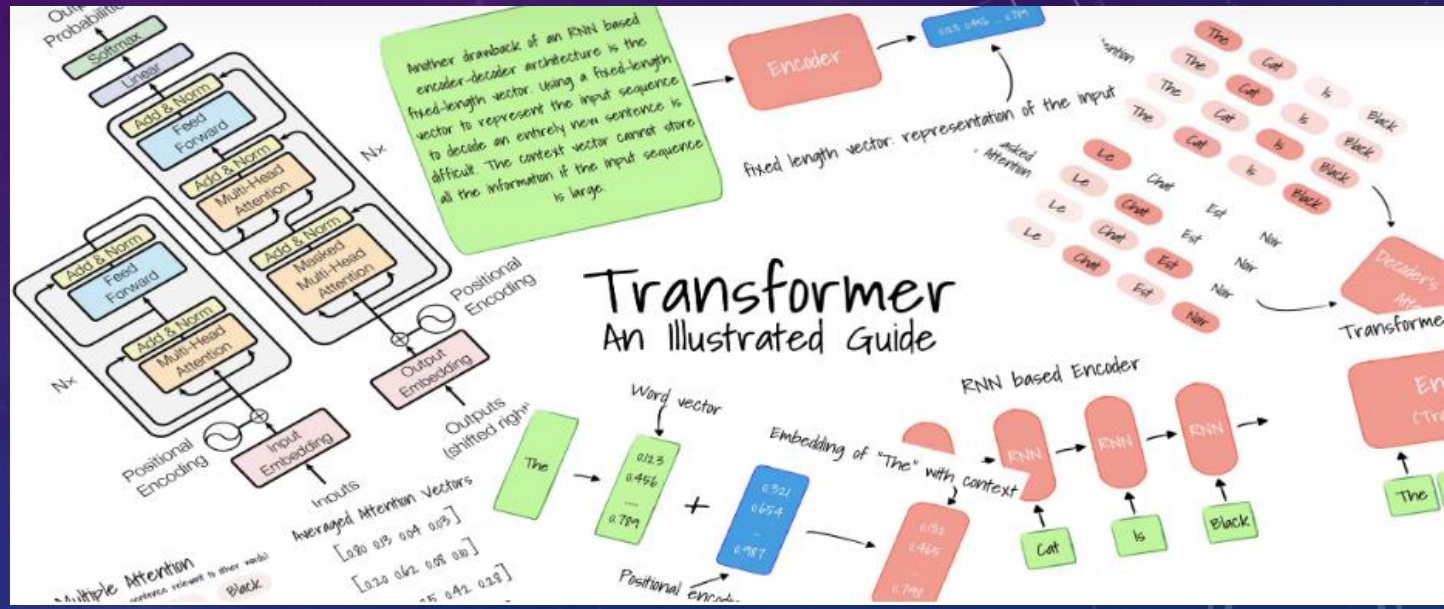
ALL YOU NEED IS ATTENTION
--
TRANSFORMER

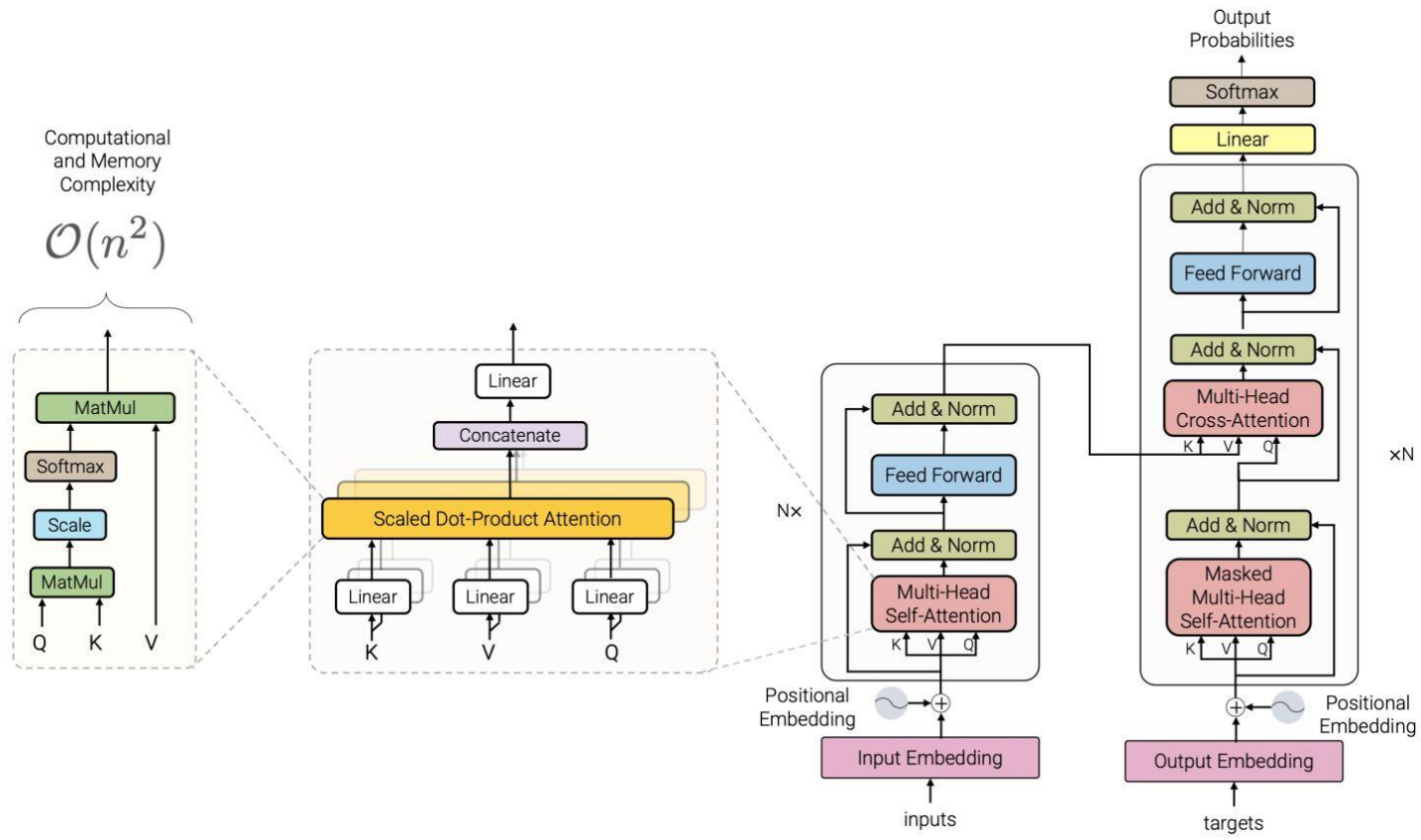
大数据1901 陶盛皿
19024075

OUTLINE

- Introduction
- Element1 : Attention -> Self attention
- Element2 : Encoder - Decoder
- Transformer Block
- Appendix



INTRODUCTION

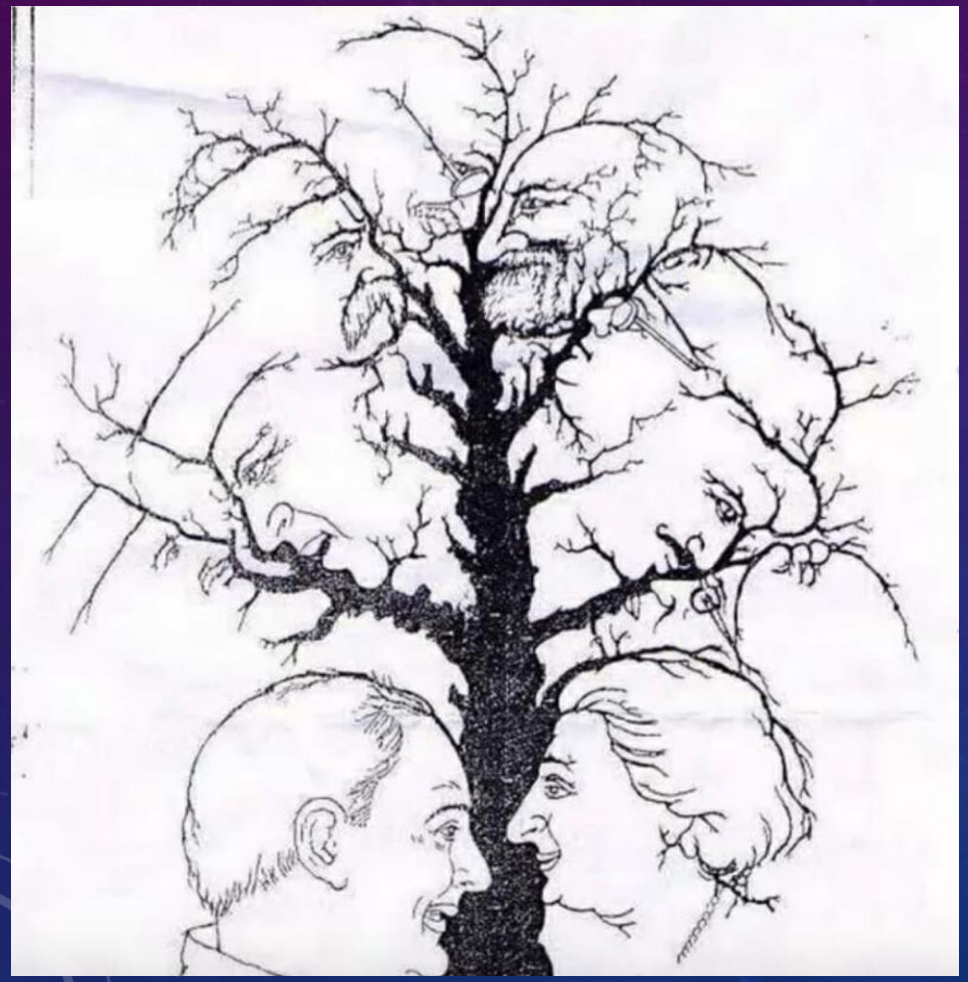


- Transformer最早提出是被作为一种序列到序列(seq2seq)模型并应用于机器翻译的。之后的一些工作表明基于Transformer的预训练模型可以在不同的任务上实现SOTA。
- 近一两年业界"X-former "模型，诸Reformer、Linformer、Performer、Longformer，它们在最初的Transformer架构基础上进行了改进，其中许多是围绕计算和内存效率进行的改进。

ELEMENT1 : ATTENTION -> SELF ATTENTION

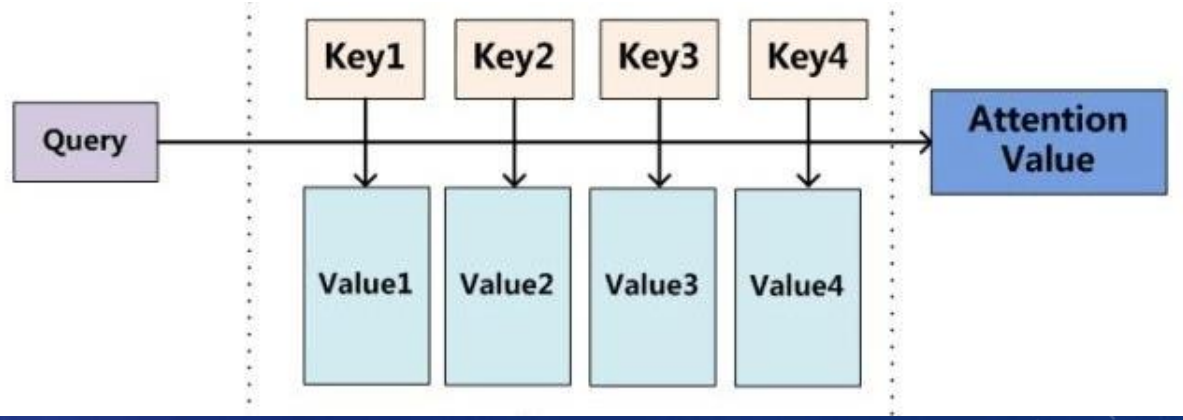
- Attention 机制是用来做什么的？
- Attention 是怎么工作的?
- Self-attention 是怎么从 Attention 过渡过来的？
- Attention 和 self-attention 的区别是什么？
- Self-attention 为什么能 work？

ATTENTION 机制



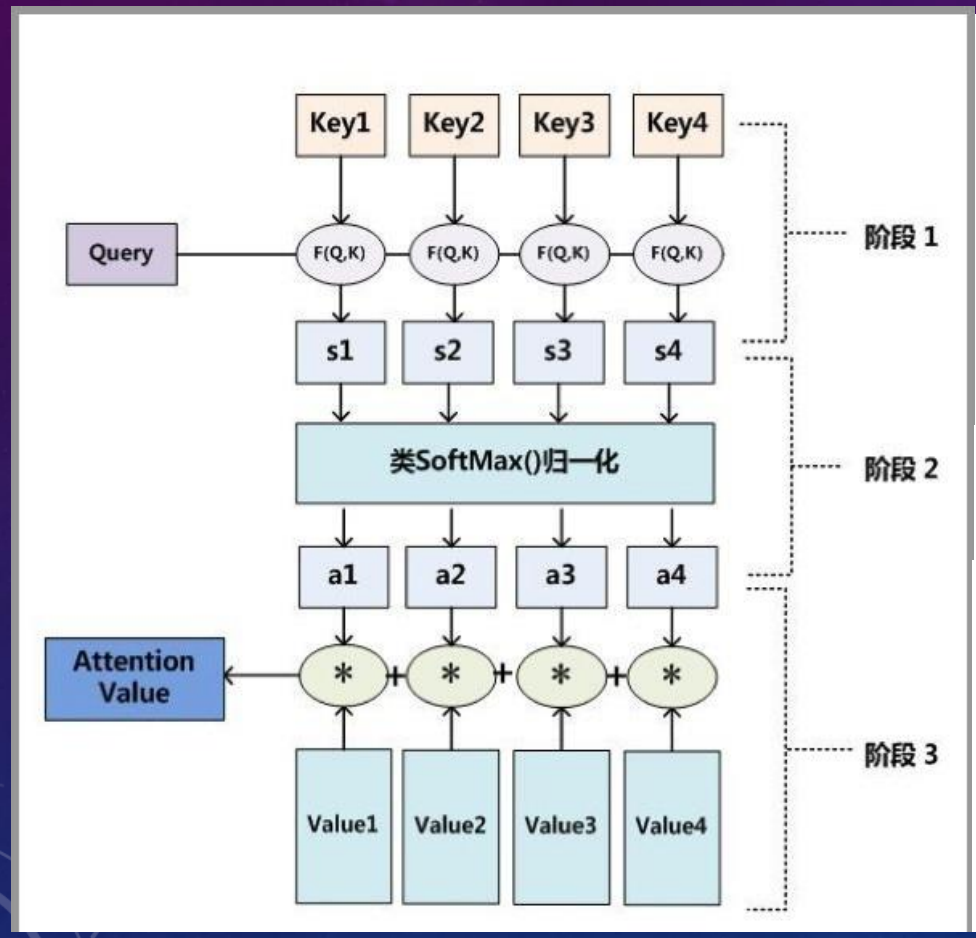
- 生物在观察、学习、思考行为中的过程的一种独特的生理机制，这种机制就是 Attention 机制。当你把注意力放在不同的位置时，你能看到的脸也并不相同。这其实就是大脑的注意力机制，可以说我们无时无刻不在使用这种能力，只是我们并没有把注意力放在上面。

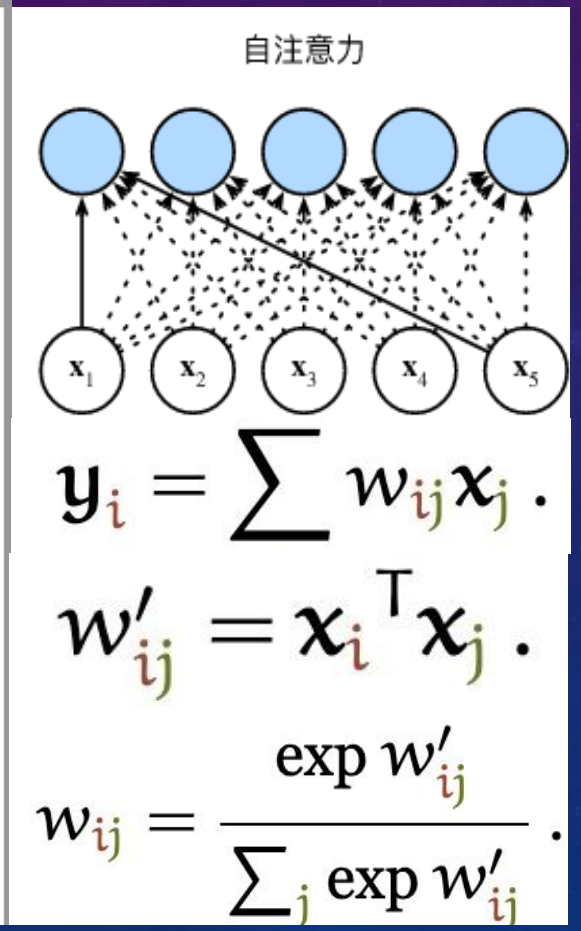
$$\text{Attention}\left(\mathrm{Q},\mathrm{K},\mathrm{V}\right)=\text{softmax}\left(\frac{\mathrm{QK}^{\top}}{\sqrt{D_k}}\right)\mathrm{V}=\mathrm{AV}$$



- query 和 key 进行相似度计算，得到一个query 和 key 相关性的分值
- 将这个分值进行归一化(softmax)，得到一个注意力的分布
- 使用注意力分布和 value 进行计算，得到一个融合注意力的更好的 value 值

ATTENTION -> SELF-ATTENTION



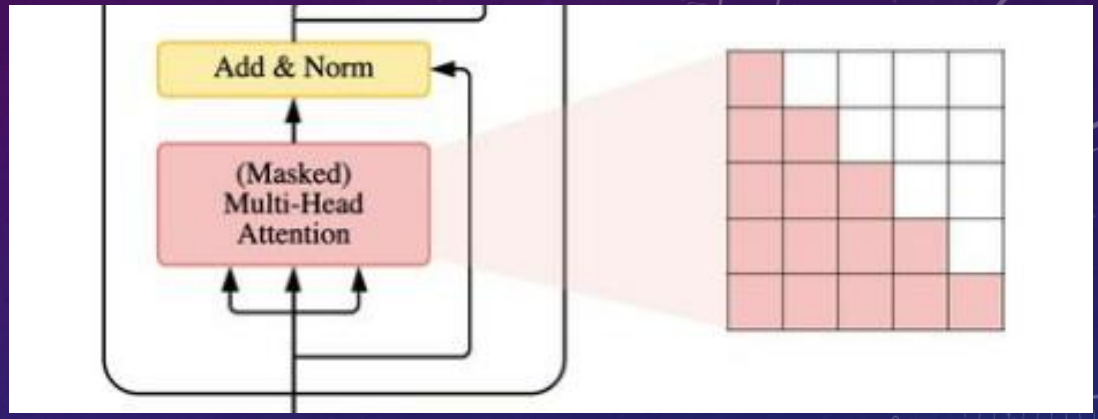


区别：

1. AT被应用在某一层的话，它更多的是被应用在输出或者是状态层上；SA更多的实在关注input上。
2. SA可以在一个模型当中被多次的、独立的使用（比如说在Transformer中，使用了18次；在Bert当中使用12次）；AT在一个模型当中经常只是被使用一次，从encoder转换到decoder 。
3. SA比较擅长在一个序列当中，寻找不同部分之间的关系；AT更擅长寻找两个序列之间的关系。
4. AT可以连接两种不同的模态，比如说图片和文字；SA更多的是被应用在同一种模态上
5. 大部分情况，SA这种结构更加的general，在很多任务作为降维、特征表示、特征交叉等功能尝试着应用，很多时候效果都不错。

TRANSFORMER中的SA

- 在Transformer中，主要涉及到三种不同的注意力类型：
  - Self-attention
  - Masked Self-attention. 在Transformer的解码器中，自注意力受到限制，使得每个位置的查询只能关注到包括该位置及之前位置的所有键值对。常规做法是将掩码矩阵(mask matrix)添加到注意力分数上，其中非法位置采用负无穷进行遮挡。这一类注意力方法也经常被称为自回归(autogressive)或者因果(causal)注意力。
  - Cross-attetnion. 查询是从前一个（解码器）层的输出投影所获得的，而键和值是使用编码器的输出投影得到的。



- Transformer并没有简单地应用单个注意力函数，而是使用了多头注意力。通过单独计算每一个注意力头，最终再将多个注意力头的结果拼接起来作为MSA模块最终的输出

$$\begin{aligned}\text{MultiHeadAttn}\left(Q,K,V\right) &= \text{Concat}\left(\text{head}_1,\cdots,\text{head}_H\right)\mathbf{W}^O\\ \text{where head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)\end{aligned}$$

TRANSFORMER中的SA

- 实际在Transformer的实现过程中，作者使用了三个tricks。

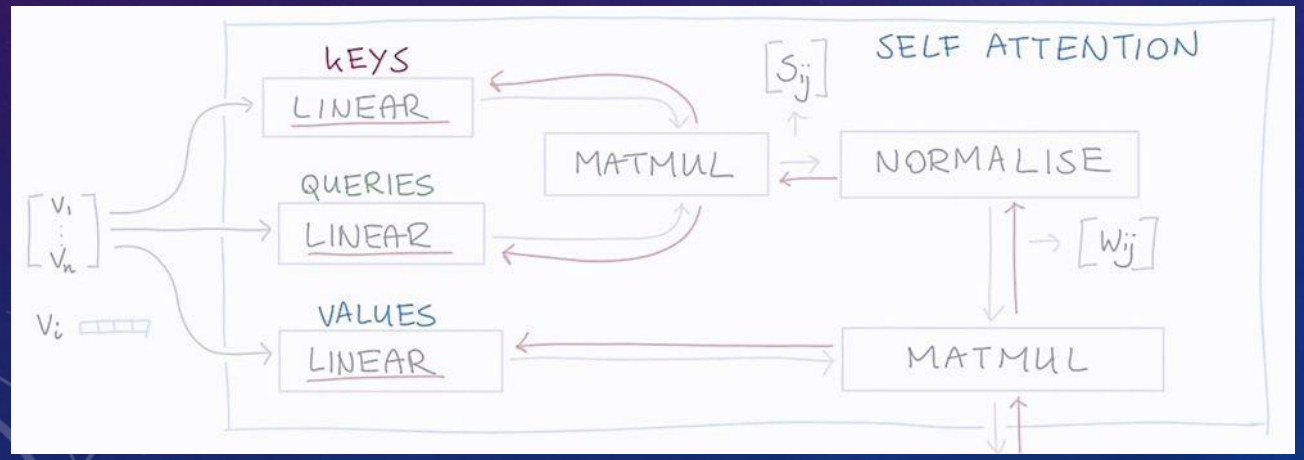
1. QKV

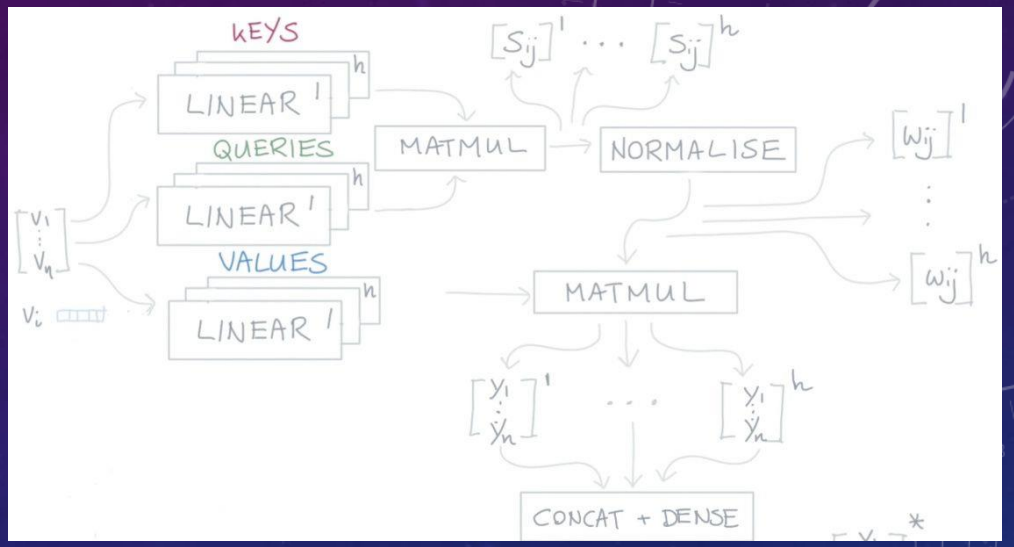
$$\mathbf{q}_i = \boldsymbol{W}_q \boldsymbol{x}_i \qquad \mathbf{k}_i = \boldsymbol{W}_k \boldsymbol{x}_i \qquad \boldsymbol{v}_i = \boldsymbol{W}_v \boldsymbol{x}_i$$

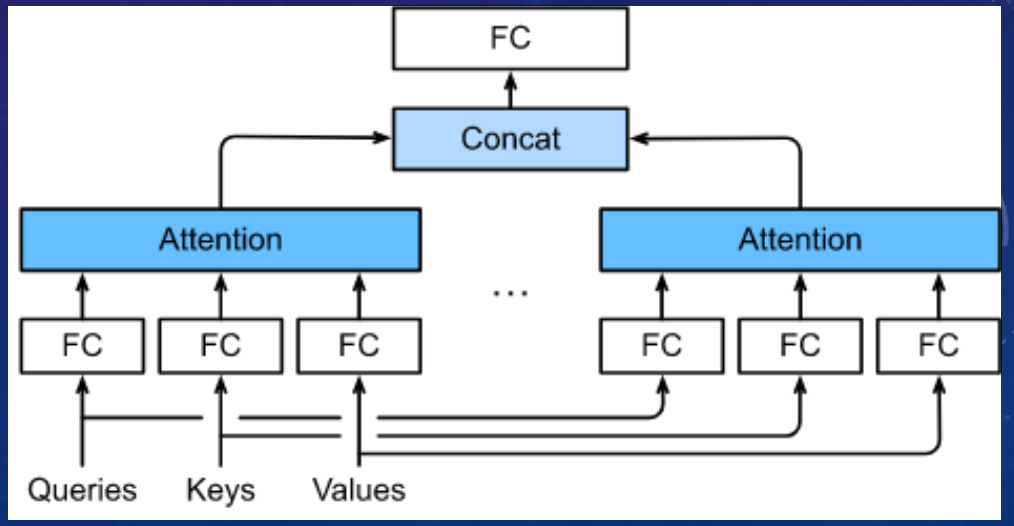
2. 缩放点积的值

$$\text{Attention}\left(\mathrm{Q}, \mathrm{K}, \mathrm{V}\right) = \text{softmax}\left(\frac{\mathrm{QK}^{\top}}{\sqrt{D_k}}\right)\mathrm{V} = \mathrm{AV}$$

3. **Multi-head attention**



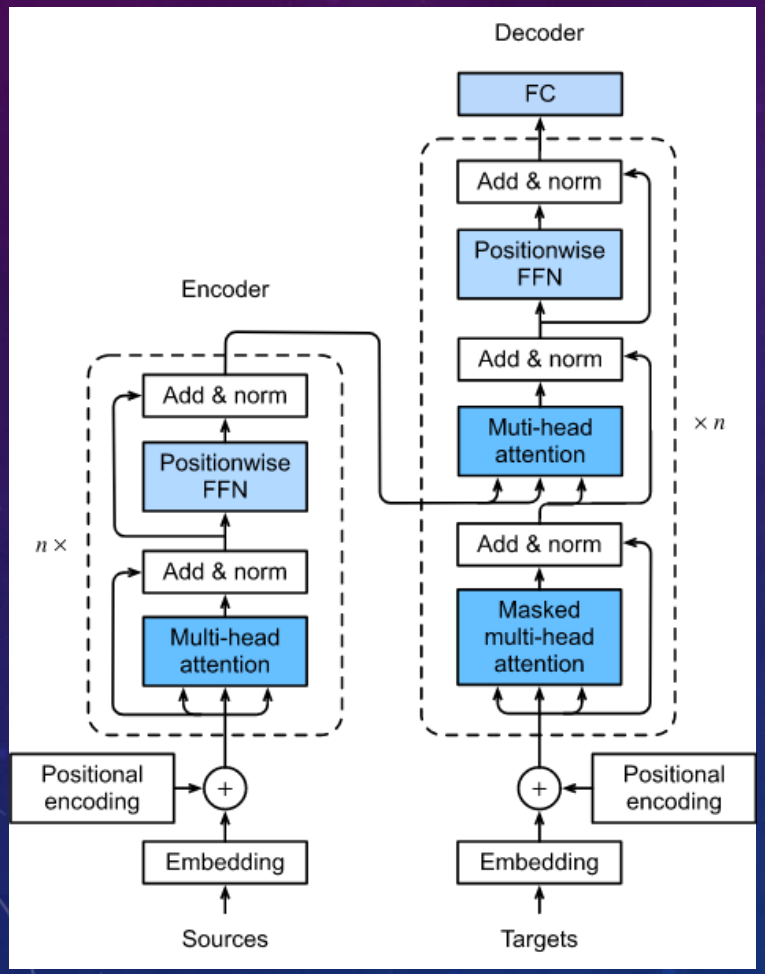


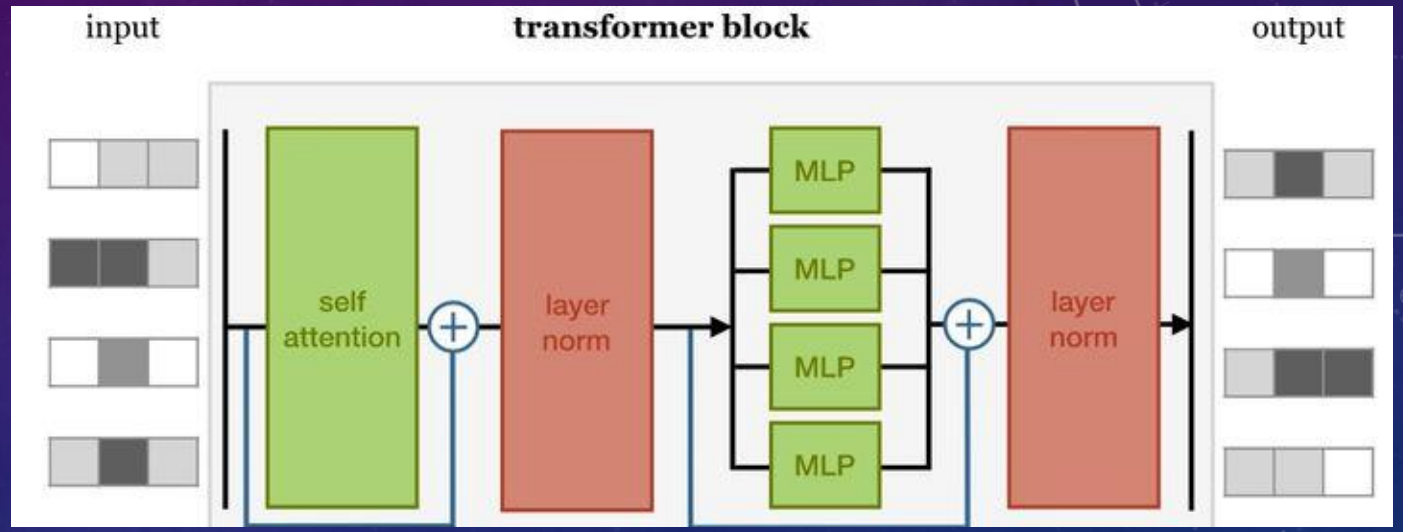


ELEMENT2 : ENCODER - DECODER

- 完整的 Transformer Block 是什么样的?
- 怎么捕获序列中的顺序信息呢？

TRANSFORMER BLOCK

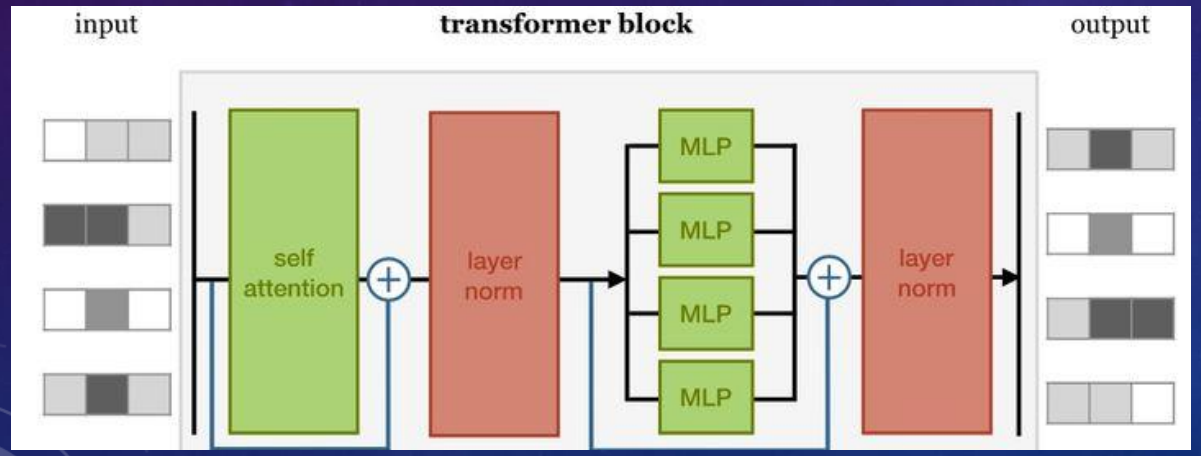




1. self-attention layer
2. **normalization layer**
3. feed forward layer
4. another normalization layer
5. residual connections

TRANSFORMER BLOCK

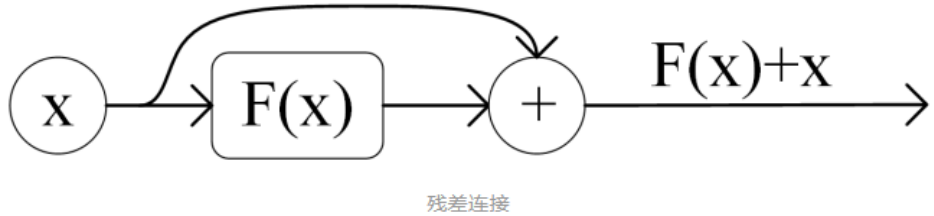
- Self-attention layer: multi-head
- Feed forward layer: 即FC，2层MLP + ReLu
- Residual connections + Layer normalization



$$\mathrm{FFN}(\mathbf{H}') = \mathrm{ReLU}\left(\mathbf{H}'\mathbf{W}^1 + \mathbf{b}^1\right)\mathbf{W}^2 + \mathbf{b}^2$$

$$\mathrm{H}' = \mathrm{LayerNorm}\left(\mathrm{SelfAttention}\left(\mathrm{X}\right) + \mathrm{X}\right)$$
$$\mathrm{H} = \mathrm{LayerNorm}\left(\mathrm{FFN}\left(\mathrm{H}'\right) + \mathrm{H}'\right)$$

详见ReSNet/RNNs



怎么捕获序列中的顺序信息呢？

- position embeddings
- position encodings
- 自注意力的最大路径长度短，因为其计算复杂度是关于序列长度的二次方，在很长的序列中计算会非常慢。
- 为了使用序列的顺序信息，我们可以通过在输入表示中添加位置编码，来注入绝对的或相对的位置信息。

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

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

$$\begin{aligned}
&\begin{bmatrix} \cos(\delta\omega_j) & \sin(\delta\omega_j) \\ -\sin(\delta\omega_j) & \cos(\delta\omega_j) \end{bmatrix} \begin{bmatrix} p_{i,2j} \\ p_{i,2j+1} \end{bmatrix} \\
&= \begin{bmatrix} \cos(\delta\omega_j)\sin(i\omega_j) + \sin(\delta\omega_j)\cos(i\omega_j) \\ -\sin(\delta\omega_j)\sin(i\omega_j) + \cos(\delta\omega_j)\cos(i\omega_j) \end{bmatrix} \\
&= \begin{bmatrix} \sin\left((i+\delta)\omega_j\right) \\ \cos\left((i+\delta)\omega_j\right) \end{bmatrix} \\
&= \begin{bmatrix} p_{i+\delta,2j} \\ p_{i+\delta,2j+1} \end{bmatrix},
\end{aligned}$$

APPENDIX

- Transformer摒弃了以往CNN、RNN的架构，完全使用SA来捕获序列特征。
- 通常比CNN或RNN具有更好的性能，具体理论原因尚有争议。
- 在许多应用场景中，集成多模态数据对于提高任务性能是有用且必要的。此外，通用人工智能还需要能够捕捉不同模态之间的语义关系。由于Transformer在文本、图像、视频和音频方面取得了巨大的成功，但多模态模态注意力的设计仍有待改进。

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O\left(T^2 \cdot D\right)$	$O(1)$	$O(1)$
Fully Connected	$O\left(T^2 \cdot D^2\right)$	$O(1)$	$O(1)$
Convolutional	$O\left(K \cdot T \cdot D^2\right)$	$O(1)$	$O\left(\log_K(T)\right)$
Recurrent	$O\left(T \cdot D^2\right)$	$O(T)$	$O(T)$

THANKS.