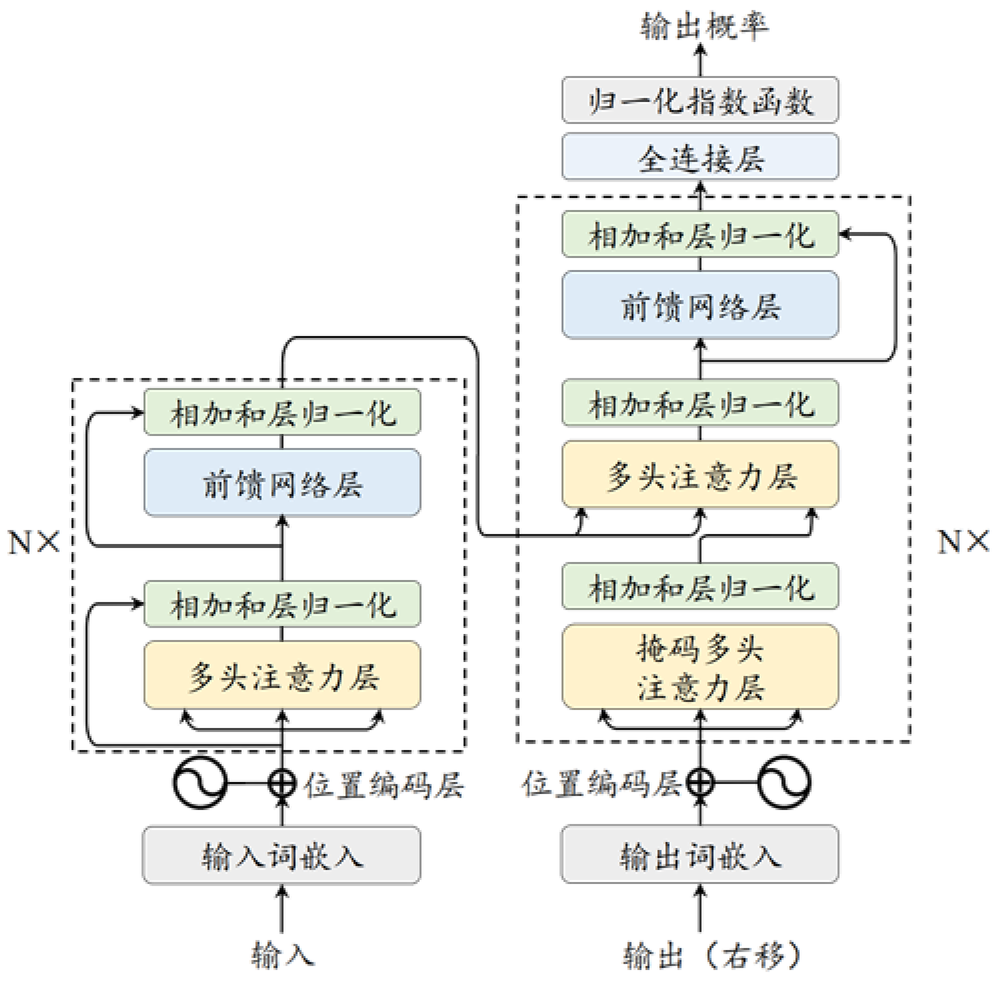
➢ 第三步：对value加权求和

$$o_i = \sum_j \alpha_{ij} v_i$$

https://web.stanford.edu/class/cs224n

Transformer模型

- 核心模块：注意力
  - Transformer 完全抛弃传统的CNN 和 RNN，整个网络结构完全由注意力机制组成
- 编码器-解码器结构
  - 编码器将输入序列变换为隐藏层特征
  - 解码器将隐藏层特征变换为输出序列



(a) 编码器 (b) 解码器

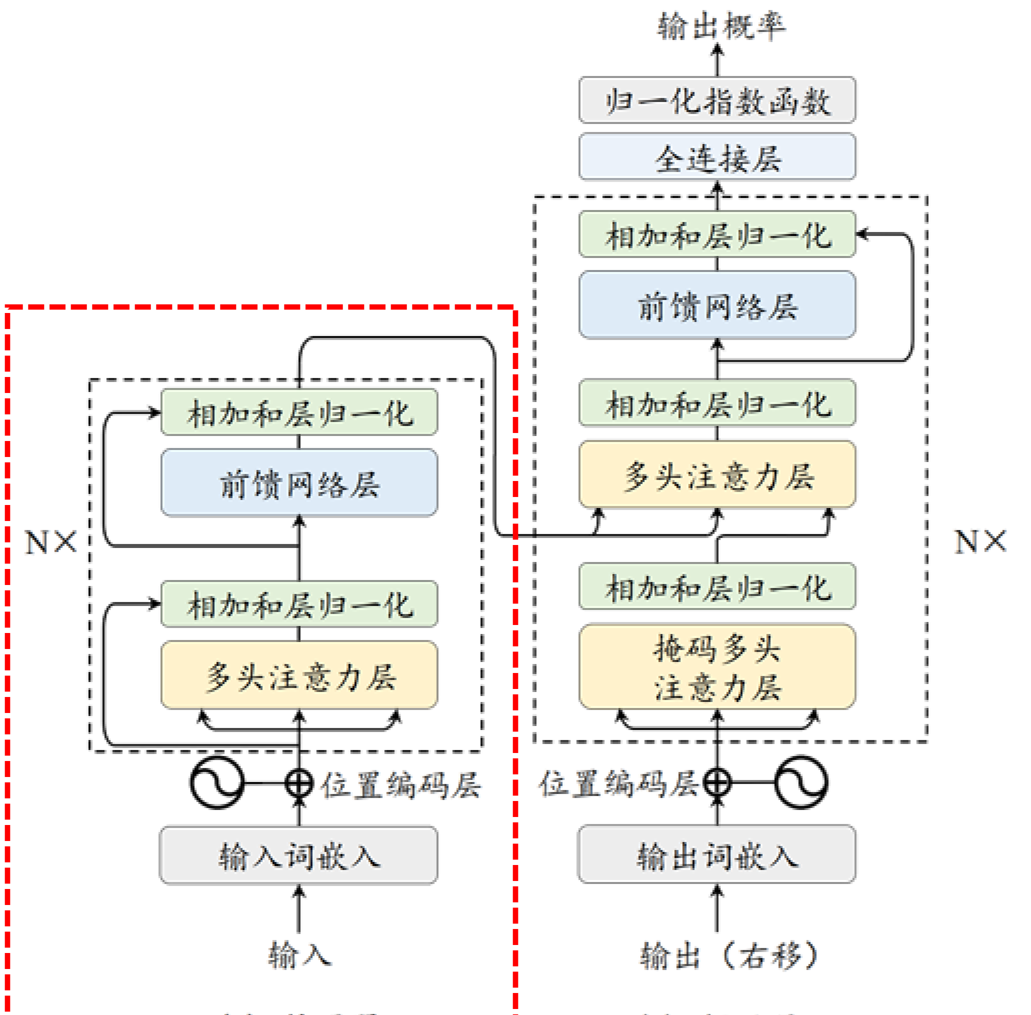
Transformer模型

- 编码器：将输入变换为隐藏层特征
  - N个堆叠的编码器层
    - 多头注意力
    - 前馈网络
    - 残差连接和层归一化

$$X_l' = \mathrm{LayerNorm}(\mathrm{MHA}(X_{l-1}) + X_{l-1}),$$

$$X_l = \mathrm{LayerNorm}(\mathrm{FFN}(X_l') + X_l'),$$

X_{l-1}：编码器第$l-1$层的输出



(a) 编码器 (b) 解码器

Transformer模型

AI Box

- 解码器：将隐藏层特征变换为自然语言序列
  - N个堆叠的解码器层
    - （掩码）多头注意力
    - 前馈网络
    - 残差连接和层归一化

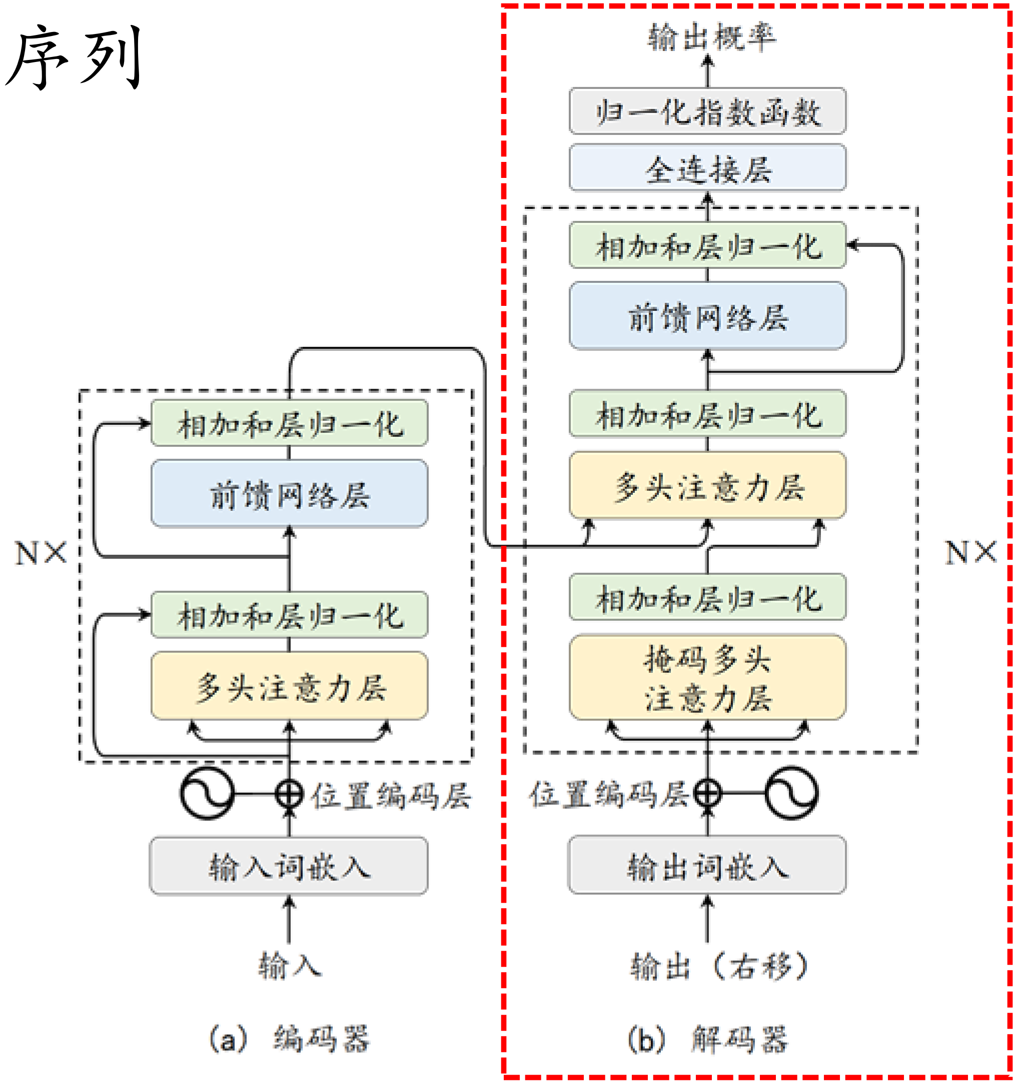
$$Y'_l = \mathrm{LayerNorm}(\mathrm{MaskedMHA}(Y_{l-1}) + Y_{l-1}),$$

$$Y''_l = \mathrm{LayerNorm}(\mathrm{CrossMHA}(Y'_l, X_L) + Y'_l),$$

$$Y_l = \mathrm{LayerNorm}(\mathrm{FFN}(Y''_l) + Y''_l),$$

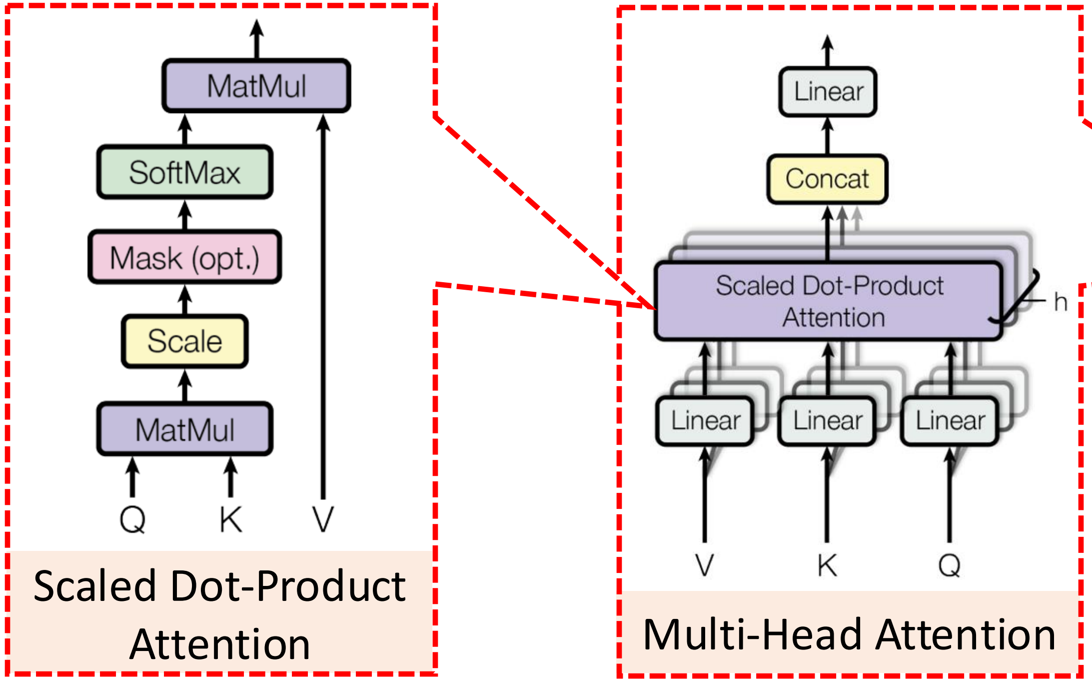
Y_{l-1}：解码器第 $l-1$ 层的输出

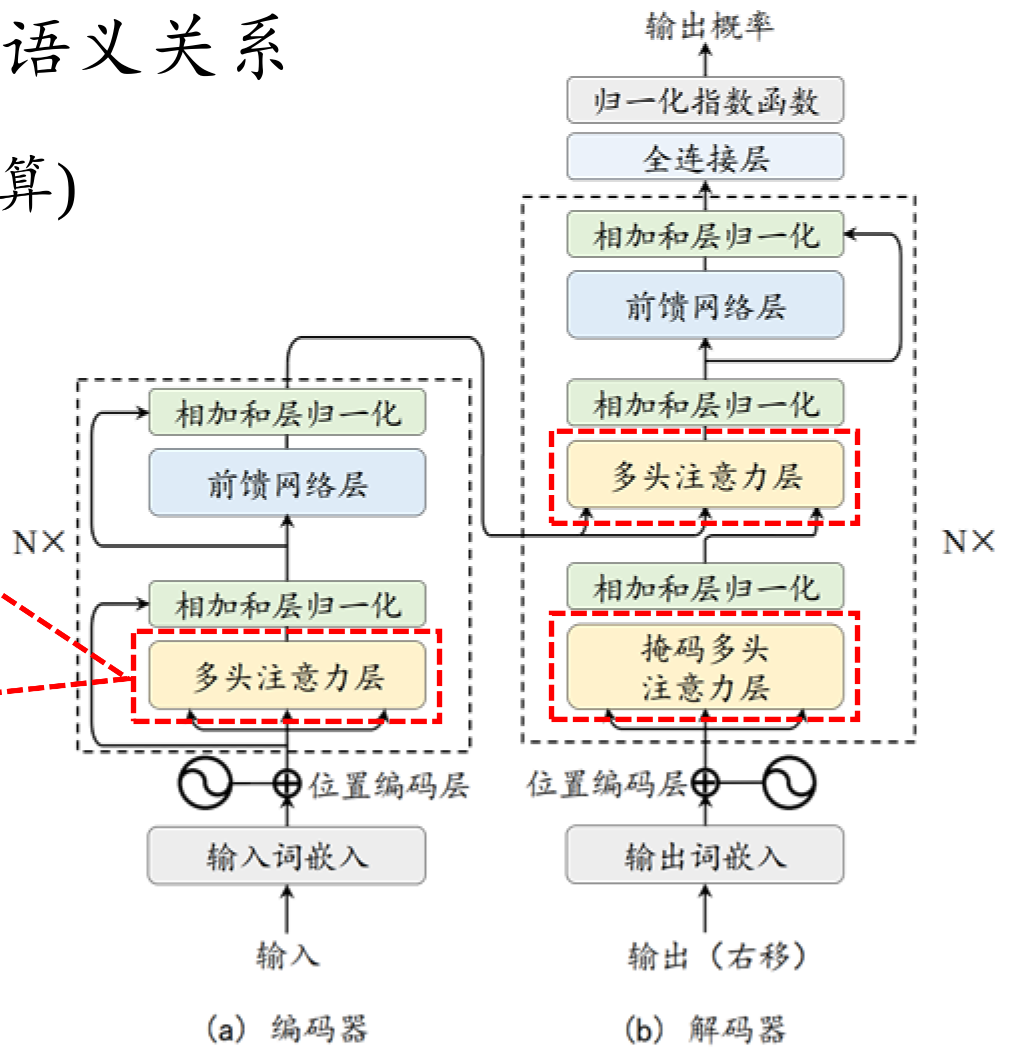
X_L：编码器第 L 层的输出



Transformer模型

- 多头注意力层：建模任意距离词元间的语义关系
  - Scaled Dot-Product Attention (单注意力头计算)
  - Multi-Head Attention (多注意力头拼接)



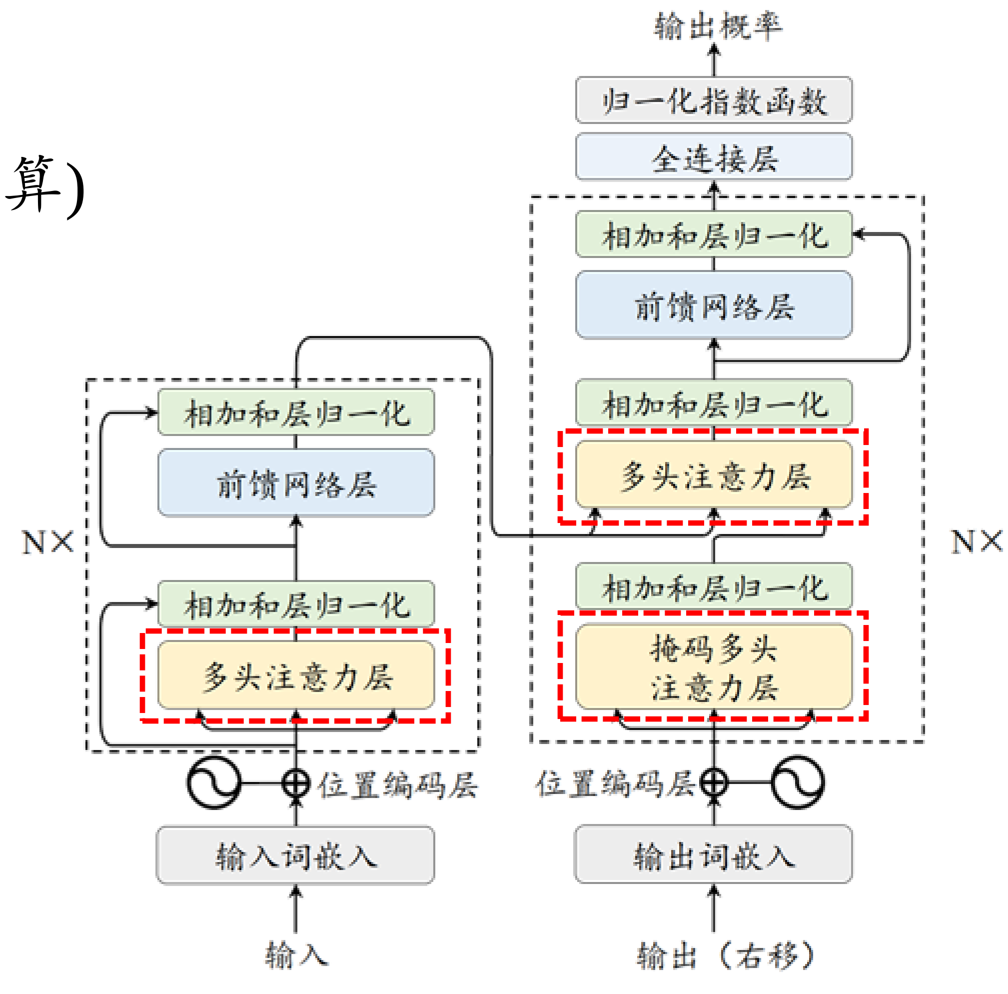


Transformer模型

- 多头注意力层
  - Scaled Dot-Product Attention (单注意力头计算)
  - Multi-Head Attention (多注意力头拼接)

思考：为什么拆分再合并?

- 将注意力分为多个头形成多个子空间，可以让模型去关注不同方面信息，最后再将各个方面的信息综合起来
- 多次注意力计算综合的结果类比 CNN 中同时使用多个卷积核的作用



(a) 编码器　(b) 解码器

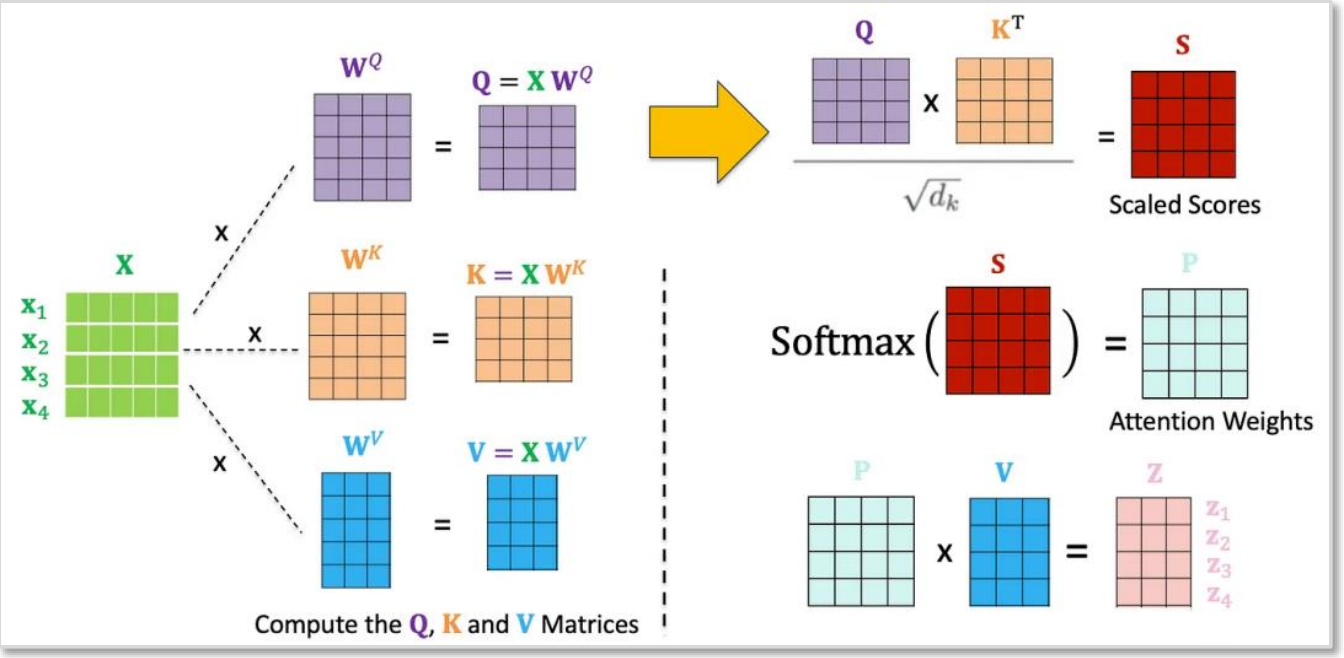
Transformer模型

AI Box

- 多头注意力层
  - Scaled Dot-Product Attention
    - 将输入 $\boldsymbol{X}$ 映射为 $\boldsymbol{Q}$、$\boldsymbol{K}$、$\boldsymbol{V}$ 矩阵

      $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{Q}$， $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{K}$， $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{V}$，
    - 注意力计算

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{\intercal}}{\sqrt{D}})\boldsymbol{V}$$



Transformer模型

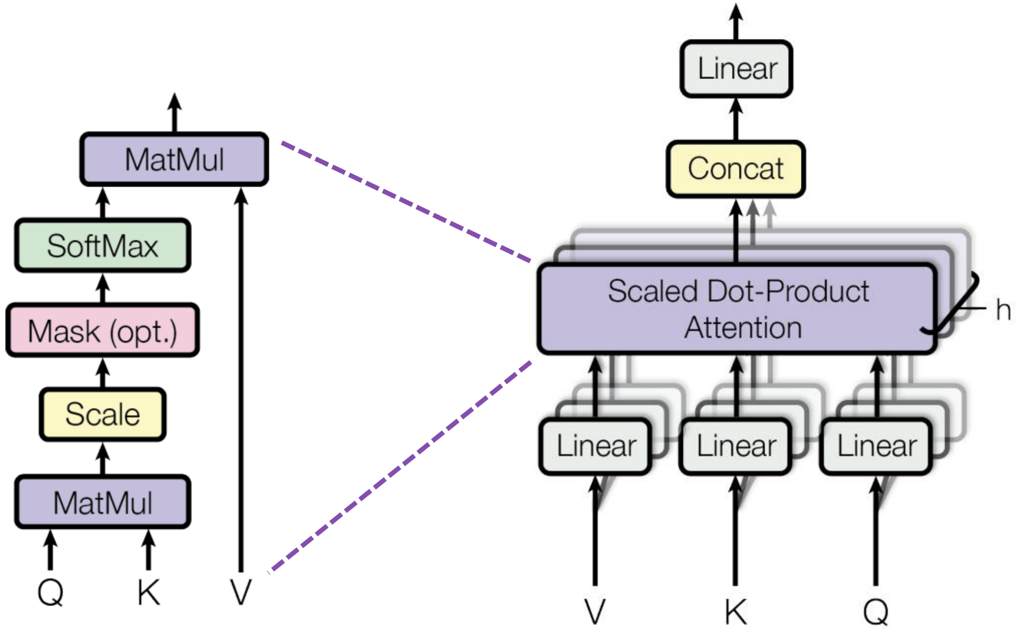
- 多头注意力层
  - Scaled Dot-Product Attention
    - 将输入 $\boldsymbol{X}$ 映射为 $\boldsymbol{Q}$、$\boldsymbol{K}$、$\boldsymbol{V}$ 矩阵

      $$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{Q}, \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{K}, \quad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{V},$$
    - 注意力计算

      $$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{\intercal}}{\sqrt{D}})\boldsymbol{V}$$
  - Multi-Head Attention
    - 拼接多个注意力头

      $$\text{MHA} = \text{Concat}(\text{head}_1, \ldots, \text{head}_N)\boldsymbol{W}^{O},$$

      $$\text{head}_n = \text{Attention}(\boldsymbol{X}\boldsymbol{W}_n^{Q}, \boldsymbol{X}\boldsymbol{W}_n^{K}, \boldsymbol{X}\boldsymbol{W}_n^{V}).$$



Scaled Dot-Product Attention Multi-Head Attention

Transformer模型

AI Box

- 思考
  - $\boldsymbol{Q}$, $\boldsymbol{K}$, $\boldsymbol{V}$在编码器、解码器中各是什么？

$$\mathrm{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \mathrm{softmax}\boxed{(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{D}})\boldsymbol{V}}$$

Transformer模型

➢ 思考

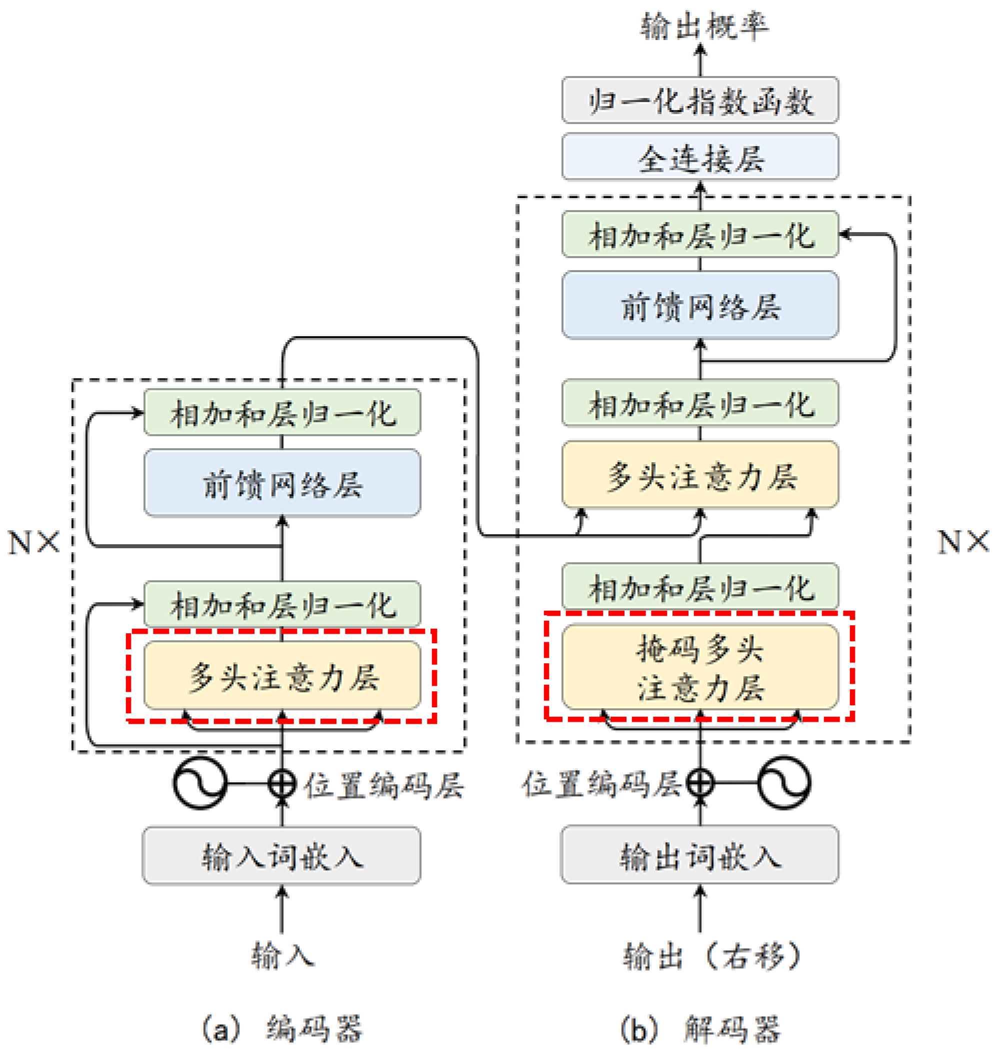
➢ ***Q*，*K*，*V*** 在编码器、解码器中各是什么？

$$\mathrm{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=\mathrm{softmax}\left(\boxed{\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{D}}}\right)\boldsymbol{V}$$

➢ 在编码器、解码器中

Q*，*K*，*V 相同，均为自身前一层的输出

(名称Self-Attention的由来)



Transformer模型

- 思考
  - $\boldsymbol{Q}$，$\boldsymbol{K}$，$\boldsymbol{V}$在编码器、解码器中各是什么？

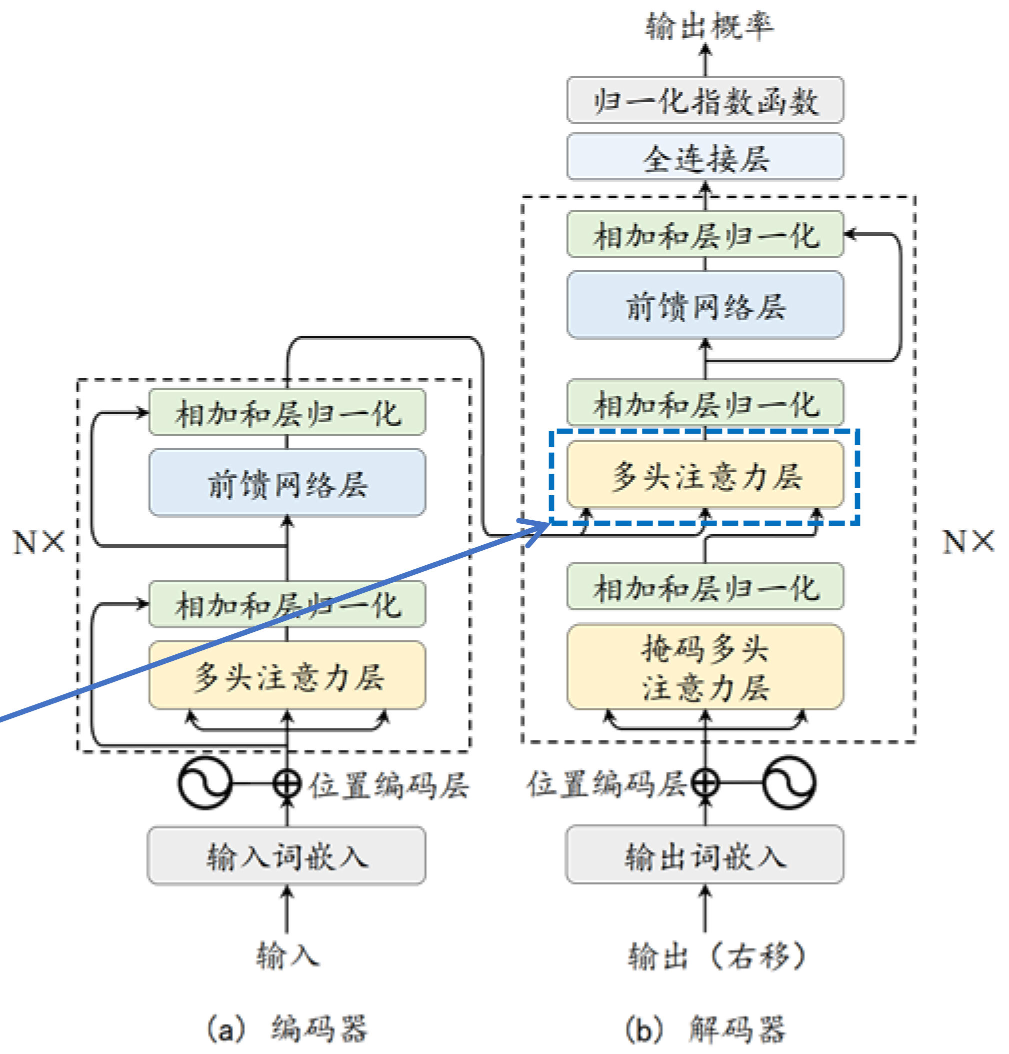
$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{softmax}\left(\boxed{\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}}{\sqrt{D}}}\right)\boldsymbol{V}$$

- 在编码器、解码器中

  $\boldsymbol{Q}$，$\boldsymbol{K}$，$\boldsymbol{V}$相同，均为自身前一层的输出

- 唯一不同

  $\boldsymbol{Q}$来自前一层输出，$\boldsymbol{K}$，$\boldsymbol{V}$是编码器输出



Transformer模型

- 思考
  - 为什么要用点积? 有其他候选吗?

$$\mathrm{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \mathrm{softmax}\left(\frac{\boxed{\boldsymbol{Q}\boldsymbol{K}^{\top}}}{\sqrt{D}}\right)\boldsymbol{V}$$

Transformer模型

- 思考
  - 为什么要用点积？有其他候选吗？

$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{softmax}\left(\frac{\boxed{\boldsymbol{Q}\boldsymbol{K}^{\top}}}{\sqrt{D}}\right)\boldsymbol{V}$$

The two most commonly used attention functions are additive attention [2], and dot-product (multi-plicative) 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.

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x}\exp\left(e_{ik}\right)}$$

$$e_{ij} = v_a^{\top}\tanh\left(W_a s_{i-1} + U_a h_j\right),$$

Transformer模型

- 思考
  - 为什么要scale，即除以$\sqrt{D}$ (D为隐藏层特征维度)?

$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\boxed{\sqrt{D}}}\right)\boldsymbol{V}$$

Transformer模型

AI Box

- 思考
  - 为什么要scale，即除以$\sqrt{D}$ (D为隐藏层特征维度)?

$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\boxed{\sqrt{D}}}\right)\boldsymbol{V}$$

d_k也是D

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 $\frac{1}{\sqrt{d_k}}$.

[4]To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k.

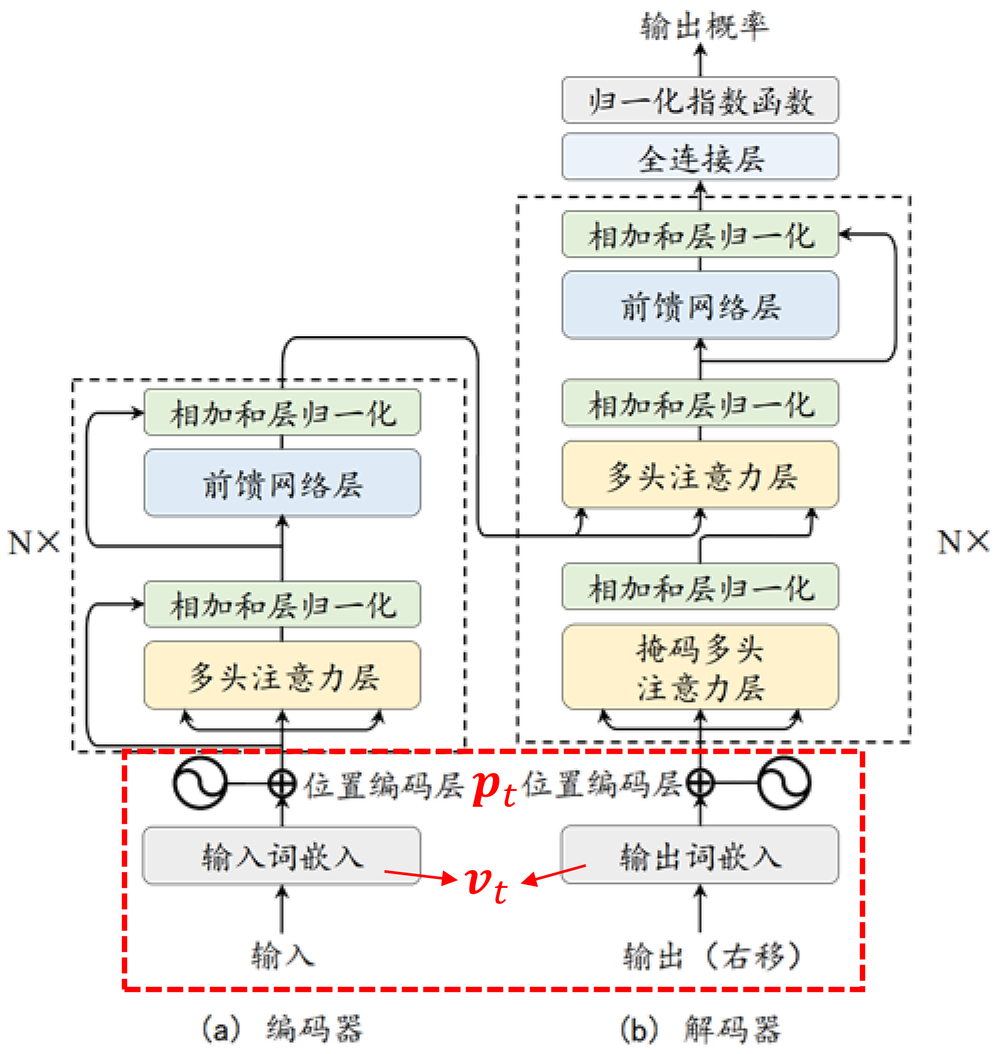
Transformer模型

AI Box

- 输入编码

$$\boldsymbol{x}_t = \boldsymbol{v}_t + \boldsymbol{p}_t$$

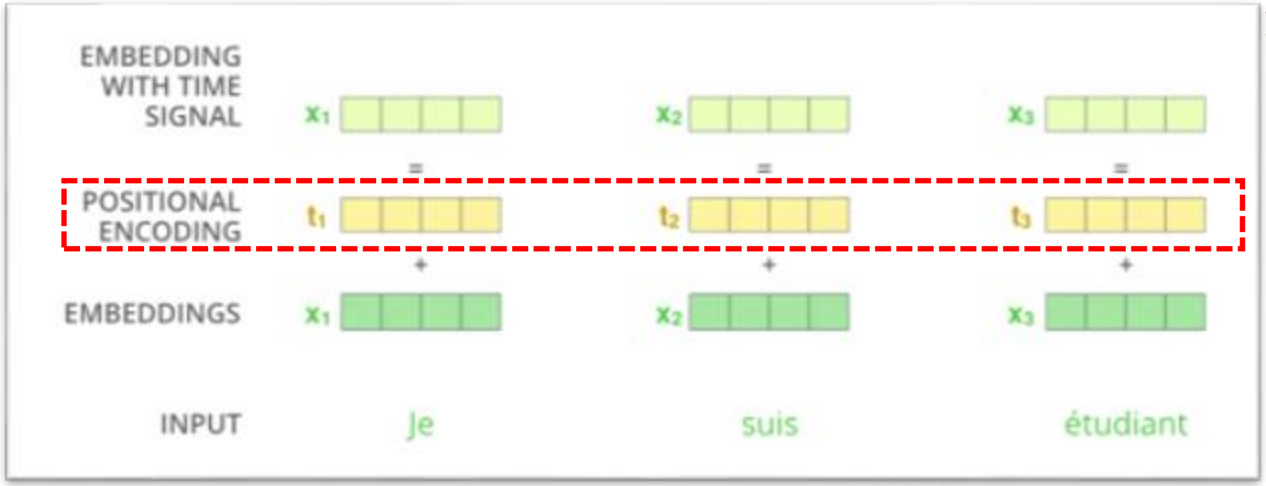
  - $\boldsymbol{v}_t$：词嵌入（语义信息）
    - 词元通过嵌入模块映射为词向量
  - $\boldsymbol{p}_t$：位置编码（位置信息）
    - 根据词元的绝对位置分配位置向量

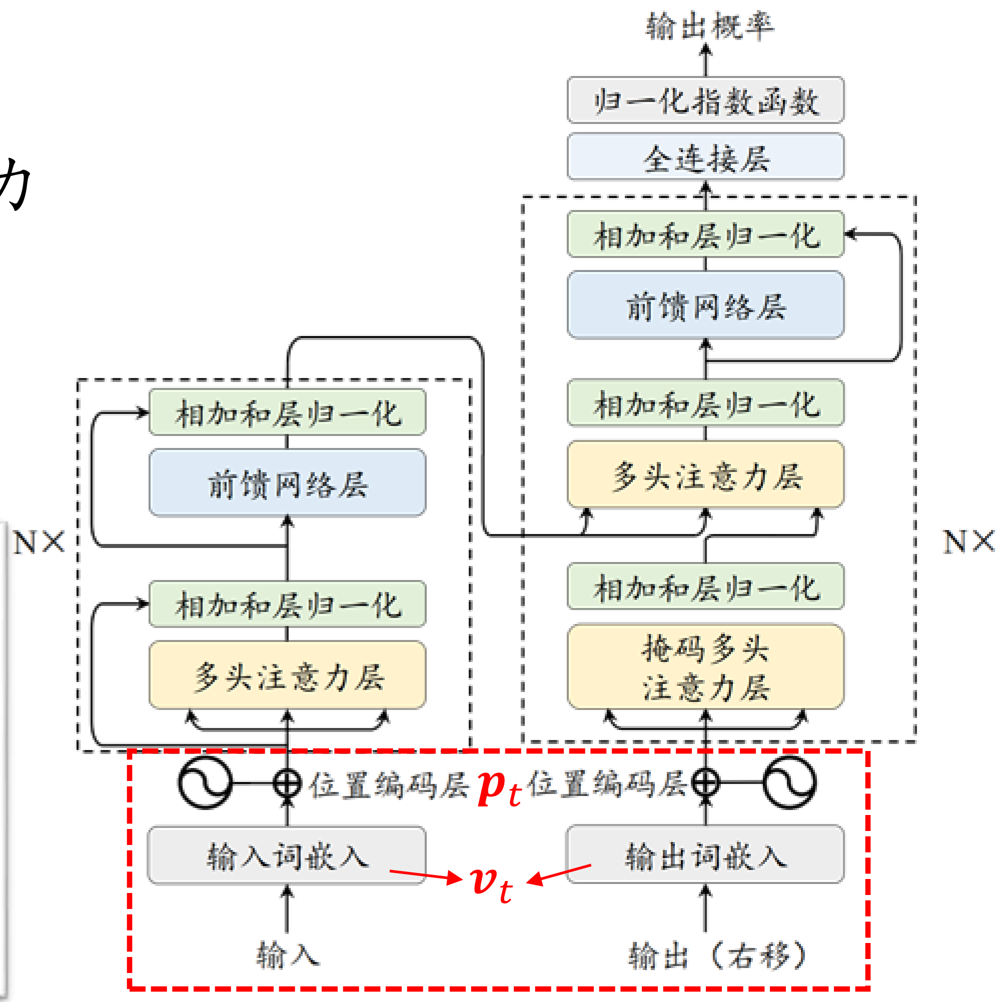


Transformer模型

AI Box

- 位置编码
  - Transformer 本身不具备建模顺序序列的能力
    - 在输入词嵌入中引入位置编码，
      将绝对或相对位置信息注入





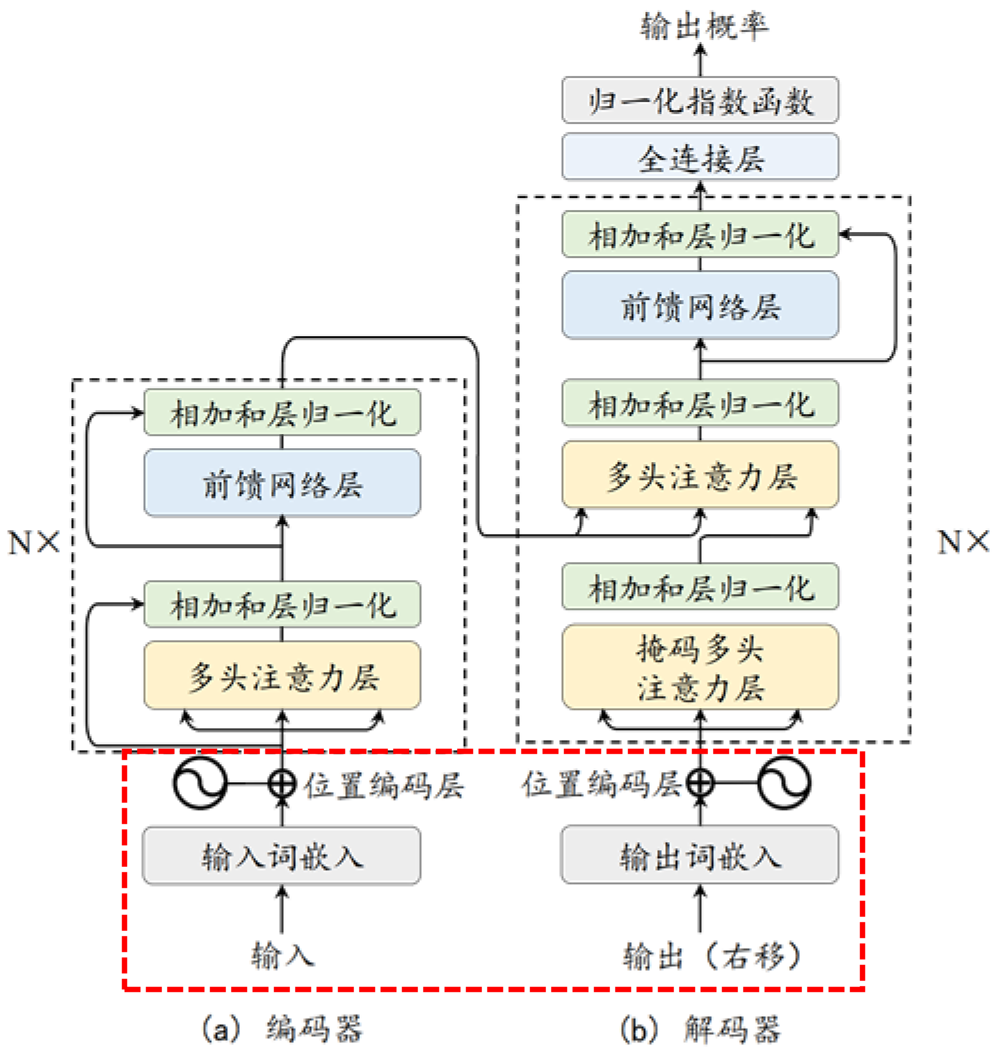
(a) 编码器　(b) 解码器

Transformer模型

- 位置编码
  - 可以通过训练学习得到，也可以事先指定
  - 论文使用 sin 和 cos 函数线性变换得到

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

pos 表示一句话中单词的位置，i 是词嵌入维度序号，d_{model}是词嵌入维度



Transformer模型

➢ 位置编码

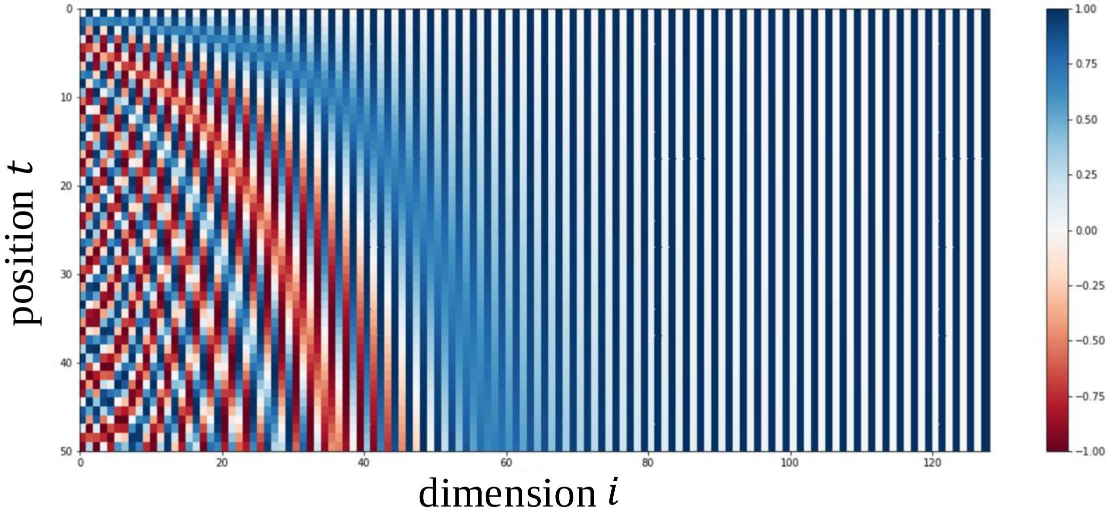
➢ 将 sin 和 cos 函数转换得到

$$p_t^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k+1 \end{cases} \qquad \omega_k = \frac{1}{10000^{2k/d}}$$

词嵌入维度序号i变大（k变大），w_k 越小，频率 $\frac{w_k}{2\pi}$也越来越小

➢ 一个词的位置编码表示如下

$$p_t = \begin{bmatrix} \sin(w_1 t) \\ \cos(w_2 t) \\ \cdots \\ \sin(w_{d/2} t) \\ \cos(w_{d/2} t) \end{bmatrix}_{d\times 1}$$



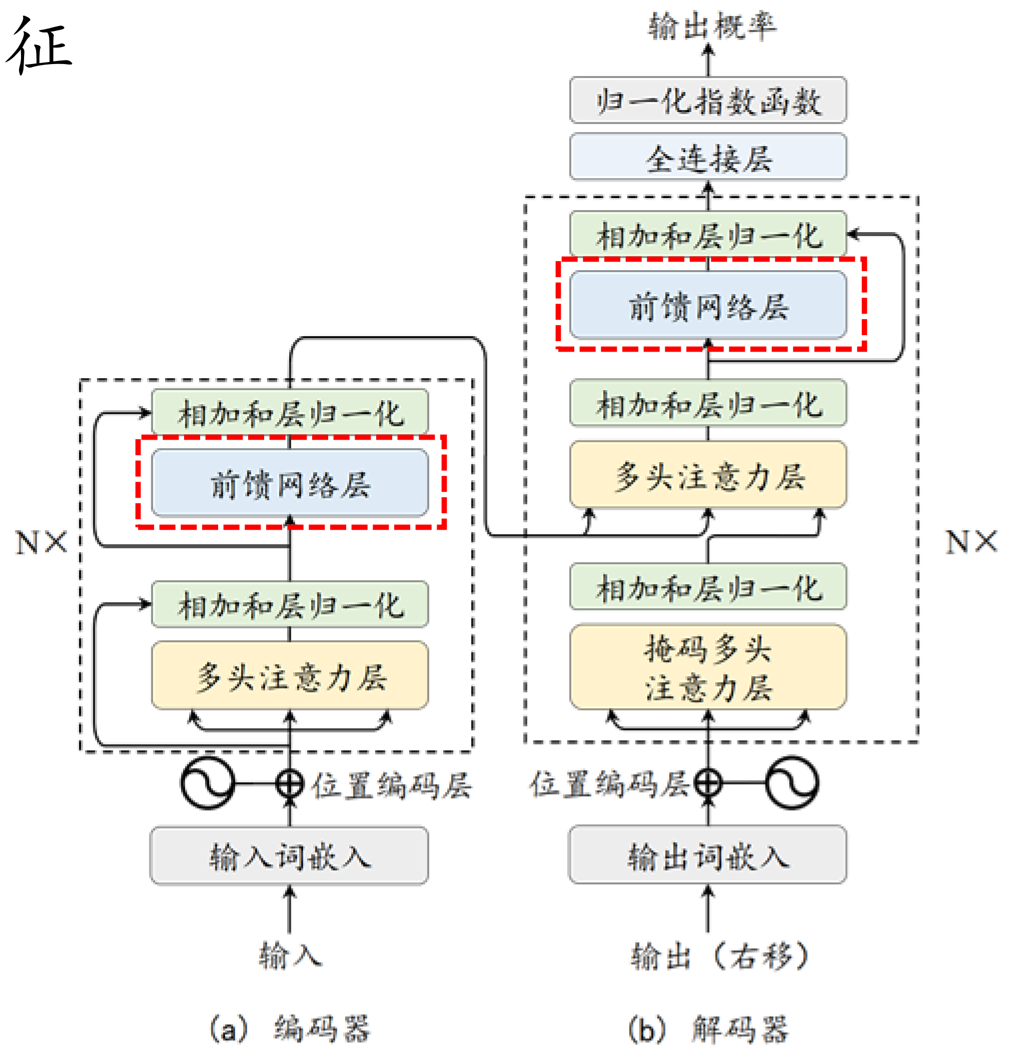
随着词嵌入维度序号i增大，该序号下的数值变化频率指数级下降

Transformer模型

- 前馈网络层：学习复杂的函数关系和特征

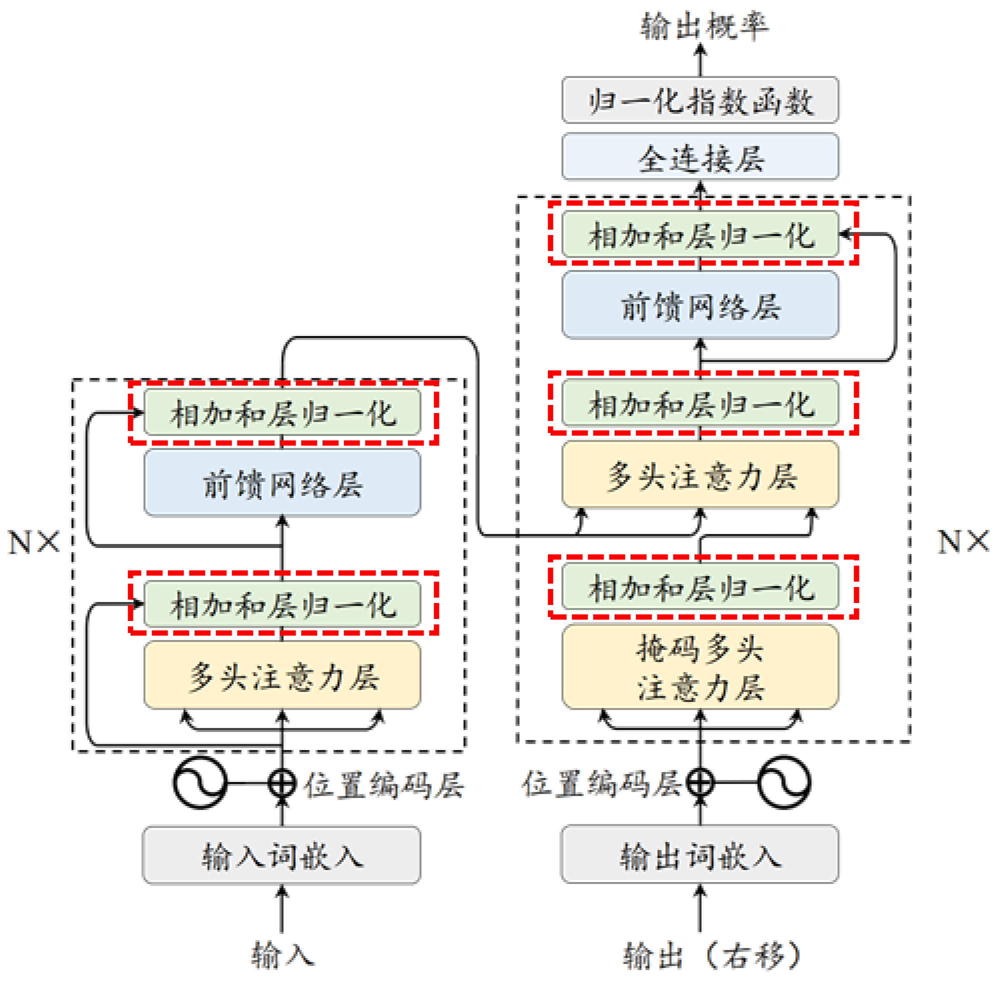
$$\mathrm{FFN}(\boldsymbol{X}) = \sigma(\boldsymbol{X}\boldsymbol{W}^U + \boldsymbol{b}_1)\boldsymbol{W}^D + \boldsymbol{b}_2$$

  - 线性变换
    - 先升维、后降维
  - 非线性激活函数σ
    - ReLU 或 GELU 等



Transformer模型

- 残差连接
  - 将输入与输出相加
  - 缓解梯度爆炸和消失
- 层归一化
  - 对数据进行重新放缩
  - 提升训练稳定性

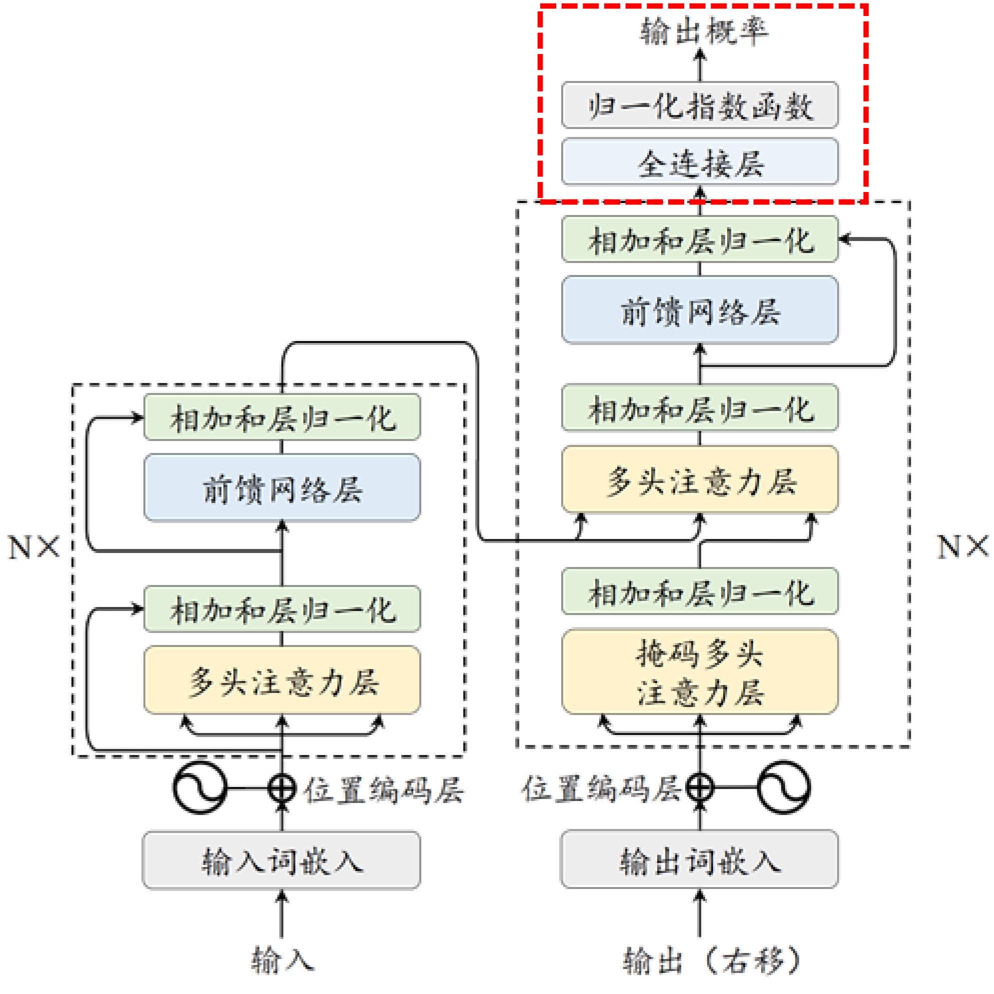


（a）编码器 （b）解码器

Transformer模型

- 输出层：生成词元概率分布
  - 全连接层
  - 归一化指数函数 Softmax

$$\boldsymbol{O} = \text{softmax}(\boldsymbol{W}^L \boldsymbol{Y}_L)$$



(a) 编码器　　(b) 解码器

谢谢