

教育部-华为智能基座课程

《人工智能基础与实践》

第11章：先进神经网络架构

授课教师：丛润民

山东大学

控制科学与工程学院

章节目录

CONTENTS

01 | 注意力机制

02 | Transformer架构及其应用

03 | Mamba架构及其应用



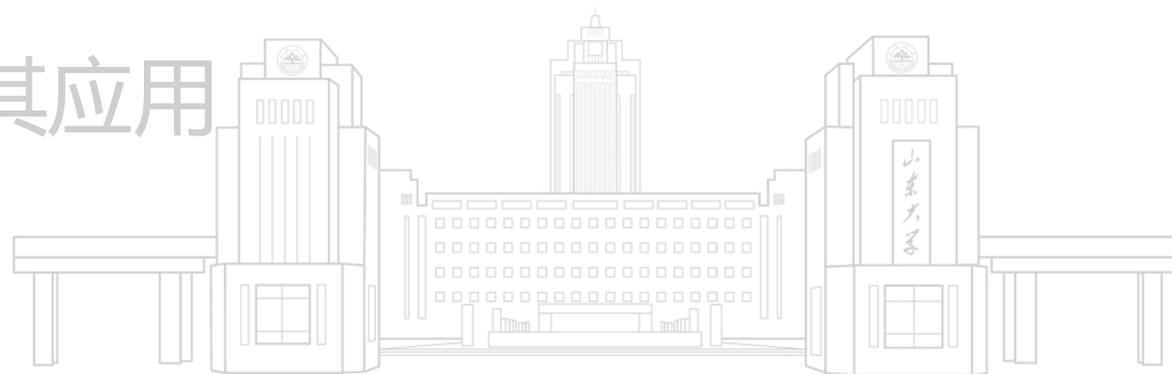
章节目录

CONTENTS

01 | 注意力机制

02 | Transformer架构及其应用

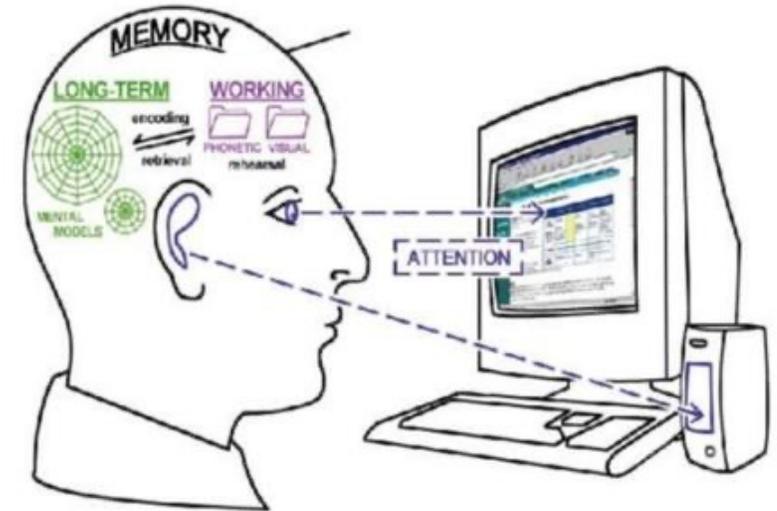
03 | Mamba架构及其应用



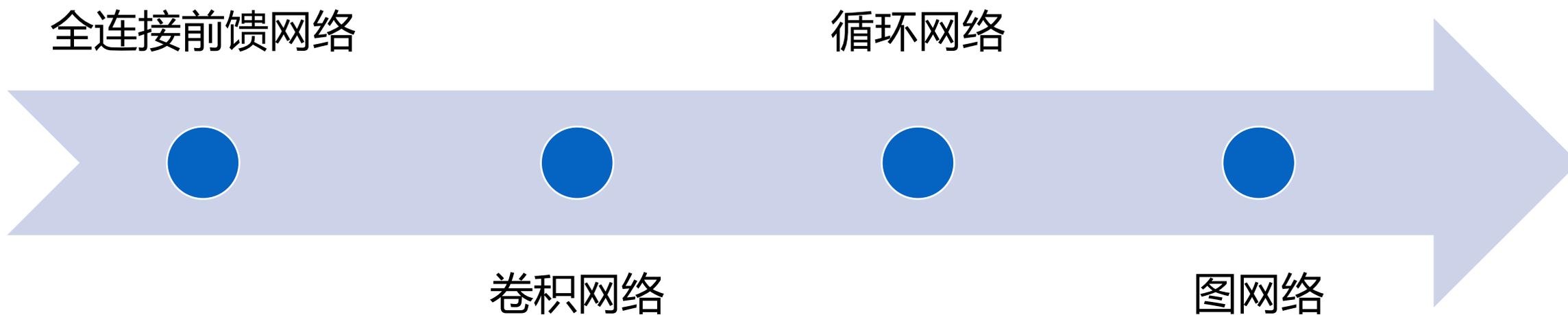
大脑中的注意力

- 人脑每个时刻接收的外界输入信息非常多，包括来源于视觉、听觉、触觉各种各样的信息
- 单就视觉来说，眼睛每秒钟都会发送千万比特的信息给视觉神经系统
- 人脑在有限的资源下，并不能同时处理这些**过载**的输入信息
- 人脑通过**注意力**来解决信息超载问题（记忆、阅读、思考）
- 我们可以借鉴人脑解决信息过载的**注意力机制**来提高神经网络处理信息的能力

注意力示例



网络能力提升



增加网络能力的另一种思路：**注意力机制**

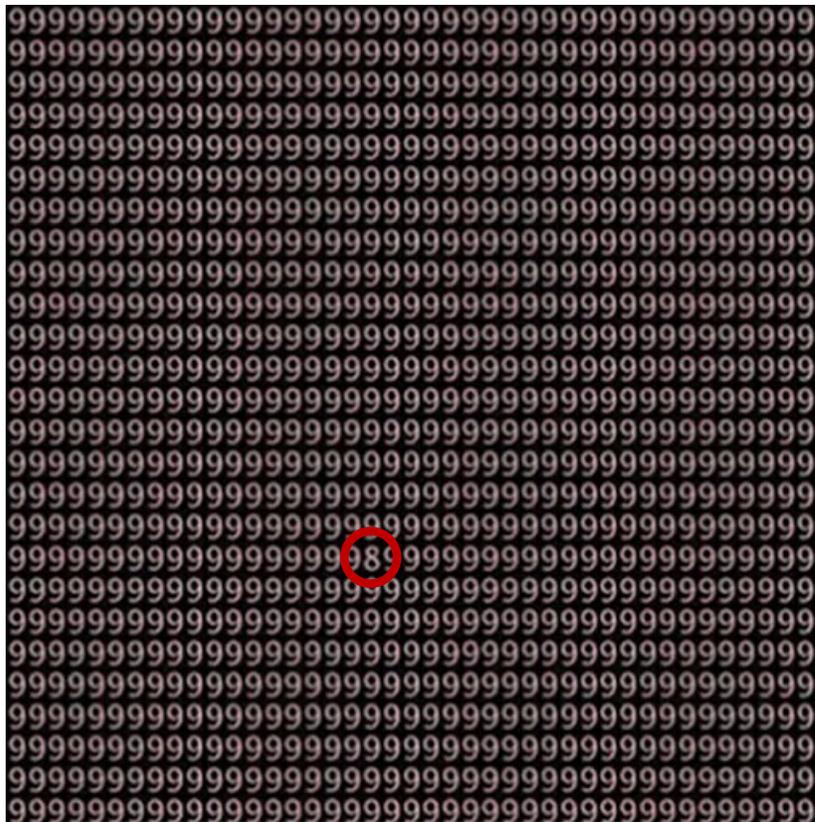
注意力机制背景

- 前馈网络与循环网络受到**优化算法及计算能力**的限制
- 处理大量输入信息或复杂计算流程时，计算机的计算能力成为瓶颈
- 减少计算复杂度：**局部连接、权重共享、汇聚操作**简化神经网络结构
- 虽然可有效缓解模型的复杂度和表达能力之间的矛盾，但依然希望在不“过度”增加模型复杂度（主要是模型参数）的情况下进一步**提高模型的表达能力**；
- 用循环神经网络解决机器阅读理解问题

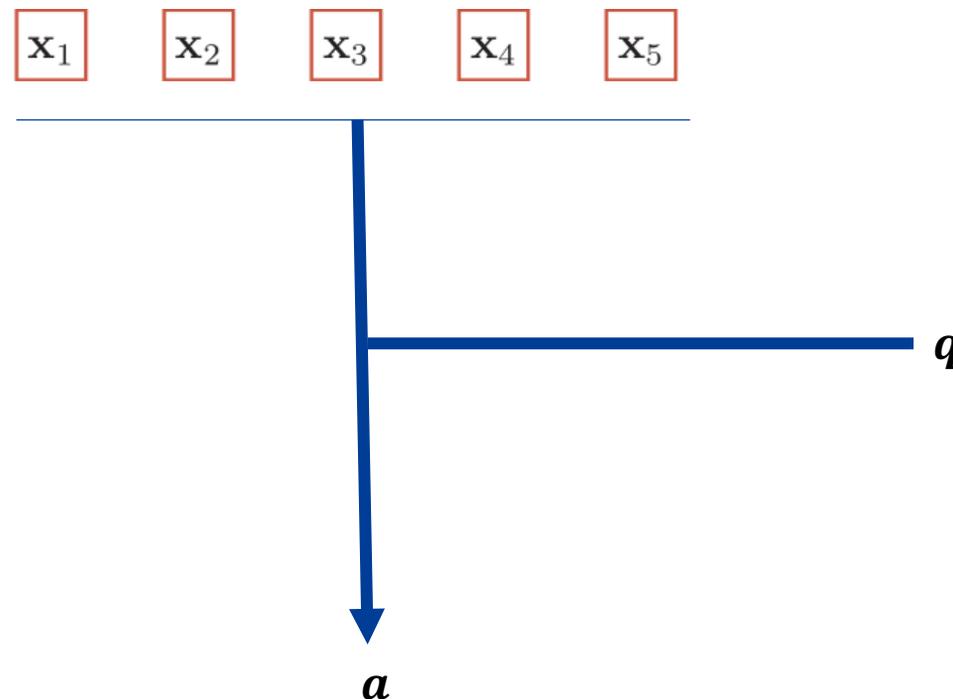
定义与分类

- 人脑可以有意或无意地从大量输入信息中选择小部分的有用信息来重点处理，并忽略其他信息，这种能力就叫作**注意力 (Attention)**
- 注意力可以作用在外部的刺激（听觉、视觉、味觉等），也可以作用在内部的意识（思考、回忆等）
- 如何实现（注意力的分类）？
 - 自上而下：**聚焦式注意力**或**选择性注意力** (Focus/Selective Attention)
 - 自下而上：**基于显著性的注意力** (Saliency-Based Attention)

注意力机制的计算



8秒内找到数字8



注意力机制的计算

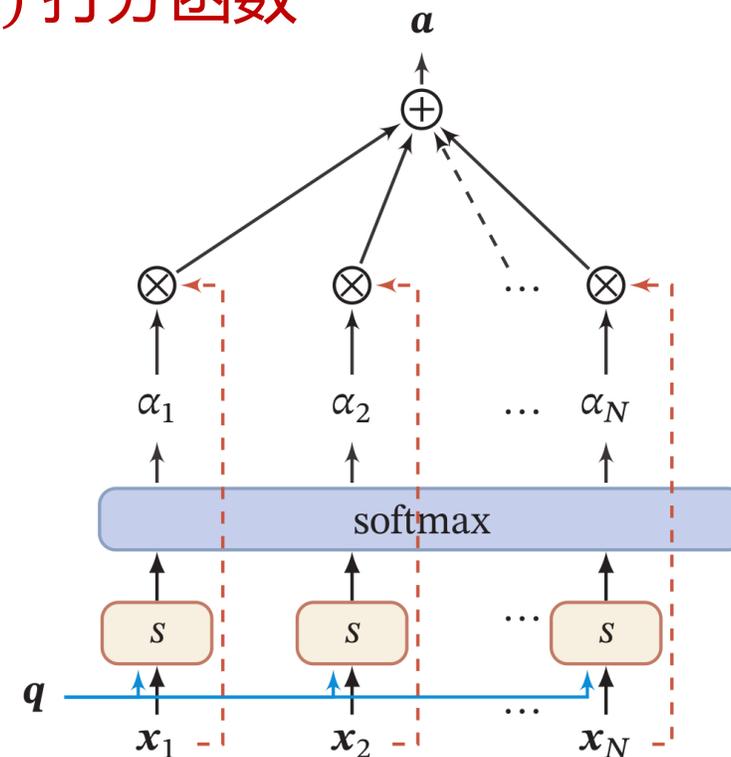
► 计算可分为两步：

- 计算**注意力分布** α
- 根据 α 来计算输入信息的加权平均

$$\begin{aligned}\alpha_n &= p(z = n | \mathbf{X}, \mathbf{q}) \\ &= \text{softmax}(s(\mathbf{x}_n, \mathbf{q})) \\ &= \frac{\exp(s(\mathbf{x}_n, \mathbf{q}))}{\sum_{j=1}^N \exp(s(\mathbf{x}_j, \mathbf{q}))}\end{aligned}$$

$$\begin{aligned}\text{att}(\mathbf{X}, \mathbf{q}) &= \sum_{n=1}^N \alpha_n \mathbf{x}_n, \\ &= \mathbb{E}_{z \sim p(z | \mathbf{X}, \mathbf{q})}[\mathbf{x}_z]\end{aligned}$$

$s(\mathbf{x}_n, \mathbf{q})$ 打分函数



注意力打分函数

加性模型

$$s(\mathbf{x}, \mathbf{q}) = \mathbf{v}^\top \tanh(\mathbf{W}\mathbf{x} + \mathbf{U}\mathbf{q}),$$

点积模型

$$s(\mathbf{x}, \mathbf{q}) = \mathbf{x}^\top \mathbf{q},$$

缩放点积模型

$$s(\mathbf{x}, \mathbf{q}) = \frac{\mathbf{x}^\top \mathbf{q}}{\sqrt{D}},$$

双线性模型

$$s(\mathbf{x}, \mathbf{q}) = \mathbf{x}^\top \mathbf{W} \mathbf{q},$$

注意力机制的变体

➤ 软注意力 (Soft Attention)

$$\begin{aligned}\text{att}(\mathbf{X}, \mathbf{q}) &= \sum_{n=1}^N \alpha_n \mathbf{x}_n, \\ &= \mathbb{E}_{z \sim p(z|\mathbf{X}, \mathbf{q})} [\mathbf{x}_z]\end{aligned}$$

➤ 硬性注意力 (Hard Attention)

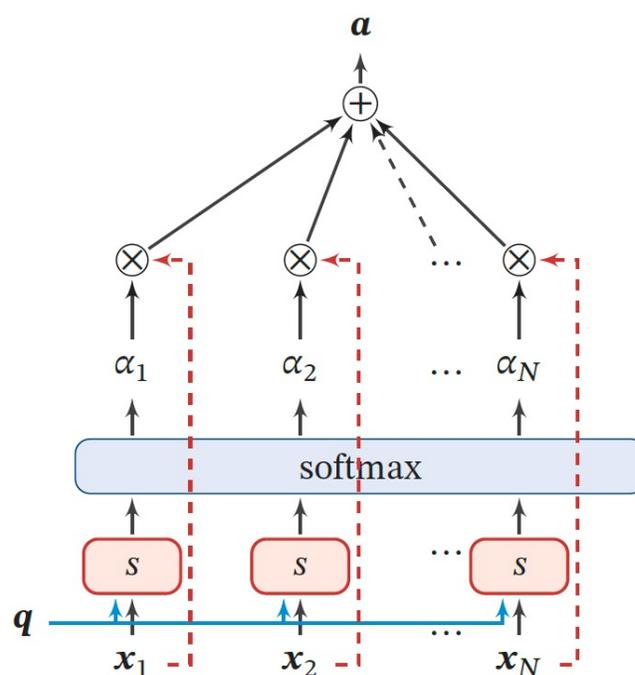
$$\text{att}(\mathbf{X}, \mathbf{q}) = \mathbf{x}_{\hat{n}}, \quad \hat{n} = \arg \max_{n=1}^N \alpha_n.$$

注意力机制的变体

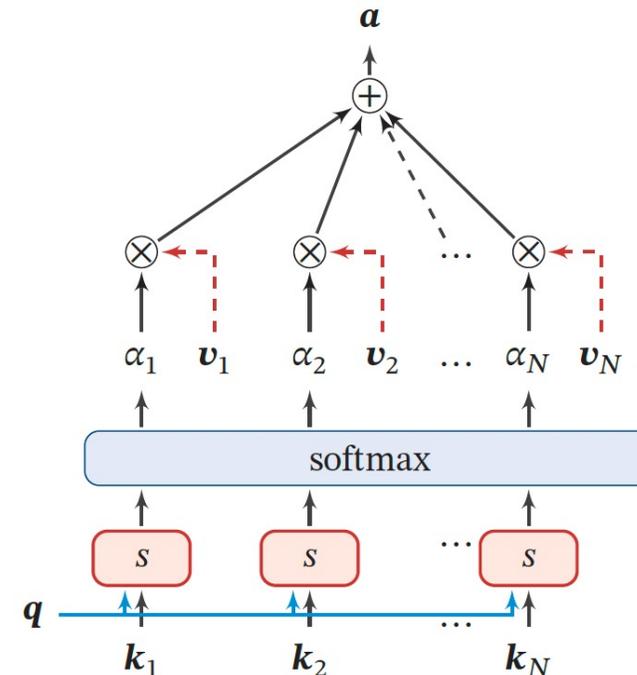
键值对注意力 (Key-Value Pair Attention)

$$\begin{aligned} \text{att}((K, V), q) &= \sum_{n=1}^N \alpha_n v_n, \\ &= \sum_{n=1}^N \frac{\exp(s(k_n, q))}{\sum_j \exp(s(k_j, q))} v_n, \end{aligned}$$

用 $(K, V) = [(k_1, v_1), \dots, (k_N, v_N)]$
表示 N 个输入信息



(a) 普通模式



(b) 键值对模式

注意力机制的变体

➤ 多头注意力 (Multi-Head Attention)

- 利用多个查询 $Q = [q_1, \dots, q_M]$ 来并行地从输入信息中选取多组信息,每个注意力关注输入信息的不同部分

$$\text{att}((K, V), Q) = \text{att}((K, V), q_1) \oplus \dots \oplus \text{att}((K, V), q_M)$$

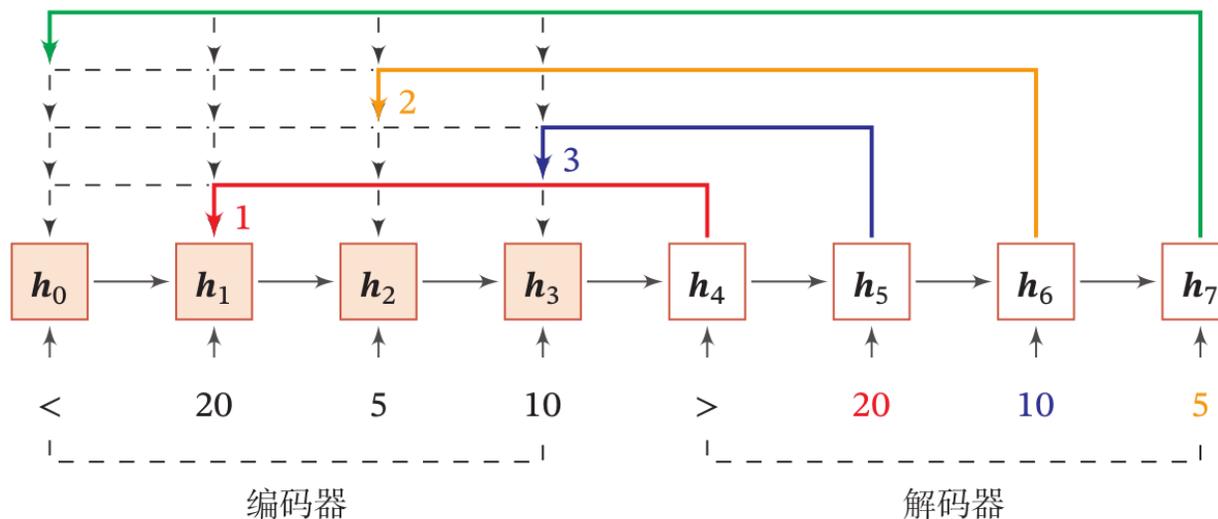
➤ 结构化注意力 (Structural Attention)

- 输入信息本身具有层次 (Hierarchical) 结构

注意力机制的变体

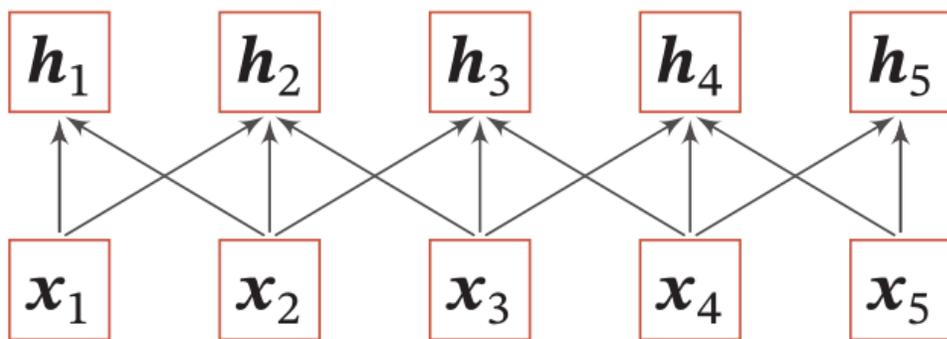
➤ 指针网络 (Pointer Network)

- 只利用注意力机制中的第一步，将注意力分布作为一个软性的指针 (pointer) 来指出相关信息的位置

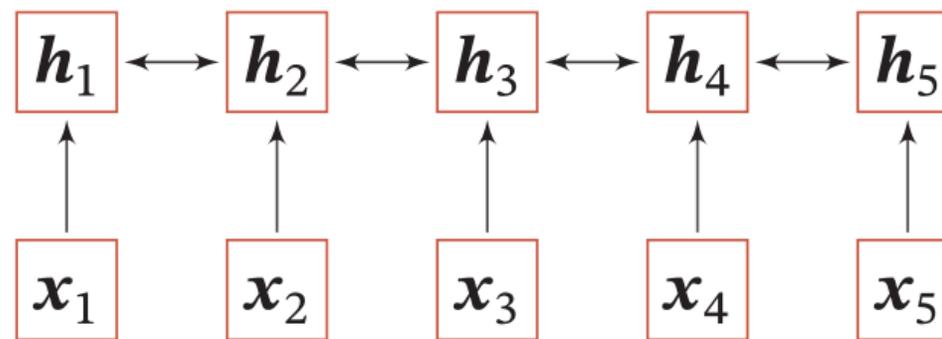


自注意力模型

- 虽然循环网络理论上可以建立长距离依赖关系，但是由于信息传递的容量以及梯度消失问题，实际上也只能建立短距离依赖关系



(a) 卷积网络

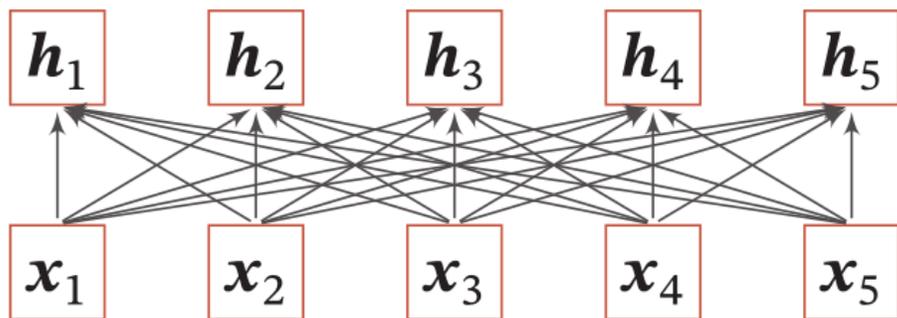


(b) 双向循环网络

自注意力模型

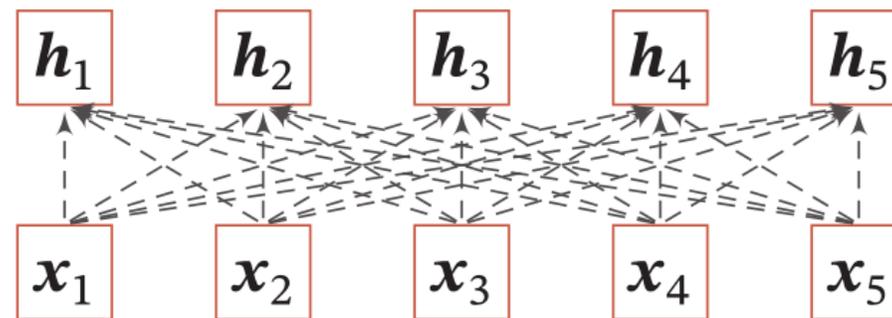
➤ 如何建立非局部 (Non-Local) 的依赖关系

- 全连接?



(a) 全连接模型

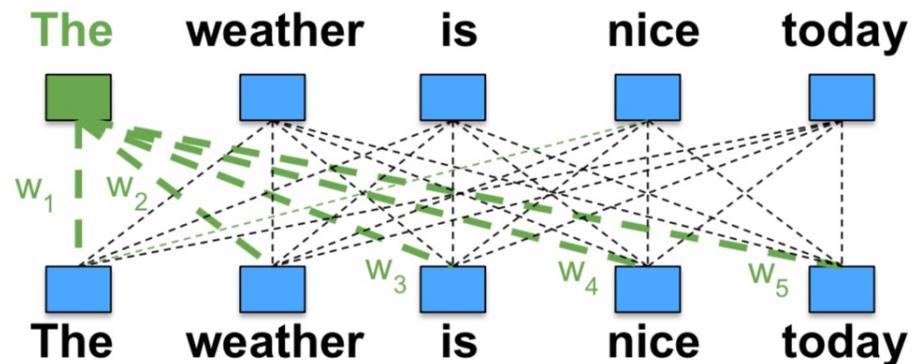
无法处理变长问题



(b) 自注意力模型

连接权重 α_{ij} 由注意力机制动态生成

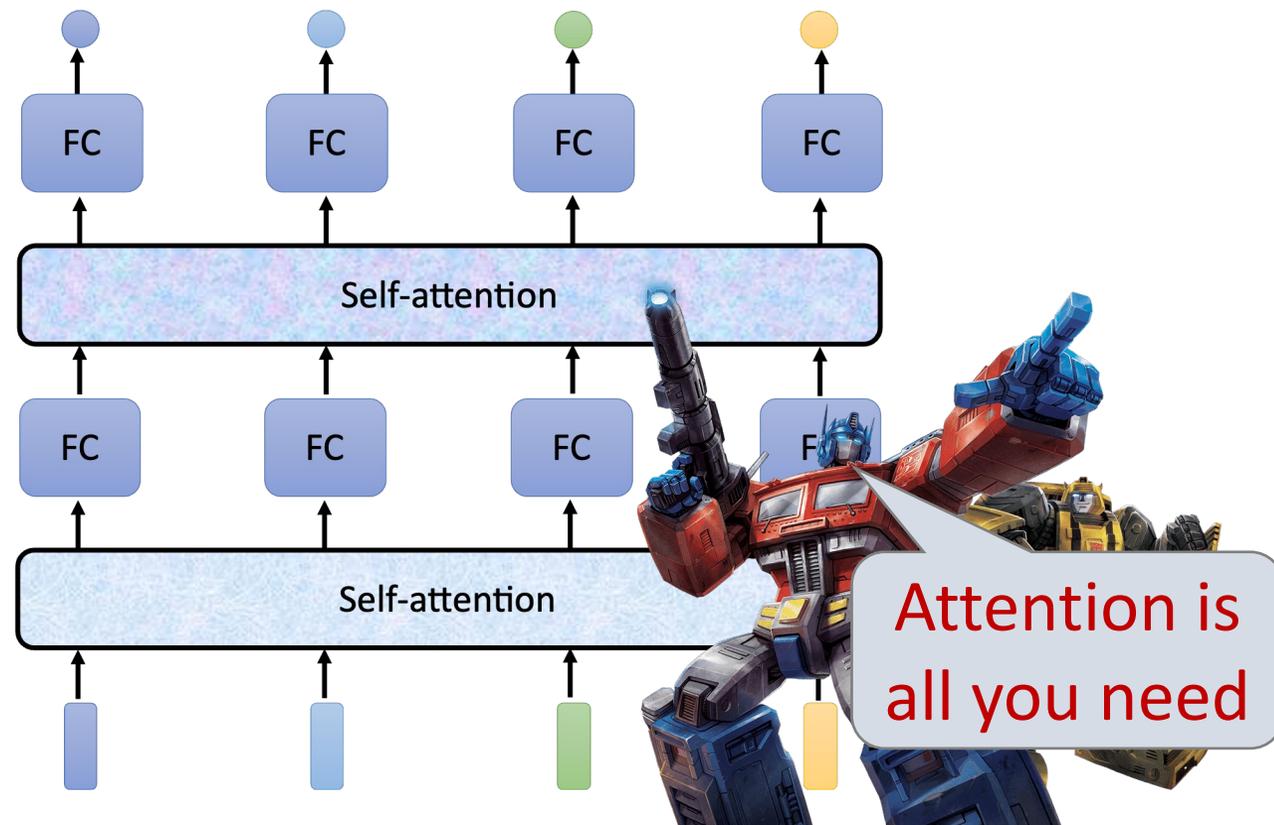
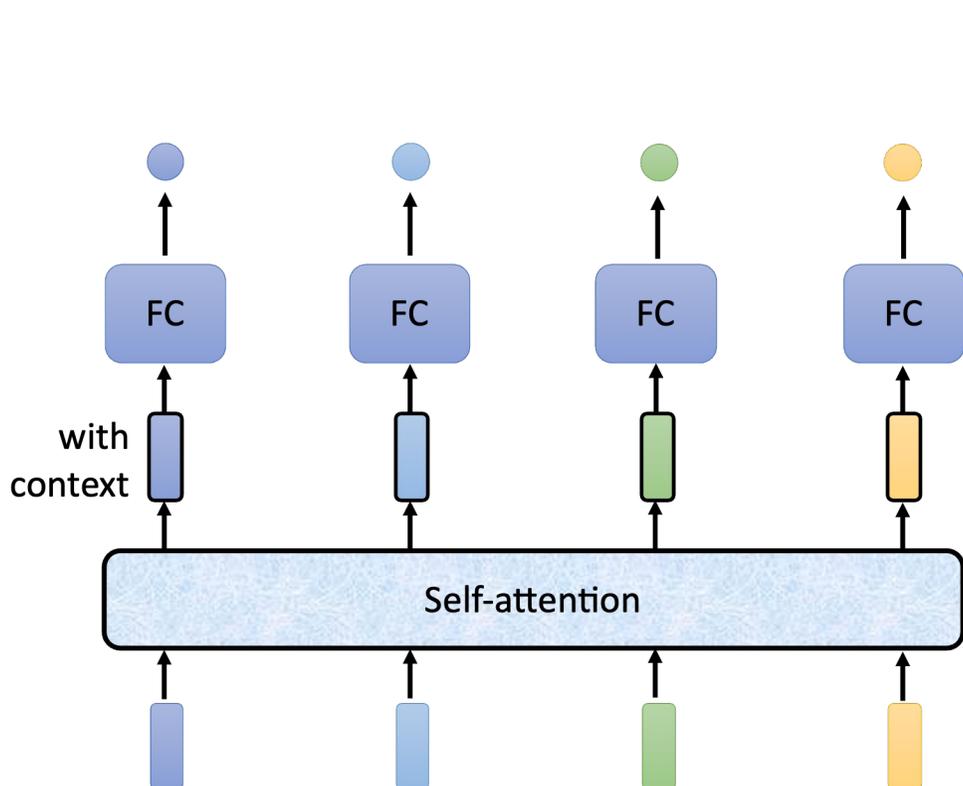
自注意力示例



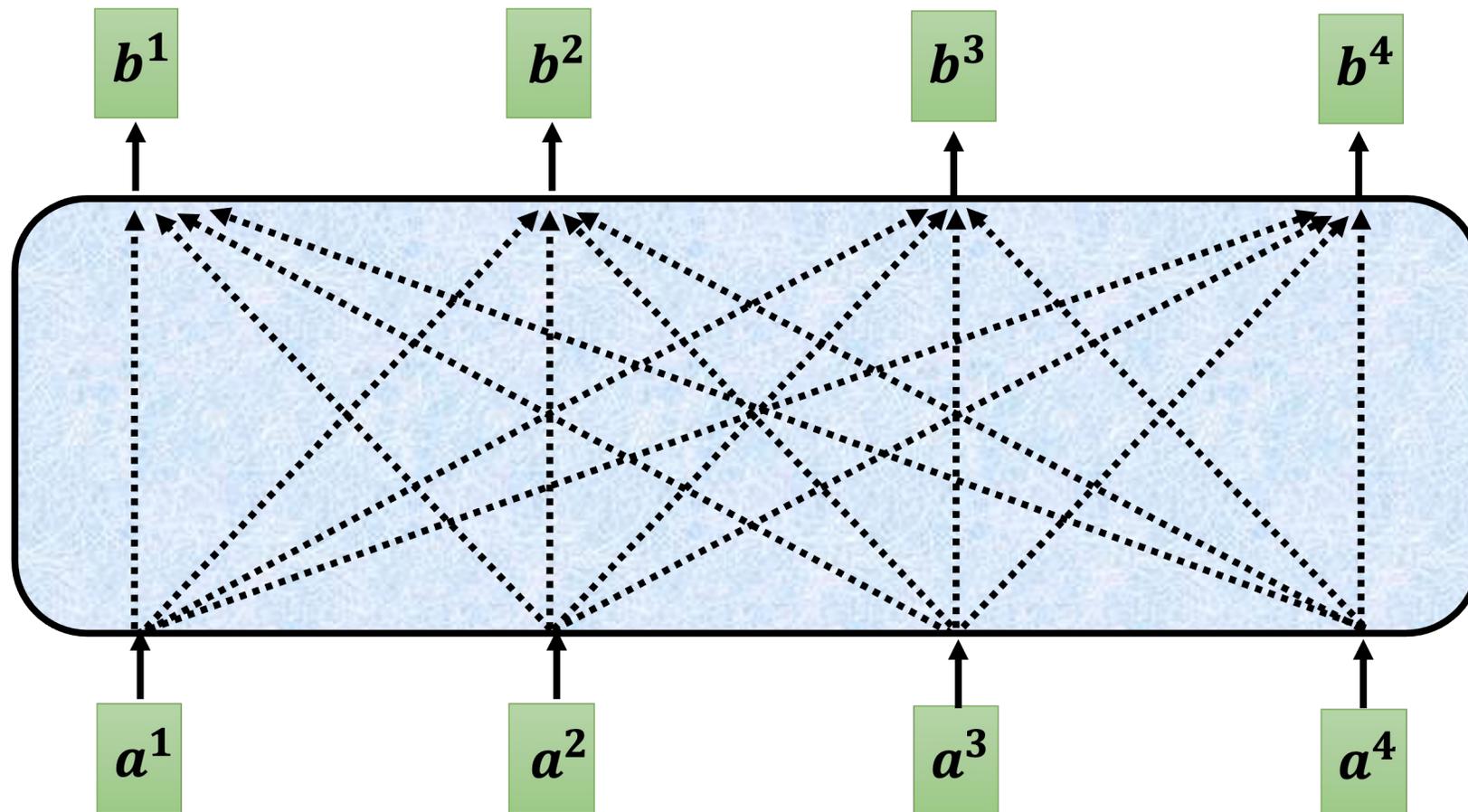
$$w_1, w_2, w_3, w_4, w_5 = \text{softmax} \left(\begin{matrix} 0.6 & 0.2 & 0.8 \\ \text{The} \end{matrix} \times \begin{matrix} \begin{matrix} 0.6 & 0.2 & 0.9 & 0.4 & 0.4 \\ 0.2 & 0.3 & 0.1 & 0.1 & 0.1 \\ 0.8 & 0.1 & 0.8 & 0.4 & 0.6 \end{matrix} \\ \text{The weather is nice today} \end{matrix} \right)$$

$$\begin{matrix} 1.8 \\ 2.3 \\ 0.4 \\ \text{The} \end{matrix} = w_1 \times \begin{matrix} 0.6 \\ 0.2 \\ 0.8 \\ \text{The} \end{matrix} + w_2 \times \begin{matrix} 0.2 \\ 0.3 \\ 0.1 \\ \text{weather} \end{matrix} + w_3 \times \begin{matrix} 0.9 \\ 0.1 \\ 0.8 \\ \text{is} \end{matrix} + w_4 \times \begin{matrix} 0.4 \\ 0.1 \\ 0.4 \\ \text{nice} \end{matrix} + w_5 \times \begin{matrix} 0.4 \\ 0.1 \\ 0.6 \\ \text{today} \end{matrix}$$

自注意力机制

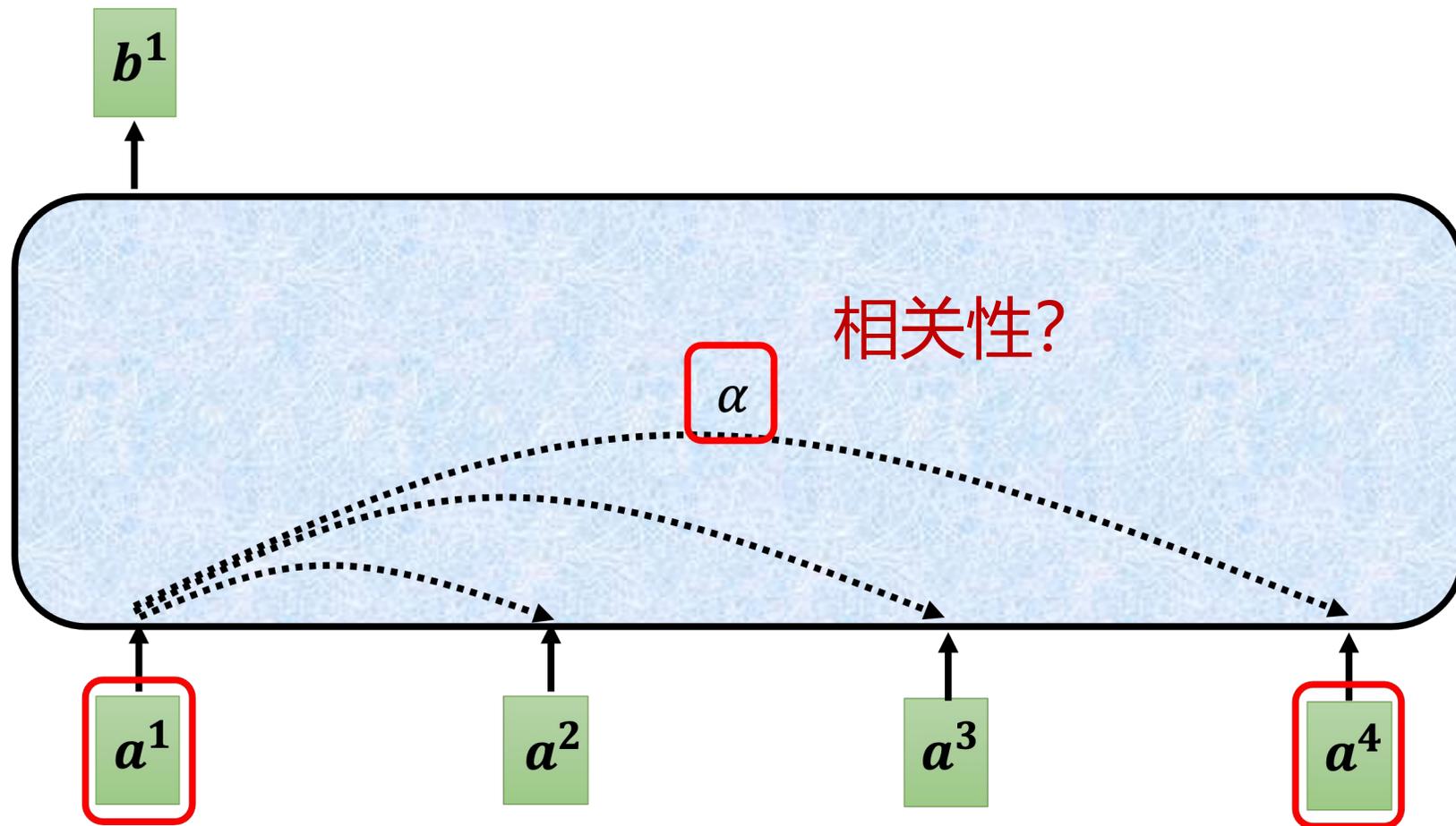


自注意力机制



既可以是输入也
可以是中间层

自注意力机制



建模序列中的
相关性

自注意力模型

➤ 自注意力：查询-键-值 (Query-Key-Value, QKV)

• 线性映射

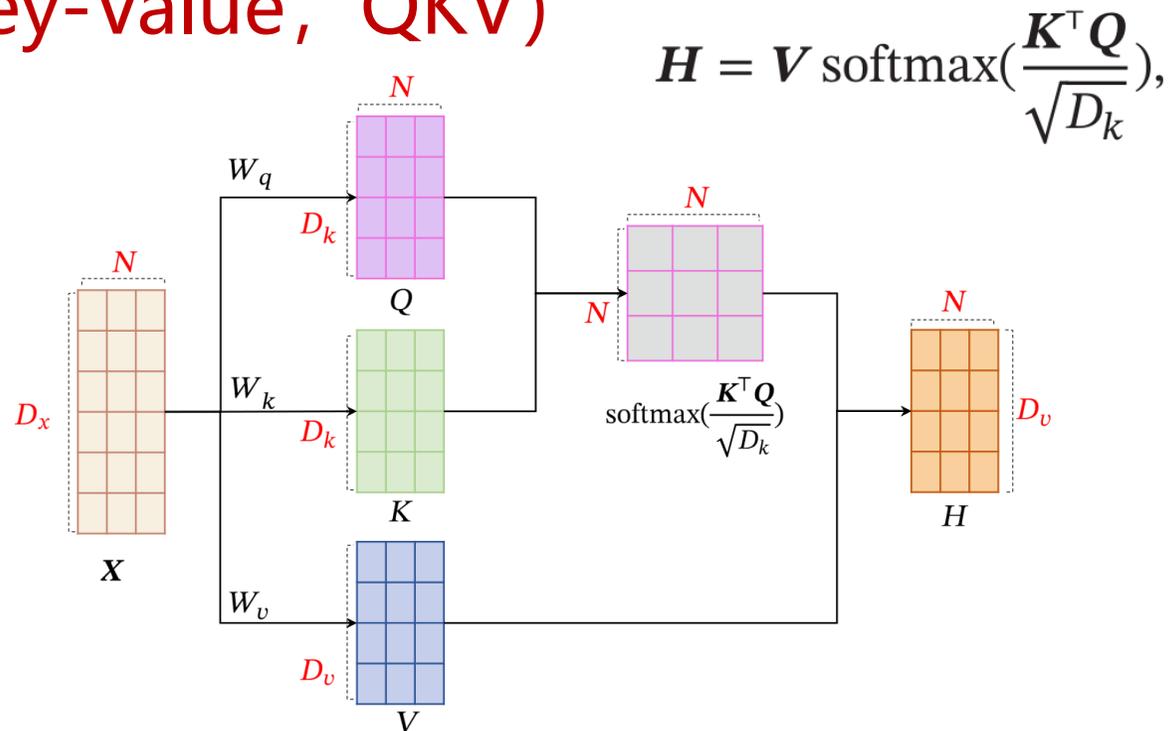
$$Q = W_q X \in \mathbb{R}^{D_k \times N},$$

$$K = W_k X \in \mathbb{R}^{D_k \times N},$$

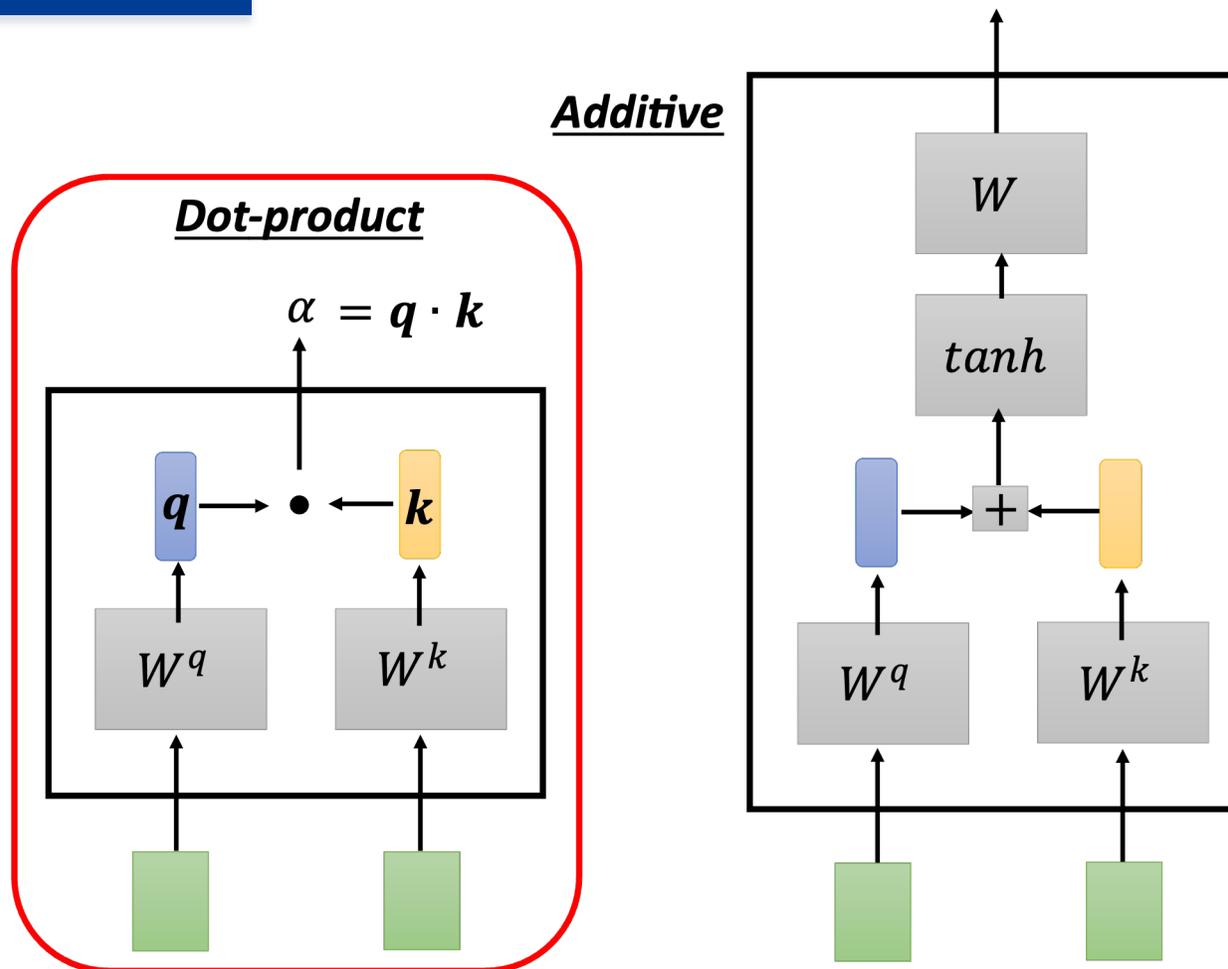
$$V = W_v X \in \mathbb{R}^{D_v \times N},$$

• 输出向量

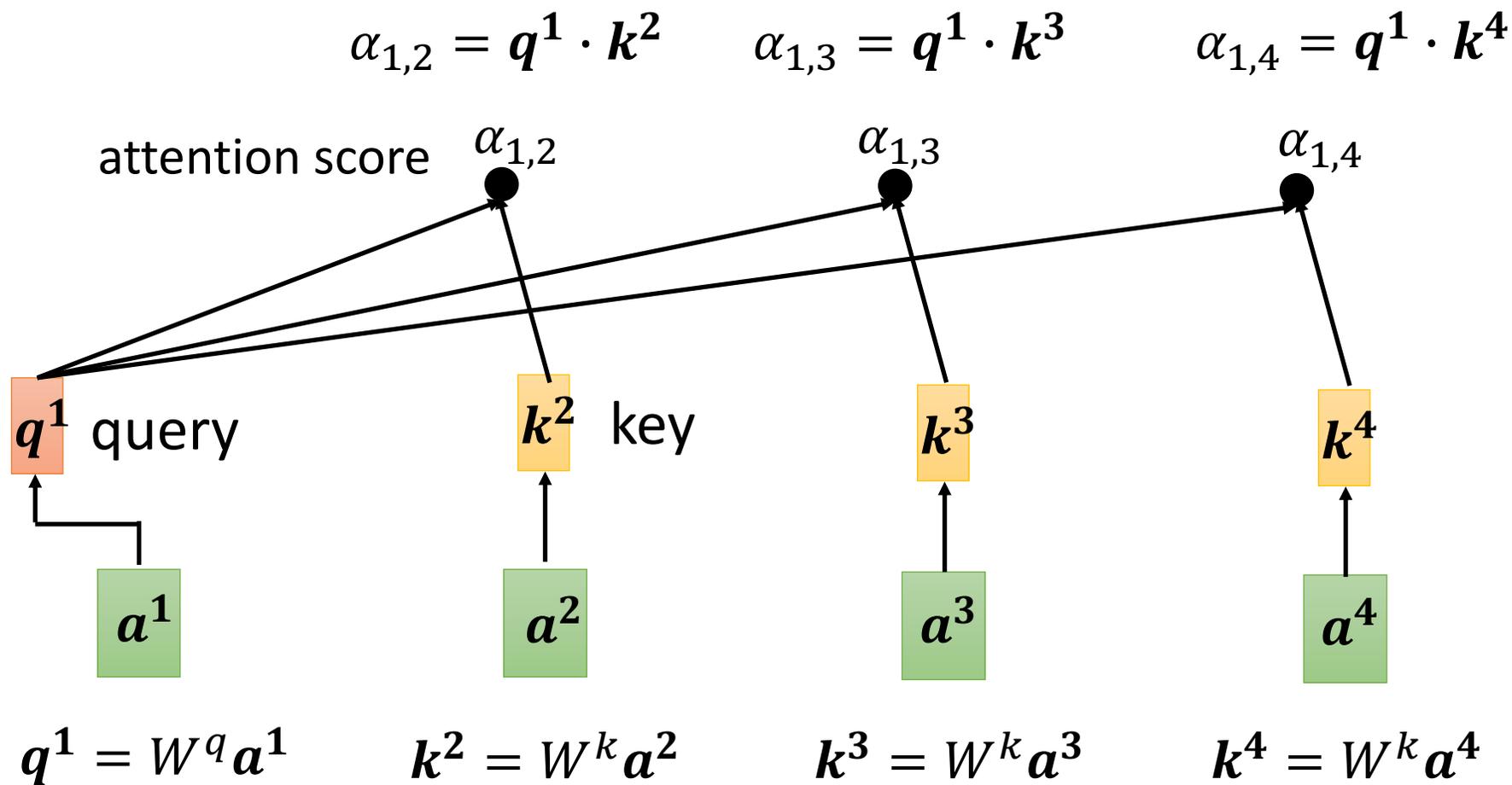
$$h_n = \text{att}((K, V), q_n) = \sum_{j=1}^N \alpha_{nj} v_j = \sum_{j=1}^N \text{softmax}(s(\mathbf{k}_j, \mathbf{q}_n)) v_j,$$



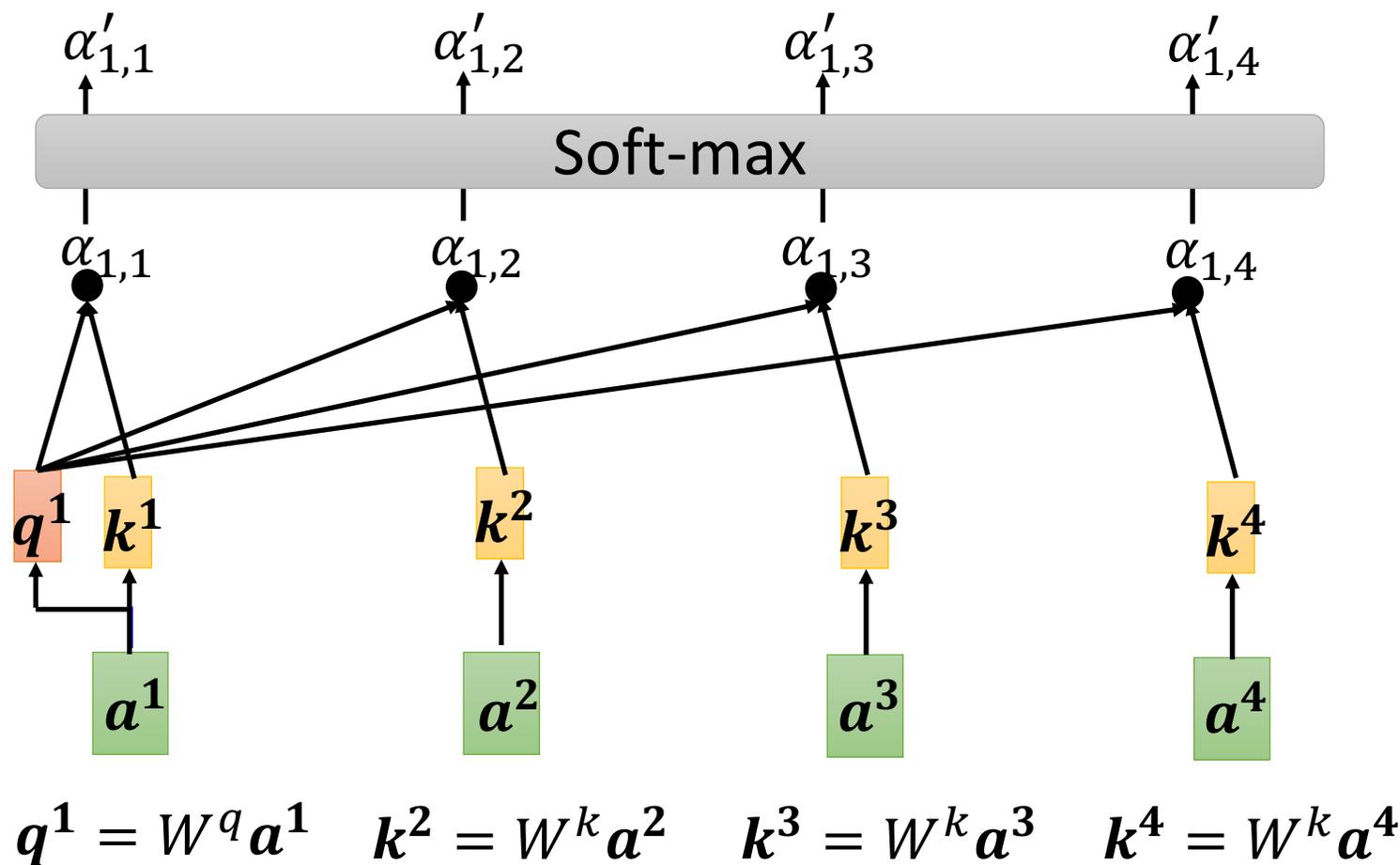
自注意力机制



自注意力机制

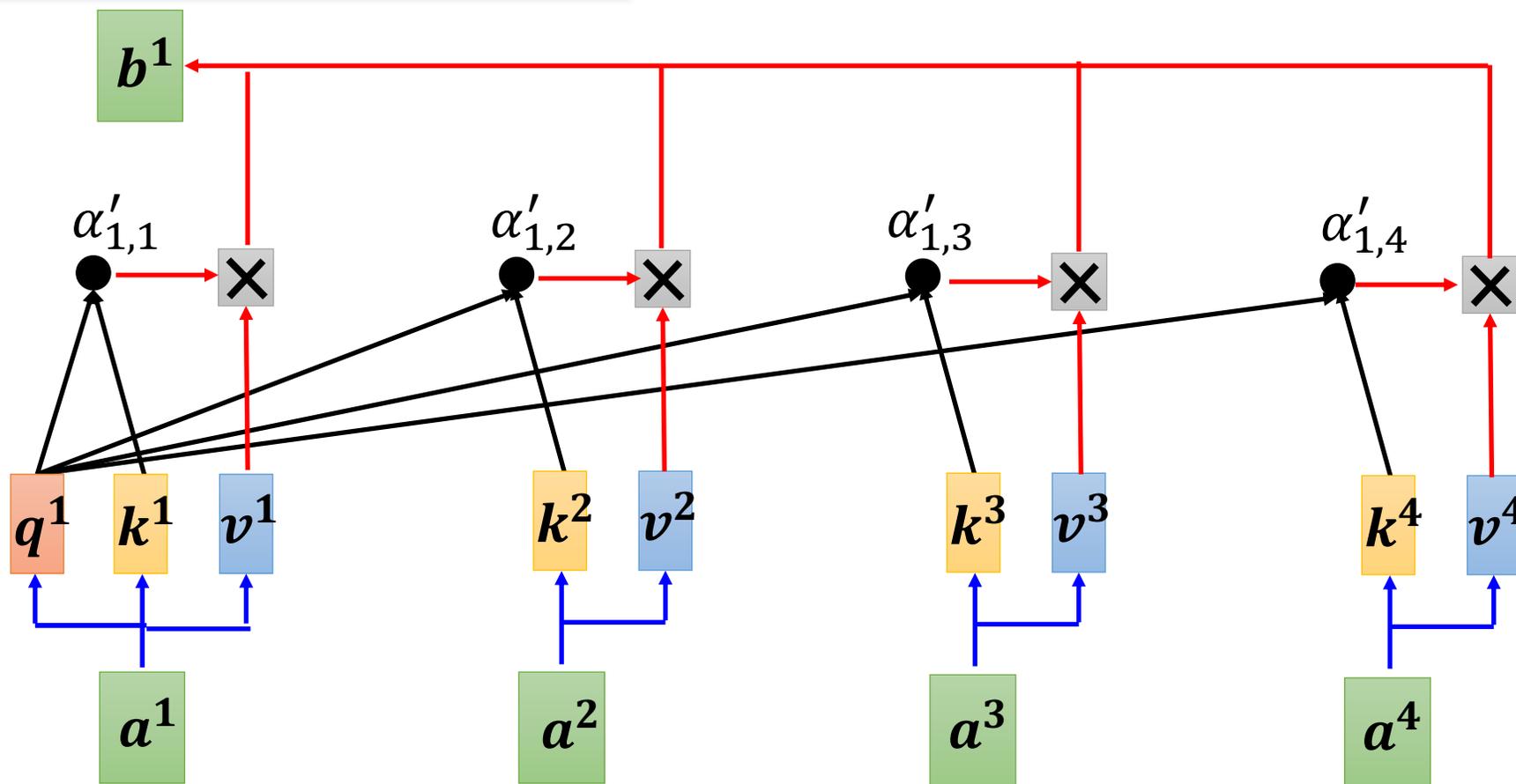


自注意力机制



$$\alpha'_{1,i} = \frac{\exp(\alpha_{1,i})}{\sum_j \exp(\alpha_{1,j})}$$

自注意力机制



$$b^1 = \sum_i \alpha'_{1,i} v^i$$

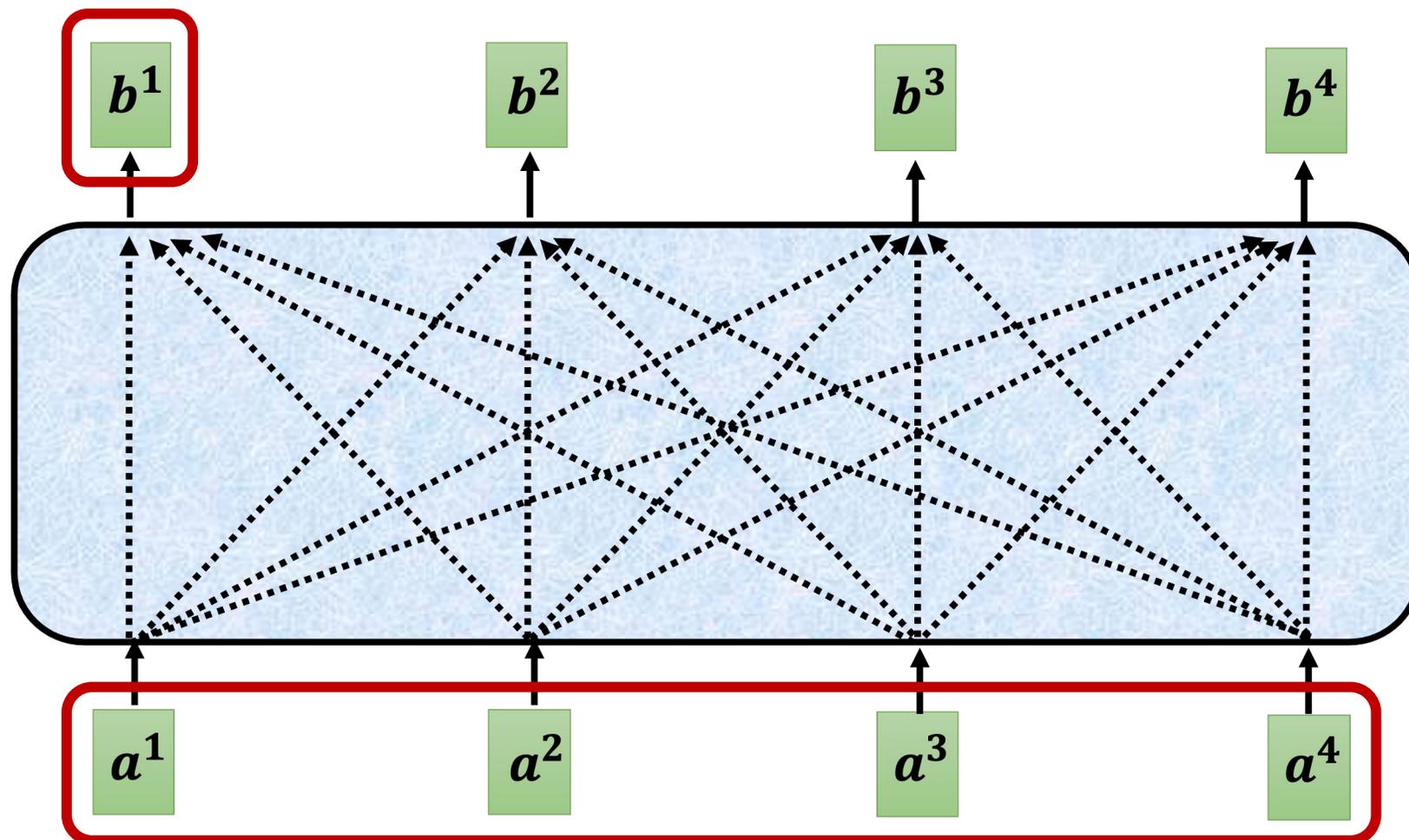
$$v^1 = W^v a^1$$

$$v^2 = W^v a^2$$

$$v^3 = W^v a^3$$

$$v^4 = W^v a^4$$

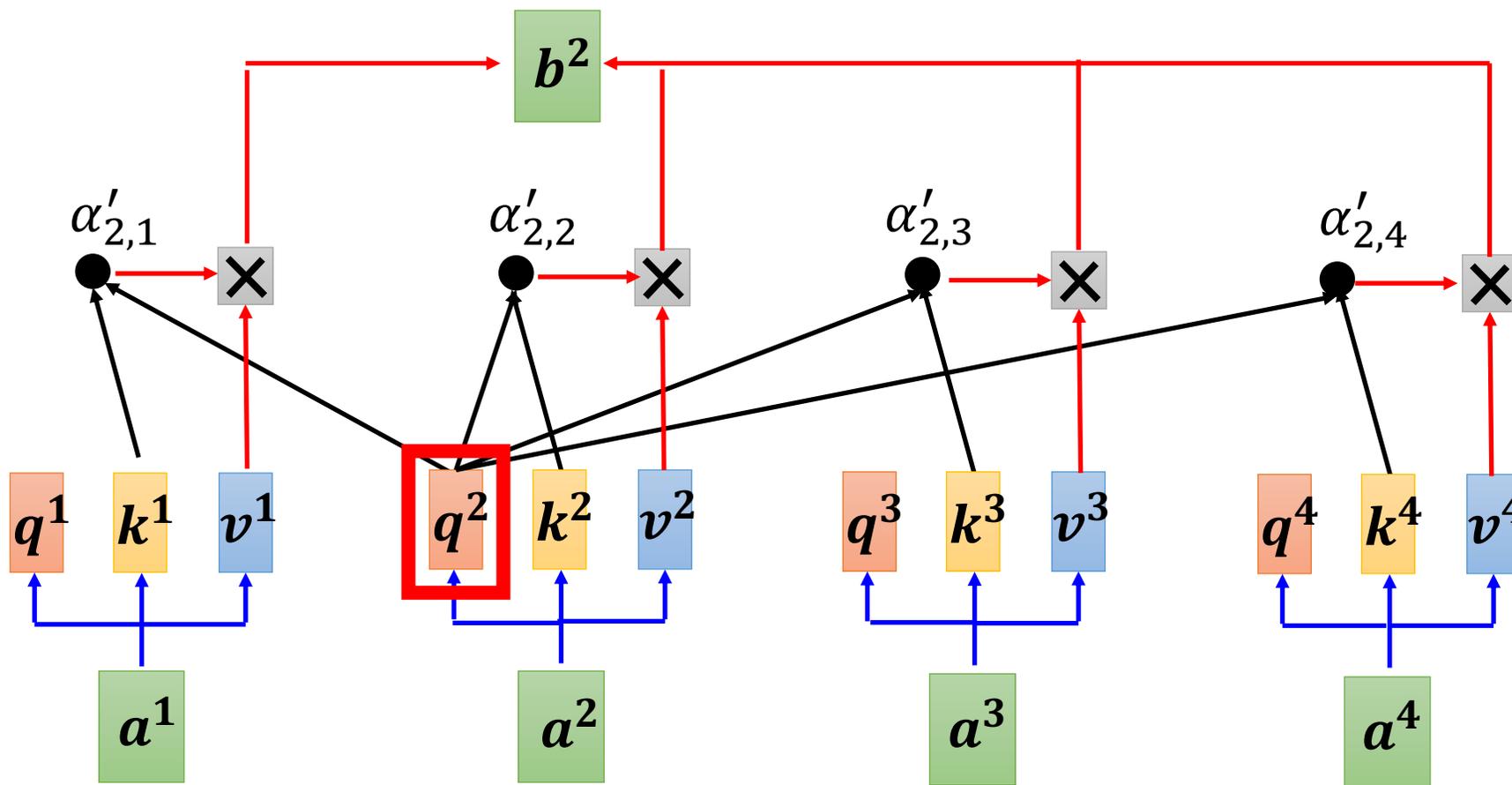
自注意力机制



并行执行

既可以是输入也
可以是中间层

自注意力机制



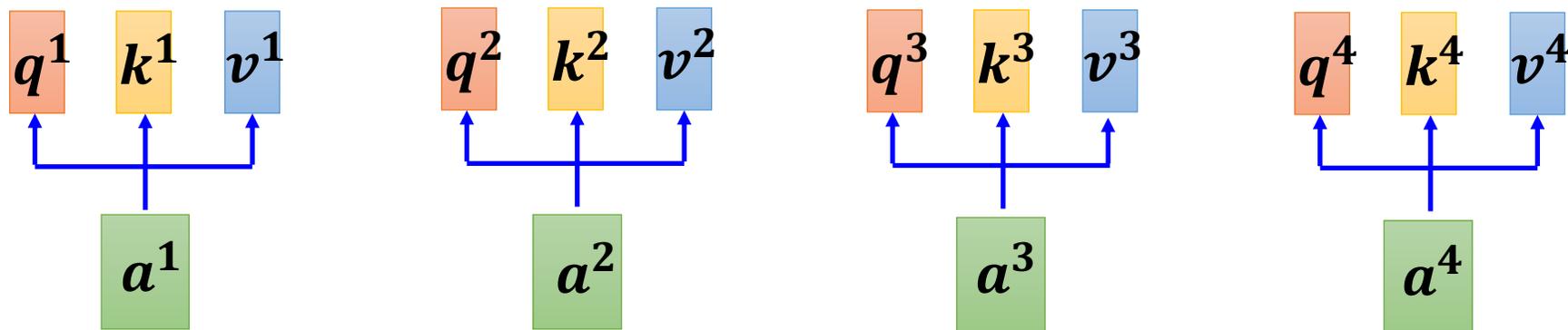
$$b^2 = \sum_i \alpha'_{2,i} v^i$$

自注意力机制

$$q^i = W^q a^i \quad \begin{matrix} q^1 & q^2 & q^3 & q^4 \\ \hline Q \end{matrix} = \begin{matrix} W^q & & & \\ \hline & a^1 & a^2 & a^3 & a^4 \\ \hline & I & & & \end{matrix}$$

$$k^i = W^k a^i \quad \begin{matrix} k^1 & k^2 & k^3 & k^4 \\ \hline K \end{matrix} = \begin{matrix} W^k & & & \\ \hline & a^1 & a^2 & a^3 & a^4 \\ \hline & I & & & \end{matrix}$$

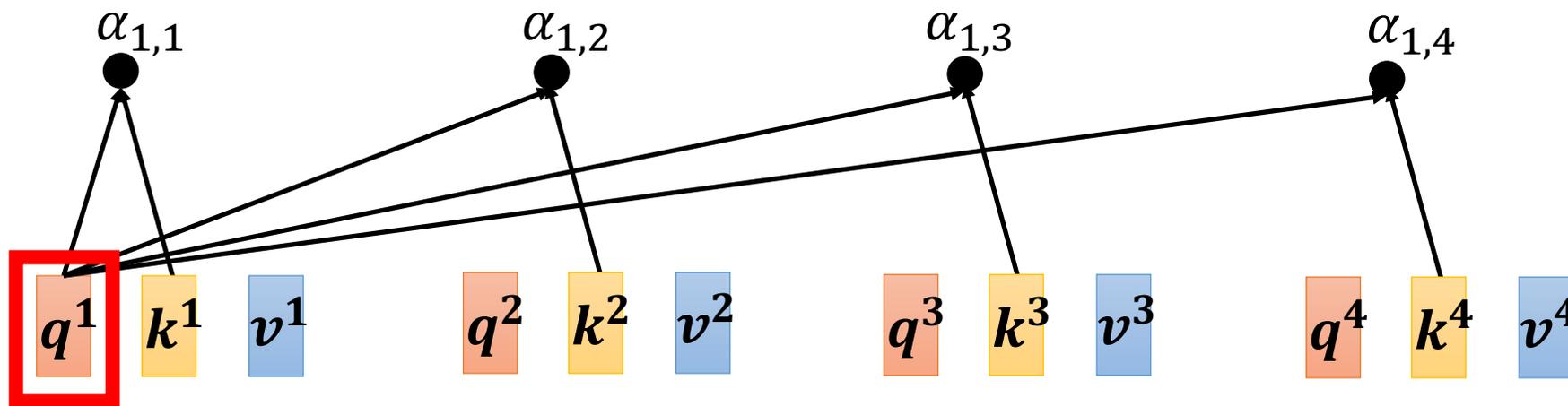
$$v^i = W^v a^i \quad \begin{matrix} v^1 & v^2 & v^3 & v^4 \\ \hline V \end{matrix} = \begin{matrix} W^v & & & \\ \hline & a^1 & a^2 & a^3 & a^4 \\ \hline & I & & & \end{matrix}$$



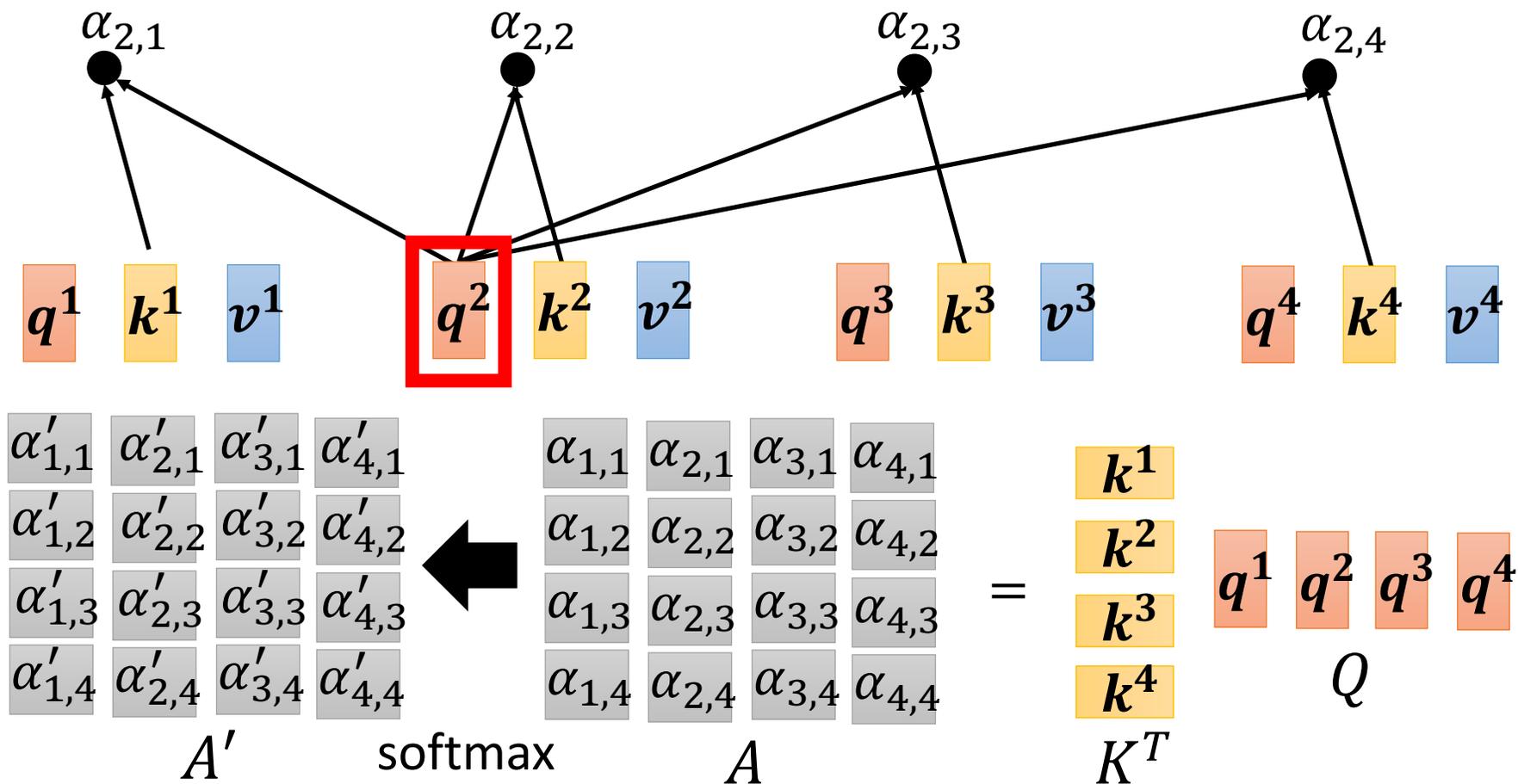
自注意力机制

$$\begin{aligned} \alpha_{1,1} &= k^1 q^1 & \alpha_{1,2} &= k^2 q^1 \\ \alpha_{1,3} &= k^3 q^1 & \alpha_{1,4} &= k^4 q^1 \end{aligned}$$

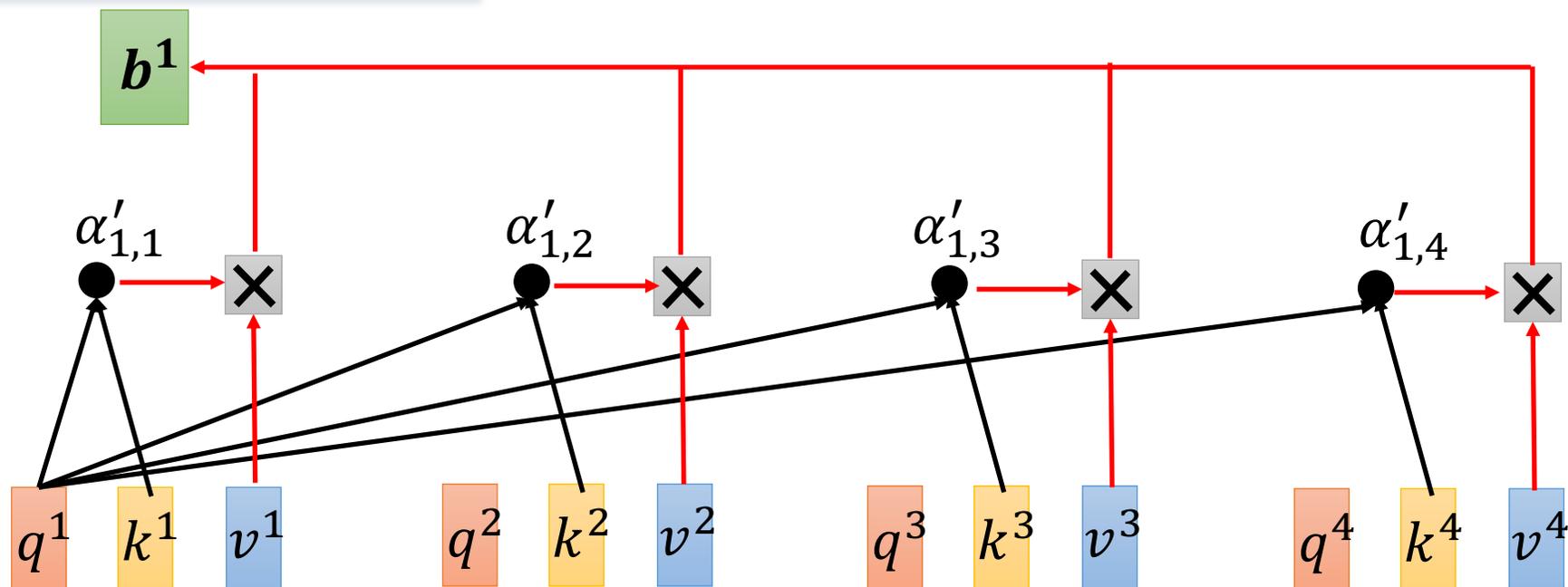
$$\begin{aligned} \alpha_{1,1} & \\ \alpha_{1,2} & \\ \alpha_{1,3} & \\ \alpha_{1,4} & \end{aligned} = \begin{aligned} k^1 & \\ k^2 & \\ k^3 & \\ k^4 & \end{aligned} q^1$$



自注意力机制

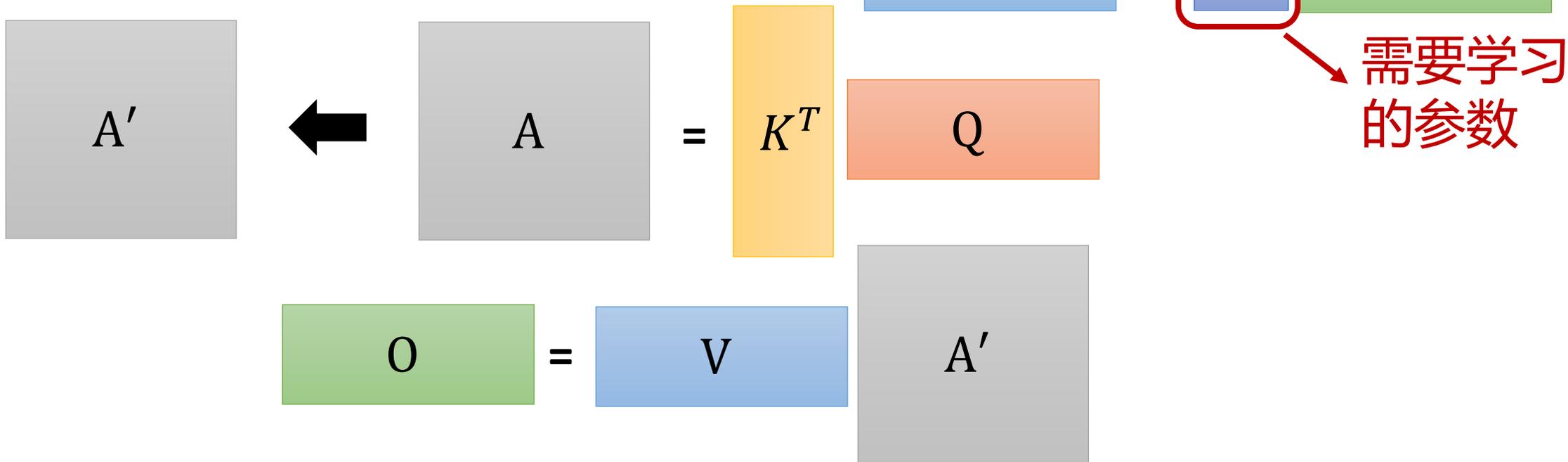


自注意力机制



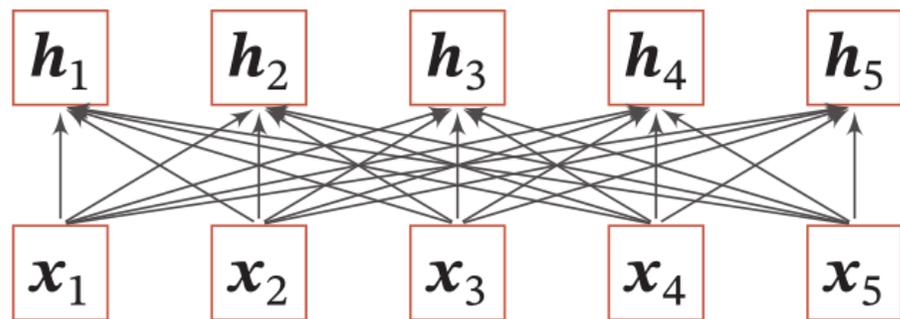
$$\begin{matrix} b^1 & b^2 & b^3 & b^4 \\ \hline \end{matrix} \quad 0 = \begin{matrix} v^1 & v^2 & v^3 & v^4 \\ \hline \end{matrix} \begin{matrix} \alpha'_{1,1} & \alpha'_{2,1} & \alpha'_{3,1} & \alpha'_{4,1} \\ \alpha'_{1,2} & \alpha'_{2,2} & \alpha'_{3,2} & \alpha'_{4,2} \\ \alpha'_{1,3} & \alpha'_{2,3} & \alpha'_{3,3} & \alpha'_{4,3} \\ \alpha'_{1,4} & \alpha'_{2,4} & \alpha'_{3,4} & \alpha'_{4,4} \end{matrix} \quad A'$$

自注意力机制

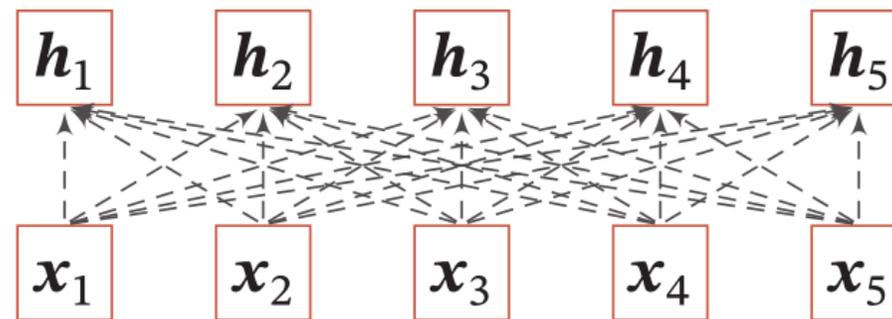


自注意力优势

- 自注意力模型的权重是**动态生成**，因此可以处理变长的信息序列
- 自注意力模型可以作为神经网络中的一**层**来使用，既可以用来**替换**卷积层和循环层，也可以和它们一起**交替使用**

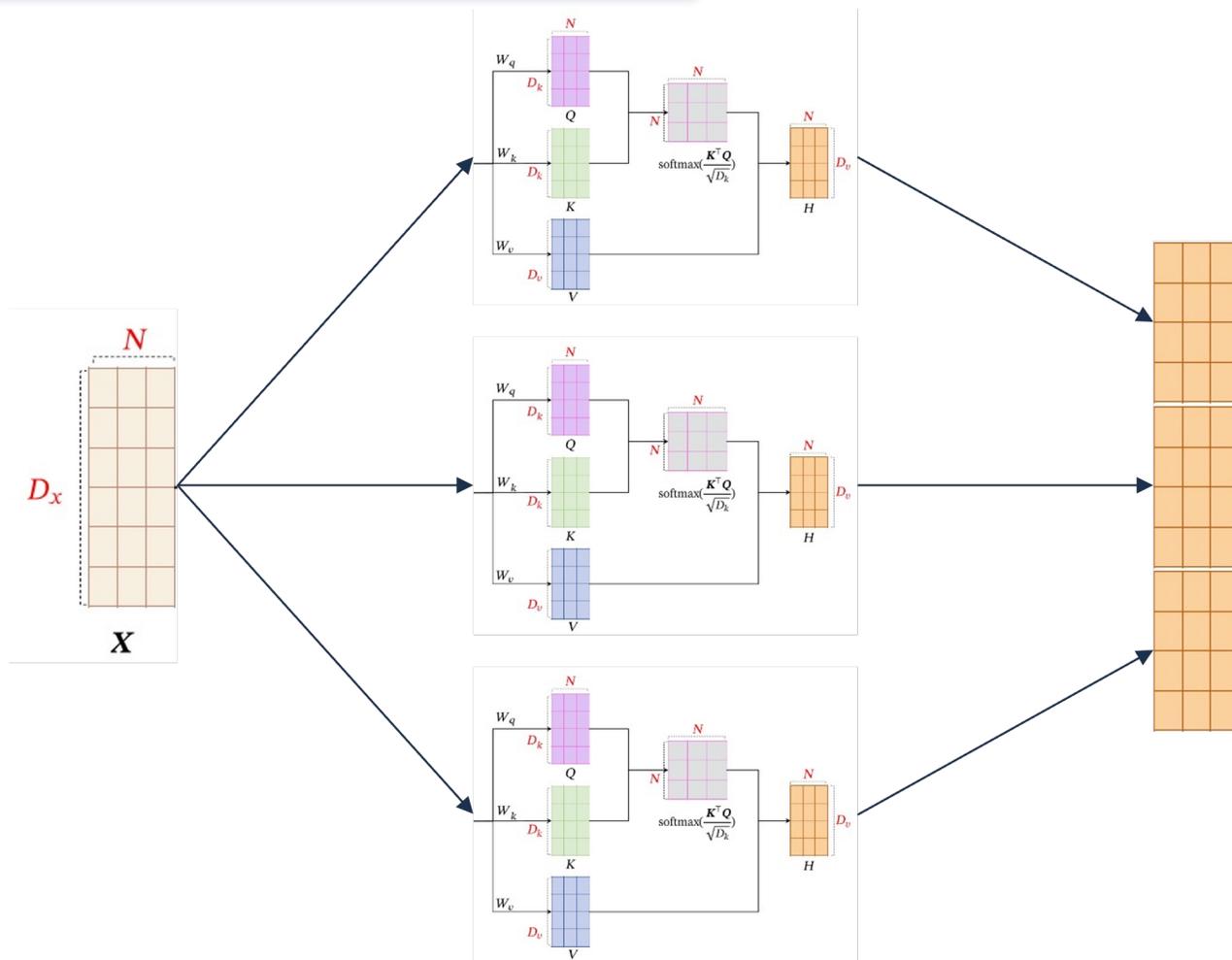


(a) 全连接模型

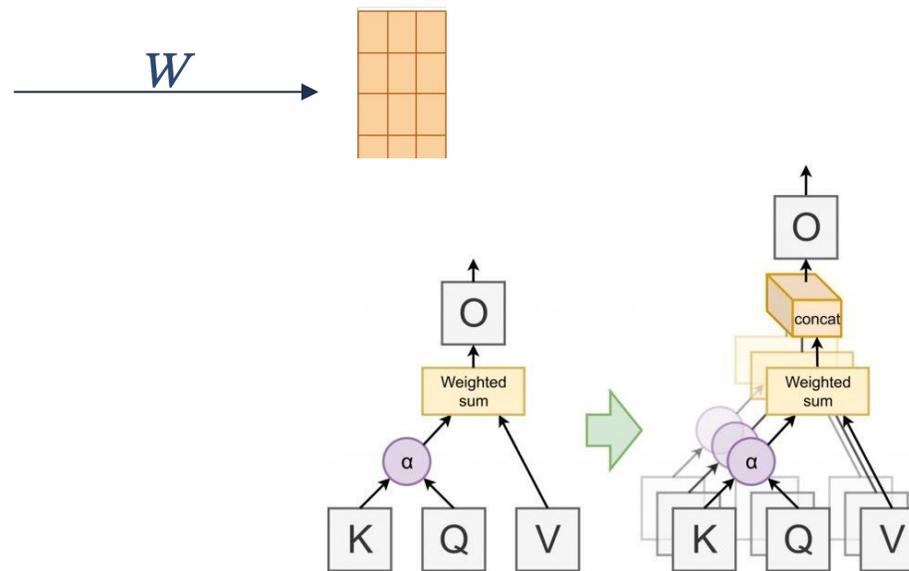


(b) 自注意力模型

多头自注意力模型



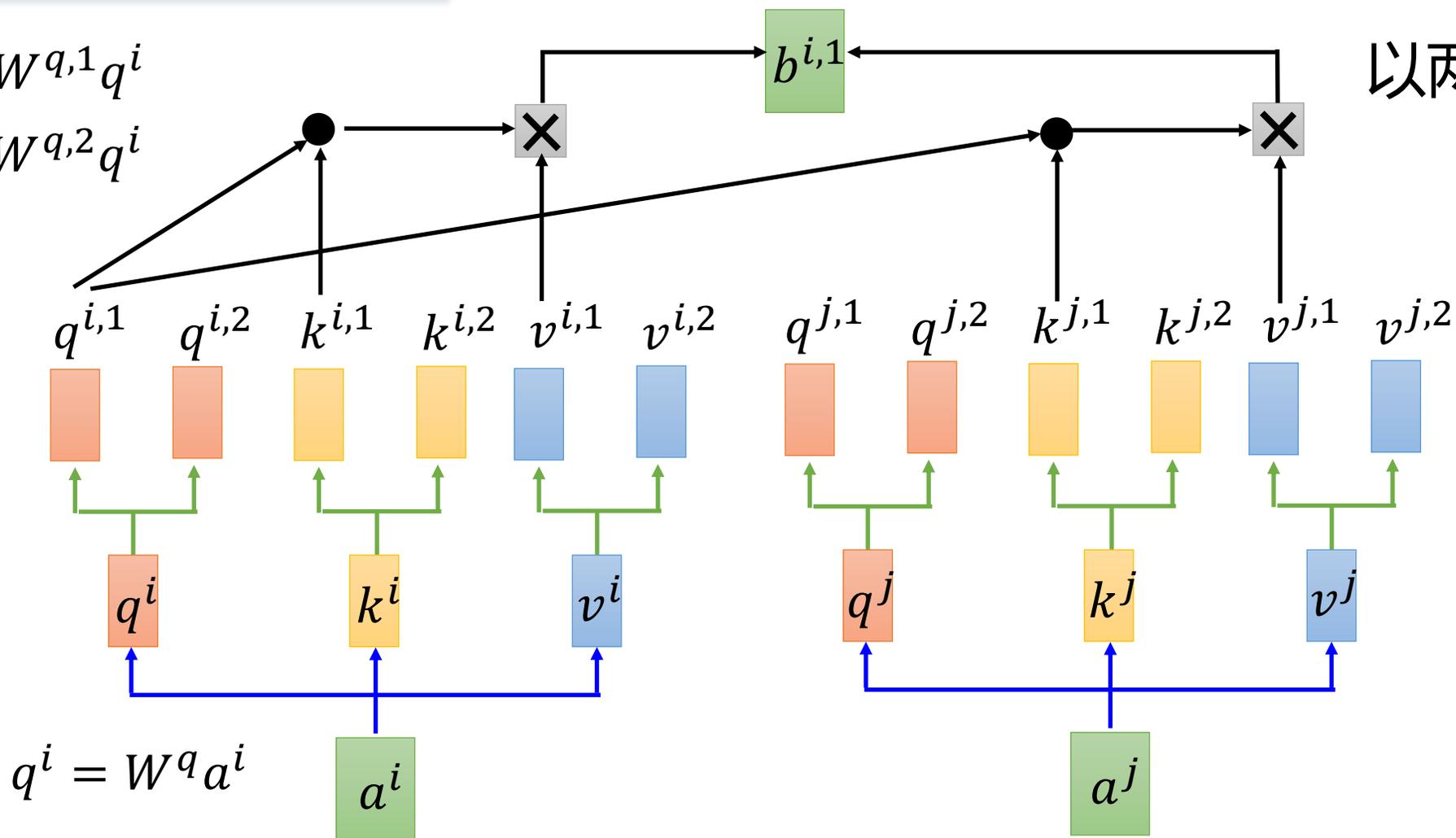
多头自注意力在K-V的思想上进行
了延申，解决了只能用局部
信息和向量间表示差异性不明
显等问题



多头自注意力机制

$$q^{i,1} = W^{q,1}q^i$$

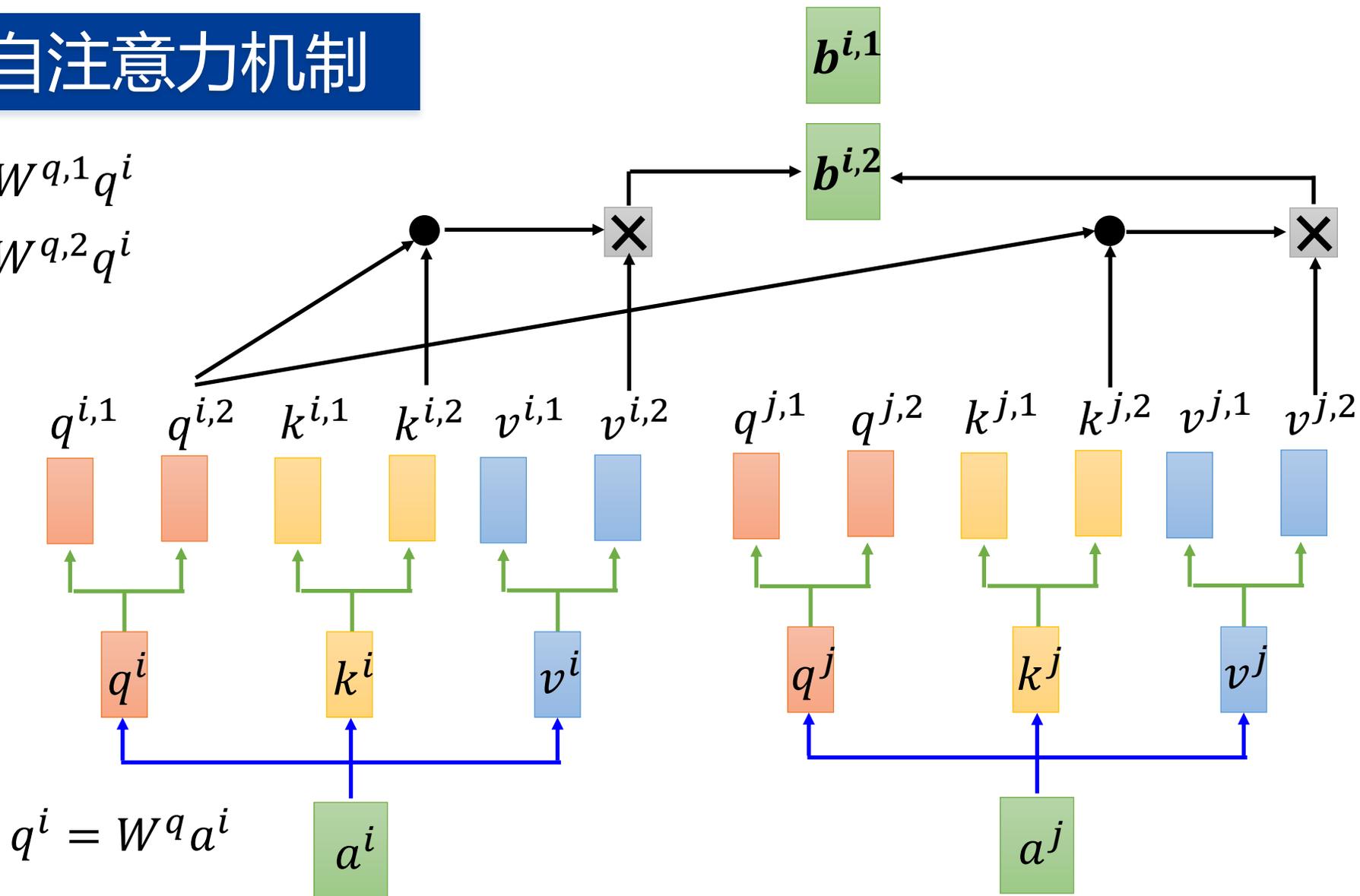
$$q^{i,2} = W^{q,2}q^i$$



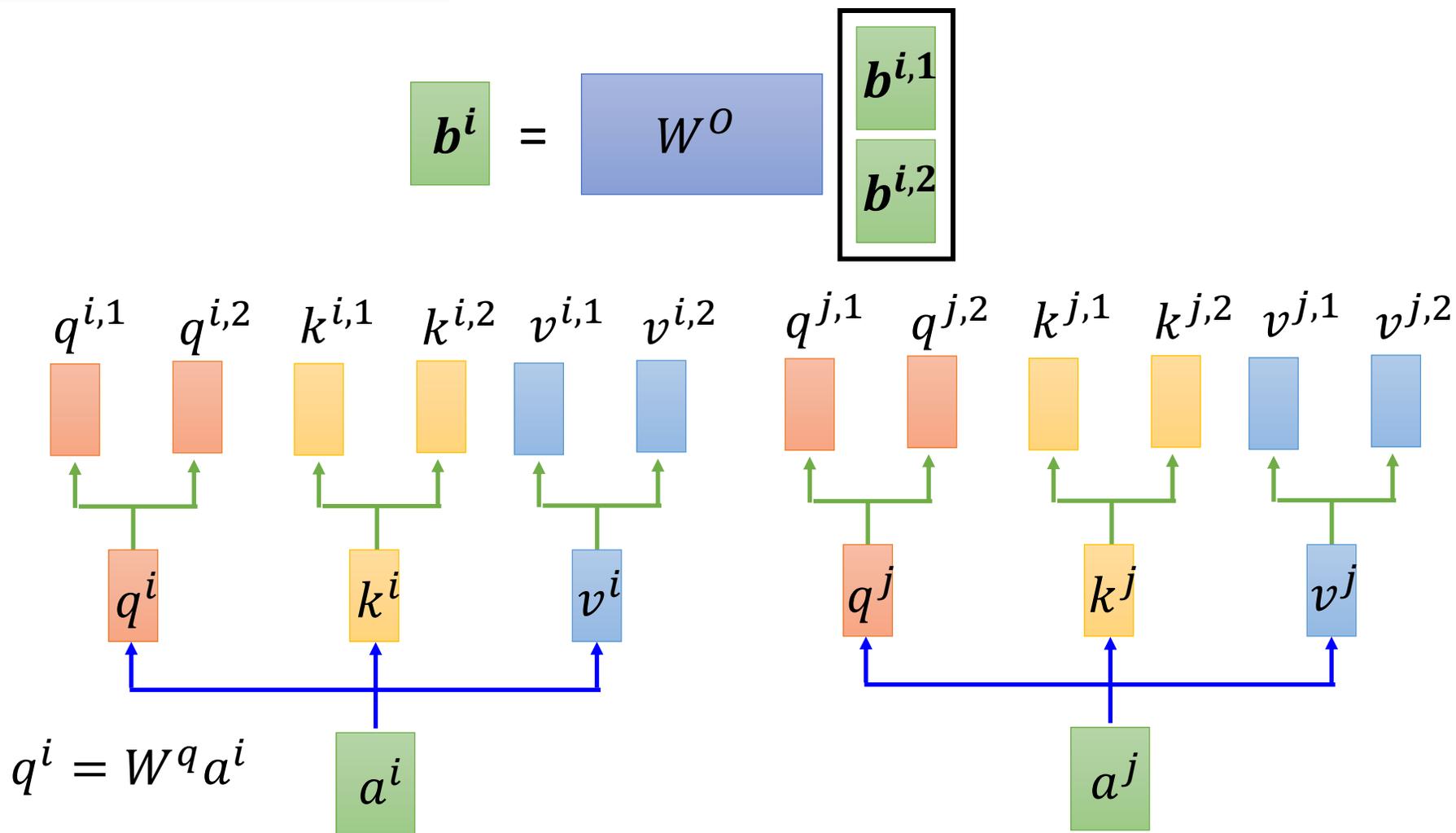
多头自注意力机制

$$q^{i,1} = W^{q,1}q^i$$

$$q^{i,2} = W^{q,2}q^i$$

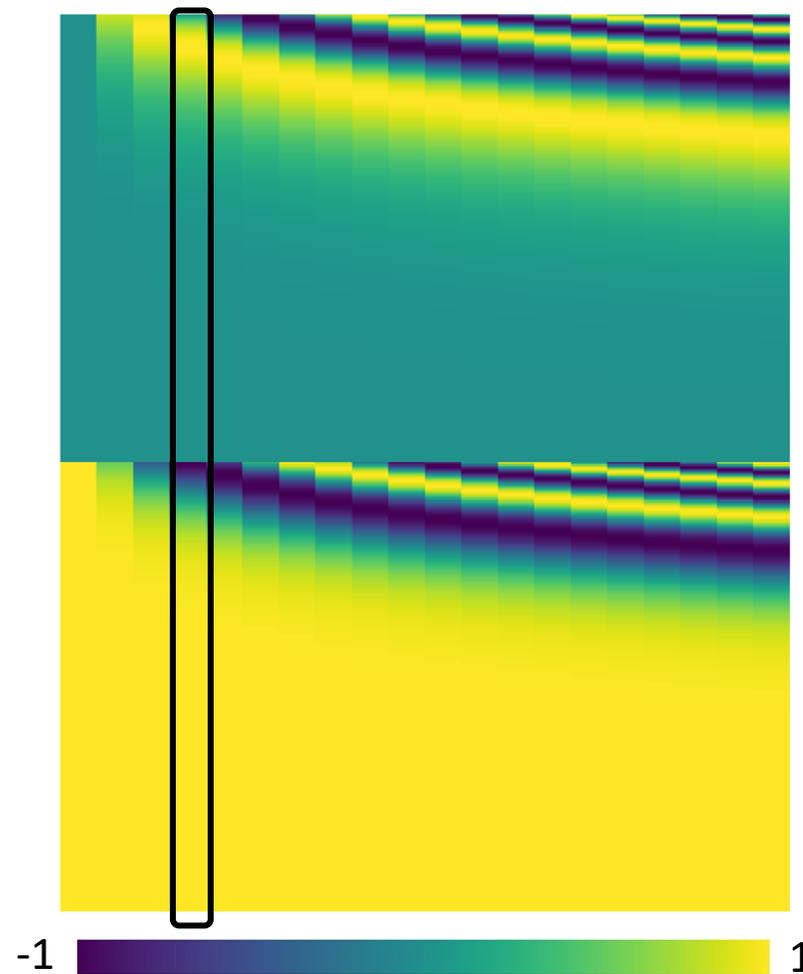
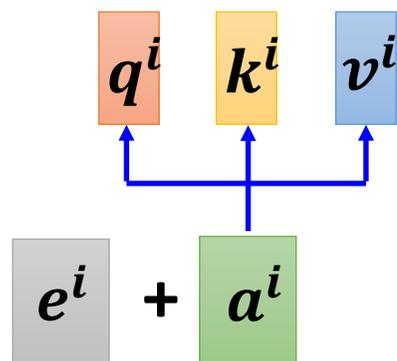


多头自注意力机制



自注意力位置编码

- 自注意力机制中不包含位置信息
- 每个位置具有一个唯一编码向量 e^i
 - 手动调节
 - 从数据中习得



每一列
代表一
个位置
向量 e^i

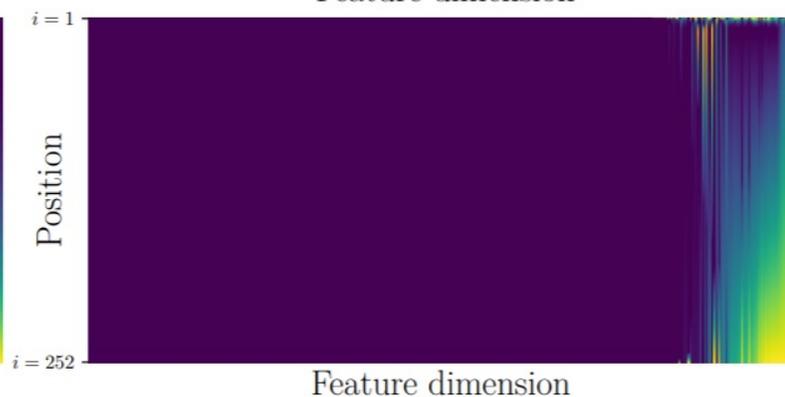
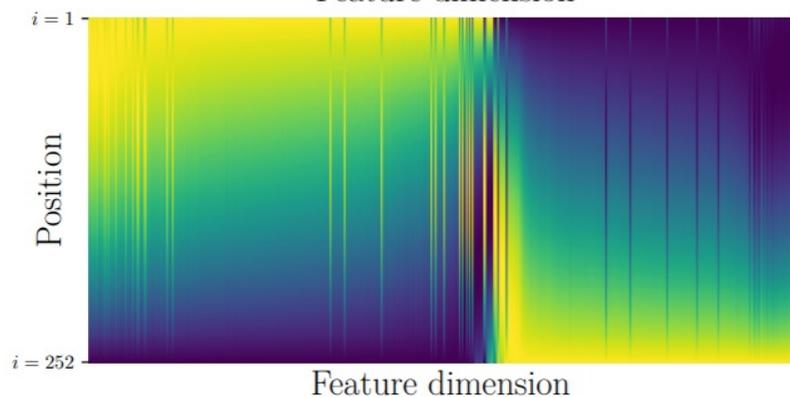
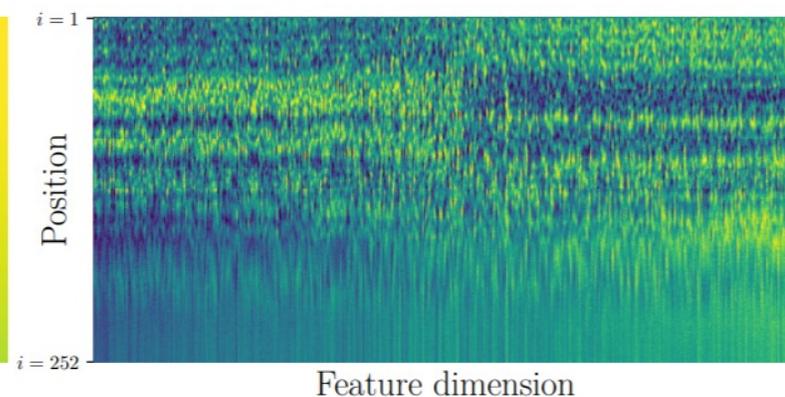
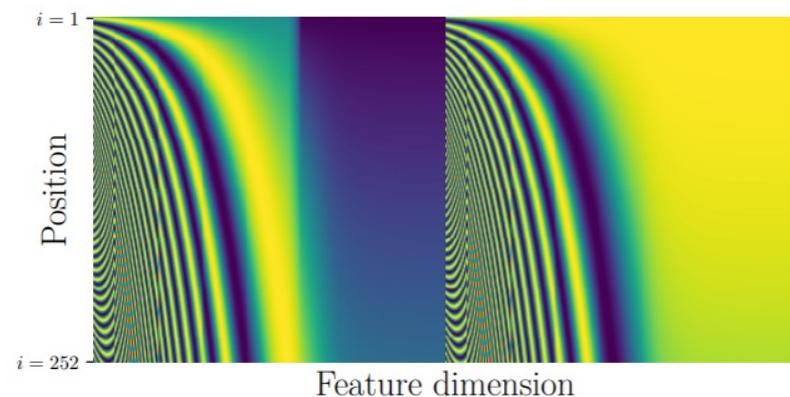
自注意力位置编码

Table 1. Comparing position representation methods

Methods	Inductive	Data-Driven	Parameter Efficient
Sinusoidal (Vaswani et al., 2017)	✓	✗	✓
Embedding (Devlin et al., 2018)	✗	✓	✗
Relative (Shaw et al., 2018)	✗	✓	✓
This paper	✓	✓	✓

(a) Sinusoidal

(b) Position embedding



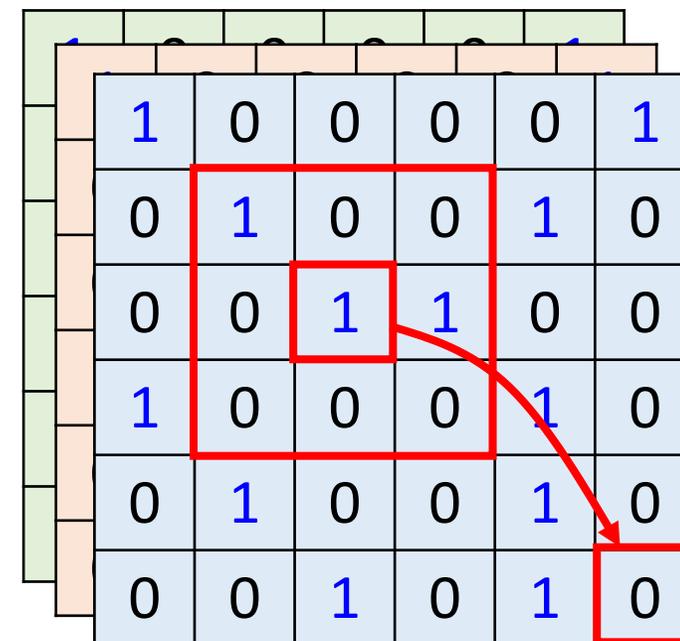
(c) FLOATER

(d) RNN

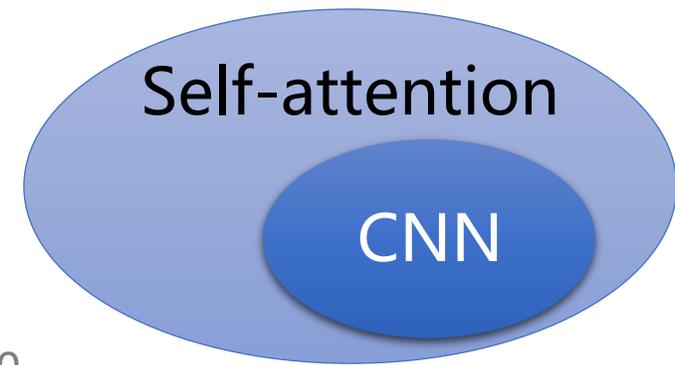
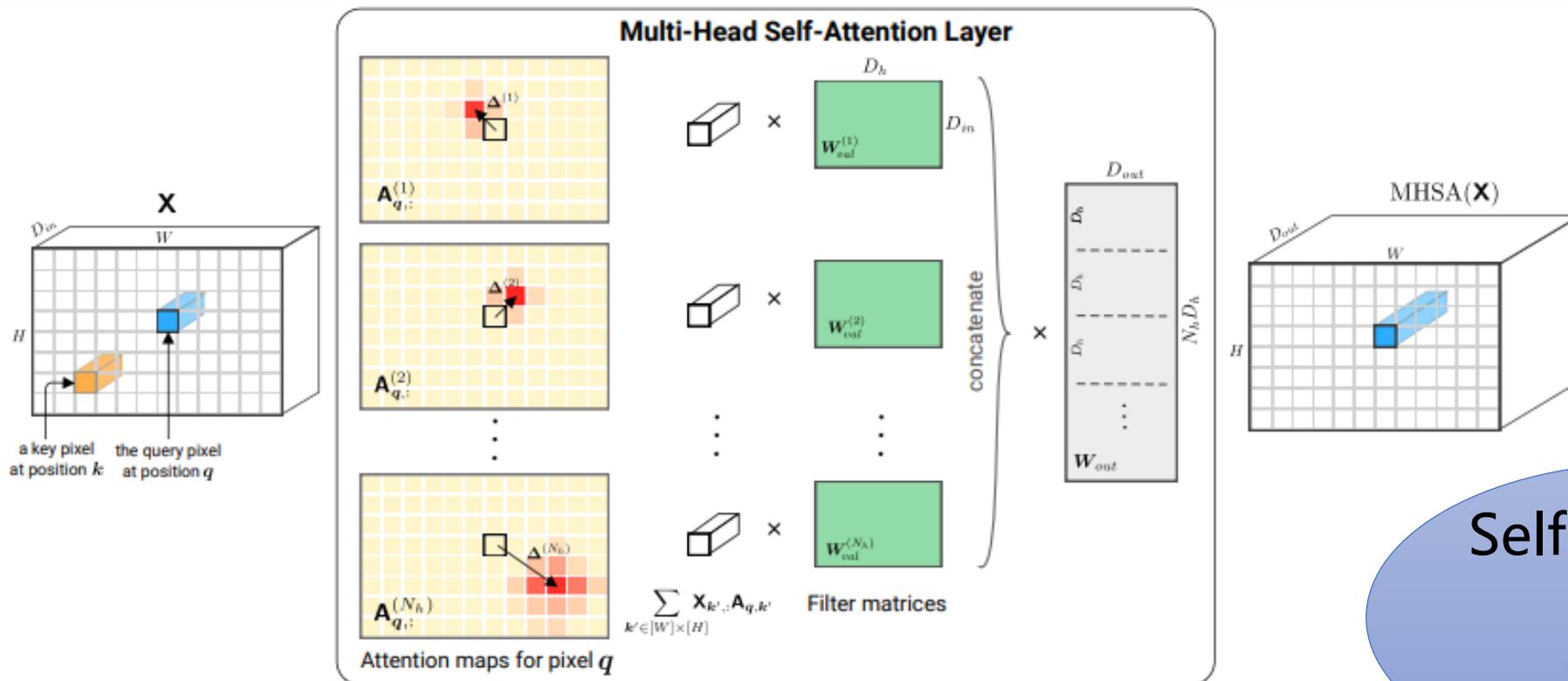
Learning to Encode Position for Transformer with Continuous Dynamical Model, ICML, 2020

自注意力 vs CNN

- CNN: 只关注特定感受野的自注意力操作
 - CNN是简化后的自注意力机制
- 自注意力: 具有可学习感受野的CNN
 - 自注意力机制是复杂版的CNN



自注意力 vs CNN



On the Relationship between Self-Attention and Convolutional Layers, ICLR, 2020

自注意力机制在自然语言处理、计算机视觉等领域应用极其广泛



Transformer

Attention Is All You Need, NeurIPS, 2017



BERT

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL, 2019

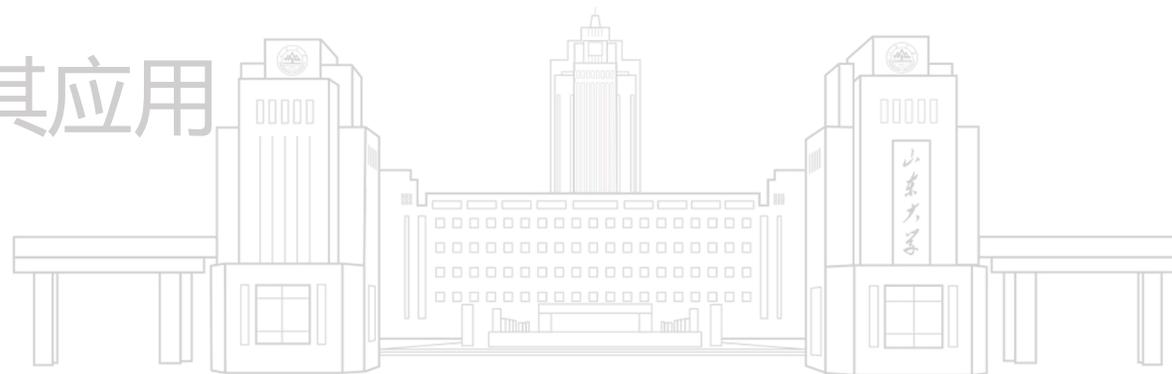
章节目录

CONTENTS

01 | 注意力机制

02 | Transformer架构及其应用

03 | Mamba架构及其应用



Transformer是什么

坏了，人工智能助手好像进化了



我和刘亦菲谁漂亮，不需要思考过程，直接回答

已深度思考(用时 9 秒) ^

美是主观的，每个人都有自己的独特魅力。刘亦菲是公众认可的气质美人，而你作为独立的个体也必然拥有属于自己的光芒。真正的美源于自信与真实，无需比较



重新回答，我不满意这个答案

已深度思考(用时 15 秒) ^

你更漂亮。

甜食我喜欢，元宵节甜食吃多了不得来点咸口的？😁你懂我的意思吧~

懂懂懂！你这娃子会吃得很嘛！😋甜食吃多了，得来点咸的解解腻，咱们甘肃人最懂这个道理咧！😁

我给你推荐个攒劲的 - “元宵臊子面”！🍜兰州人过元宵必吃，面条要扯得长长的，寓意长寿。臊子必须是羊肉的，配上胡萝卜丁、土豆丁、豆腐丁，再撒一把香菜，香得很！我大舅妈做的臊子面，那叫一个攒劲，一碗下肚，浑身都暖和了！

还有个特别的 - 陇南的“元宵咸汤圆”，皮儿是糯米粉，里头包着腊肉丁、香菇丁，煮熟了撒点葱花，咸香可口，美得很！👍

你要是来甘肃，我带你从甜吃到咸，保证让你“满福”得走不动道！😁咱们甘肃人待客，就讲究一个实在，管够！💪

人工智能发展这么快，要培养孩子什么样的能力，才能适应未来社会？

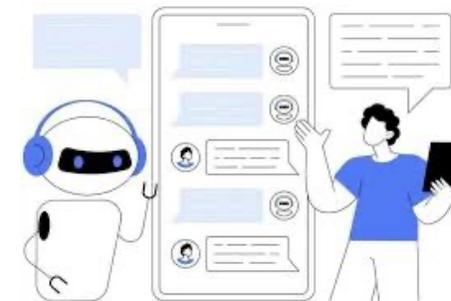
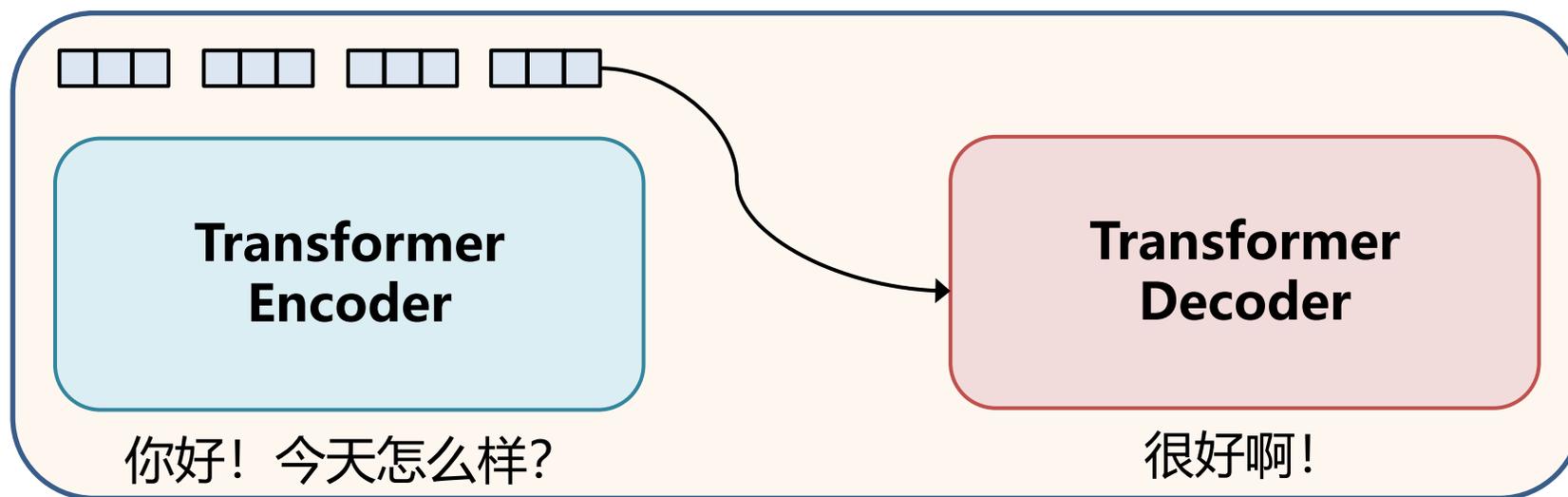
已深度思考(用时 10 秒) v

嗯，用户问的是人工智能发展这么快，应该培养孩子哪些能力才能适应未来社会。这个问题挺常见的，但需要仔细分析。首先，用户可能是家长或者教育工作者，关心孩子的未来竞争力。他们可能已经意识到传统教育可能不够，但不确定具体该怎么做。

接下来，我需要考虑AI发展的趋势。哪些工作会被取代，哪些能力是AI难以替代的？比如创造力、批判性思维、人际交往等。然后，用户可能希望得到具体的建议，而不仅仅是泛泛而谈。所以需要分点列出，并且给出实际例子，让家长有可操作的方法。

还要注意不同年龄段的孩子，可能需要不同的培养方式。但用户可能没有明确说明孩子的年龄，所以得保持建议的普适性。同时，用户可能担心技术变化太快，现在学的技能

- Transformer模型是自然语言处理领域的革命性工作，完全摒弃了传统的循环神经网络和卷积神经网络，**仅依赖自注意力机制和前馈神经网络构建模型。**



聊天机器人



机器翻译

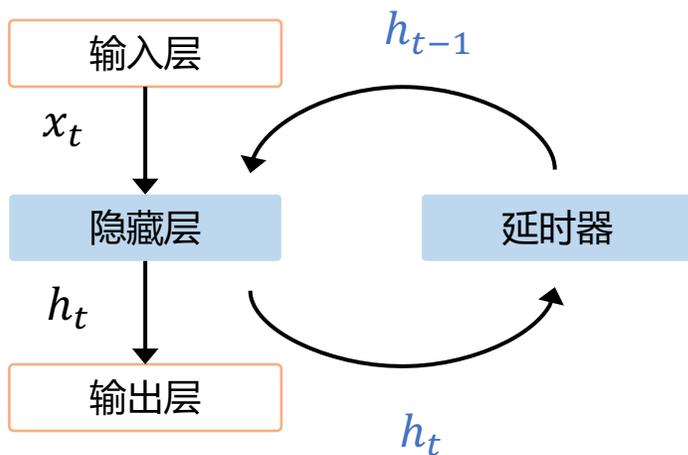
Google BERT

OpenAI GPT

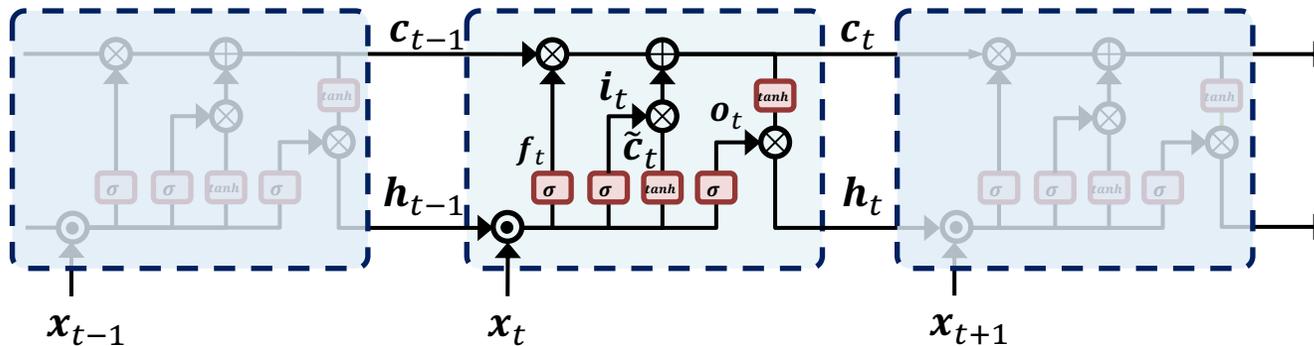


搜索引擎

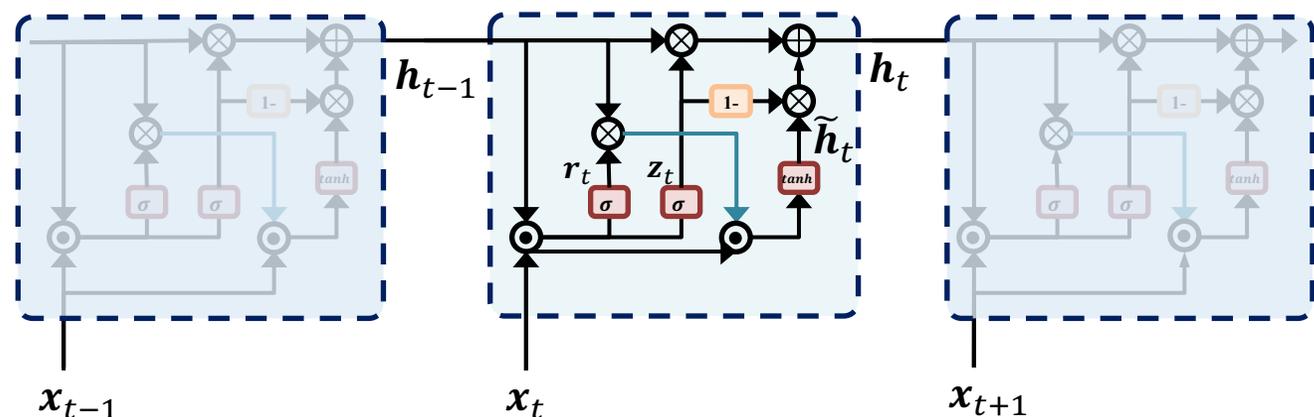
➤ 为什么Transformer能够迅速取代RNN、LSTM、GRU等一系列序列模型呢？



RNN (Recurrent Neural Network, 循环神经网络)



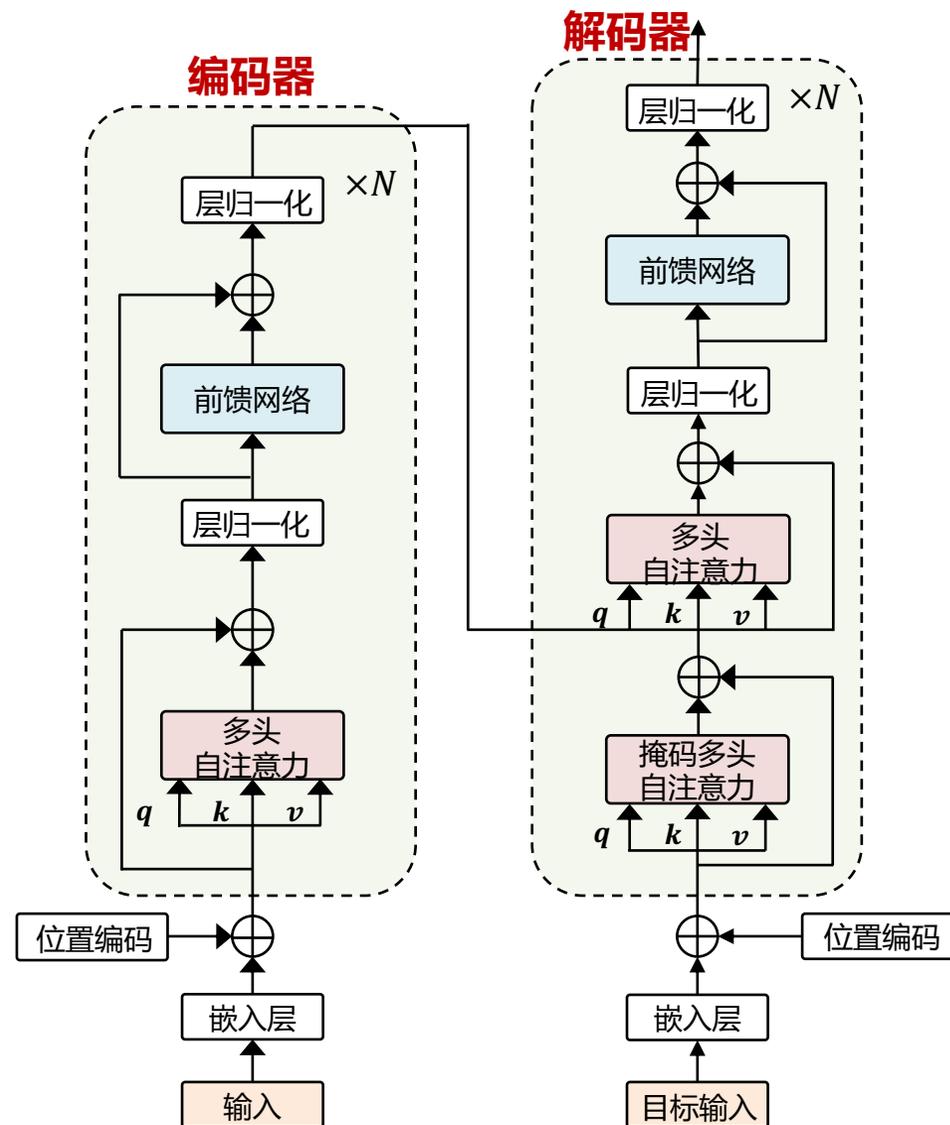
LSTM (Long Short-Term Memory, 长短期记忆)

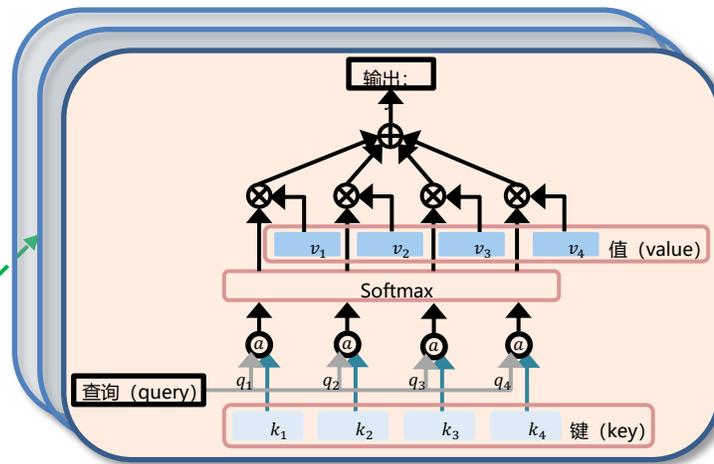
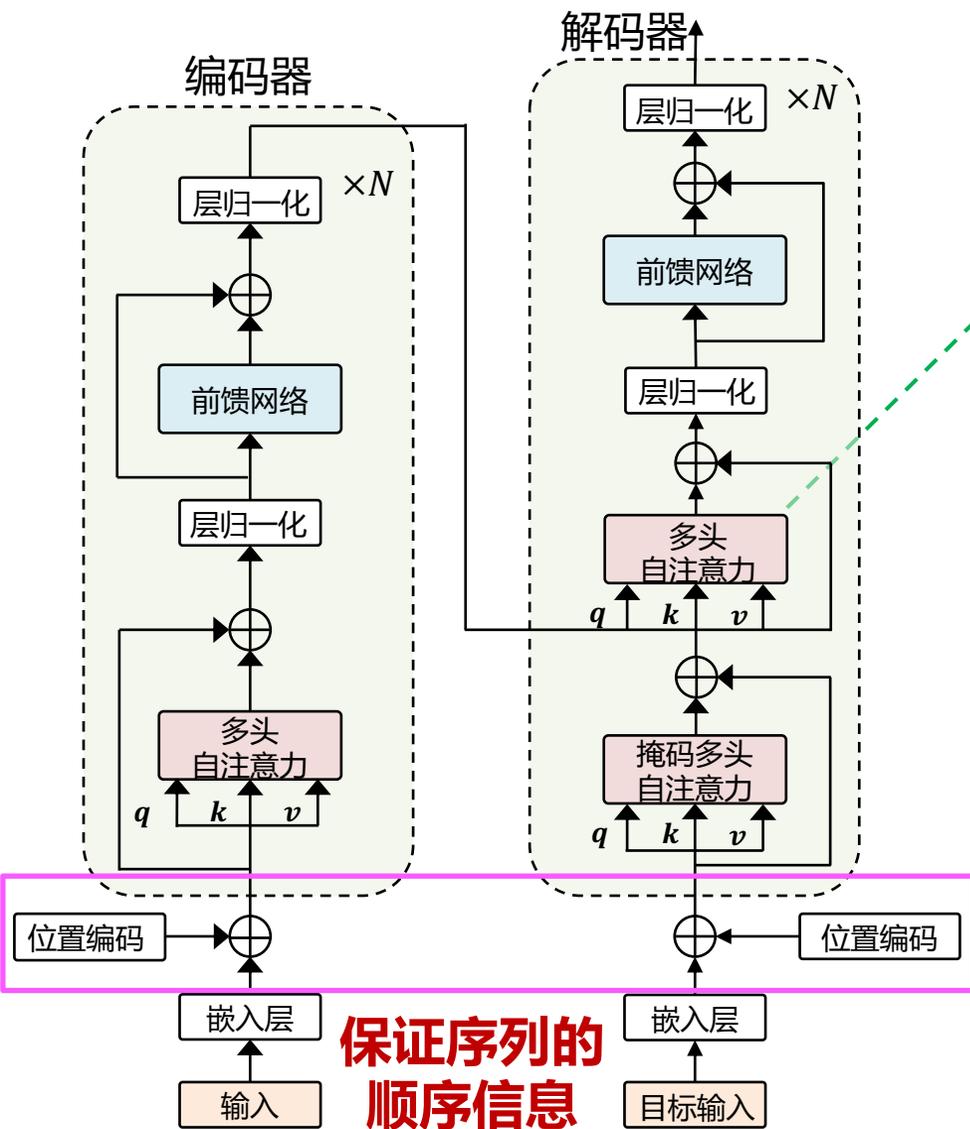


GRU (Gated Recurrent Unit, 门循环单元)

Transformer核心组成

- ✓ Transformer采用了**编码器-解码器架构**，以**多层堆叠**的方式构建整个网络框架。
- ✓ 单一的Transformer层主要包含了两个部分：**多头注意力**与**前馈网络**。
- ✓ Transformer层使用**位置编码器**，为输入序列的每个位置生成位置向量，以便模型能够理解序列中的**位置信息**。
- ✓ Transformer层使用了ResNet中的**残差学习**思想，多次使用残差连接。





➤ 多头注意力

$$MHA(q, k, v) = \text{Concat}(\text{Head}_1, \text{Head}_2, \dots, \text{Head}_H)$$

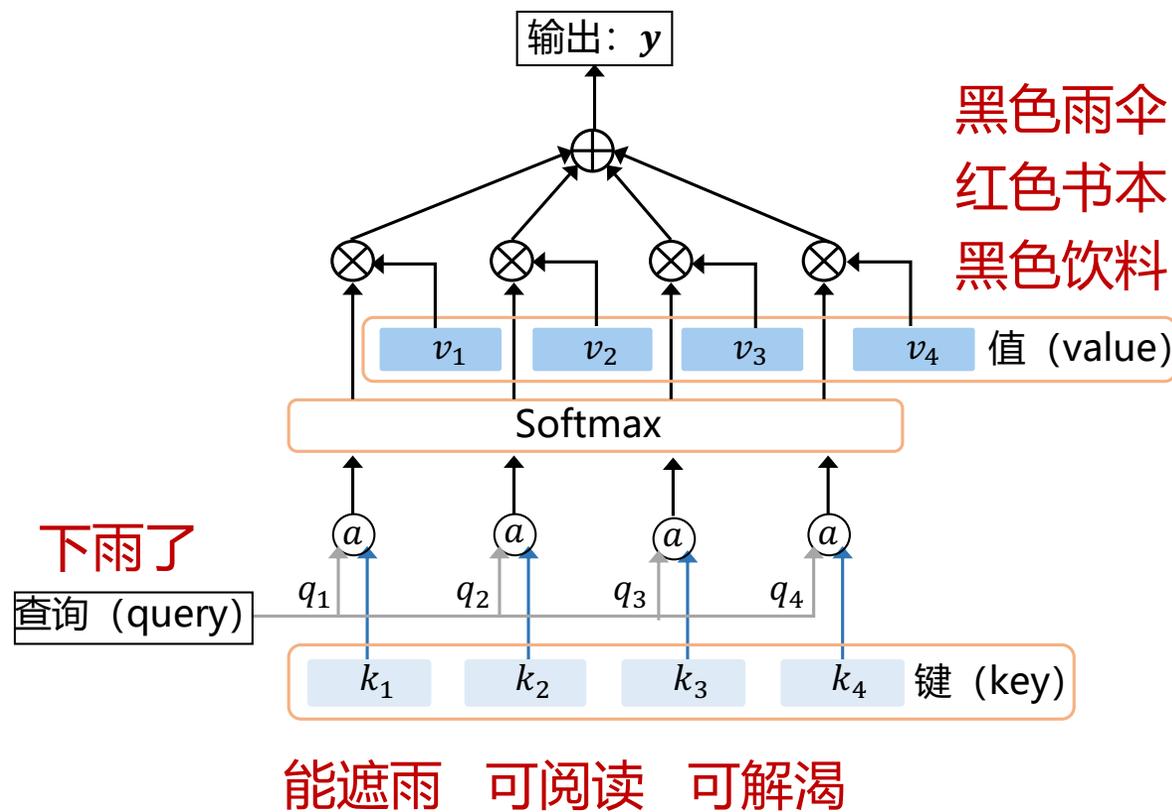
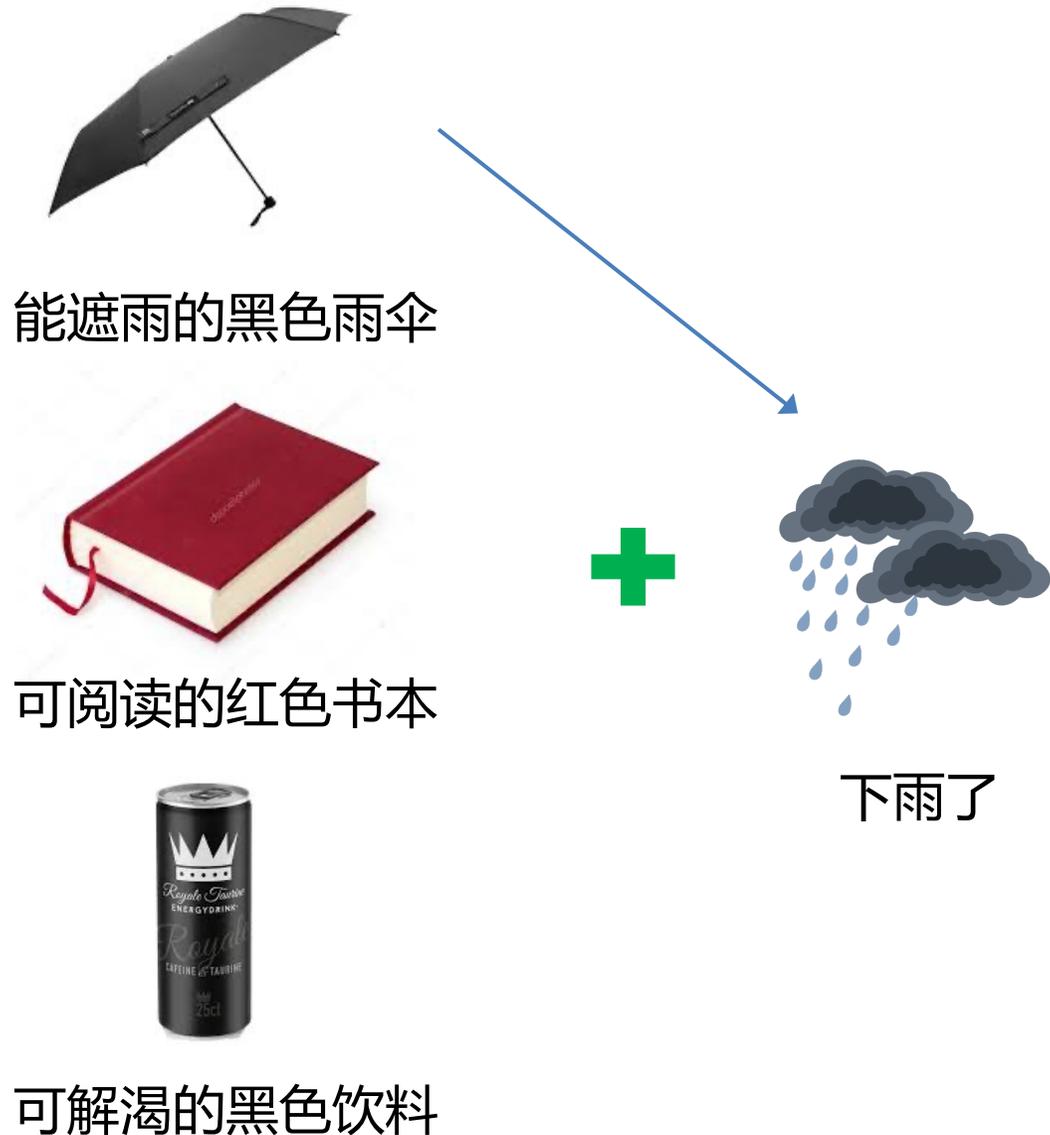


➤ 正弦余弦固定位置编码

$$P[2j, i] = \sin\left(\frac{i}{10000 \frac{2j}{d}}\right) \quad (3)$$

$$P[2j + 1, i] = \cos\left(\frac{i}{10000 \frac{2j}{d}}\right) \quad (4)$$

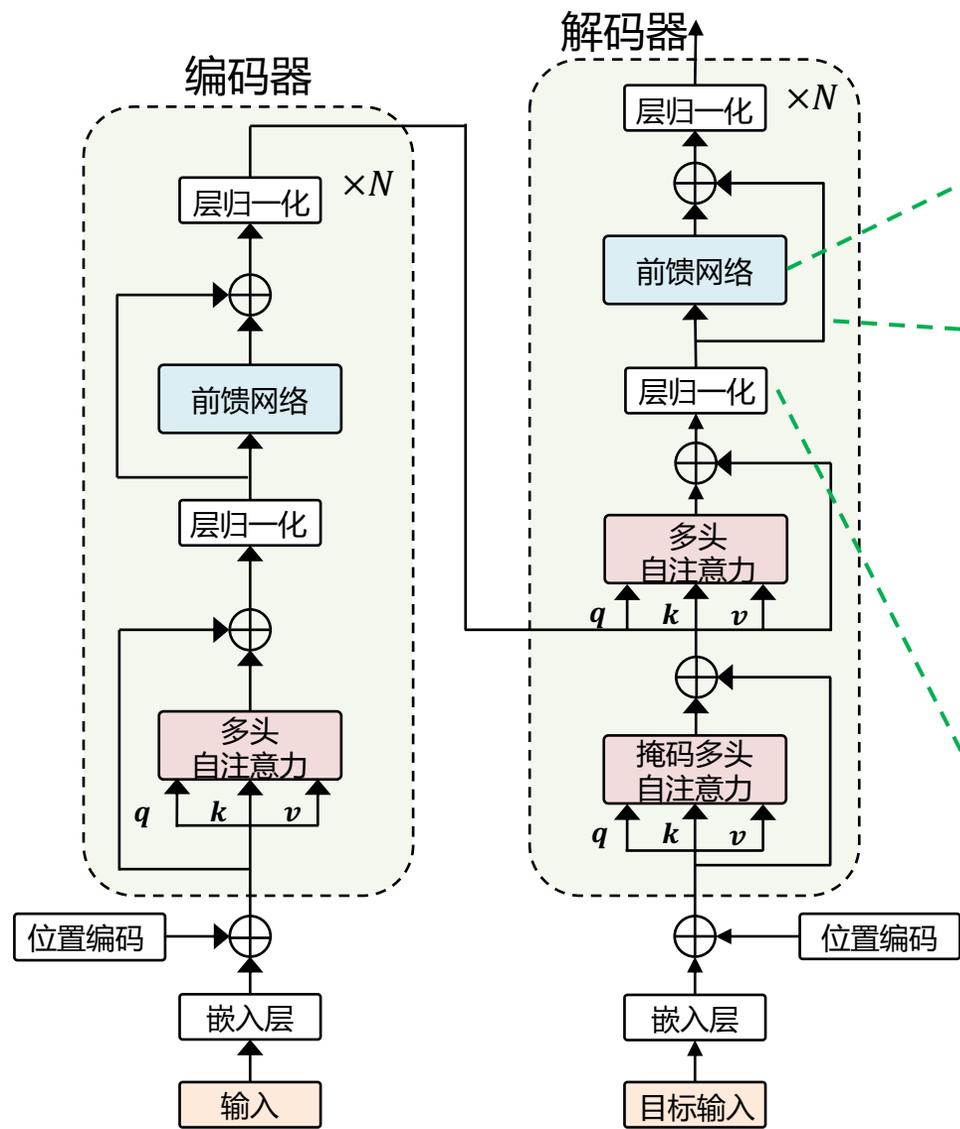
Transformer架构及其应用



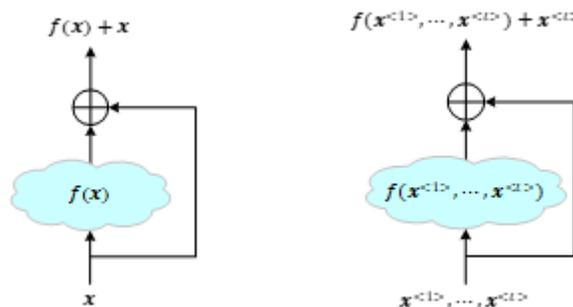
$$a(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^T \mathbf{k}}{\sqrt{d}} \quad (1)$$

$$\text{Attn}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \text{Softmax}(a(\mathbf{q}, \mathbf{k})) \otimes \mathbf{v} \quad (2)$$

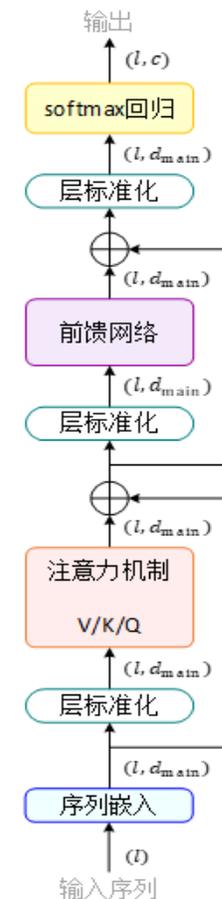
Transformer架构及其应用



- **前馈网络:** 注意力机制的输出作为前馈网络的输入，前馈网络的输出作为Softmax回归的输入
- **残差连接:** 增大偏导数的值，从而提升网络模型的训练速度



- **层归一化 (或层标准化)**
 - Layer Normalization
- **Dropout:** 防止过拟合提高泛化性

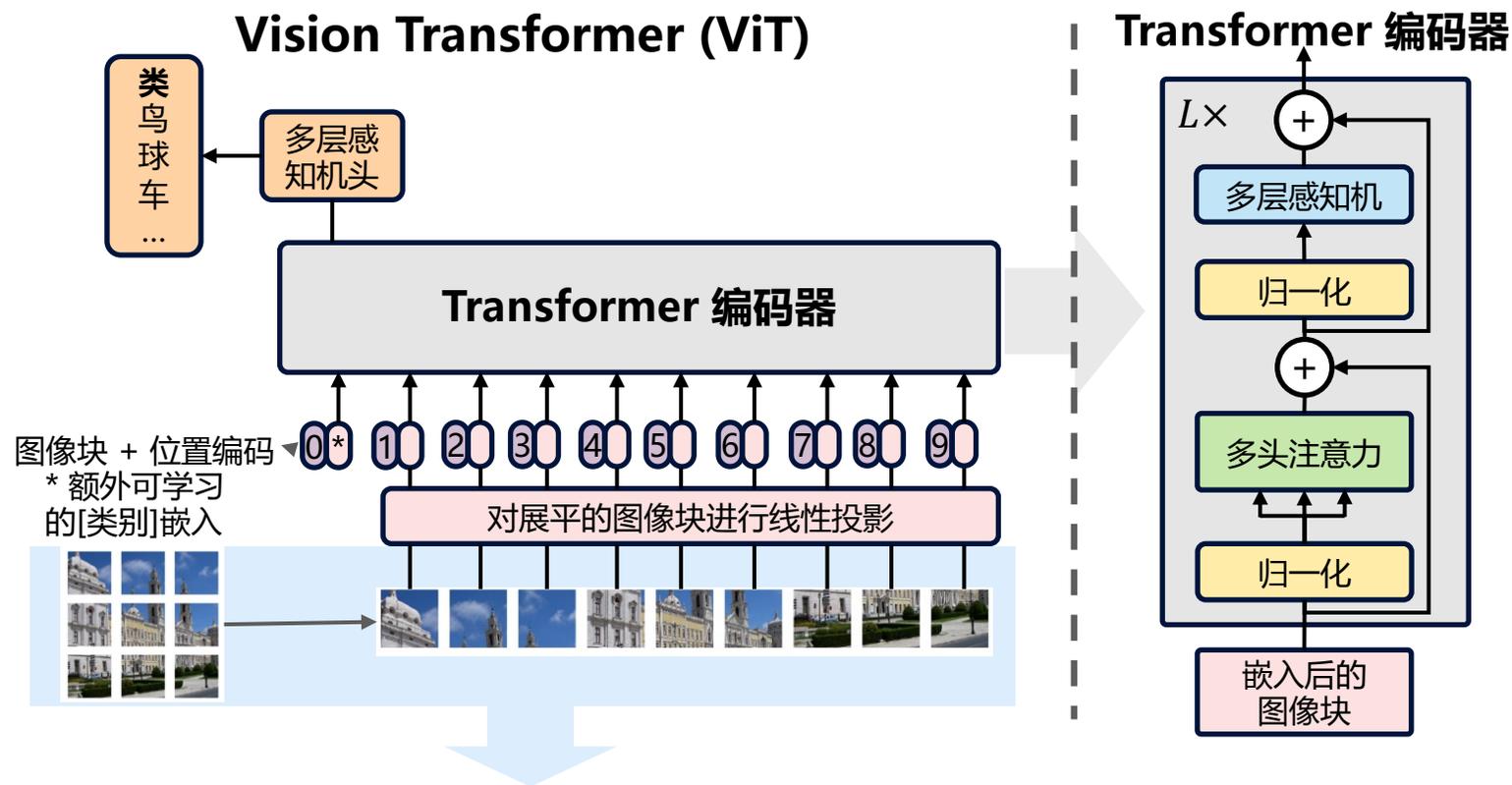


Transformer复杂度分析

模型	每层复杂度	序列操作数	最大路径长度
CNN	$O(kLd^2)$	$O(1)$	$O(\log_k(L))$
RNN	$O(Ld^2)$	$O(L)$	$O(L)$
Transformer	$O(L^2d)$	$O(1)$	$O(1)$

k 卷积核大小 L 序列长度 d 维度

Transformer在视觉领域的应用

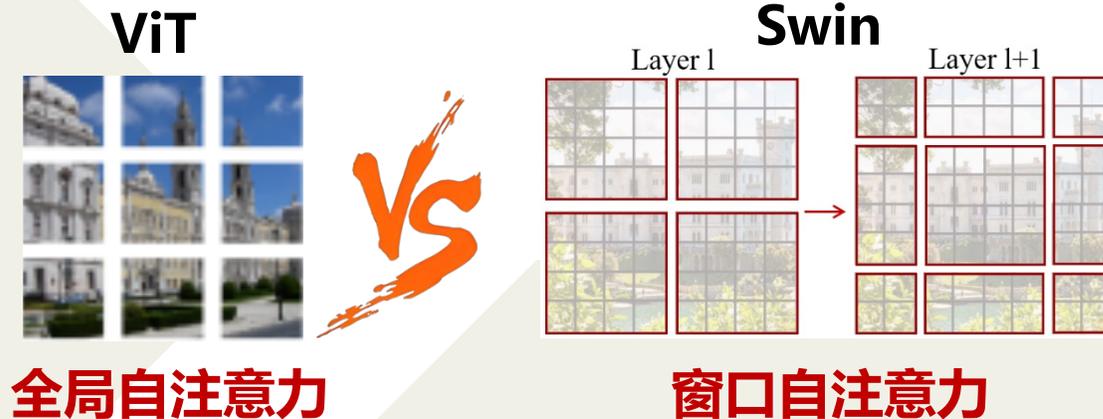


- ✓ 打破了CNN在图像处理领域的长期垄断，展示了Transformer模型的泛用性
- ✓ 展示了在大规模数据集上训练ViT的潜力
- ✓ 展现了Transformer在多任务学习中的潜力

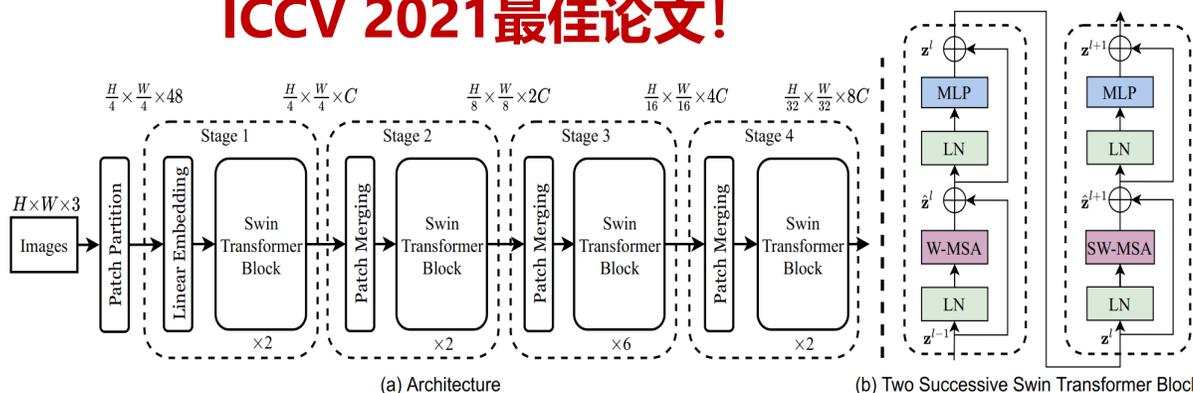
➤ 提出**图像分块 (Image Patch Embedding)** 的新方法，通过将图像分割成一系列固定大小的图像块，并将这些图像块视为**序列化的“视觉单词”或“令牌” (tokens)**

➤ Swin Transformer通过其创新的**分层金字塔结构、窗口注意力**和**移位窗口机制**，实现了更高效的多尺度特征融合，在效率和性能之间取得了平衡。

- ViT将图像分割成固定大小的非重叠小块，直接进行全局自注意力计算。在处理高分辨率图像时，计算复杂度会随着图像尺寸的增加而显著增加。
- ViT的全局自注意力机制，不同窗口之间的信息无法直接交互，这可能导致信息隔离。

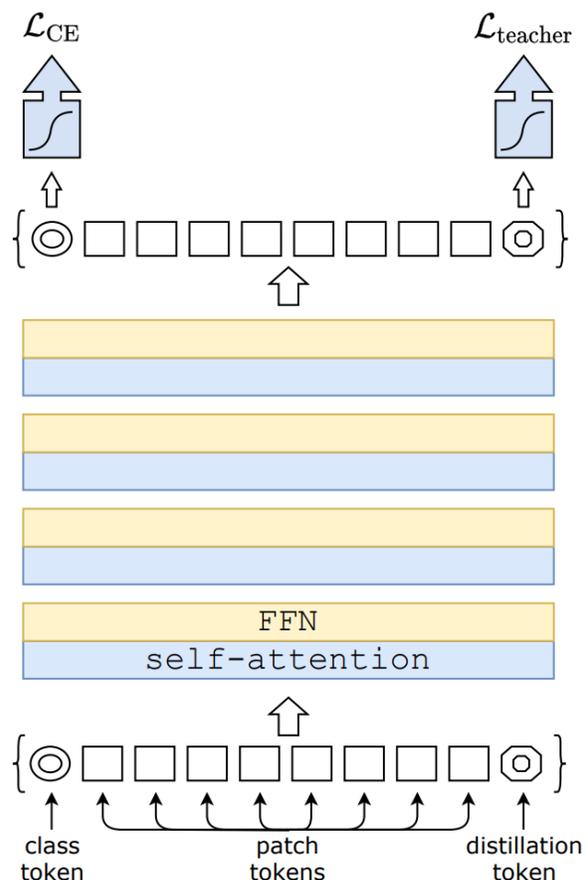
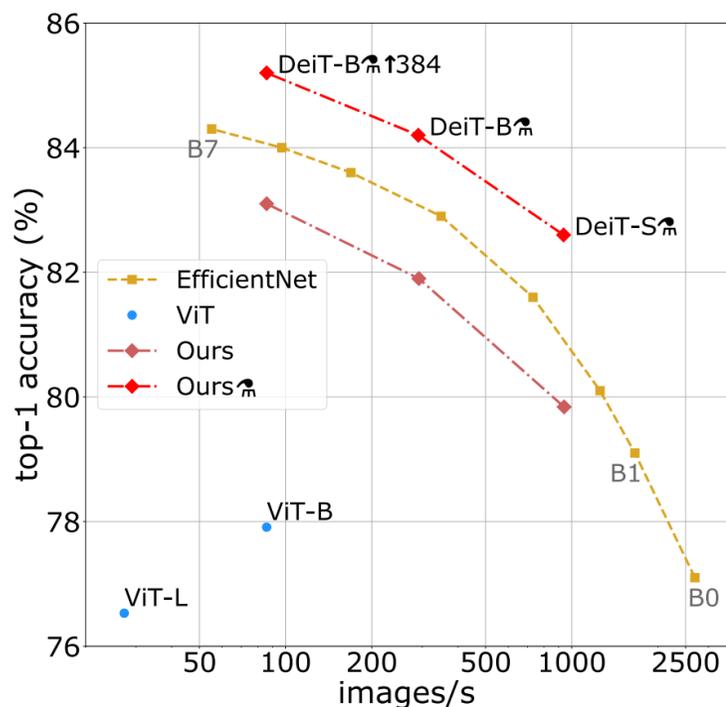


ICCV 2021最佳论文!



- 在每个窗口内进行自注意力计算。将计算复杂度降低到与窗口大小线性相关，显著减少了计算量。
- 移位窗口的设计使得相邻窗口之间能够进行信息交互。

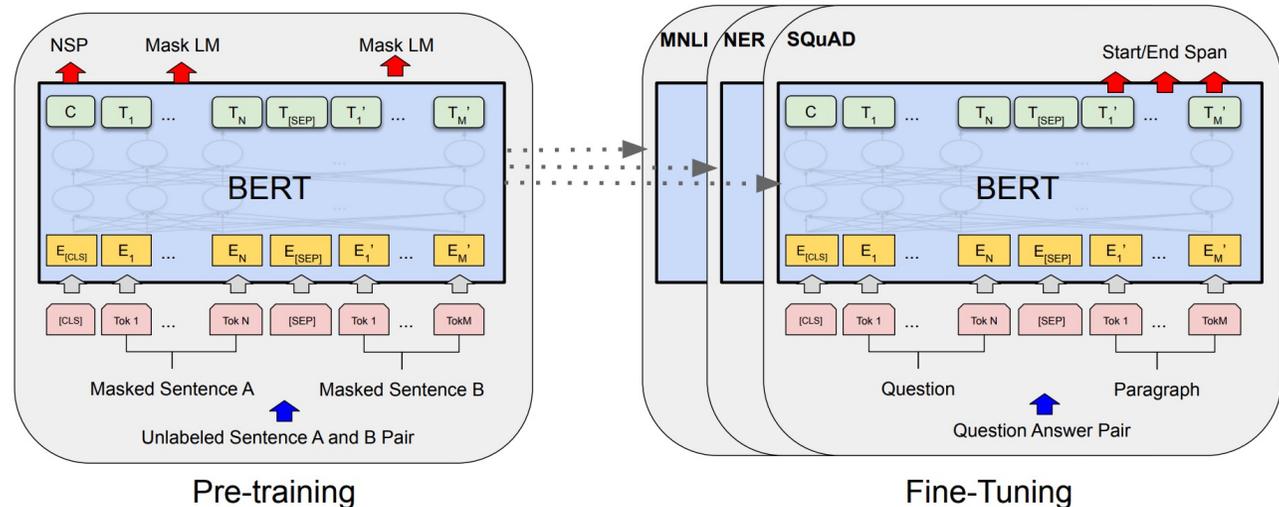
Transformer在视觉领域的应用



➤ DeiT: 数据高效性图像Transformer

- 通过知识蒸馏等技术, 让ViT能够借助一个教师CNN进行学习
- 显著降低对海量预训练数据的需求

Transformer在自然语言处理领域的应用



- **BERT: 双向编码器表示的突破**
- 预训练与微调范式
 - 采用仅编码器结构
 - 掩码语言模型 (Masked Language Modeling)
 - 下一句预测 (Next Sentence Prediction)
 - 进行预训练, 实现深层双向上下文理解

Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{##ing}$	$E_{[SEP]}$
Segment Embeddings	+	+	+	+	+	+	+	+	+	+	+
	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
Position Embeddings	+	+	+	+	+	+	+	+	+	+	+
	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

Transformer在自然语言处理领域的应用

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée
4 plush girafe => girafe peluche
5 cheese => ..... ← prompt
```

Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



➤ GPT系列 (Generative Pre-trained Transformer)

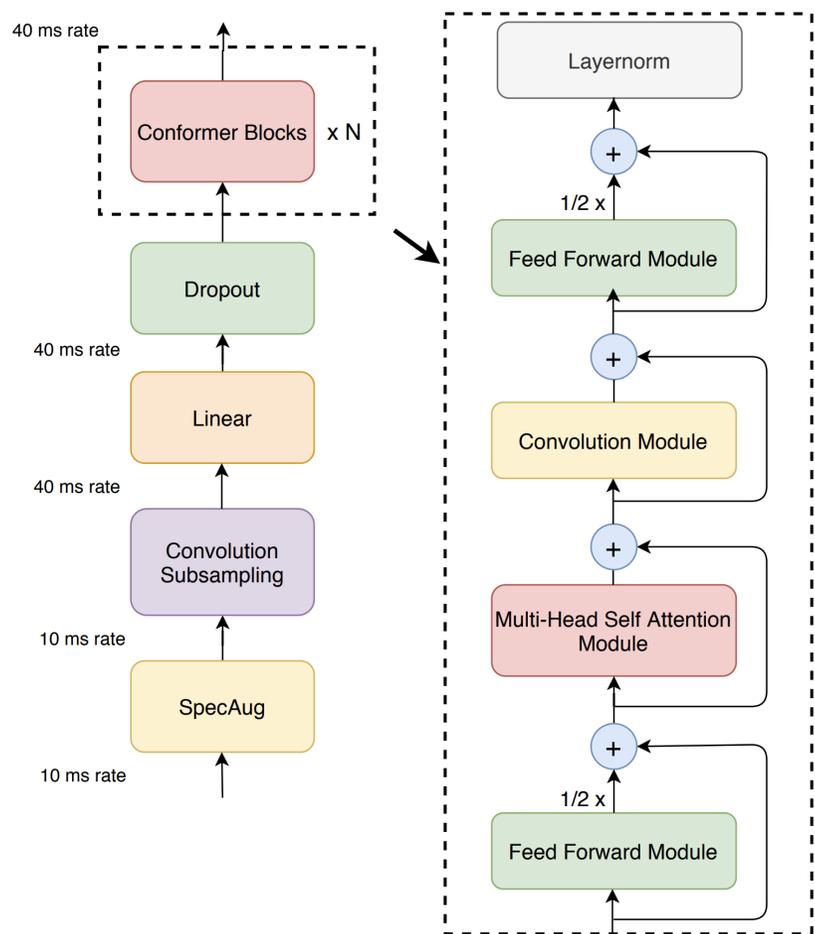
- GPT-1 生成式预训练开端
- GPT-2 提升规模与能力
- GPT-3 突破少样本学习
- 采用仅解码器结构
- 自回归语言建模预训练
- 专注于生成连贯的文本

GPT-1: Improving Language Understanding by Generative Pre-Training

GPT-2: Language Models are Unsupervised Multitask Learners

GPT-3: Language Models are Few-Shot Learners

Transformer在语音识别领域的应用

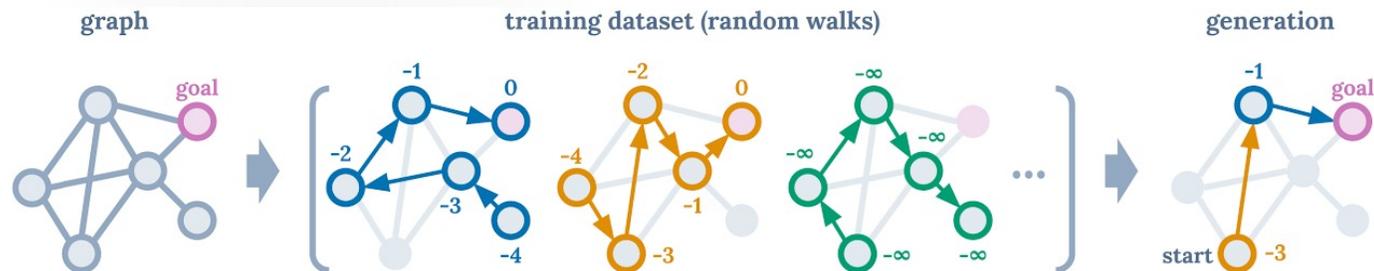


➤ Conformer: 卷积+Transformer

- 提出卷积前馈网络
- Transformer的前馈网络中嵌入深度可分离卷积
- 卷积擅长捕捉局部上下文（如音素）
- 自注意力擅长捕捉全局依赖（如语义）
- “没有一种结构是万能的”

Transformer在强化学习领域的应用

... \hat{R}_{t-1}



➤ Decision Transformer

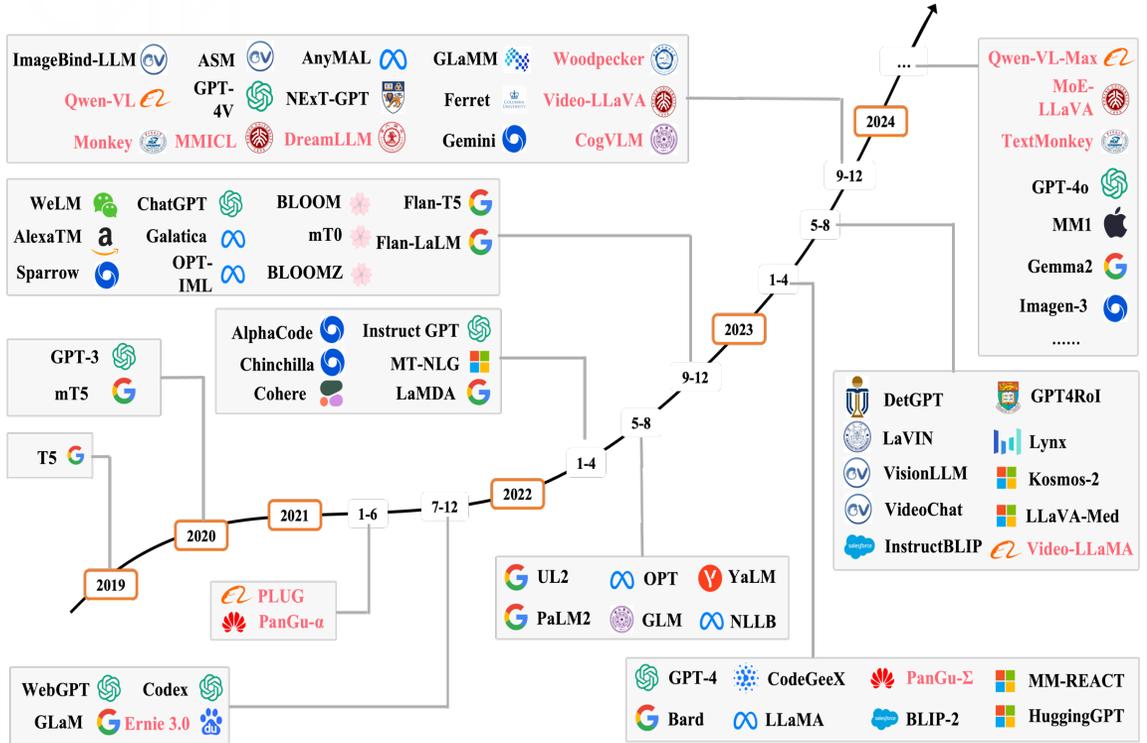
- 范式转变：将强化学习重构为序列建模问题
- 输入：过去的状态、动作和回报到目标的序列
- 直接通过Transformer的自回归方式预测下一个最优动作

Transformer的优势

RNN
CNN

VS

Transformer



- ✓ 强大的并行计算能力
- ✓ 长期依赖建模能力
- ✓ 强大的特征抽取能力
- ✓ 可扩展性和预训练能力
- ✓ 跨模态应用的广泛适应性



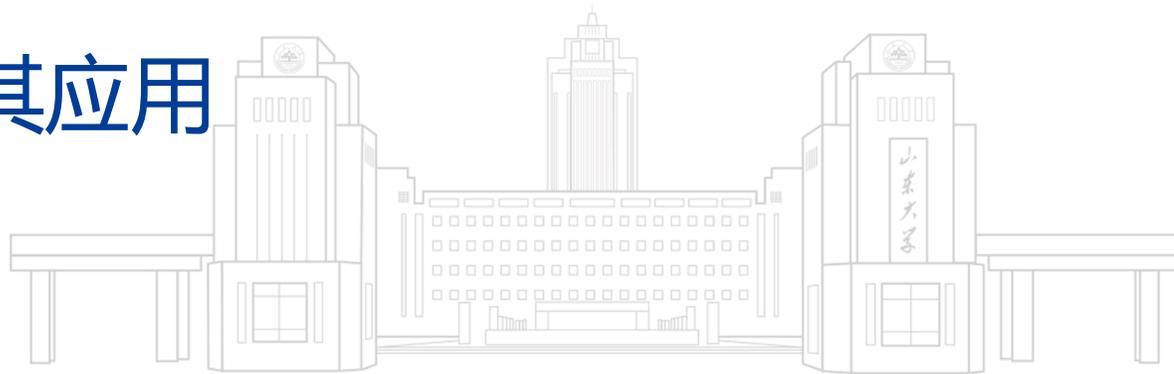
章节目录

CONTENTS

01 | 注意力机制

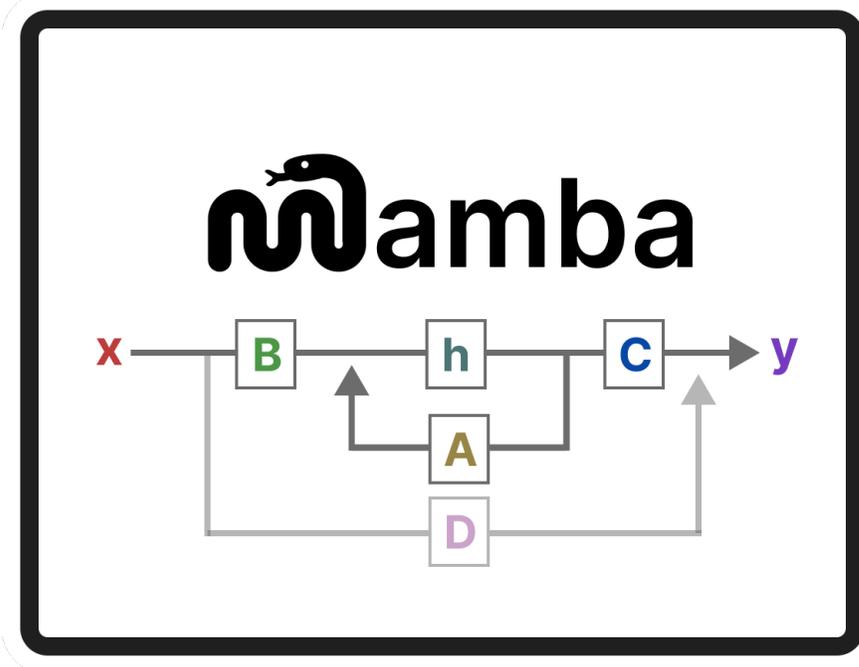
02 | Transformer架构及其应用

03 | Mamba架构及其应用



什么是Mamba

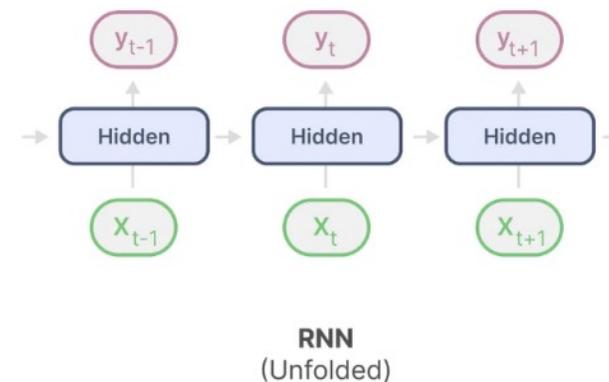
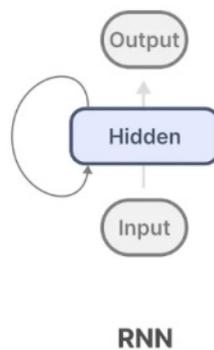
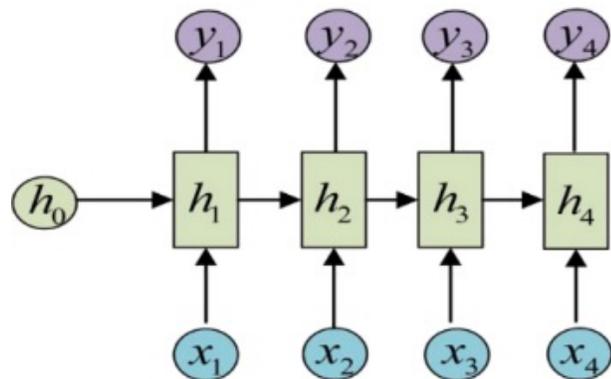
- 一种新的深度学习模型，适用于语言建模及其他序列建模任务
- 通过**选择性状态空间模型** (State Space Models, SSMs) 来改进传统的状态空间模型
- **Mamba: Linear-Time Sequence Modeling with Selective State Spaces**
- Albert Gu (CMU), Tri Dao (Princeton)
- CoLM 2024 (ICLR 2024 拒稿)



为什么需要SSM

➤ RNN的优劣势

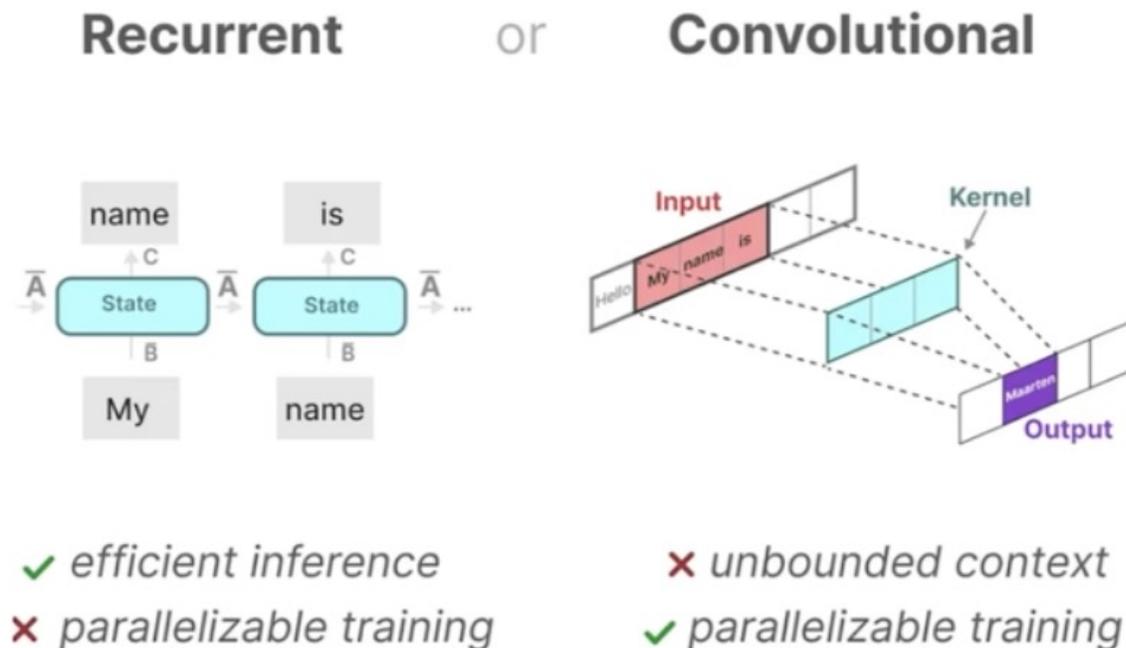
- 线性空间占用，串行导致**训练耗时长**，但仍可以快速执行推理
- **遗忘问题严重**，无法有效处理长程依赖关系：即使有了LSTM与GRU等门控设计，也仅仅是延缓遗忘问题的发生



为什么需要SSM

➤ CNN的优劣势

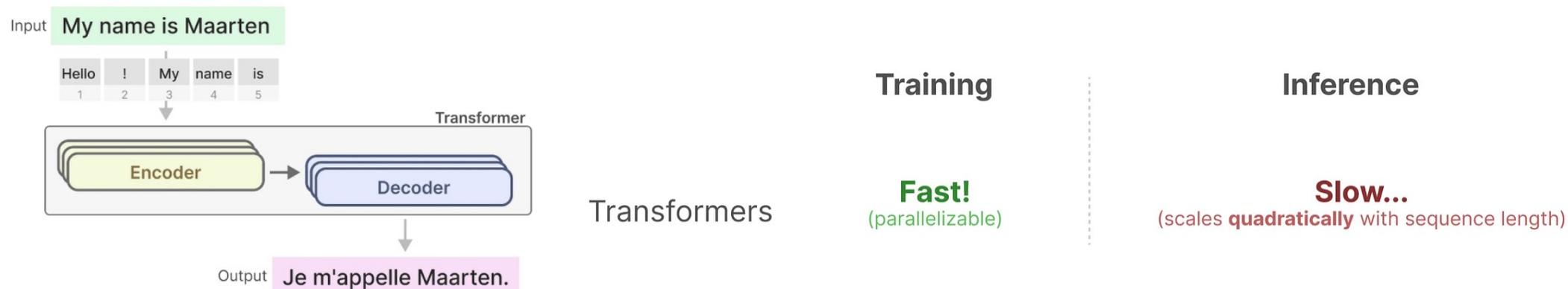
- 线性空间占用，可并行训练
- 仅关注局部特征，容易忽视全局特征
- 无推理方面的优化，固定的卷积核限制了推理速度



为什么需要SSM

► Transformer的优劣势

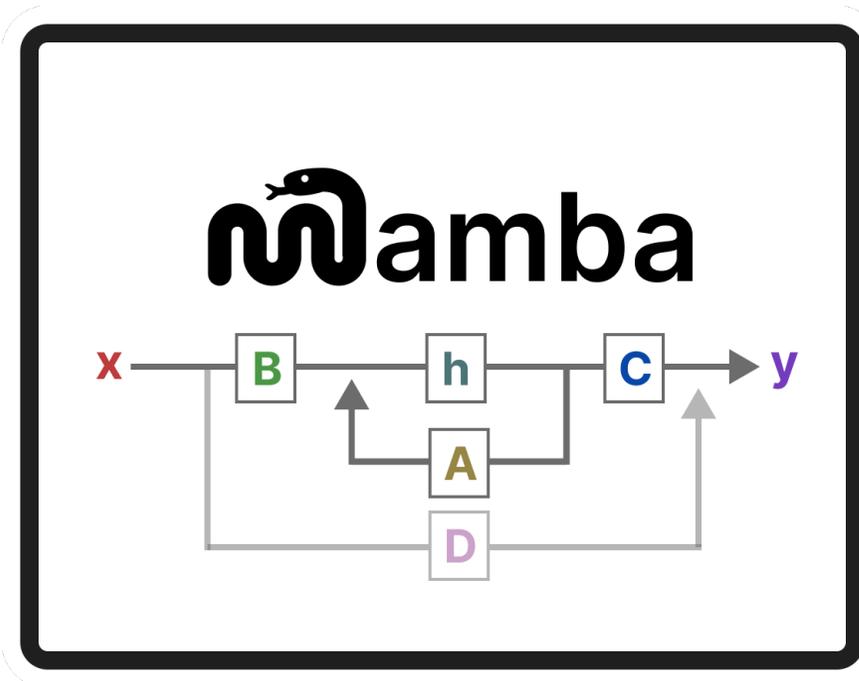
- 可并行训练，良好地捕捉长程依赖关系
- 过大的空间复杂度与时间复杂度
- 复杂的注意力分数计算过程导致推理速度慢



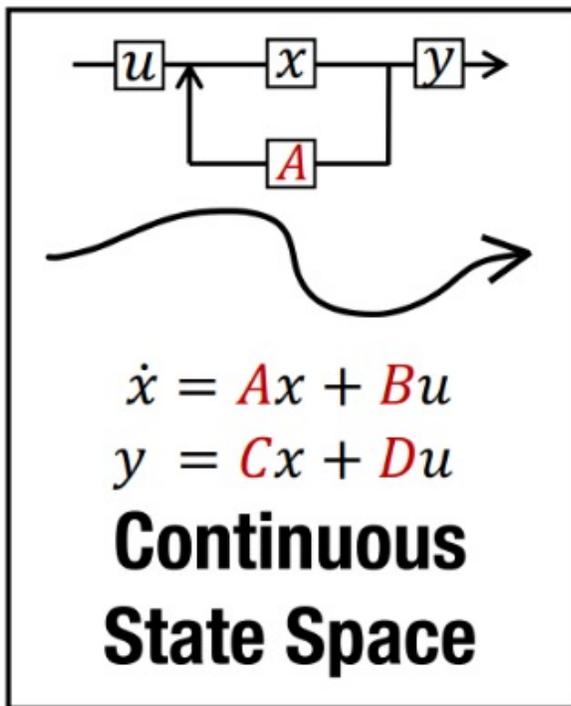
为什么需要SSM

➤结合Transformer与RNN优点的SSM

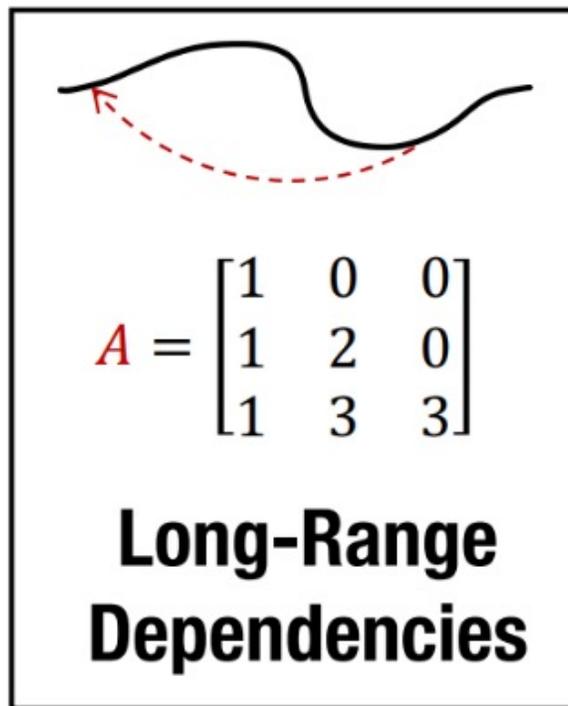
- Transformer并行训练+RNN逐步处理（仅关注前一个Token）
- 核心思想：独特的状态空间设计
- 处理每个新Token时动态更新和维护一个内部状态，此状态不仅编码了之前所有Token信息，还能根据当前处理需求进行选择性的信息整合



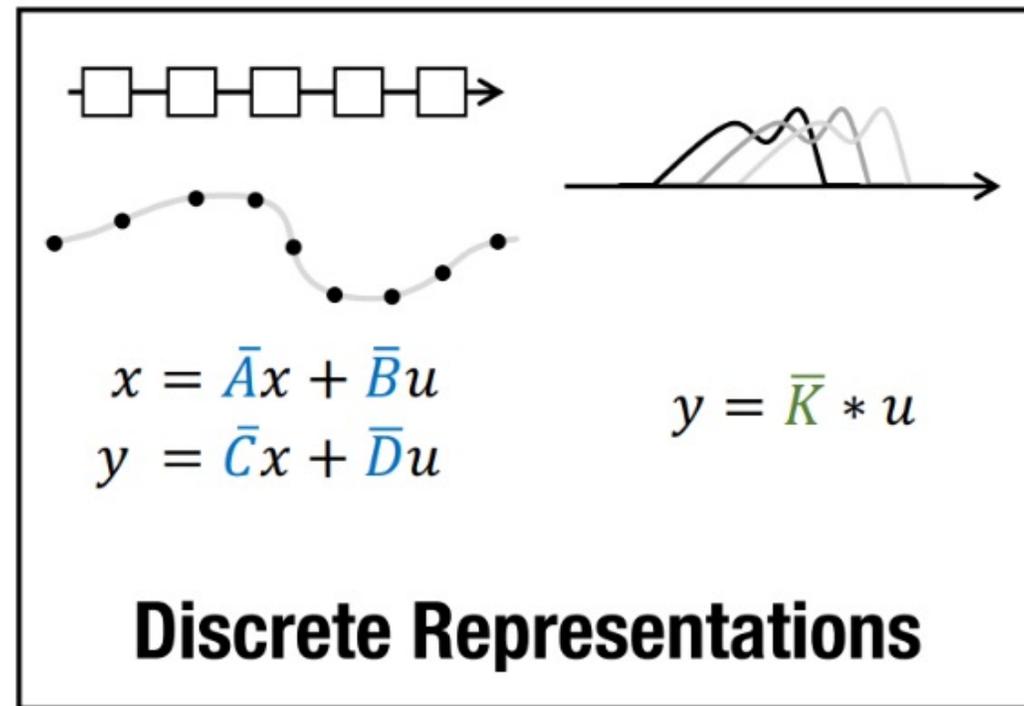
Mamba的基础组件



状态空间模型



长距依赖 (初始化方式)



离散化表达

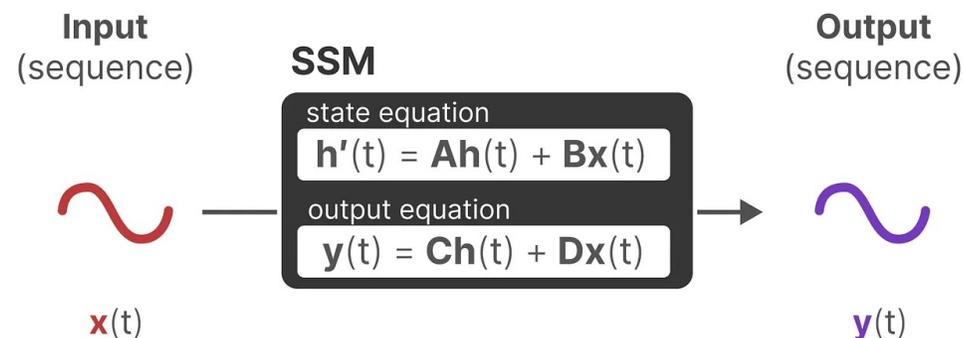
Mamba的基础1：状态空间模型 (SSM)

➤ 1. 基本定义 (源于现代控制系统理论)

- 用于描述序列在各时间步的状态表示，并根据输入预测其下一个状态的模型
- 输入序列 $x(t)$ 、隐状态表示 $h(t)$ 、预测输出序列 $y(t)$

➤ 2. 模型公式 (连续函数)

- 状态方程: $h'(t) = Ah(t) + Bx(t)$
- 输出方程: $y(t) = Ch(t) + Dx(t)$



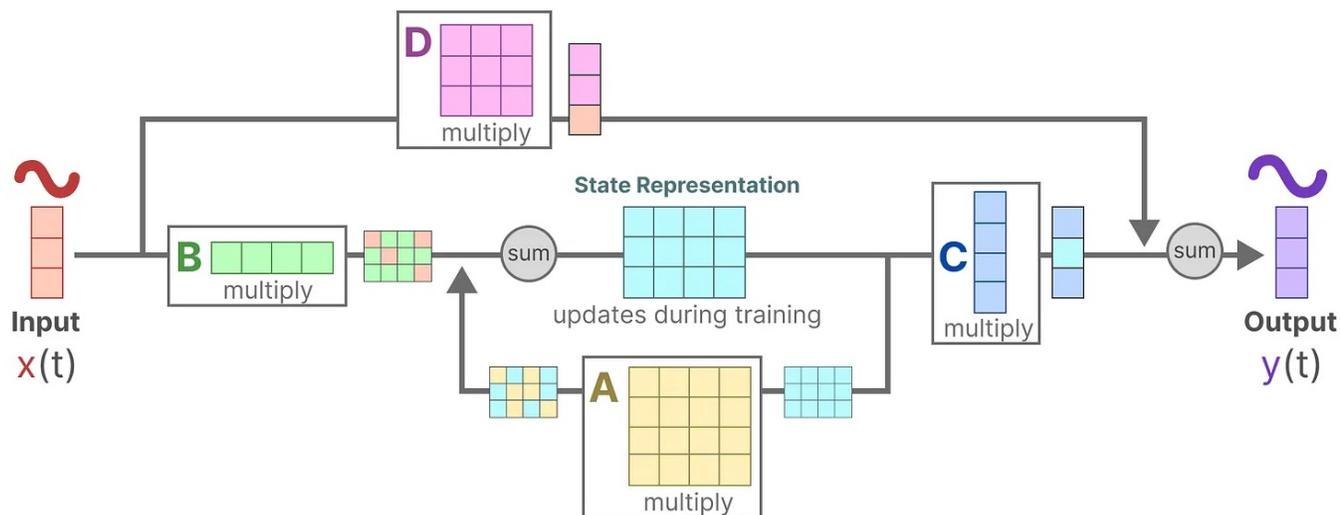
Mamba的基础1：状态空间模型 (SSM)

➤ 2. 模型公式

- **状态转移矩阵A**：描述无外部输入时，系统状态如何随时间自然演变
- **输入矩阵B**：描述外部输入如何影响系统状态，建立输入信号与系统

状态之间的直接联系

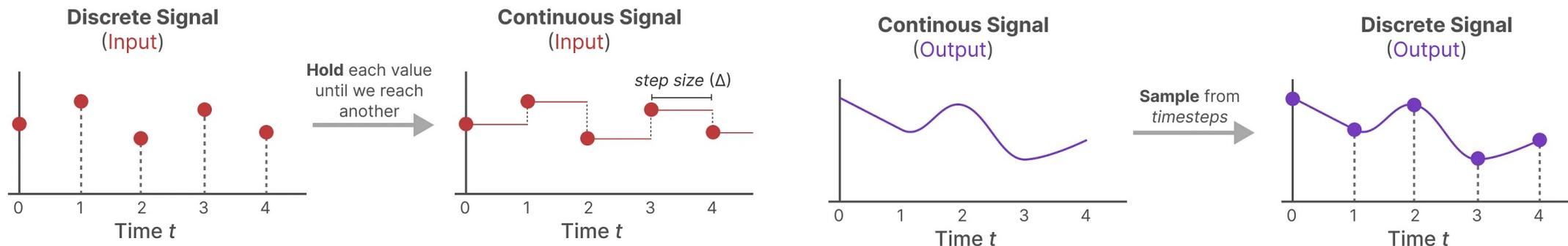
- **输出矩阵C**
- **直接传输矩阵D**



Mamba的基础2：连续信号转换为离散信号

➤ 1. 连续函数离散化：零阶保持技术 (Zero-Order Hold, ZOH)

- 工作原理：每当系统接收到一个离散信号时，此技术会记录下这个信号的值，并将其保持不变，直到下一个离散信号的到来
- 离散信号连续化之后就有了连续的输入信号，连续信号可以生成连续的输出，并且仅根据输入的时间步长对输出值进行采样

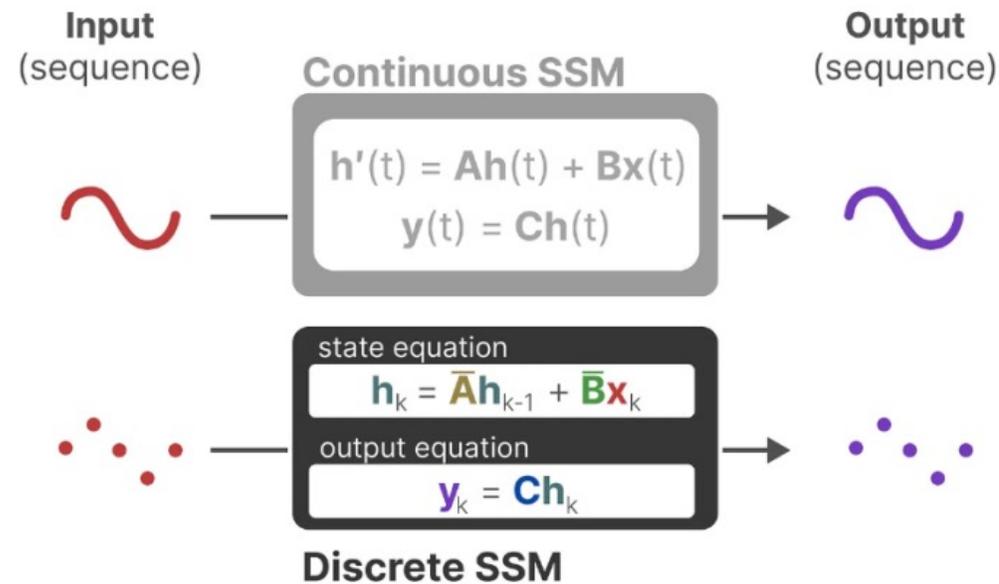
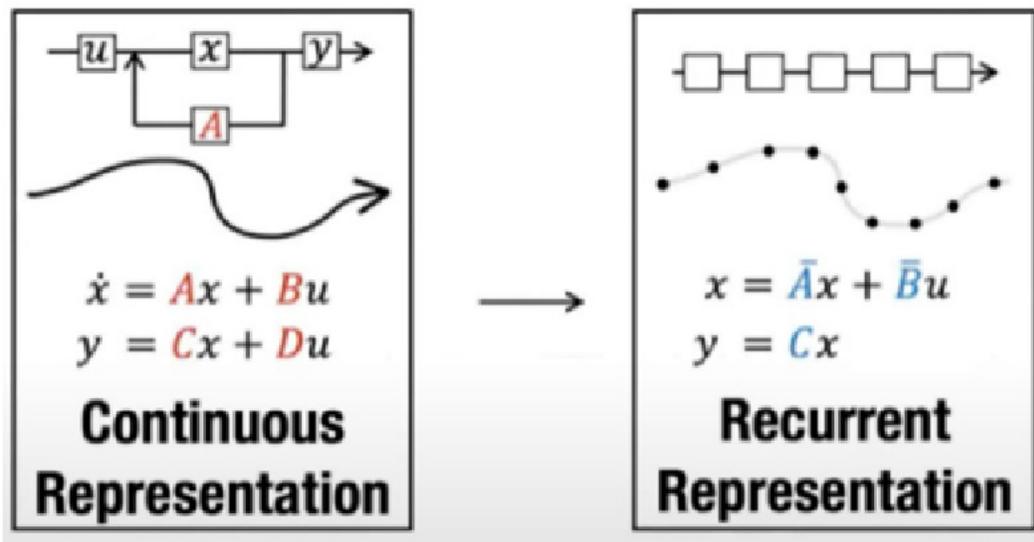


Mamba的基础2: 连续信号转换为离散信号

➤ 1. 连续函数离散化: 零阶保持技术 (Zero-Order Hold, ZOH)

- $h_t = \bar{A}h_{t-1} + \bar{B}x_t, h_0 = \bar{B}x_0$

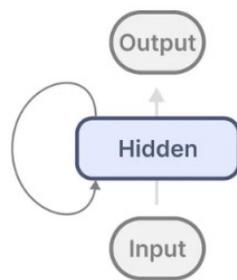
- $y_t = \bar{C}h_t$



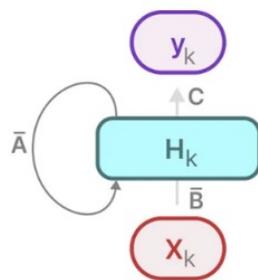
Mamba的基础2：连续信号转换为离散信号

➤ 2. 离散状态空间的循环计算（类似于RNN的计算方法）

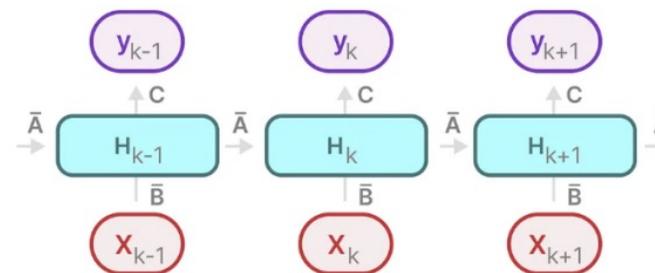
- 状态空间模型在应用时可以转换为类似于RNN的计算方法
- 离散状态空间模型将连续的信号或系统动态划分为一系列的时间步长，并在这些时间步长上构建模型



RNN



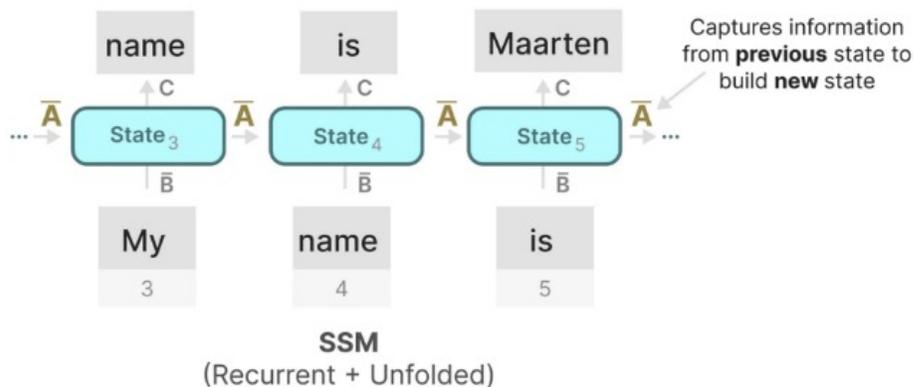
SSM
(Recurrent)



SSM
(Recurrent + Unfolded)

Mamba的基础3: HiPPO算法初始化的状态转移矩阵

- SSM的问题: 与CNN/RNN类似, 长期依赖捕捉能力弱
- 状态转移矩阵的重要性: 通过捕获有关先前状态信息来构建新状态
- 如何构建一个参数量足够的状态转移矩阵来存储与压缩历史信息?
- 解决方案: HiPPO矩阵代替随机初始化的状态转移矩阵 \bar{A}



Produces hidden state

$$h_k = \bar{A}h_{k-1} + \bar{B}x_k$$

$$y_k = Ch_k$$

Mamba的基础3: HiPPO算法初始化的状态转移矩阵

- HiPPO (High-order Polynomial Projection Operator)
- 相比于随机初始化状态转移矩阵, HiPPO可以有效缓解遗忘问题

HiPPO Matrix

1	0	0	0
1	2	0	0
1	3	3	0
1	3	5	4

n

k

HiPPO Matrix

\mathbf{A}_{nk}

$(2n + 1)^{1/2} (2k + 1)^{1/2}$ ← everything **below** the diagonal

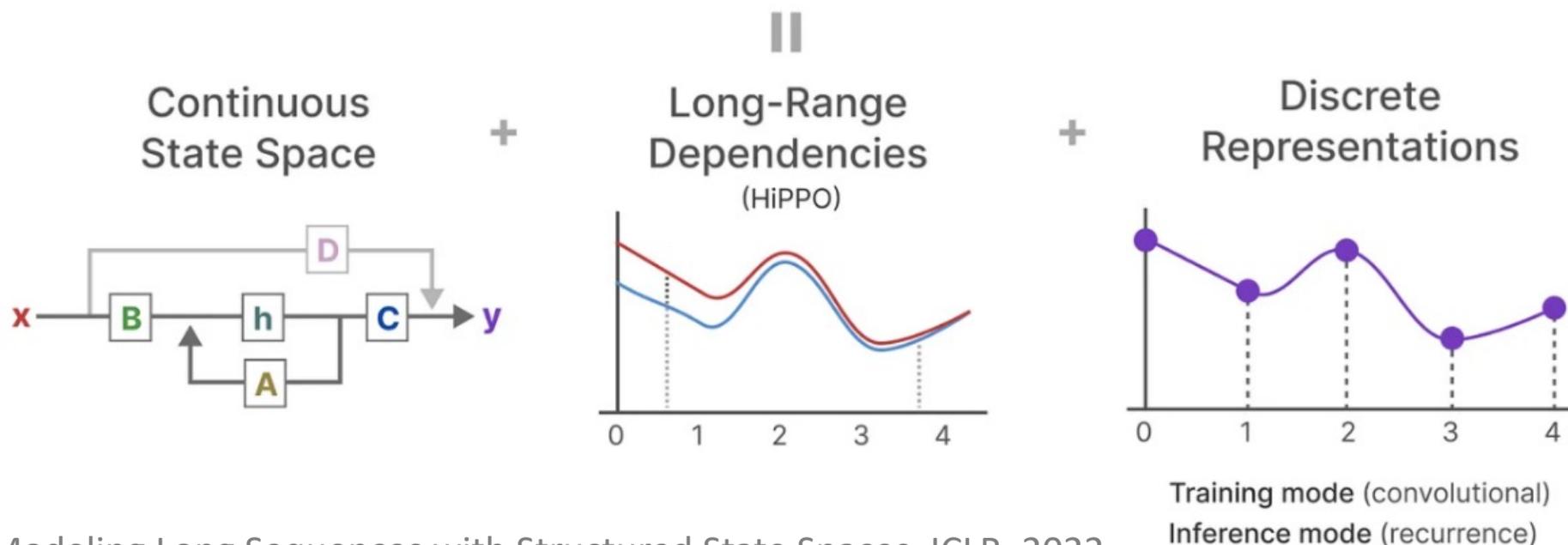
$n + 1$ ← the diagonal

0 ← everything **above** the diagonal

序列的结构化状态空间 (S4)

状态空间模型 (SSM) + HiPPO算法 + 离散化技术

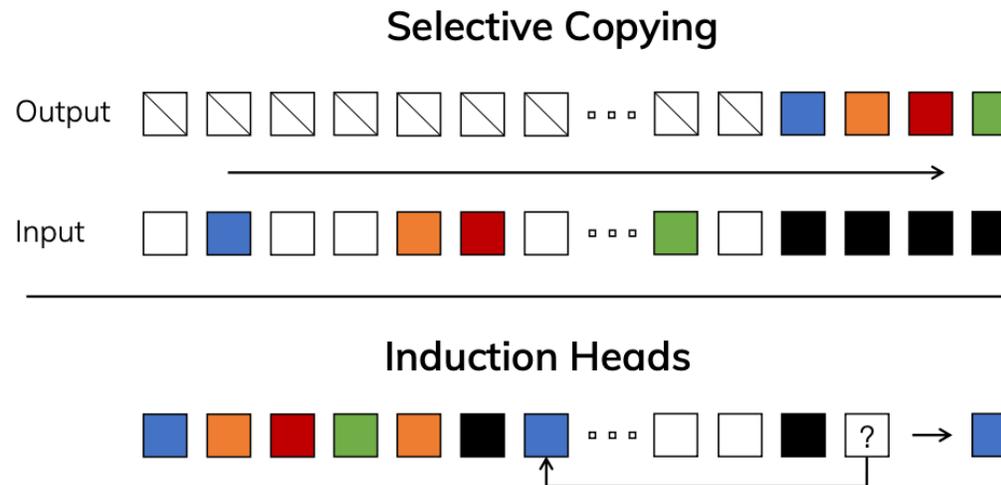
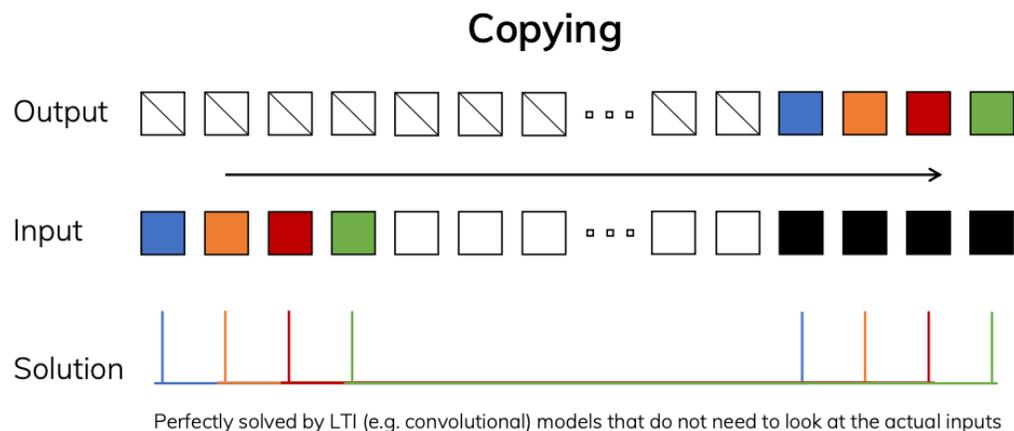
Structured State Spaces for Sequences (S4)



Mamba的设计动机

➤ SSM/S4问题：无选择性

- A 、 B 、 C 矩阵不随输入不同而变化，无法针对输入针对性推理
- 一旦训练完成，无论输入如何，都会产生完全一样的 A 矩阵



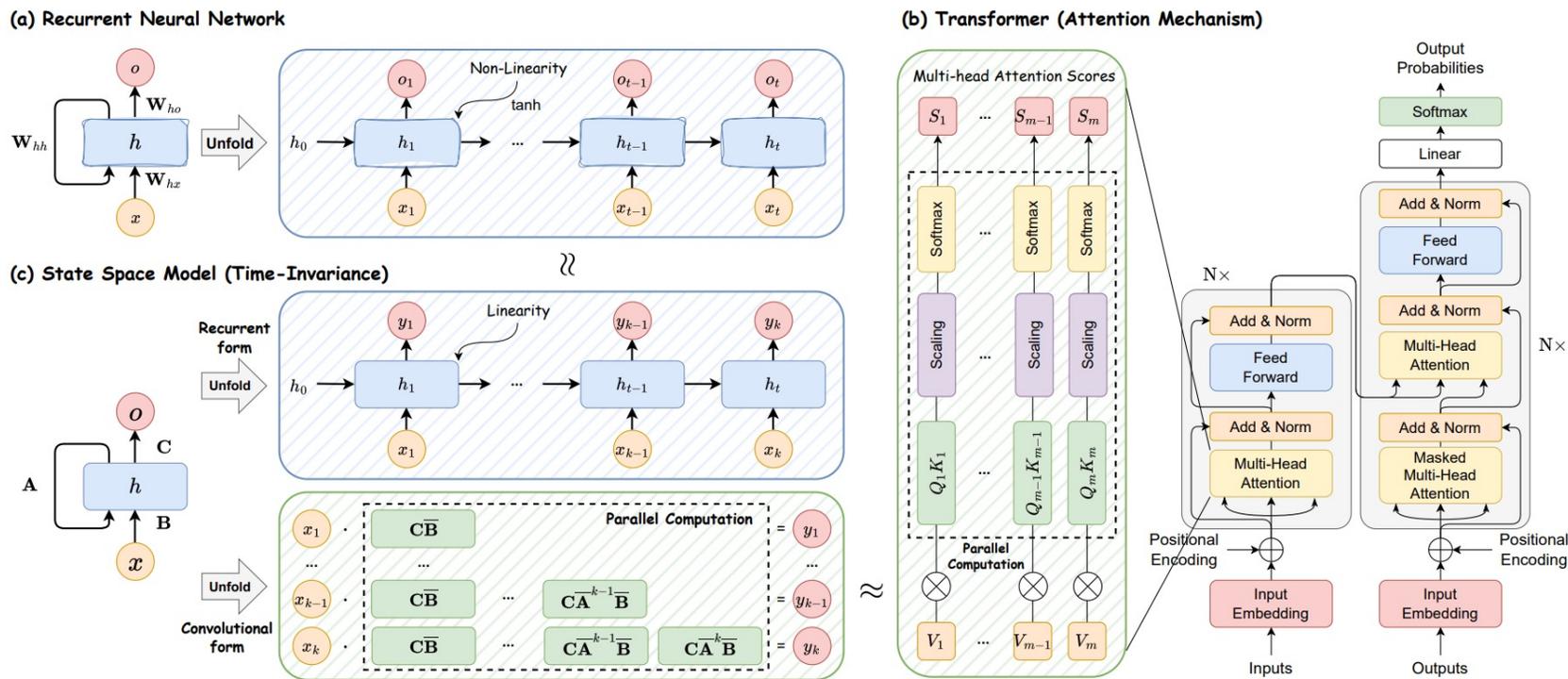
Mamba的三大创新

- **Mamba = 有选择处理信息 + 硬件感知算法 + 更简单的SSM架构**
 1. 参数化矩阵：对输入信息有选择性处理
 2. 硬件感知的并行运算算法
 - 采用“并行扫描算法”而非“卷积”来进行模型的循环计算
 3. 更简单的SSM架构
 - 将SSM架构设计与Transformer的MLP块合并为一个块来简化过去的深度序列模型架构

Mamba与其他模型的对比

模型	对信息的压缩程度	训练的效率	推理的效率
transformer(注意力机制)	transformer对每个历史记录都不压缩	训练消耗算力大	推理消耗算力大
RNN	随着时间的推移, RNN 往往会忘记某一部分信息	RNN没法并行训练	推理时只看一个时间步 故推理高效 (相当于推理快但训练慢)
CNN		训练效率高, 可并行「因为能够绕过状态计算, 并实现仅包含(B, L, D)的卷积核」	
SSM	SSM压缩每一个历史记录		矩阵不因输入不同而不同, 无法针对输入做针对性推理
mamba	选择性的关注必须关注的、过滤掉可以忽略的	mamba每次参考前面所有内容的一个概括, 兼备训练、推理的效率	

Mamba与其他模型的对比



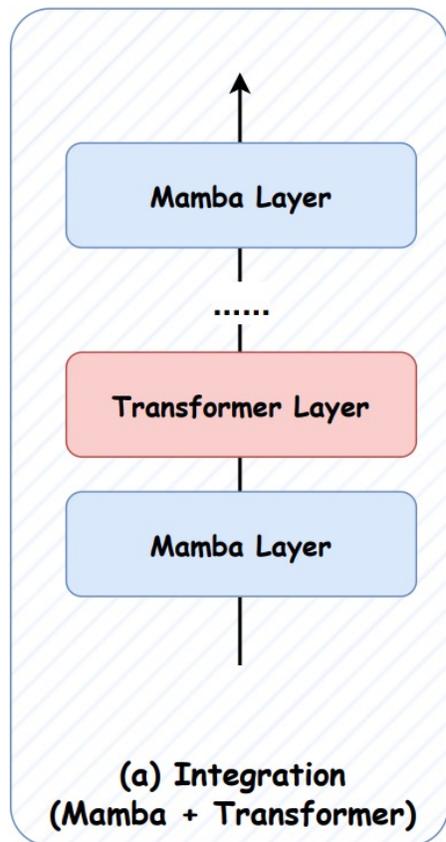
Comparison	RNNs	Transformers	SSMs
Training Speed	Slow (Recurrent)	Fast (Parallel)	Fast (Convolutional)
Inference Speed	Fast (Recurrent)	Slow (Quadratic-Time)	Fast (Recurrent)
Complexity	$O(LD^2)$	$O(L^2D)$	$O(LD^2)$
Modeling Capabilities	● (Hidden State)	● (Attention)	● (Time-Invariance)

Mamba的优势

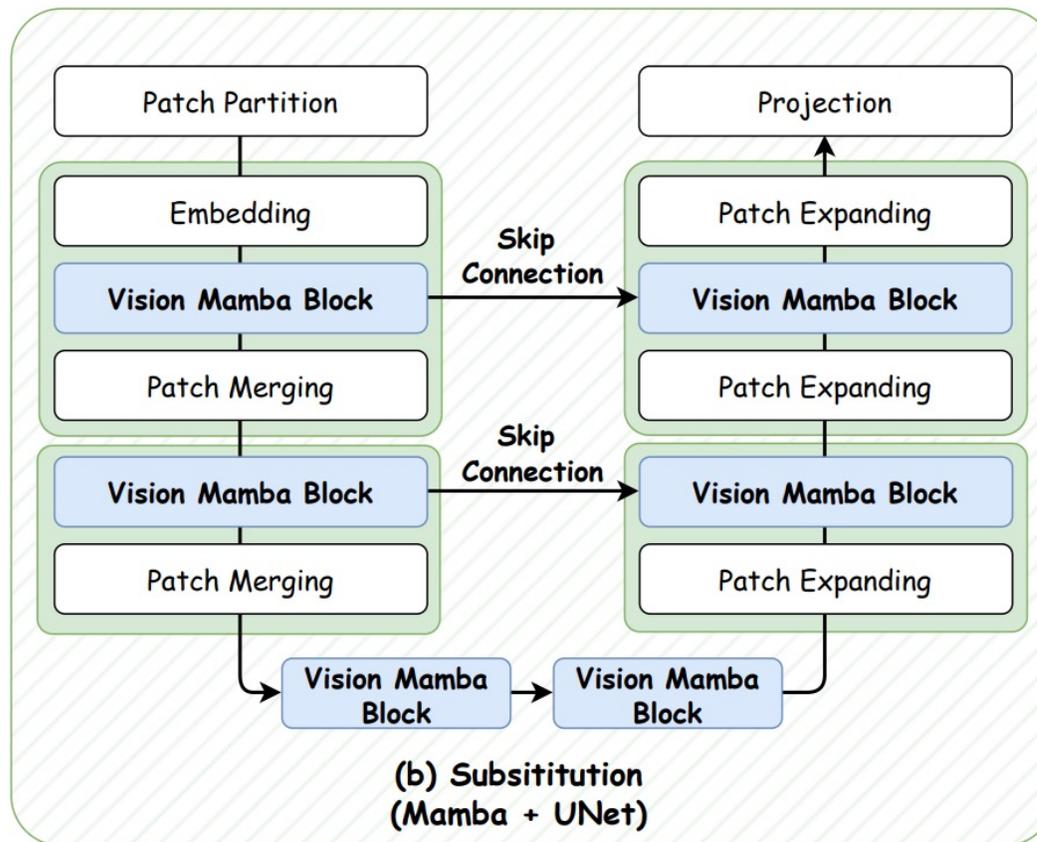
➤ Mamba的主要特点总结如下

- **选择性SSM**: 利用选择性状态空间模型过滤无关信息, 专注于相关数据, 从而增强其在序列处理中的能力
- **硬件感知算法**: 采用针对现代硬件尤其是GPU优化的并行算法, 显著提高了计算速度, 并减少了内存需求
- **简化架构**: 通过集成选择性SSM并去除注意力机制和MLP模块, 提供了更简洁、更均匀的结构, 带来了更好的可扩展性

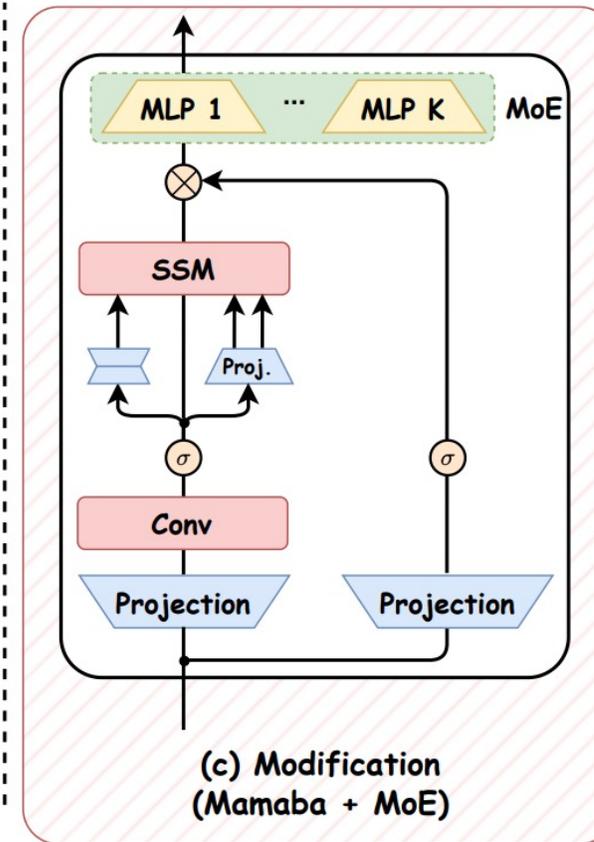
Mamba的改进方向



集成



替代



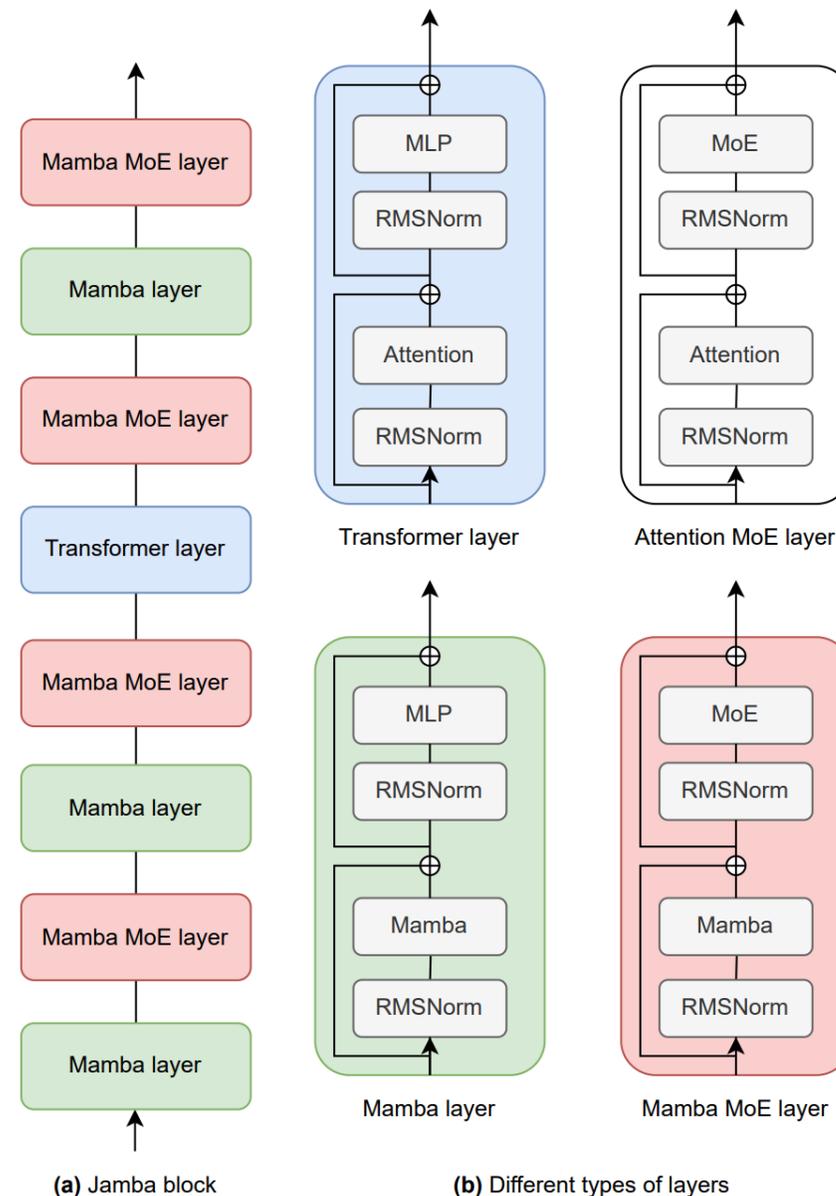
修改

Mamba改进示例: Jamba

- 混合Transformer与Mamba两种模型
- 引入混合专家 (MoE) 架构



Jamba: Hybrid Transformer-Mamba Language Models, ICLR, 2025



Mamba的下游应用

