

# 人工智能概论

Introduction to Artificial Intelligence

## 第五章 神经网络进阶



1

# 课程回顾

## 数据收集

- 公开数据集
- 自建数据集：爬虫、人工标注...

## 数据处理

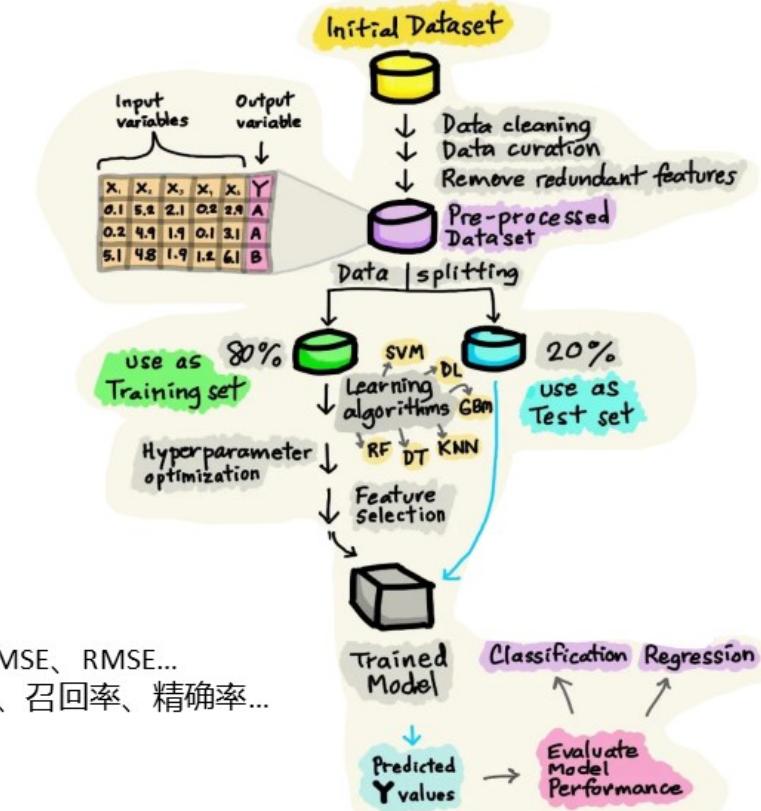
- 数据预处理
- 数据分割：训练集、测试集

## 模型训练 训练集

- 模型选择：CNN、RNN
- 损失函数、优化器设置
- 梯度下降

## 模型预测 测试集

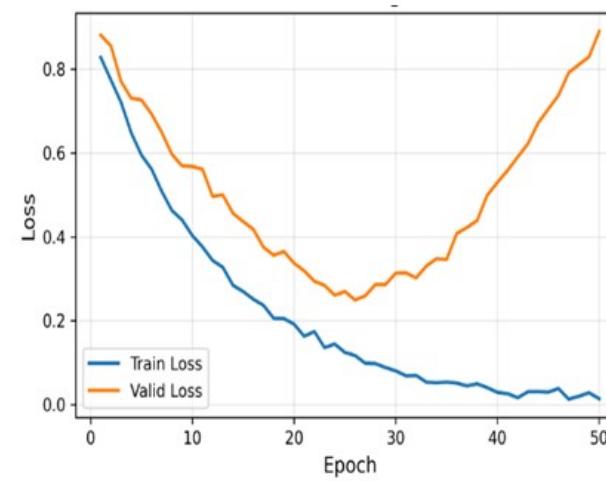
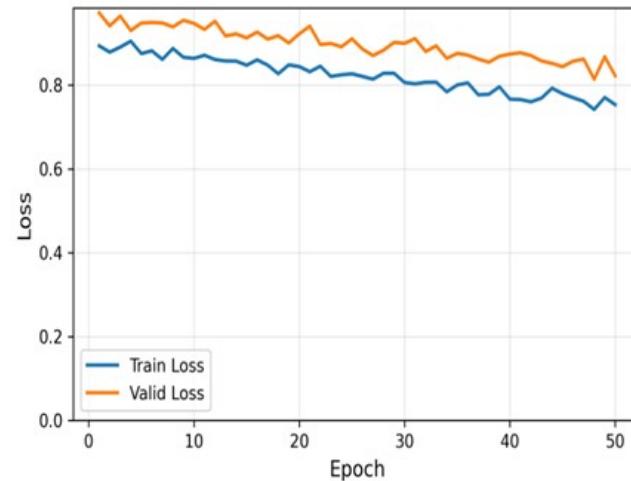
- 分类：MAE、MSE、RMSE...
- 回归：准确率、召回率、精确率...
- 神经网络



## 填空题 4分

(可多选) 结合前面学习知识, 图1可能的问题是 [填空1], 应对方法有 [填空2], 图2可能的问题是 [填空3], 应对方法有 [填空4]。

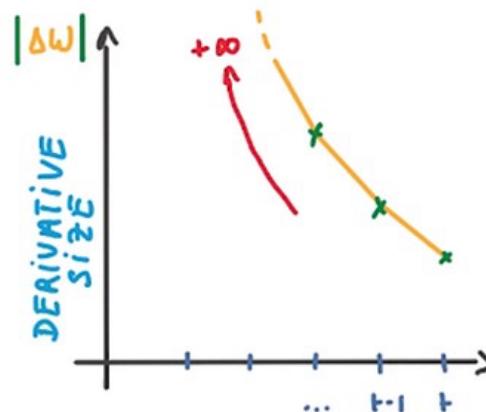
- A. 欠拟合
- B. 过拟合
- C. Dropout
- D. 使用更复杂模型
- E. 数据增广



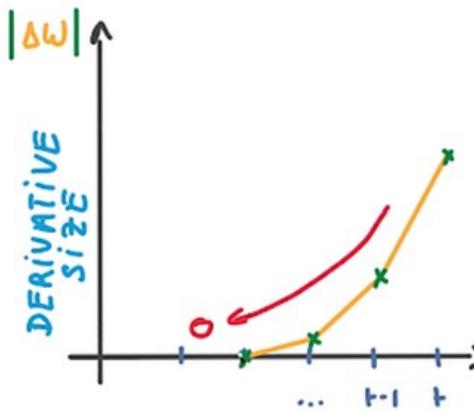
填空题 2分

结合前面学习知识，左图（红线代表更新方向）描述的问题是 [填空1]，  
右图（红线代表更新方向）描述的问题是 [填空2]

A. 梯度爆炸



B. 梯度消失



**提示：**

- 在反向传播中，若每层的梯度系数小于 1，层数越多梯度越趋近于 0，产生梯度消失；
- 若每层的梯度系数大于 1，层数越多梯度越趋近于无穷大，产生梯度爆炸。
- 处理梯度爆炸的最根本方法是参数裁剪或梯度裁剪
- 但是梯度消失的处理在当时一直是一个难题

## 填空题 4分

### 分类模型评价指标

#### ➤ 准确率 (Accuracy)

- 指所有分类（无论是正类还是负类）正确分类的比例，理想场景1.0

#### ➤ 精确率 (Precision)

- 在模型识别为正类的样本中，真正为正类的样本所占的比例。

精确率越高，模型对负样本的区分能力越强

#### ➤ 召回率 (Recall)

- 模型正确认别出为正类的样本的数量占总的正类样本数量的比值。

召回率越高，模型对正样本的识别能力越强

#### ➤ F1分数 F1-Score

- 精确率和召回率的调和平均数。

混淆矩阵		真实值	
		Positive	Negative
预测值	Positive	TP	FP
	Negative	FN	TN

混淆矩阵		真实值	
		猫	不是猫
预测值	猫	10	3
	不是猫	8	45

准确率= [填空1] (示例: m/n)

精确率= [填空2]

召回率= [填空3]

F1分数= [填空4]

5

# 人工智能概论

Introduction to Artificial Intelligence

## 第五章 神经网络进阶



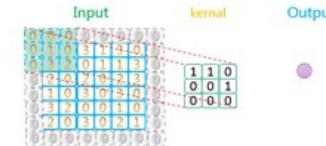
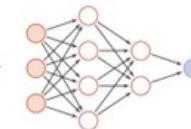
# 深度学习模型

神经  
网络  
模  
型



前馈神经网络

无时序上依赖关系的数据：图像，表格数据  
**多层感知机，CNN**

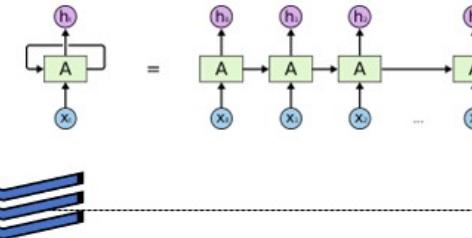


反  
馈  
神  
经  
网  
络



反馈神经网络

序列数据：文本、语音  
**RNN**



网络结构

关注模型的内部结  
构和信息流动方式



AE, VAE

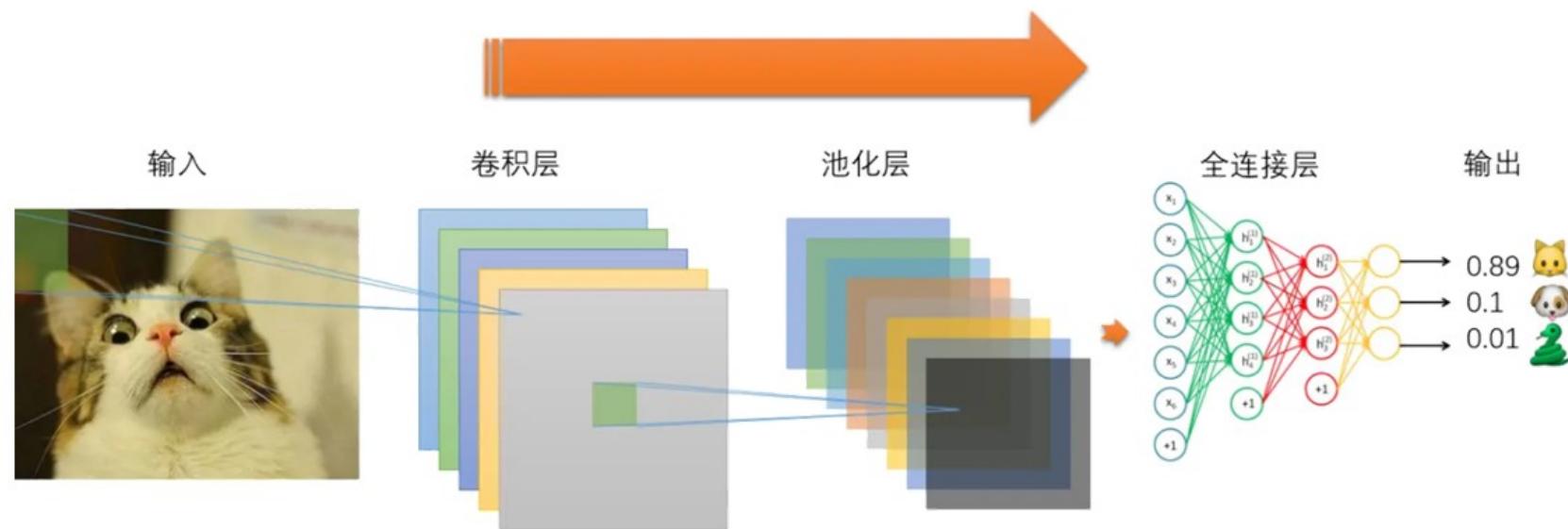
GAN

Diffusion

学习范式

关注模型的训练  
目标和优化框架

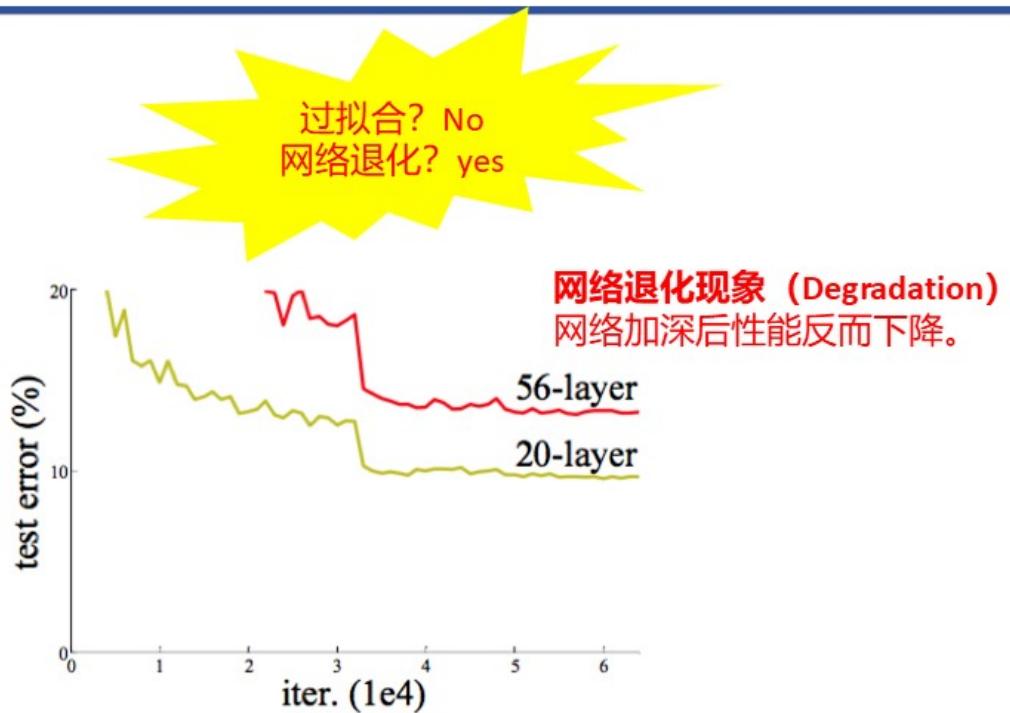
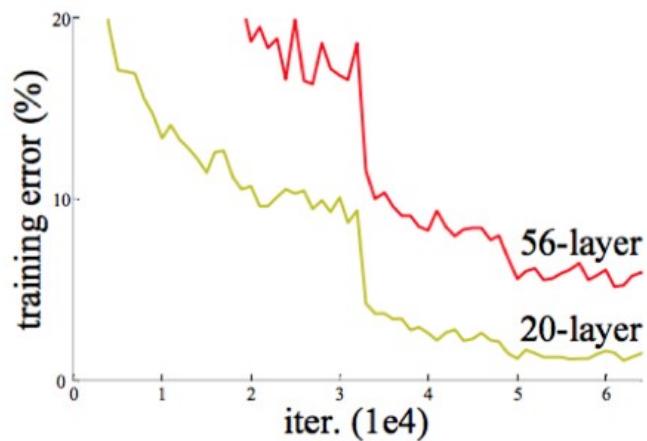
# 回顾：CNN



8

# CNN：网络退化问题

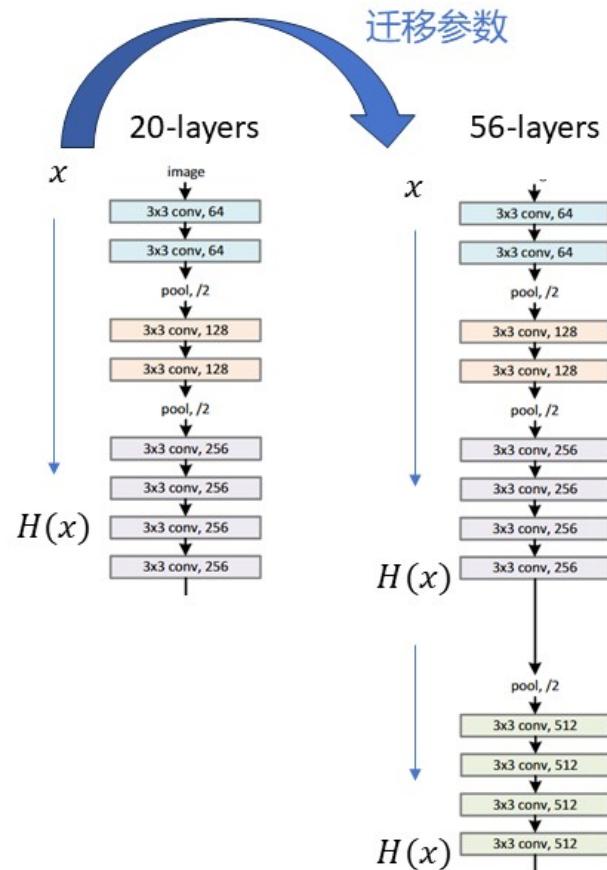
网络层数越高→模型效果越好？



56层网络在训练集和测试集上的误差都高于20层网络

# Motivation

模型深度有问题?



- 1) 前20层, 直接搬过来!
- 2) 后36层, 啥也不干, 输入什么就输出什么!

$H(x)$



网络学不会原封不动地输出

退化问题不是深度的问题→网络结构问题?

如何避免信息在层数过多时丢失?

10

# ResNet 残差网络

- 如何避免信息在层数过多时丢失 → **残差网络 跳跃连接**

写一篇作文，

- 先打草稿（原始想法），然后反复修改（加工信息）
- 如果修改时把草稿扔了，改到最后可能完全偏离最初的主题

终稿=“草稿+修改稿”

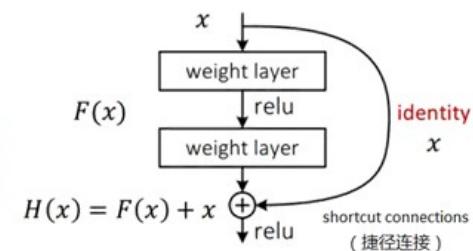
CNN

- 原始数据输入CNN，经过多层“卷积”、“池化”等操作
- 网络无法原封不动地输出

输出 = 原始输入 + 隐藏层变换

- 第一层输出： $f_1(x)$  (加工1次)；  
- 第二层输出： $f_2(f_1(x))$  (加工2次，基于第一层的结果)；  
- 第n层输出： $f_n(\dots f_2(f_1(x))\dots)$  (加工n次，完全基于前层结果)。

- 第一层输出： $x + f_1(x)$  (原始x + 第一次加工)；  
- 第二层输出： $(x + f_1(x)) + f_2(x + f_1(x))$  (保留了x，又加入第二次加工)；  
- 第n层输出： $x + f_1(x) + f_2(\dots) + \dots + f_n(\dots)$  (原始x始终存在，叠加了所有加工结果)。



# ResNet 残差网络

ResNet还能缓解梯度消失！

残差网络表达式：

$$x_{l+1} = x_l + \mathcal{F}(x_l, W_l) \quad x_L = x_l + \sum_{i=l}^{L-1} \mathcal{F}(x_i, W_i)$$

L层可表示为任意较浅的l层加上它们之间的残差，表明残差网络在训练中既保留原始信息，又学习新的特征。

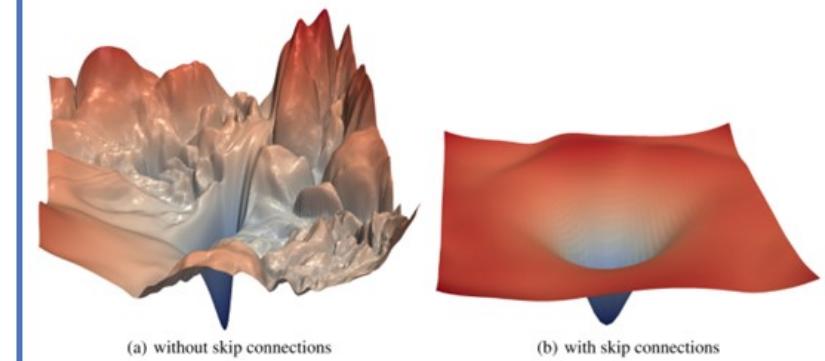
残差网络反向传播：

$$\frac{\partial \epsilon}{\partial a^{l_1}} = \frac{\partial \epsilon}{\partial a^{l_2}} \left( 1 + \frac{\partial \sum_{i=l_1}^{l_2-1} F(a^i)}{\partial a^{l_1}} \right)$$

损失函数的梯度表明，**误差信号可直接传递至低层**，而无需逐层链式传递，从而有效缓解梯度消失问题。

**【回顾】传统神经网络反向传播：**  $\frac{\partial \epsilon}{\partial a^{l_1}} = \frac{\partial \epsilon}{\partial a^{l_2}} \frac{\partial a^{l_2}}{\partial a^{l_1}} = \frac{\partial \epsilon}{\partial a^{l_2}} \frac{\partial a^{l_2}}{\partial a^{l_2-1}} \cdots \frac{\partial a^{l_1+1}}{\partial a^{l_1}}$

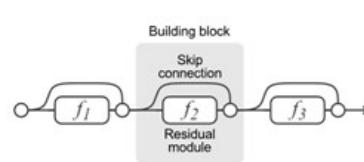
损失函数对权重梯度图：



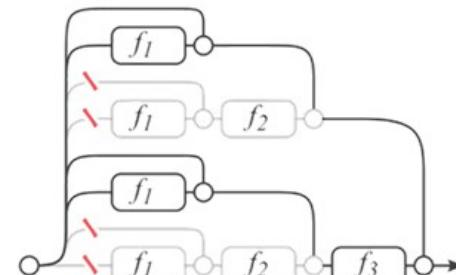
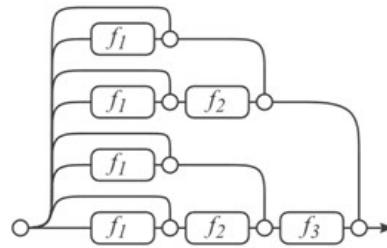
12

# ResNet 残差网络

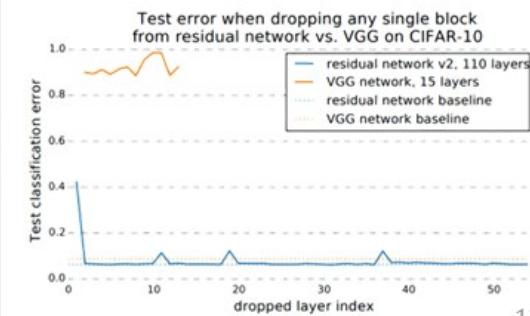
ResNet鲁棒性更好!



=

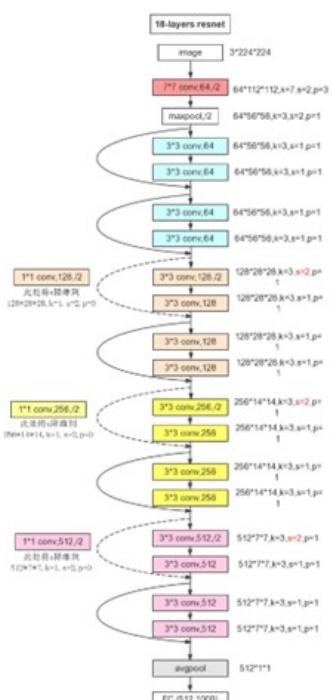


(a) Deleting  $f_2$  from unraveled view



13

# ResNet 残差网络



Kaiming He 何恺明

Associate Professor, EECS, MIT

Distinguished Scientist, Google DeepMind

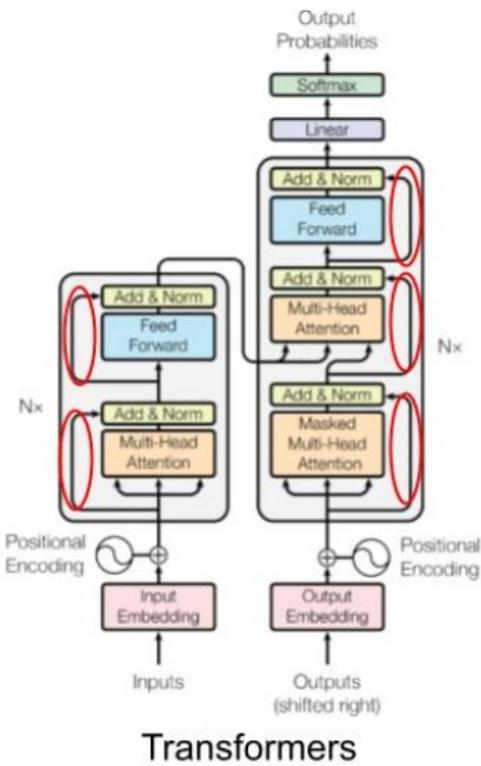
kaiming@mit.edu

45-701H, 51 Vassar St, Cambridge, MA 02139

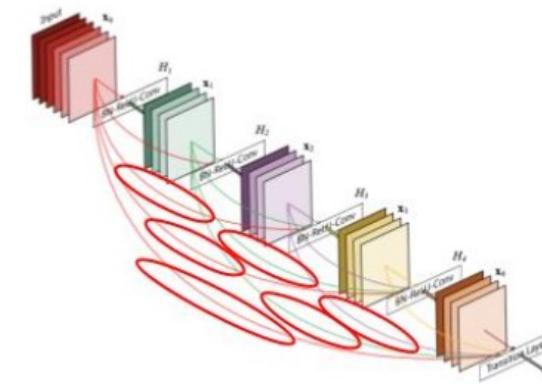
I am an Associate Professor with tenure in the [Department of EECS](#) at [MIT](#). I also work part-time as a Distinguished Scientist at Google DeepMind. My research areas include computer vision and deep learning.

I am best known for my work on [Deep Residual Networks \(ResNets\)](#), which is the most-cited paper of the twenty-first century, according to a [Nature](#) article. The proposed residual connections are now being used everywhere in modern deep learning models, including Transformers (e.g., ChatGPT), AlphaGo Zero, AlphaFold, and virtually all GenAI models today, among many others. I am also known for my work on visual perception (e.g., [Faster R-CNN](#), [Mask R-CNN](#)) and self-supervised learning (e.g., [MoCo](#), [MAE](#)). My publications have over [700,000](#) citations (as of May 2025).

# ResNet 残差网络



Transformers



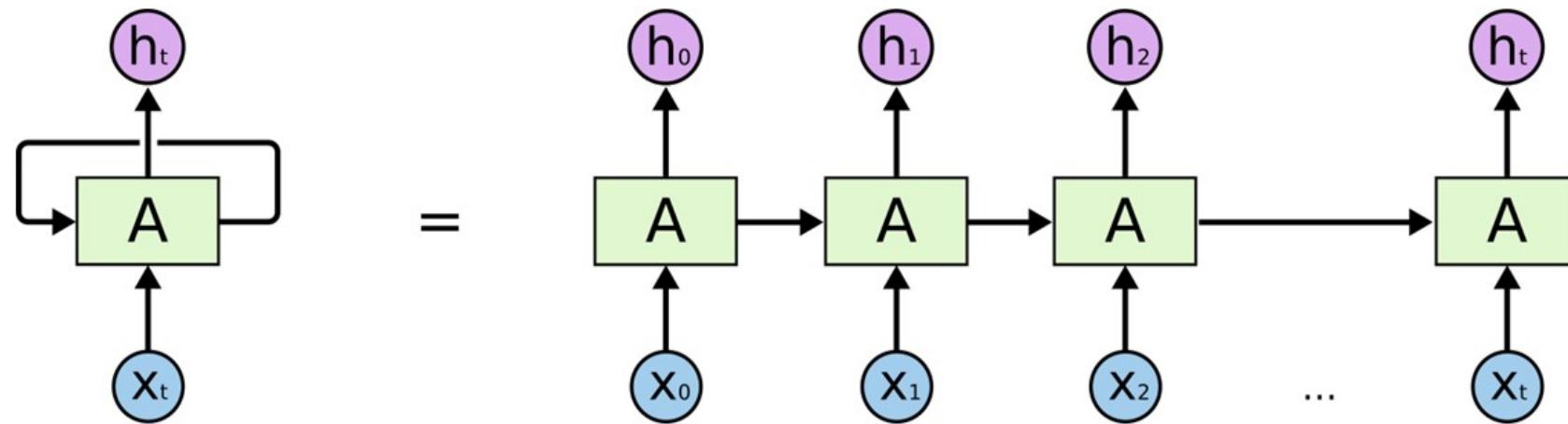
DenseNet

15

## 回顾：RNN

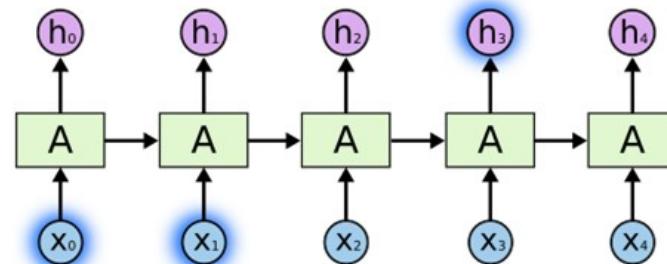
循环神经网络（Recurrent Neural Network, RNN）是用于处理序列数据的神经网络。

循环神经网络是一种共享参数的网络，参数在每个时间点上共享。

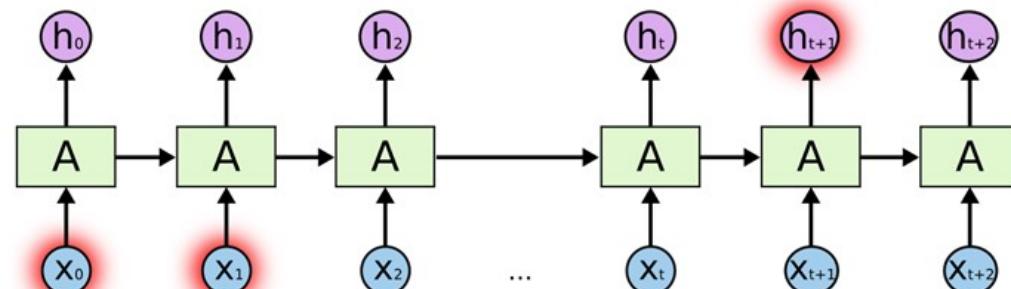


# RNN：长期依赖问题

RNN的吸引力之一在于它能够将先前的信息与当前任务联系起来，对于比较短的序列来说，RNN 可以学习利用过去的信息。



但是，随着序列长度的增加，RNN很难处理这种  
**长期依赖**关系。

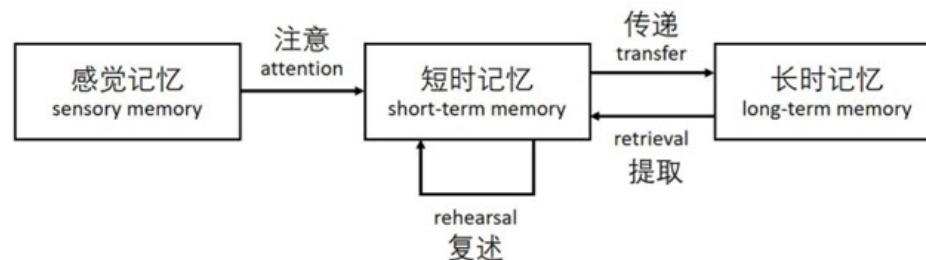


17

# LSTM Long Short Term Memory

## 记忆模型

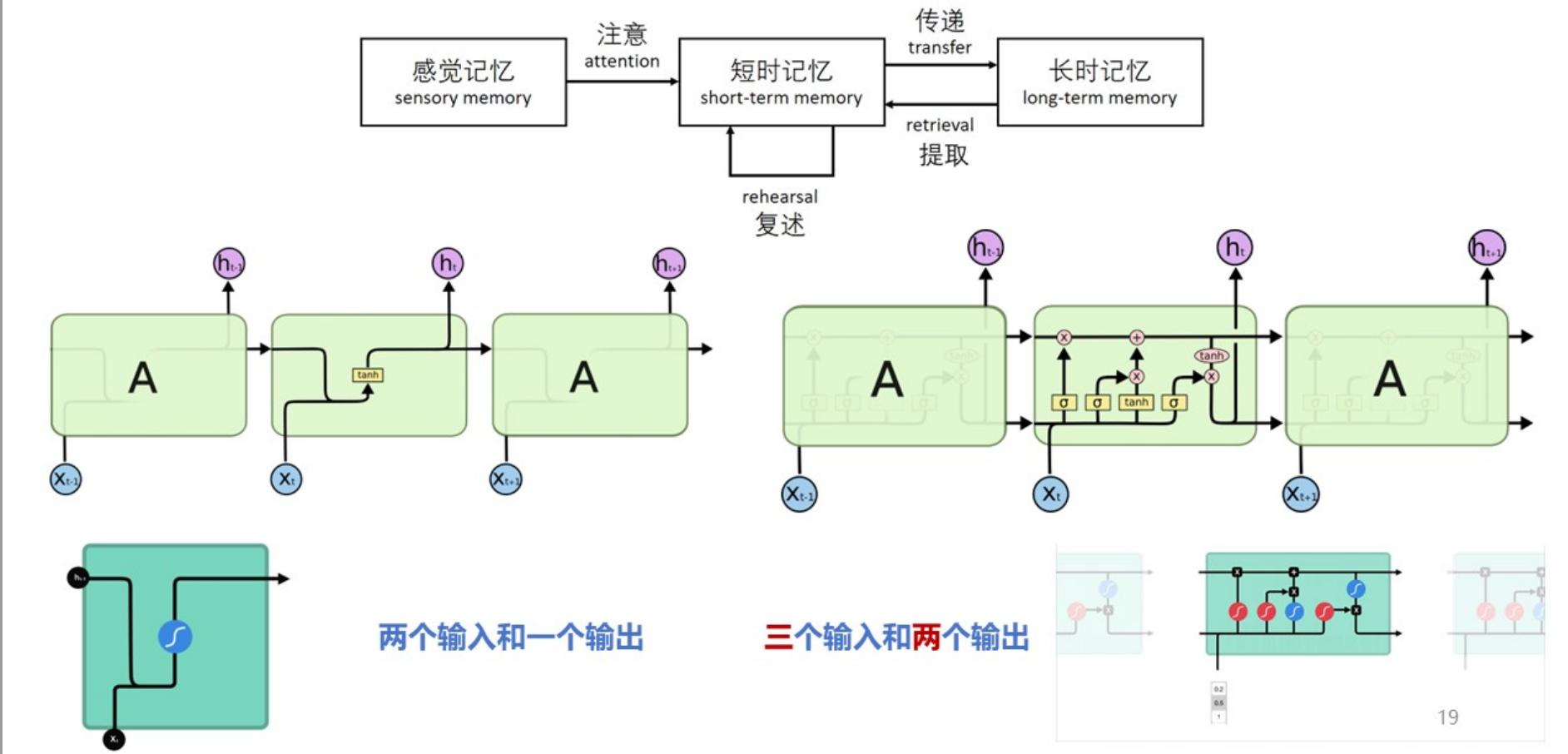
- 信息首先被存储在感觉记忆中，被注意选择的事件将进入短时记忆。
- 一旦进入短时记忆，如果事件被复述则可以进入长时记忆。
- 信息在每一个阶段都可能遗失，其原因可能是衰退、干扰，或者两者的结合。
- 信息从感觉记忆计入短时记忆，然后才能够进入长时记忆。



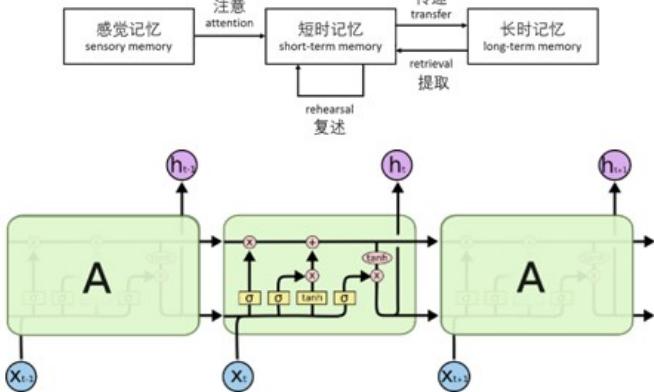
这个笔记本非常棒，**纸很厚**，料很足，用笔写起来手感非常舒服，而且**没有一股刺鼻的油墨味**；更加好的是这个笔记本不但**便宜还做工优良**，我上次在别家买的笔记本裁纸都裁不好，还会割伤手……

[1]Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

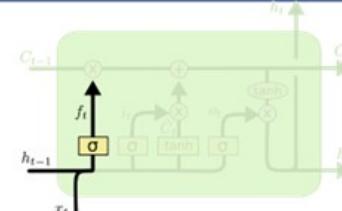
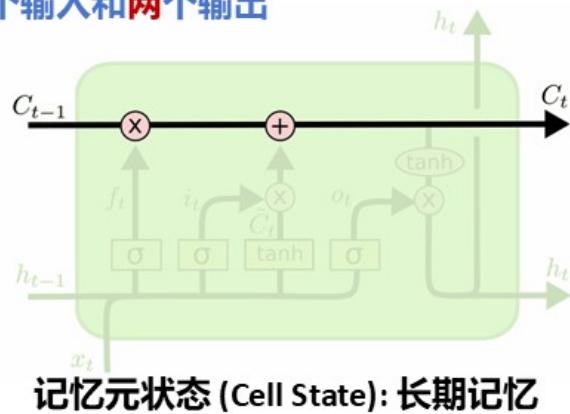
# LSTM：记忆的存储、修改与使用



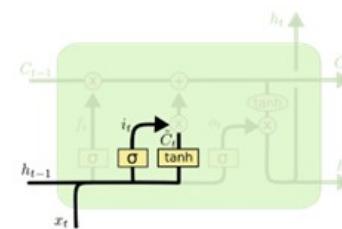
# LSTM：记忆的存储、修改与使用



三个输入和两个输出

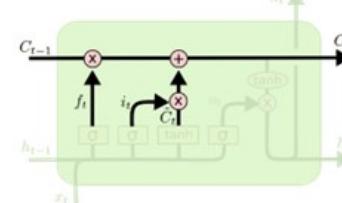


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

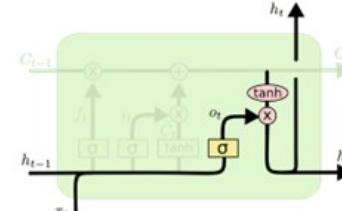


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

**遗忘门 (Forget Gate)**  
决定哪些信息从记忆中丢弃

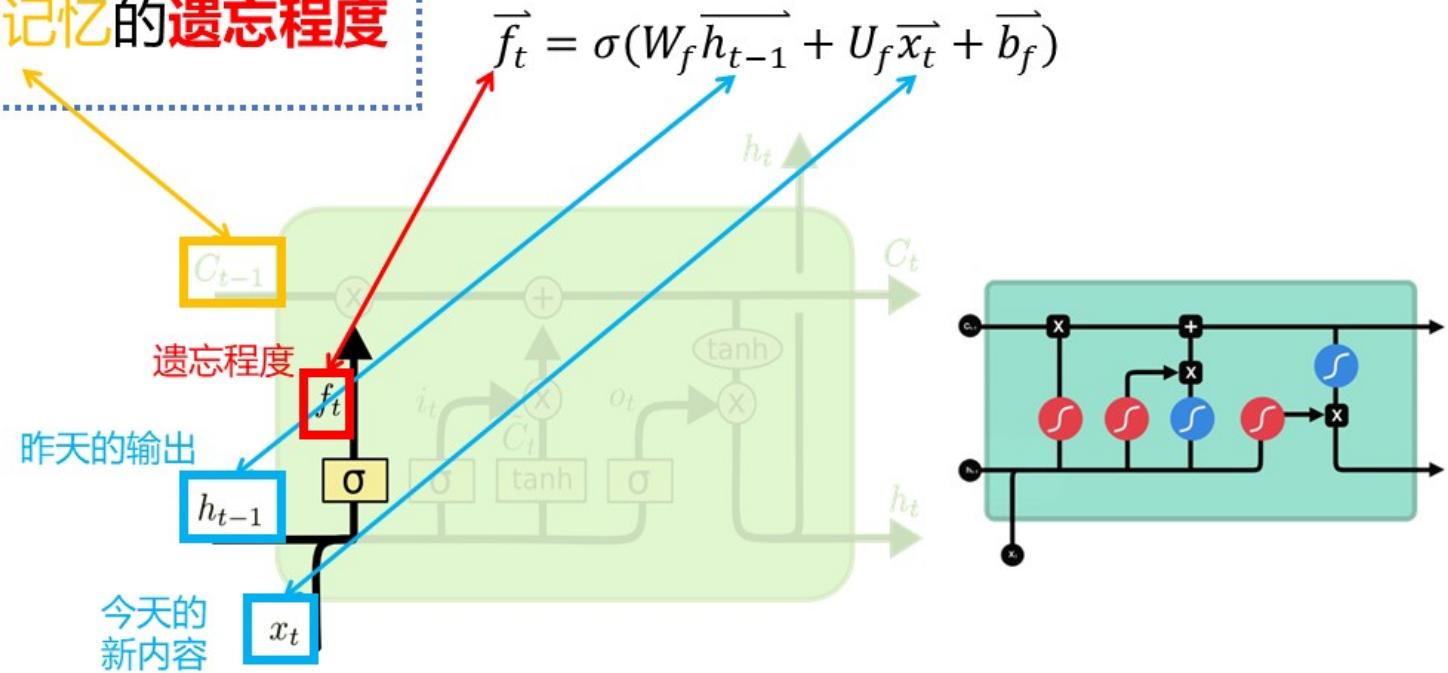
**输入门 (Input Gate)**  
决定哪些信息要加入到记忆

**记忆元状态更新**

**输出门 (Output Gate)**  
决定了当前状态的哪些信息会被输出

## LSTM: 遗忘门

作用：控制旧记忆的遗忘程度

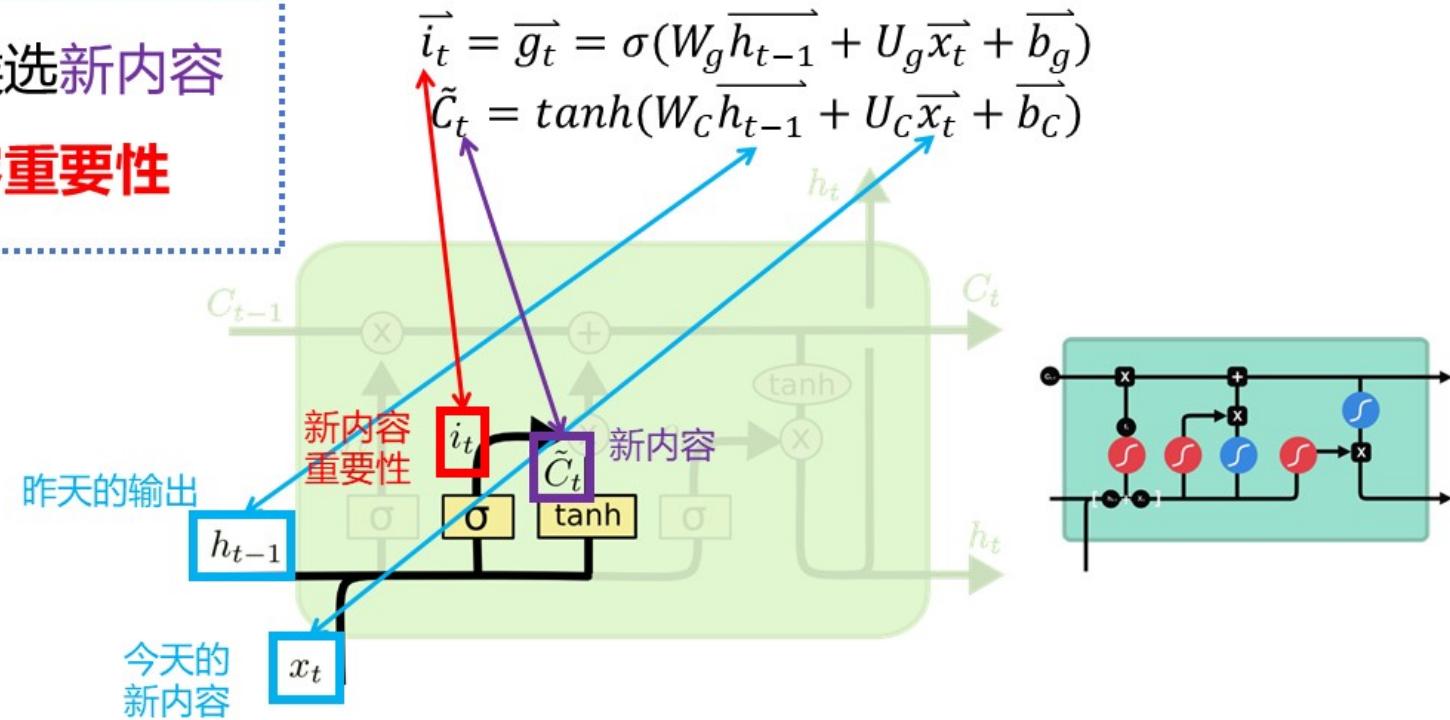


$\vec{b}_f$ 为遗忘门的偏置向量， $W_f$ 为遗忘门的循环权重， $U_f$ 为遗忘门的输入权重。  
 $\sigma$ 为逐元素的sigmoid函数（输出范围为(0,1)）

21

# LSTM: 输入门

作用：生成候选**新内容**  
和控制**新内容重要性**



$\overrightarrow{b_g}$  为输入门的偏置向量,  $W_g$  为输入门的循环权重,  $U_g$  为输入门的输入权重。

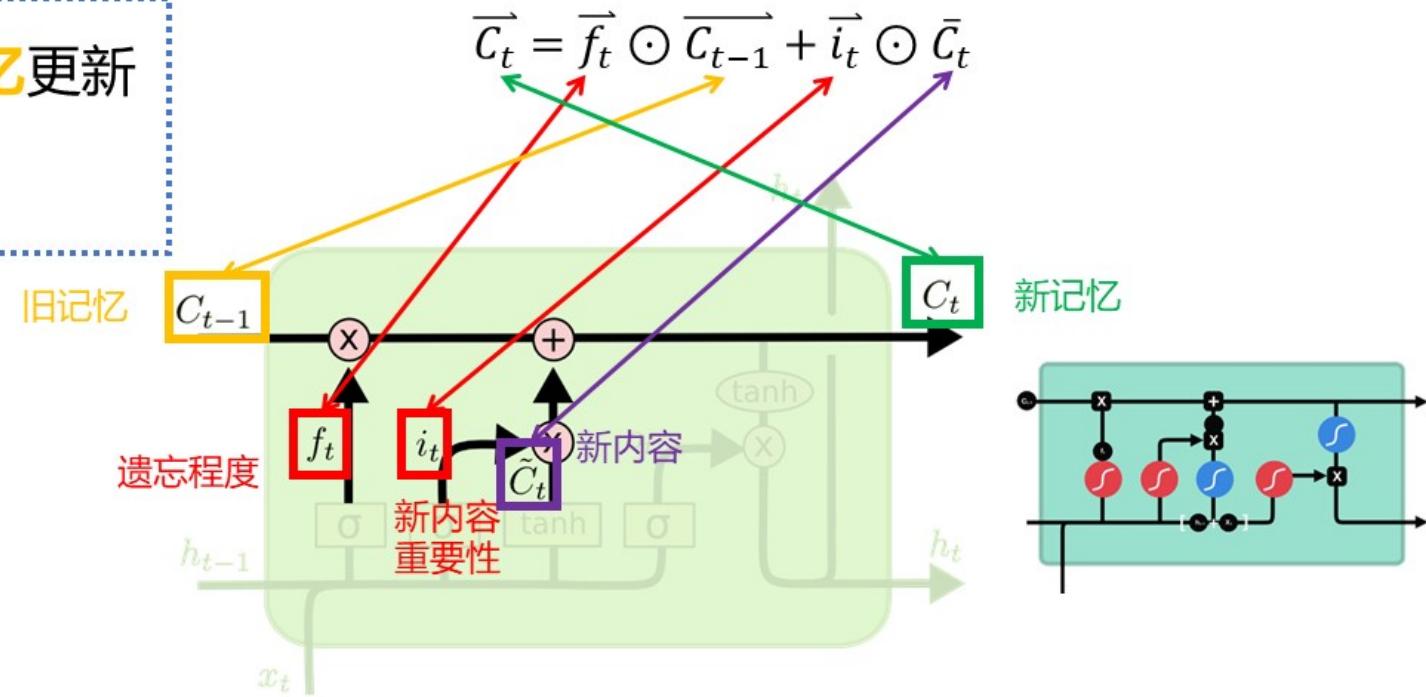
$\overrightarrow{b_C}$  为记忆元的偏置向量,  $W_C$  为记忆元的循环权重,  $U_C$  为记忆元的输入权重。

$\sigma$ 、 $\tanh$ 分别为逐元素的sigmoid和tanh函数 (输出范围分别为(0,1)和[-1,1])

22

## LSTM：记忆元更新

作用：把旧记忆更新  
成新记忆

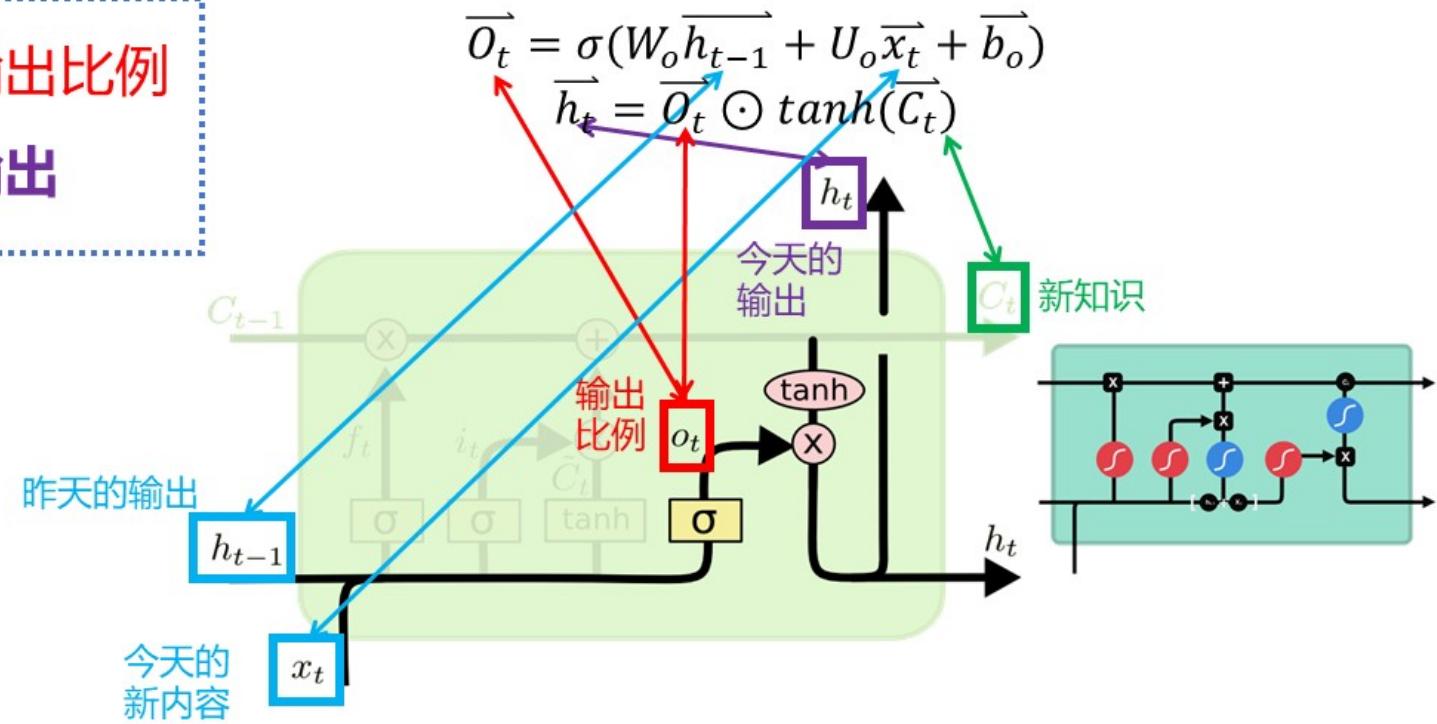


其中 $\odot$ 为逐元素相乘。

23

# LSTM: 输出门

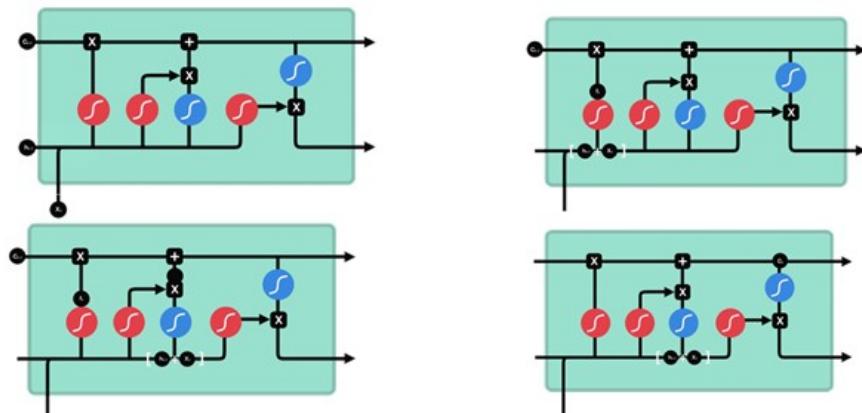
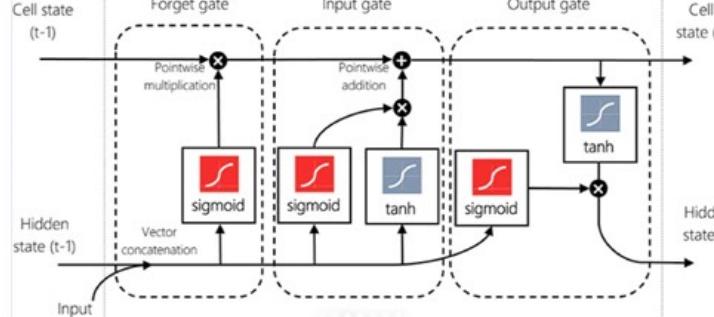
作用：控制输出比例  
并输出最终输出



$\overrightarrow{b_o}$  为输出门的偏置向量，  $W_o$  为输出门的循环权重，  $U_o$  为输出门的输入权重。  
 $\odot$  为逐元素相乘。

24

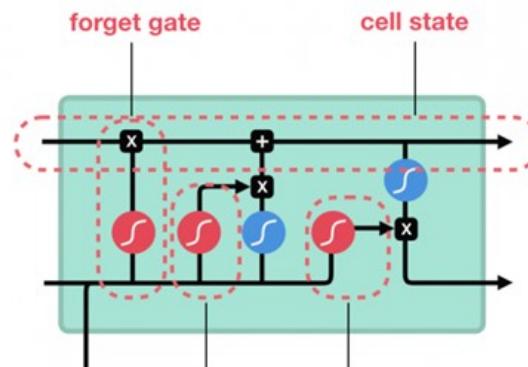
# LSTM：记忆的存储、修改与使用



	类比理解
遗忘门	遗忘程度 $\vec{f}_t = \text{sigmoid}$ (分析 [昨天的记忆 $\overrightarrow{h_{t-1}}$ , 今天的新内容 $\overrightarrow{x_t}$ ])
输入门	新内容重要性 $\vec{g}_t = \text{sigmoid}$ (分析 [昨天的记忆 $\overrightarrow{h_{t-1}}$ , 今天的新内容 $\overrightarrow{x_t}$ ]) 候选新内容 $\hat{c}_t = \tanh$ (分析 [昨天的记忆 $\overrightarrow{h_{t-1}}$ , 今天的新内容 $\overrightarrow{x_t}$ ])
记忆元状态	新记忆 $\overrightarrow{c_t} = \text{遗忘程度 } \vec{f}_t * \text{旧记忆 } \overrightarrow{c_{t-1}} + \text{新内容重要性 } \vec{g}_t * \text{候选新内容 } \hat{c}_t$
输出门	输出比例 $\overrightarrow{o_t} = \text{sigmoid}$ (分析 [昨天的记忆 $\overrightarrow{h_{t-1}}$ , 今天的新内容 $\overrightarrow{x_t}$ ]) 最终输出 $\overrightarrow{h_t} = \text{输出比例 } \overrightarrow{o_t} * \tanh(\text{新记忆 } \overrightarrow{c_t})$

# LSTM vs GRU

- 摆脫了cell状态，直接用隱藏状态传递信息
- 只有两个門：重置門(上一时刻多少信息需要被遗忘)和更新門 (当前时刻多少有用信息需要被传下去)



$$i_t = \sigma(W_i [h_{t-1}, x_t])$$

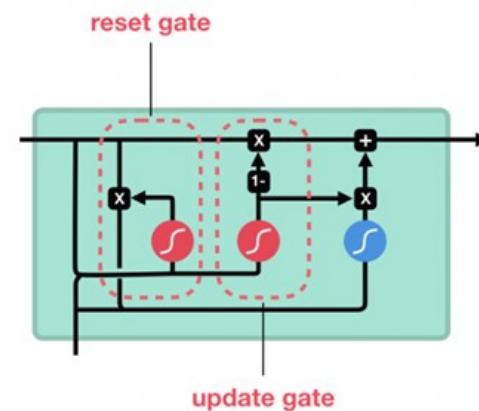
$$f_t = \sigma(W_f [h_{t-1}, x_t])$$

$$o_t = \sigma(W_o [h_{t-1}, x_t])$$

$$\tilde{C}_t = \tanh(W_c [h_{t-1}, x_t])$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$



$$r_t = \sigma(W_r [h_{t-1}, x_t])$$

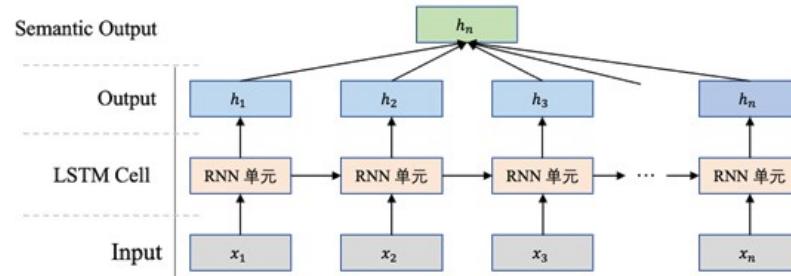
$$z_t = \sigma(W_z [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W_h [r_t \odot h_{t-1}, x_t])$$

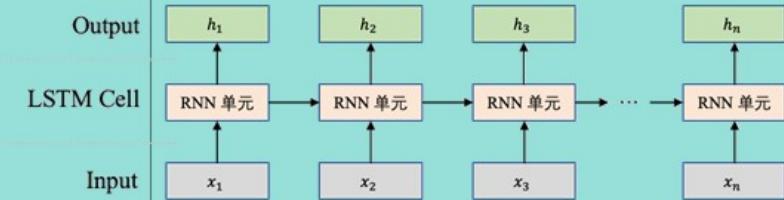
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

26

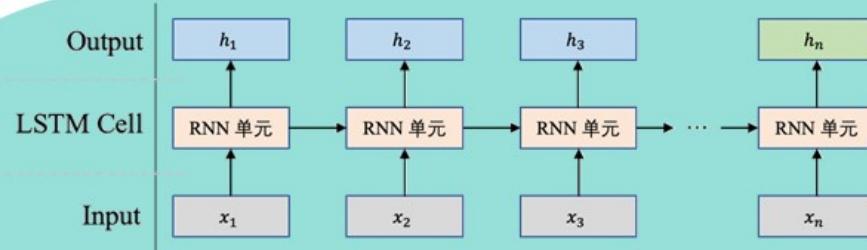
# Sequence-to-Sequence(seq2seq)



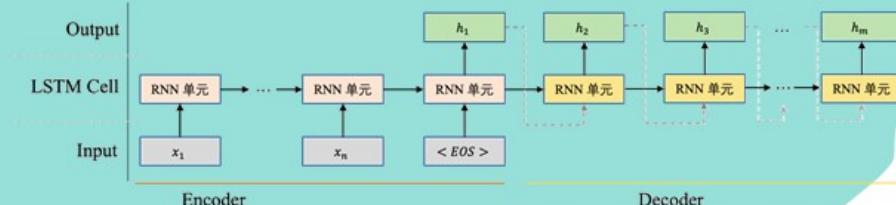
Sequence-to-Class



序列标注(Sequence Labeling)



Sequence-to-Sequence (seq2seq)



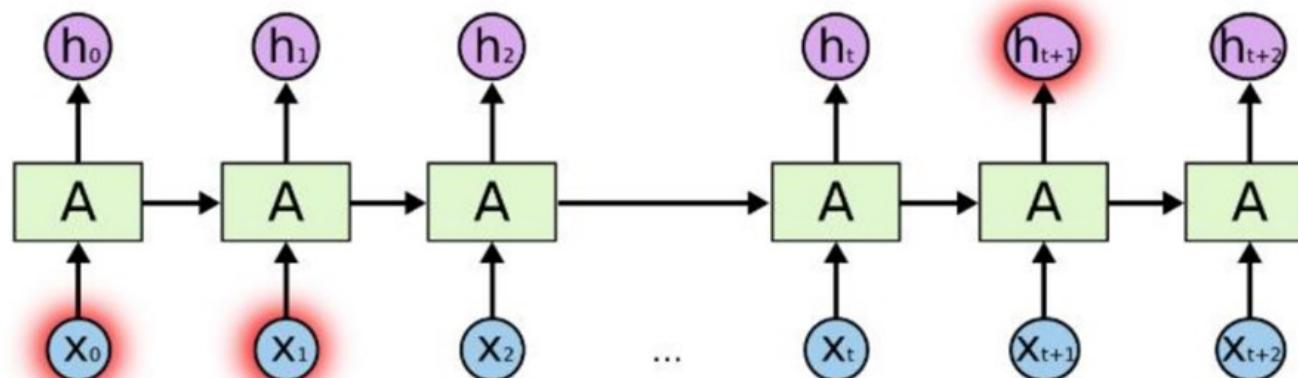
编码器-解码器模型(encoder-decoder)

27

# LSTM的不足

虽然 LSTM (长短期记忆网络) 在一定程度上缓解了传统 RNN 的问题，使模型能够更好地捕捉较长距离的依赖关系

但它仍然**难以建模更长程的依赖**: LSTM的信息依然是顺序传播的，当需要建立句子开头和结尾之间联系时，信息（和梯度）仍需逐步传递整个序列，距离过长时依旧容易衰减。

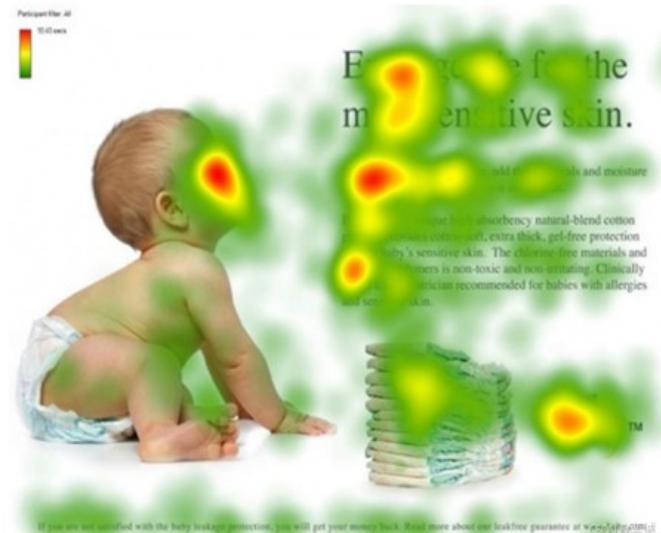


28

# 注意力机制 (Attention)

- 注意：人类信息加工过程中的一项重要的心理调节机制，它能够对有限的信息加工资源进行分配，使感知具备选择能力。
- 注意力机制 (Attention)：集中关注信息的关键部分来提取出更加重要的内容

Attention本质：关注全部 -> 关注重点



The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .  
The FBI is chasing a criminal on the run .

29

# 注意力机制 (Attention)

**本质：关注全部 -> 关注重点**



a: 渣男的自身经济实力

Query (Q): 渣男的择偶需求

Key (K): 备胎的自身条件

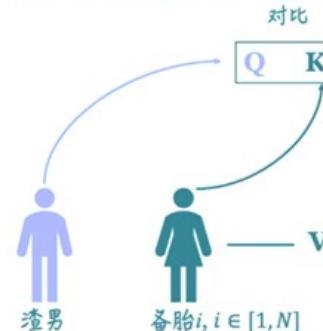
Value (V): 备胎的经济实力

b: 渣男的自身经济实力 (吃上软饭版)

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

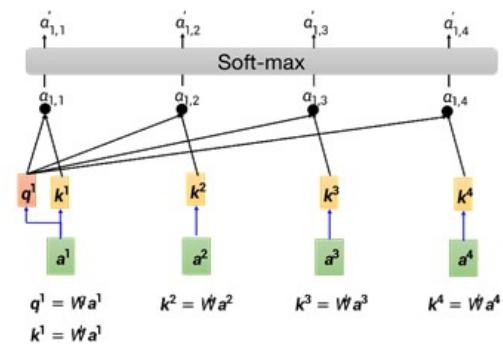
$$\begin{array}{ccc} \text{Query} & \text{Key} & \text{Value} \\ Q = W^Q X & K = W^K X & V = W^V X \\ \text{查询} & \text{索引} & \text{内容} \end{array}$$

渣男和备胎匹配程度

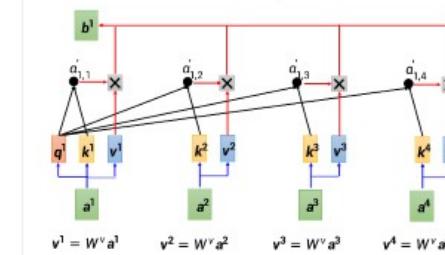


Attention: 渣男对某个备胎的关注度

$$a'_{1,i} = \exp(a_{1,i}) / \sum_j \exp(a_{1,j})$$



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



b: 渣男的自身经济实力 (吃上软饭版)

30

# 注意力机制 (Attention)

本质：关注全部 -> 关注重点



a: 渣男的自身经济实力

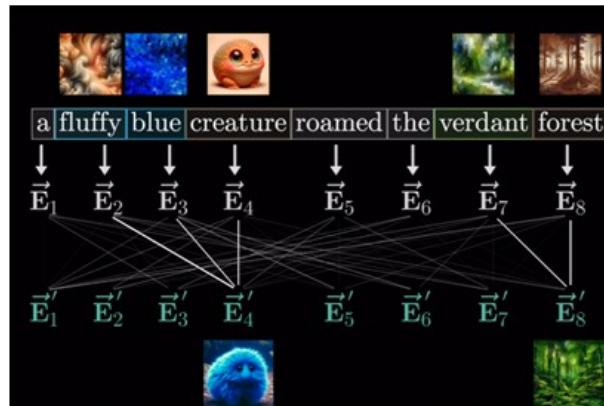
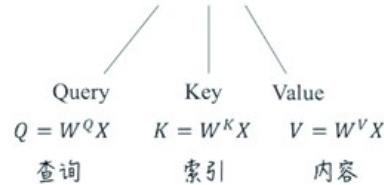
Query (Q): 渣男的择偶需求

Key (K): 备胎的自身条件

Value (V): 备胎的经济实力

b: 渣男的自身经济实力 (吃上软饭版)

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



a: 当前词原始的语义信息

Query (Q): 当前词对其他词的信息需求

Key (K): 表示各个词被检索时的特征

Value (V): 被检索词本身的语义信息

b: 当前词在上下文中的语义信息

- a: “creature” – 生物、怪物、动物
- Q (creature) : 我是名词“生物”，我要找形容词。
- K (fluffy, blue): “fluffy”: 我是‘毛茸茸’的形容词； “blue”: 我是‘蓝色’的形容词。
- V (fluffy, blue): 被检索词最原始的语义信息。blue, 蓝色, 忧郁.....
- b: 生物 -> 蓝色毛绒绒的生物

- 每个词都能根据整句内容，更新自己的语义表示
- 得到的表示就不只是“词义”，而是“语境下的词义”

31

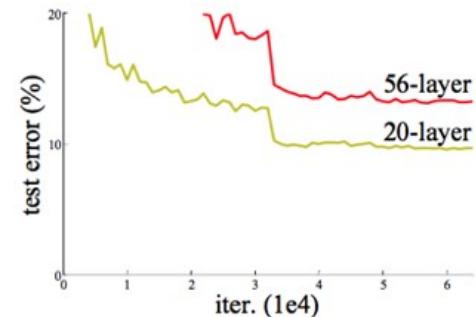
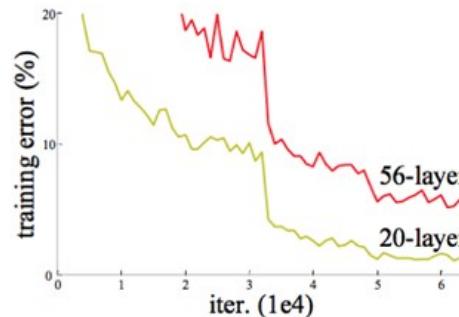
# 课程回顾

5. 神经网络进阶: ResNet 残差网络	47 份
5. 神经网络进阶: 注意力机制 Attention	47 份
5. 神经网络进阶: LSTM Long Short Term Memory	46 份
3. 神经网络基础: 卷积神经网络CNN	42 份
3. 神经网络基础: 循环神经网络RNN	42 份
2. 机器学习基础: 贝叶斯网络	41 份
2. 机器学习基础: K-邻近算法	41 份
3. 神经网络基础: 前向传播与反向传播	41 份
2. 机器学习基础: 支持向量机	39 份
4. 模型训练与优化: 训练过程优化方法	38 份
4. 模型训练与优化: 模型表现分析	37 份
2. 机器学习基础: 逻辑回归	35 份
2. 机器学习基础: 线性回归	34 份
	32

单选题 1分

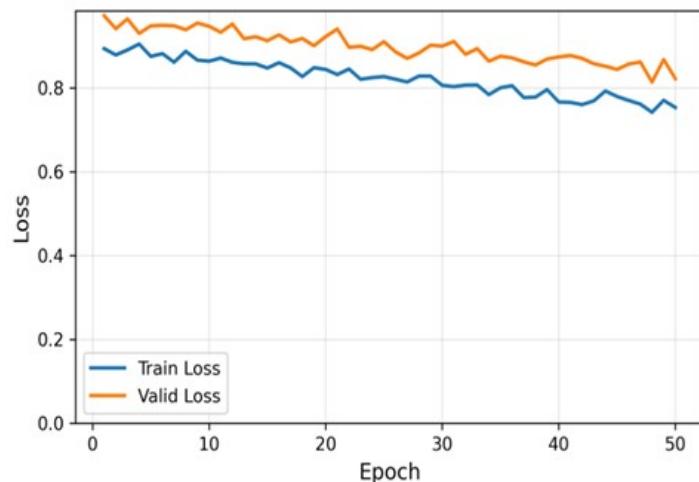
图中是什么问题？

- A 过拟合
- B 欠拟合
- C 网络退化
- D 梯度消失
- E 梯度下降

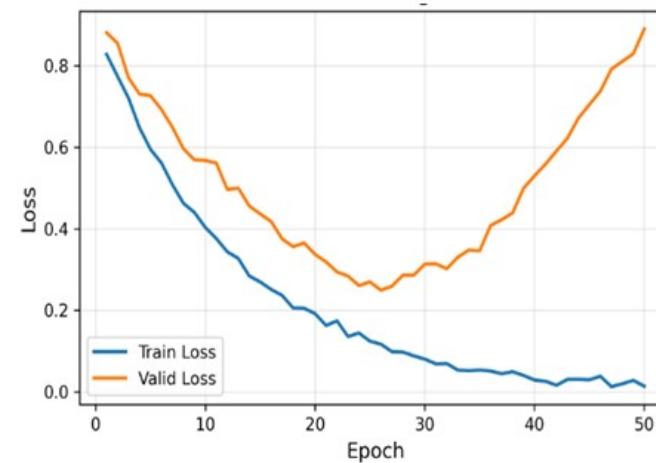


# 课程回顾

欠拟合 (Underfitting)



过拟合 (Overfitting)



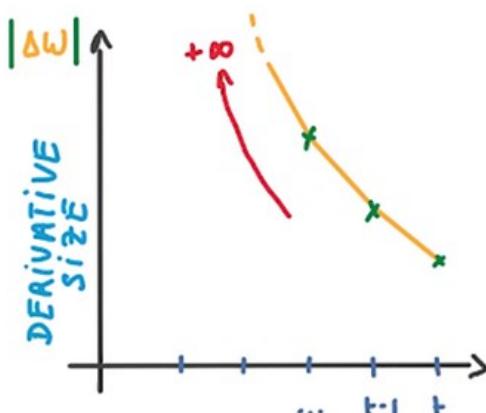
- 考试前只粗略地翻了翻课本，连最基本的概念和公式都没掌握。
- 无论是在练习题上还是在期末考试中，分数都很低。
- 大脑（模型）因为太过简单，没有学到数据中的精髓。
- **训练和测试误差都很高**

- 把练习册上的每一道题，包括答案和解题步骤、题目旁边的污渍都背得滚瓜烂熟。在做练习题时能拿到满分，但一到期末考试，题目稍微变个样，你就束手无策了。
- 不仅学到了知识，还学到了练习题特有的“噪音”和无关细节，导致泛化能力极差。
- **训练误差很低，测试误差很高**

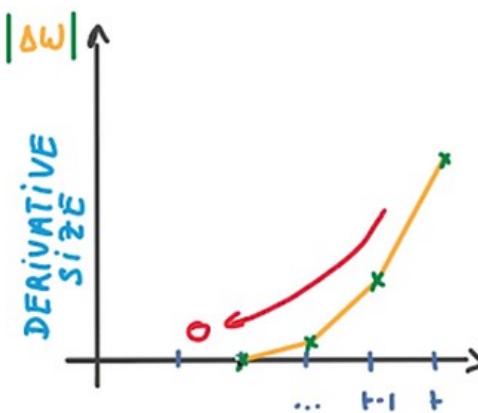
34

# 课程回顾

梯度爆炸



梯度消失



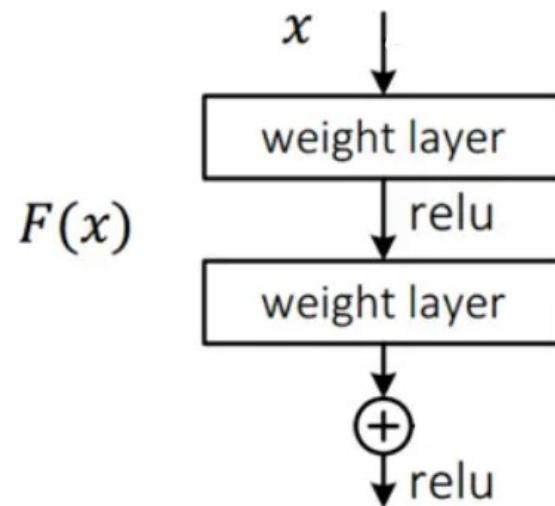
梯度下降：连乘

- 在反向传播中，若每层的梯度系数小于 1，层数越多梯度越趋近于 0，产生梯度消失；
- 若每层的梯度系数大于 1，层数越多梯度越趋近于无穷大，产生梯度爆炸。

35

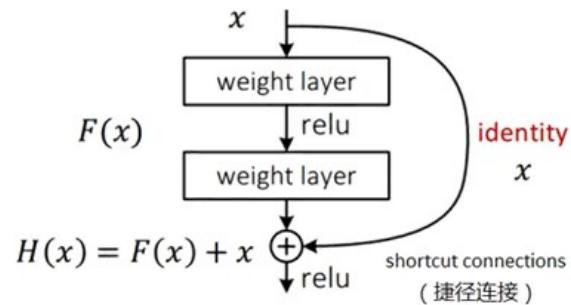
主观题 10分

请你将以下的网络改造成resnet网络（加一条边），并写出改造后的网络输出 $H(x)$ 的表达式



# ResNet 残差网络

终稿=“草稿+修改稿”

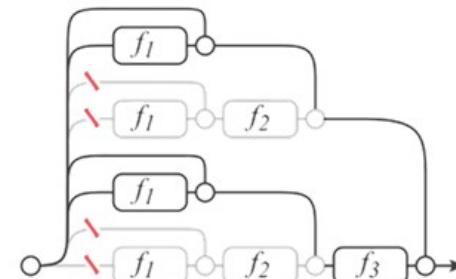


$$x_L = x_l + \sum_{i=l}^{L-1} \mathcal{F}(x_i, W_i)$$

$$\frac{\partial \epsilon}{\partial a^{l_1}} = \frac{\partial \epsilon}{\partial a^{l_2}} \left( 1 + \frac{\partial \sum_{i=l_1}^{l_2-1} F(a^i)}{a^{l_1}} \right)$$

输出 = 原始输入 + 隐藏层变换

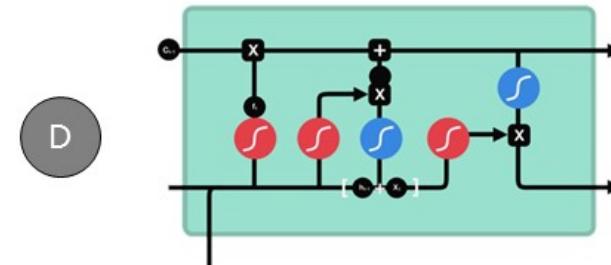
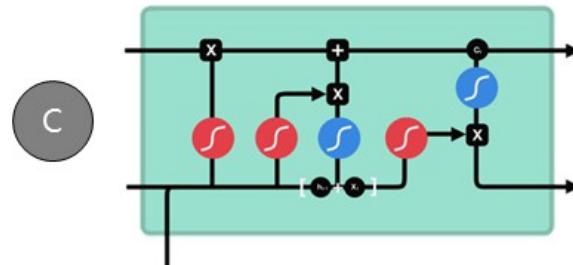
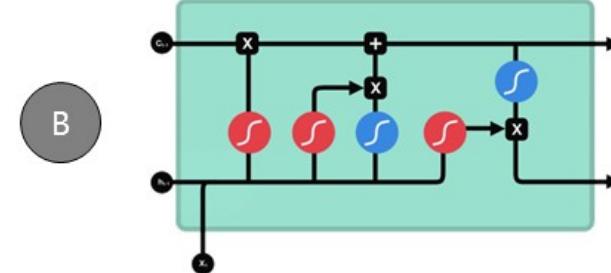
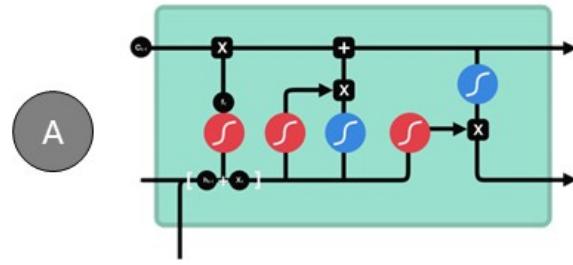
- 不忘初心 → 网络退化
- 打破连乘 → 梯度消失
- 跳跃连接 → 提高鲁棒



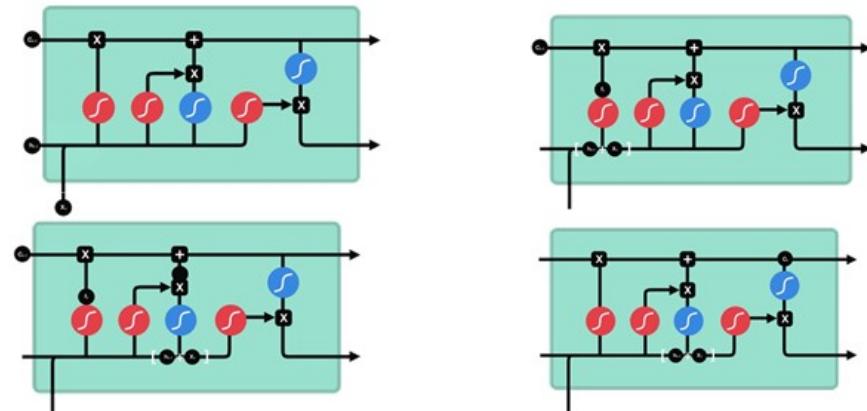
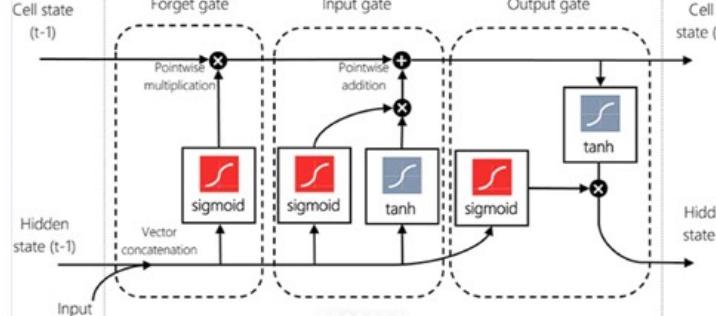
(a) Deleting  $f_2$  from unraveled view

填空题 4分

图为LSTM的主要信息传播模块，请按照输入门、遗忘门、记忆元状态、输出门的顺序进行排序 [填空1] [填空2] [填空3] [填空4]



# LSTM：记忆的存储、修改与使用



	类比理解	
遗忘门	遗忘程度 $\vec{f}_t = \text{sigmoid}$ (分析[昨天的记忆 $\overrightarrow{h_{t-1}}$ , 今天的新内容 $\overrightarrow{x_t}$ ])	$\vec{f}_t = \sigma(W_f \overrightarrow{h_{t-1}} + U_f \overrightarrow{x_t} + \vec{b}_f)$
输入门	新内容重要性 $\vec{g}_t = \text{sigmoid}$ (分析[昨天的记忆 $\overrightarrow{h_{t-1}}$ , 今天的新内容 $\overrightarrow{x_t}$ ]) 候选新内容 $\hat{C}_t = \tanh$ (分析[昨天的记忆 $\overrightarrow{h_{t-1}}$ , 今天的新内容 $\overrightarrow{x_t}$ ])	$\vec{i}_t = \vec{g}_t = \sigma(W_g \overrightarrow{h_{t-1}} + U_g \overrightarrow{x_t} + \vec{b}_g)$ $\hat{C}_t = \tanh(W_c \overrightarrow{h_{t-1}} + U_c \overrightarrow{x_t} + \vec{b}_c)$
记忆元状态	新记忆 $\overrightarrow{C_t}$ = 遗忘程度 $\vec{f}_t$ * 旧记忆 $\overrightarrow{C_{t-1}}$ + 新内容重要性 $\vec{i}_t$ * 候选新内容 $\hat{C}_t$	$\overrightarrow{C_t} = \vec{f}_t \odot \overrightarrow{C_{t-1}} + \vec{i}_t \odot \hat{C}_t$
输出门	输出比例 $\overrightarrow{o_t} = \text{sigmoid}$ (分析[昨天的记忆 $\overrightarrow{h_{t-1}}$ , 今天的新内容 $\overrightarrow{x_t}$ ]) 最终输出 $\overrightarrow{h_t}$ = 输出比例 $\overrightarrow{o_t}$ * $\tanh(\text{新记忆 } \overrightarrow{C_t})$	$\overrightarrow{o_t} = \sigma(W_o \overrightarrow{h_{t-1}} + U_o \overrightarrow{x_t} + \vec{b}_o)$ $\overrightarrow{h_t} = \overrightarrow{o_t} \odot \tanh(\overrightarrow{C_t})$

39

填空题 3分

(单选题) LSTM 输入门的作用是 [填空1]，遗忘门的作用是 [填空2]，输出门的作用是 [填空3]。

- A. 控制旧记忆的遗忘程度
- B. 生成候选新内容和控制新内容重要性
- C. 把旧记忆更新成新记忆
- D. 控制输出比例并输出最终输出

填空题 1分

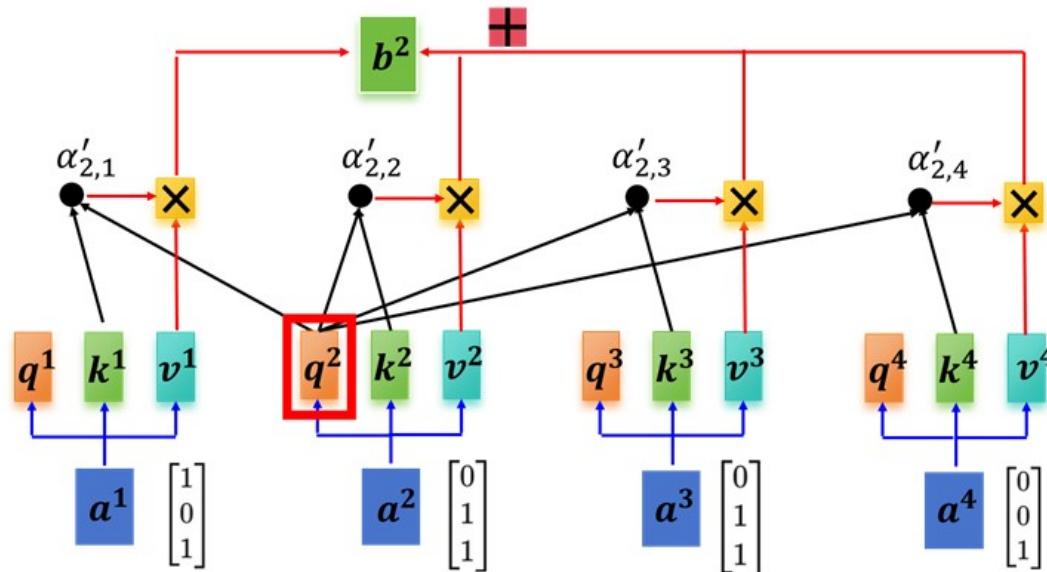
计算attention, 忽略softmax和sqrt运算

(为了方便计算, 假设 softmax( $x$ )= $x$ , sqrt( $x$ )=1)

作答形式: [1,2,3]。Answer: [填空1]

假设所有  $W^Q$ 、 $W^K$ 、 $W^V$  都是一样的

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$



1.  $score(q_i, k_j) = (q_i^T \cdot k_j) / \text{sqrt}(d_k), \quad q_i \in R^d, k_j \in R^d$
2.  $\alpha_{ij} = \text{softmax}(score_{ij}),$
3.  $b_i = \sum_j (\alpha_{ij} v_j),$

# Attention

## 步骤 1: 计算 Q, K, V 向量

给定的权重矩阵  $\mathbf{W}$  和输入向量  $a_i$  分别为:

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$a_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \quad a_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad a_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad a_4 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

进行矩阵-向量乘法:

$$\bullet v_1 = k_1 = q_1 = \mathbf{W}a_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

$$\bullet v_2 = k_2 = q_2 = \mathbf{W}a_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

$$\bullet v_3 = k_3 = q_3 = \mathbf{W}a_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

$$\bullet v_4 = k_4 = q_4 = \mathbf{W}a_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

## 步骤 2: 计算注意力得分 (Attention Scores)

接下来, 我们计算  $q_2$  对所有  $k_j$  的点积得分  $\alpha'_{2,j}$ :

我们的查询向量是  $q_2 = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$ 。

$$\bullet \alpha'_{2,1} = q_2^T \cdot k_1 = [0 \quad 2 \quad 1] \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = (0 \times 1) + (2 \times 1) + (1 \times 2) = 0 + 2 + 2 = 4$$

$$\bullet \alpha'_{2,2} = q_2^T \cdot k_2 = [0 \quad 2 \quad 1] \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} = (0 \times 0) + (2 \times 2) + (1 \times 1) = 0 + 4 + 1 = 5$$

$$\bullet \alpha'_{2,3} = q_2^T \cdot k_3 = [0 \quad 2 \quad 1] \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} = (0 \times 0) + (2 \times 2) + (1 \times 1) = 0 + 4 + 1 = 5$$

$$\bullet \alpha'_{2,4} = q_2^T \cdot k_4 = [0 \quad 2 \quad 1] \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = (0 \times 0) + (2 \times 1) + (1 \times 1) = 0 + 2 + 1 = 3$$

所以, 我们得到的注意力得分序列为  $[4, 5, 5, 3]$ 。

**步骤3: 因为我们设置 softmax(x) = x, 所以第三步计算注意力权重的结果不变, 还是[4,5,5,3]**

42

# Attention

## 步骤4: 计算最终输出向量 $b^2$

最后, 我们将新的注意力得分作为权重, 对所有的  $v_j$  向量进行加权求和。

$$b_2 = \sum_{j=1}^4 \alpha'_{2,j} v_j = \alpha'_{2,1} v_1 + \alpha'_{2,2} v_2 + \alpha'_{2,3} v_3 + \alpha'_{2,4} v_4$$

$$b_2 = 4 \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} + 5 \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} + 5 \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} + 3 \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

分别计算每个维度:

- 第一维:  $(4 \times 1) + (5 \times 0) + (5 \times 0) + (3 \times 0) = 4$
- 第二维:  $(4 \times 1) + (5 \times 2) + (5 \times 2) + (3 \times 1) = 4 + 10 + 10 + 3 = 27$
- 第三维:  $(4 \times 2) + (5 \times 1) + (5 \times 1) + (3 \times 1) = 8 + 5 + 5 + 3 = 21$

所以, 修正后的最终输出向量为:

$$b_2 = \begin{bmatrix} 4 \\ 27 \\ 21 \end{bmatrix}$$

43

# 注意力机制 (Attention)

**本质：关注全部 -> 关注重点**



a: 渣男的自身经济实力

Query (Q): 渣男的择偶需求

Key (K): 备胎的自身条件

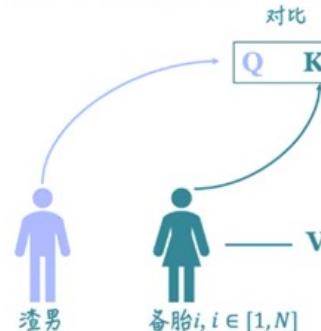
Value (V): 备胎的经济实力

b: 渣男的自身经济实力 (吃上软饭版)

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

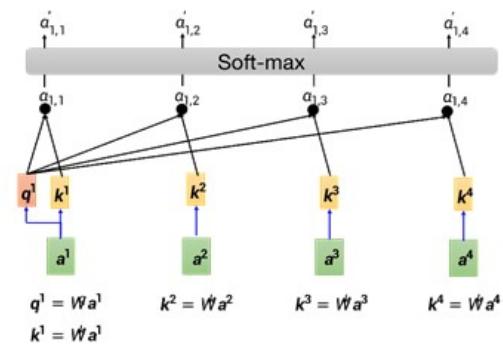
$$\begin{array}{ccc} \text{Query} & \text{Key} & \text{Value} \\ Q = W^Q X & K = W^K X & V = W^V X \\ \text{查询} & \text{索引} & \text{内容} \end{array}$$

渣男和备胎匹配程度

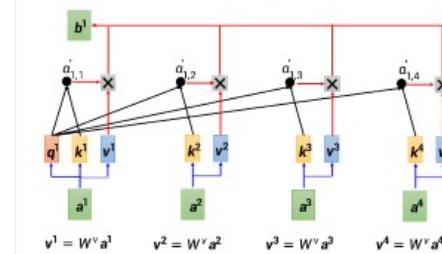


Attention: 渣男对某个备胎的关注度

$$a'_{1,i} = \exp(a_{1,i}) / \sum_j \exp(a_{1,j})$$



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



b: 渣男的自身经济实力 (吃上软饭版)

44

# Attention

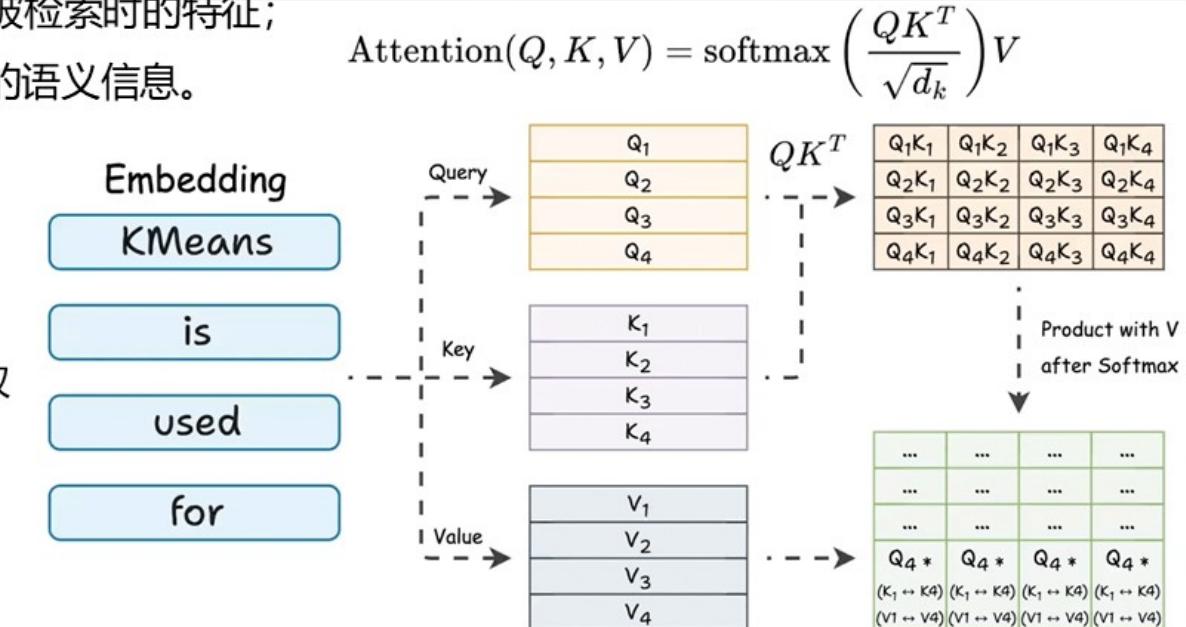
## 从输入到 Q、K、V 的映射：

对于输入序列中的每一个词  $a$  (Embedding形式) , 模型会通过线性变换生成三个不同的表示

- 查询向量 Q (Query) 表示当前词对其他词的信息需求;
- 键向量 K (Key) 表示各个词被检索时的特征;
- 值向量 V (Value) 承载词语的语义信息。

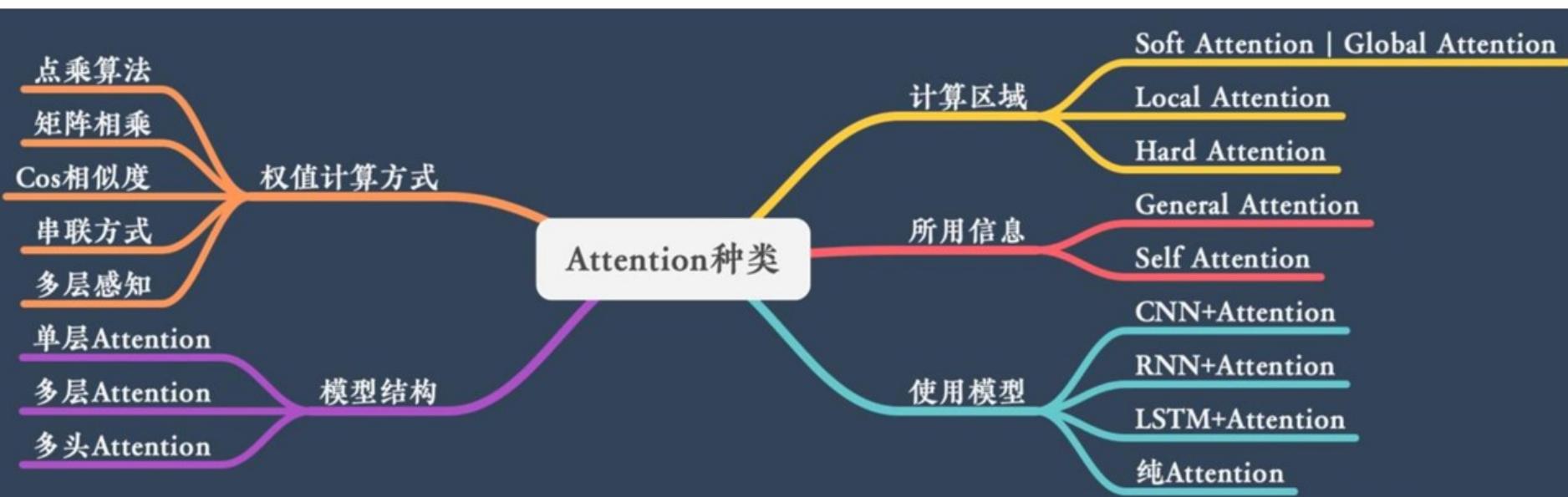
### 在注意力计算中

1. 通过 **QK 矩阵** 计算所有单词之间的**关联度**
2. **softmax** 函数来计算每个单词的权重, 确定相关单词的**重要程度**
3. 根据重要程度**加权求和**对应的 **V**, 从而得到**更新后的词表示**



# Attention

这里介绍的是目前最常用的 Transformer 架构中的 Attention 机制。实际上，Attention 最早起源于计算机视觉 (cv) 领域，随后在 RNN、LSTM 等模型中也得到了应用。



Transformer 是第一个完全依赖 Attention 的模型架构，并使这一机制得到了广泛推广与发展。

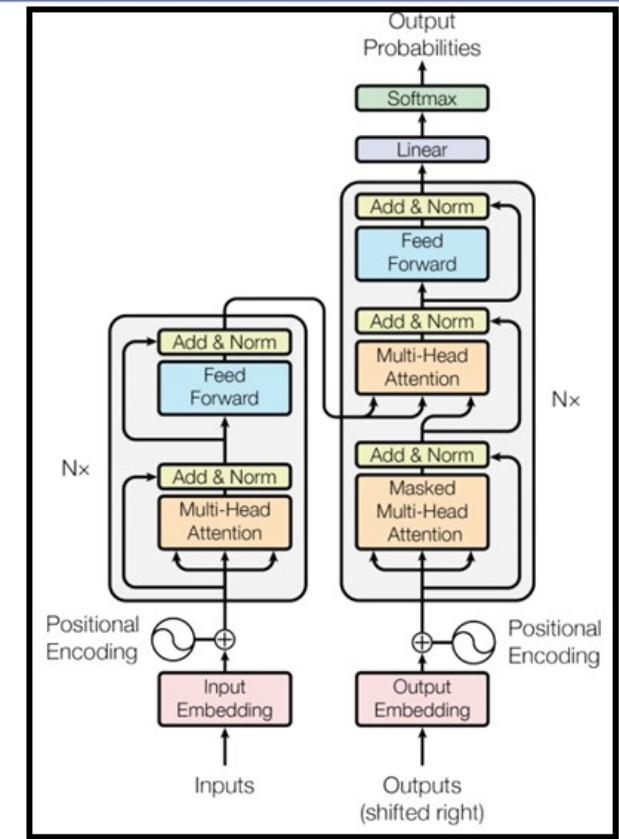
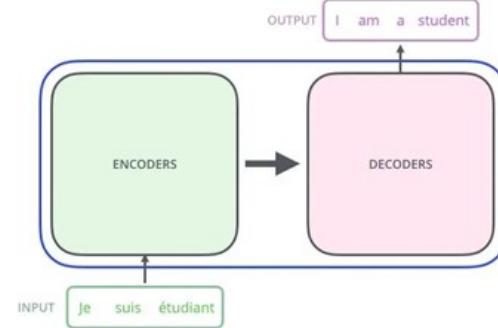
46

# Transformer

Transformer在2017年由Google的团队提出，完全使用Attention机制，而没有传统的RNN或CNN网络架构

Transformer也是Encoder-Decoder模型：

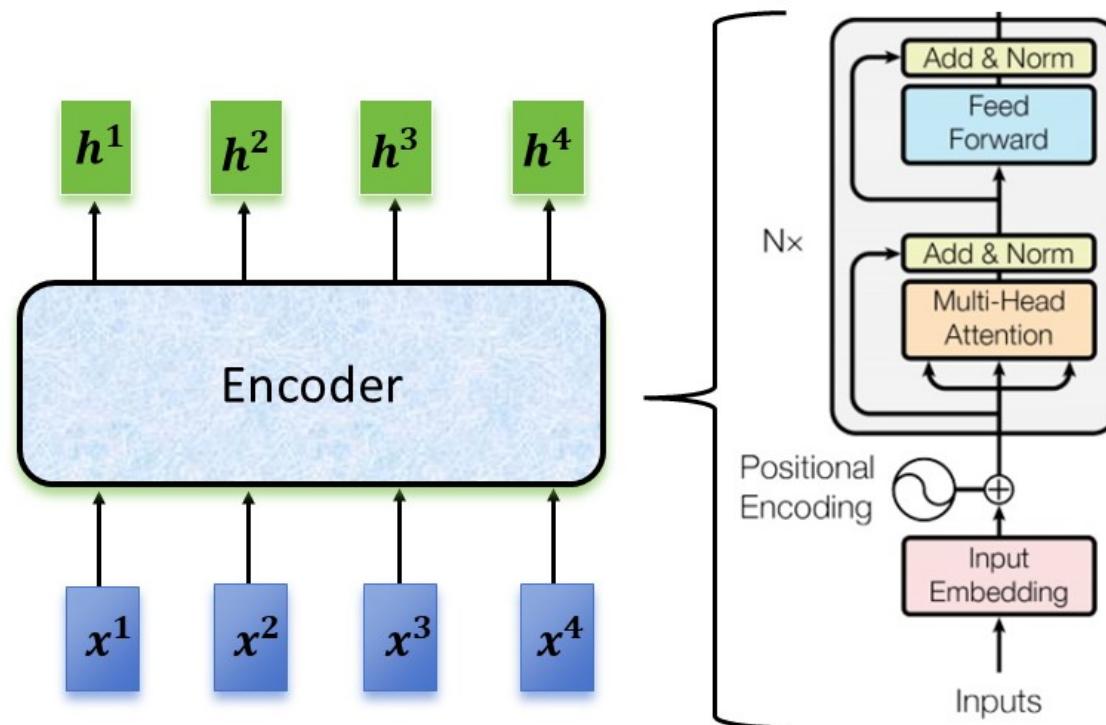
- Embedding
- Positional Encoding
- Self-Attention
- Multi-Head Attention
- Masked Self-Attention
- Cross-Attention



[1] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

47

# Transformer-Encoder

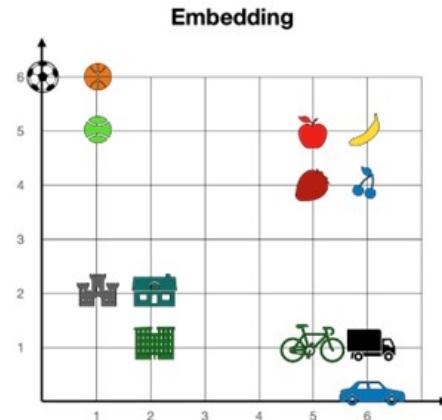


48

# Embedding

将离散数据映射为连续向量，捕捉潜在关系

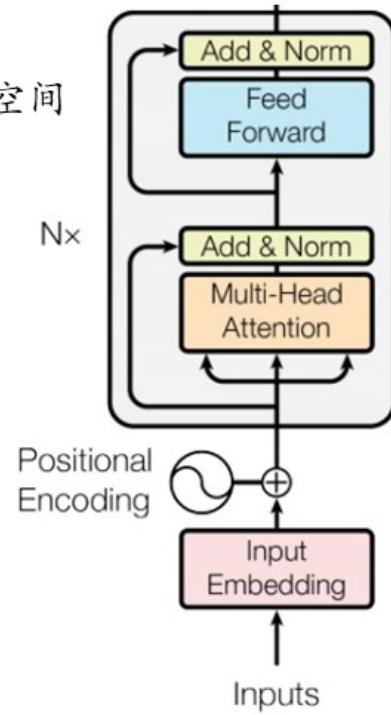
能够捕捉到物体之间的相似性和关系，在映射到高维特征空间后，相似的物体在空间中会聚集在一起，而不同的物体会被分隔开。



Embedding

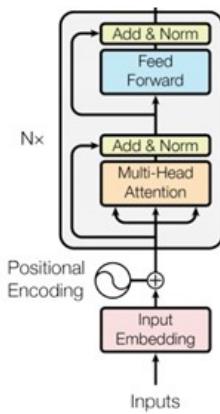
Write	→	2.13	-0.42	...	-1.03
A	→	0.91	0.56	...	0.23
story	→	-1.56	1.34	...	0.14
.	→	1.42	-1.32	...	-2.41

. good -0.1242 -0.0674 -0.1430 -0.0005 -0.0345 ...  
. day 0.0320 0.0381 -0.0299 -0.0745 -0.0624 ...  
. three 0.0304 0.0070 -0.0708 0.0689 -0.0005 ...  
. know -0.0370 -0.0138 0.0392 -0.0395 -0.1591 ...

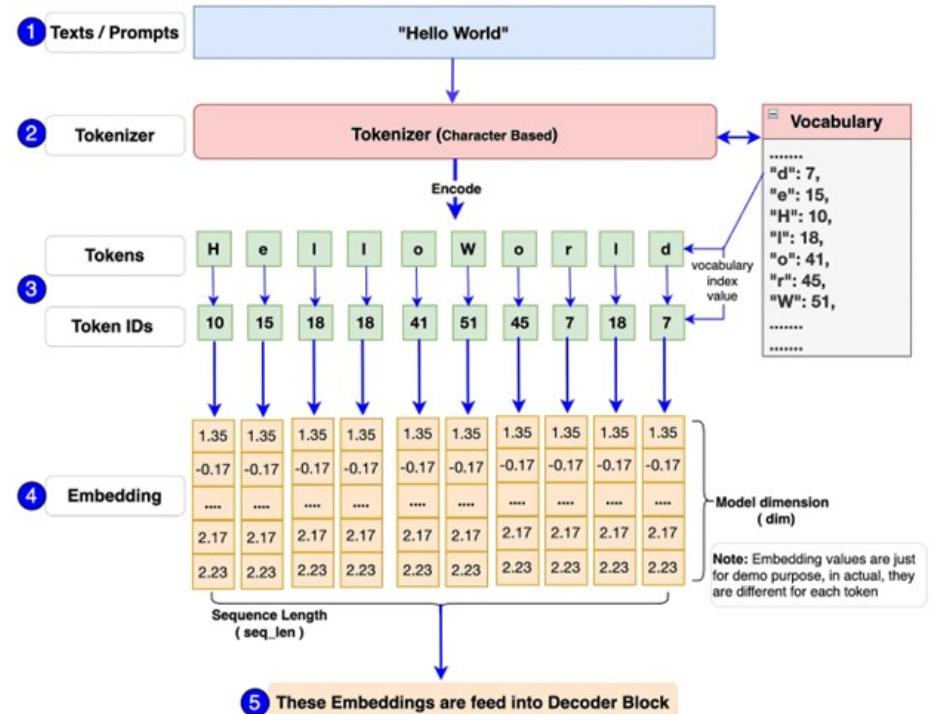


# Embedding

在嵌入层将输入词序列转换为向量表示：



- 1. 词元化 (Tokenization):** 将文本切分为基本单元（词元）。
- 2. 索引化 (Indexing):** 将每个词元映射为唯一的整数ID。
- 3. 嵌入 (Embedding):** 将整数ID查询为高维、密集的浮点数向量。

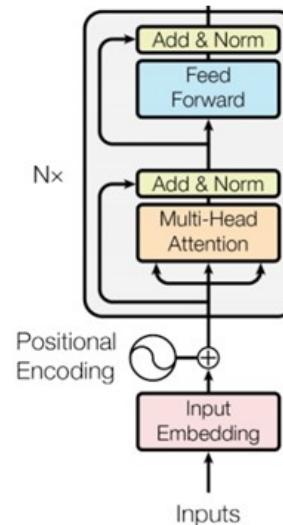


# Positional Encoding

为什么需要位置信息？

在序列，词在句子中不同位置的信息应该是不同的。

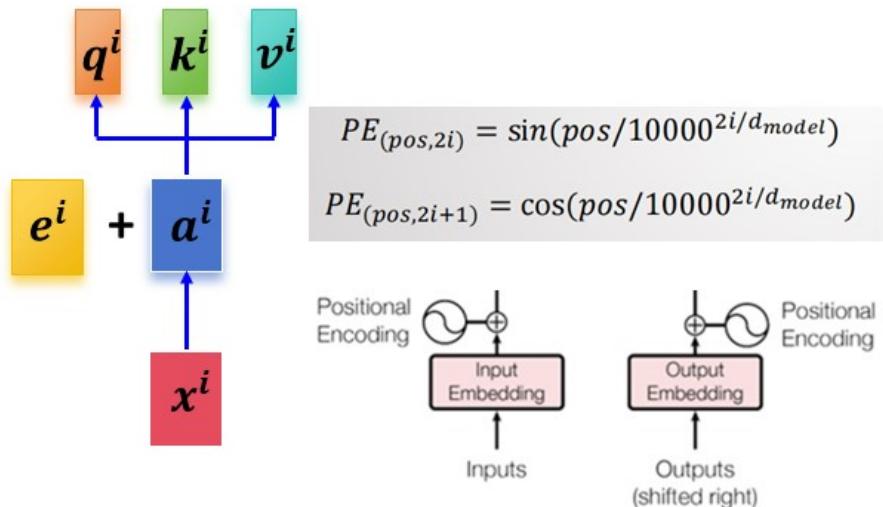
如果没有位置信息：



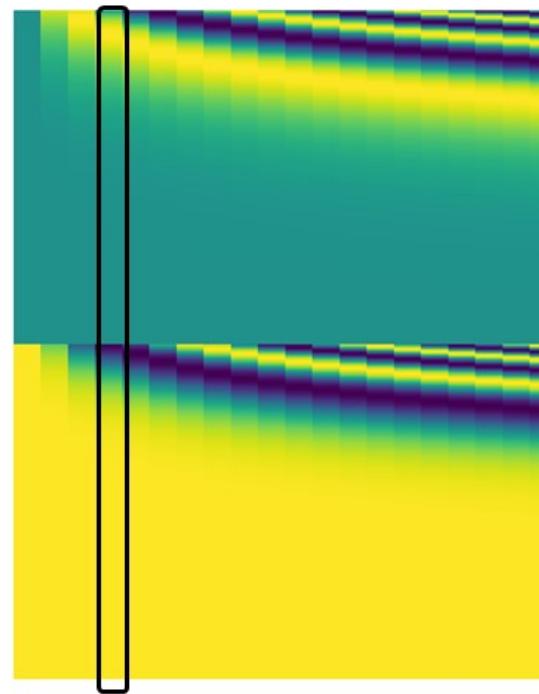
只要输入的词一样，输出的结果是一样的。不符合序列的有序性。

# Positional Encoding

- 在Self-Attention中没有位置信息
- Transformer中给每个位置一个独特的位置向量 $e^i$
- 位置向量 $e^i$ 是可以通过计算规则得到



- 每一列表示的是 $e^i$ 向量



52

# Positional Encoding

什么编码适合用来表示位置信息？

Transformer用的是**正弦和余弦位置编码**

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

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

- **pos** 是当前单词的位置（序列中的索引位置）。
- **i** 是位置编码中的维度索引。
- **$d_{model}$**  是模型的隐藏层维度（即位置编码的维度）。

**优点：**

1. 正弦和余弦函数的周期性使得它们特别适合表示**相对位置关系**

$$\sin(\alpha + \beta) = \sin \alpha \cos \beta + \cos \alpha \sin \beta$$

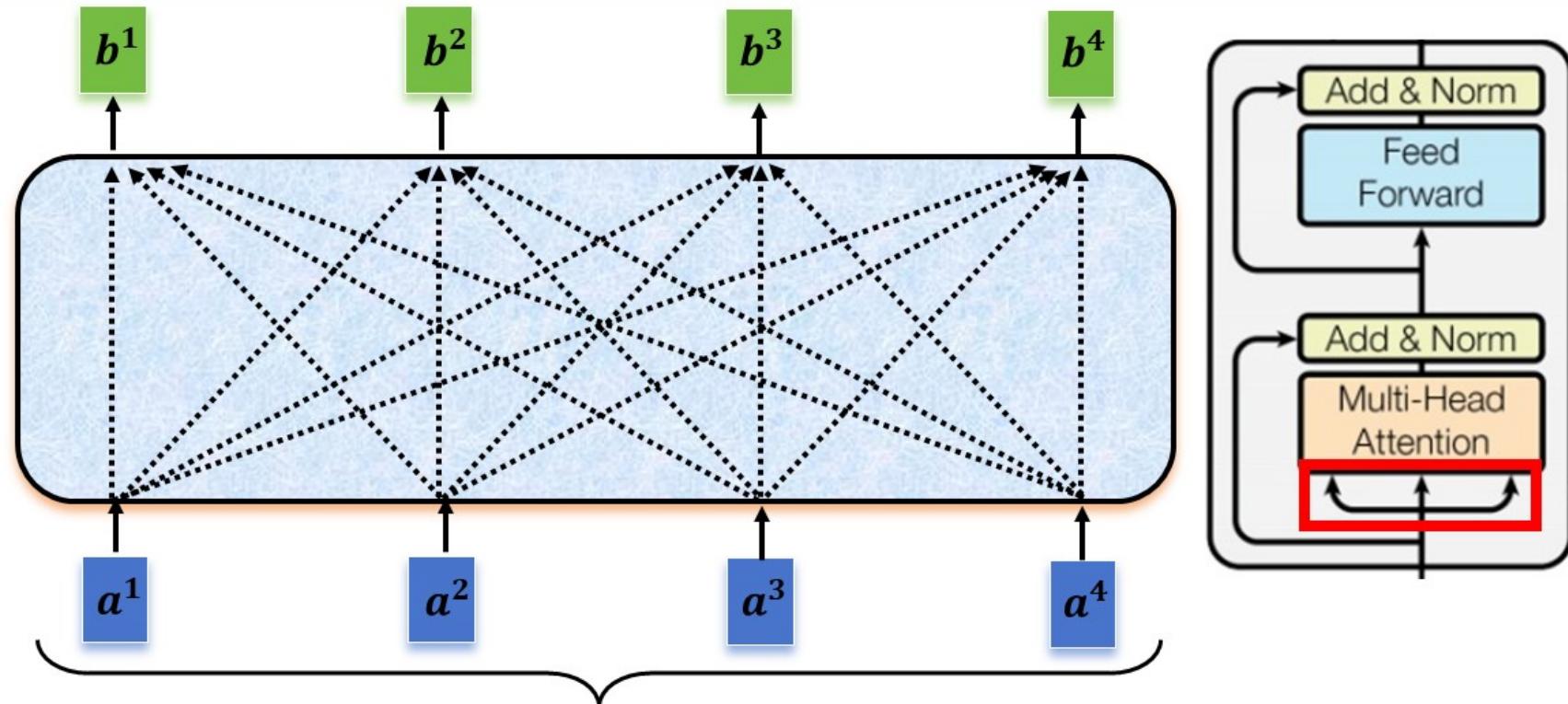
$$\cos(\alpha + \beta) = \cos \alpha \cos \beta - \sin \alpha \sin \beta$$

即：位置  $\alpha + \beta$  的向量可以表示层位置  $\alpha$  的向量的线性变换

2. 无参数、易于计算

53

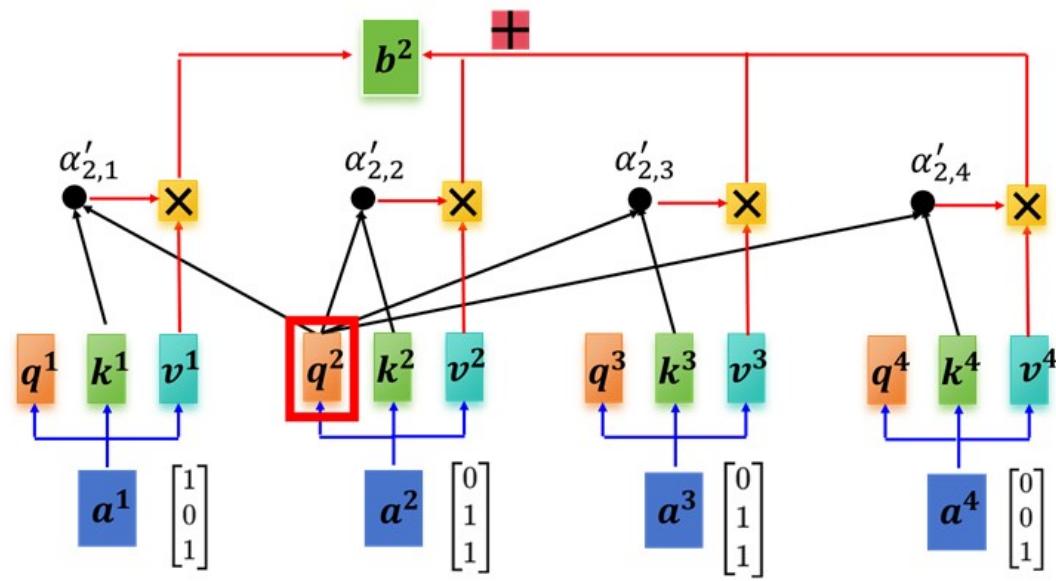
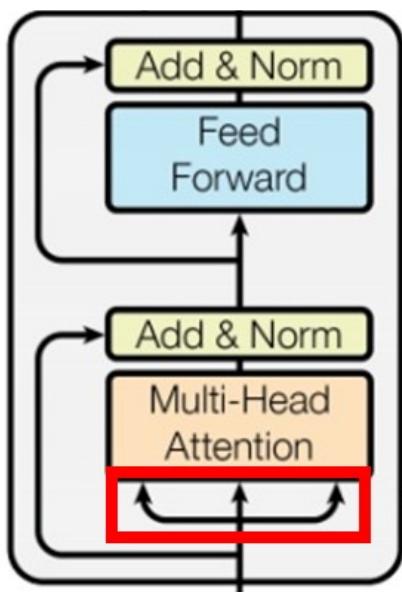
# Self-Attention



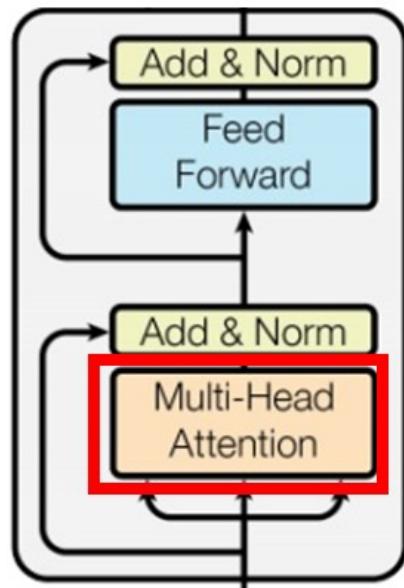
可以来自input或一个hidden layer的输出

54

# Self-Attention



# Multi-Head Self-Attention



人类的感知系统：观察一个物体时，大脑会同时从多个角度处理信息

- **视觉皮层**关注形状和轮廓
- **颜色处理区域**专注于色彩信息
- **运动检测区域**负责追踪物体移动
- **深度感知系统**判断距离和空间关系

每个区域都有自己的"专长"，最后大脑将这些信息整合成完整的认知。

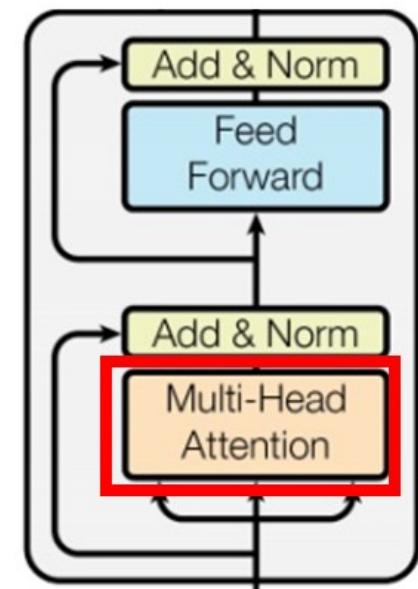
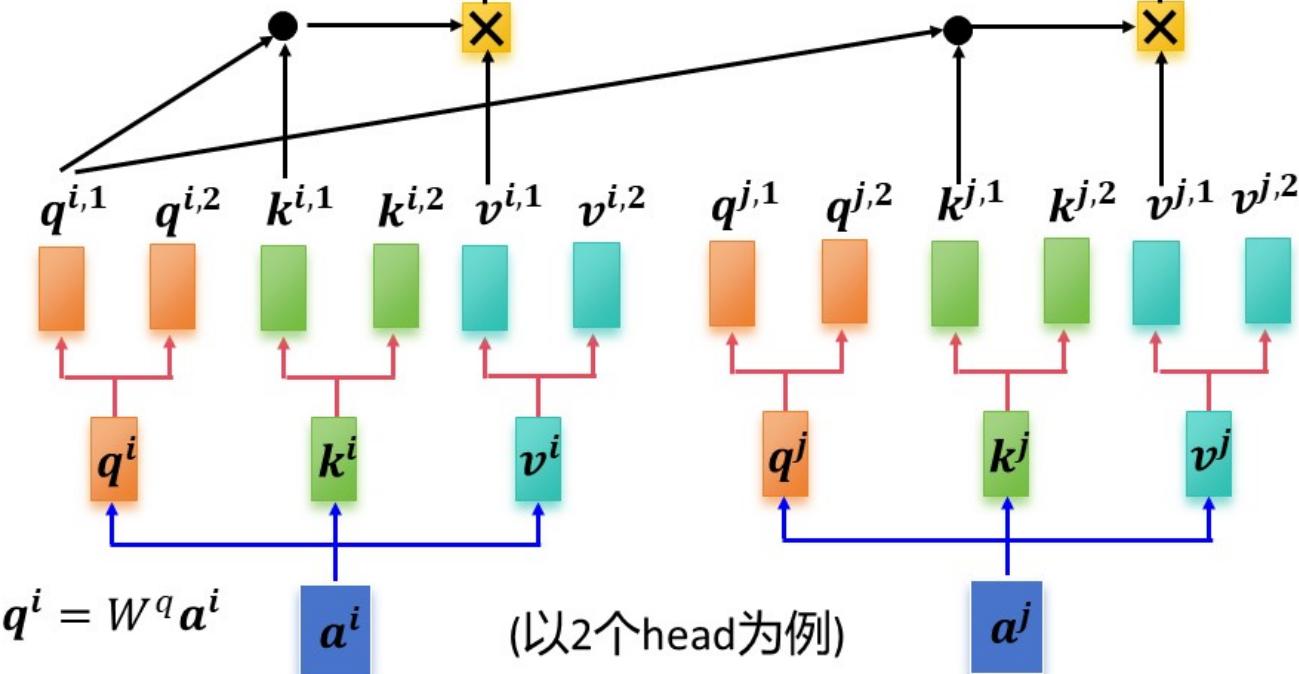
**多头注意力机制：让不同的注意力头专注于不同类型的语言现象，然后将它们的发现组合起来形成更全面的理解。**

- Head 1：专注于语法关系（主谓一致、代词指代等）
- Head 2：专注于语义相似性（词义相关性）
- Head 3：专注于位置关系（距离、顺序）
- Head 4：专注于上下文逻辑（因果关系、时间关系）

## Multi-Head Self-Attention

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

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

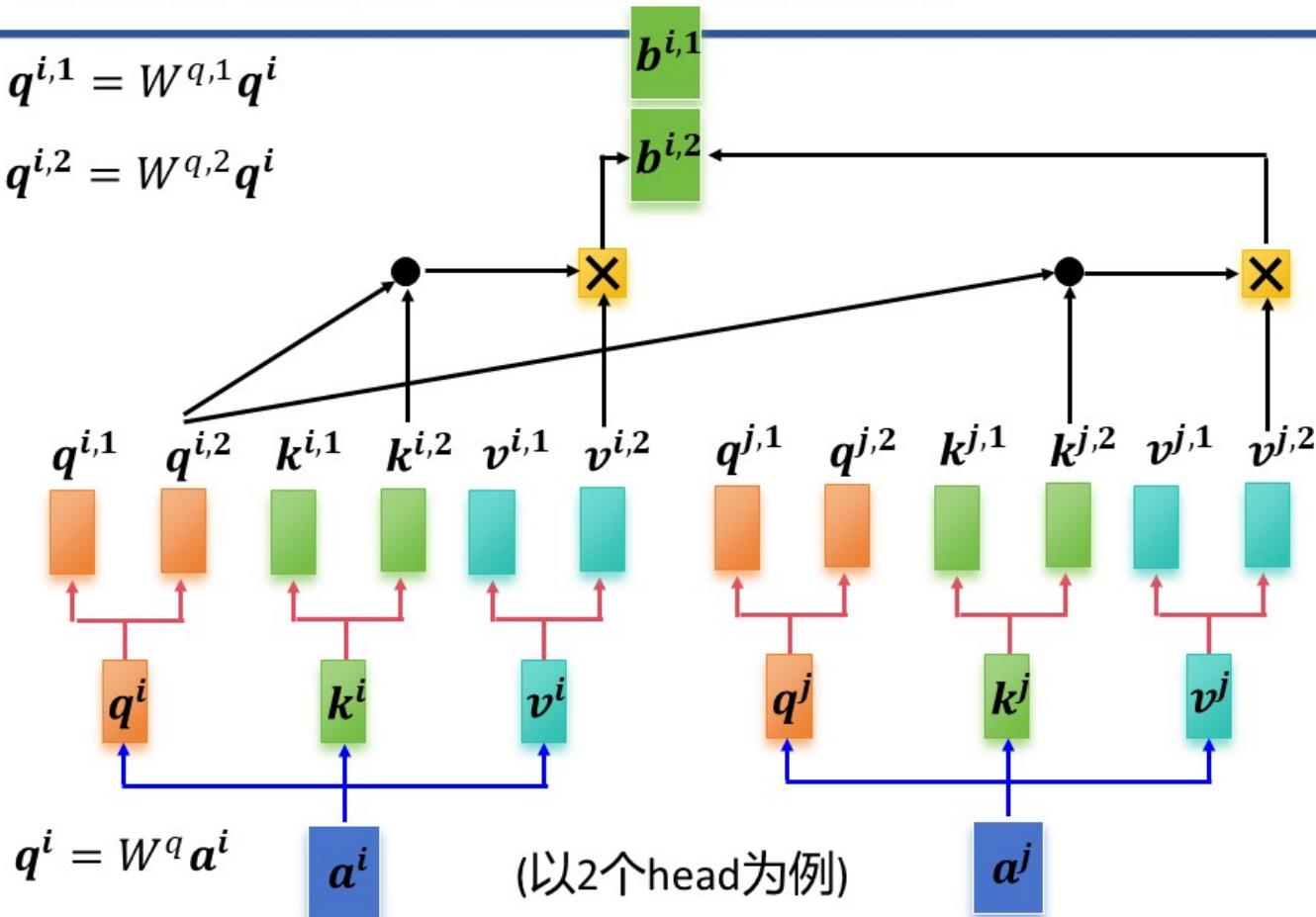


57

## Multi-Head Self-Attention

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

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



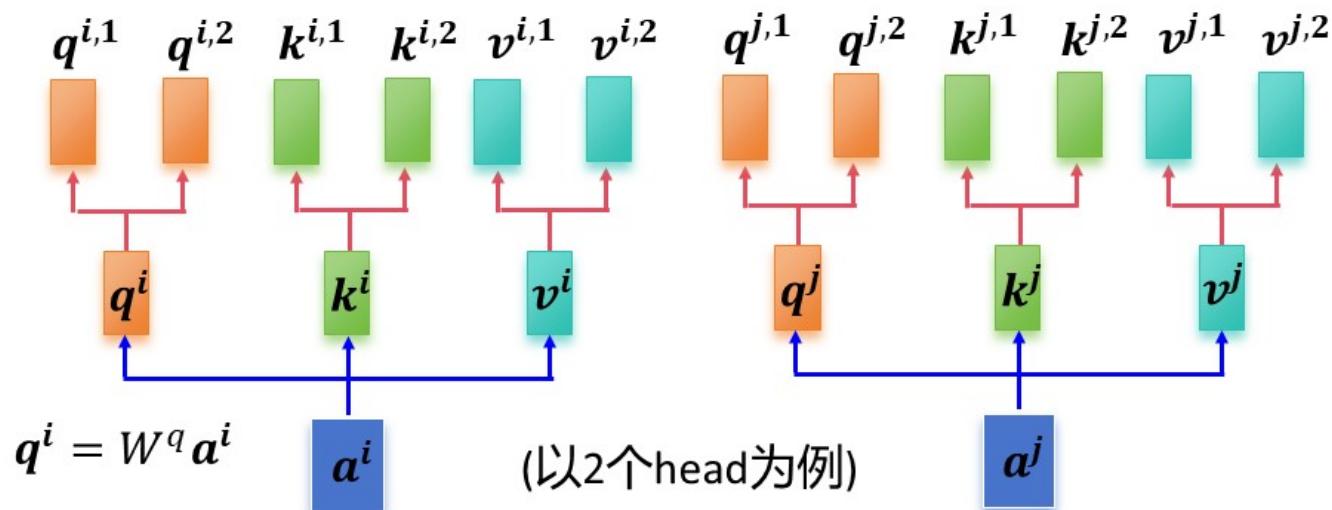
58

# Multi-Head Self-Attention

$$\mathbf{b}^i = W^O \begin{bmatrix} \mathbf{b}^{i,1} \\ \mathbf{b}^{i,2} \end{bmatrix}$$

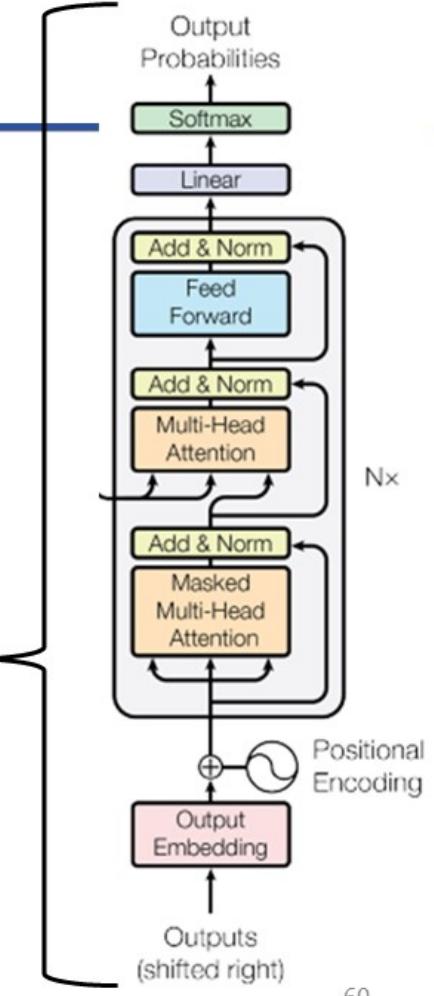
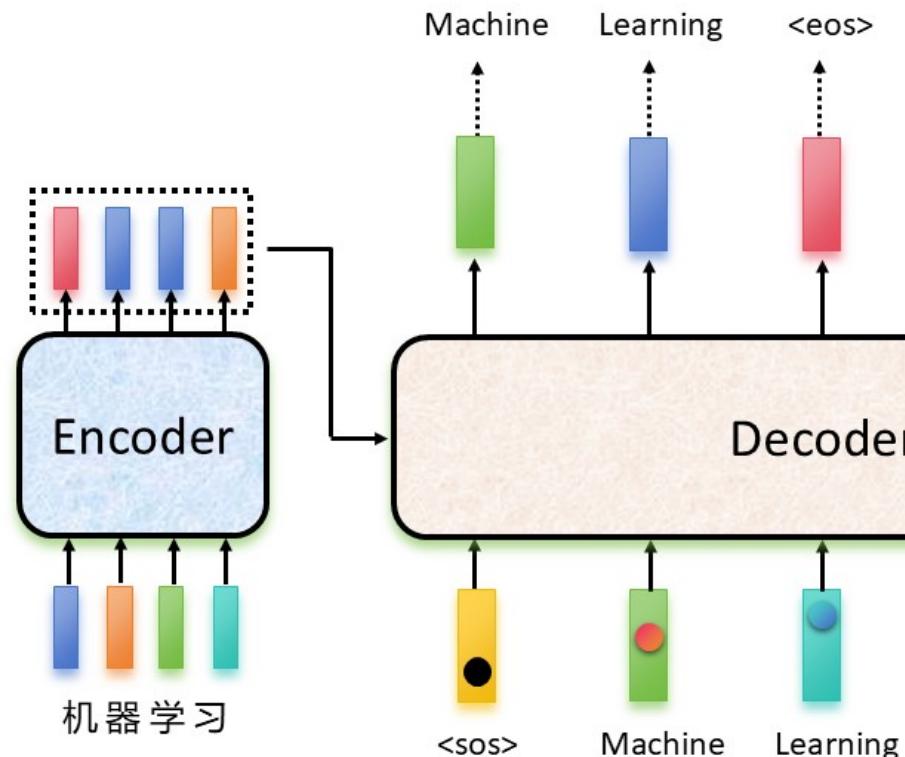
作用:

1. 并行计算
2. 同时捕捉输入序列在不同子空间中的信息,从而增强模型的表达能力



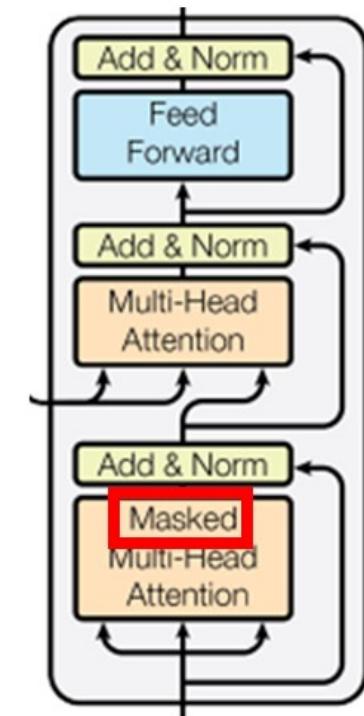
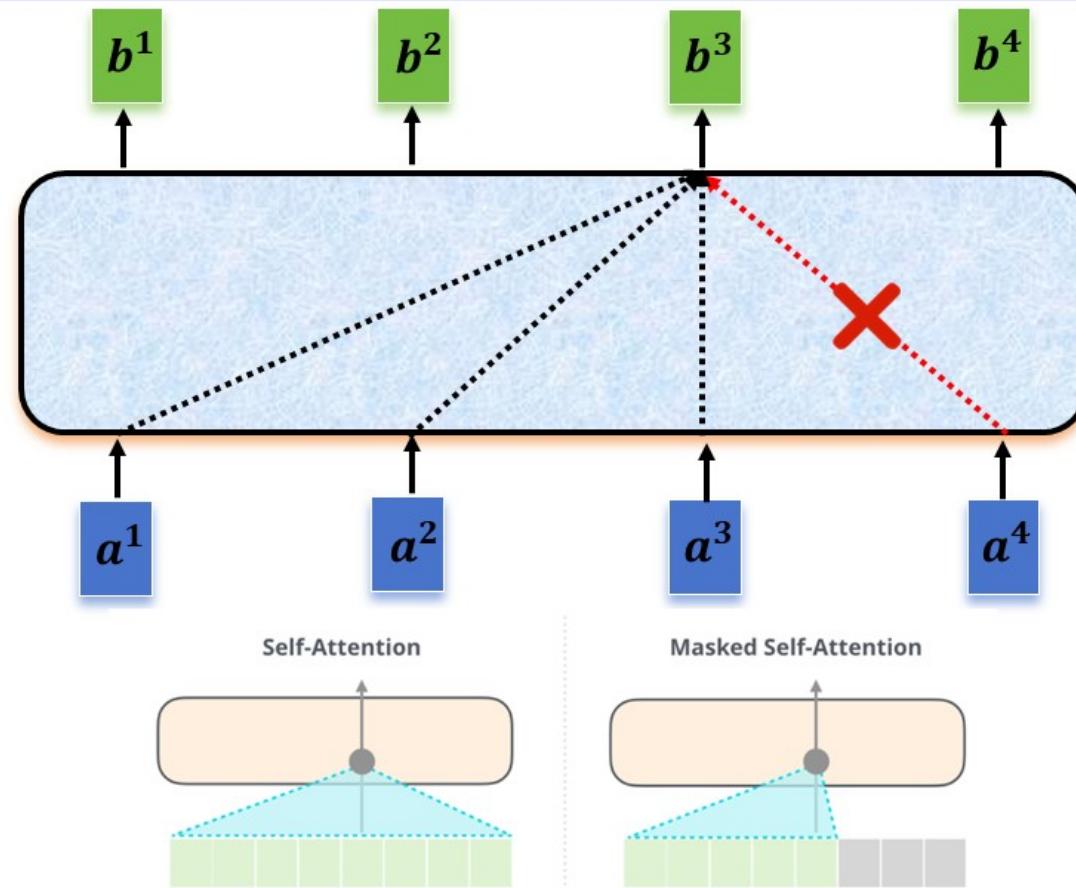
59

# Transformer-Decoder



60

# Masked Self-Attention



61

# Masked Self-Attention

$$A_{\text{masked}} = A + M$$

$$A' = \text{softmax}(A_{\text{masked}})$$

	Your	journey	starts	with	one	step
Your	0.19	0.16	0.16	0.15	0.17	0.15
journey	0.20	0.16	0.16	0.14	0.16	0.14
starts	0.20	0.16	0.16	0.14	0.16	0.14
with	0.18	0.16	0.16	0.15	0.16	0.15
one	0.18	0.16	0.16	0.15	0.16	0.15
step	0.19	0.16	0.16	0.15	0.16	0.15



注意力分数矩阵  $A$

0	-∞	-∞	-∞	-∞	-∞
0	0	-∞	-∞	-∞	-∞
0	0	0	-∞	-∞	-∞
0	0	0	0	-∞	-∞
0	0	0	0	0	-∞
0	0	0	0	0	0

掩码矩阵  $M$



	Your	journey	starts	with	one	step
Your	1.0					
journey	0.55	0.44				
starts	0.38	0.30	0.31			
with	0.27	0.24	0.24	0.23		
one	0.21	0.19	0.19	0.18	0.19	
step	0.19	0.16	0.16	0.15	0.16	0.15

新的分数矩阵

为什么使用掩码?

62

# Masked Self-Attention

$$A_{masked} = A + M$$

$$A' = \text{softmax}(A_{masked})$$

	Your	journey	starts	with	one	step
Your	1.0					
journey	0.55	0.44				
starts	0.38	0.30	0.31			
with	0.27	0.24	0.24	0.23		
one	0.21	0.19	0.19	0.18	0.19	
step	0.19	0.16	0.16	0.15	0.16	0.15

新的分数矩阵

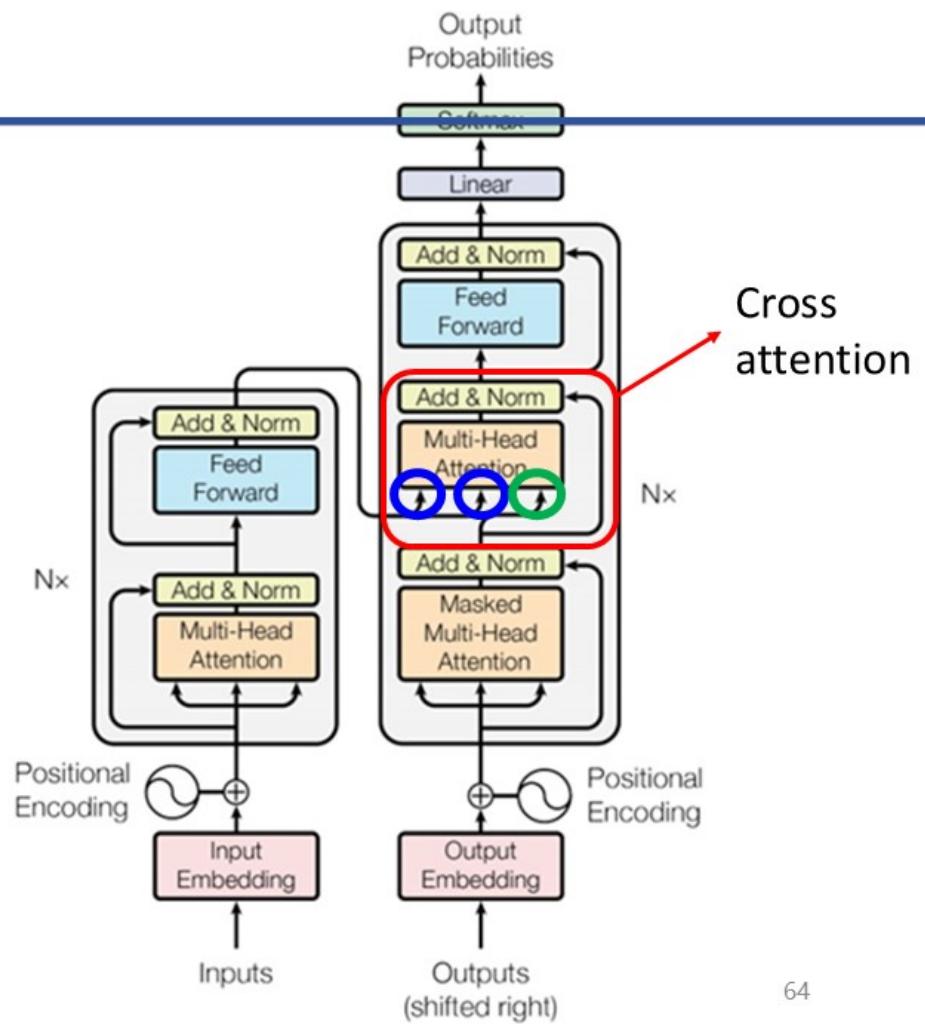
## 为什么使用掩码？

自回归：解码器的任务是预测下一个词。  
在  $t$  时刻，它需要根据已经生成的序  
列  $y_1, \dots, y_{t-1}$  来预测  $y_t$ 。  
掩码只能让当前位置看到过去的信息。

# Cross-Attention

Cross-Attention发生在Encoder  
编码的信息传输到Decoder时

即：此时的输入不再是以自我  
为输入，而是有两部分输入



# Cross-Attention

## 为什么使用交叉注意力?

对于解码器来说，在生成每个词时，需要知道输入句子的哪些部分最相关。  
这时就需要一个“注意力机制” — 它可以让解码器“回头看”编码器的输出。

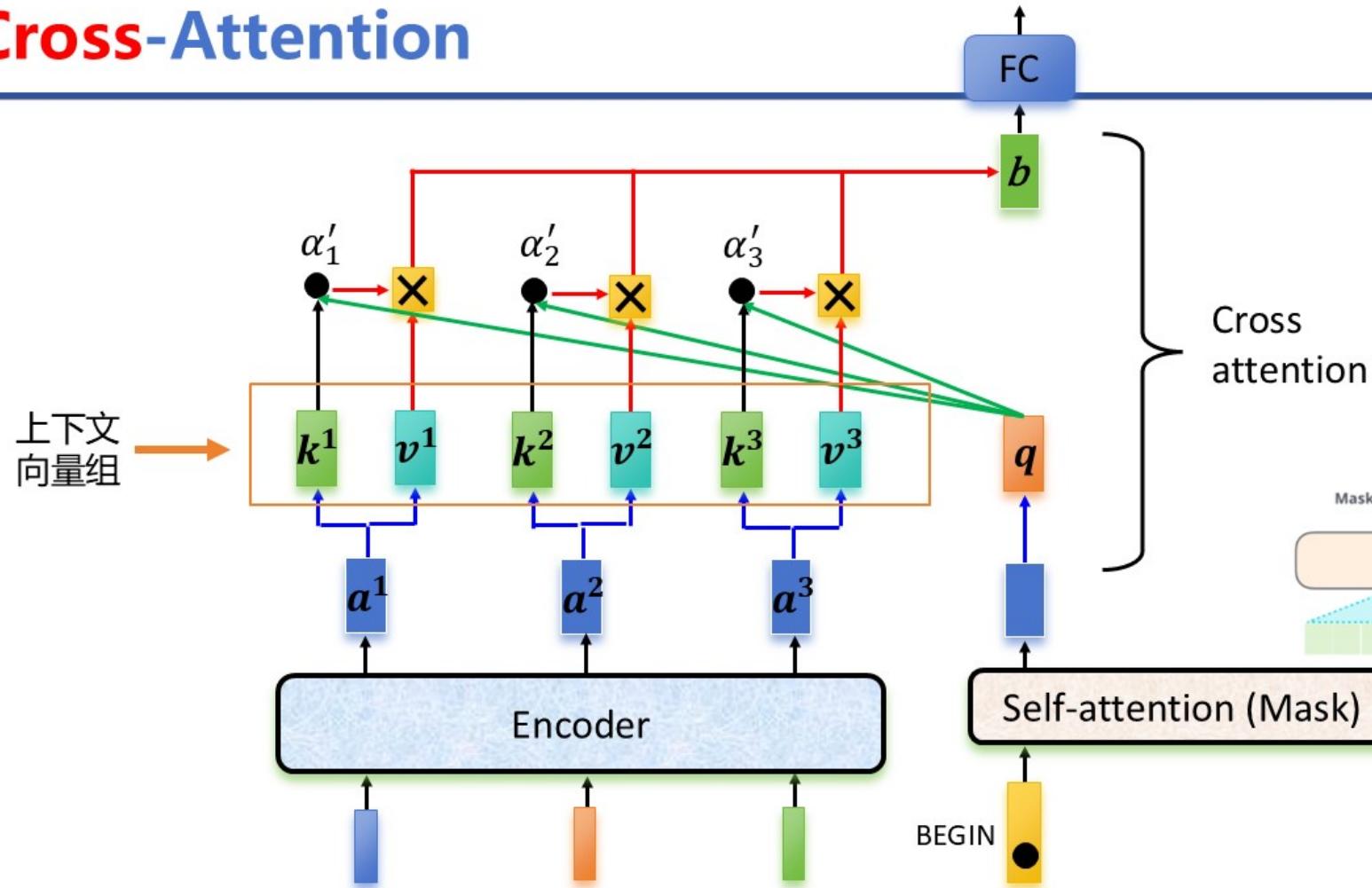
例如：在翻译中，[The boy is eating an apple.](#)

Encoder通过自注意力（Self-Attention）学会内部结构，比如：“boy”和“eating”有语义联系（谁在吃），“an”和“apple”关联（限定词 + 名词）

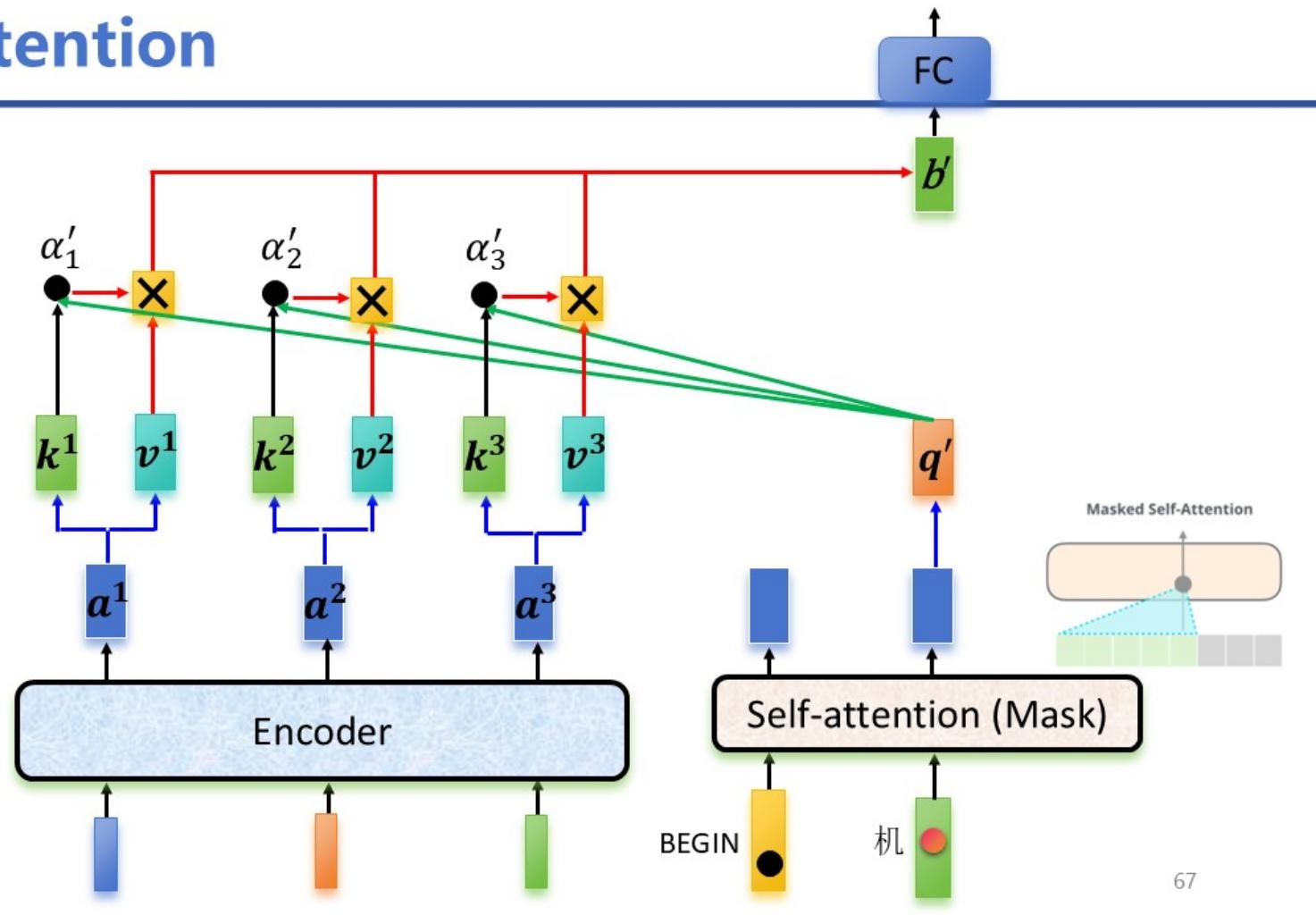
而对于Decoder来说：

阶段	Decoder当前输出	Decoder查询(Q) 想问的问题	Encoder提供的信息(K/V)	主要关注的词	输出
1	(空)	句首是什么？	The, boy, is, eating, an, apple	The	那个
2	那个	主语是谁？	...	boy	男孩
3	那个男孩	动作是什么？	...	is eating	正在吃
4	那个男孩正在吃	宾语是什么？	...	an apple	一个苹果

## Cross-Attention



## Cross-Attention



# 区别与联系

	区别	应用
Self-Attention	QKV由同一个输入a通过矩阵产生，同时可以互相影响	一段句子输入到Encoder里面，句子里面的词相互影响
Masked Self-Attention	QKV由同一个输入a通过矩阵产生，但是Q不能被相对序列后的KV向量影响	在一个个词通过Decoder生成的过程中，后生成的词不能对前生成的词产生影响
Cross-Attention	Q和KV由不同的输入产生	Decoder中下一个词（如翻译）的生成要借鉴Encoder中的KV矩阵

# Transformer

Encoder和Decoder分别在训练和推理阶段  
如何运行？

# Transformer

阶段	过程	Encoder 输入	Encoder 输出	Decoder 输入	Decoder 输出	目标
训练	一次性并行计算	[我, 爱, 你]	上下文向量	[<sos>, I, love, you]	每个位置的概率分布	[I, love, you, <eos>]
推理	t=1	[我, 爱, 你]	上下文向量	[<sos>]	$P(\text{词} \mid <\text{sos}>)$	"I"
(自回归)	t=2	(复用)	(复用)	[<sos>, I]	$P(\text{词} \mid <\text{sos}>, \text{I})$	"love"
	t=3	(复用)	(复用)	[<sos>, I, love]	$P(\text{词} \mid <\text{sos}>, \text{I}, \text{love})$	"you"
	t=4	(复用)	(复用)	[<sos>, I, love, you]	$P(\text{词} \mid <\text{sos}>, \text{I}, \text{love}, \text{you})$	<eos>

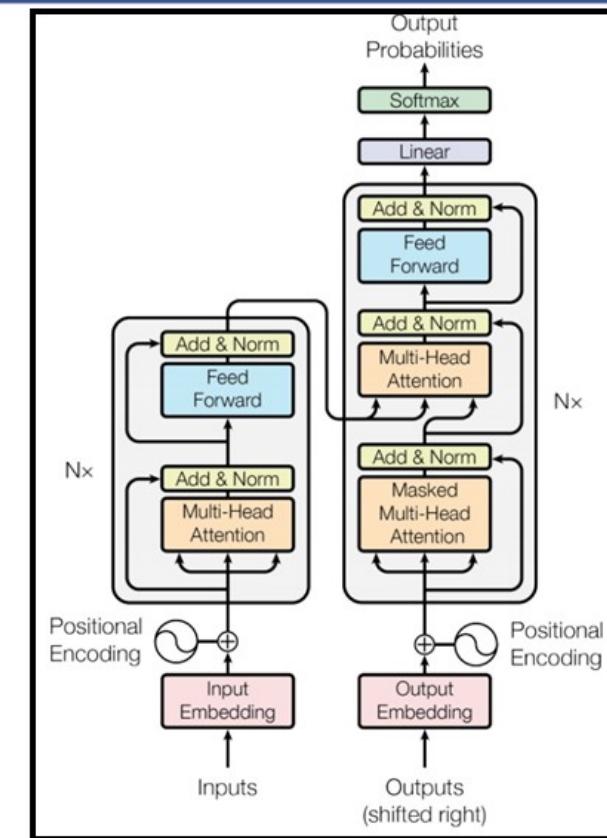
70

# Transformer

Transformer在2017年由Google的团队提出，完全使用Attention机制，而没有传统的RNN或CNN网络架构

Transformer也是Encoder-Decoder模型：

- Embedding
- Self-Attention
- Masked Self-Attention
- Cross-Attention
- Multi-Head Attention
- Positional Encoding



[1] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

71

单选题 1分

Transformer 的生成过程是自回归的，也就是说，它只能逐个 token 地生成输出。因此，在解码器对已生成部分进行处理时，采用了一种特殊的注意力机制：

- A Self-Attention
- B Cross-Attention
- C Masked Self-Attention

# 网络结构与学习范式

## 网络结构：关注模型的内部结构和信息流动方式

- ResNet: 通过残差连接构建深度前馈网络。
- LSTM: 通过门控循环单元处理序列信息。
- Transformer: 通过自注意力模块并行处理全局信息。

## 学习范式：关注模型的训练目标和优化框架

已经接触过：监督学习范式（图像分类、机器翻译）

即将学习：无监督学习范式

- 重构：AE、VAE
- 对抗：GAN
- 扩散：Diffusion

## 监督学习的问题

监督数据的获取非常的费时费力！



ImageNet 的建立是一个极其庞大的人工标注工程，动用了上万名标注者、耗时数年、花费巨大。

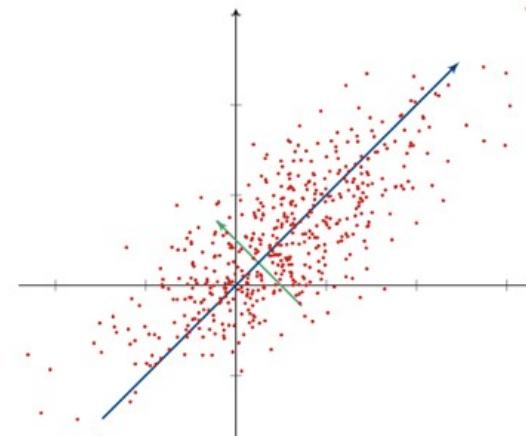
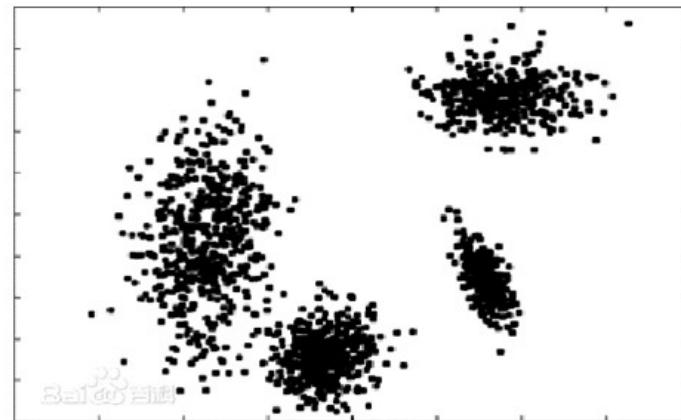
但是现实中其实存在着大量的无标签的数据，如果没有人工打上标签的话，这些标签就不能利用了吗？

# 概念：无监督学习



无监督学习是什么？

- **无监督学习** (Unsupervised Learning, UL) : 从**无标签**的训练数据中学习出一些隐藏在数据中的有价值信息。比如：有效的特征、类别、结构以及概率分布等。



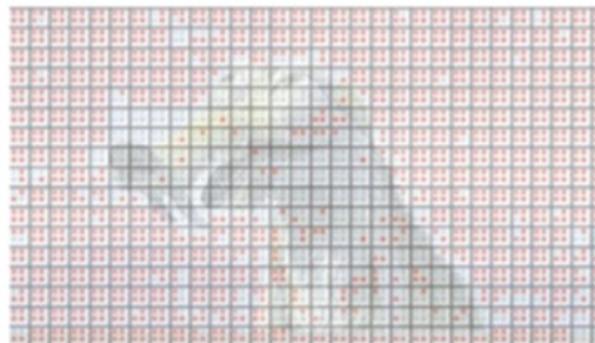
## 概念：无监督学习

图片等数据中存在着很多的**冗余信息**

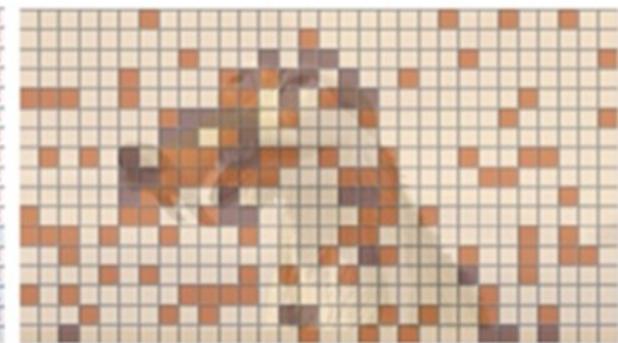
我们用**更少的数据**即可成功表示原来的图片 -> **数据压缩**



原始图像



亮度冗余



色度冗余

- 现在我想训练一个模型来实现数据压缩。
- 如果模型能根据**压缩后的表示准确地还原出原始图像**，那就说明压  
缩表示中保留了足够的重要信息。

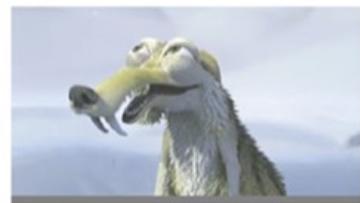
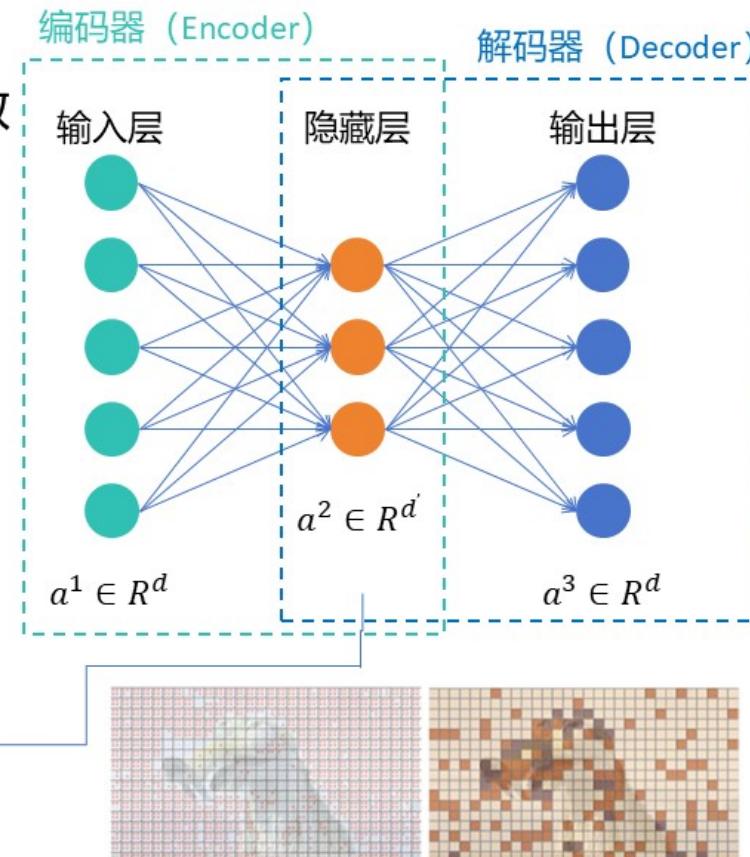
76

# AutoEncoder

自编码器 (Auto-Encoder, AE)  
是通过无监督的方式来学习一组数据的有效编码 (或表示)。



原始图像



原始图像

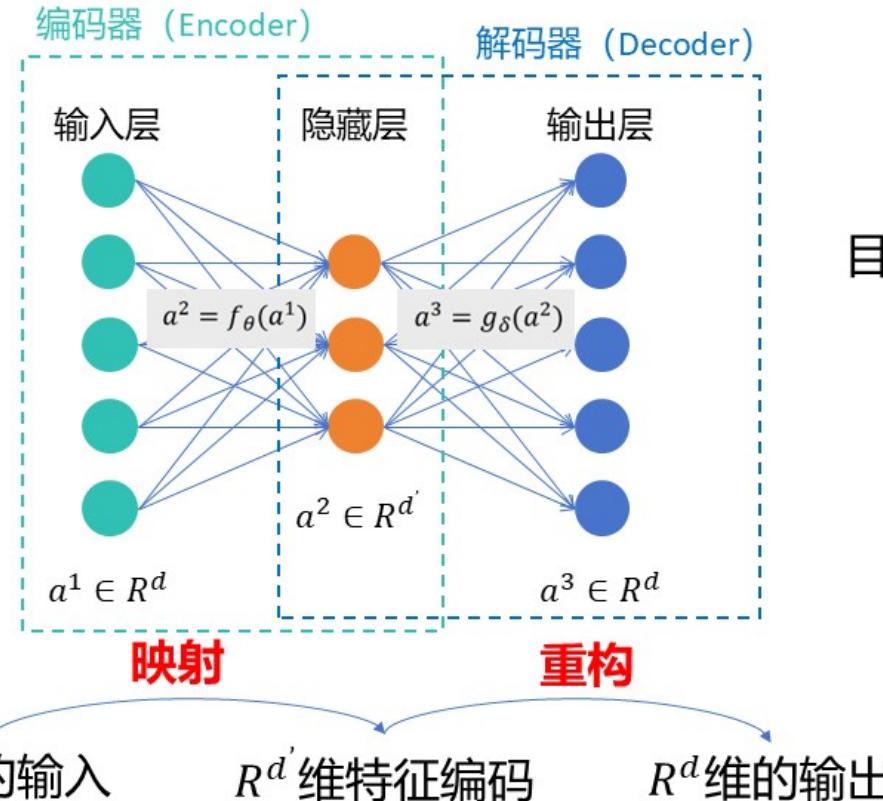
隐藏层 (隐变量) 可以认为是训练数据的**有效表示**。

77

# AutoEncoder

? 如何训练?

编码器将输入  
映射到隐空间，  
压缩为一个隐  
变量，解码器  
重构隐变量。



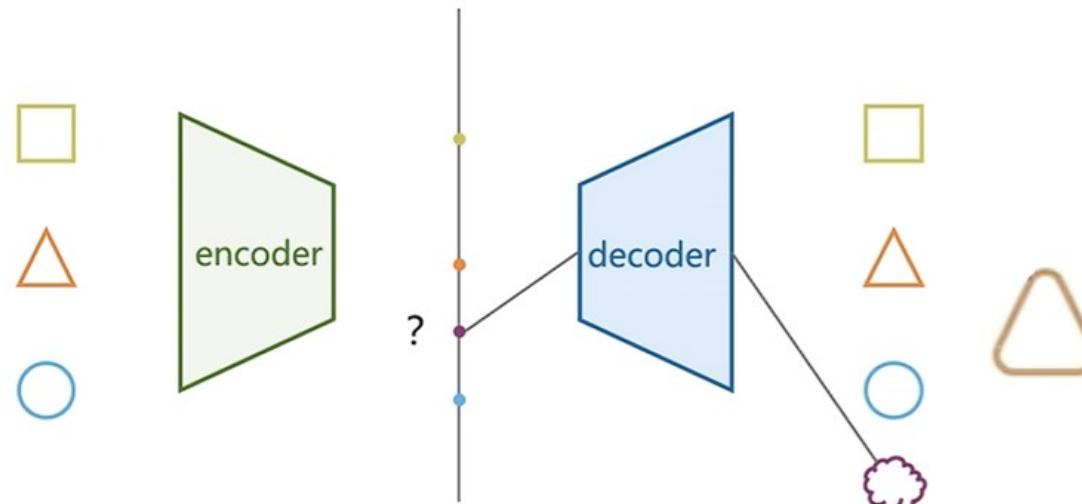
新的学习范式!  
重构

目标：让输出  $a^3$  逼近输入  $a^1$ ：

$$\begin{aligned} Loss &= \sum_{n=1}^N \|a_i^1 - a_i^3\|^2 \\ &= \sum_{n=1}^N \|a_i^1 - g_\delta(f_\theta(a_i^1))\|^2 \end{aligned}$$

# AutoEncoder

选择一个**三角**和**圆**之间的隐变量，能否生成一个**圆的三角**？

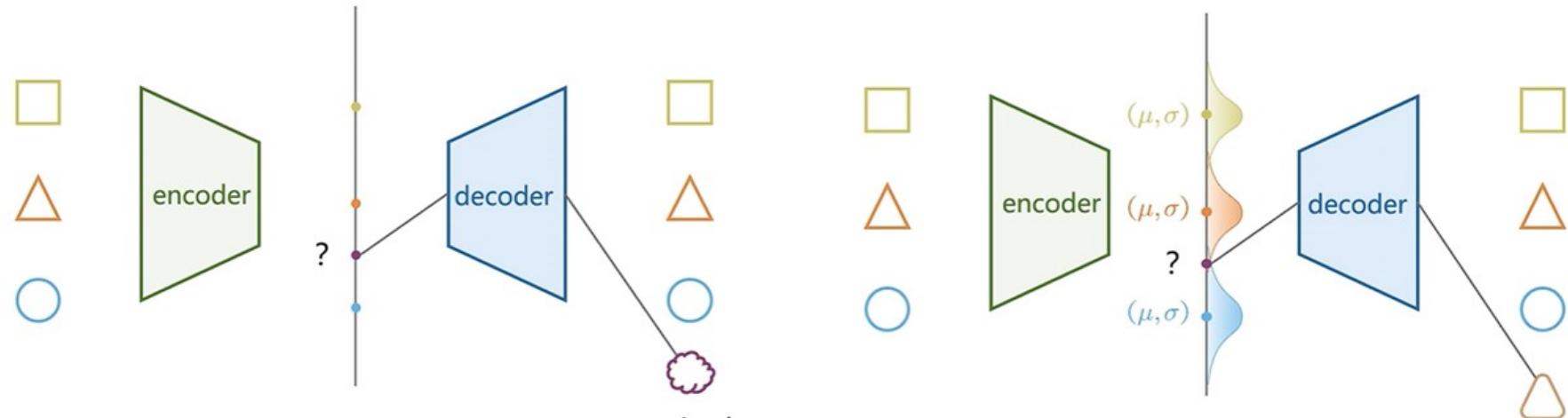


**不行。**结果会生成一个乱七八糟的图片

主要问题在于潜在空间的结构混乱且缺乏规律性

理想的解决方案是构建一个**结构规整**的潜在空间

# Variational AutoEncoder



编码器不再输出固定的数值向量，而是输出一个概率分布（通常是高斯分布）

作用：隐空间由离散点变为连续空间，从而模型能够更平滑地捕捉不同特征之间的关系。

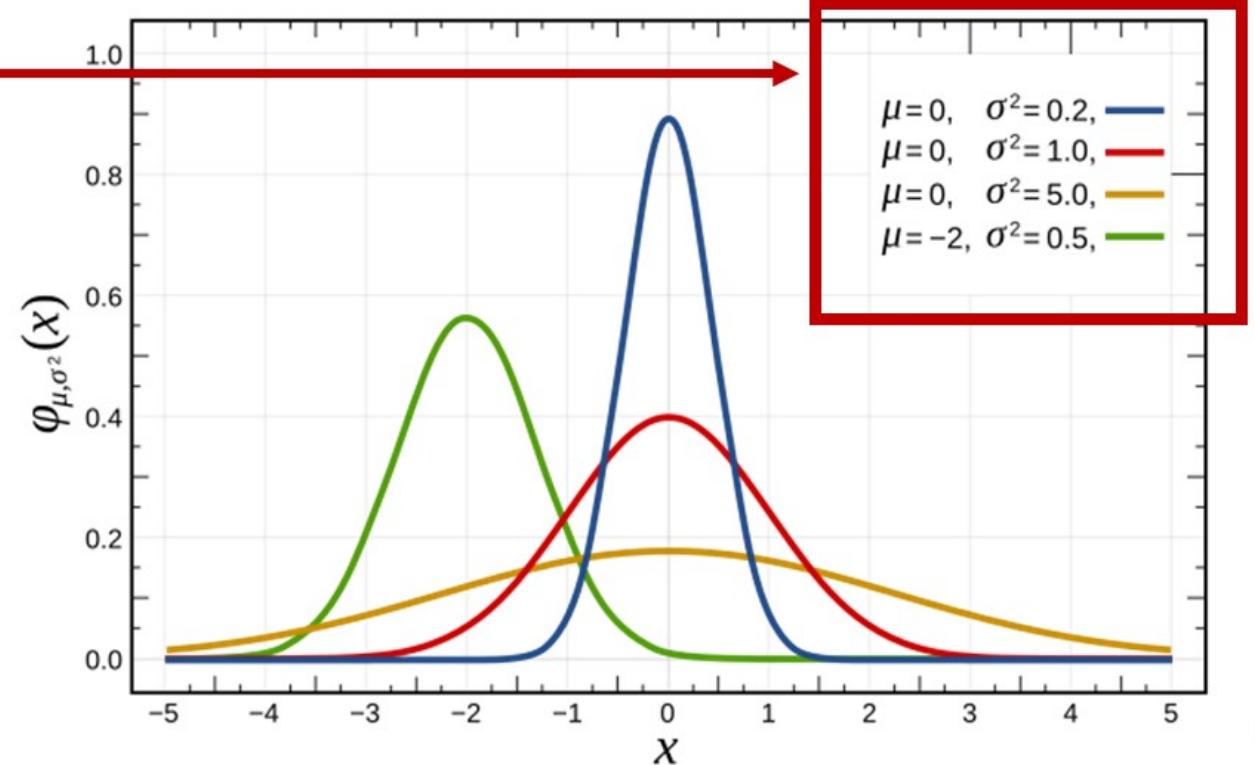
80

# Variational AutoEncoder

如何确定一个高斯函数?

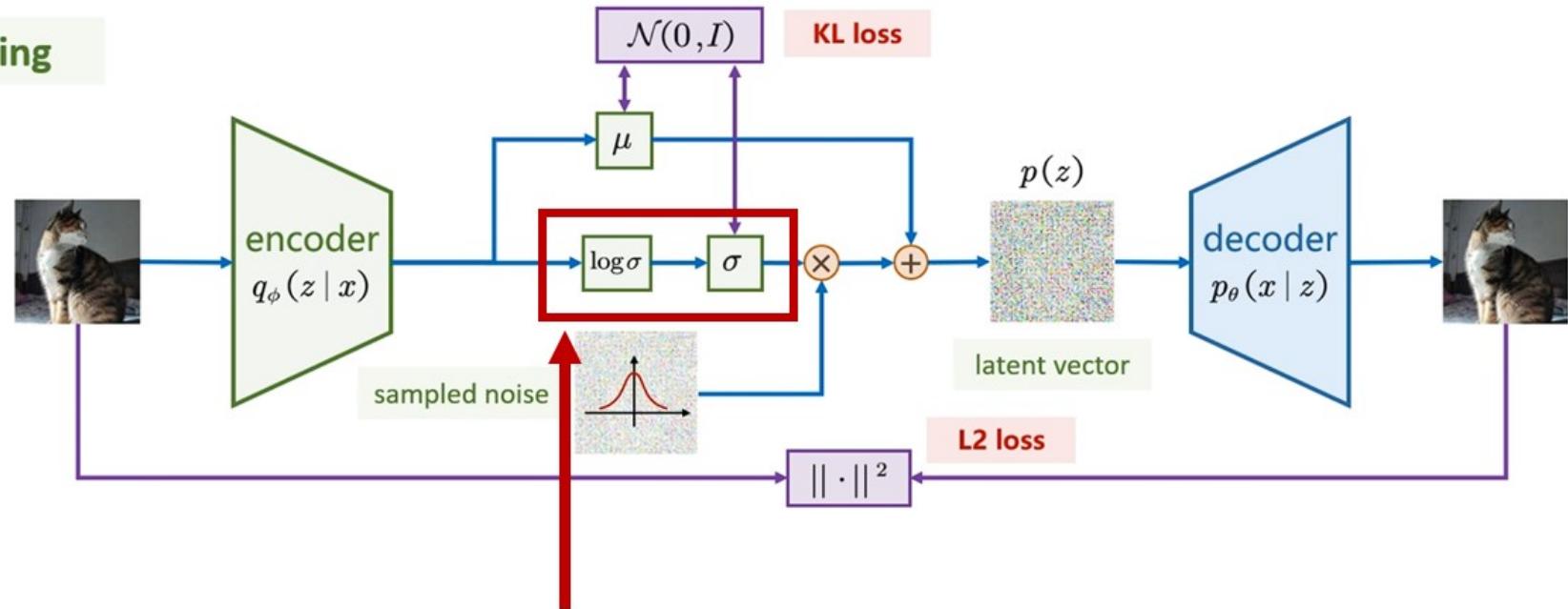
- 一个均值
- 一个方差

因此, VAE模型的目标:  
对于每个隐藏值, 生  
成一个均值和一个方  
差



# Variational AutoEncoder

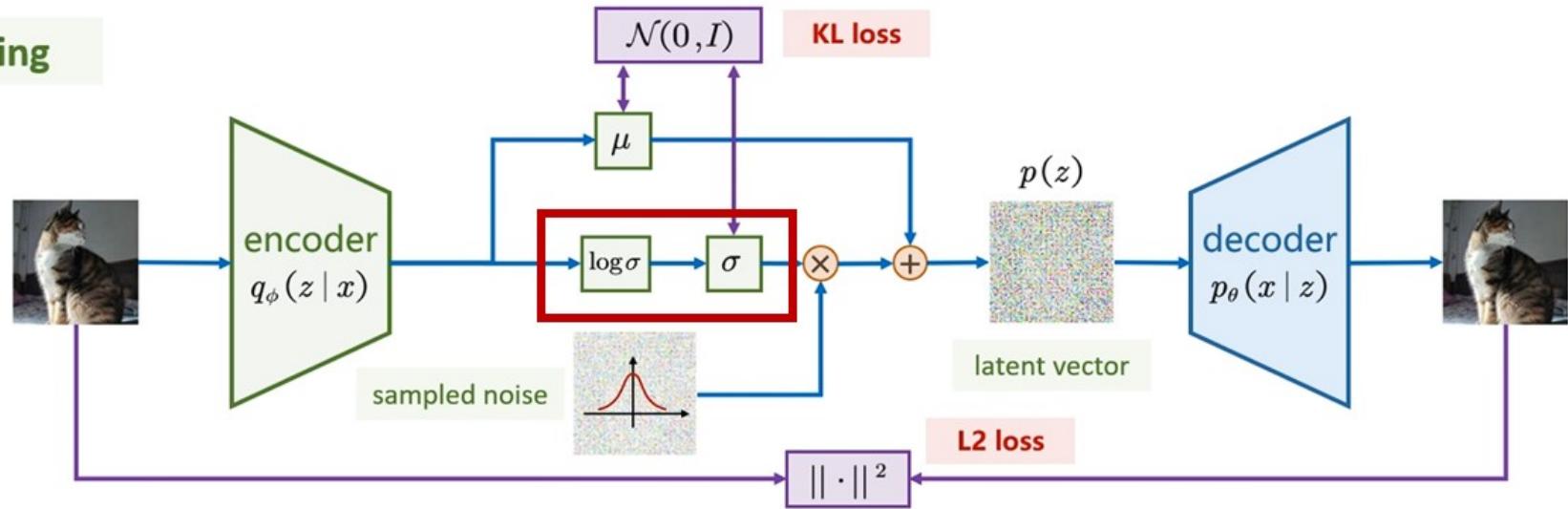
## 1.Training



由于神经网络的输出可能为负值，而方差必须大于0，因此我们通常对模型输出取指数（例如输出对数方差  $\log \sigma$ ），以确保**方差始终为正**。

# Variational AutoEncoder

## 1. Training



$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)\|p(z))$$

重构误差 (同AE)

KL 散度项

约束编码器输出的分布  
不要太偏离整体先验，使隐  
空间结构更有序、更连续。

人为设置的先验概率:  $p(z) = \mathcal{N}(0, I)$

# Variational AutoEncoder

$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)\|p(z))$$

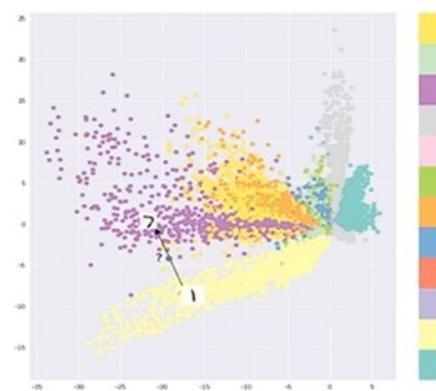
重构误差 (同AE)

KL 散度项

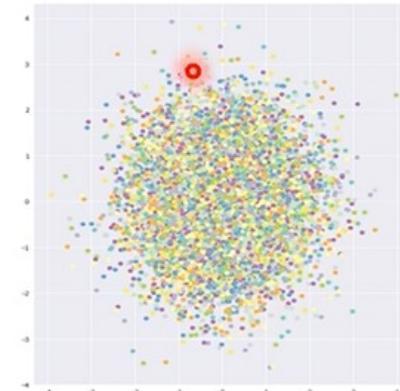
约束编码器输出的分布不要太偏离整体先验，使隐空间结构更有序、更连续。

人为设置的先验概率： $p(z) = \mathcal{N}(0, I)$

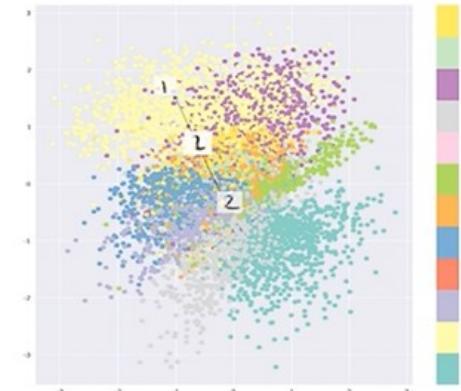
MINIST  
dataset



only reconstruction loss



only KL divergence



combination

87

# Variational AutoEncoder

VAE应用：  
图像生成



88

# 对抗生成模型：GAN

VAE：重构式生成模型

那还有没有其他的生成式模型？

GAN (Generative Adversarial Network) :

对抗生成模型

新的学习范式！  
对抗博弈

生成对抗网络 (GAN, Generative Adversarial Networks) 是由 Ian Goodfellow 等人在 2014 年提出的一种深度学习模型。它采用**对抗思想**，通过两个网络的博弈，生成逼真的数据，广泛应用于图像生成、数据增强、文本生成等领域。

2014年，当时还在蒙特利尔大学读博士的Goodfellow，在一家酒吧微醺后，想到了一种称为“生成对抗网络 (generative adversarial networks)”或GAN的AI技术。尽管这个想法来自几罐啤酒，其仍不失为一个非常优雅的设计：**一个AI尝试创造它认为真实的图像，而第二个AI分析结果，并尝试确定图像是真实还是假的。** Goodfellow说：“你可以把它们当作艺术家和艺术评论家，生成模型想要愚弄艺术评论家——让艺术批评家把它所产生的图像当成真的”。因为第二个AI努力地识别造出来的假图像，所以第一个AI得以学会模仿真实世界。这种方式，是一个单独的AI无法完成的。在这个过程中，这两个神经网络可以将AI推向某一天，电脑宣布独立于他们的人类老师。

# GAN

## GAN的组成：

**生成器：**生成一个逼真的图像  
(目标：**骗过判别器**)



**判别器：**从一个真图像和一个假图像中找出真图像  
(目标：**抓出生成器**)

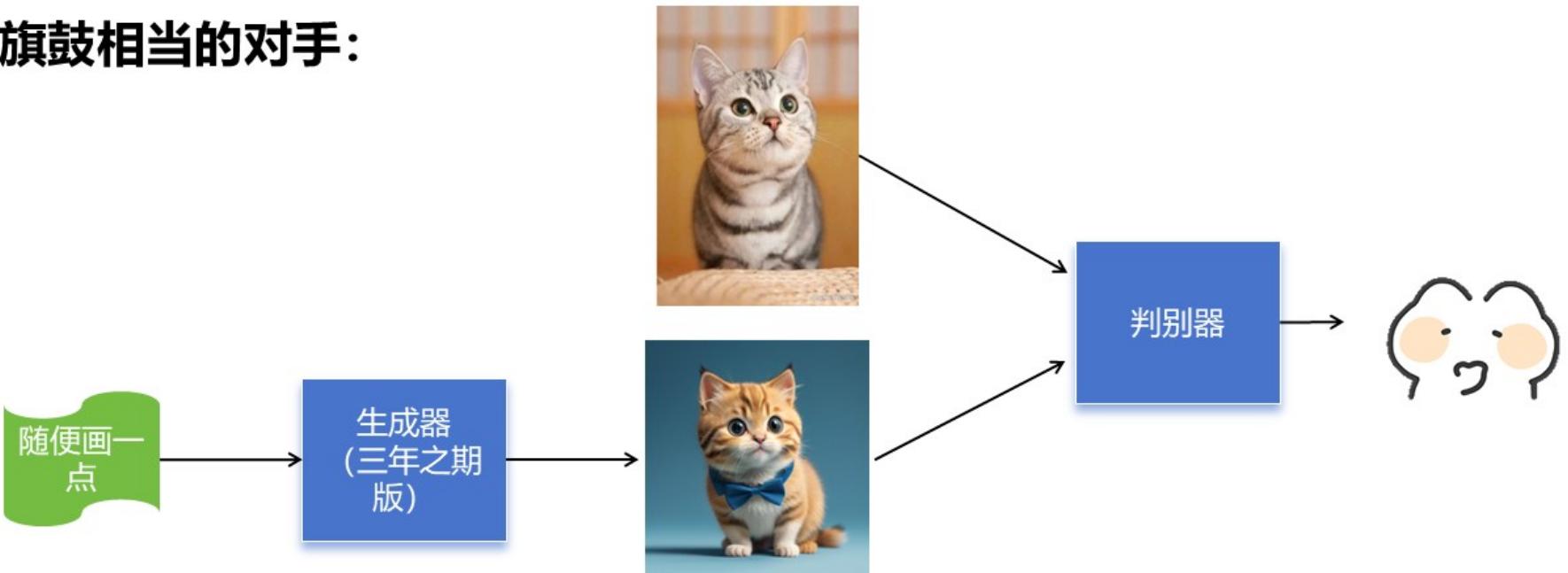
## 生成器太菜时：



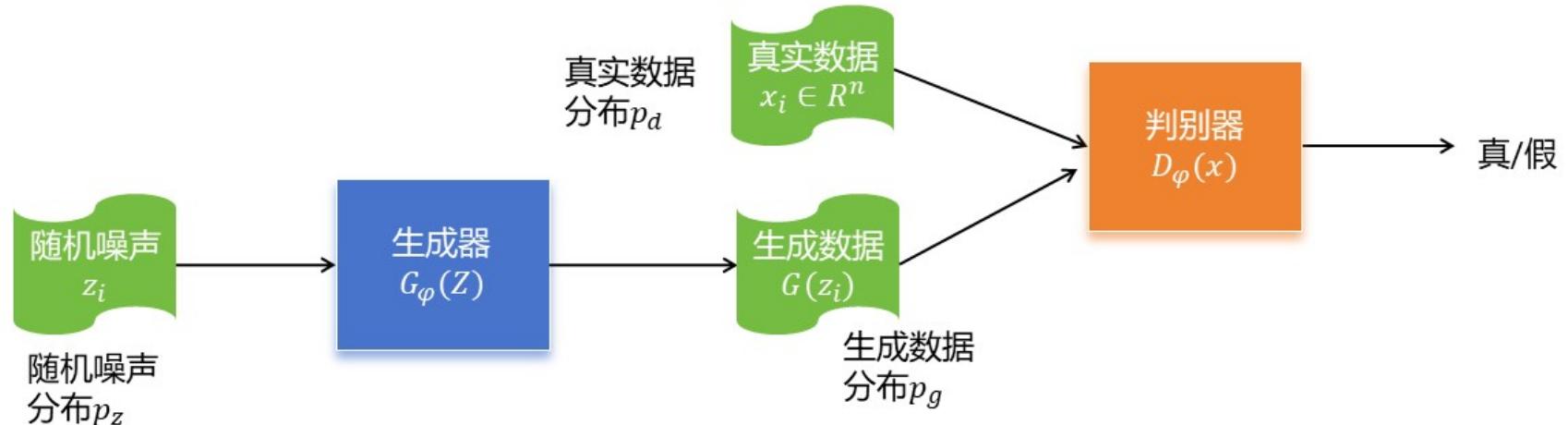
90

# GAN

旗鼓相当的对手：



# GAN



目标函数：

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))]$$

- 对于判别器D来说，鼓励识别真实样本为真，同时鼓励识别生成样本为假
- 对于生成器G来说，让自己的样本不会被识别

92

**对抗博弈训练：**在训练过程中交替优化判别器和生成器

# GAN

当 $p_d = p_g$  即生成数据分布等于真实数据分布时，最优**判别器** $D(x)=\frac{1}{2}$ ，即判别器不能分辨出真实样本和生成样本。  
此时，将**生成器单独取出**，就是一个优秀的生成模型了。



# 扩散式生成模型：Diffusion

- **VAE**: 重构式生成模型
- **GAN**: 对抗生成模型

那还有没有其他的生成式模型？

**DDPM (Denoising Diffusion Probabilistic Models)** : 去噪扩散概率模型

扩散模型最早在2015年的一篇论文中被提出，但由于当时生成效果不理想，未能获得广泛关注。直到2020年，去噪扩散概率模型 (Denoising Diffusion Probabilistic Models, DDPM) 的提出，才将扩散模型的发展推向新高度，并引发了广泛的研究热潮。

## Denoising Diffusion Probabilistic Models

### Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decomposition scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at <https://github.com/jonathanho/diffusion>.

### 1 Introduction

Deep generative models of all kinds have recently exhibited high quality samples in a wide variety of data modalities. Generative adversarial networks (GANs), autoregressive models, flows, and variational autoencoders (VAEs) have synthesized striking image and audio samples [14, 27, 3, 58, 38, 25, 10, 32, 44, 57, 26, 33, 45], and there have been remarkable advances in energy-based modeling and score matching that have produced images comparable to those of GANs [11, 55].



Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

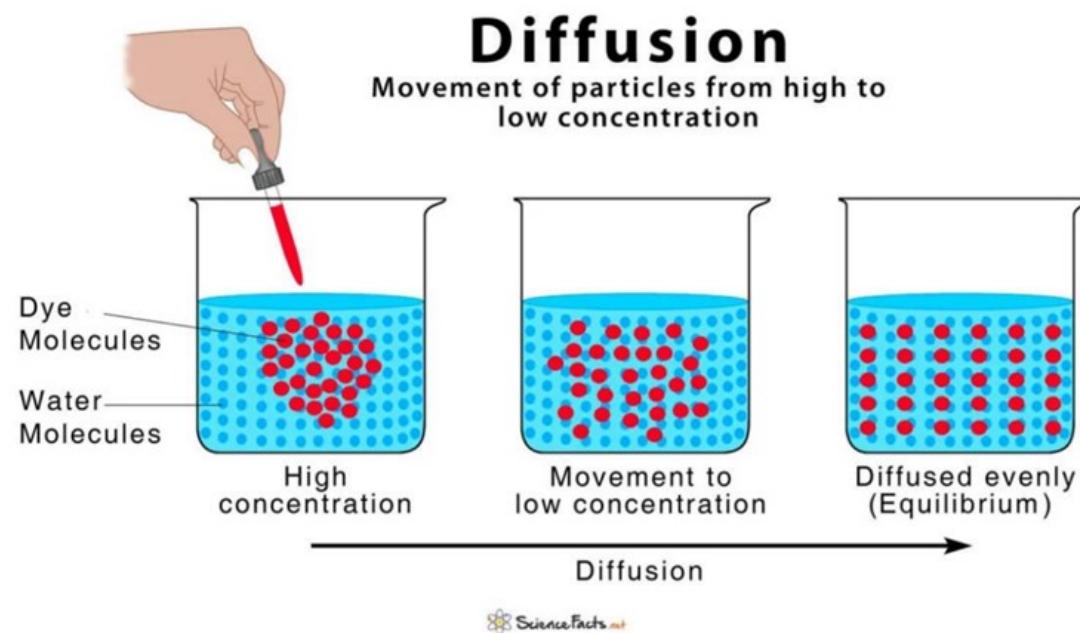
[1]Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in neural information processing systems 33 (2020): 6840-6851.

94

# Diffusion

扩散模型受到**非平衡热力学**的启发，构建了一个在扩散过程中逐步添加噪声的**马尔可夫链**。该过程通过缓慢地向原始数据中注入随机噪声，最终将其转化为纯噪声分布；随后，模型通过学习反向扩散过程，从噪声中逐步还原出所需的数据样本。

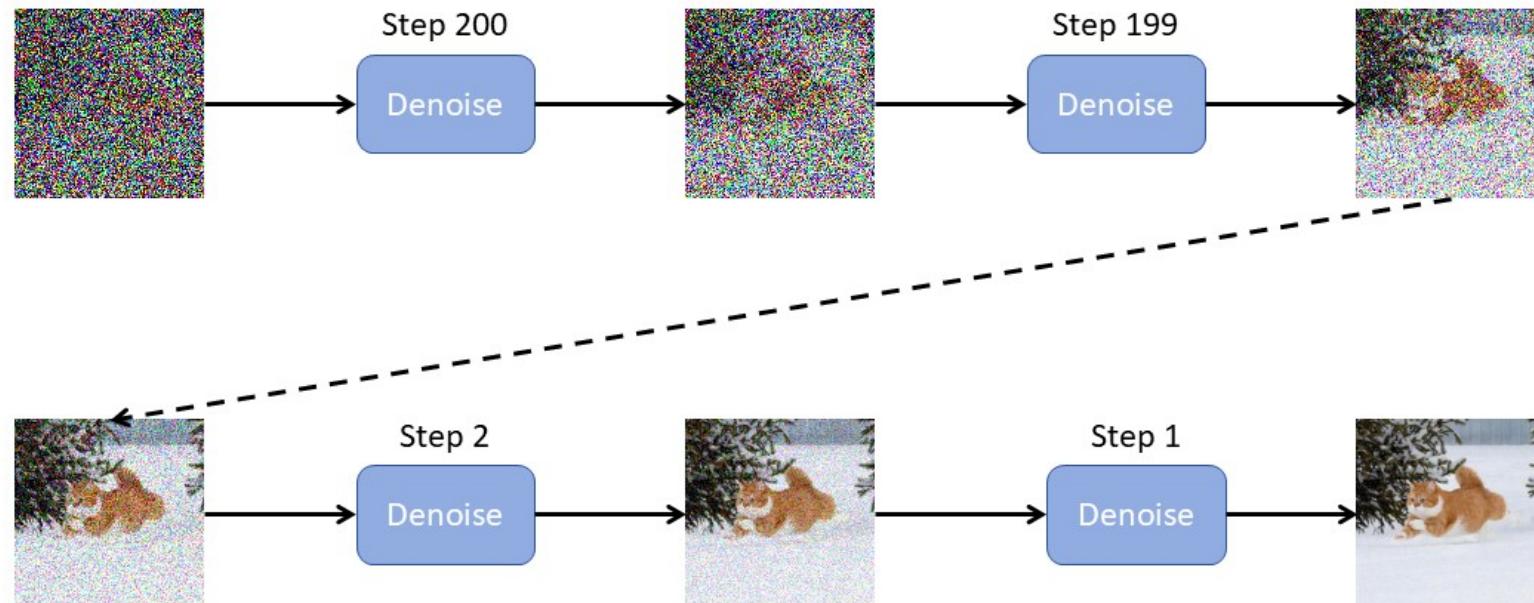
新的学习范式！  
反向扩散



95

# Diffusion Model

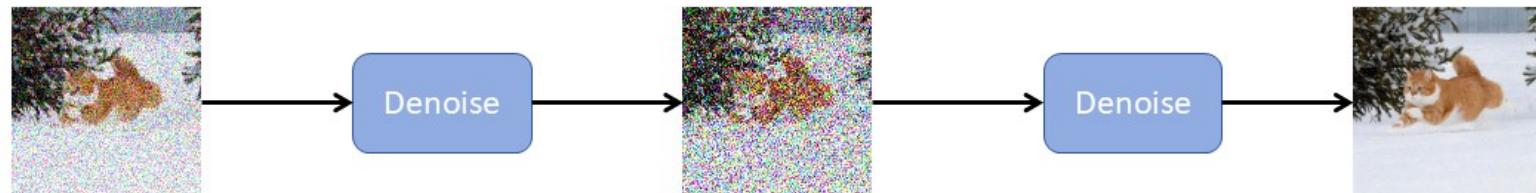
扩散模型的运作原理：把一张全是噪声的图片慢慢去噪（Denoise），直到变成原图。



反向过程

96

# Diffusion Model

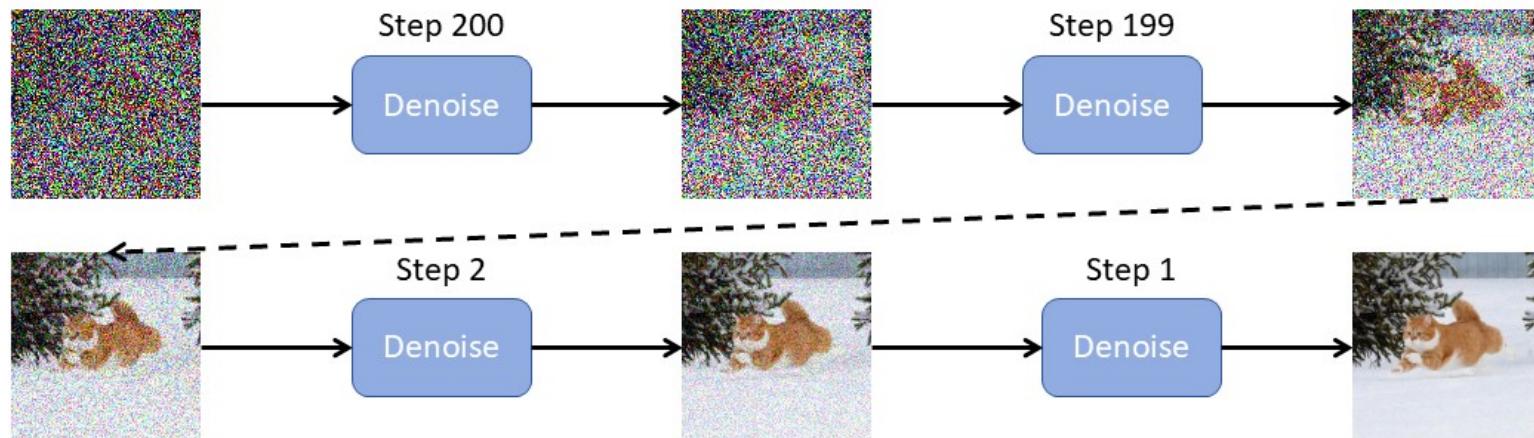


The sculpture is already complete within the marble block, before I start my work. It is already there, I just have to chisel away the superfluous material. - **Michelangelo**

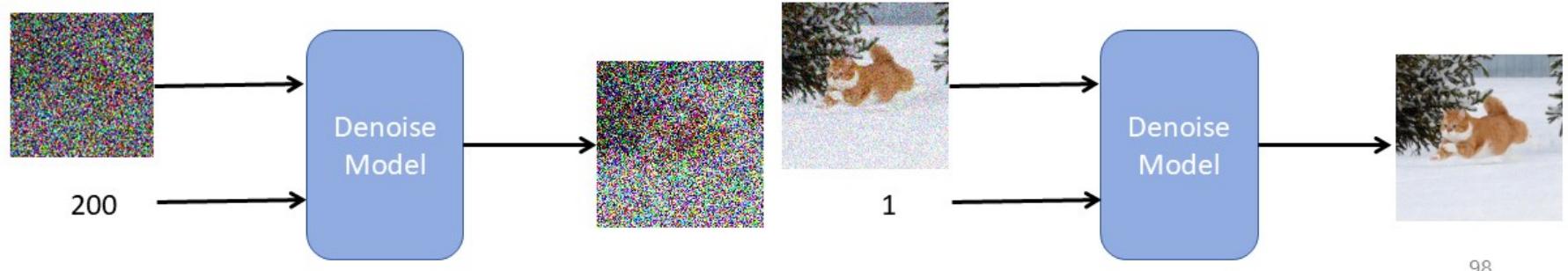


97

# Diffusion Model



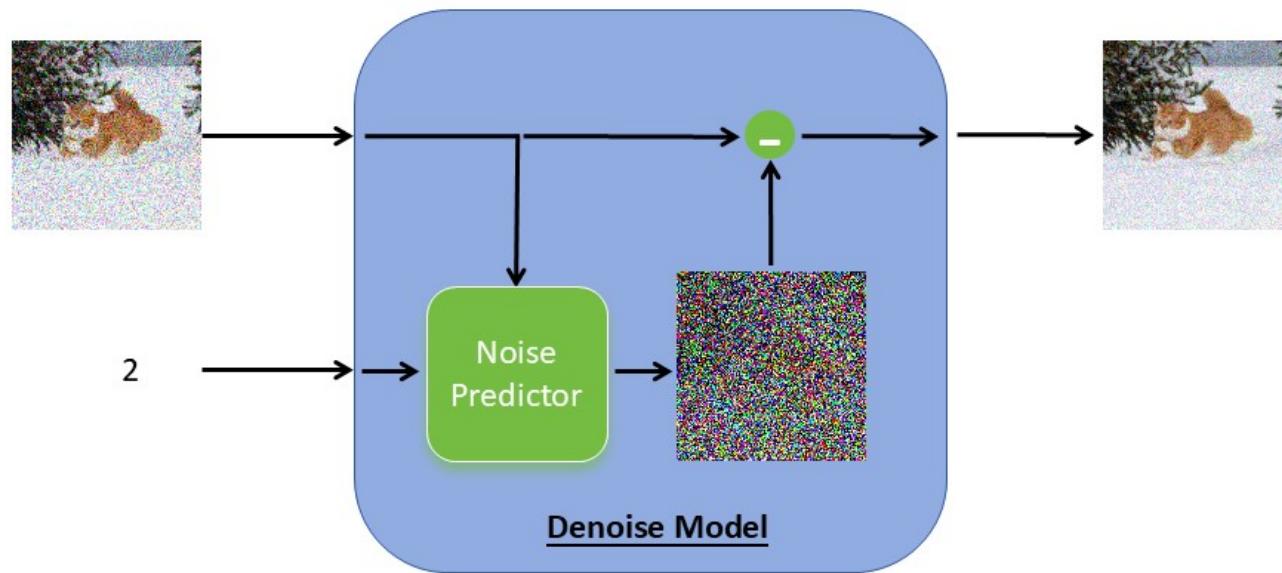
所有的过程都是使用同一个去噪模型，只是额外需要一个输入Step：



98

# Diffusion Model

去噪模型的内部实际做的事情：用Noise Predictor预测图片的噪声，然后减掉。

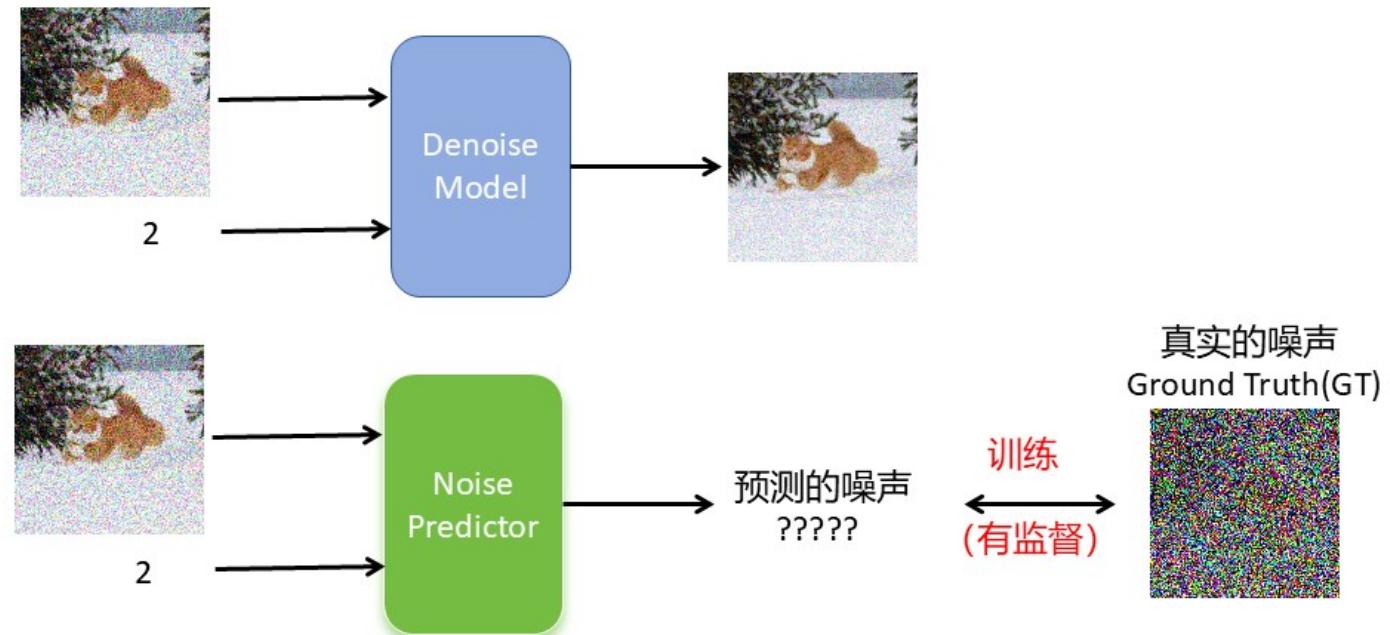


? 如何训练?

99

# Diffusion Model

如何训练模型：

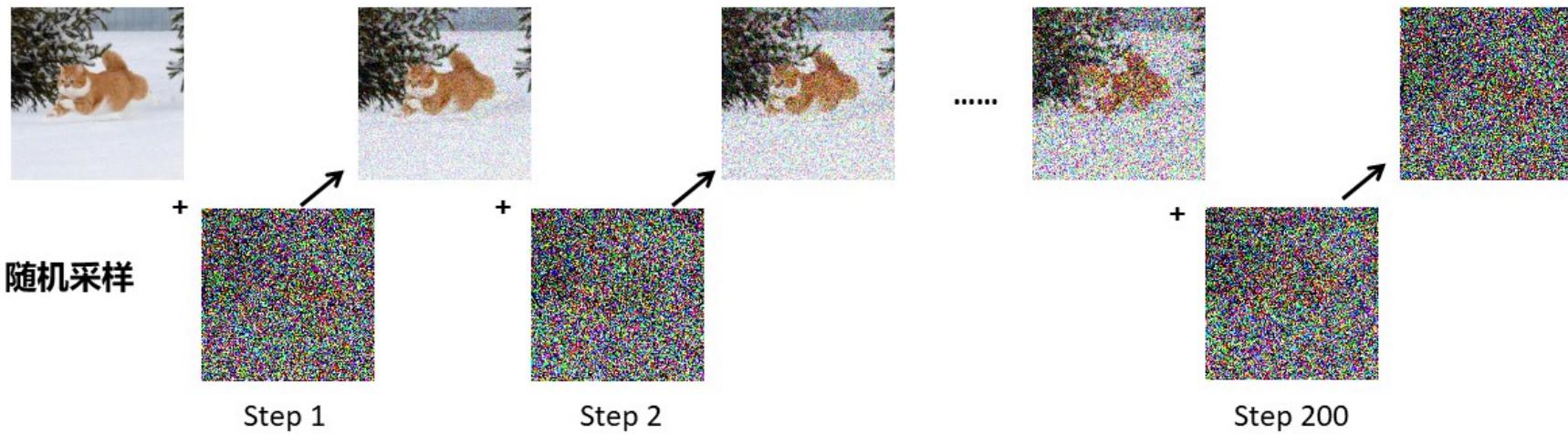


最后一个问题是：Ground Truth怎么来？

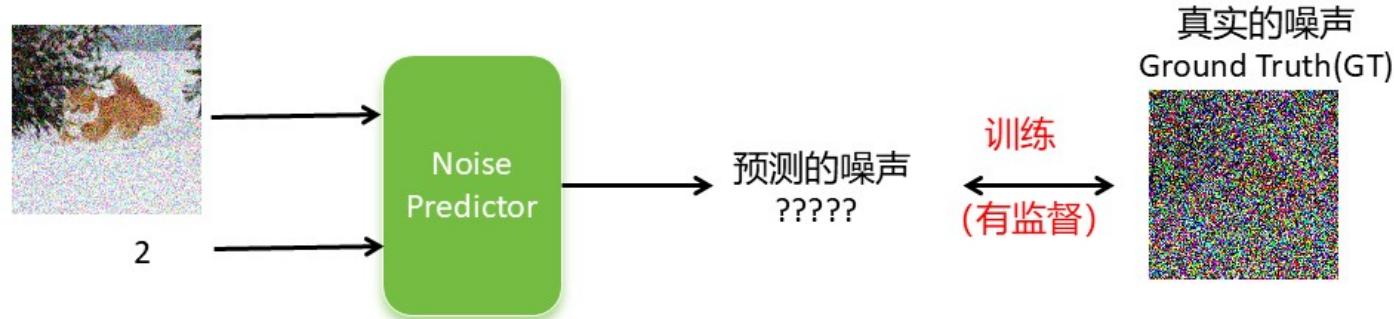
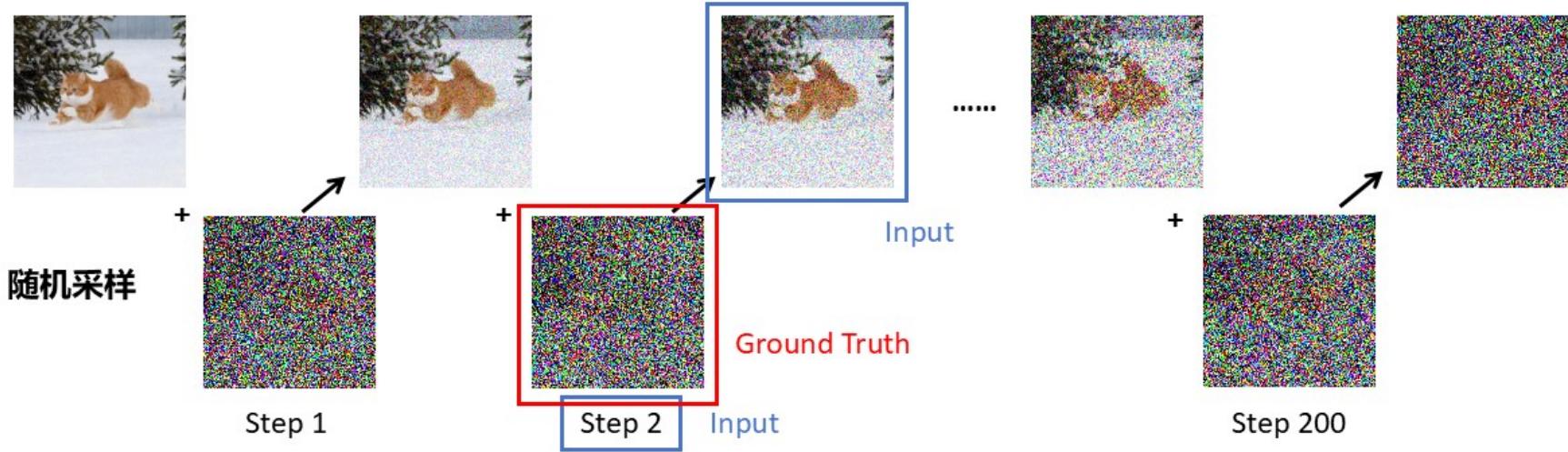
100

# Diffusion Model

对于Ground Truth(GT), 我们可以模拟加噪的过程:



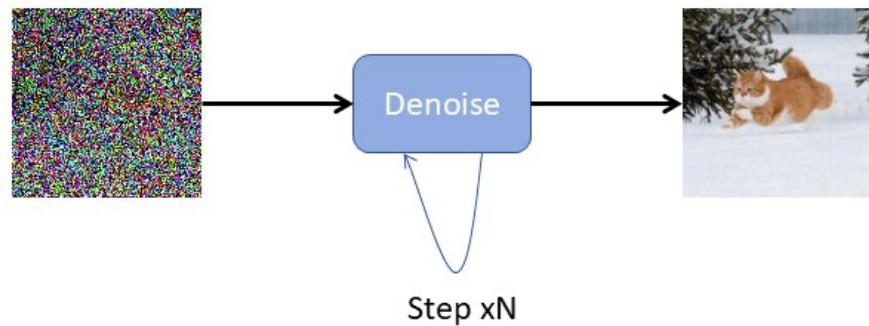
# Diffusion Model



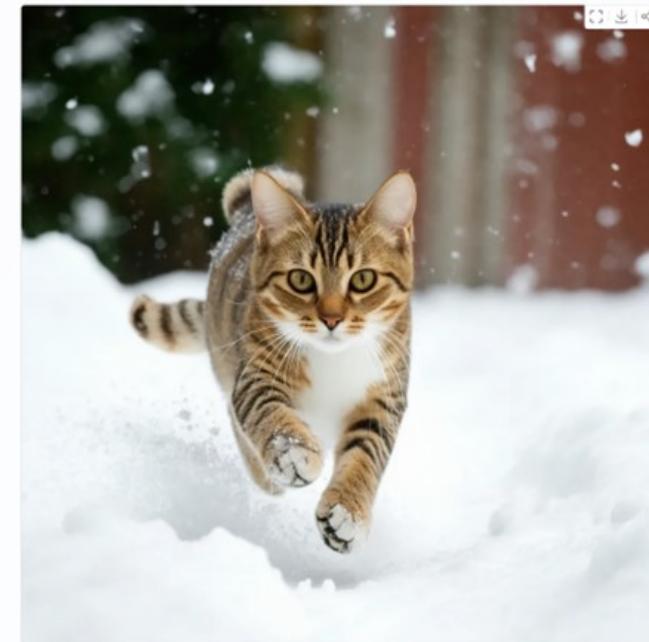
102

# Text-to-Image

现在我们完成的事情：



a cat running in the snow Run



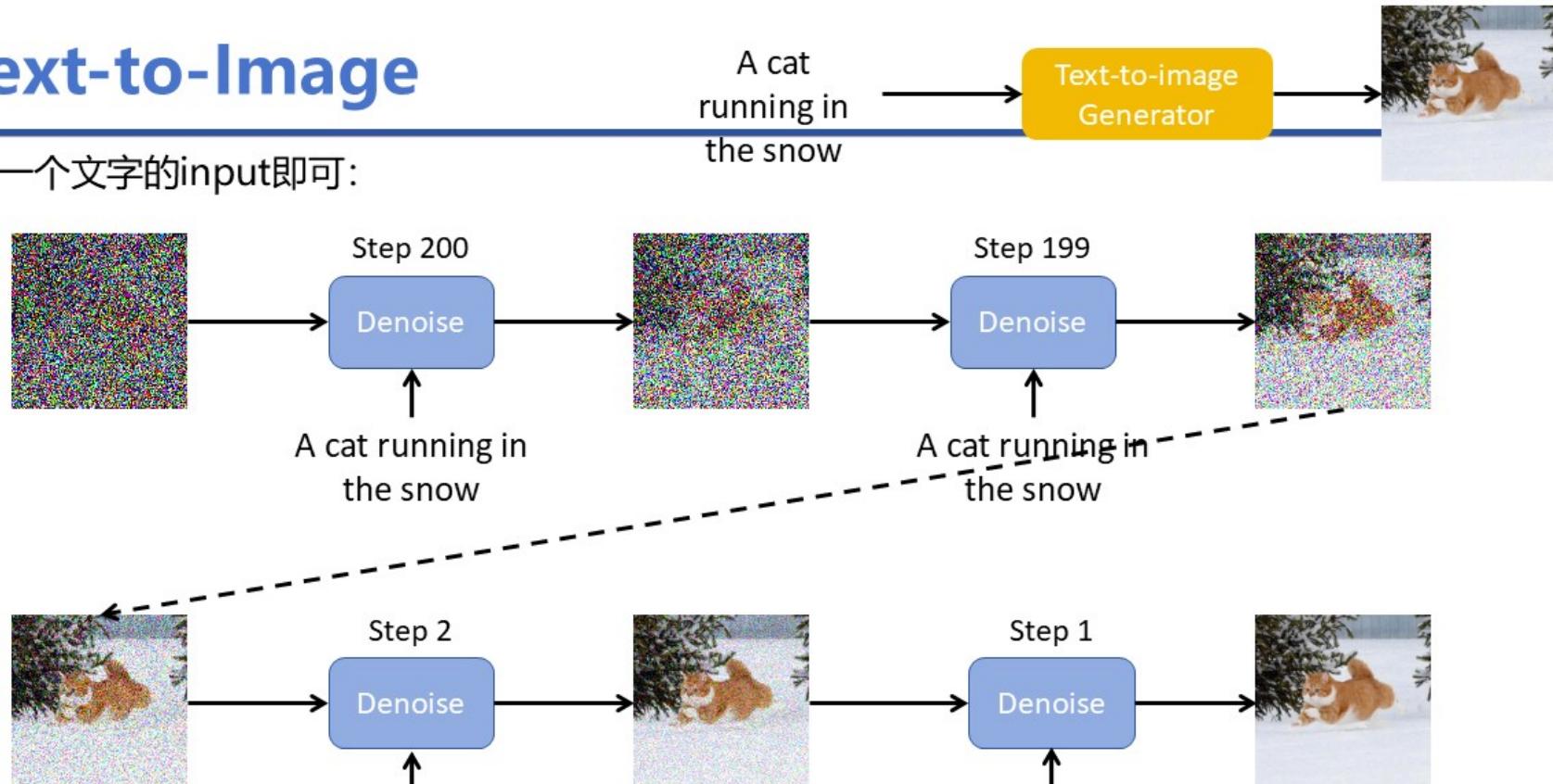
我想做的：用文字描述来生成图片。



103

# Text-to-Image

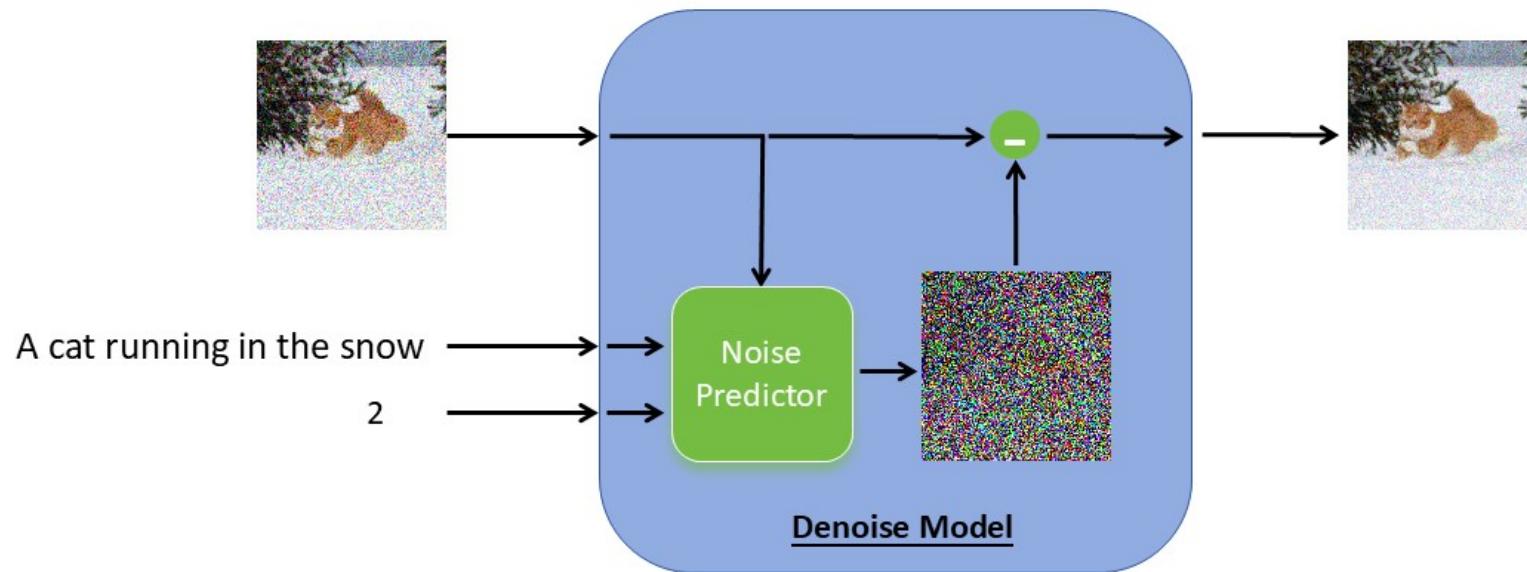
再加一个文字的input即可：



反向过程

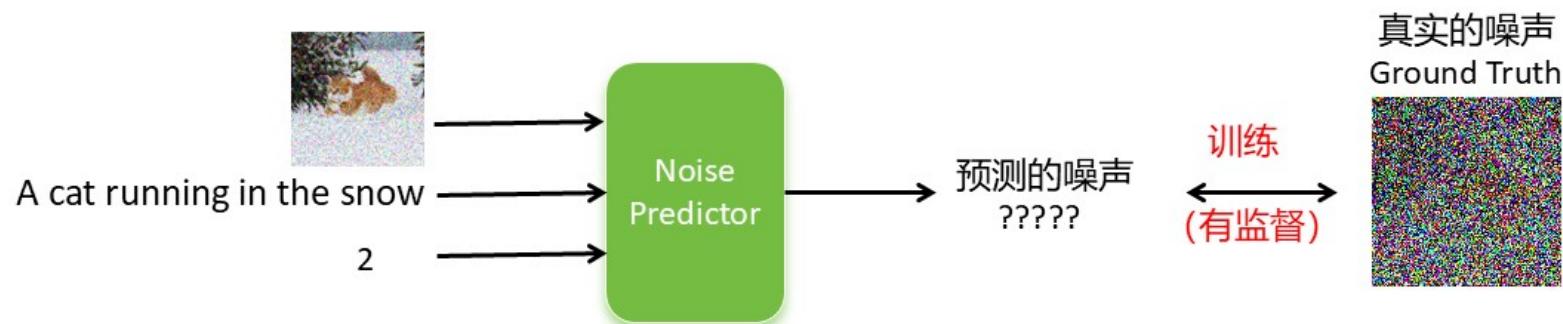
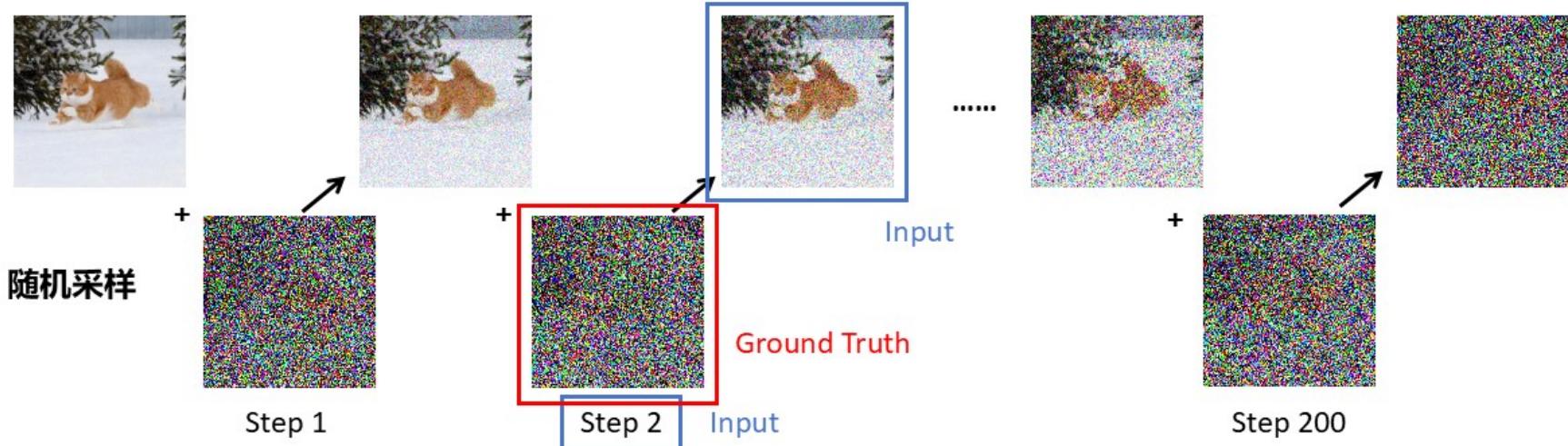
104

# Text-to-Image



105

# Text-to-Image



106

# Diffusion Model

## Algorithm 1 Training

```
1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
        $\nabla_{\theta} \left[ \epsilon - \epsilon_{\theta} (\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon | t) \right]^2$ 
6: until converged
```

GT噪声

加噪t次的图片

Noise  
Predictor

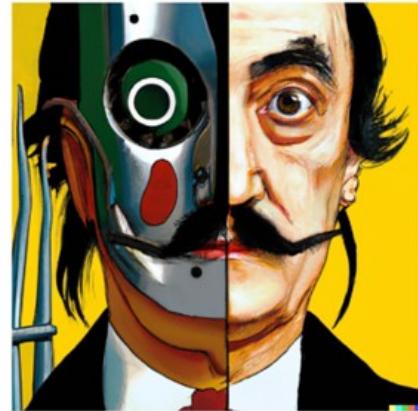
## Algorithm 2 Sampling

```
1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left( \mathbf{x}_t - \frac{1 - \bar{\alpha}_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 
```

根据图片和Step  
预测的噪声

作用：增加随机扰动  
以提升生成多样性

# Diffusion Model



vibrant portrait painting of Salvador Dalí with a robotic half face



a shiba inu wearing a beret and black turtleneck



a close up of a handpalm with leaves growing from it



an espresso machine that makes coffee from human souls, artstation



panda mad scientist mixing sparkling chemicals, artstation



a corgi's head depicted as an explosion of a nebula

108

# 人工智能概论

Introduction to Artificial Intelligence

谢谢！



109