

- 音乐、语言、文本和视频都是连续的
 - 标题“狗咬人”远没有“人咬狗”那么令人惊讶
- 大地震发生后，很有可能会有几次小地震
- 人的互动是连续的，从网上的吵架可以看出
- 预测明天的股价要比填补昨天遗失的股价要更困难

序列 (x_1, x_2, \dots, x_T) 出现的概率，可以用条件概率公式计算：

$$p(x_1, x_2, \dots, x_T) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1x_2) \dots \cdot p(x_T|x_1, \dots, x_{T-1})$$

对 (x_1, x_2, \dots, x_T) 进行建模，使得给定模型的输入 (x_1, x_2, \dots, x_t) ，
得到 $p(x_1, x_2, \dots, x_t, x_{t+1})$

$$P(\text{deep, learning, is, fun}) = P(\text{deep})P(\text{learning} \mid \text{deep})P(\text{is} \mid \text{deep, learning})P(\text{fun} \mid \text{deep, learning, is}).$$

x_t 依赖前 $t - 1$ 个时刻的数据，模型的参数量将随着 t 增加指数增长，由此引入隐变量模型

$$P(x_t | x_{t-1}, \dots, x_1) \approx P(x_t | h_{t-1})$$

其中 h_{t-1} 是 **隐状态 (hidden state)**，也称 **隐藏变量 (hidden variable)**，它存储了时间步 $t - 1$ 的序列信息。通常，我们可以基于当前的输入 x_t 和先前的隐状态 h_{t-1} 来计算时间步 t 处的任何时间的隐状态

$$h_t = f(x_t, h_{t-1})$$

无隐状态的神经网络（全连接层）：

$$\begin{aligned} H &= \phi(XW_{xh} + b_h) \\ O &= HW_{hq} + b_q \end{aligned}$$

有隐藏层的神经网络（循环神经网络）：

$$\begin{aligned} H_t &= \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h) \\ O_t &= H_t W_{hq} + b_q \end{aligned}$$

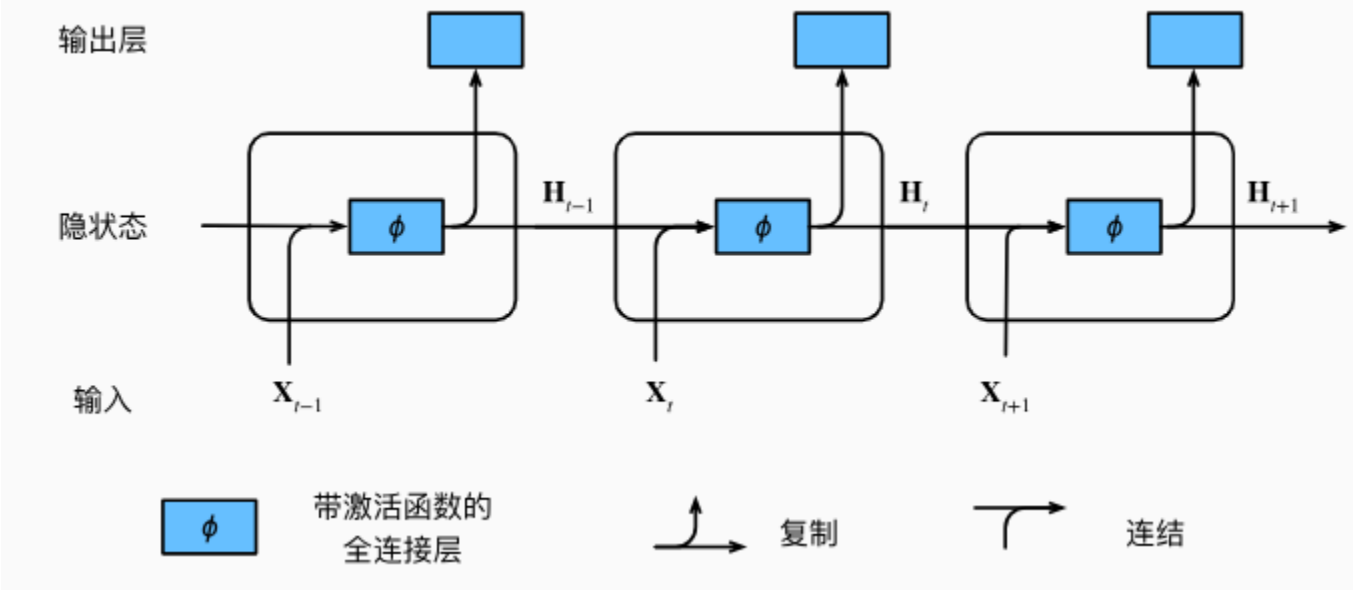
W_{xh}, W_{hh}, b_h
在所有时间步
 t 都是相同的

循环神经网络 (RNN)

$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h)$$

$$O_t = H_t W_{hq} + b_q$$

引入 $H_{t-1} W_{hh}$, W_{hh} 捕获 H_t 和 H_{t-1} 之间的关系, H_t 中保留了序列直到当前时间步的历史信息, 就如同当前时间步下神经网络的状态或记忆。当前时间步中隐状态的计算过程和前一个过程相同, 因此计算是循环的。



对序列进行建模

缺点：数值不稳定

假设 ϕ 是一个线性函数, 则有 $H_t \sim H_{t-N} W_{hh}^N$

现代循环神经网络（门控循环神经网络GRU）

门控隐状态：模型有专门的机制来确定应该何时**更新隐状态**，何时**重置隐状态**。并且，这些机制是可学习的。门控机制的引入不仅增强了模型的表达能力，还有效限制了数值范围，避免梯度爆炸的问题

$$\begin{aligned} \mathbf{R}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xr} + \mathbf{H}_{t-1} \mathbf{W}_{hr} + \mathbf{b}_r), \\ \mathbf{Z}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xz} + \mathbf{H}_{t-1} \mathbf{W}_{hz} + \mathbf{b}_z), \end{aligned}$$

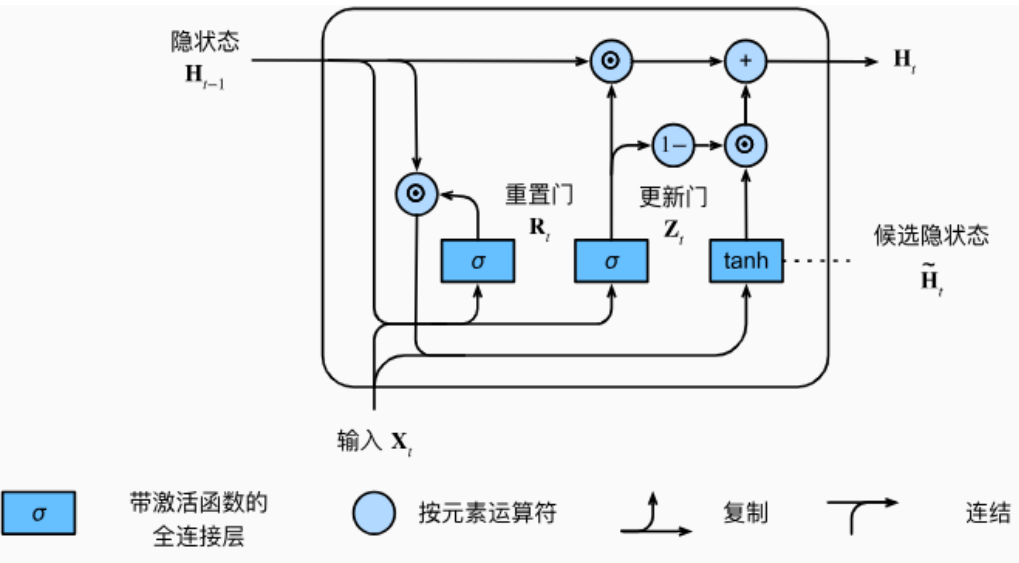
重置门和更新门, $\sigma = \text{sigmoid}$
将输入映射到[0,1]

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h),$$

候选隐状态 $\tilde{\mathbf{H}}_t$ ，当 R_t 等于1时，就是普通的循环神经网络

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t.$$

更新门 Z_t 在旧状态和候选隐状态之间选择，当 Z_t 接近1时，模型倾向于只保留旧状态



- 总之，门控循环单元具有以下两个显著特征：
- 重置门有助于捕获序列中的短期依赖关系（及时丢掉过去的无用信息）；
 - 更新门有助于捕获序列中的长期依赖关系（保留过去信息）。

现代循环神经网络（长短期记忆网络LSTM）

长短期记忆网络的设计灵感来自于计算机的逻辑门。长短期记忆网络引入了**记忆元 (memory cell)**，或简称为**单元 (cell)**。有些文献认为记忆元是隐状态的一种特殊类型，它们与隐状态具有相同的形状，其设计目的是用于记录附加的信息。

$$\begin{aligned} \mathbf{I}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i), \\ \mathbf{F}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f), \\ \mathbf{O}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o), \end{aligned}$$

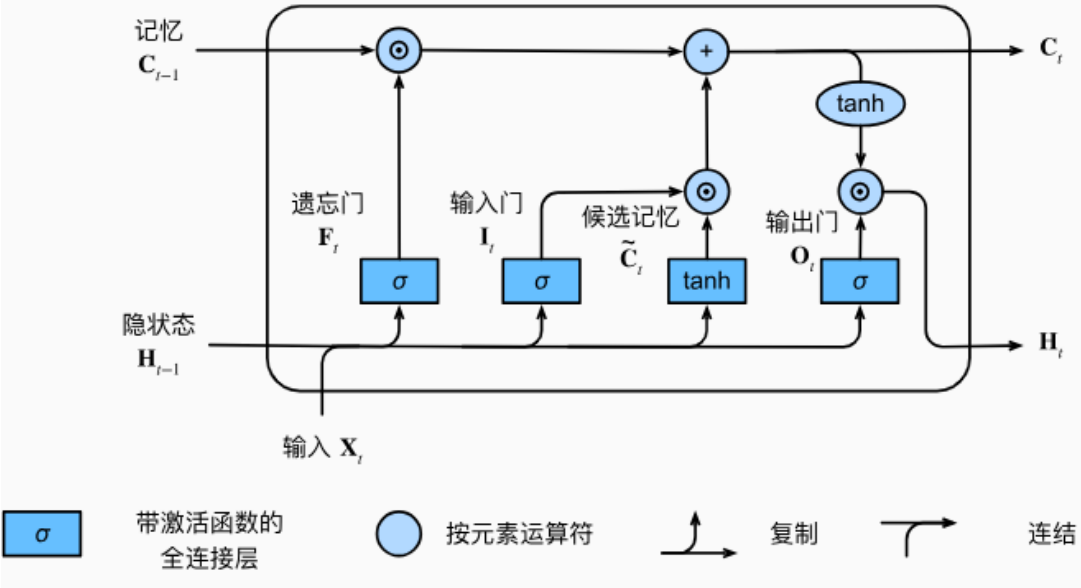
类似门控循环神经网络，输入门、遗忘门和输出门，也使用 $\sigma = \text{sigmoid}$ 将输入映射到 $[0,1]$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \mathbf{b}_c),$$

和普通循环神经网络相同，候选记忆元 $\hat{\mathbf{C}}_t$ 由输入和隐状态生成

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t.$$

输入门和遗忘门共同决定记忆的更新（门控神经网络是二选一）

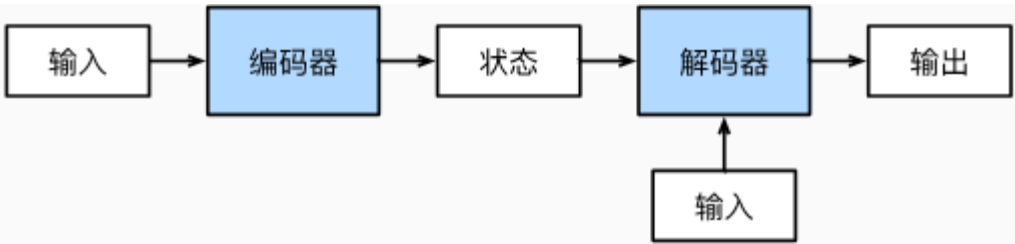


$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t).$$

只要输出门接近1，我们就能够有效地将所有记忆信息传递给预测部分，而对于输出门接近0，我们只保留记忆元内的所有信息，而不需要更新隐状态。

编码器-解码器架构

前述的循环神经网络很好的实现了序列数据的**建模**，对于常见的**序列到序列**问题（例如机器翻译），其输入序列和输出序列的**长度都是可变的**，为了处理这种类型的输入和输出，设计了**编码器——解码器**架构：

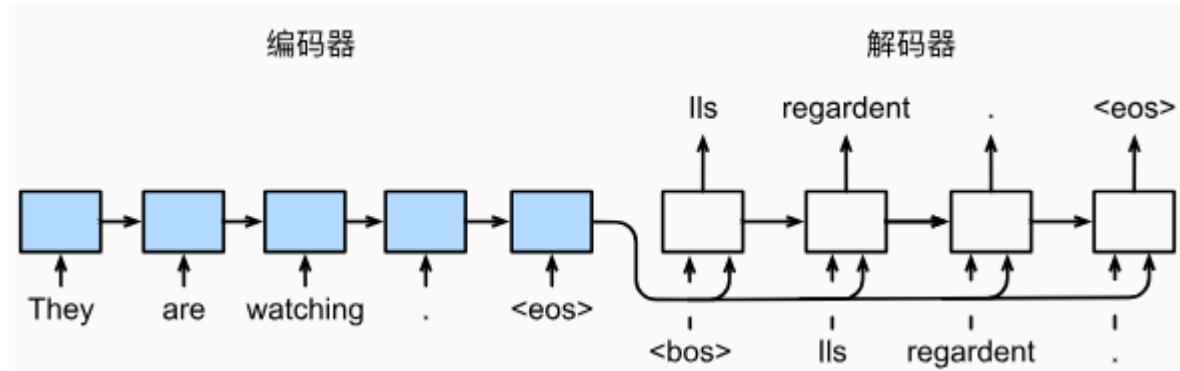


编码器 (Encoder)： 接受一个长度可变的序列作为输入，并将其转化为具有固定形状的编码状态（或者说将输入映射到某个向量空间）。

解码器 (Decoder)： 将固定形状的编码状态映射到长度可变的序列（将向量空间的向量逆映射回人类理解的数据结构）。

基于循环神经网络的编码器-解码器 (Seq2Seq学习)

以机器翻译为例，介绍如何使用编码器——解码器架构实现Seq2Seq的学习



如图是使用RNN作为编码器和解码器的机器翻译模型，其中<eos>和<bos>分别表示序列的结束和开始。RNN编码器使用长度可变的序列作为输入，将其转换为固定形状的隐状态（将输入编码到隐状态）。然后独立的RNN解码器基于输入序列的编码信息和输出序列已经看见的或生成的词元来预测下一个。

编码器:

首先将输入编码为各个时间步的隐状态

$$h_t = f(x_t, h_{t-1})$$

根据隐状态生成上下文变量 c

$$c = q(h_1, \dots, h_T)$$

解码器:

基于上一次的预测、上下文生成隐状态

$$s_{t'} = g(y_{t'-1}, c, s_{t'-1})$$

根据 t' 时刻的隐状态生成预测值（全连接层）

$$o = \text{softmax}(s_{t'} w_{sq} + b_q)$$

图中的例子 $c = h_T$ ，只用到了最后一个隐状态

基于循环神经网络的编码器-解码器 (Seq2Seq学习)

编码器:

首先将输入编码为各个时间步的隐状态

$$h_t = f(x_t, h_{t-1})$$

根据隐状态生成上下文变量 c

$$c = q(h_1, \dots, h_T)$$

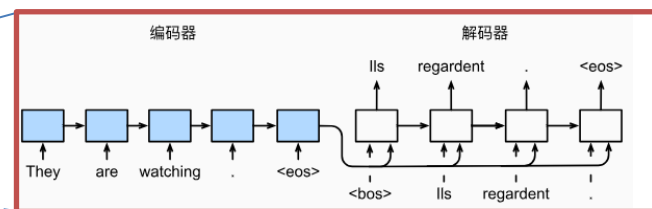
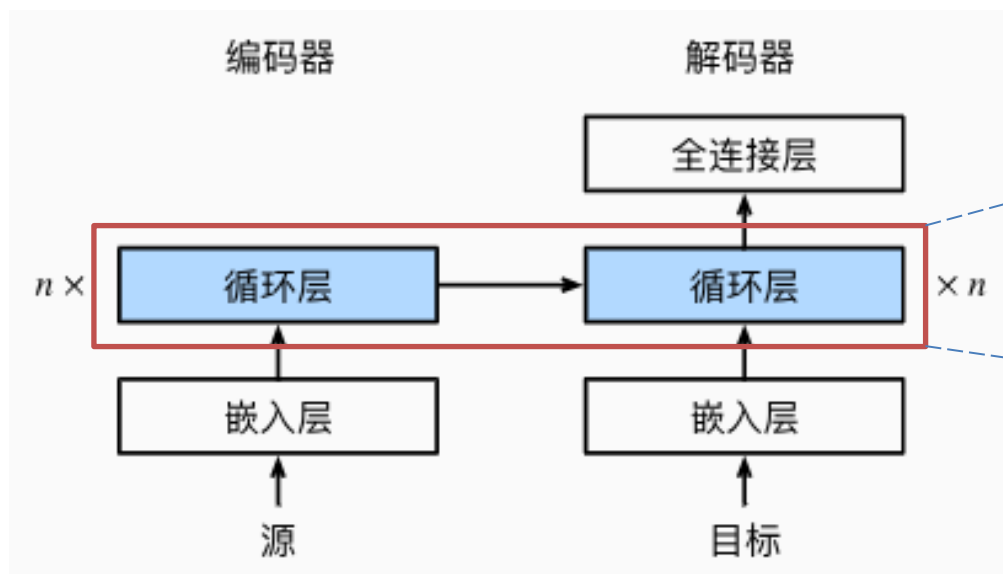
解码器:

基于上一次的预测、上下文生成隐状态

$$s_{t'} = g(y_{t'-1}, c, s_{t'-1})$$

根据 t' 时刻的隐状态生成预测值 (全连接层)

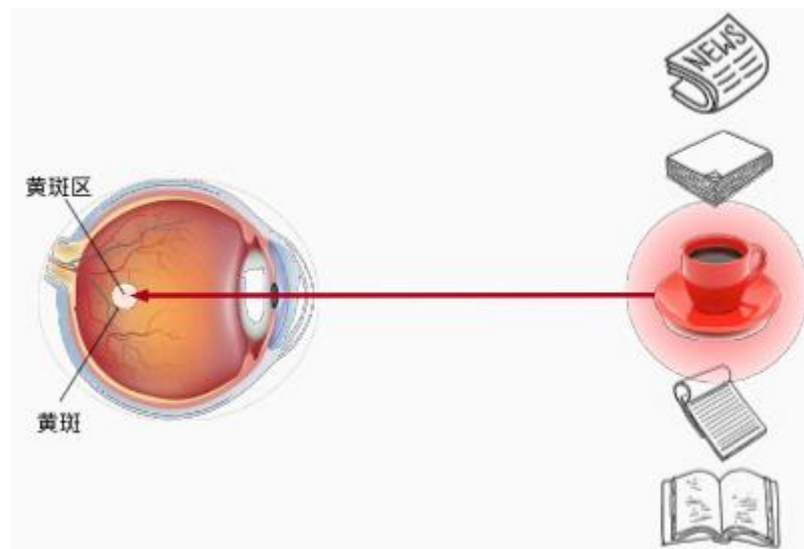
$$o_{t'} = \text{softmax}(s_{t'} w_{sq} + b_q)$$



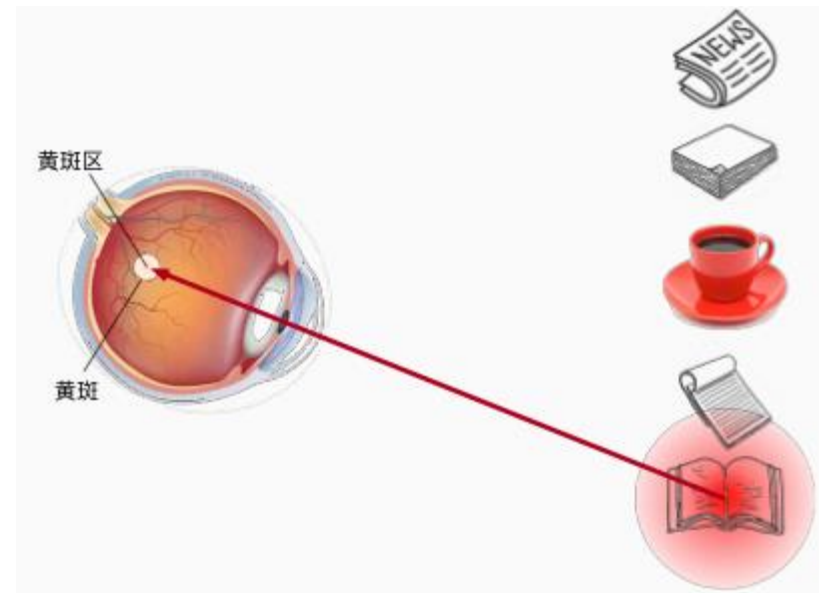
注意力机制 (Attention)

注意力是一种稀缺资源，因为生物的精力有限，将注意力分配到感兴趣的事情上才能提高能量的利用效率。

生物学上的注意力提示：**非自主性提示**和**自主性提示**



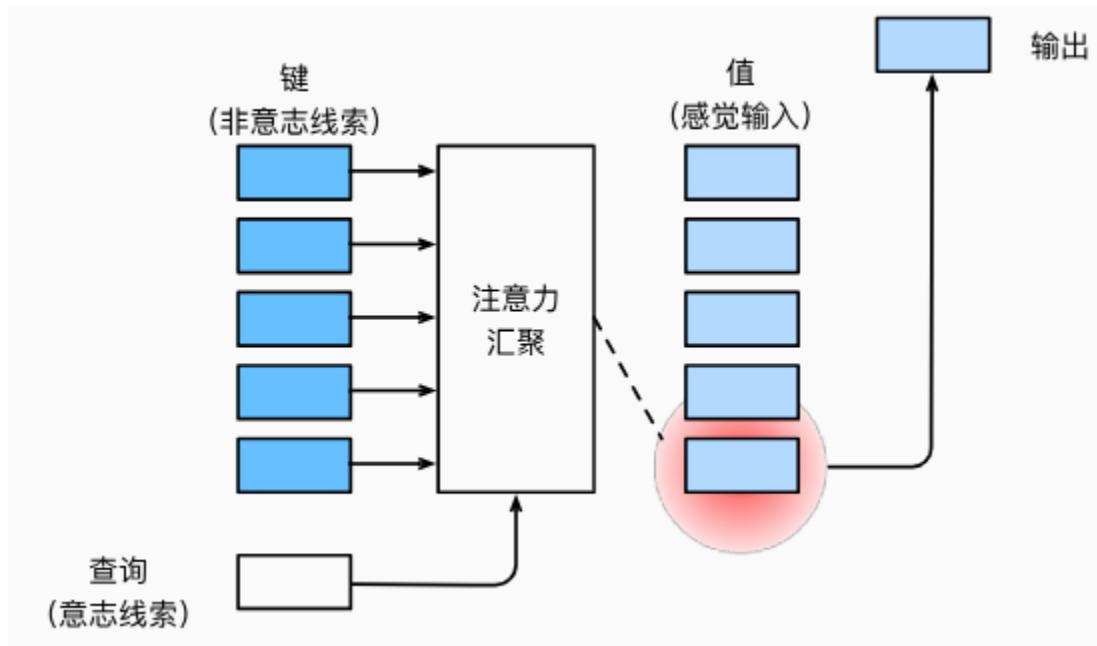
非自主性提示 (**红色**咖啡杯)



自主性提示 (喝完咖啡**想**读书)

注意力机制 (Attention)

基于生物注意力提示设计的神经网络注意力机制框架



注意力机制通过注意力汇聚将**查询**（自主性提示）和**键**（非自主性提示）结合在一起，实现对**值**（感官输入）的选择倾向

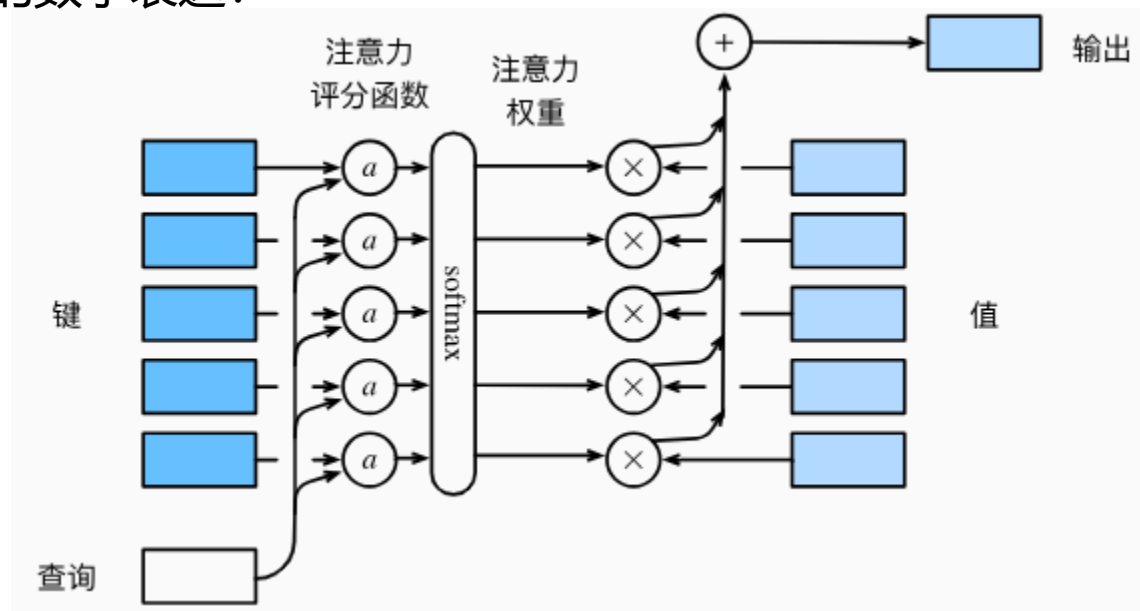
注意力汇聚的一般表达： $f(x) = \sum_{i=1}^n \alpha(x, x_i) y_i$

相当于动态加权，
权重在推理时生成

其中 x 是查询， (x_i, y_i) 是键值对，注意力汇聚是 y_i 的加权平均，将查询 x 和键 x_i 之间的关系建模为**注意力权重** $\alpha(x, x_i)$

注意力机制 (Attention)

注意力汇聚的数学表达:



假设有一个查询 $q \in \mathbb{R}^q$ 和 m 个“键-值”对 $(k_1, v_1), \dots, (k_m, v_m)$, 其中 $k_i \in \mathbb{R}^k$ $v_i \in \mathbb{R}^v$. 注意力汇聚函数 f 就被表示成值的加权和:

$$f(\mathbf{q}, (\mathbf{k}_1, \mathbf{v}_1), \dots, (\mathbf{k}_m, \mathbf{v}_m)) = \sum_{i=1}^m \alpha(\mathbf{q}, \mathbf{k}_i) \mathbf{v}_i \in \mathbb{R}^v$$

其中查询 q 和键 k_i 的注意力权重 (标量) 是通过**注意力评分函数** a 将两个向量映射成标量, 再经过 $softmax$ 运算得到的

将注意力转化为
概率分布 (稳定的
分布可以加快
训练过程)

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \text{softmax}(a(\mathbf{q}, \mathbf{k}_i)) = \frac{\exp(a(\mathbf{q}, \mathbf{k}_i))}{\sum_{j=1}^m \exp(a(\mathbf{q}, \mathbf{k}_j))}$$

注意力的
主要实现

注意力机制 (Attention)

注意力汇聚的数学表达:

$$f(\mathbf{q}, (\mathbf{k}_1, \mathbf{v}_1), \dots, (\mathbf{k}_m, \mathbf{v}_m)) = \sum_{i=1}^m \alpha(\mathbf{q}, \mathbf{k}_i) \mathbf{v}_i \in \mathbb{R}^v, \quad \alpha(\mathbf{q}, \mathbf{k}_i) = \text{softmax}(a(\mathbf{q}, \mathbf{k}_i)) = \frac{\exp(a(\mathbf{q}, \mathbf{k}_i))}{\sum_{j=1}^m \exp(a(\mathbf{q}, \mathbf{k}_j))}$$

两种常见的注意力评分函数 a :

当查询和键是**不同长度**的矢量时, 可以使用加性注意力作为评分函数。给定查询 $q \in \mathbb{R}^q$ 和键 $k \in \mathbb{R}^k$, **加性注意力 (additive attention)** 的评分函数为:

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^\top \tanh(\mathbf{W}_q \mathbf{q} + \mathbf{W}_k \mathbf{k}) \in \mathbb{R}$$

其中可学习的参数是 $W_q \in \mathbb{R}^{h \times q}$ 、 $W_k \in \mathbb{R}^{h \times k}$ 和 $w_v \in \mathbb{R}^h$ 。通过 W_q 和 W_k 将长度不同的 q, k 映射到相同的空间, 使用累加将它们连接起来, 而后使用经过一个多层感知机 (MLP) (对应公式中的权重 W_v^T)得到查询和键的相关性。

使用点积可以得到计算效率更高的评分函数, 但是点积操作要求查询和键具有**相同的长度** d , 为确保无论向量长度如何, 点积的方差在不考虑向量长度的情况下仍然是1, 我们再将点积除以 \sqrt{d} , 则**缩放点积注意力 (scaled dot-product attention)** 评分函数为:

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^\top \mathbf{k} / \sqrt{d}.$$

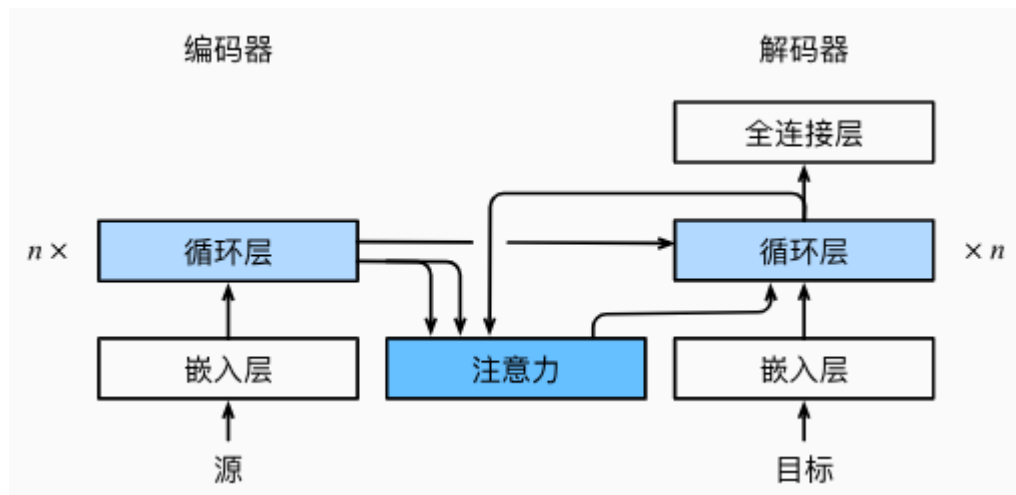
加性注意力相当于神经网络自己去寻找 q, k 之间的相似度联系, 而缩放点积注意力则是人根据经验设计的相似度关系。

注意力机制 (Attention) +RNN

在前面提到的机器翻译模型中，在编码器和解码器之间交流信息的上下文 $c = h_T$ 表示只关注输入序列最后一个数据。然而实际上，输入序列 t' 可能与任意时刻的输入都有关系，将上下文替换为带注意力的上下文：

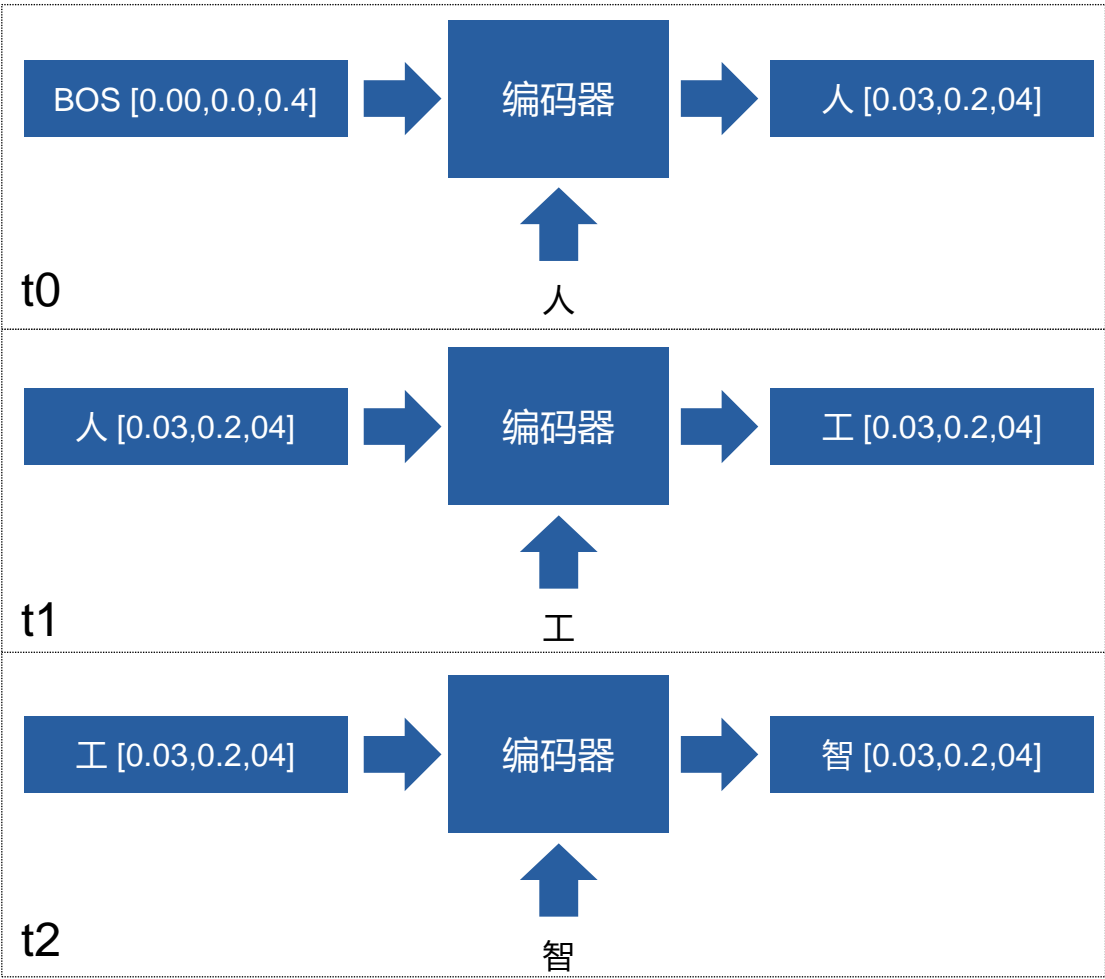
$$c_{t'} = \sum_{t=1}^T \alpha(s_{t'-1}, h_t) h_t$$

在每个时间步 t' 都要重新生成上下文。其中时间步 $t' - 1$ 时的解码器隐状态 $s_{t'-1}$ 是查询，编码器隐状态 h_t 即是键也是值，注意力评分使用加性注意力



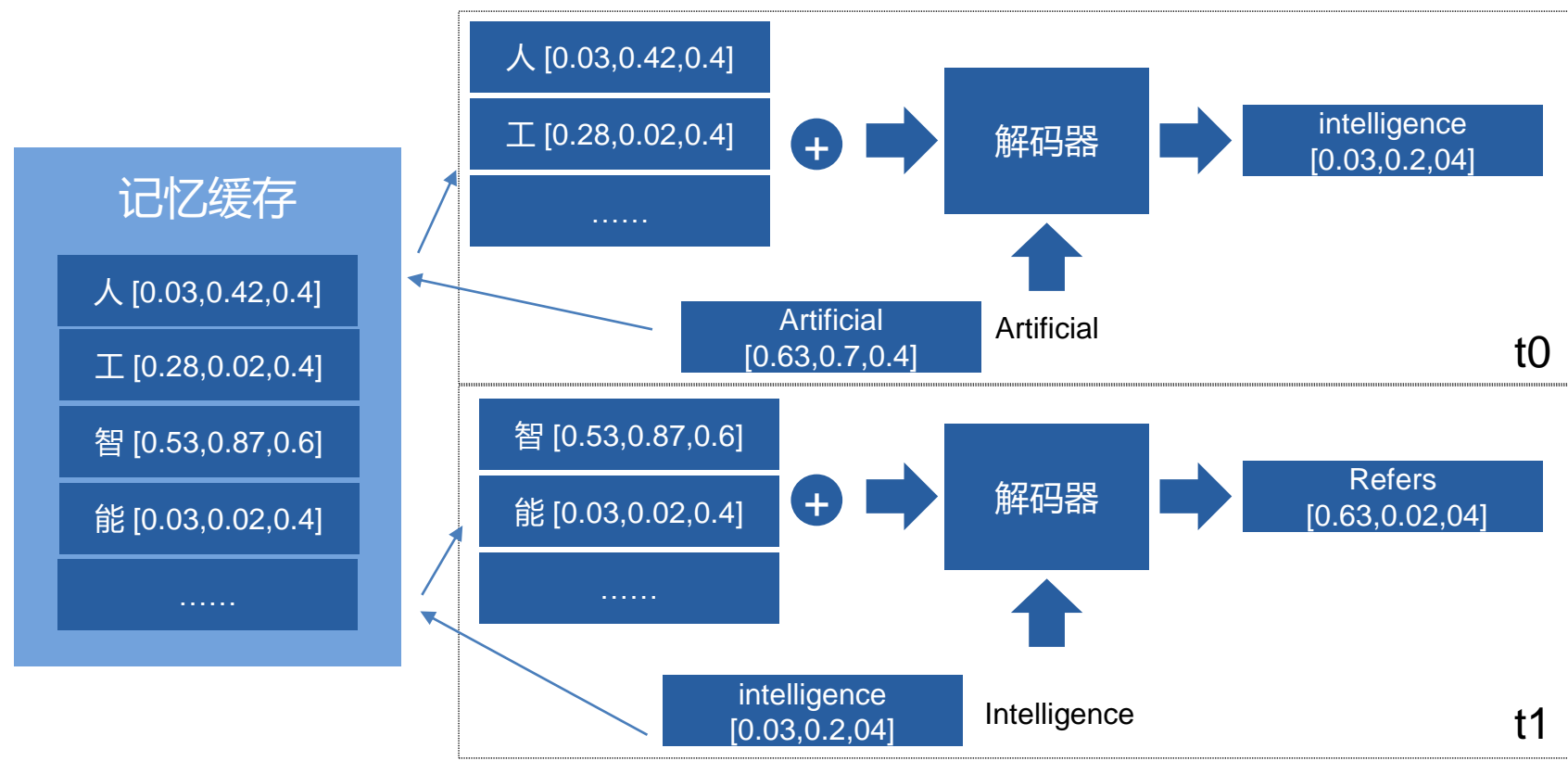
注意力机制 (Attention) +RNN

人工智能指由人制造出来的机器所表现出来的智慧
 Artificial intelligence refers to the intelligence of a machine made by a human beings



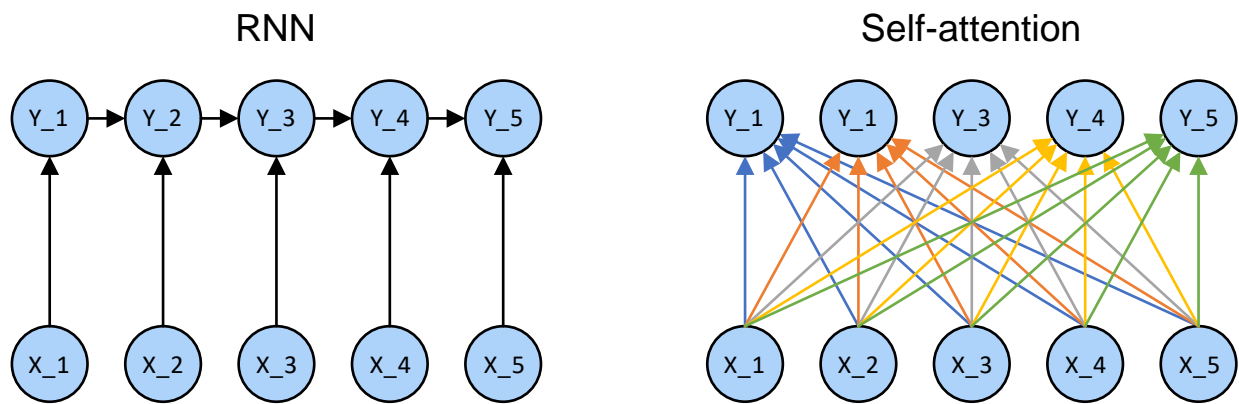
注意力机制 (Attention) +RNN

人工智能指由人制造出来的机器所表现出来的智慧
 Artificial intelligence refers to the intelligence of a machine made by a human beings



自注意力 (self-attention)

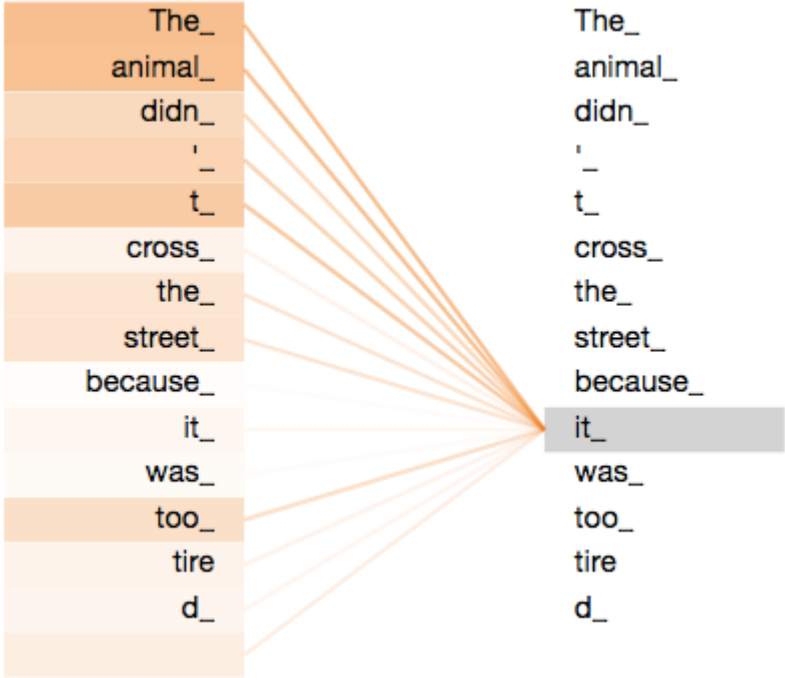
上面的例子中，查询、键和价值都来自以RNN为基础的编码器和解码器。然而RNN每次只能输入一个时刻的数据，无法实现并行计算。考虑对所有的输入数据使用注意力机制进行加权求和，也可以实现类似RNN的效果，这就是**自注意力 (self-attention)**



同RNN的序列Y一样，self-attention的输出序列Y也包含输入X之间的关联信息。
 值得注意的是，self-attention也可以看到后面的信息（双向RNN也可以实现这个效果）
 Self-attention没有RNN对前一个时刻的数据依赖问题，
 则可以实现大规模并行

自注意力 (self-attention)

上面的例子中，查询、键和值都来自以RNN为基础的编码器和解码器。然而RNN每次只能输入一个时刻的数据，无法实现并行计算。考虑对所有的输入数据使用注意力机制进行加权求和，也可以实现类似RNN的效果，这就是**自注意力 (self-attention)**



The animal didn't cross the street
because **it** was too tired

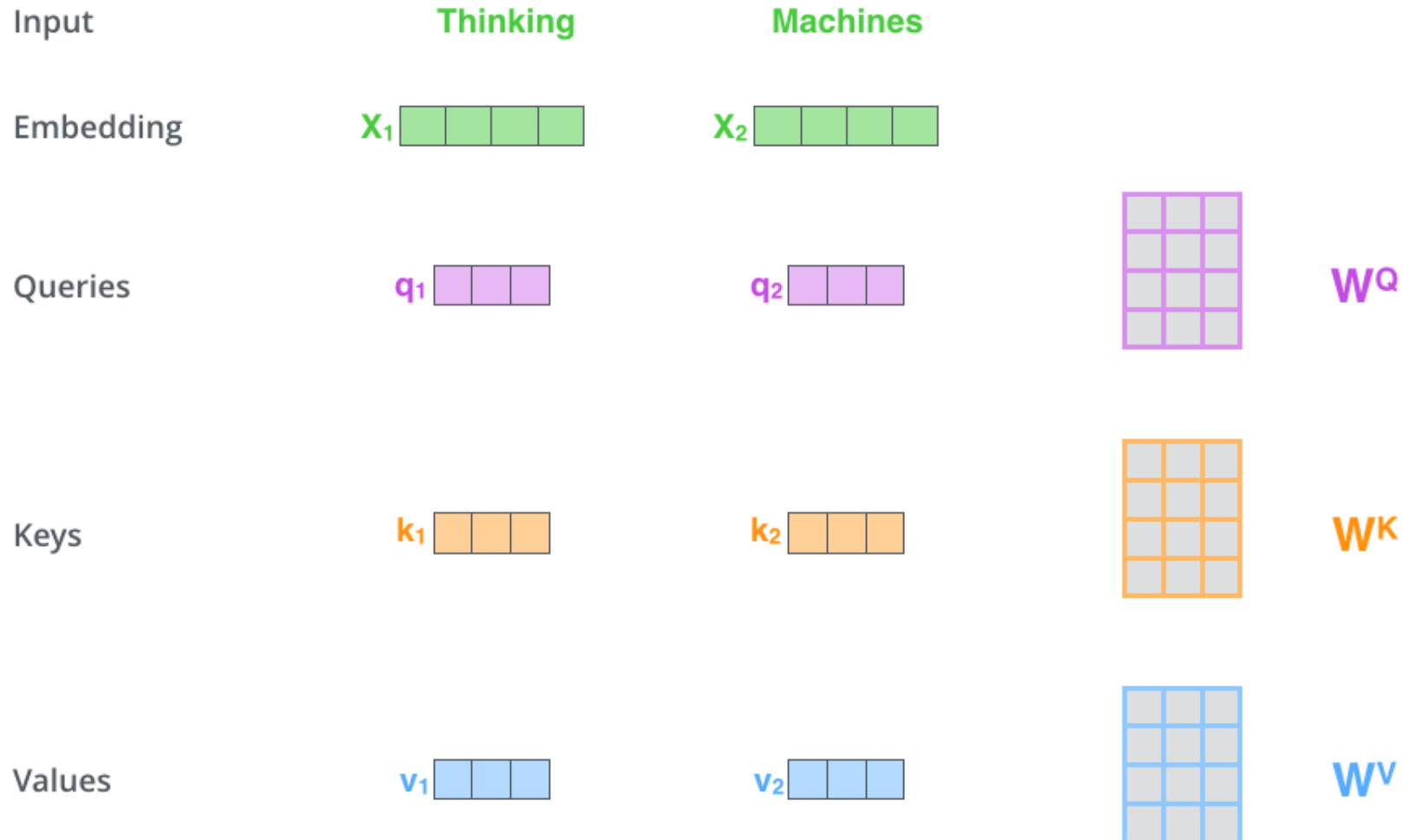


Attention

The animal didn't cross the street
because **the animal** was too tired

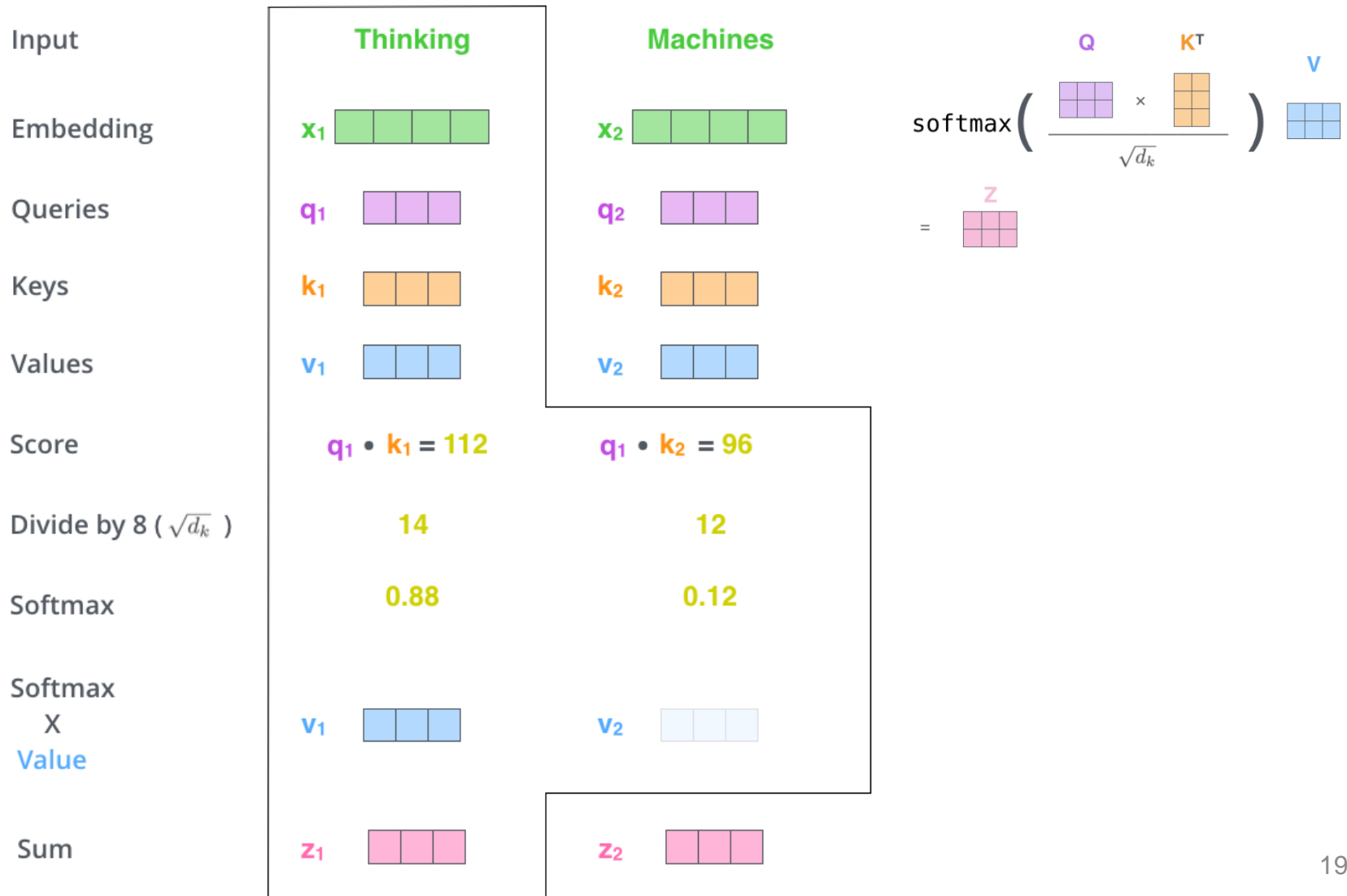
自注意力（ self-attention ） 计算过程

- Self, 自学习输入X到QKV的建立过程



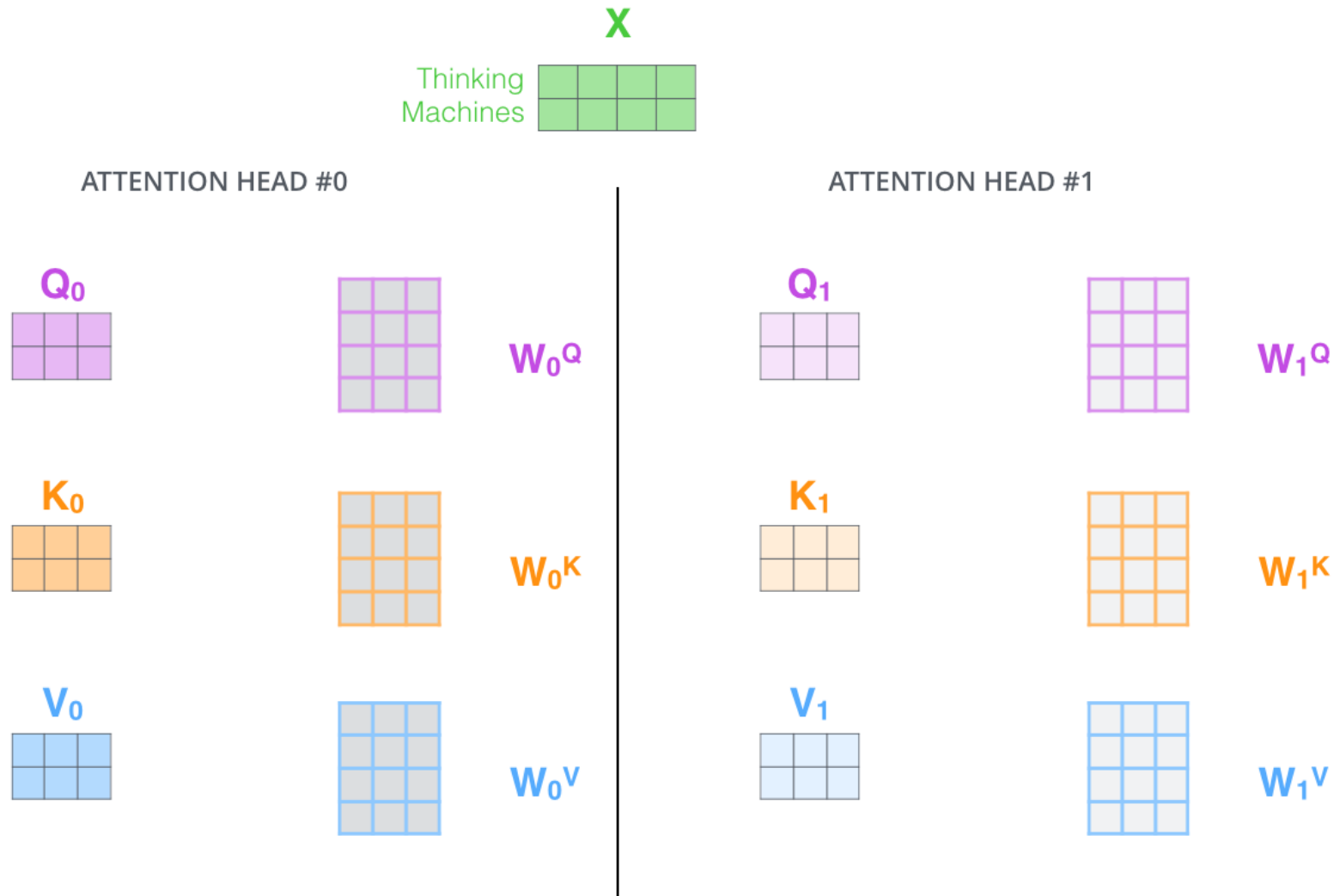
自注意力（self-attention）计算过程

- Attention, 汇聚注意力（加权求和）



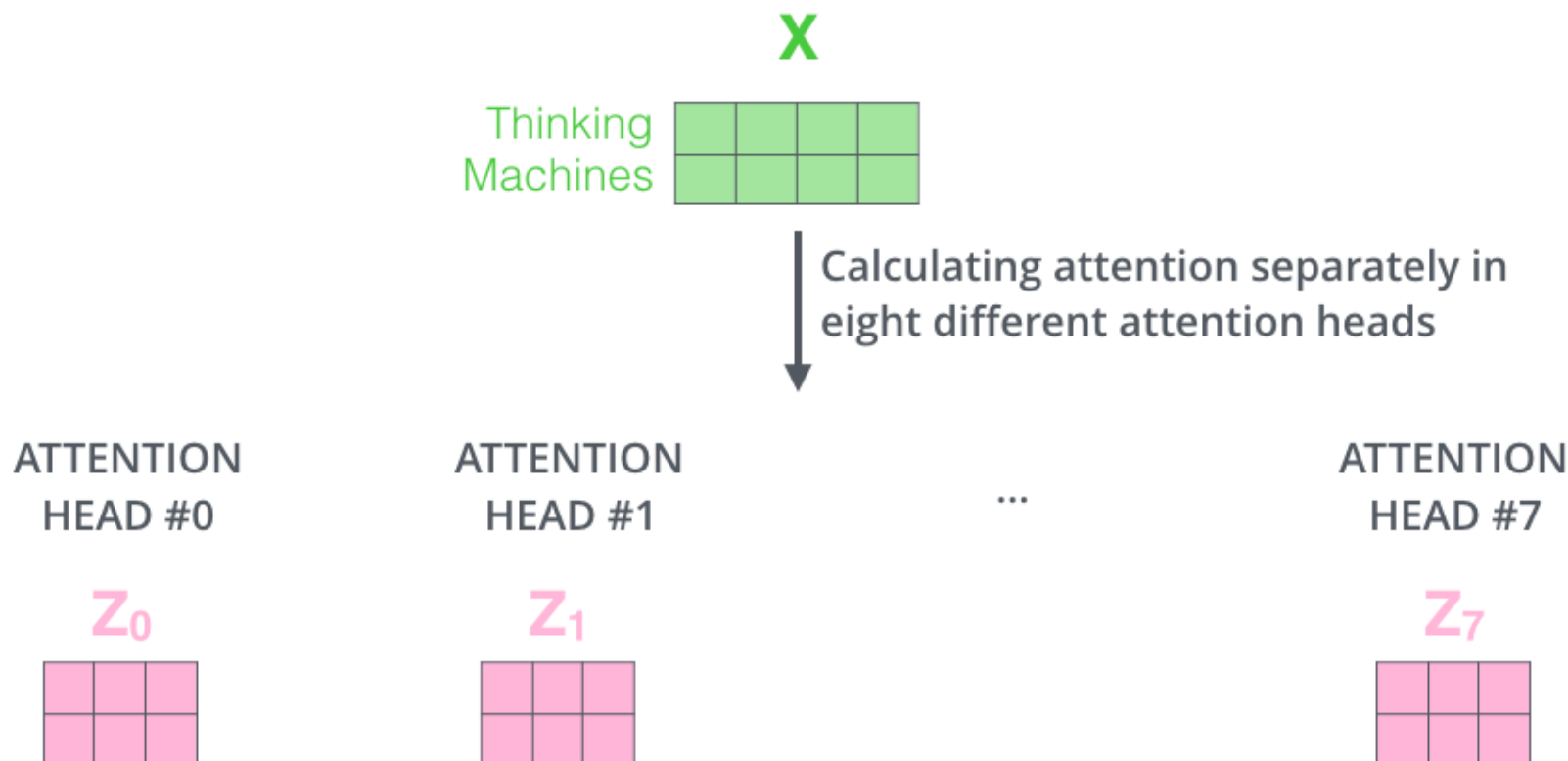
多头自注意力 (self-attention)

多头注意力其实就是有更多不同的QKV建立过程



多头自注意力 (self-attention)

每一头都有一个有关输入的注意力输出

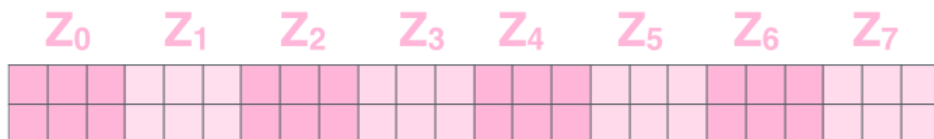


每个单词产生了8个向量 (8个注意力头)

多头自注意力 (self-attention)

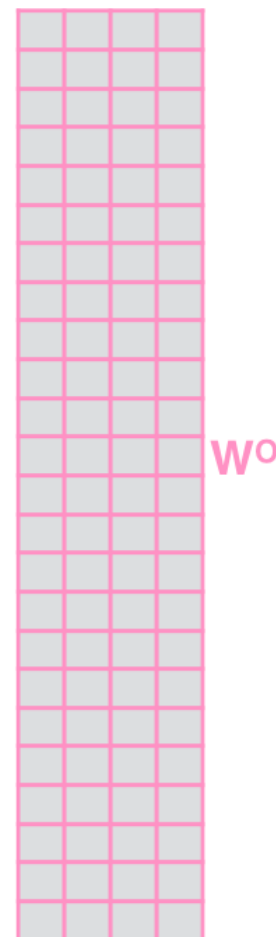
我们希望每个单词仍然对应一个向量，以降低后续计算量

1) Concatenate all the attention heads



2) Multiply with a weight matrix W^O that was trained jointly with the model

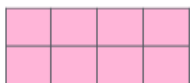
\times



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

Z

=



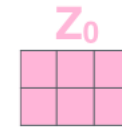
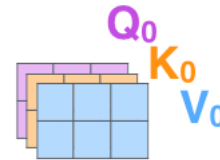
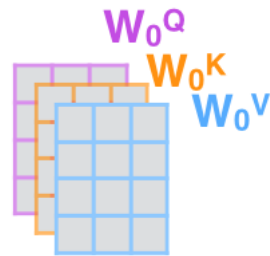
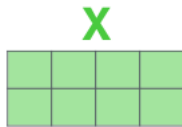
输出Z可以看成是所有头的信息的融合

多头自注意力 (self-attention)

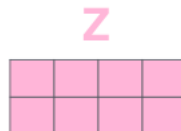
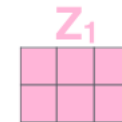
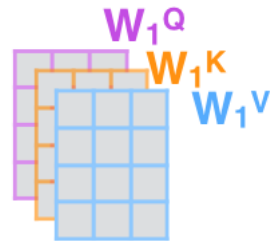
总结:

- 1) This is our input sentence*
- 2) We embed each word*
- 3) Split into 8 heads. We multiply X or R with weight matrices
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

Thinking
Machines



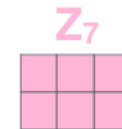
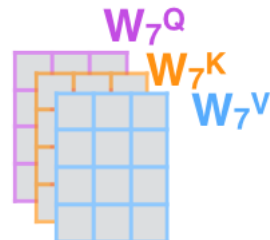
W^O



...

...

...

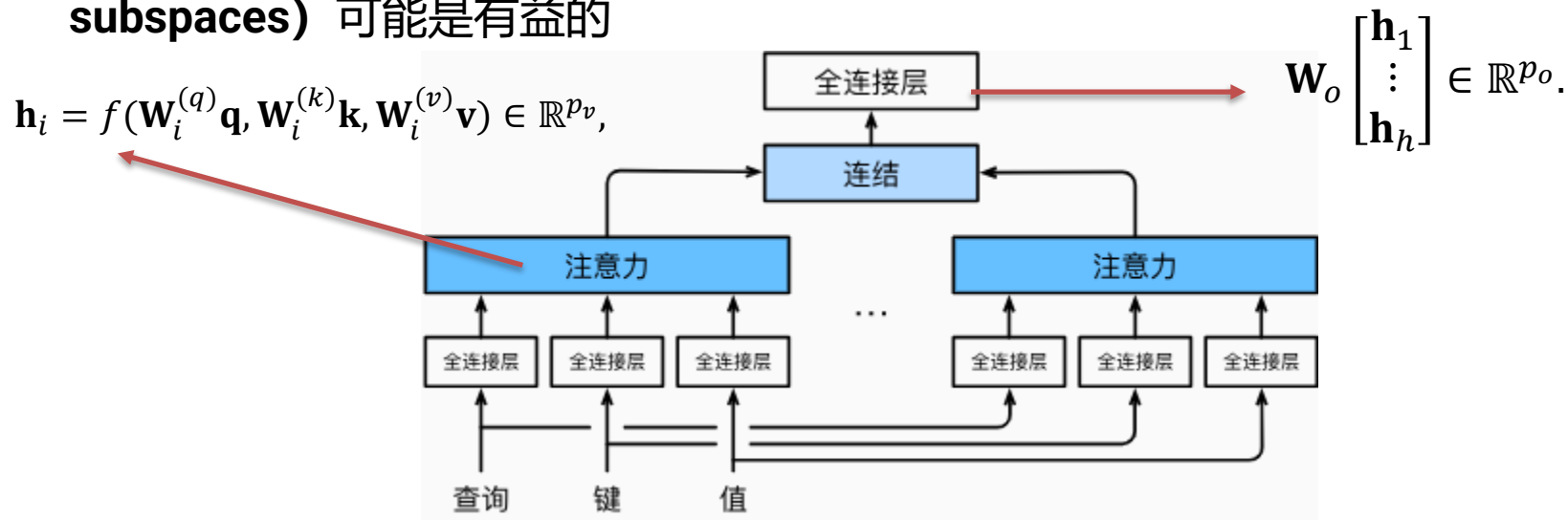


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



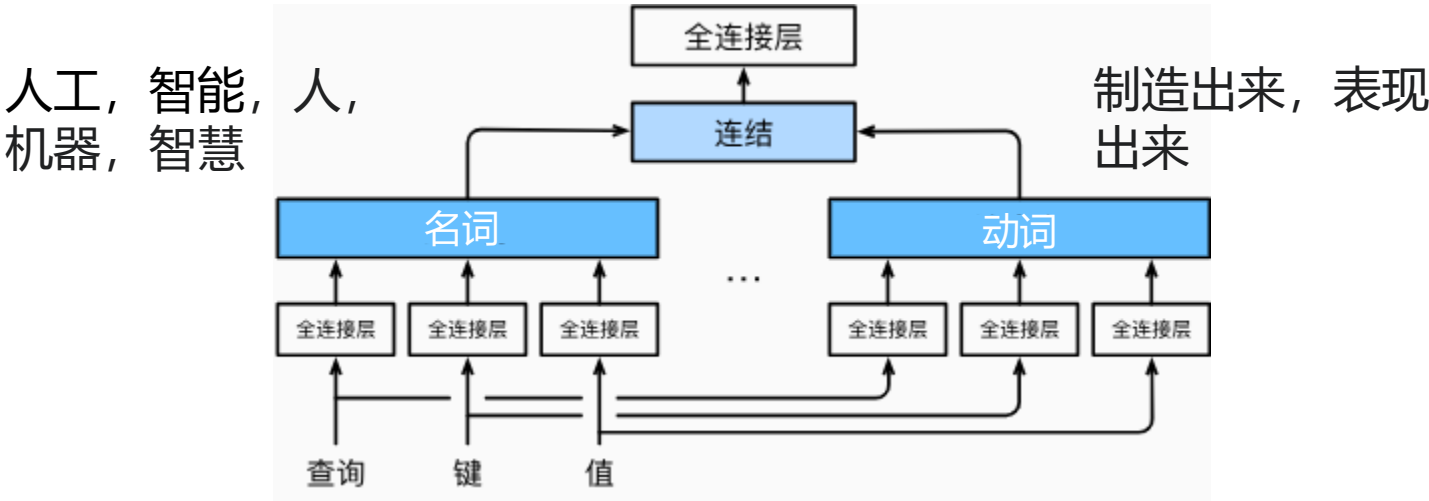
多头注意力

在实践中，当给定相同的查询、键和值的集合时，我们希望模型可以**基于相同的注意力机制学习到不同的行为**，然后将不同的行为作为**知识组合起来**，捕获序列内各种范围的依赖关系（例如，短距离依赖和长距离依赖关系）。因此，允许注意力机制组合使用查询、键和值的不同**子空间表示（representation subspaces）**可能是有益的



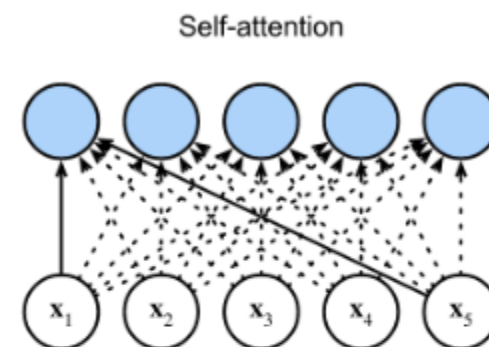
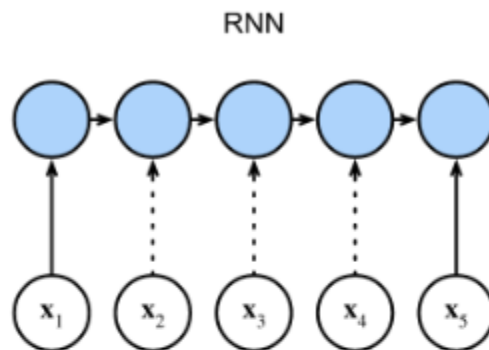
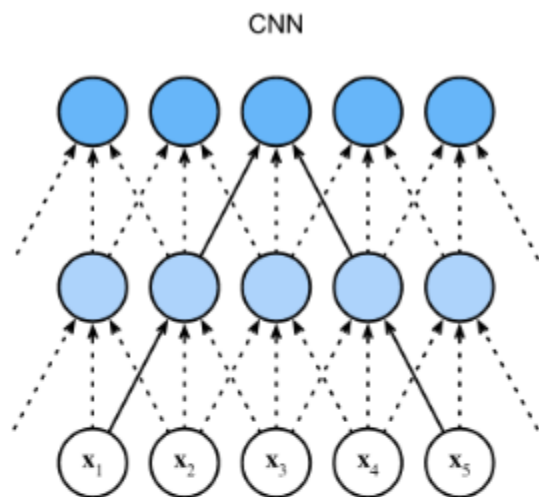
用独立学习得到的 h 组不同的**线性投影（linear projections）**来变换查询、键和值

有关多头注意力的理解：神经网络从不同的角度挖掘信息



人工智能指由人制造出来的机器所表现出来的智慧

对比卷积神经网络、循环神经网络和自注意力



	CNN	RNN	自注意力
计算复杂度	$O(knd^2)$	$O(nd^2)$	$O(n^2d)$
并行度	$O(n)$	$O(1)$	$O(n)$
最长路径	$O(n/k)$	$O(n)$	$O(1)$

长序列计算慢

并行计算和

最短的最大路径长度

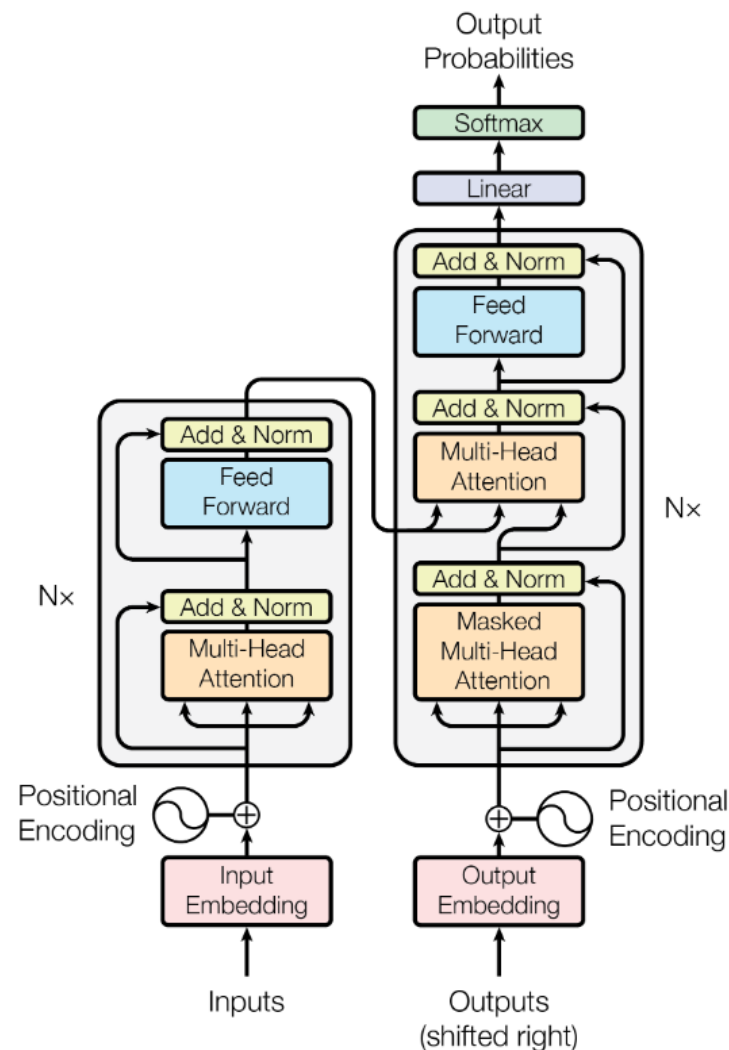
Transformer

- Transformer完全基于注意力机制，没有任何卷积层或循环神经网络层。
- Transformer的编码器和解码器是基于自注意力的模块叠加而成的

每个层都有两个子层（子层表示为 sublayer）。第一个子层是**多头自注意力**（multi-head self-attention）**汇聚**；第二个子层是**基于位置的前馈网络**（position-wise feed-forward network）

加&规范化进行**残差连接**和**归一化**：

- 加的操作参考率ResNet的残差连接
- 归一化是保证输入数据的分布特征稳定，利用模型收敛，不同于CV任务，序列任务中常用的是Layer-Norm*



Transformer

1. 嵌入层 (Embedding)

将非结构数据转为结构数据 (向量)

One-hot编码:

Car -> [1,0,...,0]

Vehicle -> [0,1,...,0]

Oneshot编码出的单词之间是独立分布的，体现不了关联性

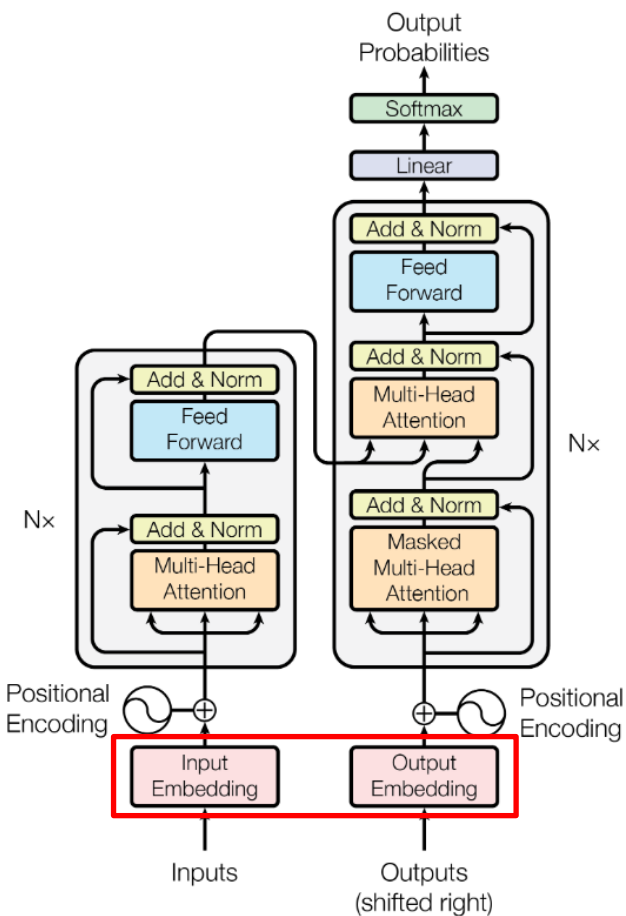
向量编码:

Car -> [0.9,0.2,...,0]

Vehicle -> [0.8,0.3,...,0]

使用神经网络或其它统计学方法进行编码，具有相近意义的单词所编码出的向量是近似的

嵌入层就是一个查找表，输入一个单词返回所存储的对应的向量

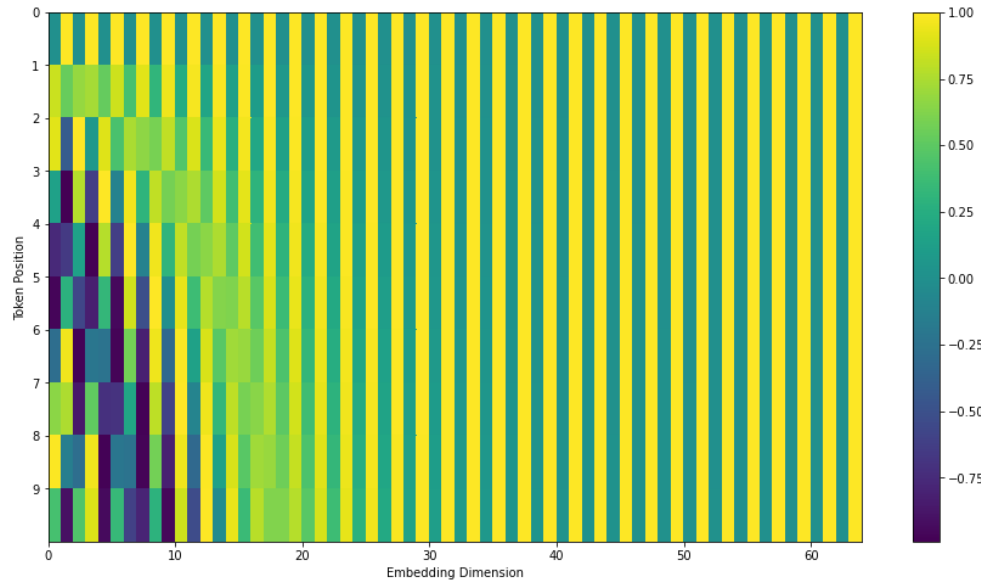


2.位置编码 (Position Encoding)

由于Self-Attention同时并行输入n个样本，缺少位置信息，因此增加位置编码。

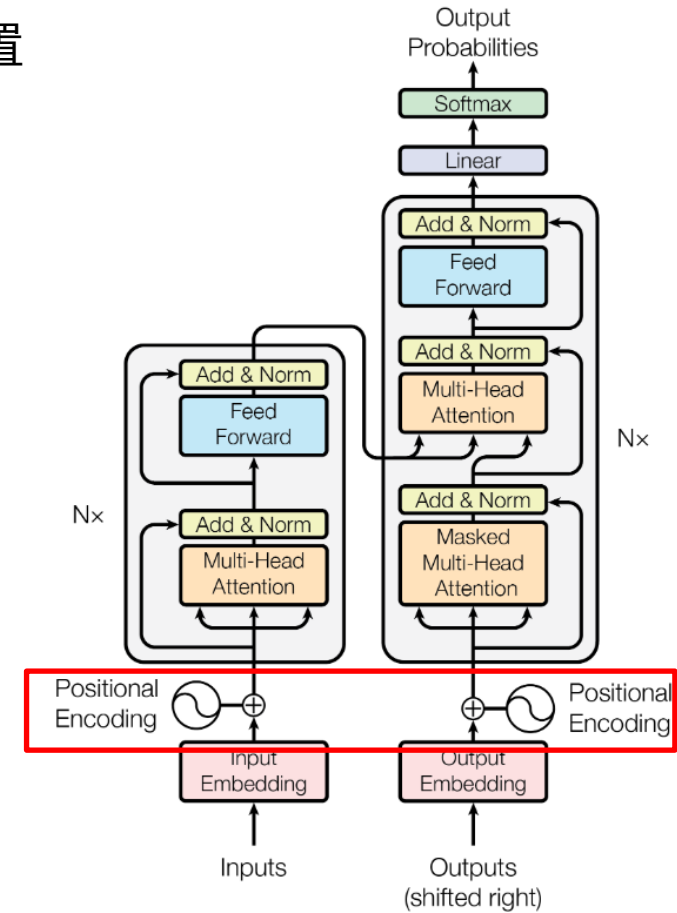
$$e_{t,2i} = \sin\left(\frac{t}{10000^{\frac{2i}{D}}}\right)$$

$$e_{t,2i+1} = \cos\left(\frac{t}{10000^{\frac{2i}{D}}}\right)$$

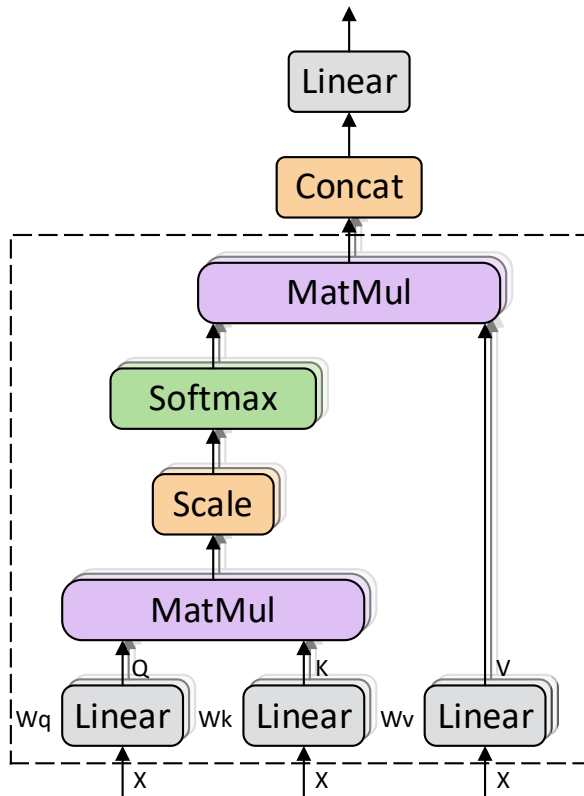


0	000
1	001
2	010
3	011

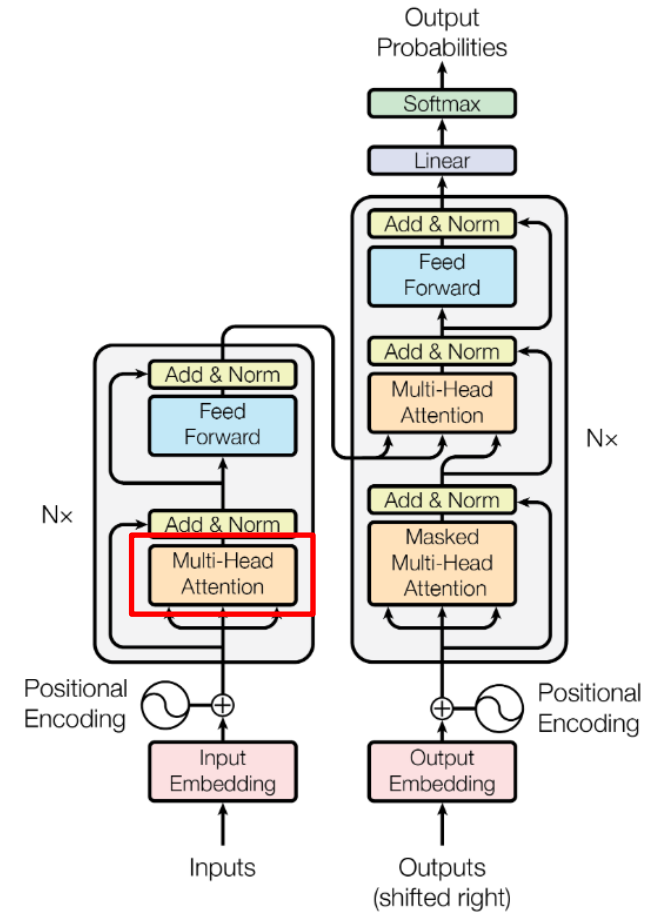
类似二进制编码，对应位置变化的频率不同



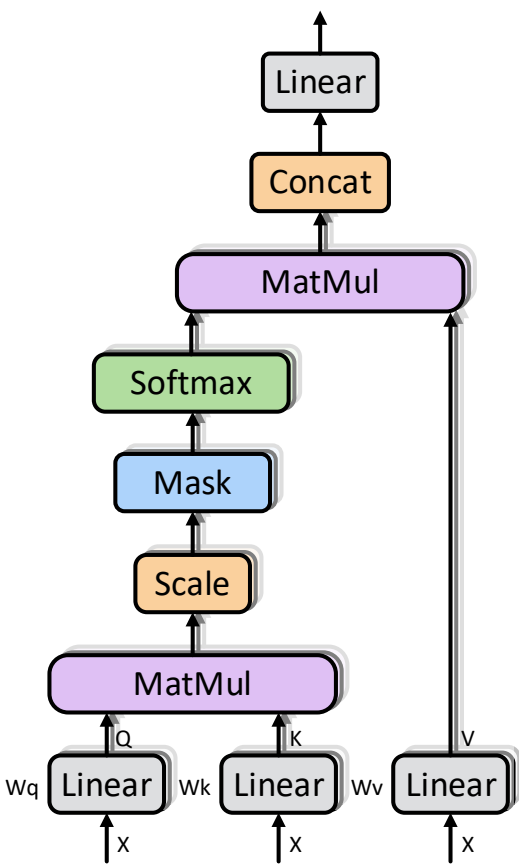
3.多头注意力 (Multi-Head Attention)



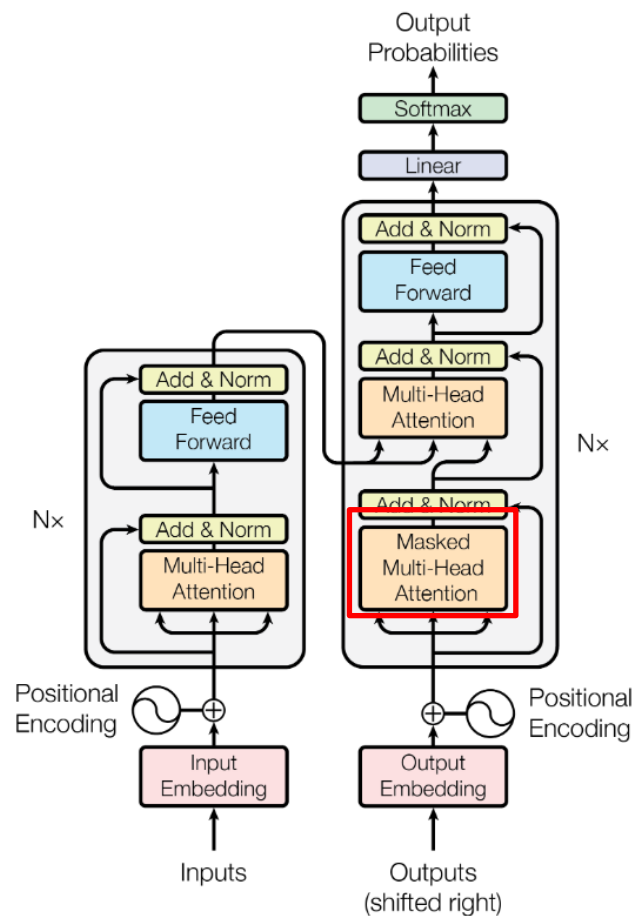
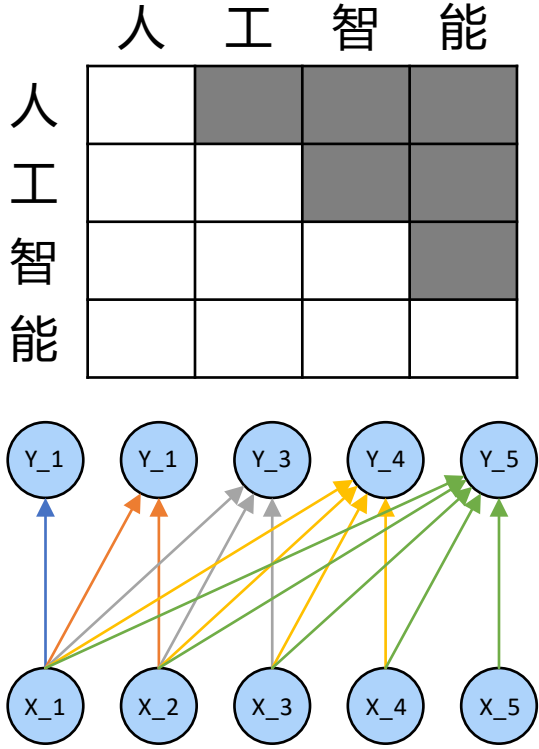
Encoder的Attention
就是普通的Attention



3.多头注意力 (Multi-Head Attention)



Decoder中有个Masked Multi-Head Attention, 训练时MASK防止模型看到未来的信息:



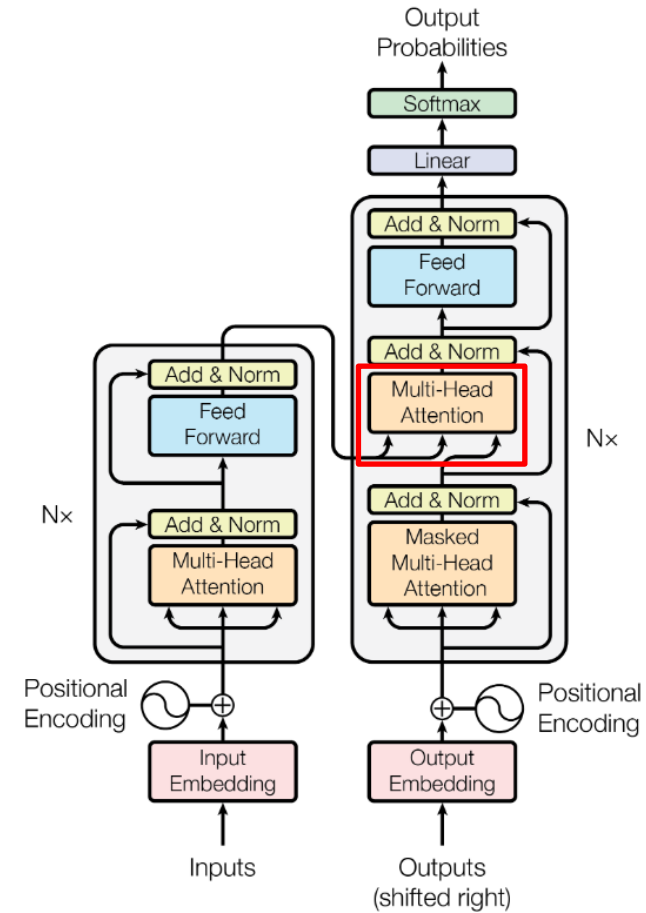
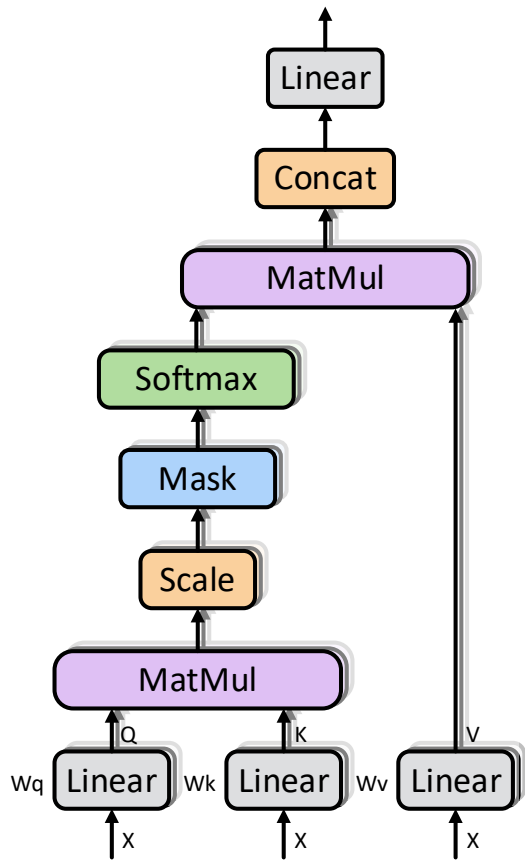
3.多头注意力 (Multi-Head Attention)

Decoder中两个Attention, 一个是Self-Attention, 另一个是Cross-Attention, Key和Value来自Encoder

人工
智能
指
由

Artificial Intelligence refers to

...

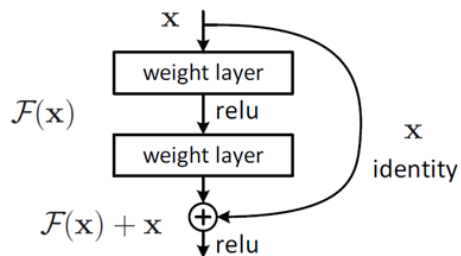


4.加&规范化

加：借鉴Resnet

规范化：稳定数据分布

ResNet引入残差单元，使得深层网络的学习变得简单



规范化：让数据分布特征稳定

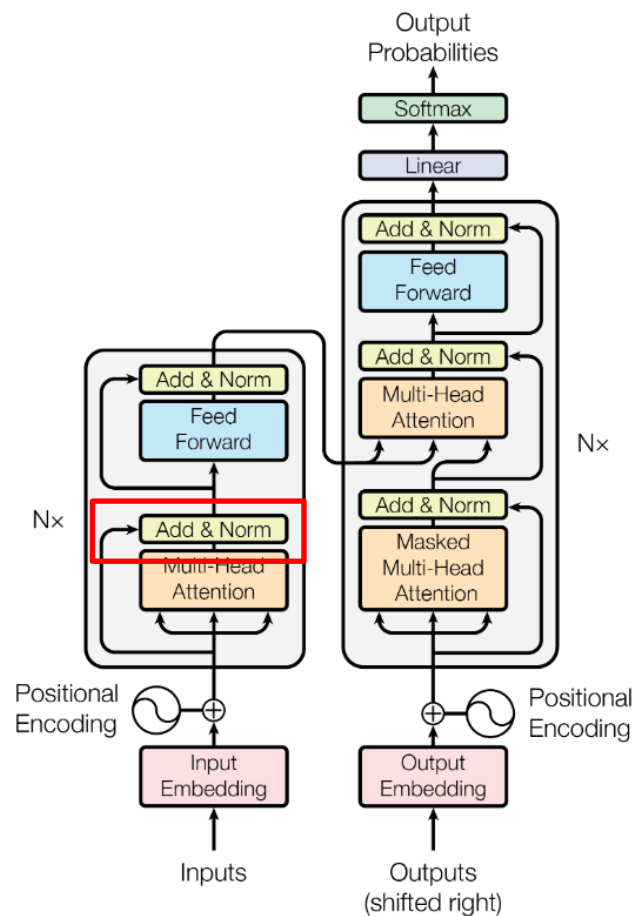
$$y = \frac{x - E[x]}{\sqrt{\text{Var}[x] + \epsilon}} * \gamma + \beta$$

$E[x]$: 均值

$\text{Var}[x]$: 方差

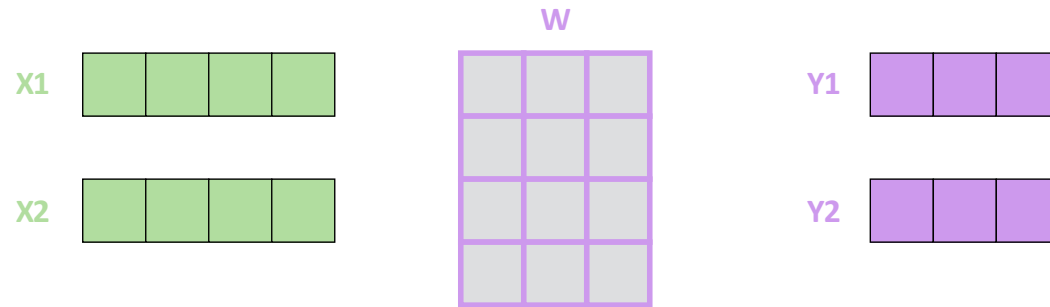
γ, β : 可学习参数

ϵ : $1e^{-6}$ 防止分母为0

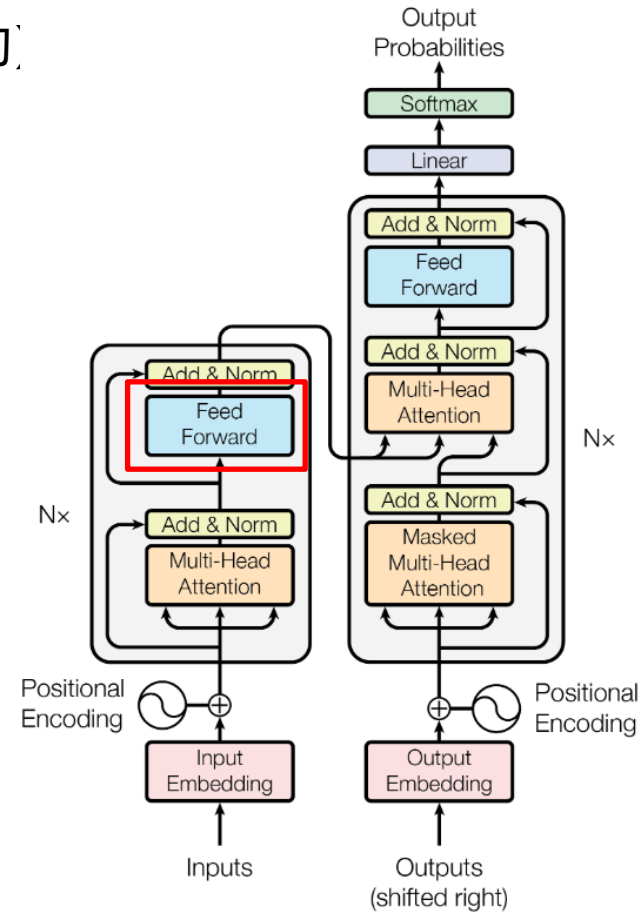


5. 逐位前馈网络

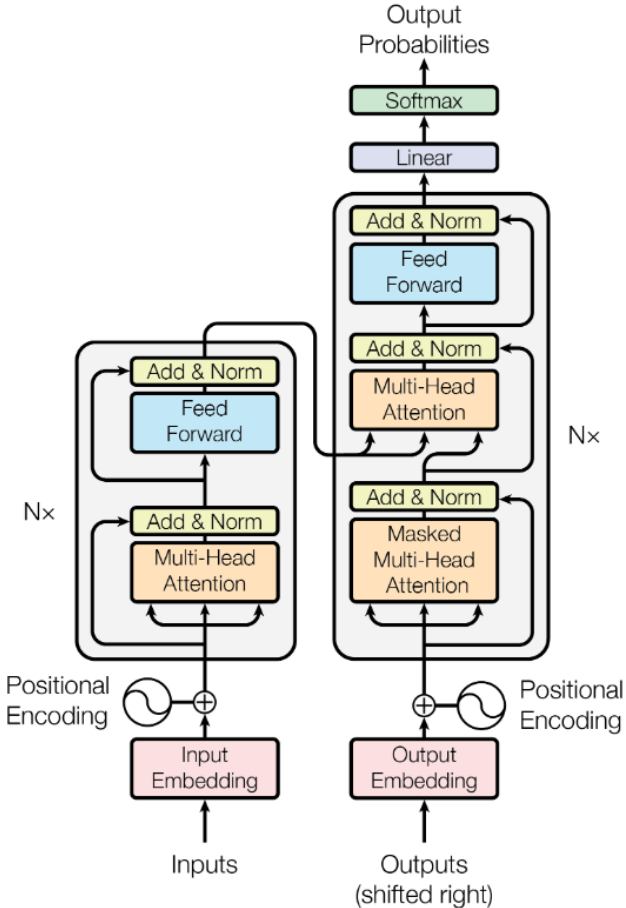
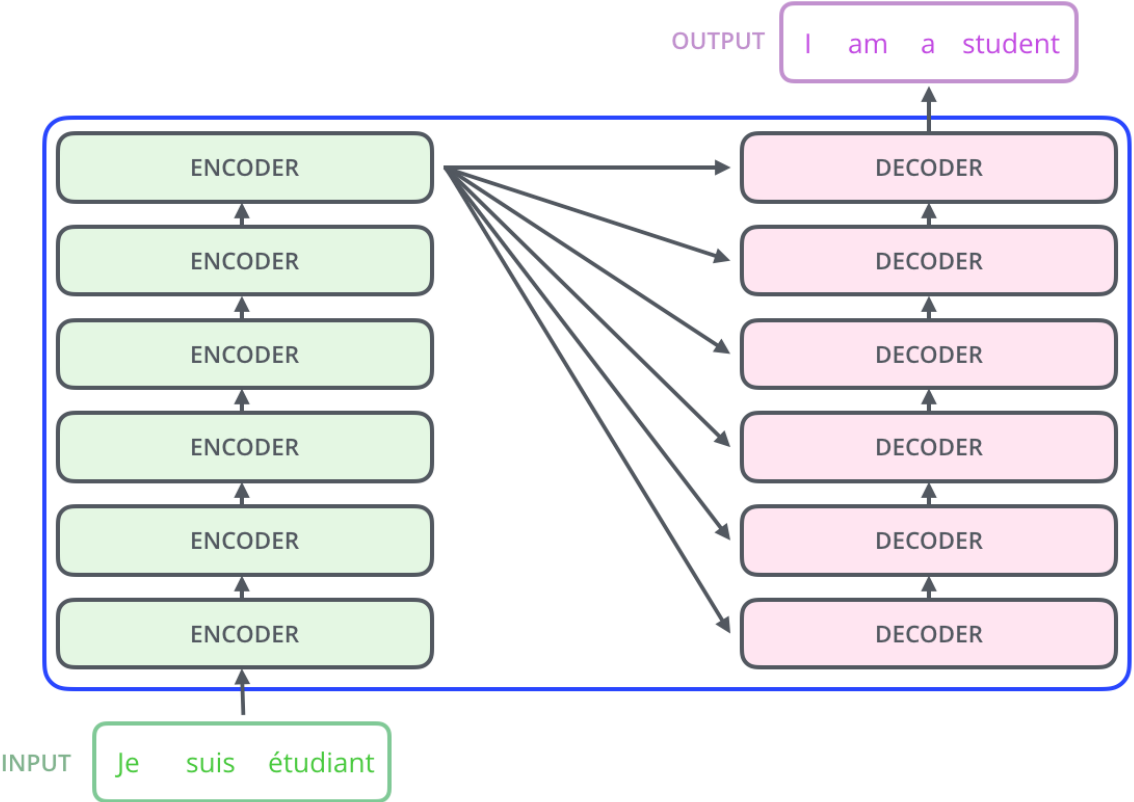
逐位：基于位置的（每个位置所应用的变换是相同的）
前馈网络：前连接层



$$Y = \text{Relu}(xW_1 + b_1)W_2 + b_2$$



Transformer中的Encoder-Decoder架构

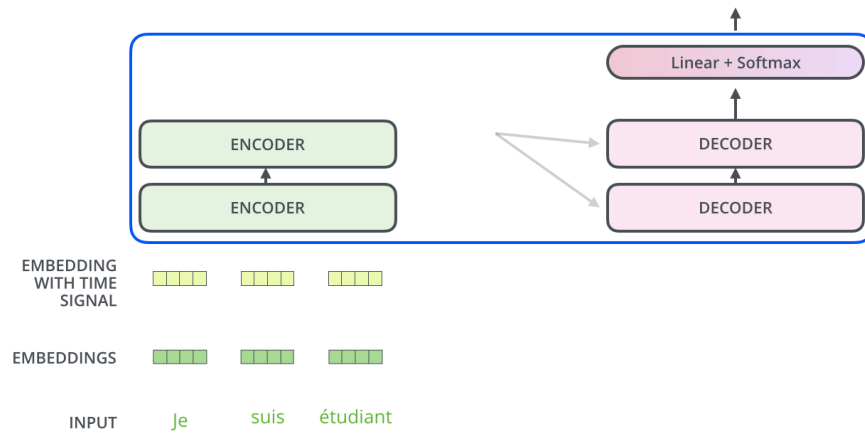


每个Encoder包含：一个多头自注意力，一个FFN
 每个Decoder包含：一个Masked的多头自注意力，
 一个多头交叉注意力，一个FFN

Transformer中的Encoder-Decoder架构

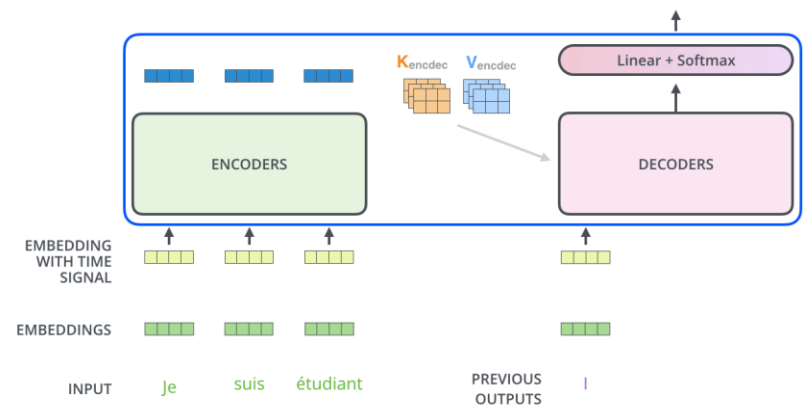
Decoding time step: 1 2 3 4 5 6

OUTPUT



Decoding time step: 1 2 3 4 5 6

OUTPUT



Transformer的加速

针对通用CNN和RNN设计的加速器不能直接用于Transformer^[1]:

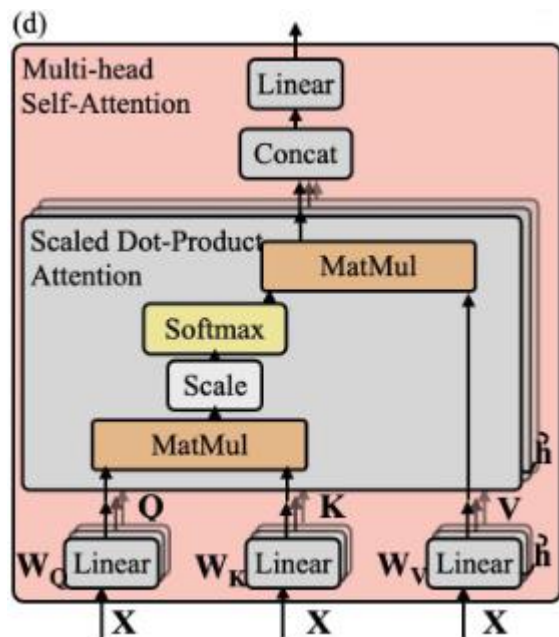
- Transformer中有大量矩阵乘矩阵，而且其中的参数都是来自上一层的中间结果，先前设计的加速器需要重新编程计算阵列的权重。
- Transformer引入了缩放点积注意力，计算模式更复杂。
- 先前设计的加速器流水线粒度是层，对于transformer来说较粗。

本文的贡献：

- 提出了ReTransformer，基于ReRAM的存内计算架构，用于加速Transformer的推理
- 使用矩阵分解优化缩放点积注意力中的矩阵乘法，消除数据依赖，降低计算延迟
- 使用存内计算实现的逻辑计算来实现混合softmax
- 子矩阵级的流水线粒度

[1] X. Yang, B. Yan, H. Li, and Y. Chen, "ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration," in *Proceedings of the 39th International Conference on Computer-Aided Design*, in ICCAD '20. New York, NY, USA: Association for Computing Machinery, Dec. 2020, pp. 1–9. doi: [10.1145/3400302.3415640](https://doi.org/10.1145/3400302.3415640).

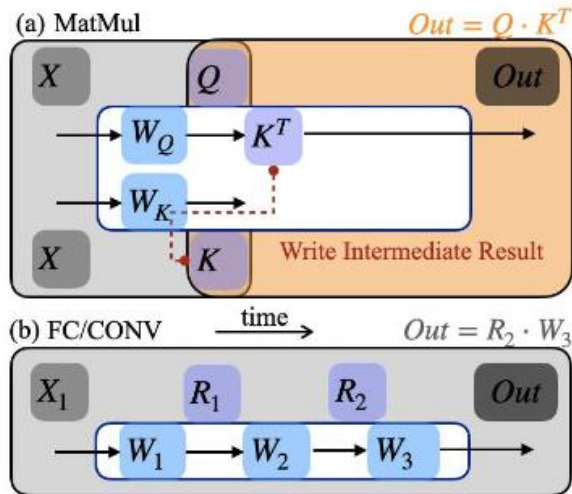
Transformer的加速



$$Q, K, V = X \cdot W_Q, X \cdot W_K, X \cdot W_V.$$

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

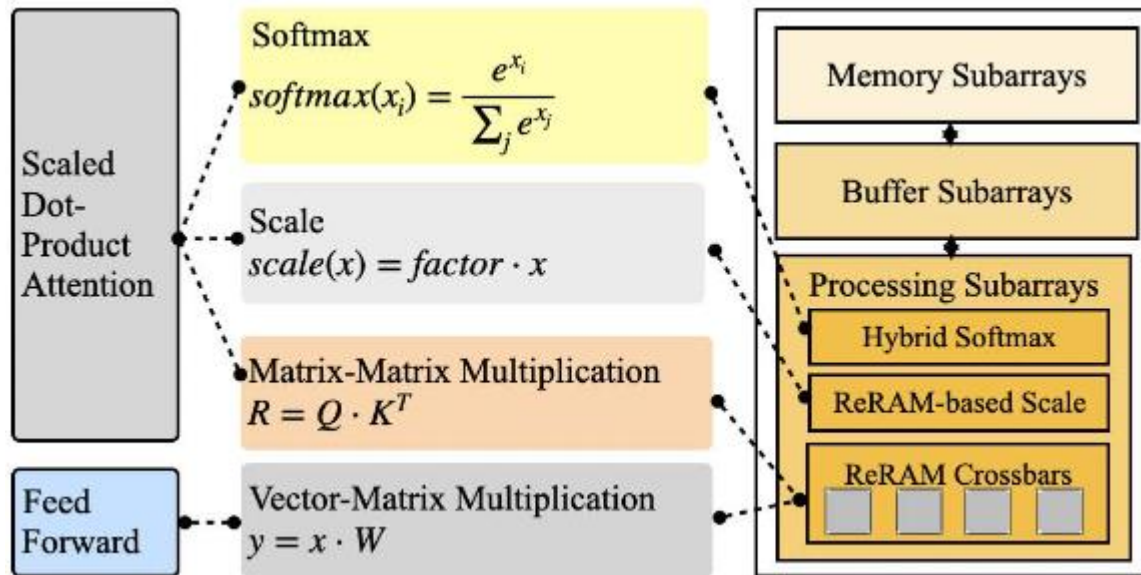
$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{d_k} e^{x_j}}.$$



- FC/CONV Computation
- MatMul Computation
- Input
- Trainable Weight
- Intermediate Result
- Output
- Write into ReRAM Device

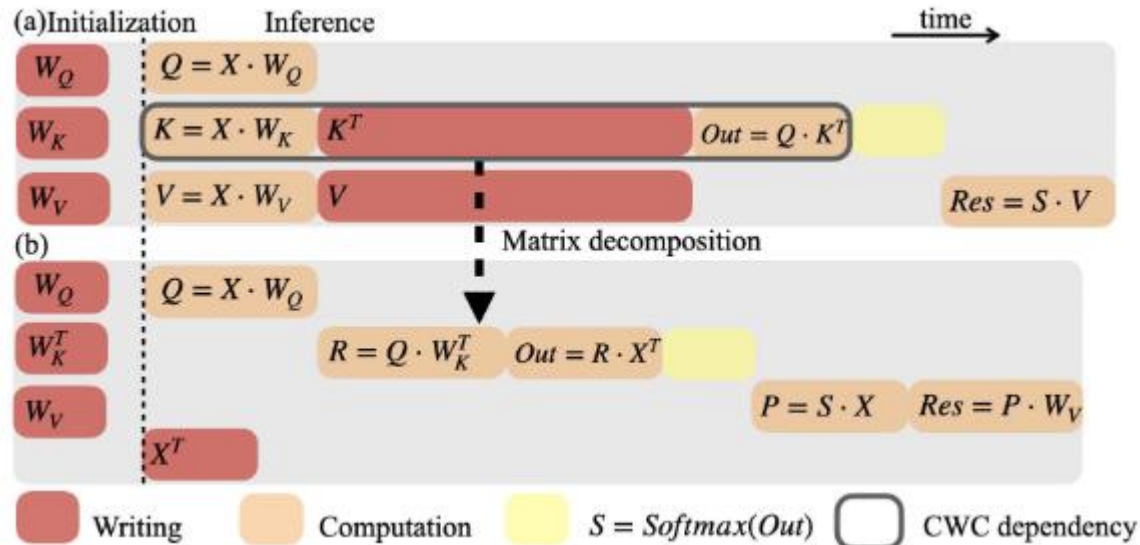
- K^T 是中间计算结果，需要在运行中对 ReRAM 重新编程，造成较高的写延迟
- Softmax 的计算涉及除法和指数运算，如何高效的使用存内计算实现
- 计算资源的利用率低（数据依赖太多）

ReTransformer架构:



Transformer的加速

- 使用矩阵分解解决写延迟问题



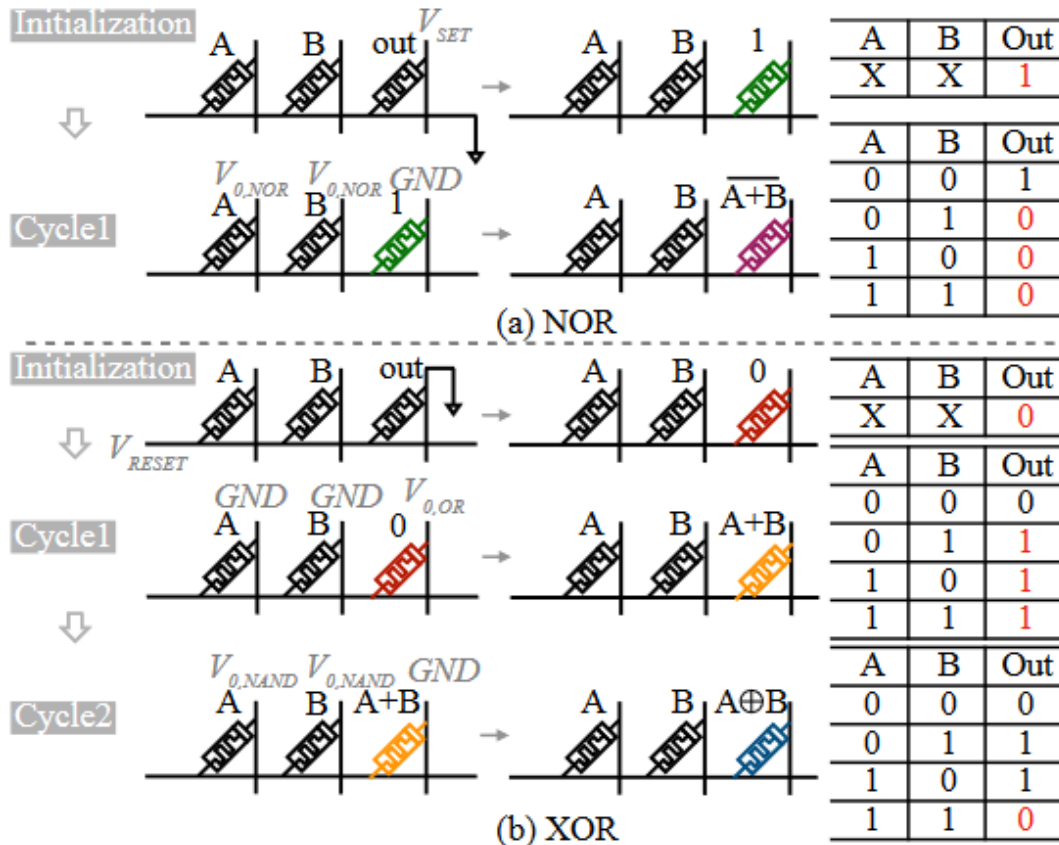
$$Out = Q \cdot K^T = Q \cdot (X \cdot W_K)^T = (Q \cdot W_K^T) \cdot X^T.$$

$$Res = S \cdot V = S \cdot (X \cdot W_V) = (S \cdot X) \cdot W_V = P \cdot W_V$$

Transformer的加速

- 基于ReRAM的混合Softmax

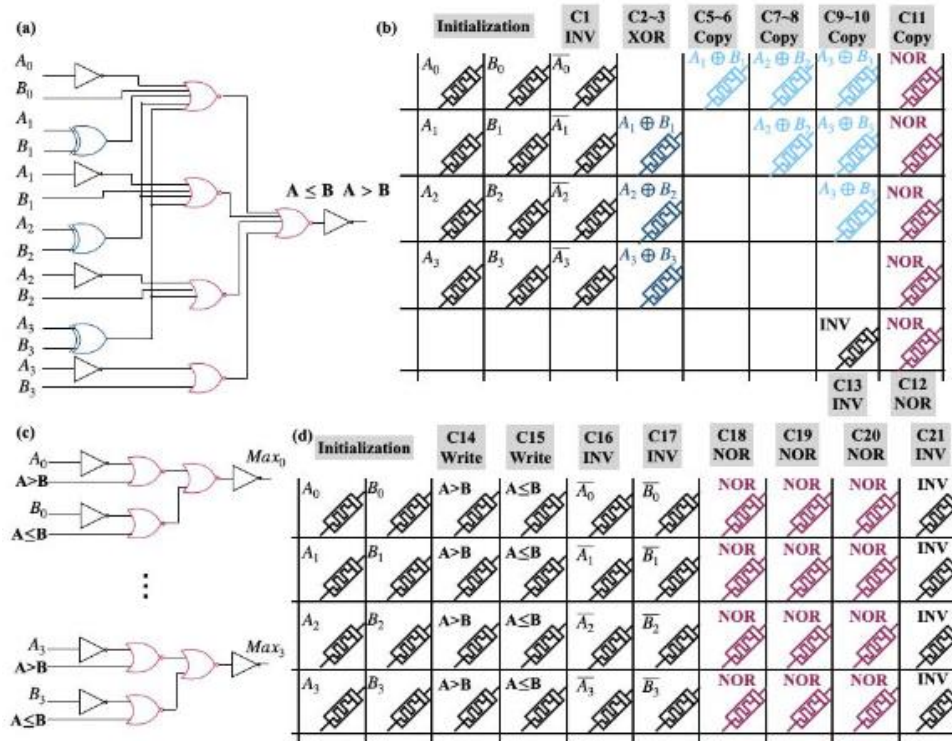
1: 高电导, 0: 低电导, X:未知状态



ReRAM可以实现逻辑运算:
NOR和XOR

Transformer的加速

- 基于ReRAM的混合Softmax

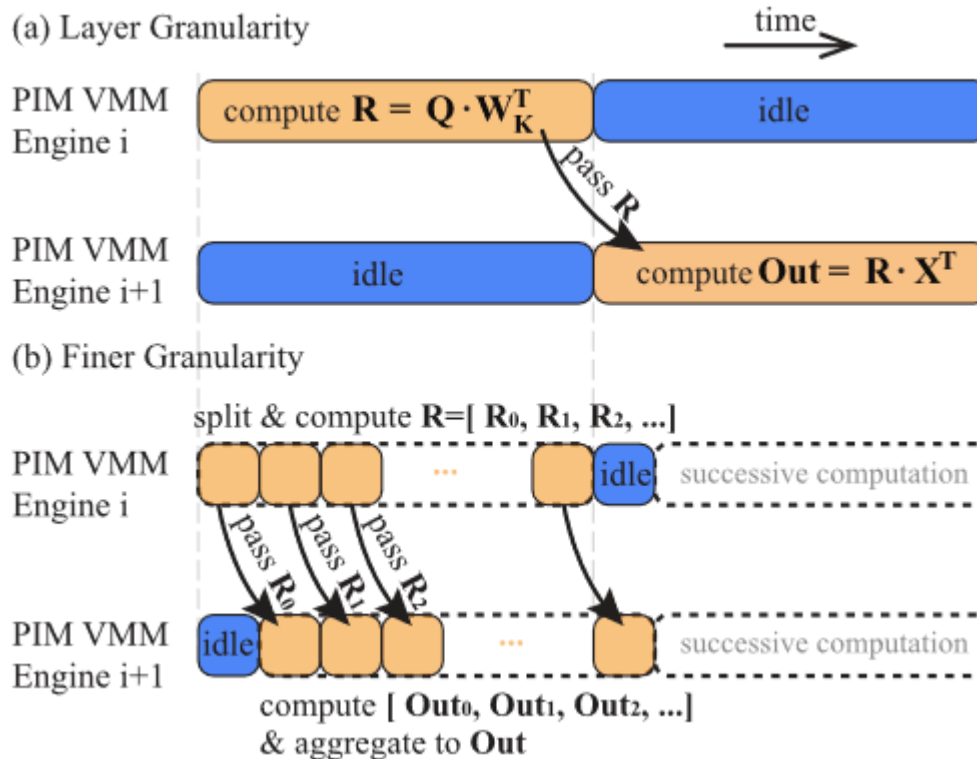


$$\begin{aligned} softmax(x_i) &= \frac{e^{x_i}}{\sum_{j=1}^{d_k} e^{x_j}} = \frac{e^{x_i - x_{max}}}{\sum_{j=1}^{d_k} e^{x_j - x_{max}}} \\ &= \exp[x_i - x_{max} - \log(\sum_{j=1}^{d_k} e^{x_j - x_{max}})]. \end{aligned}$$

ReRAM实现最大值的查找，而后使用查找表得到最终的softmax计算结果

Transformer的加速

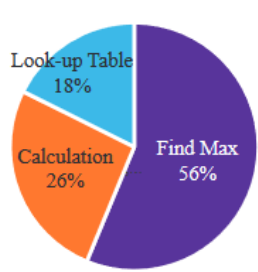
- 更细粒度的调度



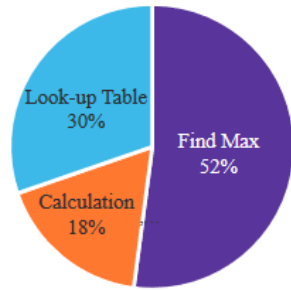
使用分块矩阵

$$\begin{aligned}
 R[i, 0 : d_{model} - 1] &= R_i[:, :] \\
 &= \text{Vec_Q}[0, (i - 1) * d_k : i * d_k - 1] \cdot W_K^T \\
 &= Q[i, 0 : d_q - 1] \cdot W_K^T.
 \end{aligned}$$

Transformer的加速

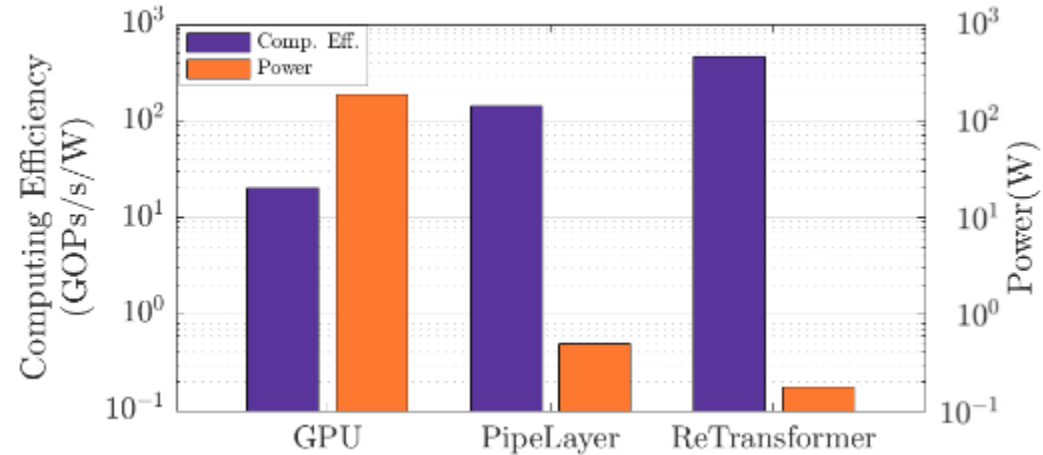


(a) Hybrid Softmax
Total Power: 0.691mW



(b) CMOS-based Softmax
Total Power: 1.023mW

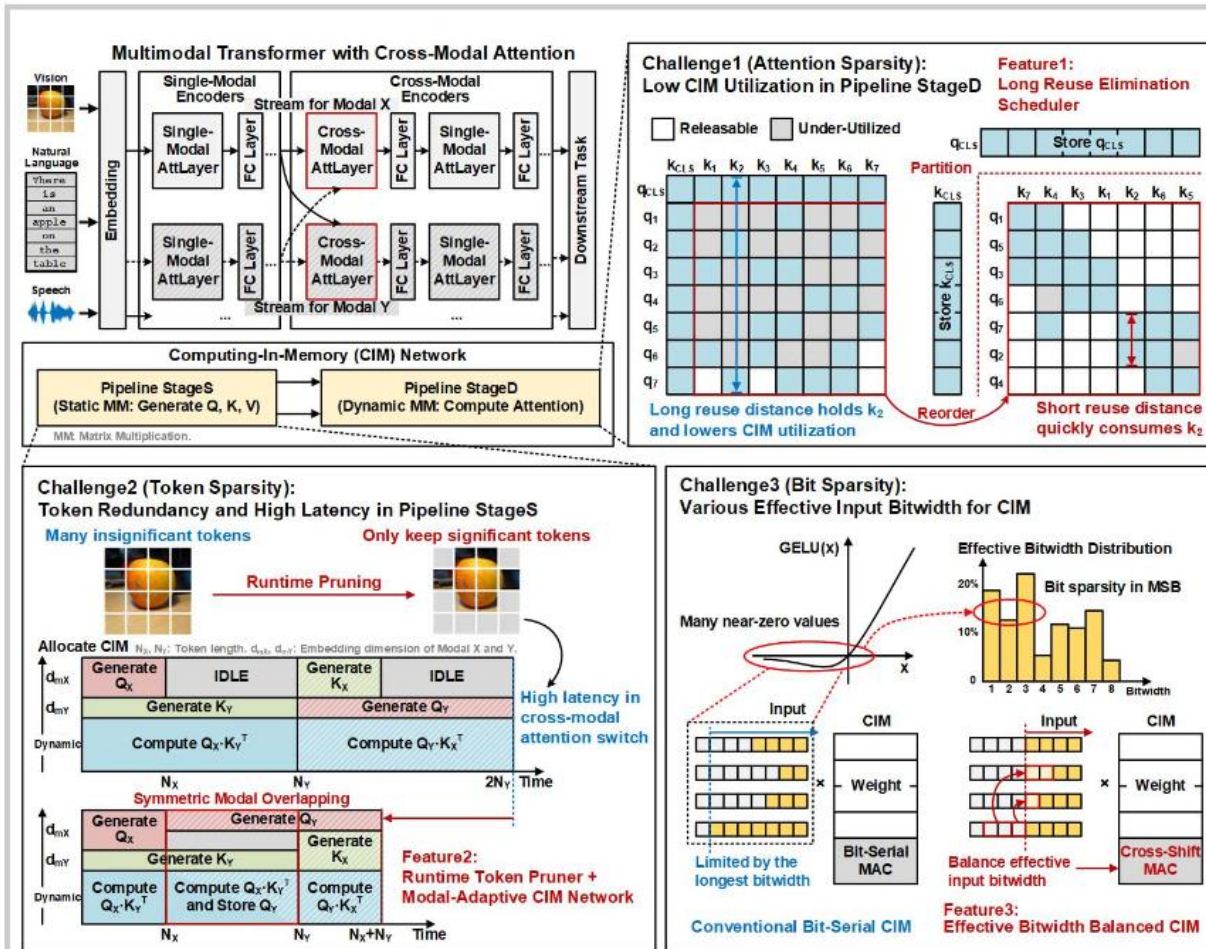
降低功耗



性能: 467.68GOPs/s/W,
23.31xCPU, 3.25xPipeLayer

降低功耗: 1086xCPU,
2.82xPipeLayer

Transformer的加速



Attention Sparsity

Token Sparsity

Bit Sparsity

Technology	TSMC 28nm CMOS	
Die Area	4.46mm×3.22mm	
Supply Voltage	0.6–1.0V	
Frequency	85–275MHz	
Buffer Size	192KB	
CIM Size	128KB	
Data Precision	INT8, INT16	
Work Mode	Pipeline/Parallel	
Power	29.83mW @0.6V, 85MHz	
	153.62mW @1.0V, 275MHz	
Peak Performance ^{*1}	INT8	3.55TOPS
	INT16	0.89TOPS
Area Efficiency ^{*1}	INT8	0.25TOPS/mm ²
	INT16	0.06TOPS/mm ²
Energy Efficiency ^{*2,3}	INT8	48.4-101.1TOPS/W
	INT16	12.1-60.3TOPS/W
Energy/Token ^{*2}	VILBERT-base	2.24μJ/Token
	VILBERT-large	7.72μJ/Token

*1: Measured at the highest performance point, 1.0V, 275MHz. Bit sparsity is exploited since it's an inherent data feature.

*2: Measured at the highest efficiency point, 0.7V, 160MHz.

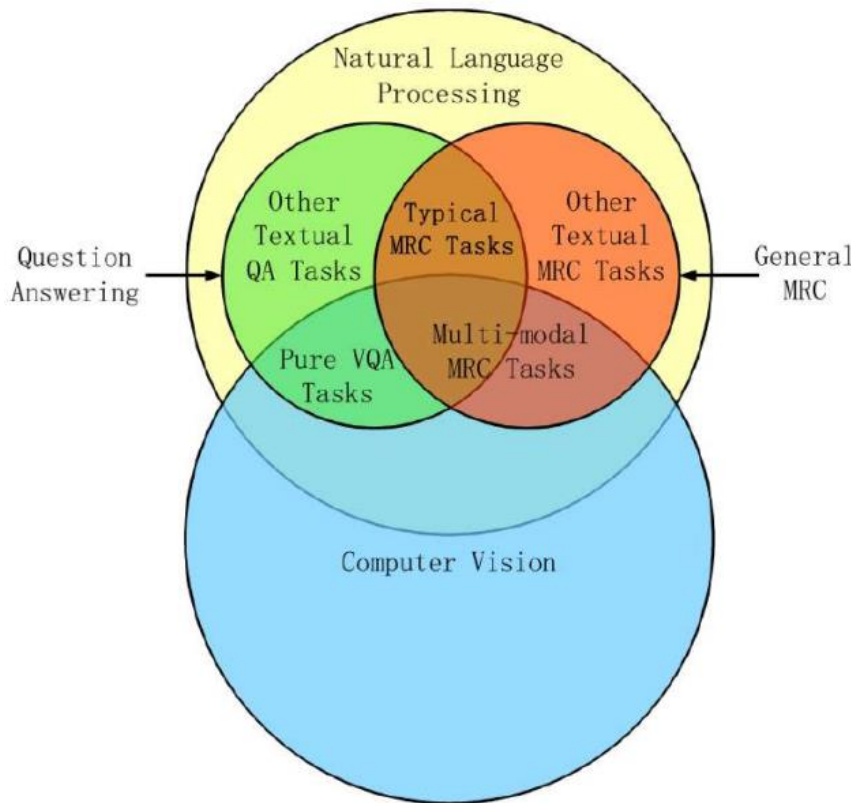
*3: Low point: Only bit sparsity exploitation. High point: Peak efficiency evaluated on the ViLBERT-base model with attention-token-bit hybrid sparsity.

[2] F. Tu et al., "16.1 MultCIM: A 28nm 2.24 uJ/Token Attention-Token-Bit Hybrid Sparse Digital CIM-Based Accelerator for Multimodal Transformers," in *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2023, pp. 248–250. doi: [10.1109/ISSCC42615.2023.10067842](https://doi.org/10.1109/ISSCC42615.2023.10067842).

视觉问答任务 (Visual Question Answer)

什么是VQA:

VQA 介于图像理解 (CV) 和自然语言处理 (NLP) 的交集。VQA 任务的目的是开发出一种系统来回答有关输入图像的特定问题。答案可以采用以下任何形式: 单词, 短语, 二元答案, 多项选择答案或文本填空。




Pure VQA一般没有引入额外的context, 只是单纯的{图, 问句, 回答}
Multi-modal MRC任务引入了额外的知识, 更注重自然语言的理解

MRC: 机器阅读理解 (Machine Reading Comprehension)

视觉问答任务 (Visual Question Answer)

VQA和TQA的区别：数据集信息形式不同

Visual Question Answering		Textual Question Answering	
Visual		Textual	Context: There is a cat and a dog in the image. A cat is on the table. A grey cat. A black dog. A dog is next to the fen. A plant is next to the cat. A green plant. The cat is looking at the plant.
	Question1: What color is the cat? Question2: Which animal in this image is able to climb trees? (Extra Knowledge: Cat is able to climb trees.)		Question1: What color is the cat? Question2: Which animal in this image is able to climb trees? (Extra Knowledge: Cat is able to climb trees.)
↓		↓	
Textual	Answer1: grey Answer2: cat	Textual	Answer1: grey Answer2: cat

Why VQA?

- 目前的多数图像任务并不完全理解图像所包含的信息。比如图像分类，物体检测、动作识别
- VQA的问题可以是任意的，实际上包含了一系列的CV问题：
 - Object recognition - What is in the image?
 - Object detection - Are there any cats in the image?
 - Attribute classification - What color is the cat?
 - Scene classification - Is it sunny?
 - Counting - How many cats are in the image?
- 更复杂的问题：
 - Spatial relationship - What is between the cat and the sofa?
 - Common sense reasoning questions - Why is the girl crying?
 - ...

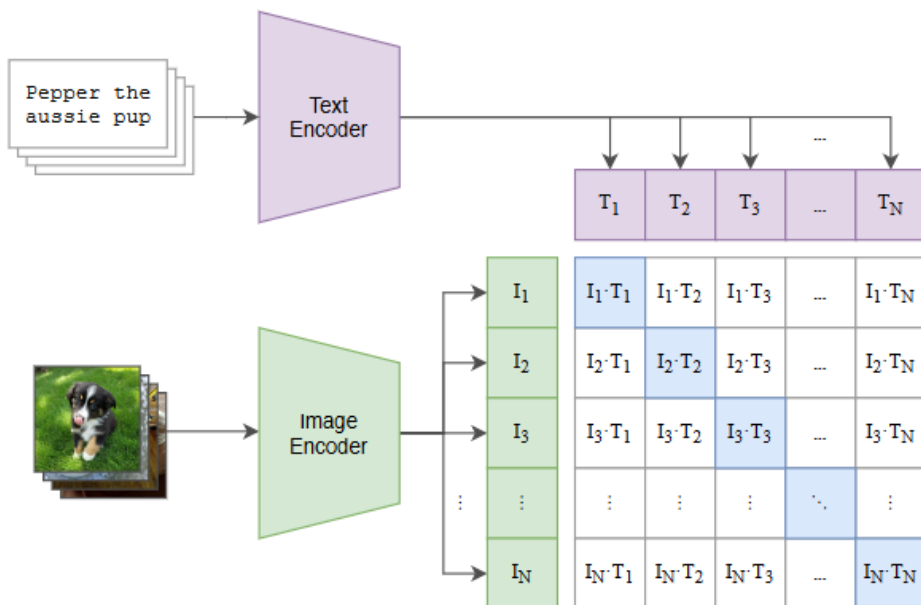
因此， VQA相比CV实现起来也更加困难

视觉问答任务 (Visual Question Answer)

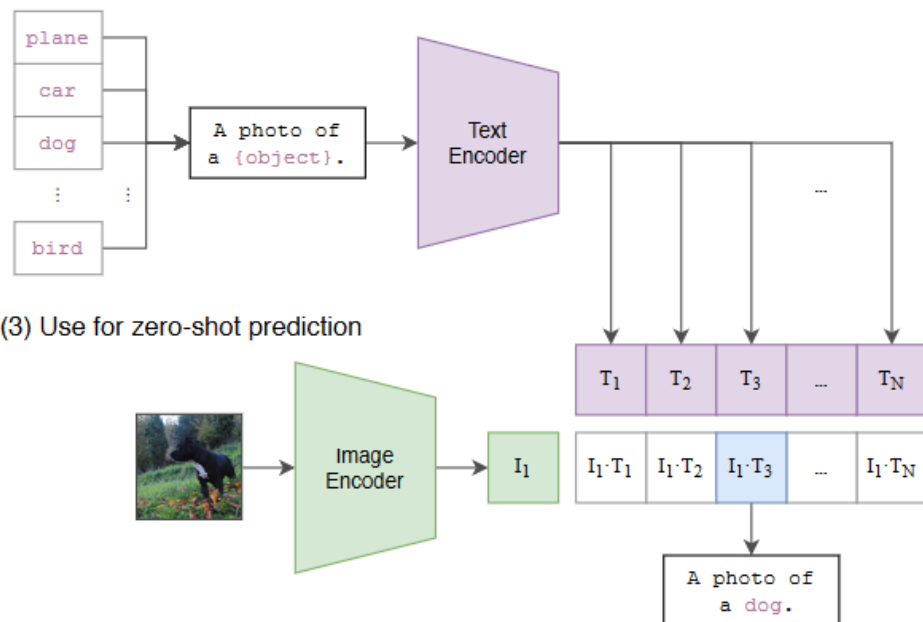
主流模型与方法:

- 从问题中提取特征 (LSTM, GRU, BERT)
- 从图像中提取特征 (VGGNet, ResNet, GoogLeNet)
- 结合这些特征来生成一个答案 (分类和生成)

(1) Contrastive pre-training

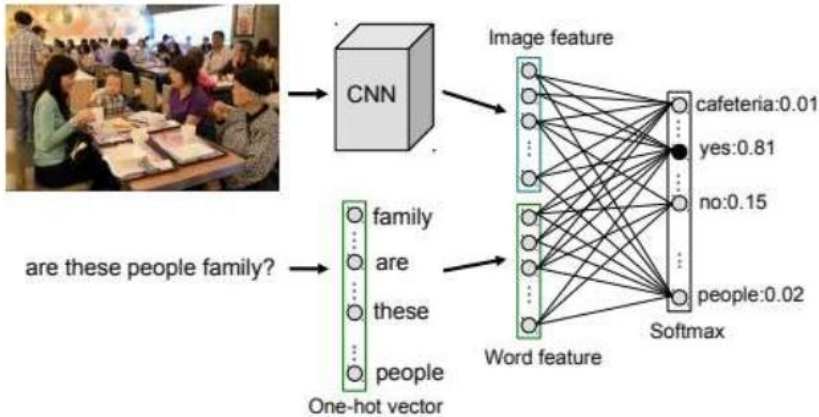


(2) Create dataset classifier from label text



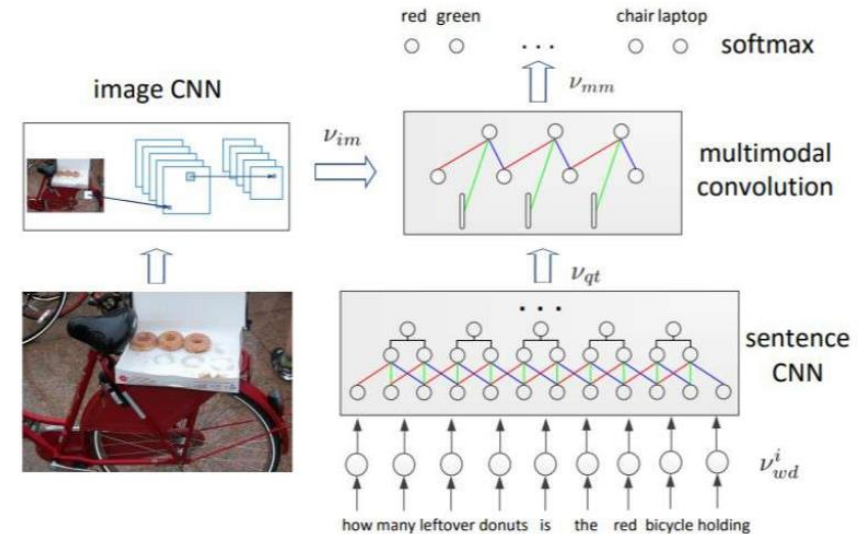
视觉问答任务 (Visual Question Answer)

无注意力机制的深度学习模型



Zhou B, Tian Y, Sukhbaatar S, et al. Simple baseline for visual question answering[J]. arXiv preprint arXiv:1512.02167, 2015.

- CNN
- BoW

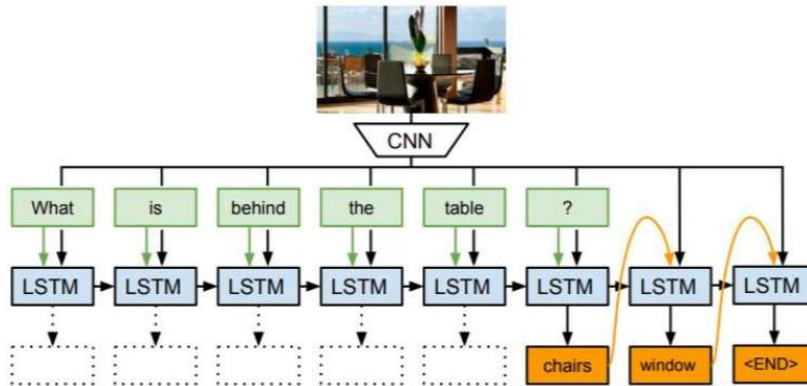


Ma L, Lu Z, Li H. Learning to answer questions from image using convolutional neural network[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2016, 30(1).

- CNN only

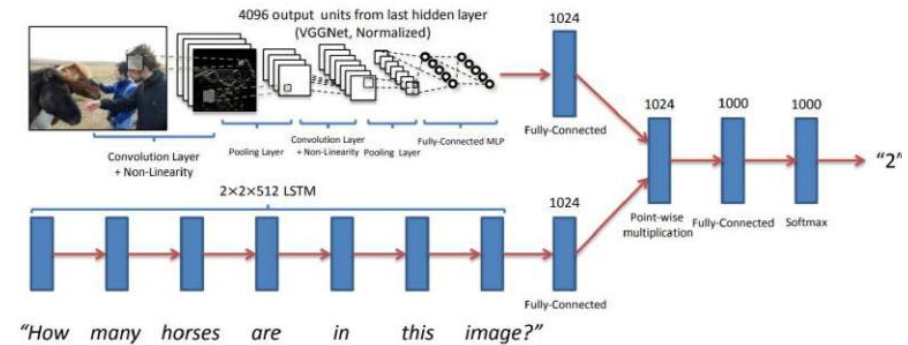
视觉问答任务 (Visual Question Answer)

无注意力机制的深度学习模型



Malinowski M, Rohrbach M, Fritz M. Ask your neurons: A deep learning approach to visual question answering[J]. International Journal of Computer Vision, 2017, 125: 110-135.

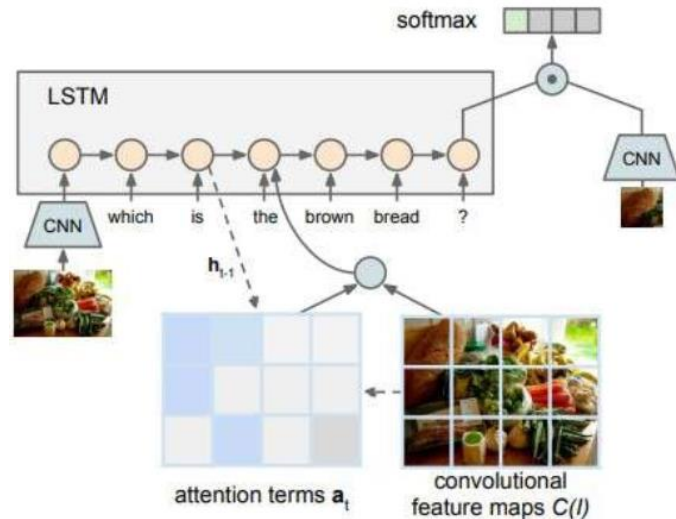
- CNN
- LSTM



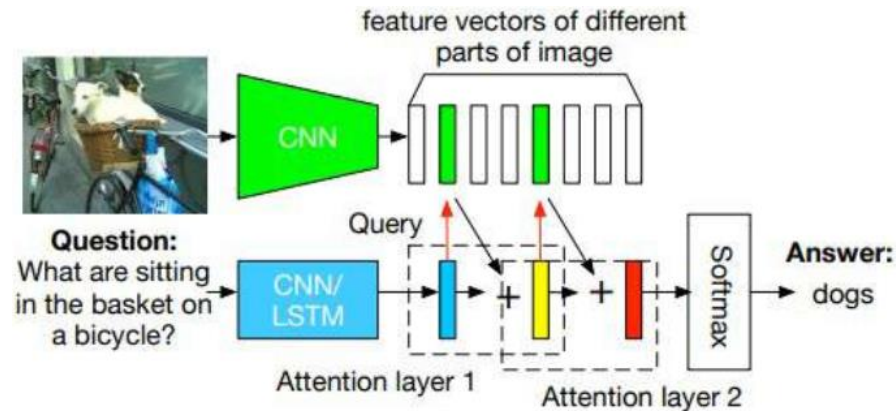
Antol S, Agrawal A, Lu J, et al. Vqa: Visual question answering[C]//Proceedings of the IEEE international conference on computer vision. 2015: 2425-2433.

- CNN
- LSTM

基于注意力机制的深度学习模型



Q: Which is the brown bread?



Yang Z, He X, Gao J, et al. Stacked attention networks for image question answering[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 21-29.

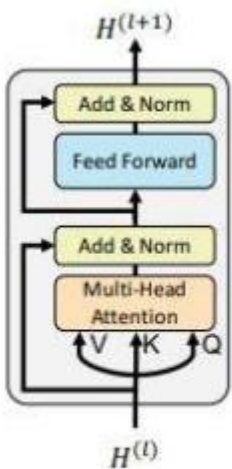
- CNN
- LSTM

Zhu Y, Groth O, Bernstein M, et al. Visual7w: Grounded question answering in images[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 4995-5004.

- CNN
- LSTM

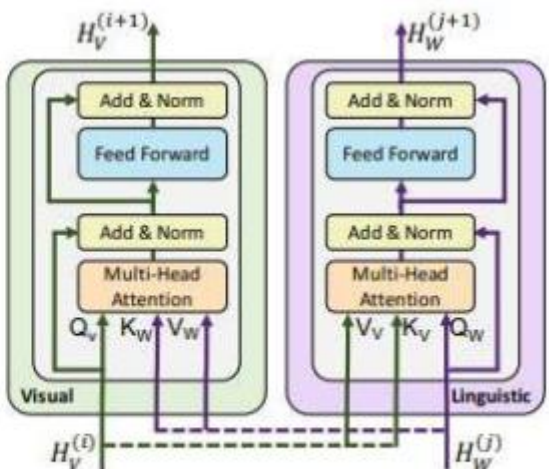
基于Transformer的深度学习模型

单流模型和双流模型:



(a) Standard encoder transformer block

QKV都是同一个输入

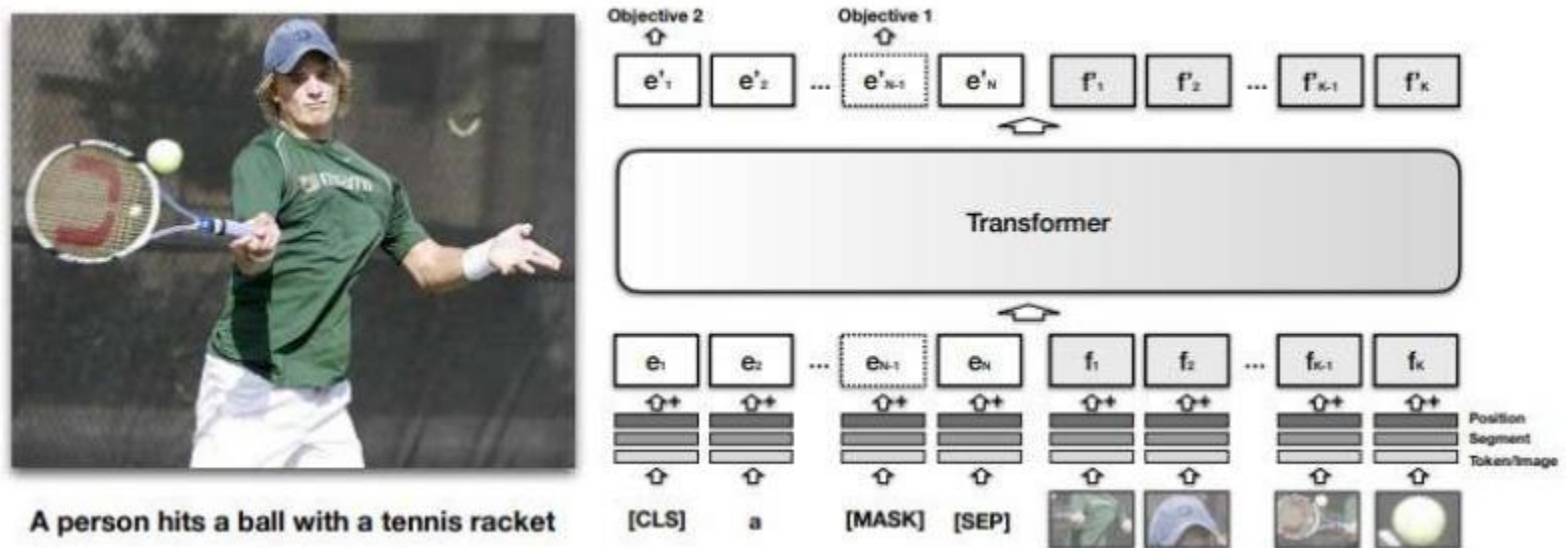


(b) Our co-attention transformer layer

QKV是不同输入

视觉问答任务 (Visual Question Answer)

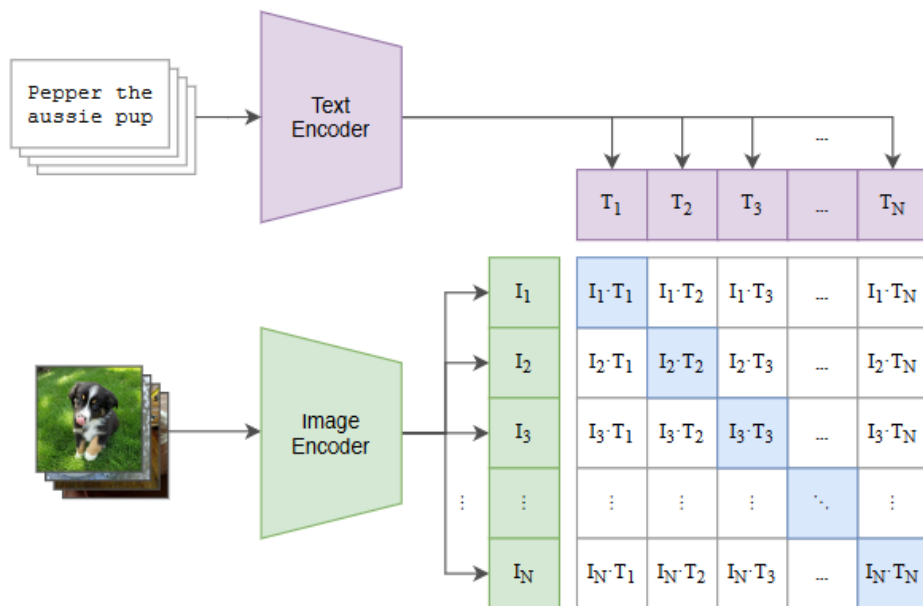
基于Transformer的深度学习模型



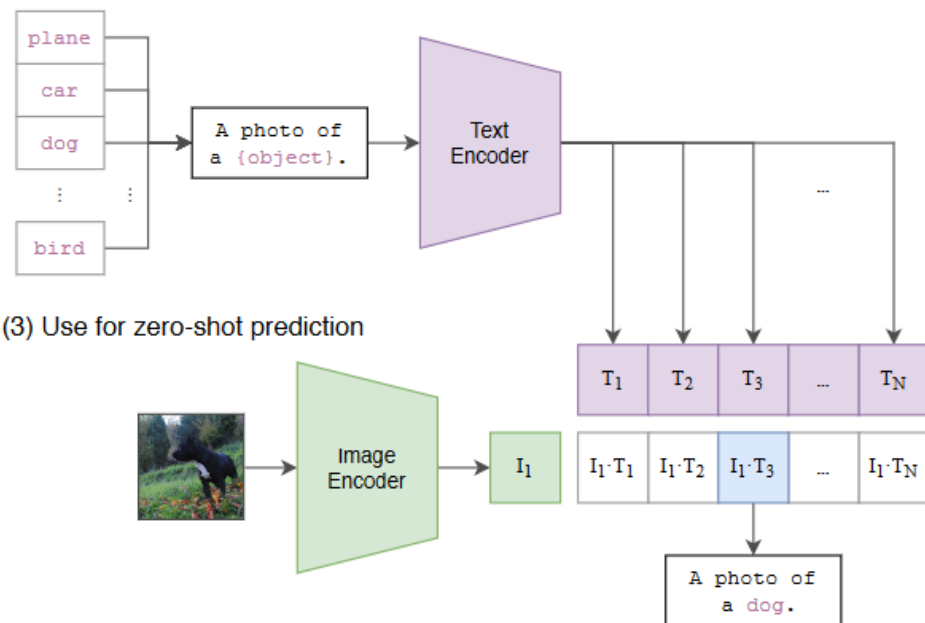
Li L H, Yatskar M, Yin D, et al. Visualbert: A simple and performant baseline for vision and language[J]. arXiv preprint arXiv:1908.03557, 2019.

多模态CLIP

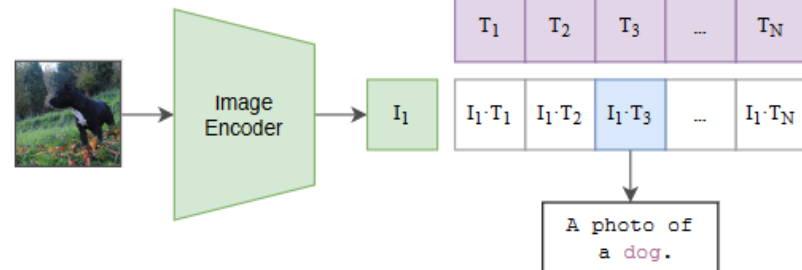
(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



对比学习：
 拉近正样本之间的向量
 拉远负样本之间的向量
 距离度量：欧氏距离、余弦距离或汉明距离

准备文本数据，实现图像分类
 准备图像数据，可以实现文本搜图

两大编码器结构：CNN和Transformer，基本上实现所有信息的编码工作

模态	特征提取网络	详情
图像/视频	ViT	参考OmniMAE的做法，针对图像和视频都采用ViT结构。
深度	ViT	看作是单通道的图像
热力	ViT	看作是单通道的图像
文本	采用CLIP中的文本特征提取模块	
音频	ViT	按照 AST对音频进行编码，并使用 128 mel-spectrogram bins 将以 16kHz 采样的 2 秒音频转换为频谱图。由于频谱图也是像图像一样的二维信号，因此我们使用补丁大小为 16 且步幅为 10 的 ViT
IMU	Transformer	提取由 X、Y 和 Z 轴上的加速度计和陀螺仪测量值组成的 IMU 信号。我们使用 5 秒的剪辑产生 2K 时间步长的 IMU 读数，这些读数使用内核大小为 8 的一维卷积进行投影。然后用 Transformer编码。

[1] R. Girdhar *et al.*, “ImageBind: One Embedding Space To Bind Them All.” arXiv, May 31, 2023. doi: [10.48550/arXiv.2305.05665](https://doi.org/10.48550/arXiv.2305.05665).

1) Cross-Modal Retrieval

Audio	Images & Videos	Depth	Text
 Crackle of a Fire			"A fire crackles while a pan of food is frying on the fire." "Fire is crackling then wind starts blowing." "Firewood crackles then music..."
 Baby Cooing			"A baby is crying while a toddler is laughing." "A baby is laughing while an adult is laughing." "A baby laughs and something..."

2) Embedding-Space Arithmetic



Diagram illustrating Embedding-Space Arithmetic: An image of a white egret is combined with the audio of waves (represented by a speaker icon and the word "Waves") to generate a new image of a white egret standing in water.

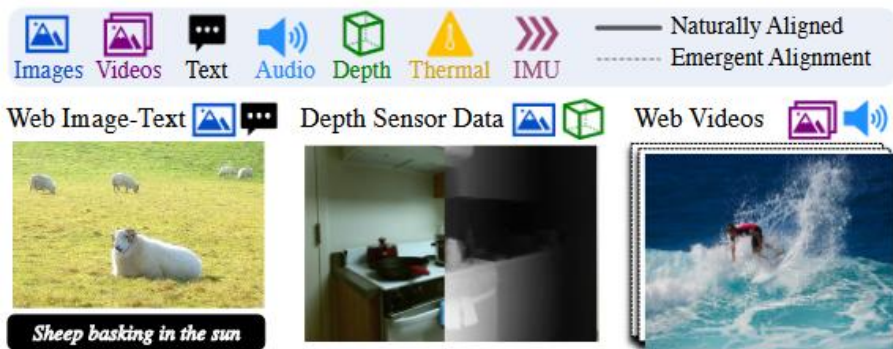
3) Audio to Image Generation



Diagram illustrating Audio to Image Generation: Audio inputs (represented by speaker icons) are used to generate corresponding images. Examples include: "Dog" audio generates a dog image; "Engine" audio generates a bus image; "Fire" audio generates a fire image; "Rain" audio generates a rain image.

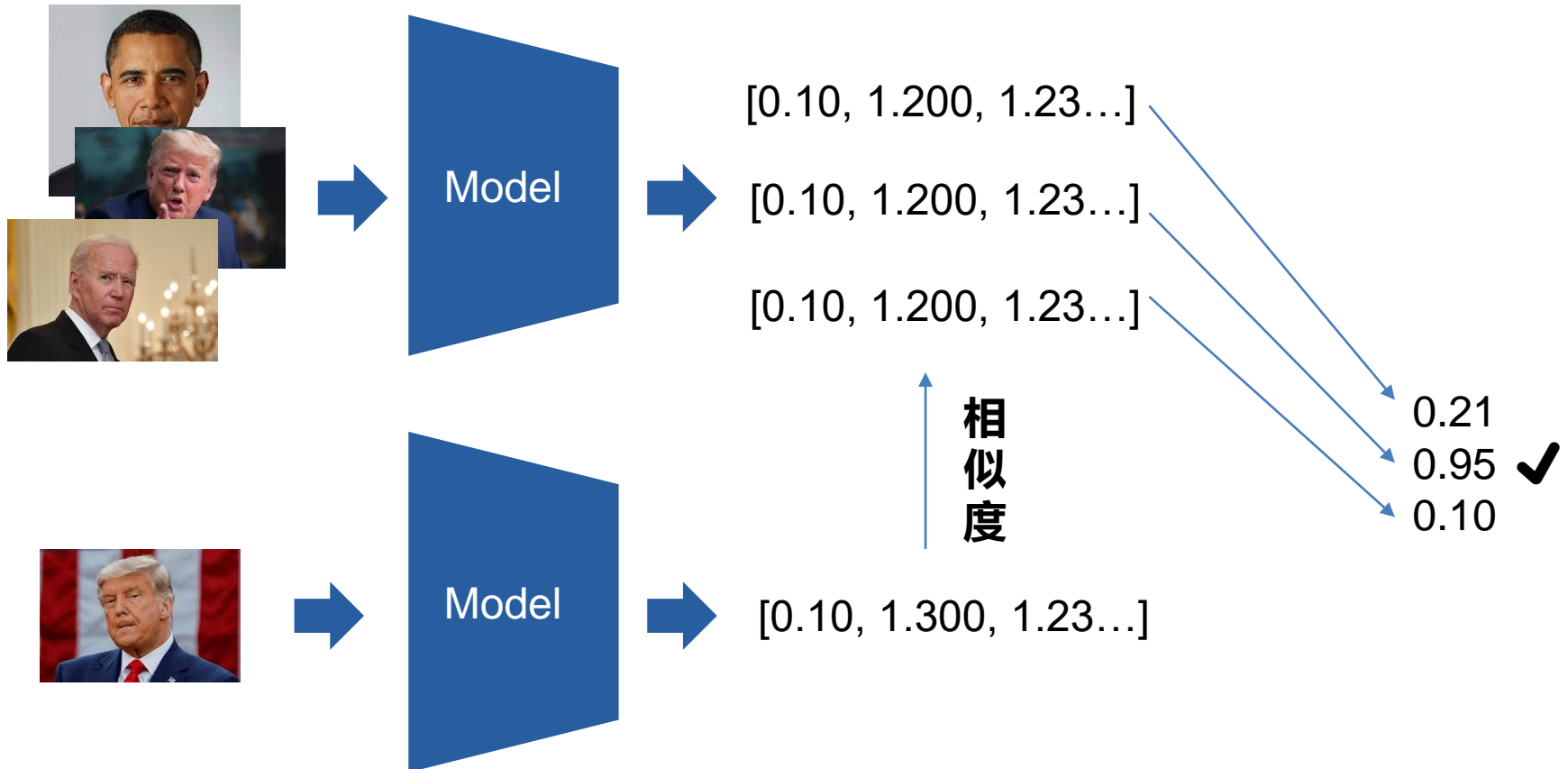
- 跨模态检索
- 模态组合运算
- 跨模态生成

多模态ImageBind

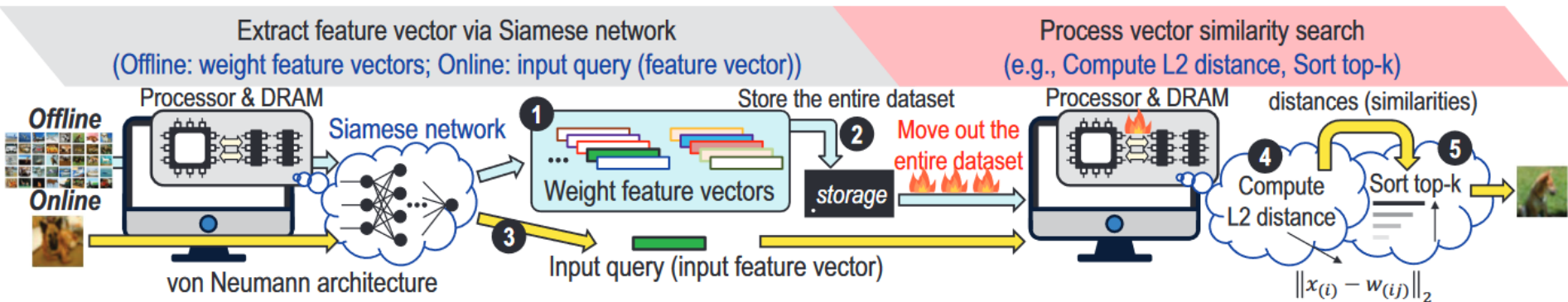


将不同模态的信息通过对比学习投影到图像嵌入的空间相互之间就实现了对齐

人脸识别



多模态在边缘端加速

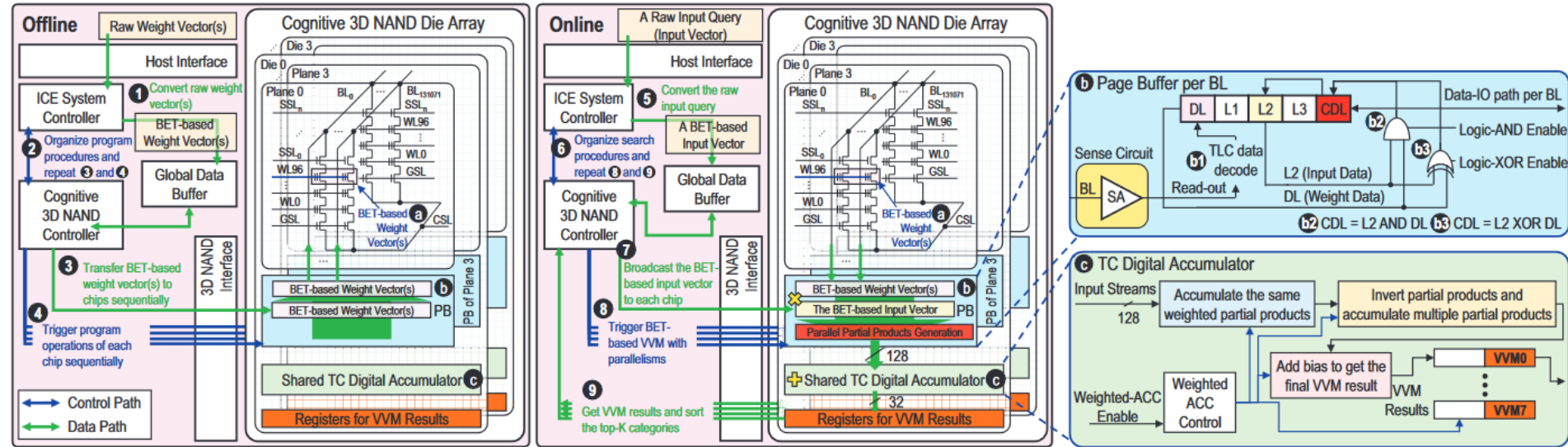


欧氏距离：

$$\|x - w_i\|_2^2 = \|x\|^2 + \|w_i\|^2 - 2(x \cdot w_i)$$

x 和 w_i 归一化后为1，因此主要计算为 $x \cdot w_i$ ，即VVM，向量乘向量

多模态在边缘端加速





Thanks!