山东大学 SHANDONG UNIVERSITY

第4讲 深度学习

主讲：丛润民

山东大学 SHANDONG UNIVERSITY

目录

CONTENT

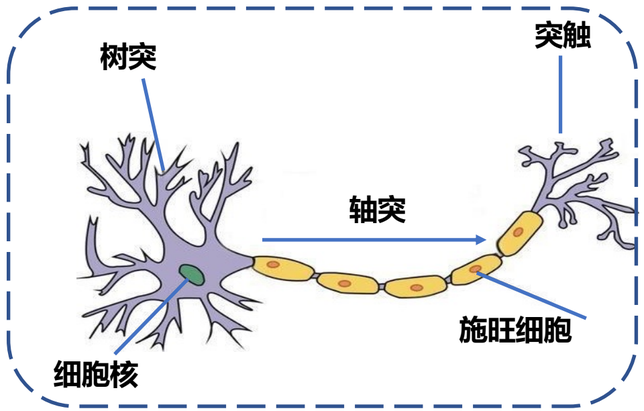
01

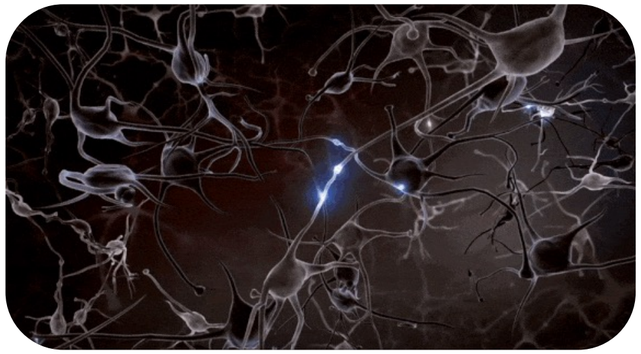
从感知机到深度神经网络

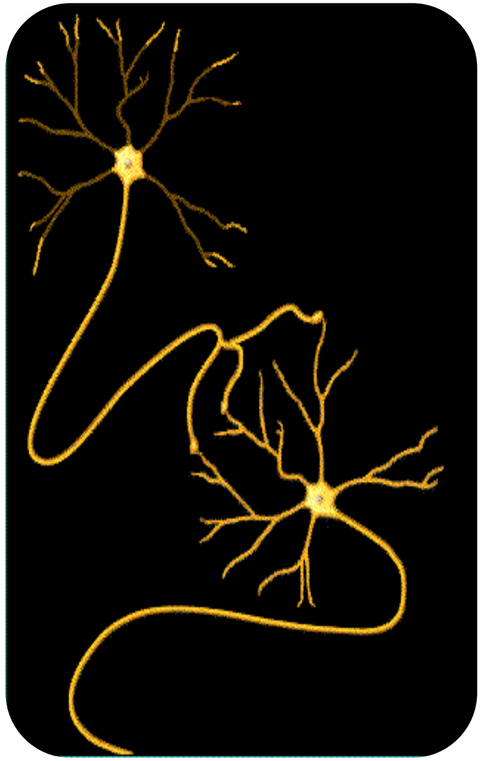
1.1 最简单的神经网络——感知机

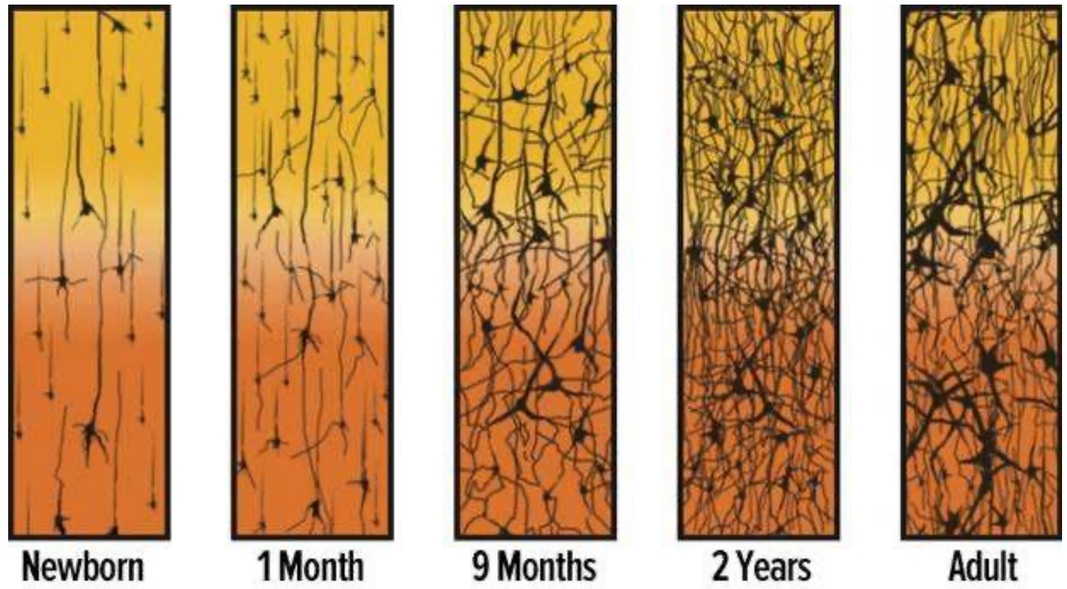
- 神经元是神经系统最基本的结构和功能单位，每个神经元结构大致都可以分为细胞体和突起两部分，突起进一步分为树突和轴突。神经元之间通过突触进行传导，具体传导过程涉及电信号到化学信号再到电信号的转换。

神经元结构示意图





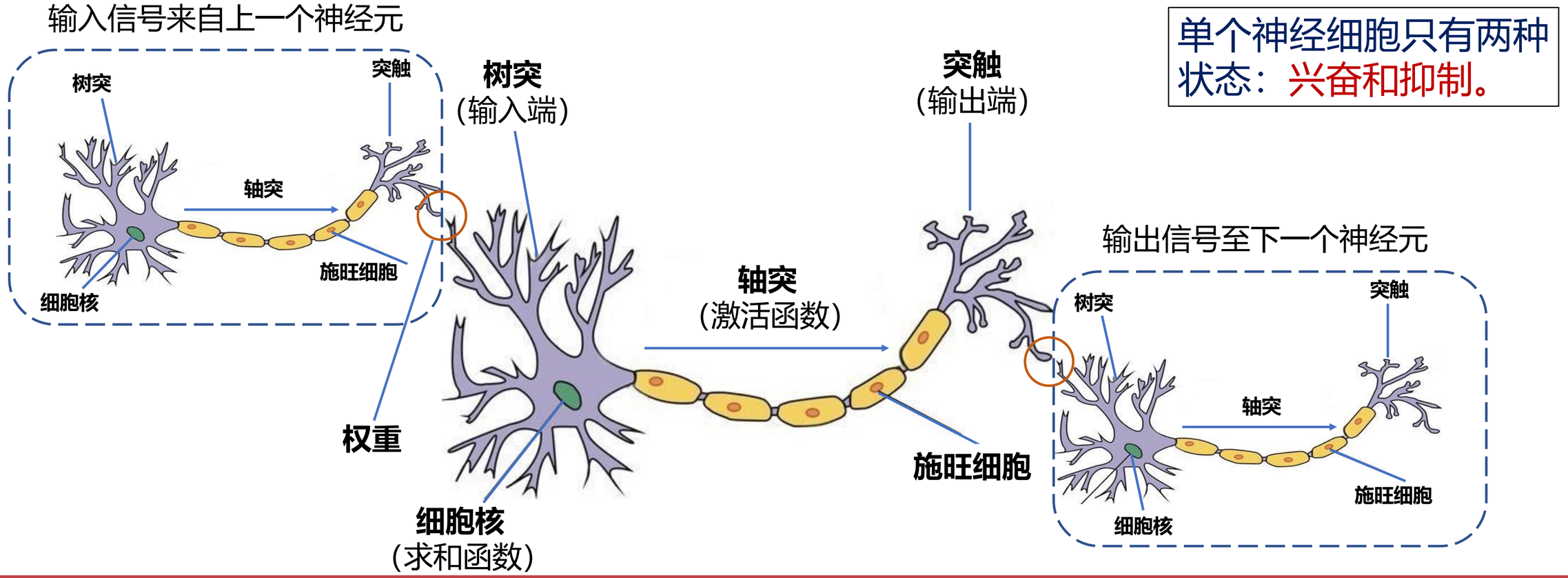




人脑神经网络的发育过程

1.1 最简单的神经网络——感知机

- 20世纪40年代，科学家们开始探索大脑神经元的连接模型，用数学模型描述复杂的网络，M-P（McCulloch-Pitts）神经元模型即麦卡洛克-皮茨模型，是1943年提出的一个简化的神经元模型，奠定了现代神经网络的基础。

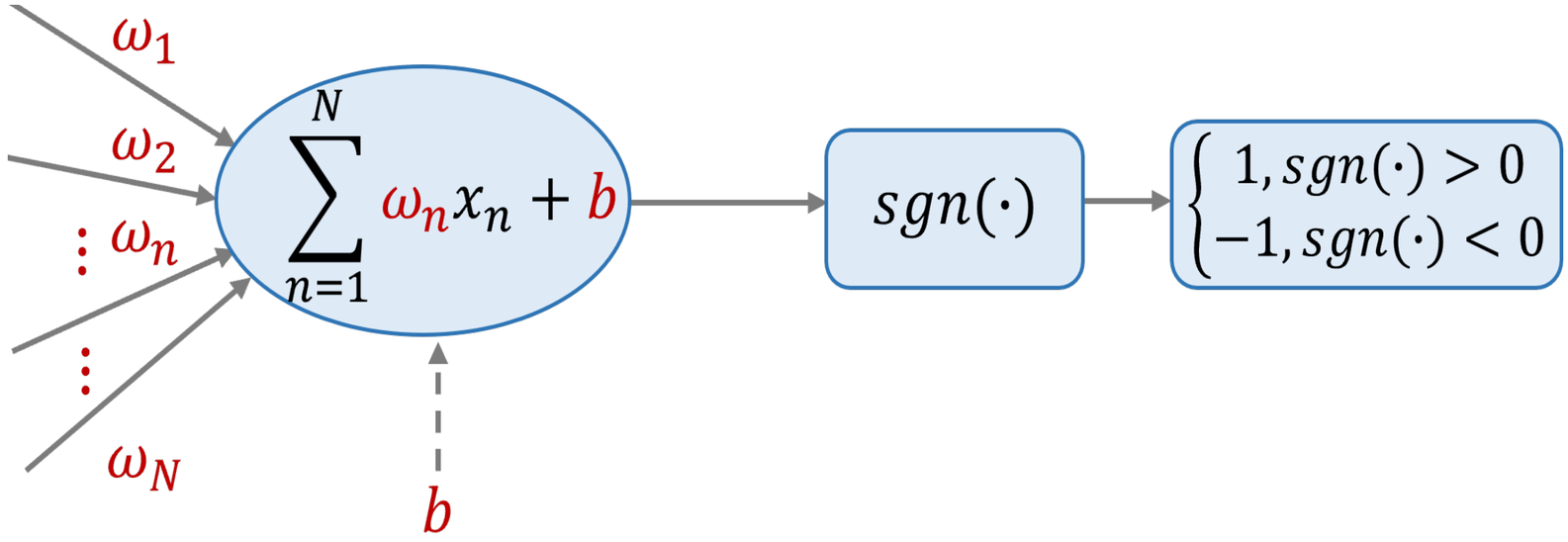


1.1 最简单的神经网络——感知机

- 感知机由美国人工智能领域著名心理学家弗兰克·罗森布拉特（Frank Rosenblatt）在 1957 年提出，是神经网络发展史上的**第一个人工神经网络模型**。

1.1 最简单的神经网络——感知机

- 感知机由美国人工智能领域著名心理学家弗兰克·罗森布拉特（Frank Rosenblatt）在 1957 年提出，是神经网络发展史上的**第一个人工神经网络模型**。
- 感知机的核心思想是通过模拟生物神经元的工作方式，**实现对输入数据的二分类任务**。它通过加权求和和激活函数（通常是阶跃函数）来判断输入数据的类别。

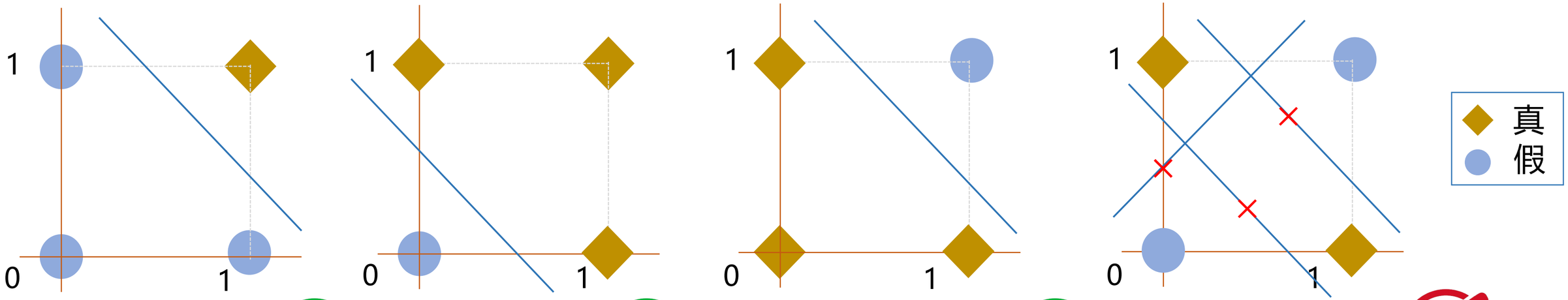


假设有N个样本的训练集$\{x_n, y_n\}_{n=1}^{N}$，其中$y \in \{+1, -1\}$，定义为：$\hat{y} = sgn(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b})$

1.1 最简单的神经网络——感知机

➢ 感知机中的逻辑运算

- 感知机可通过模拟逻辑与（AND）、或（OR）、与非（NAND）函数对数据进行二分类，但**无法模拟非线性可分的逻辑函数**，如异或（XOR）。



逻辑与（AND）

逻辑或（OR）

逻辑与非（NAND）

逻辑异或（XOR）

单层感知机无法解决XOR问题！

“小罗”与“小明”的学术之争

“小罗”与“小明”的学术之争

■ Rosenblatt & Perceptron

1957年，罗森布拉特提出了具有自组织、自学习能力的数学模型**感知机（Perceptron）**，并乐观预测期最终可以“学习，做决定，翻译语言”，美国海军也曾出资支持，期望它“以后可以自己走、说话、看、读，自我复制，甚至拥有自我意识。”

■ Rosenblatt vs. Minsky

1969年，马文·明斯基出版新书“感知机：计算几何简介”中论证了2个关键问题：

- 单层神经网络**无法解决不可线性划分的问题**，比如经典的异或问题；
- 神经网络需要**超大的计算量**才能完成计算；

而且他在书中评说道：“罗森布拉特写的大部分内容……毫无科学价值”。

2004 年，IEEE 计算智能学会设立了罗森布拉特奖（IEEE Frank Rosenblatt Award），奖励在生物及语言启发计算领域做出卓越贡献的个人。

1.2 由1到2的突破——多层感知机

- 输入为$[x_1; x_2]$，网络包含两层
  - ✓ **隐藏层**包含两个神经元：

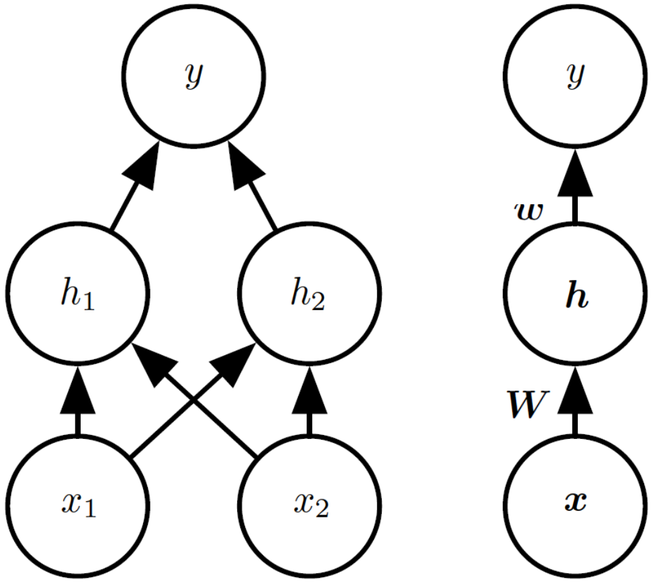
    $$\boldsymbol{h} = f^{(1)}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c})$$

  - ✓ **输出层**包含一个神经元：

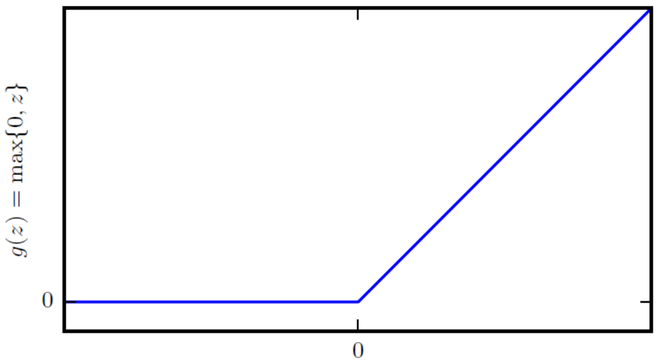
    $$y = f^{(2)}(\boldsymbol{h}; \boldsymbol{w}, b)$$

  - ✓ 隐藏层采用线性整流激活函数(ReLU)，则整个模型为：

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = f^{(2)}(f^{(1)}(\boldsymbol{x}))$$
$$= \boldsymbol{w}^{\mathrm{T}} \max\{0, \boldsymbol{W}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{c}\} + b$$



双层感知器



ReLU函数 $g(z) = \max\{0, z\}$

1.2 由1到2的突破——多层感知机

- 给出异或问题的一个解：

$$\boldsymbol{W}=\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}, \boldsymbol{c}=\begin{bmatrix}0\\ -1\end{bmatrix}, \boldsymbol{w}=\begin{bmatrix}1\\ -2\end{bmatrix}, b=0$$

模型处理流程如下：

① 输入4个样本的矩阵表示为：

$$\boldsymbol{X}=\begin{bmatrix}0 & 0\\ 0 & 1\\ 1 & 0\\ 1 & 1\end{bmatrix}$$

② 乘以第一层权重矩阵，得到：

$$\boldsymbol{XW}=\begin{bmatrix}0 & 0\\ 1 & 1\\ 1 & 1\\ 2 & 2\end{bmatrix}$$

③ 加上偏置向量$\boldsymbol{c}$，得到：

$$\boldsymbol{XW}+\boldsymbol{c}=\begin{bmatrix}0 & -1\\ 1 & 0\\ 1 & 0\\ 2 & 1\end{bmatrix}$$

在这个空间中，所有样本都处在一条斜率为1的直线上。

④ 使用整流线性变换，得到：

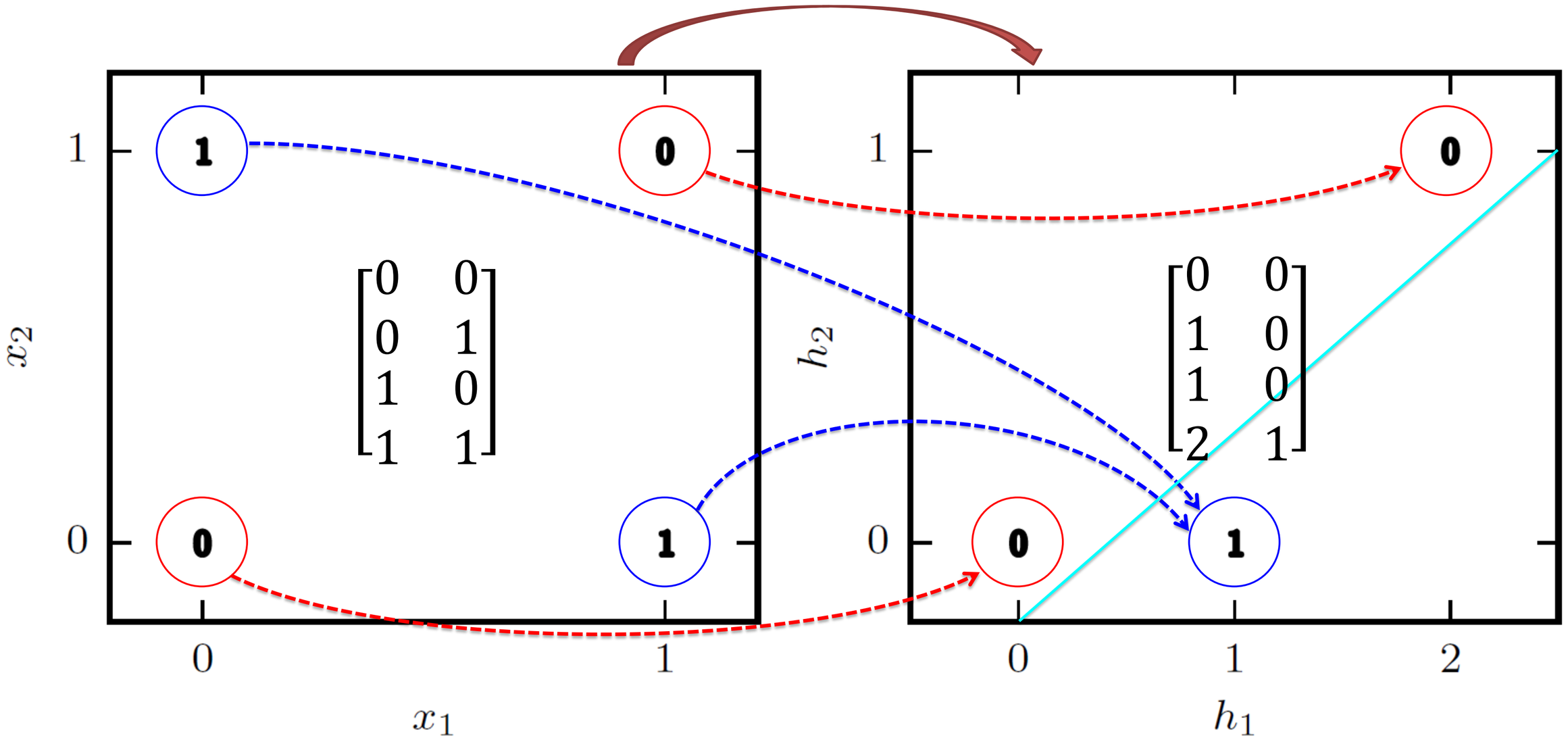
$$\max\{\boldsymbol{0}, \boldsymbol{XW}+\boldsymbol{c}\}=\begin{bmatrix}0 & 0\\ 1 & 0\\ 1 & 0\\ 2 & 1\end{bmatrix}$$

在这个空间中，所有样本不再处在同一条直线上了。

⑤ 乘以第二层权重向量$\boldsymbol{w}$，得到：

$$\boldsymbol{y}=\begin{bmatrix}0\\ 1\\ 1\\ 0\end{bmatrix}$$

1.2 由1到2的突破——多层感知机

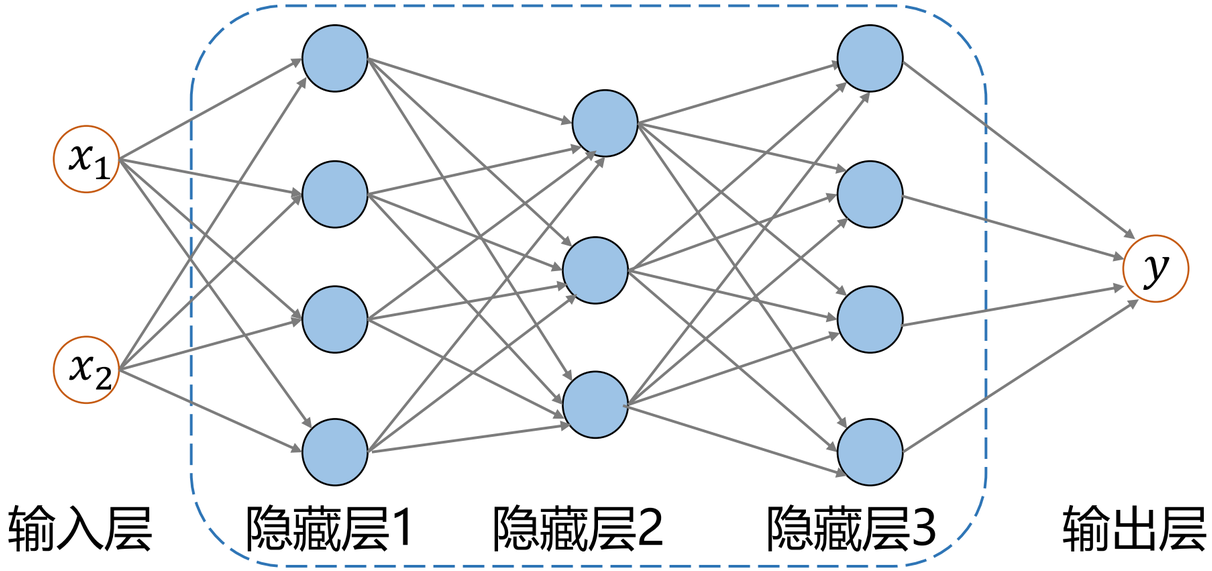


原始的$\boldsymbol{x}$空间

学习得到的隐藏$\boldsymbol{h}$空间

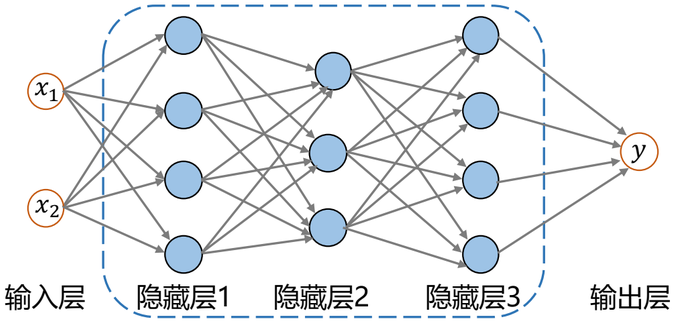
1.3 多层感知机的原理

- 多层感知机（Muti-Layer Perceptron，MLP）是由多个神经元（感知机）组成的神经网络，它包含输入层、输出层以及**至少一个隐藏层**，层与层之间通过神经元的连接权重进行信息传递，能处理和学习复杂的非线性关系。
- MLP也被称为前馈神经网络（Feedforward Neural Network，FFN）。



多层感知机示意图

1.3 多层感知机的原理



① 给定包含3层隐藏层前馈神经网络，对于输入向量x：

$$\boldsymbol{y} = \boldsymbol{\omega}^{(3)}(\boldsymbol{\omega}^{(2)}(\ \boldsymbol{\omega}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}\ ) + \boldsymbol{b}^{(3)}$$

② 将每个隐藏层输出看作是前一层信号**仿射变换**，则整个网络可以看作是嵌套的仿射函数：

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{B} = f(\boldsymbol{W}, \boldsymbol{B}; \boldsymbol{x})$$

$$= \boldsymbol{\omega}^{(3)}\boldsymbol{\omega}^{(2)}\boldsymbol{\omega}^{(1)}\boldsymbol{x} + \boldsymbol{\omega}^{(3)}\boldsymbol{\omega}^{(2)}\boldsymbol{b}^{(1)} + \boldsymbol{\omega}^{(3)}\boldsymbol{b}^{(2)} + \boldsymbol{b}^{(3)}$$

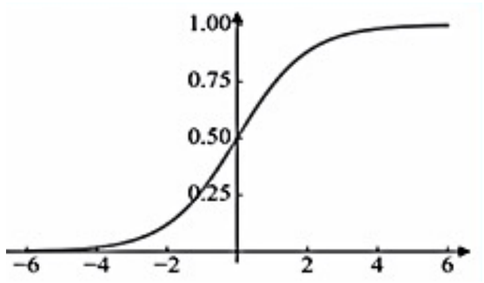
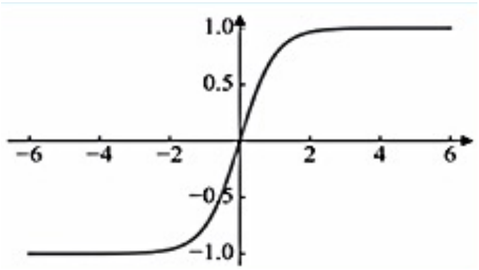
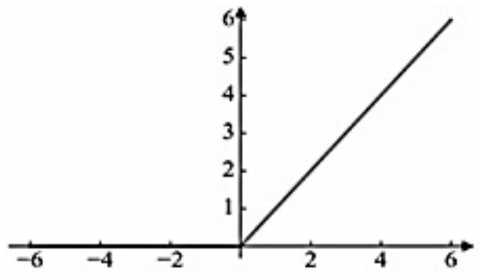
③ 在构建MLP时，相邻隐藏层之间会引入**非线性映射**作为激活函数，调整参数。

1.3 多层感知机的原理

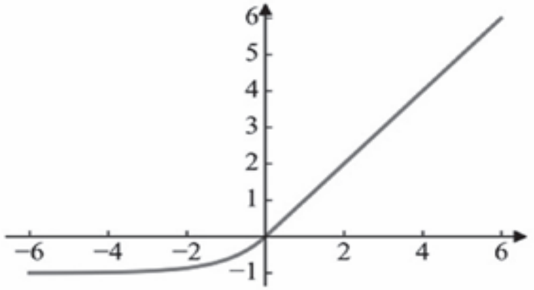
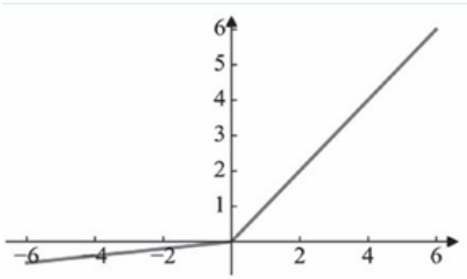
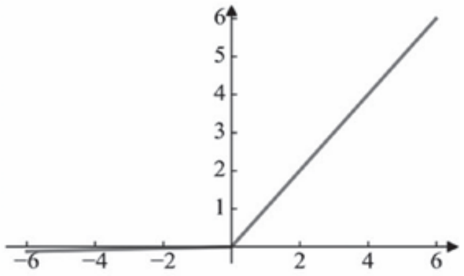
➢ 神经网络的“智能开关”：激活函数

- 激活函数在神经网络中就像是一个“开关”，具有**引入非线性**、**增加模型表达能力**、**决定神经元输出状态**等功能。
  - ✓ 激活函数能够对神经元的输入进行非线性变换，使神经网络可以拟合任意复杂的非线性函数，让神经网络学习到更丰富、更复杂的特征组合，大大提高了模型的表达能力和适应性。
  - ✓ 合适的激活函数可以缓解梯度消失或爆炸问题，并且使神经网络的训练过程更加高效，加速模型的收敛速度。
- 激活函数通常具备以下特点：
  - ✓ 连续可导的非线性函数
  - ✓ 激活函数及导函数尽可能简单，提高网络计算效率
  - ✓ 激活函数的导函数的值域有限

1.3 多层感知机的原理

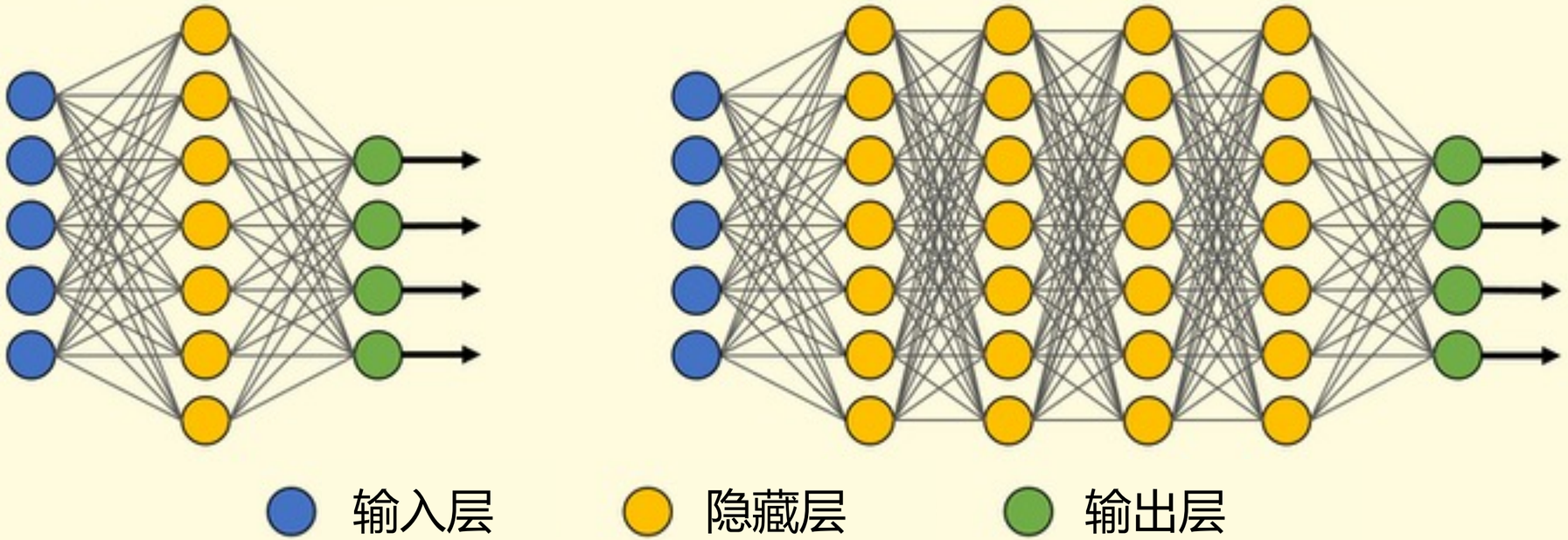
名称	数学表达式	函数图像	特点
Sigmoid	$f(x)=\frac{1}{1+e^{-x}}$		输入的取值范围没有限制。存在梯度消失问题，随着网络深度增加而变得愈发严重，已经很少作为激活函数使用。
Tanh	$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$		与sigmoid函数较为相似，值域为(-1,1)，在某些情况下更加稳定。同样面临梯度消失的问题，也很少用作激活函数。
ReLU	$f(x)=\max(0,x)$		计算简单，训练速度快，且在实践中表现出色。但存在“神经元死亡”问题。

1.3 多层感知机的原理

名称	数学表达式	函数图像	特点
ELU	$f(x)=\begin{cases}x & if\ x\geq 0\\ \gamma(e^x-1) & if\ x<0\end{cases}$		在正数区域保持线性，负数区域呈指数衰减，有助于缓解神经元死亡问题，并且可以输出负值。
PReLU	$f(x)=\begin{cases}x & if\ x\geq 0\\ \alpha x & if\ x<0\end{cases}$ α为可学习参数（初始值为0.1）		在负值域引入了一个可学习非零斜率α，可以在训练过程中自动调整，有效避免ReLU激活函数中“神经元死亡”问题。
Leaky ReLU	$f(x)=\begin{cases}x & if\ x\geq 0\\ \alpha x & if\ x<0\end{cases}$ α为预设固定常数		与PReLU相似，在非正值域引入了一个非零斜率，旨在解决“死亡ReLU”问题。斜率α是一个预设固定值，相对来说不够灵活。

1.4 深度前馈神经网络结构

简单前馈神经网络



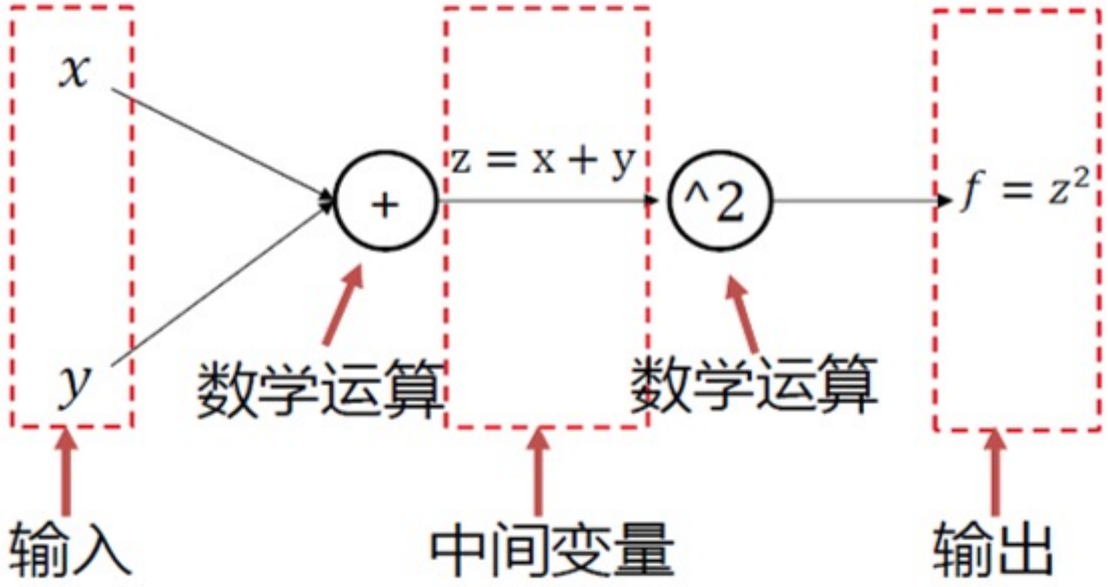
1.4 深度前馈神经网络结构

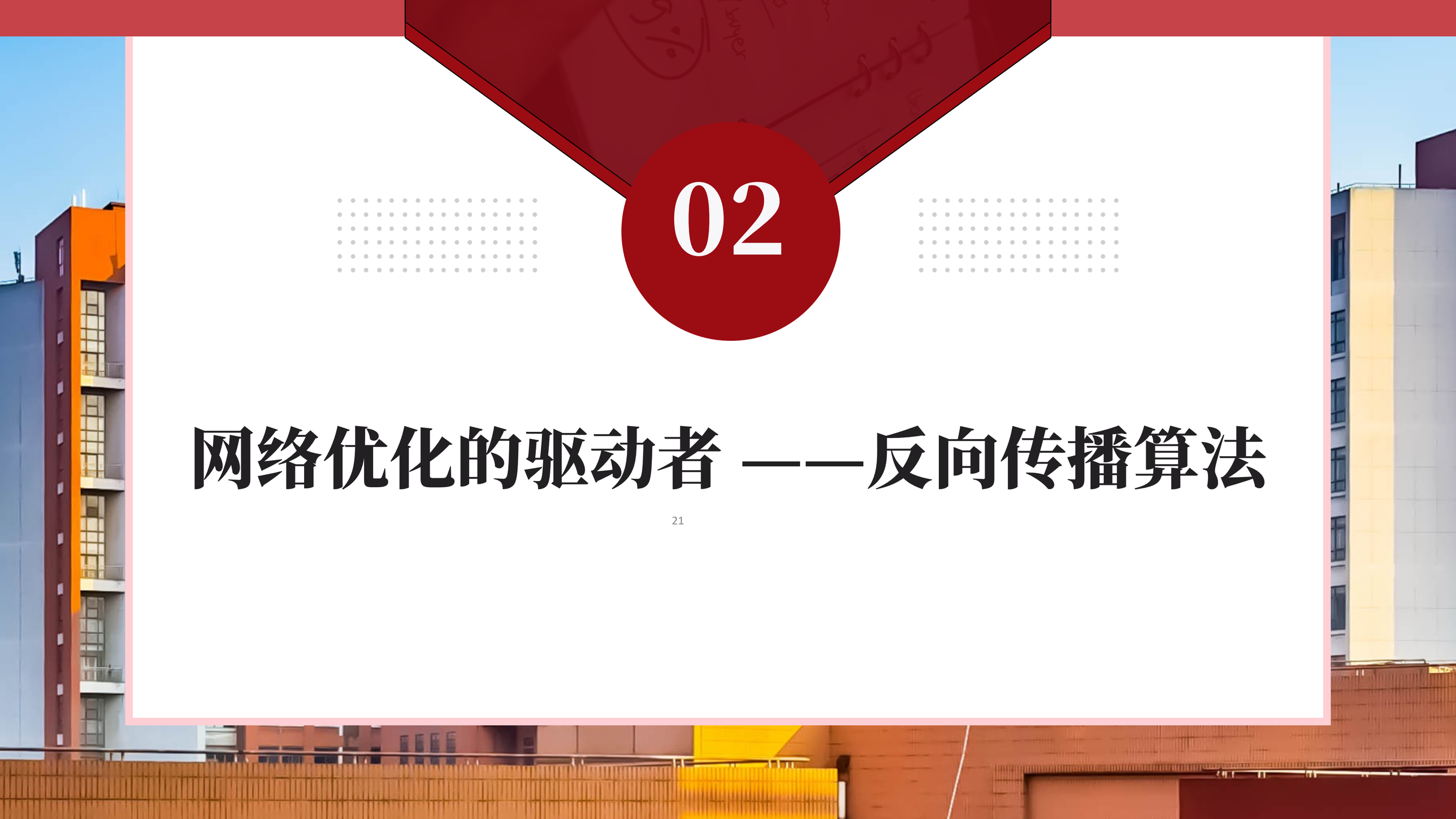
➢ 深度神经网络的前向传播过程

□ 前向传播（Forward Propagation）是深度神经网络进行预测和分类的基础过程，用于通过网络每一层传递输入数据并生成输出结果。

● 示例：函数 $f = (x + y)^2$ 的前向传播过程

说明：这里采用**计算图**表示深度神经网络的前向传播过程。计算图是一种有向图，它用来表达输入、输出以及中间变量之间的计算关系，图中每个节点对应一种数学运算。



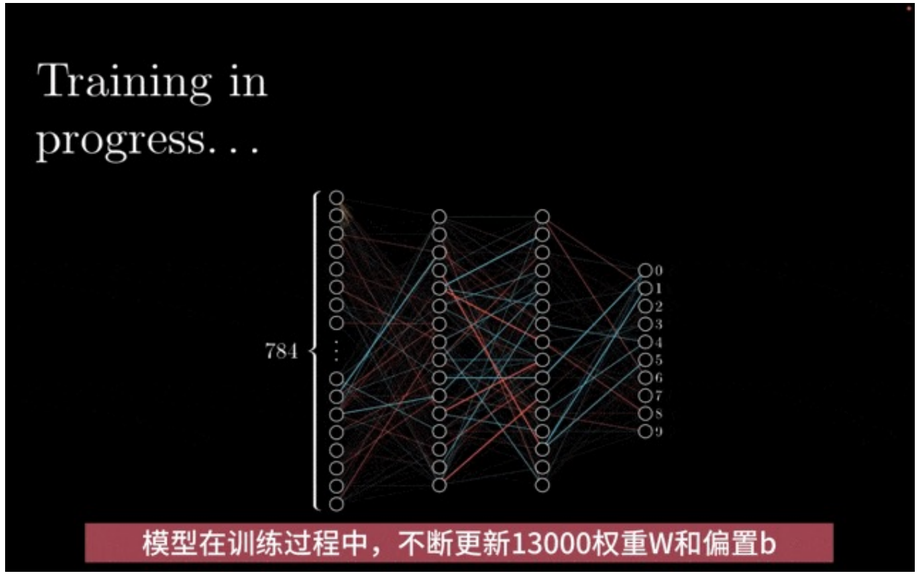


02

网络优化的驱动者 ——反向传播算法

2.1 误差传递的“逆流而上”—— 网络的自我调整

- 为了更好地完成一项任务时，总会留下一系列的物理**变化**，比如大脑中的细胞或神经网络中的数值会发生相应的变化，这些变化是自我升级的基础。
- 大脑或神经网络等系统**如何准确地计算出需要做出哪些改变**并不简单，在这个问题中，大脑或人工智能系统必须查明信息传递过程中存在错误的原因，然后做出必要的改变，这一过程可称之为**信用分配问题**。



2.1 误差传递的“逆流而上”——“神奇尺子”

■ 损失函数（Loss Function）衡量模型预测$\hat{y}_n = f(x_n, W)$与真实标签y_n之间**误差**。

$$L = \frac{1}{N}\sum_n L_n(f(x_n, W), y_n)$$

- x_n表示数据集中第n个输入样本
- $f(x_n, W)$为模型对x_n的预测结果
- y_n为样本n真实标签
- L_n为第n个样本预测的损失值

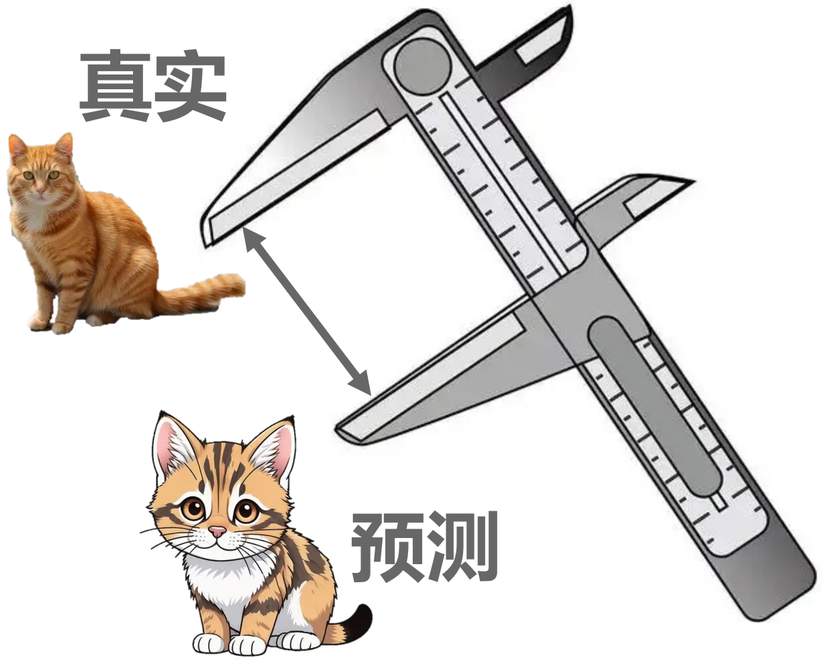
■ 常见损失函数：

- 均方误差（Mean Square Error, MSE）

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

- 平均绝对误差（Mean Absolute Error, MAE ）

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|\hat{y}_n - y_n|$$



2.1 误差传递的“逆流而上”——“指南针”

■ 梯度表示函数在某一点处的斜率或方向导数，指引并调整模型参数以最小化误差。

➢ 梯度计算方法：数值法

一维变量，函数求导：$$\frac{dL(w)}{dw}=\lim_{h\to 0}\frac{L(w+h)-L(w)}{h}$$

计算量大，不精确！

示例：损失函数在$L(w)=w^2$在$w=1$点处的梯度

$$\frac{dL(w)}{dw}=\lim_{h\to 0}\frac{L(w+h)-L(w)}{h}\approx\frac{L(1+0.0001)-L(1)}{0.0001}=2.0001$$

➢ 梯度计算方法：解析法

$$\nabla L(w)=2w;\quad \nabla_{w=1}L(w)=2$$

精确，速度快，导数计算易错！

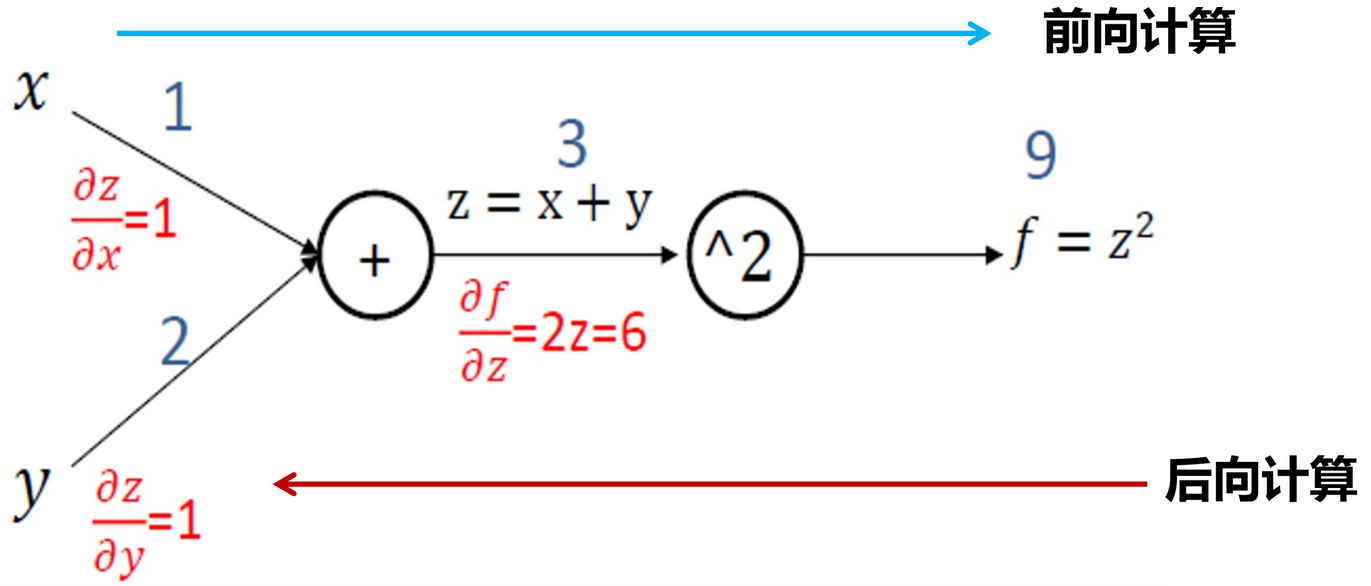
2.1 误差传递的“逆流而上”——“魔法秘籍”

- 就像一场紧张的接力赛，“链式求导法则”先计算出顶层神经元对最终结果的影响程度，然后把这个“接力棒”逆向传递给上层神经元。上层神经元接“棒”后，结合传来的影响，算出自己对结果的影响，再传递给更上一层。就这样层层传递，直到第一层。通过这种方式，每个神经元都清楚知道自己该怎么进行参数调整。
- 示例：函数$f=(x+y)^2$的前向与后向计算过程。

$$\frac{\partial f}{\partial x}=\frac{\partial f}{\partial z}\frac{\partial z}{\partial x}$$

链式法则求输出-中间层(输入)的梯度

$$\frac{\partial f}{\partial y}=\frac{\partial f}{\partial z}\frac{\partial z}{\partial y}$$

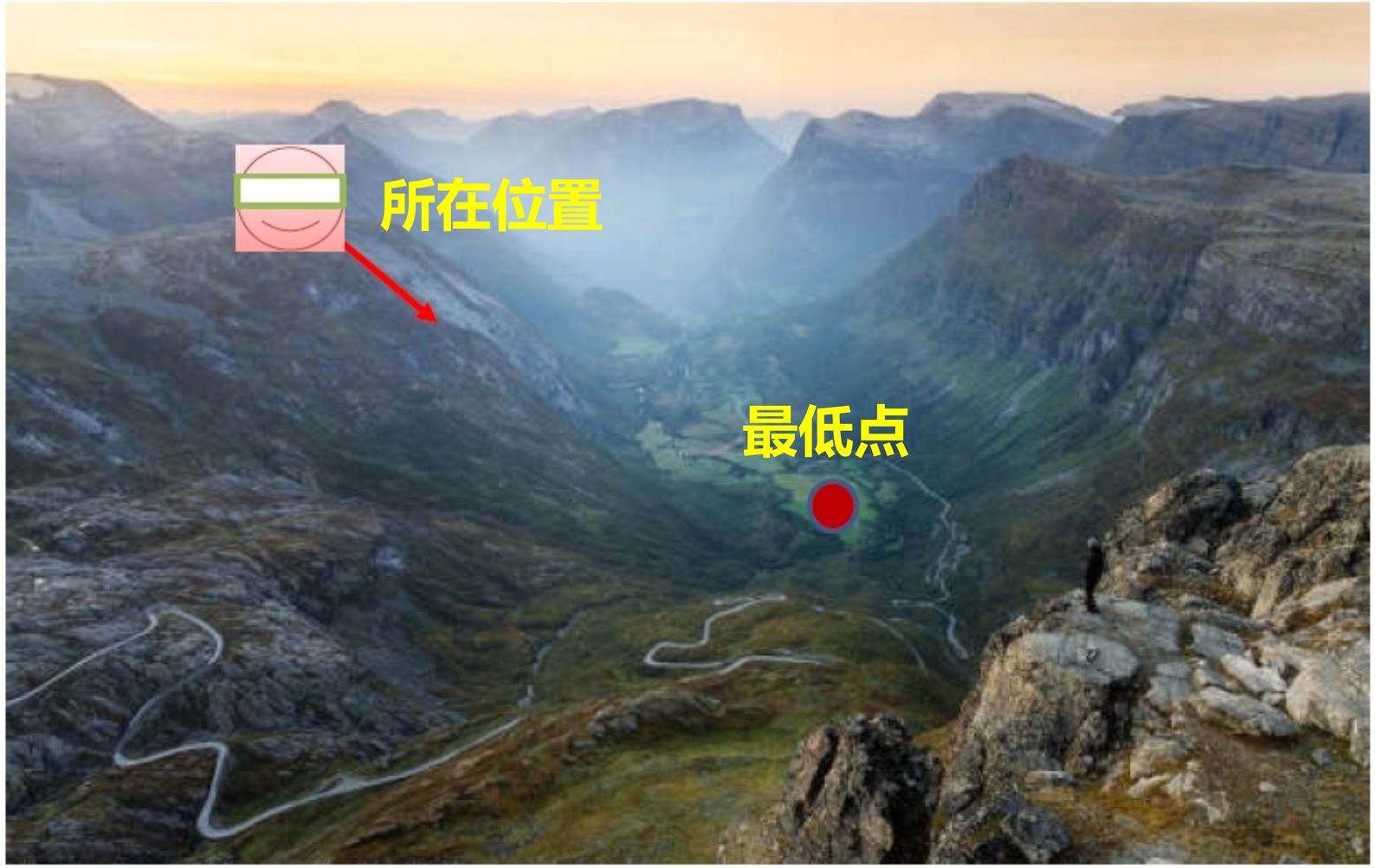


2.2 逐步优化的“调整大师”—— 梯度下降法

- 模型优化过程中的最低点导航问题——“山谷”问题

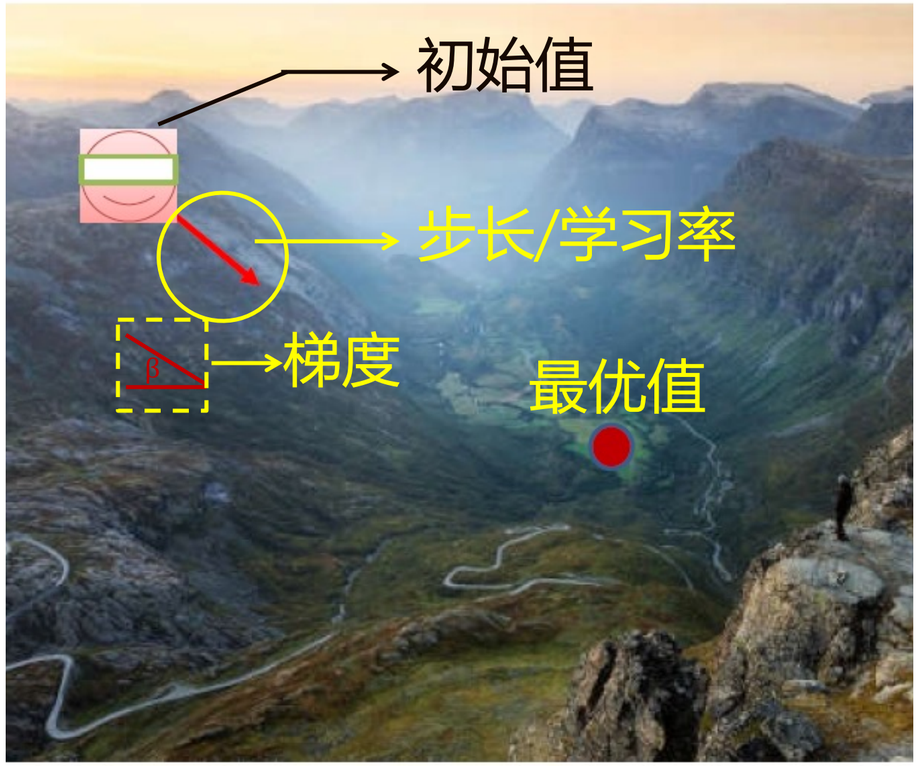
问题1：往哪里走?

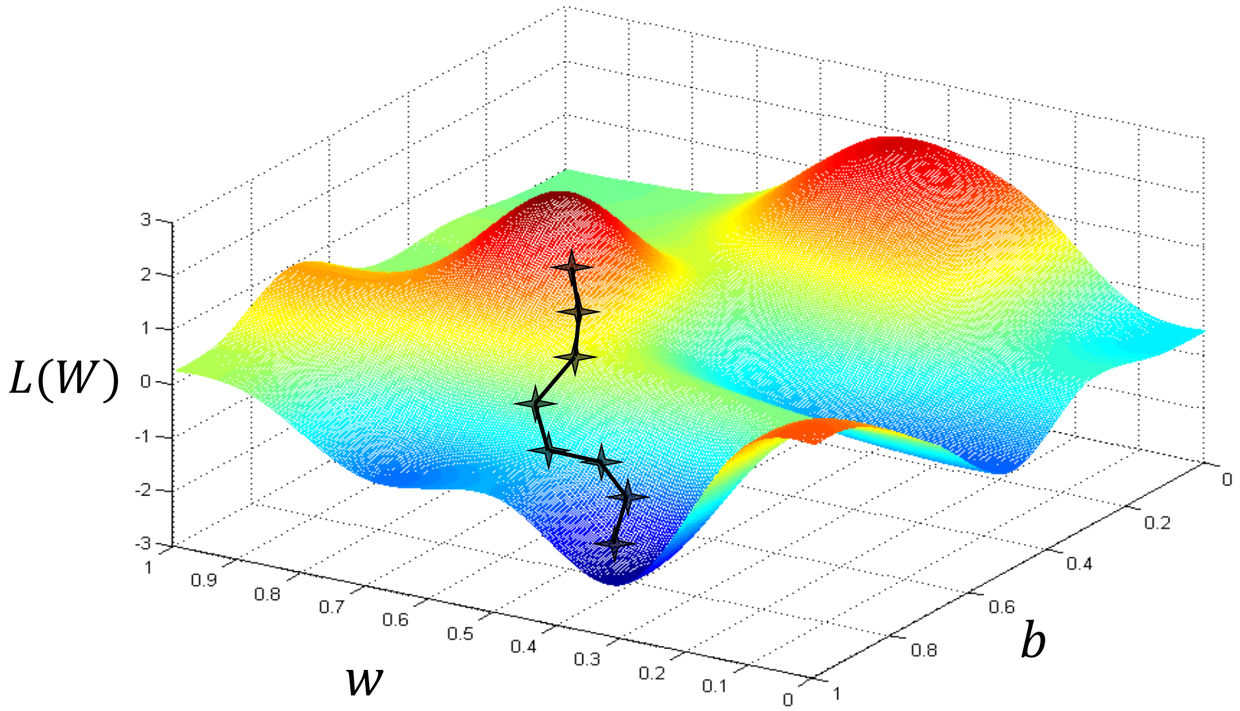
问题2：一步走多远?



2.2 逐步优化的“调整大师”—— 梯度下降法

- 从山谷问题到梯度下降法





参数空间坐标系

梯度下降法以损失函数为研究对象，刻画参数空间中从初始参数到最优参数的优化过程。

2.2 逐步优化的“调整大师”—— 梯度下降法

➢ 梯度下降法的一般流程

- **第1版本：从初始值到最优值的过程描述**

最优值 = 初始值 + 带有方向信息的步长

- **第2版本：如何刻画“带有方向信息的步长”？**

梯度 ⊗ 步长

模型优化中的智能导航系统

- **第3版本：整合第1版本+第2版本**

最优值 = 初始值 + (**-梯度**×步长)

$$w \leftarrow w_0 - \alpha \nabla L(w)$$

$$b \leftarrow b_0 - \alpha \nabla L(b)$$

2.2 逐步优化的“调整大师”—— 梯度下降法

- **反向传播**是损失函数信息通过网络从后向前进行**梯度递归计算**。

假设一个输入为x，真实标签为y，损失函数为$L(y,\hat{y})$的前馈神经网络，拆解为：

$$1)\ \boldsymbol{z}^{(1)}=\boldsymbol{w}\boldsymbol{x};\quad 2)\ \boldsymbol{z}^{(2)}=\boldsymbol{z}^{(1)}+\boldsymbol{b};\quad 3)\ \hat{\boldsymbol{y}}=\sigma\left(\boldsymbol{z}^{(2)}\right)$$

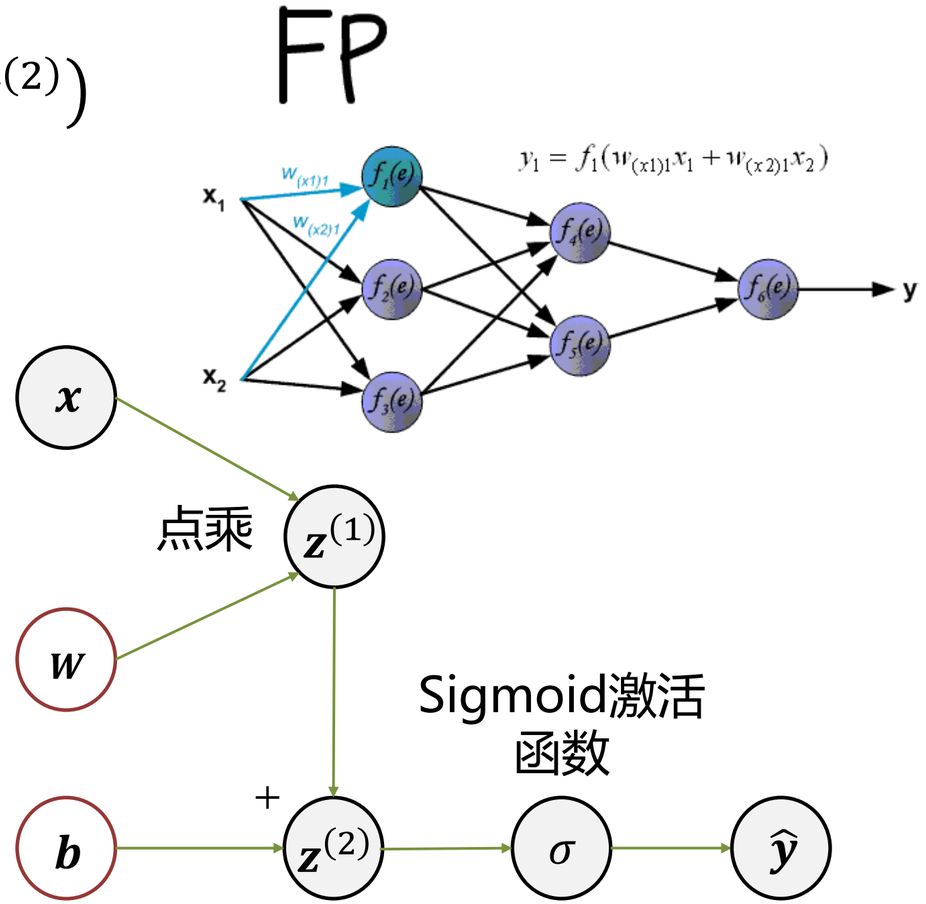
(1) L关于w的偏导数$\frac{\partial L(y,\hat{y})}{\partial w}$，通过链式法则可以计算为：

$$\frac{\partial L(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial \boldsymbol{w}}=\frac{\partial L(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}}\frac{\partial(\sigma(\boldsymbol{z}^{(2)})}{\partial \boldsymbol{z}^{(2)}}\frac{\partial \boldsymbol{z}^{(2)}}{\partial z^{(1)}}\frac{\partial \boldsymbol{z}^{(1)}}{\partial \boldsymbol{w}}$$

(2) 利用梯度下降法，优化与微调参数w与b：

$$\boldsymbol{w}:=\boldsymbol{w}-\alpha\frac{\partial L(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial \boldsymbol{w}}=\boldsymbol{w}-\alpha\left(\frac{\partial L(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}}\frac{\partial(\sigma(\boldsymbol{z}^{(2)})}{\partial \boldsymbol{z}^{(2)}}\frac{\partial \boldsymbol{z}^{(2)}}{\partial \boldsymbol{z}^{(1)}}\frac{\partial \boldsymbol{z}^{(1)}}{\partial \boldsymbol{w}}\right)$$

$$\boldsymbol{b}:=\boldsymbol{b}-\alpha\left(\frac{\partial L(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}}\frac{\partial(\sigma(\boldsymbol{z}^{(2)})}{\partial \boldsymbol{z}^{(2)}}\frac{\partial \boldsymbol{z}^{(2)}}{\partial \boldsymbol{z}^{(1)}}\frac{\partial \boldsymbol{z}^{(1)}}{\partial \boldsymbol{w}}\right)$$



保罗·韦尔博斯（Paul Werbos）于1974年在他哈佛大学的博士论文中首次提出了“超越回归（Beyond Regression）”，也就是反向传播（Backpropogation，BP）算法，成为了BP算法的第一人，荣获IEEE神经网络先驱奖，但当时并未引起广泛的关注。

1986年由大卫·莱姆哈特（David E. Rumelhart）、杰弗里·辛顿（Geoffrey Hinton）和罗纳尔多•威廉姆斯（Ronald J. Williams）共同在《Nature》上发表的论文《Learning representations by back-propagating errors》，它不仅展示了BP算法的原理，还证明了其在多层网络中的强大能力，为多层神经网络的学习训练提供了切实可行的方法，极大地推动了神经网络的研究。

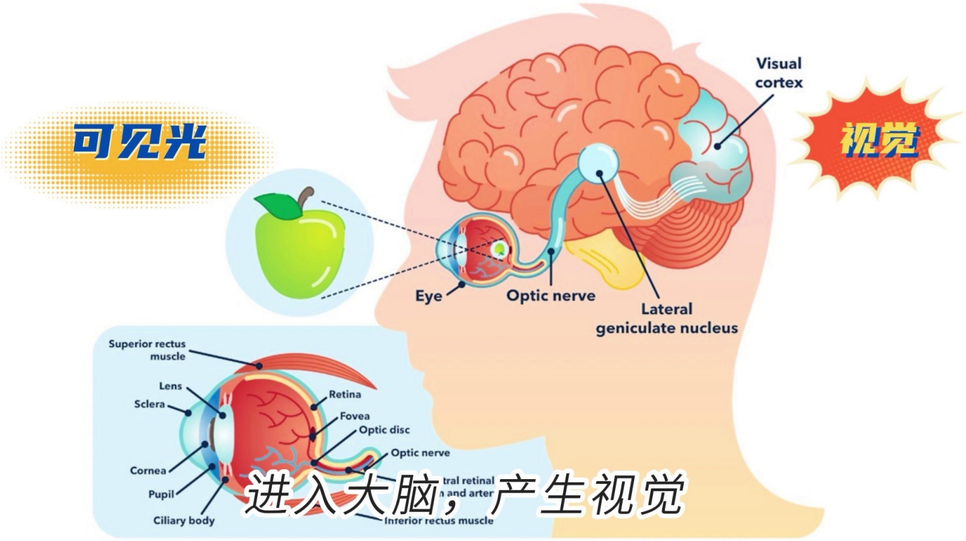
03

图像解析能手——卷积神经网络

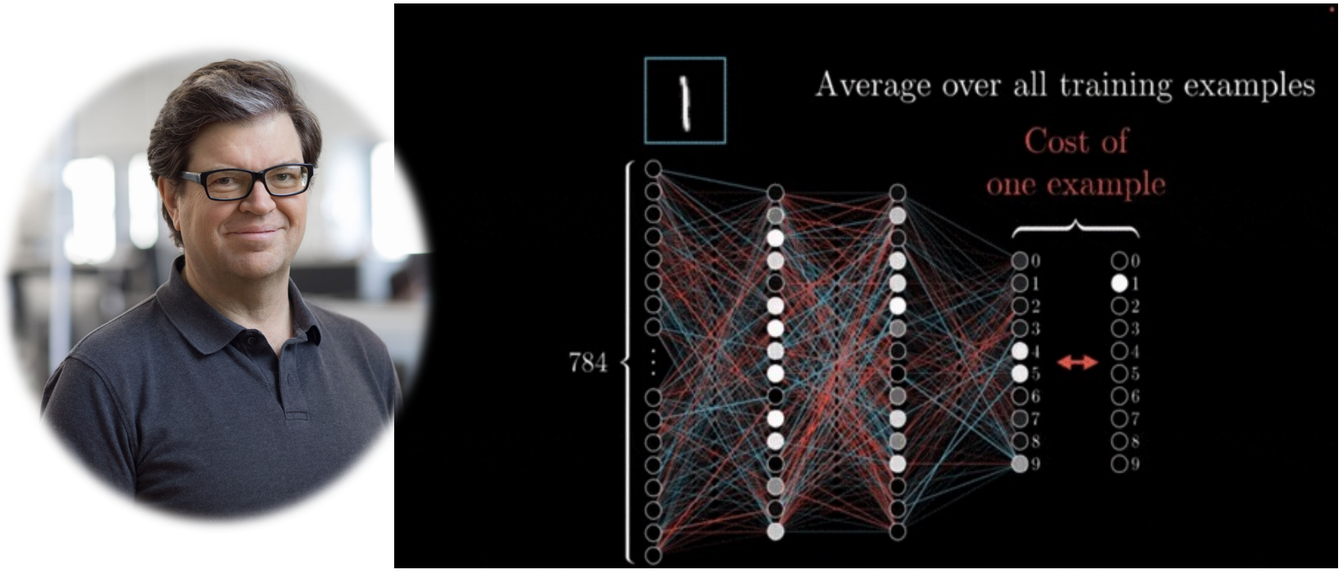
3.1 卷积神经网络的核心组成

- 卷积神经网络（Convolutional Neural Network，CNN）是一种深度学习模型，常用来分析视觉图像。它的出现受到了生物处理过程的启发，因为神经元之间的连接模式类似于动物的视觉皮层组织。

- 图灵奖得主Yann Lecun是CNN的创始人之一， 1998 年提出了 LeNet-5 模型，这是第一个通过卷积神经网络解决手写数字识别的网络，并应用于美国邮政系统。



人脑与视觉



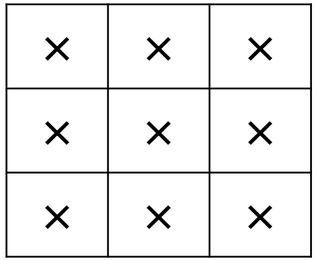
Lecun与他发明的可用于手写字体识别的CNN

3.1 卷积神经网络的核心组成

- 日常生活中，我们使用图像编辑软件中的滤镜来调整图片的颜色、对比度。这些滤镜通过特定的算法对每个像素及其周围的像素进行计算，从而改变图像的外观。

- 类似地，在卷积神经网络中，**卷积操作就像给图像应用一系列的智能滤镜**。每个滤镜（称之为卷积核（Kernel））都通过滑动窗口的方式提取图像中的局部特征，如边缘、纹理与形状等。

图像



不同卷积核

边缘

加噪

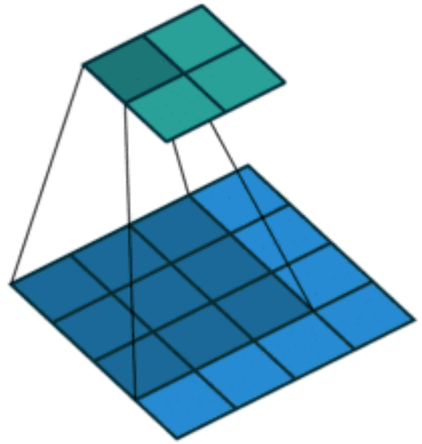
灰度化

3.1 卷积神经网络的核心组成——卷积层：图像的“智能滤镜”

➢ 卷积的计算原理

给定二维图像$\boldsymbol{X} \in R^{H\times W}$，二维卷积核$\boldsymbol{\omega} \in R^{U\times V}$，卷积运算可以表示为：

$$\boldsymbol{Y} = \boldsymbol{\omega} * \boldsymbol{X} = \sum_{u=1}^{U}\sum_{v=1}^{V} \omega_{uv} x_{i-u+1,j-v+1}$$



假设图像尺寸为5×5，卷积核尺寸为3×3，那么卷积计算过程如下：

1	1	1	1	1
-1	0	-3	0	1
2	1	1	-1	0
0	-1	1	2	1
1	2	1	1	1

输入信号 $\boldsymbol{X}$

$*$

1	0	0
0	0	0
0	0	-1

卷积核 $\boldsymbol{\omega}$

$=$

0	2	-1
2	2	4
-1	0	0

输出信号 $\boldsymbol{Y}$

$= 0\times(-1) + (-3)\times 0 + 0\times 0 + 1\times 0 + 1\times 0 + (-1)\times 0 + (-1)\times 0 + 1\times 0 + 2\times 1$

$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

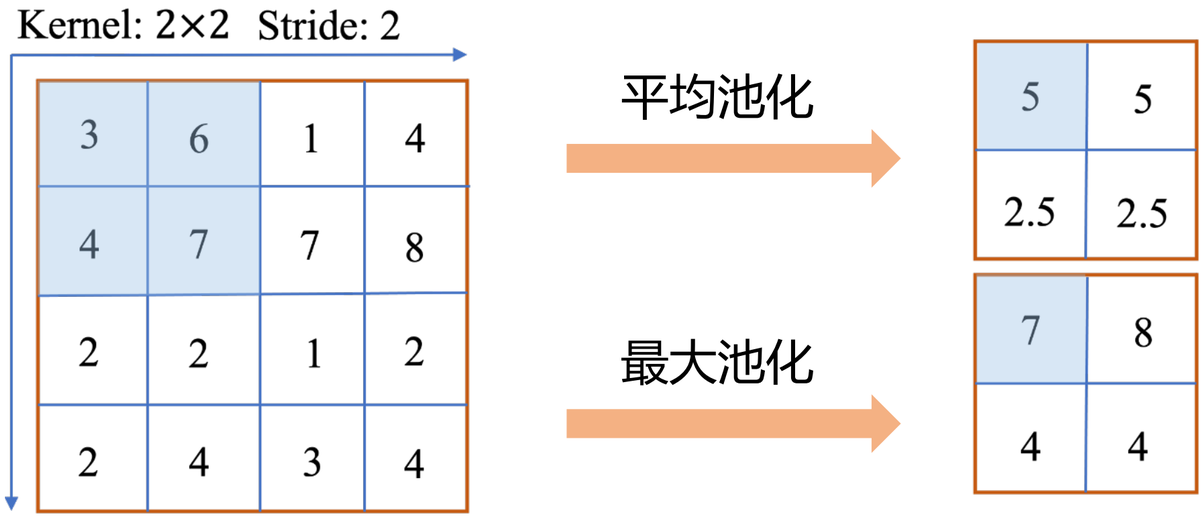
4		

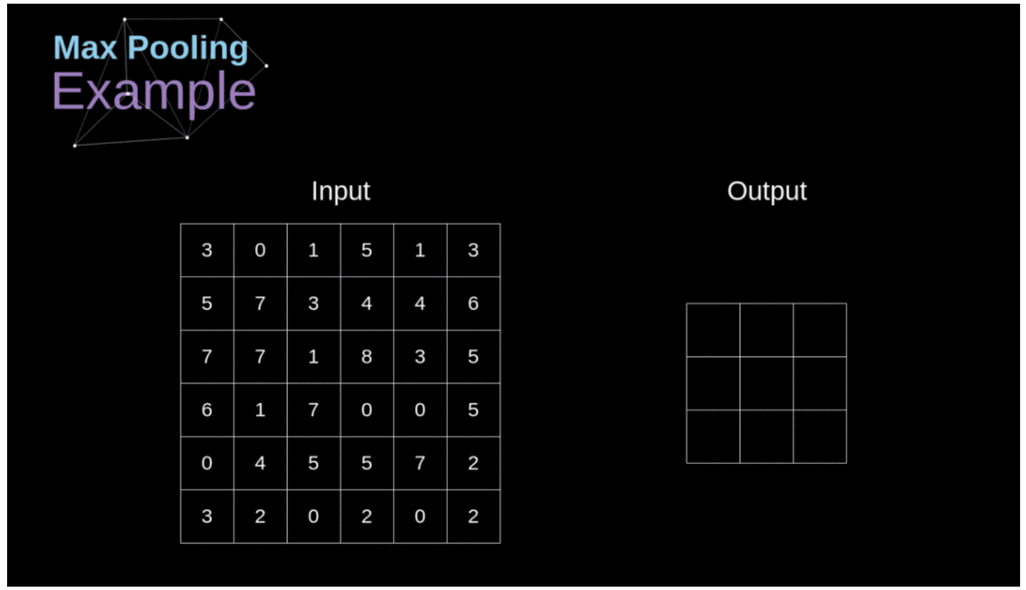
Convolved Feature

3.1 卷积神经网络的核心组成——池化层：信息的“压缩专家”

- **池化（Pooling）**可以形象地看作图像信息的“压缩专家”，利用某一区域的统计信息（均值、中位数、最大值等）替代区域中所有点的取值，简化计算复杂度，增强网络学习特征稳定性，防止过拟合。

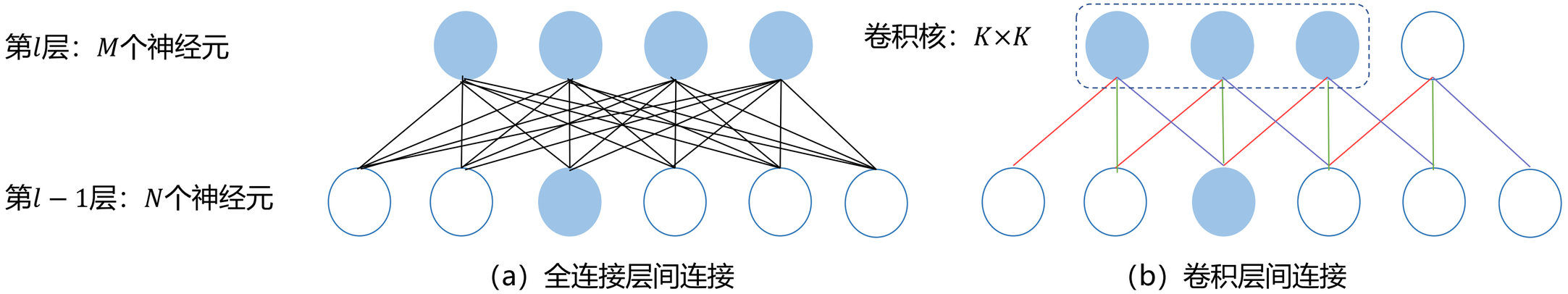
  - **平均池化（Average Pooling）**：计算区域子块所有点平均值代表该区域所有信息。
  - **最大池化（Max Pooling）**：选取区域子块所有点中最大值代表该区域所有信息。





3.1 卷积神经网络的核心组成——主要特点

- **稀疏交互（局部连接、局部感受野）**：在卷积层中的每一个神经元都只和下一层中**某个局部窗口内**的神经元相连，构成一个**局部连接网络**。

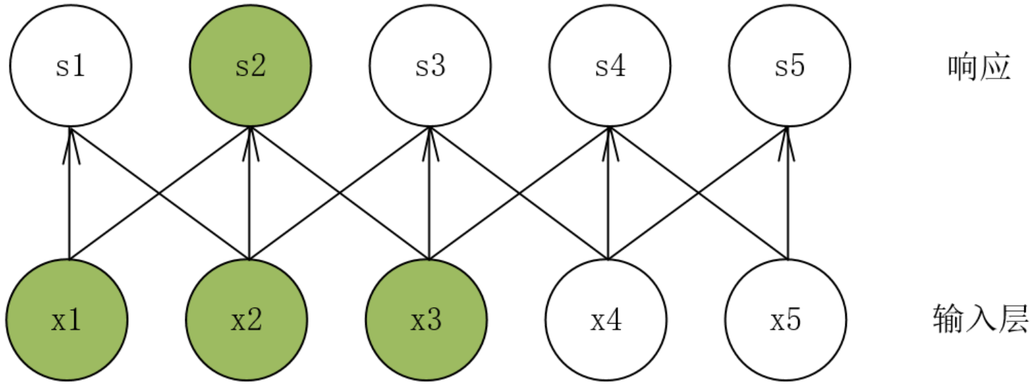


（a）全连接层间连接　　（b）卷积层间连接

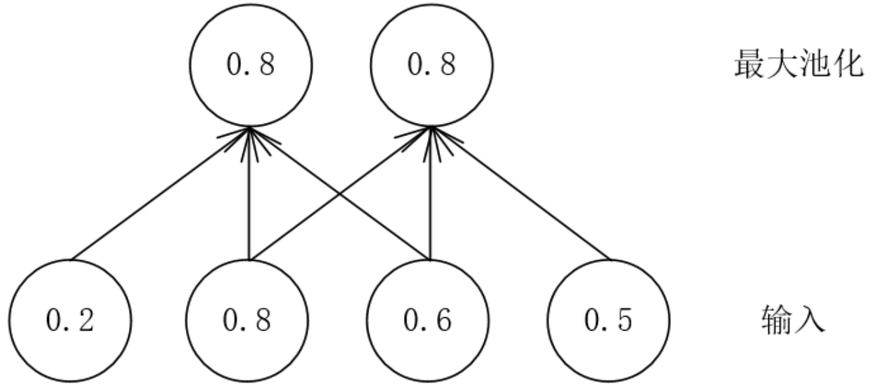
- **权值共享**：**同一个卷积核**在整个图像上进行滑动操作时，其**权重是固定不变的**。也就是说，无论卷积核在图像的哪个位置，它所执行的特征提取操作都是相同的。

3.1 卷积神经网络的核心组成

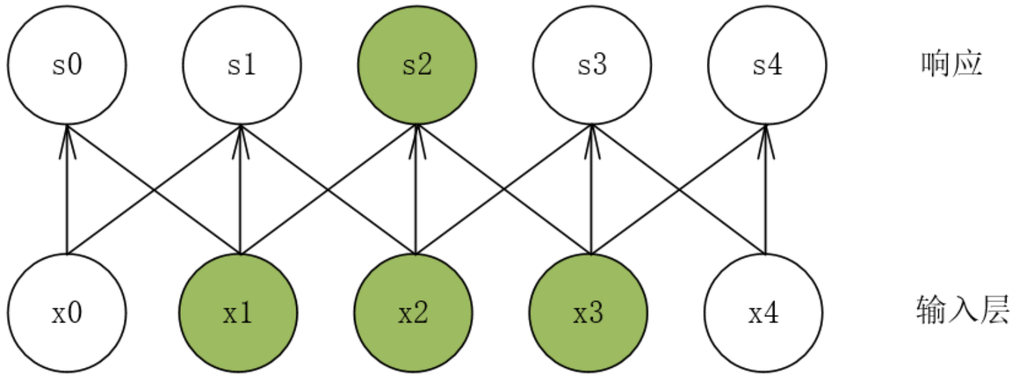
- **平移不变性**：平移不变性是指当输入数据**发生平移时**，卷积网络的输出结果**保持不变或基本不变**，它是由卷积+池化共同实现的。

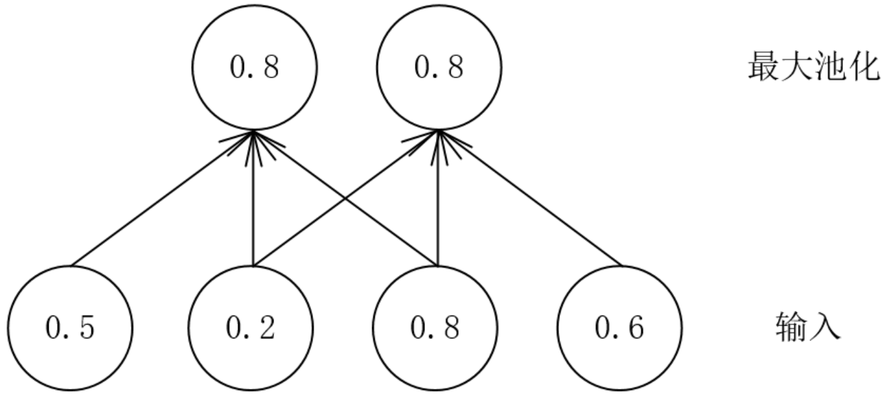


卷积的平移不变性



池化的平移不变性





3.2 卷积神经网络的一般架构

卷积层

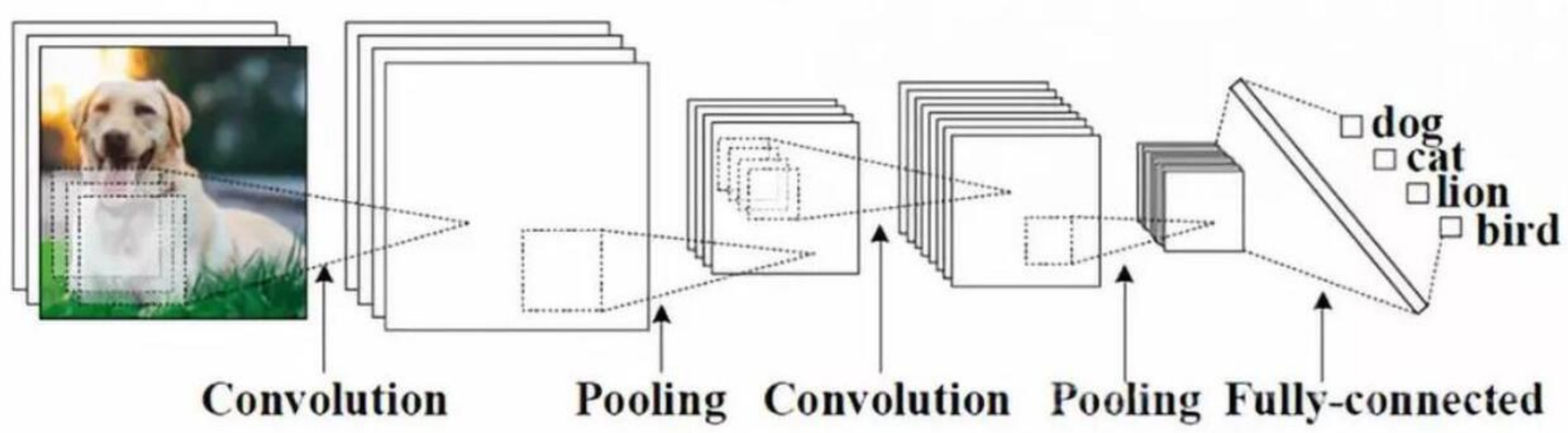
- 提取特征

池化层

- 降维、防止过拟合

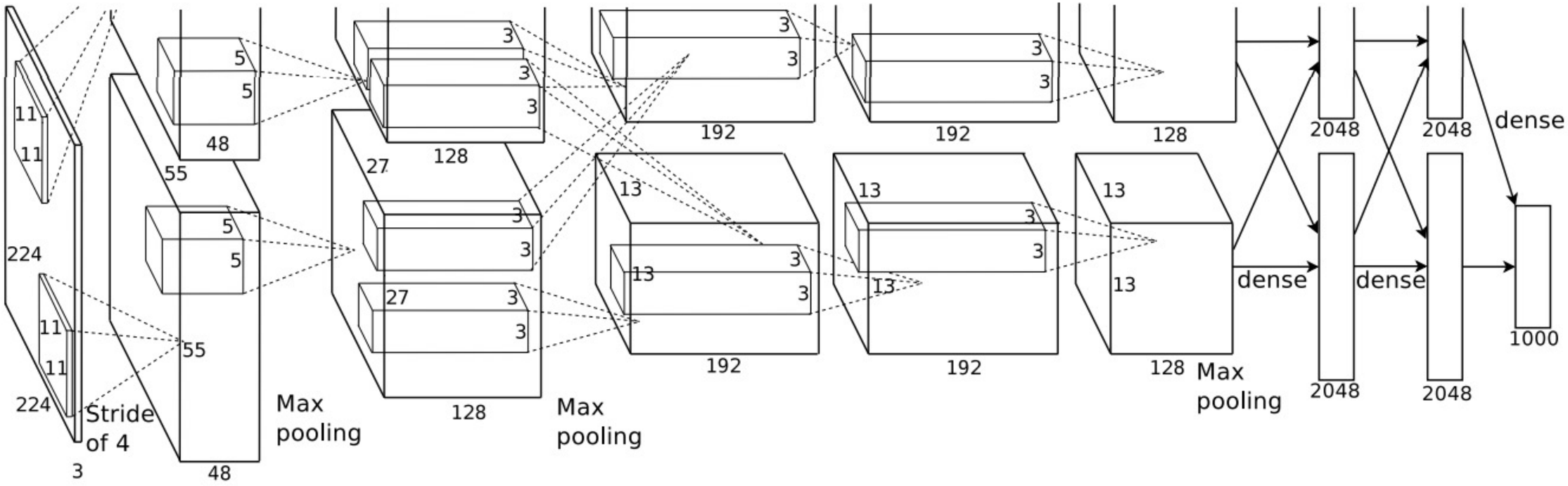
全连接层

- 输出结果



3.2 卷积神经网络的工作过程——以AlexNet为例

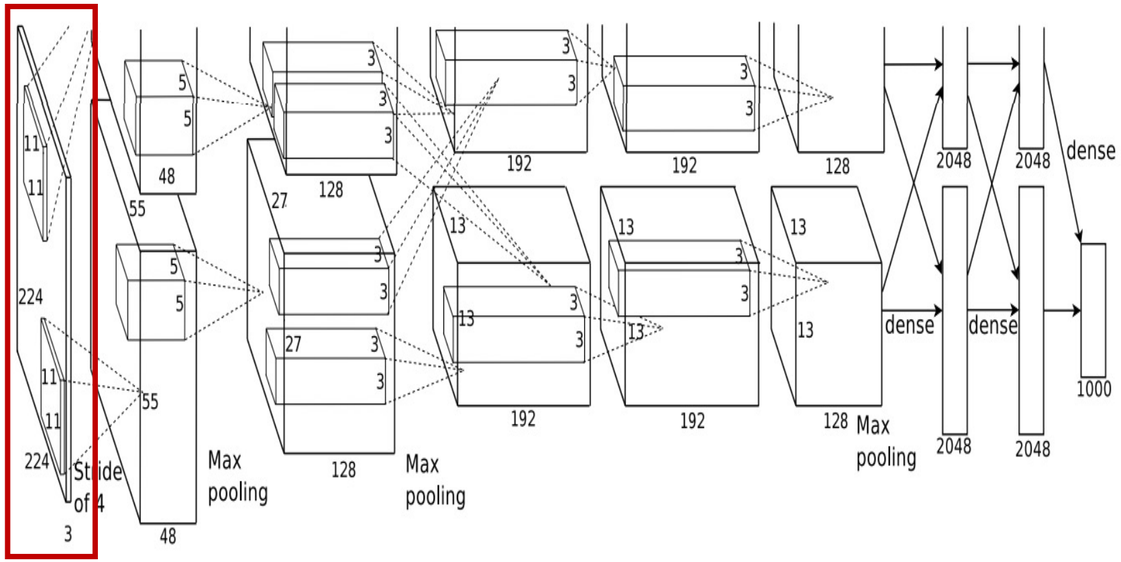
- **AlexNet**：2012年ImageNet竞赛冠军，ReLU激活函数替代Sigmoid或Tanh激活。引入Dropout正则化技术，重叠最大池化策略。（参数：60M）



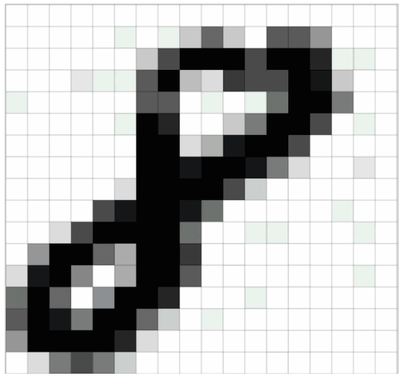
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012).

3.2 卷积神经网络的工作过程——以AlexNet为例

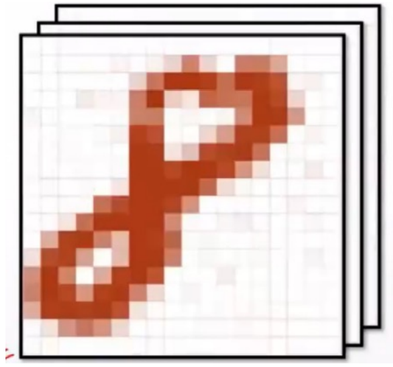
- **输入层**：接收原始图像数据。图像通常由三个颜色通道（红、绿、蓝）组成，也可以是灰度图像，形成一个多维矩阵，表示像素的强度值。一般来说，AlexNet的输入图像尺寸固定为 224×224 像素。如果输入图像的原始尺寸不是224×224，则需要进行预处理操作，如裁剪或缩放，将图像调整到合适的尺寸。

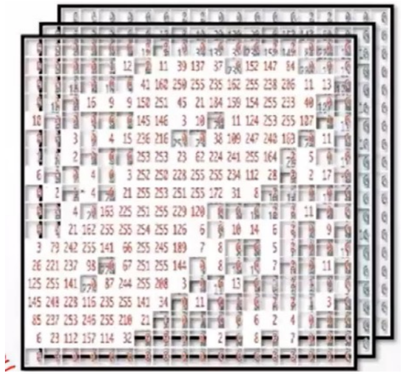


输入层



数字8的灰度图像及其二维矩阵

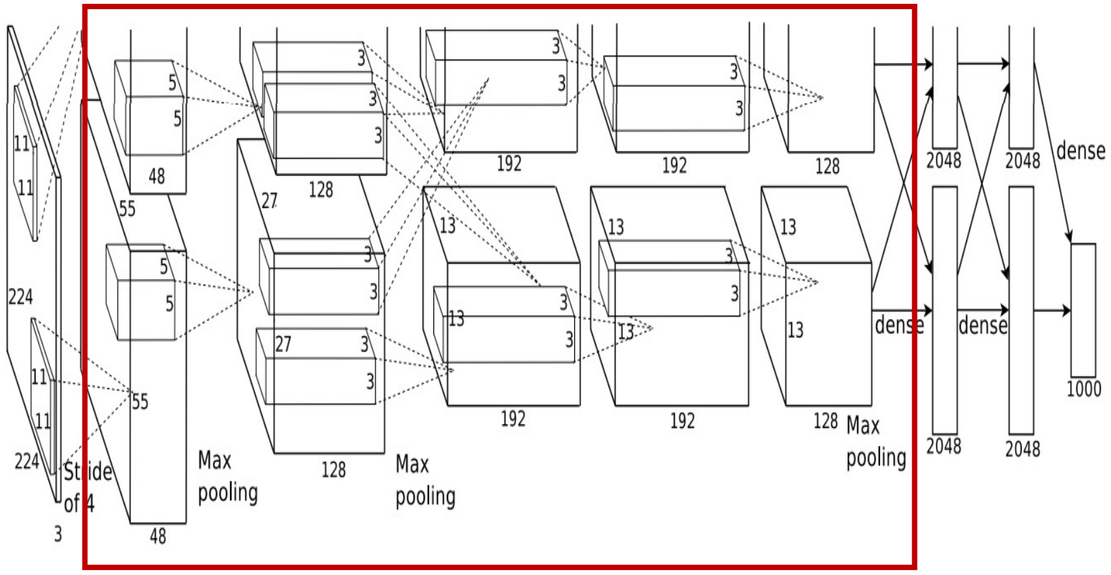




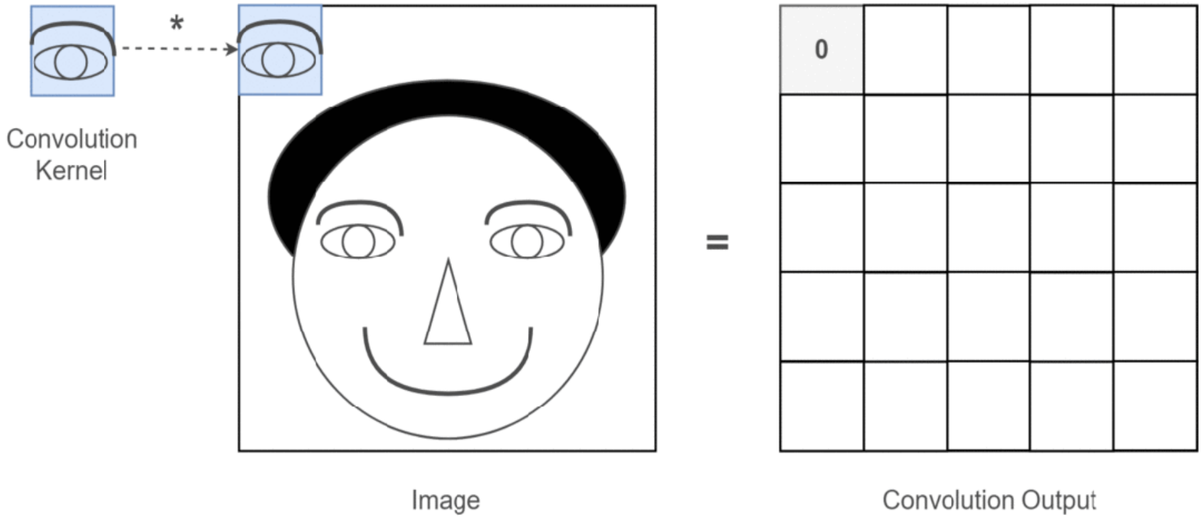
数字8的彩色图像及其三维矩阵

3.2 卷积神经网络的工作过程——以AlexNet为例

- **卷积和池化层**：将输入图像与卷积核进行卷积操作。然后，通过应用激活函数（如ReLU）来引入非线性。同时，池化层通过减小特征图的大小来减少计算复杂性，这有助于提取最重要的特征。多个卷积和池化层的堆叠组成**多层CNN**，以逐渐提取更高级别的特征，深层次的特征可以表示更复杂的模式。



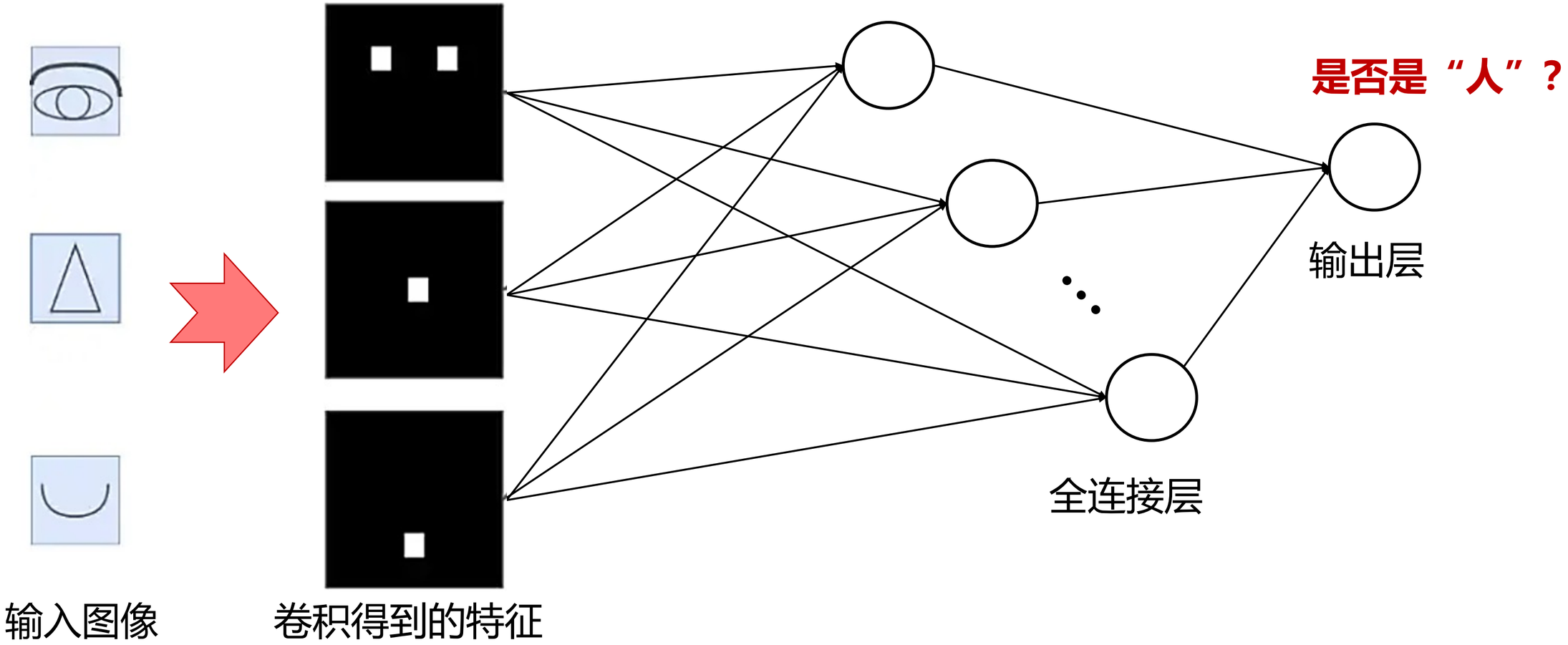
卷积与池化层



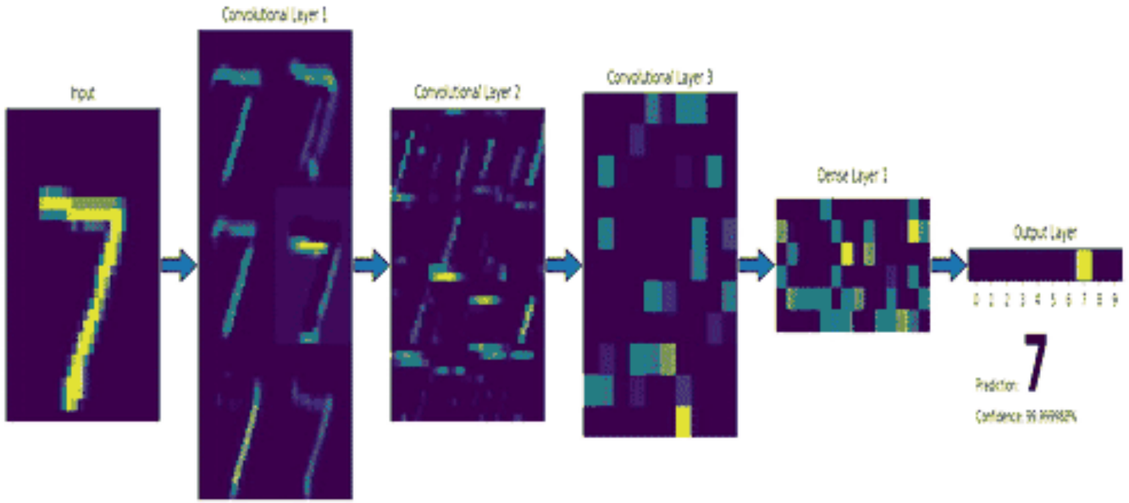
卷积提取人眼特征的过程

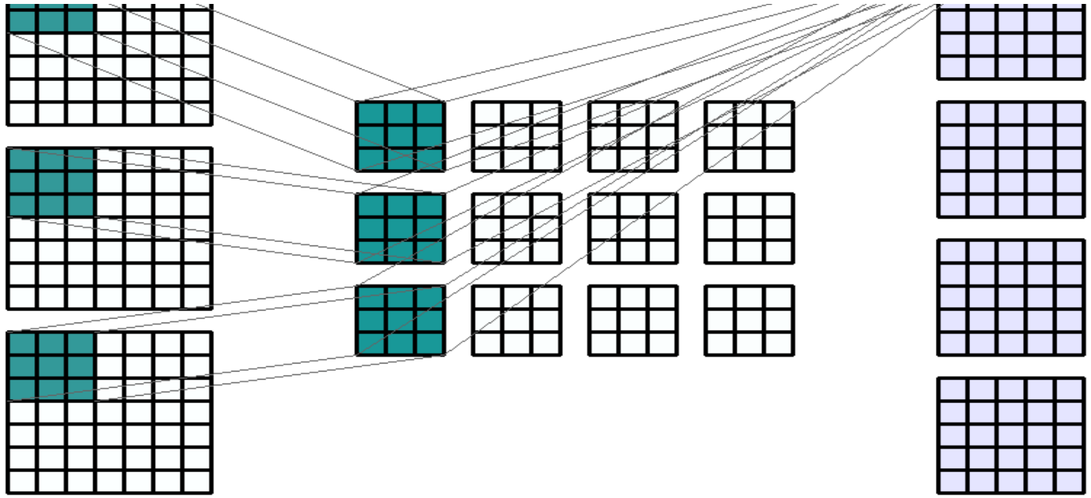
3.2 卷积神经网络的工作过程——以AlexNet为例

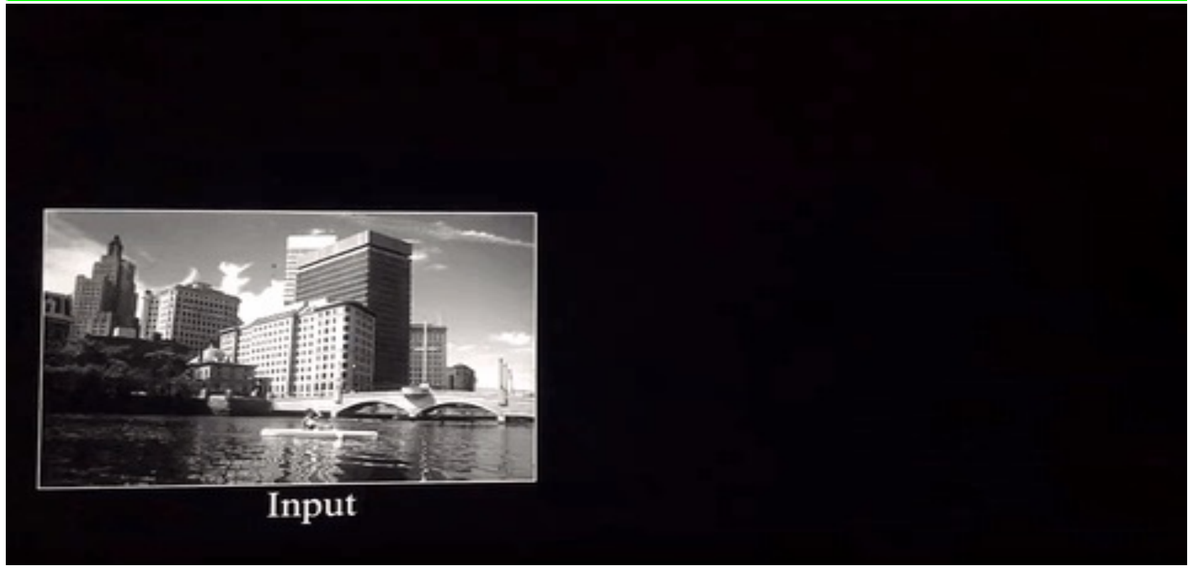
- **全连接和输出层：**全连接层将提取的特征映射进行“展平”，得到网络的最终输出。这可以是一个分类标签、回归值或其他图像识别任务的预测结果。



3.2 卷积神经网络的工作过程——以AlexNet为例

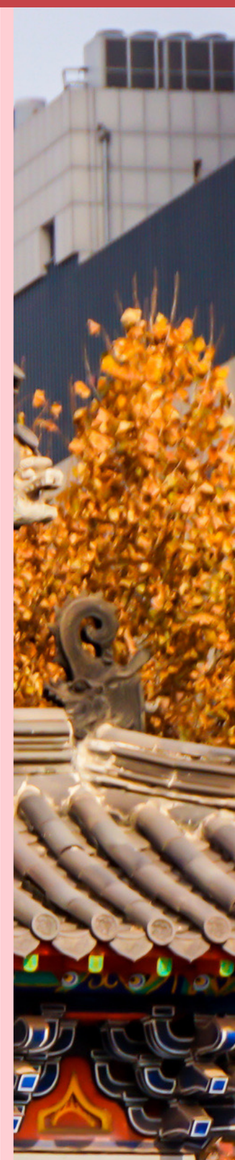




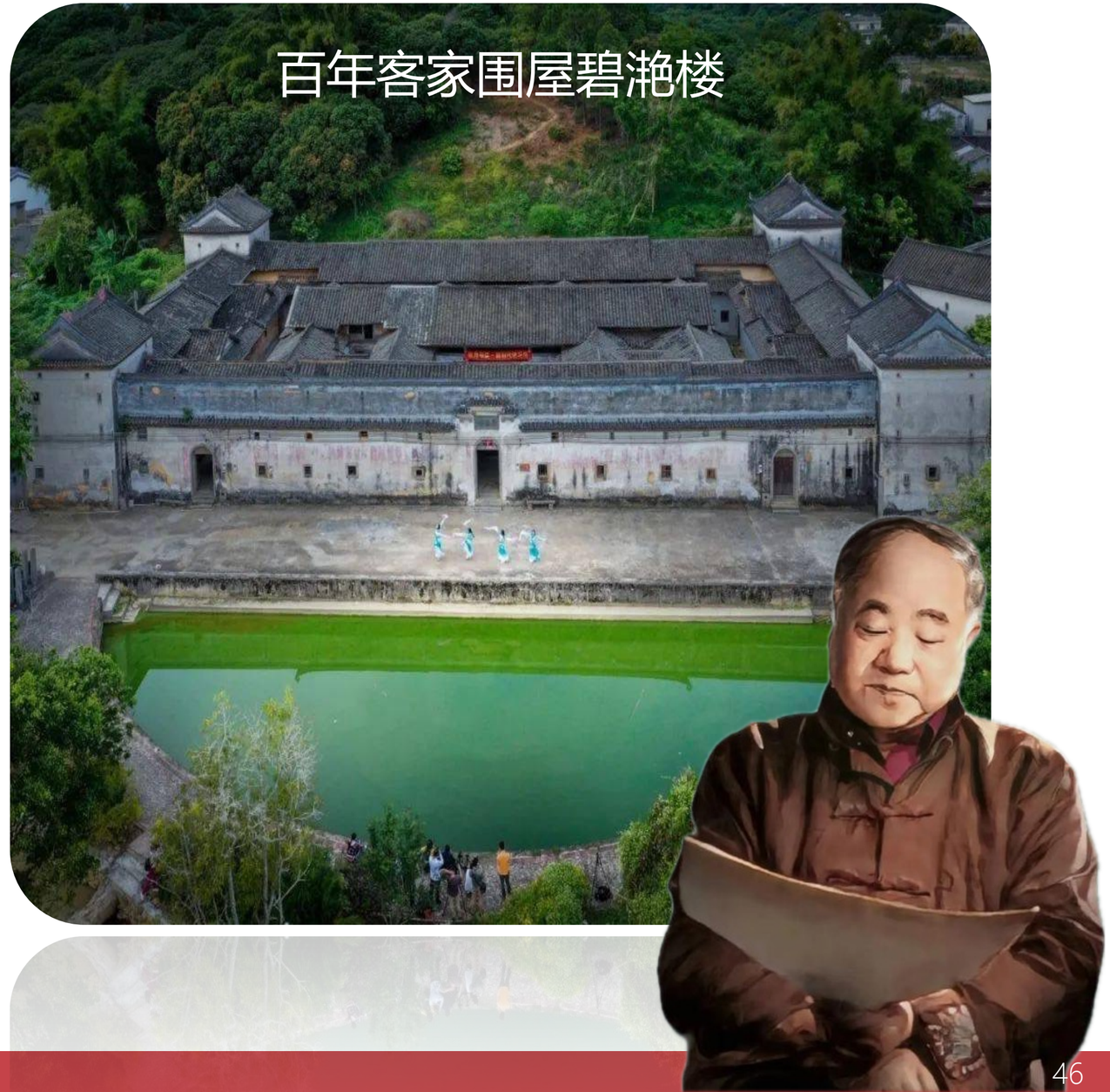


04

序列建模神器 ——循环神经网络

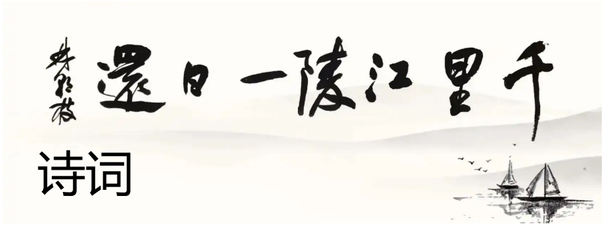


4.1 循环神经网络的工作原理



4.1 循环神经网络的工作原理

- 序列数据是指数据点存在**时间或顺序依赖关系**的一种特殊数据类型。
- **文本型序列数据：**文本数据通常被切分成一个个单词或字符，这些单词按照它们在文本中出现的顺序排列，形成序列。
- **时间序列数据：**按照时间顺序记录的数据，如股票价格、气温、心电图等，每个时间点数据都可以看作序列的一个元素。
- **音频信号：**可以看成一个连续的序列，每个时间点振幅值构成序列的一部分。
- **生物序列数据：**由一系列的碱基组成，而蛋白质序列则是由氨基酸组成，一般包括DNA序列和蛋白质序列。



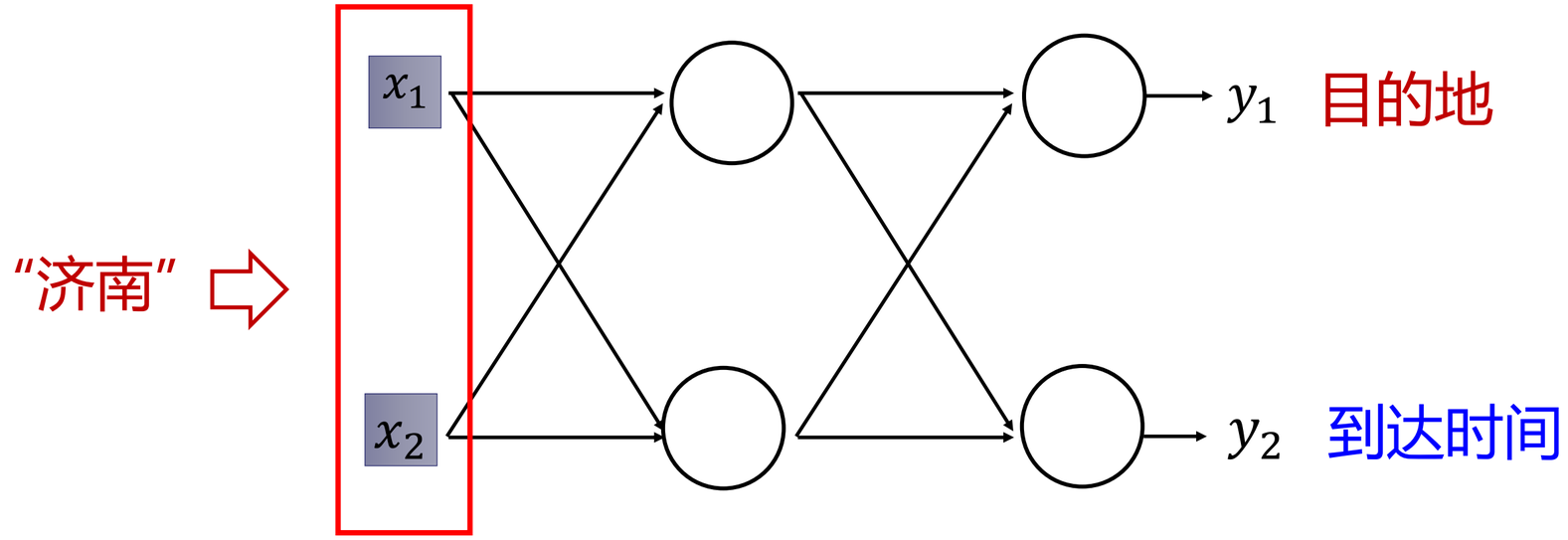
4.1 循环神经网络的工作原理——为什么要记忆?

任务：假如需要利用神经网络从一个文本序列中预测目的地与到达时间。

输入：一个词向量。 例如：我/预计/到达/济南/的/时间/为/2月18日

输出：该词属于目的地或到达时间的概率。

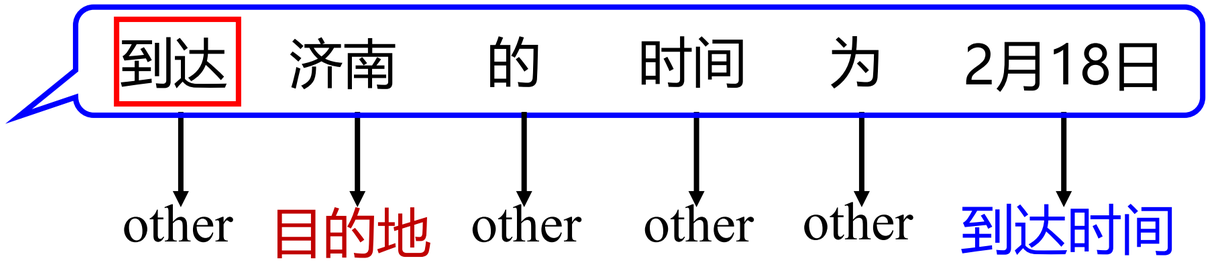
模型：



4.1 循环神经网络的工作原理——为什么要记忆?

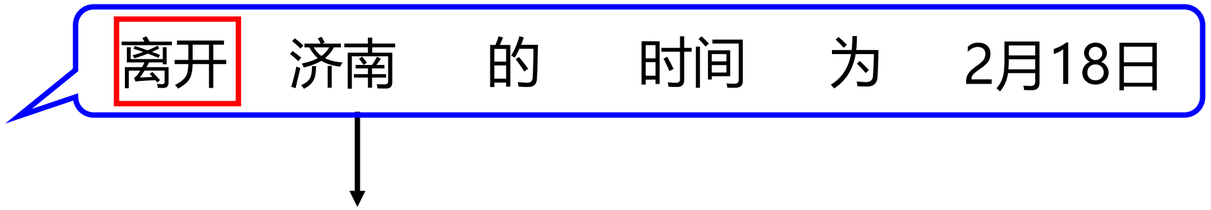
➢ 前馈神经网络的局限性

示例1:



神经网络需要记忆!

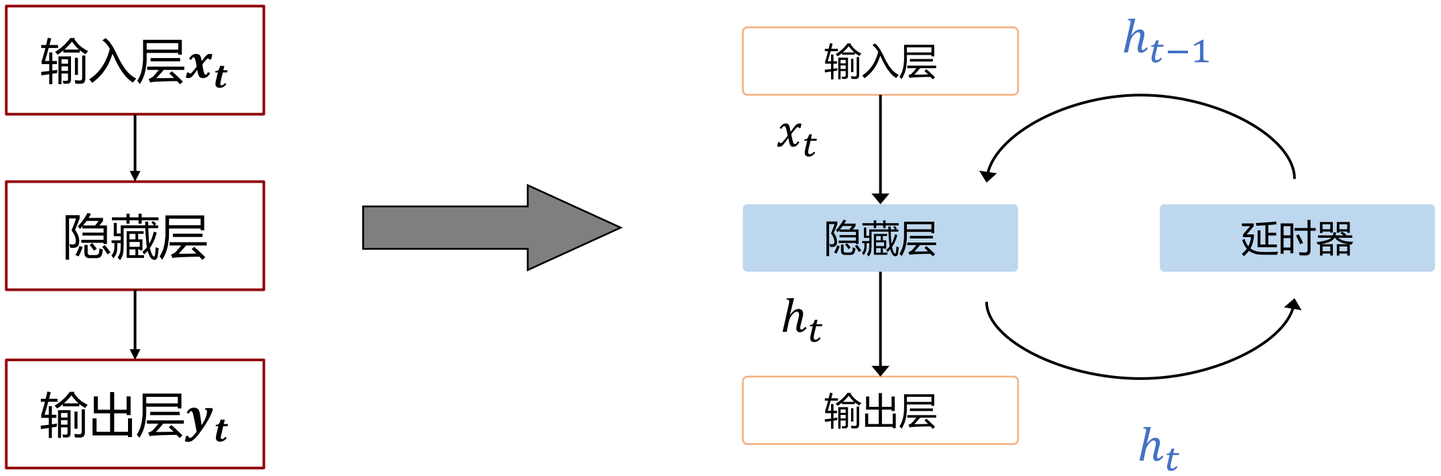
示例2:



4.1 循环神经网络的工作原理

➢ **如何赋予网络记忆功能？ ⇨ 循环神经网络（Recurrent Neural Network，RNN）**

□ **方案：**具体来说，RNN会在每一步处理输入数据x_t的同时，设置延时器W，存储并将上一步的隐藏状态h_{t-1}传递到下一步，形成一种“信息回流”。这就像一个人每次做出决策时都会结合新的输入和过去的记忆。

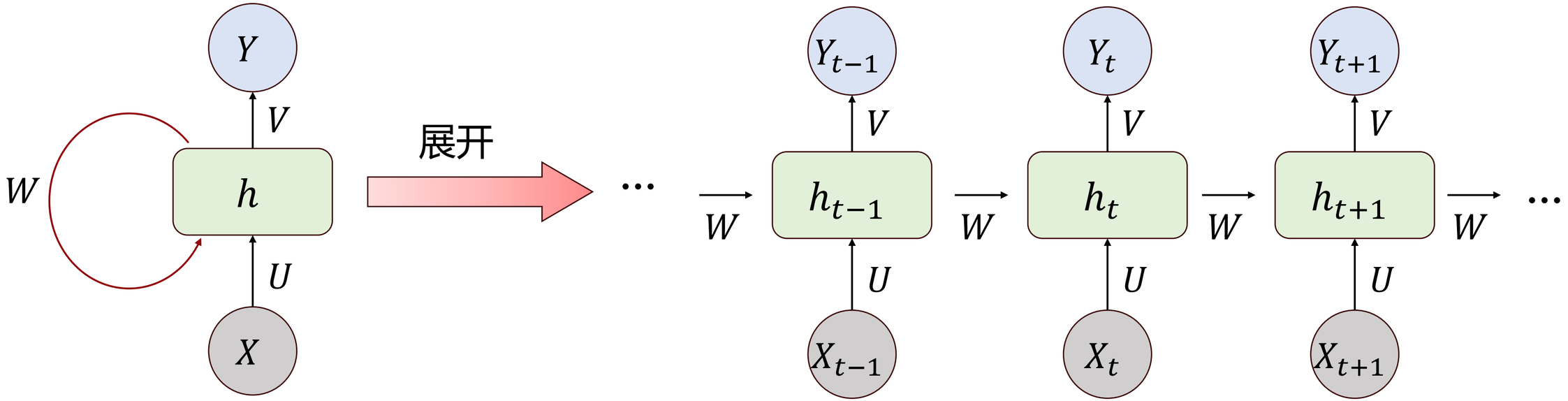


- 初始状态 $h_0 = 0$
- 隐藏状态 h_t 由前馈神经网络 f 计算得到：

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

4.1 循环神经网络的工作原理——信息传递方式

相比于前馈神经网络，RNN网络结构不是数据简单地从输入层到输出层的单向流动，而是通过一个**循环机制**让信息不断流动起来，形成类似于“信息接力赛”的效果。



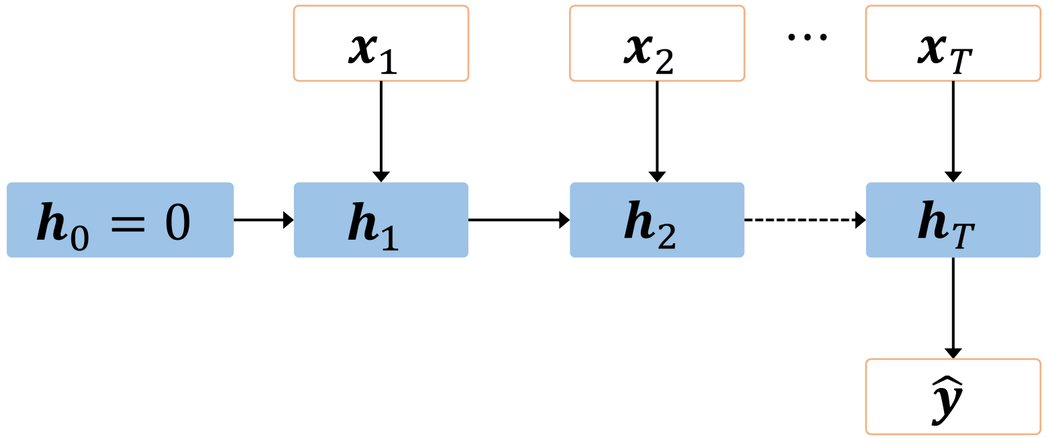
- **时间步**：输入序列中每个时刻的数据点在RNN中的一个处理单元，例如句子中的一个单词或者时间序列中的一个时点。
- **隐藏态**：存储了序列中已经处理过的部分的信息，其作用类似于“记忆”。

4.1 循环神经网络的工作原理——信息传递方式

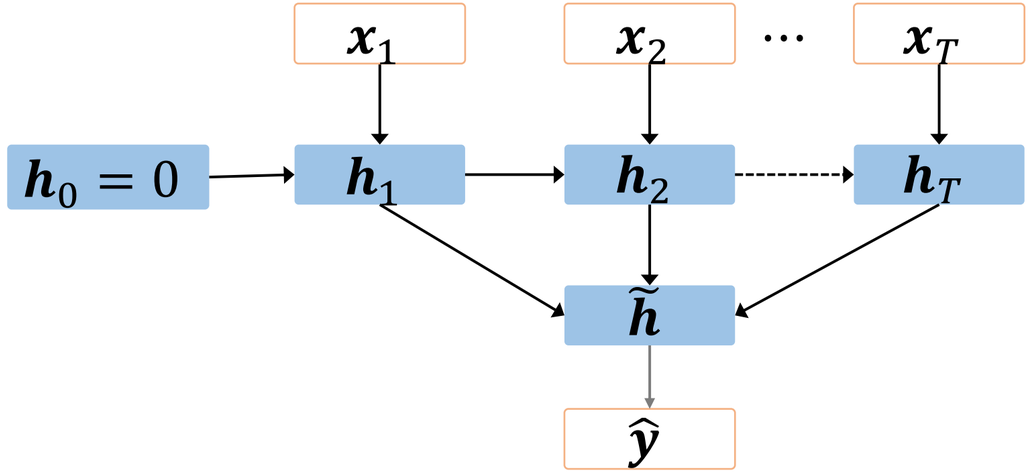
➢ 扩展：如何用RNN解决序列预测任务

□ 序列到类别模式：用于序列数据的分类问题，输入为序列，输出为类别。

给定长度为T的输入序列数据$\boldsymbol{x}_{1:T} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$，要求输出类别为$\boldsymbol{y} \in \{1,2 \ldots, N\}$。



(a) 输入序列按时间步展开，预测分类结果

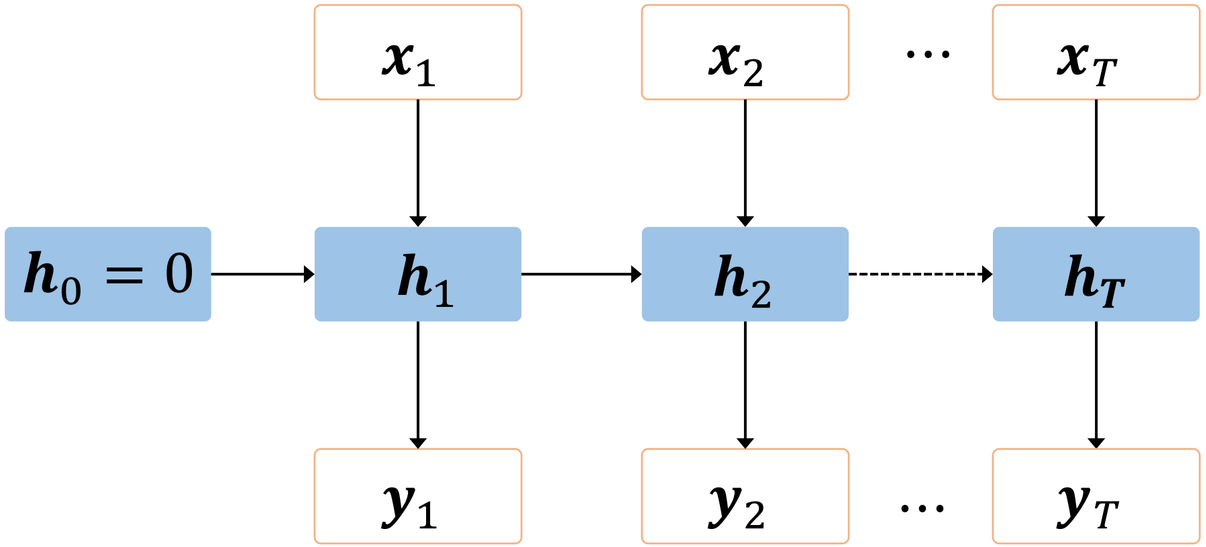


(b) 输入序列按时间步展开，按时间长度进行平均采样

4.1 循环神经网络的工作原理——信息传递方式

➢ 扩展：如何用RNN解决序列预测任务

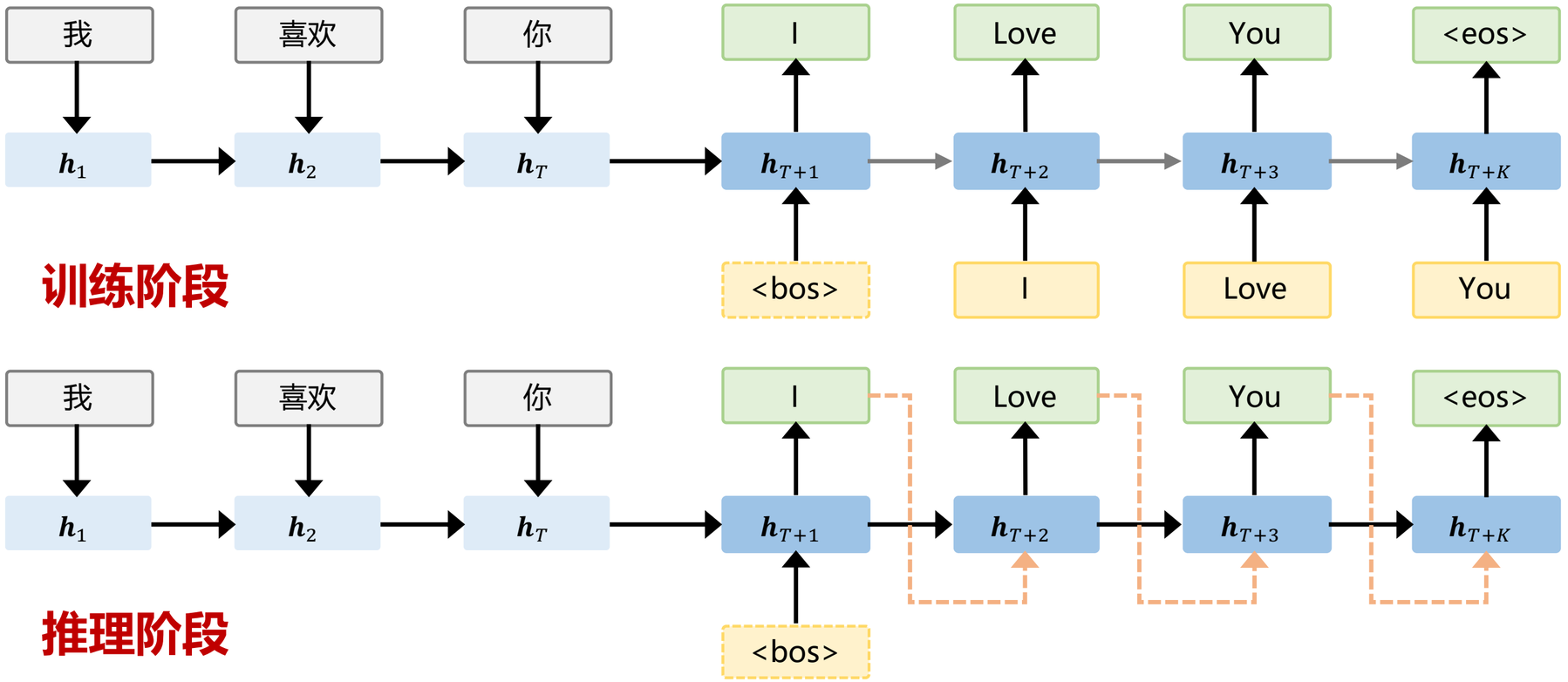
□ **同步序列到序列模式：**输入序列和输出序列保持同步，每个时间步输出只依赖当前时间步的输入和之前的上下文信息，而不是依赖于整个输入序列。



输入与输出序列长度相同，常用于**词性标注任务**

4.1 循环神经网络的工作原理——信息传递方式

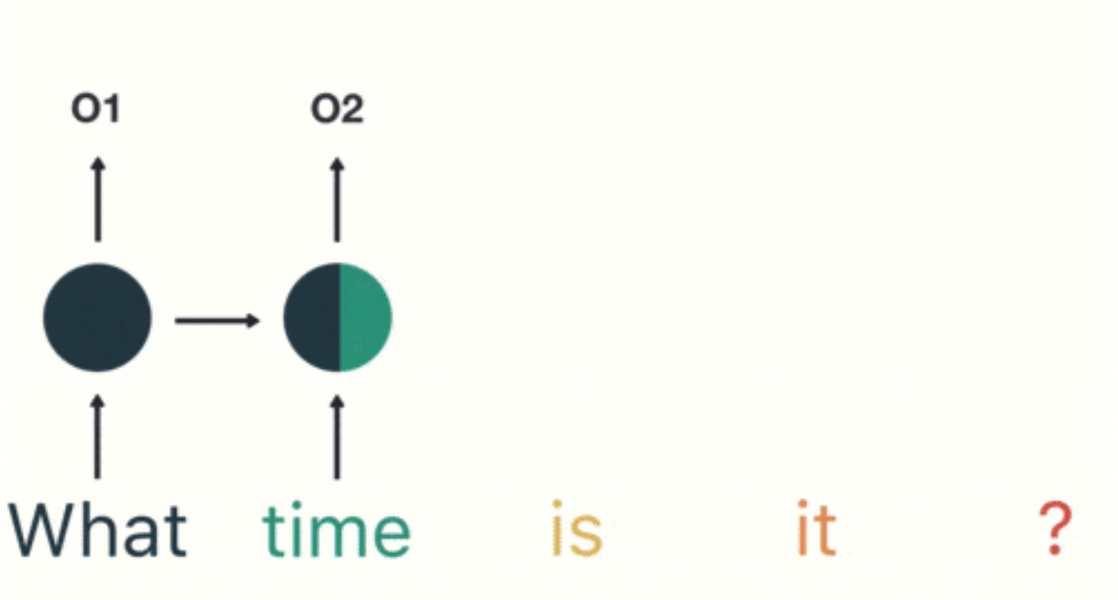
- 扩展：如何用RNN解决序列预测任务
  - **异步序列到序列模式：**输入序列和输出序列不需要同步进行处理，也不需要保持相同长度，常用于**机器翻译任务。**

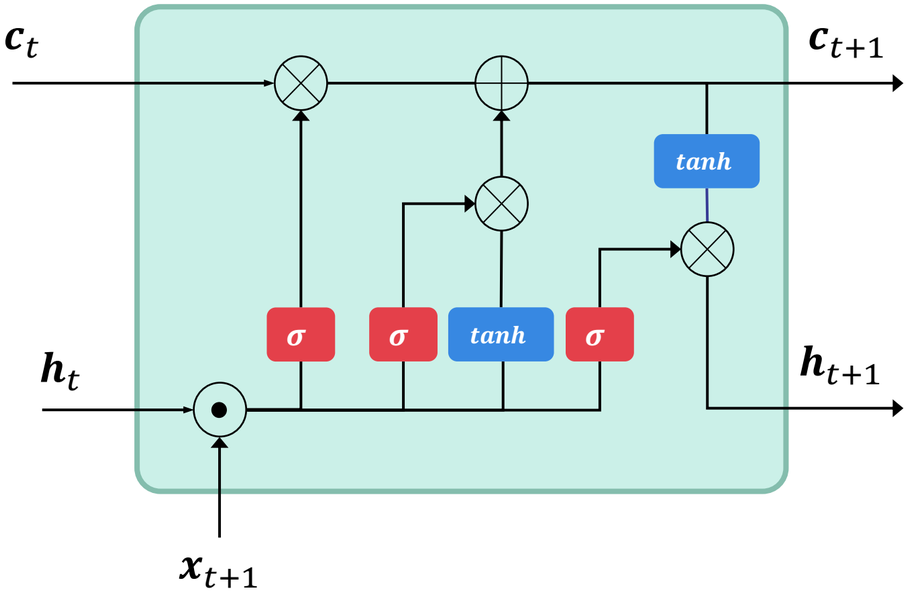


4.2 循环神经网络的变体——长短期记忆网络

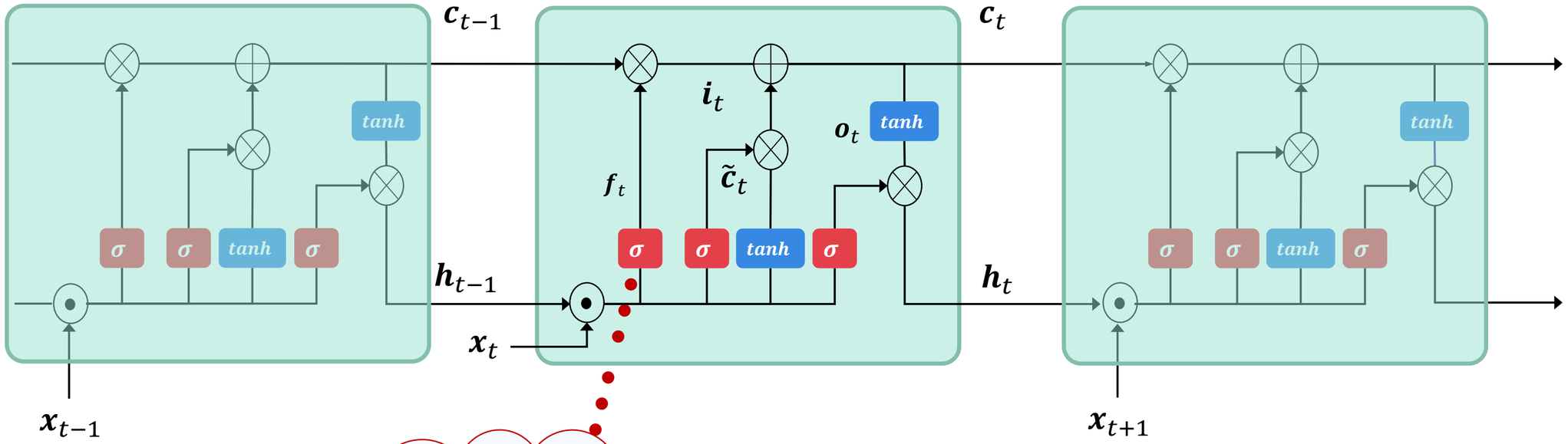
问题：RNN在处理信息的时候，如果碰到的内容太长，在处理到后面内容的时候，可能已经遗忘前面所包含的信息，这会影响神经网络的训练效果。 **长程依赖问题**

定义：LSTM全称是长短期记忆网络（Long Short-Term Memory Network），是一种基于循环神经网络的架构。通过引入**门控机制**解决RNN中的长程依赖问题。

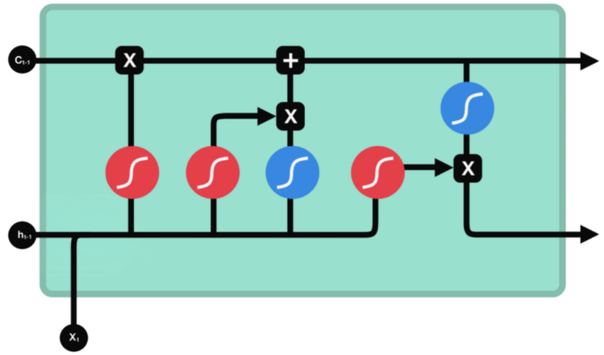




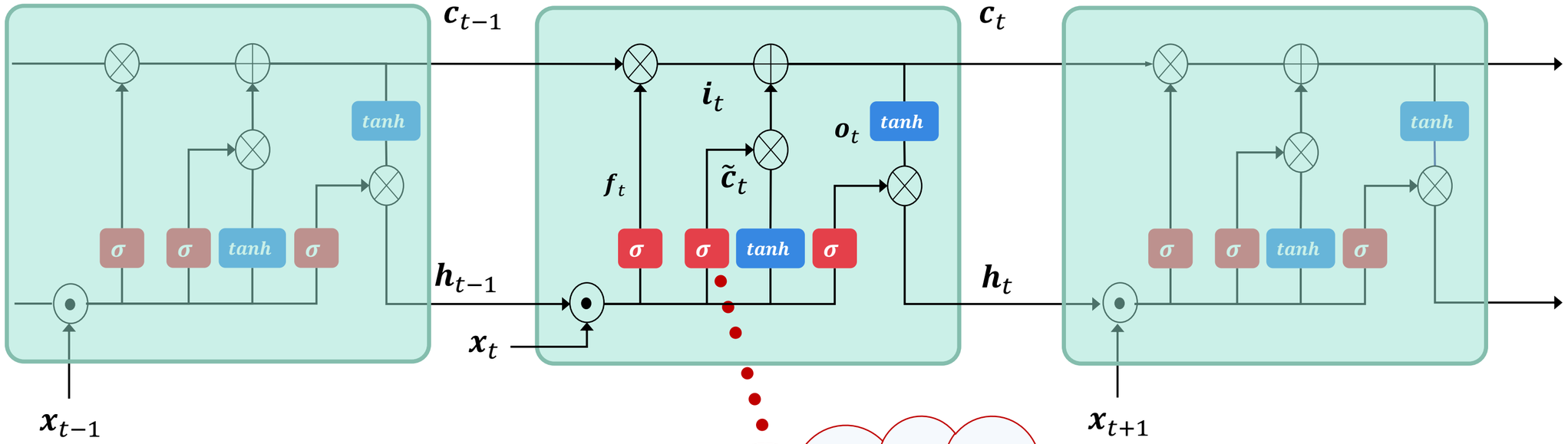
4.2 循环神经网络的变体——长短期记忆网络

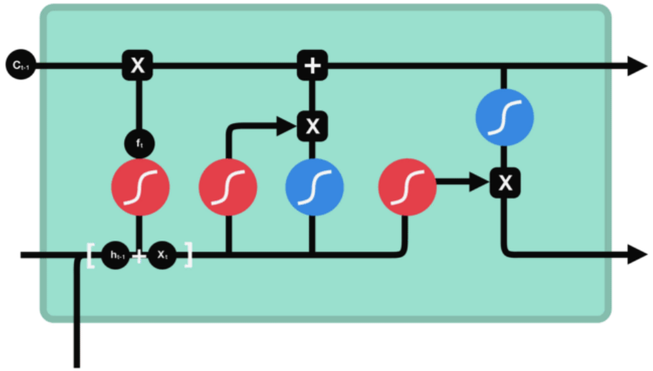


遗忘门：帮助网络决定哪些旧的记忆应该被丢弃，以便为新的信息腾出空间。



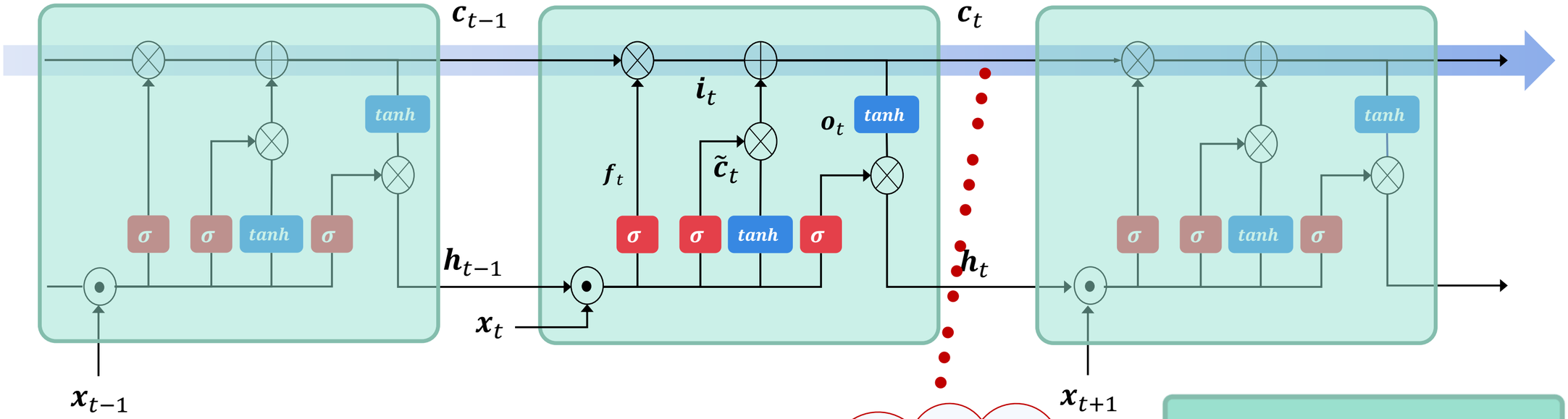
4.2 循环神经网络的变体——长短期记忆网络



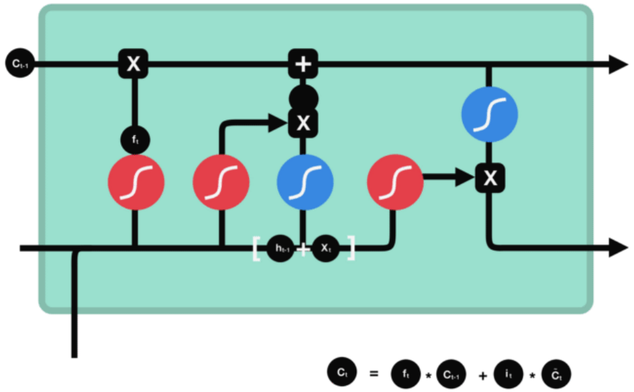


输入门：决定哪些新信息会被添加到记忆中。

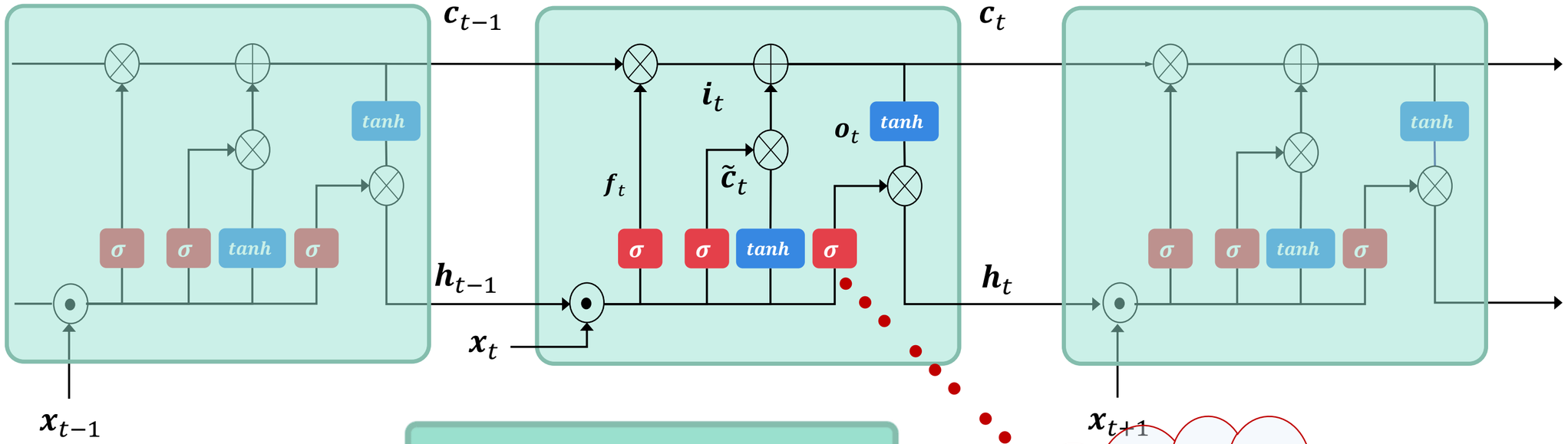
4.2 循环神经网络的变体——长短期记忆网络

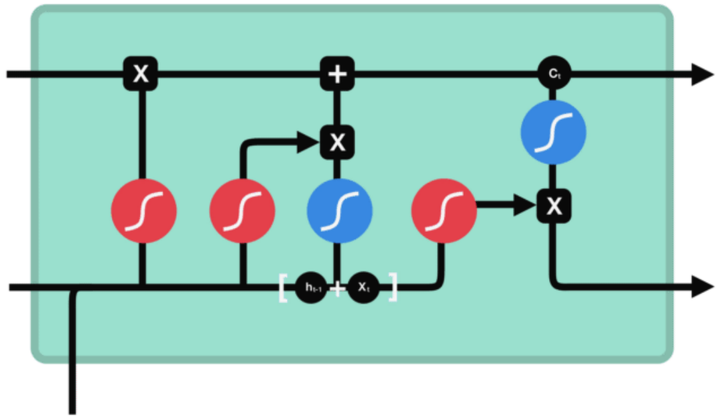


更新单元状态(cell state)，让它来保存长期的状态记忆。



4.2 循环神经网络的变体——长短期记忆网络

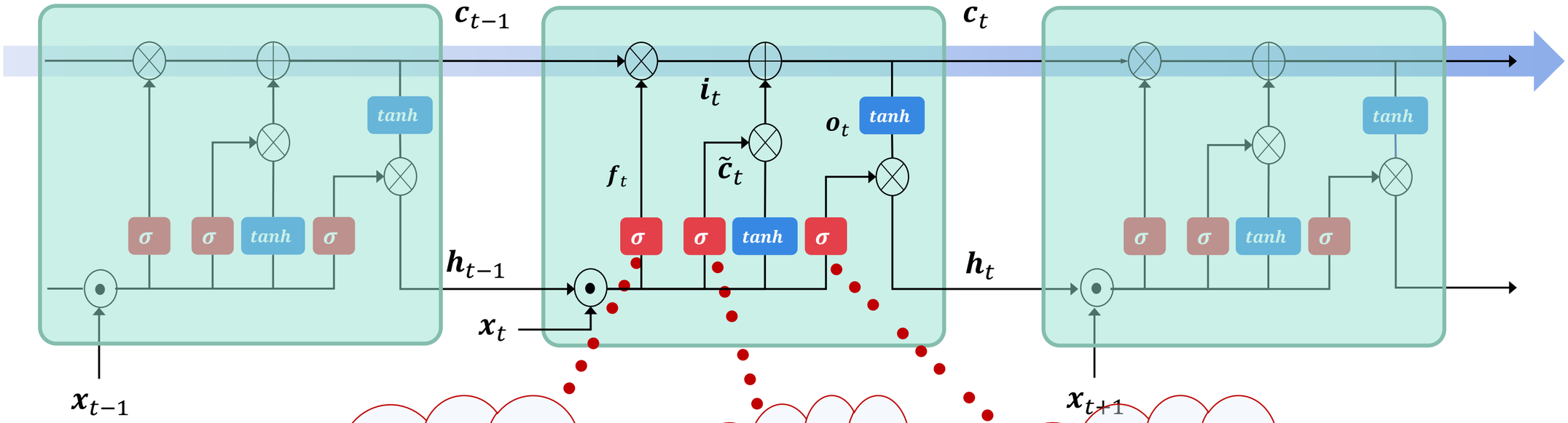




输出门：决定了哪些部分的记忆会被用来生成当前的输出。

4.2 循环神经网络的变体——长短期记忆网络

增加状态c，称为单元状态(cell state)，让它来保存长期的状态



遗忘门：帮助网络决定哪些旧的记忆应该被丢弃，以便为新的信息腾出空间。

输入门：决定哪些新信息会被添加到记忆中。

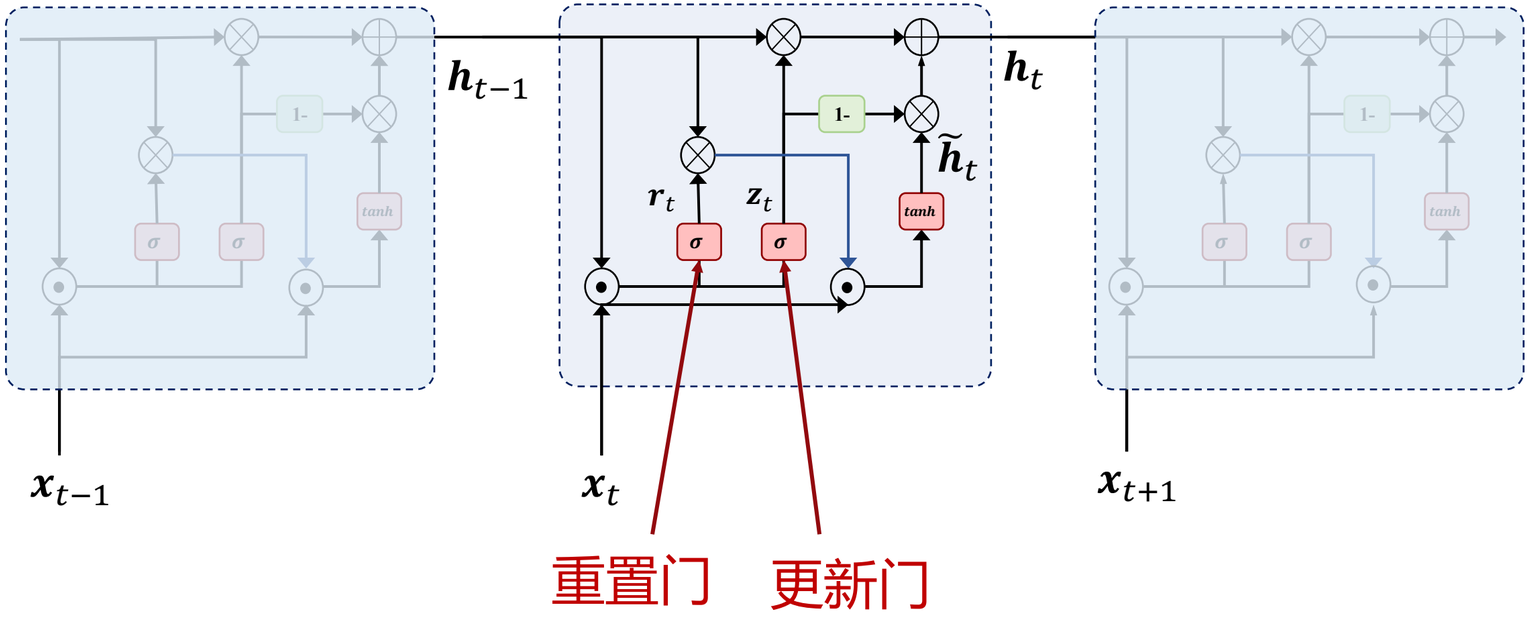
输出门：决定了哪些部分的记忆会被用来生成当前的输出。

4.2 循环神经网络的变体——门控循环单元

问题：LSTM形式较为复杂，同时也存在着训练时间和预测时间较长的问题。

定义： GRU全称是门控循环单元（Gated Recurrent Unit），是LSTM结构的升级版本，不引入额外记忆单元，用更新门（Update Gate）和重置门（Reset Gate）控制信息流动。

GRU模块



- **更新门**：将LSTM中的输入门与遗忘门整合在一起。
- **重置门**：允许模型有选择性地忘记过去信息，专注于新输入。

05

注意力即一切：Transformer架构

5.1 Transformer核心组成

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）

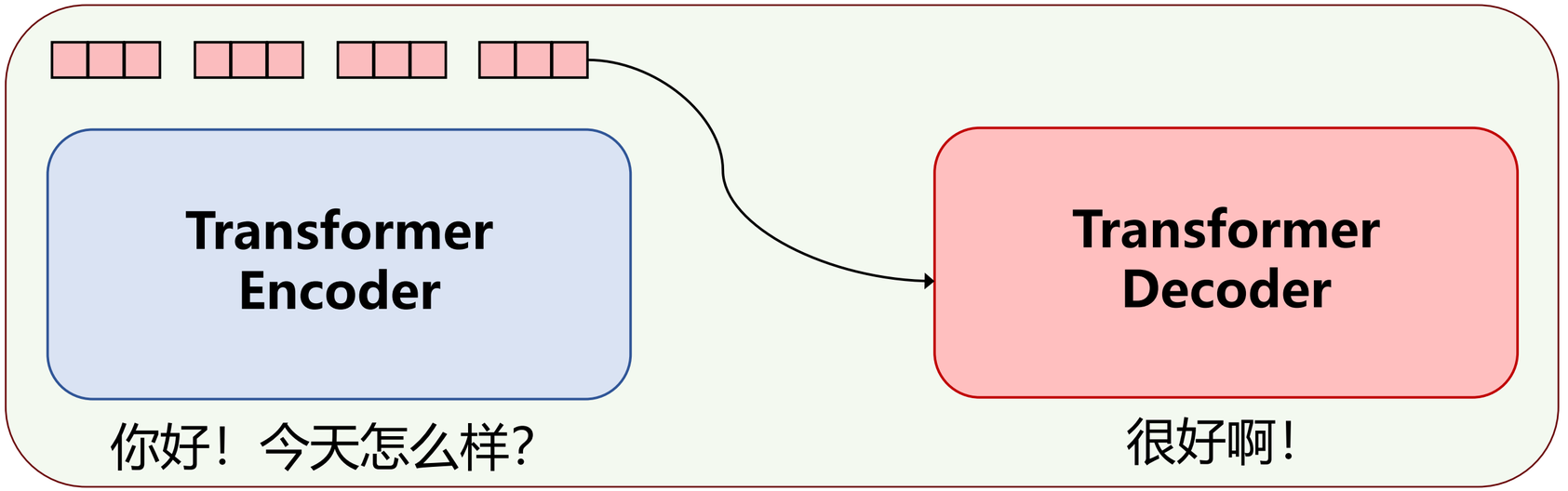
嗯，用户问的是人工智能发展这么快，应该培养孩子哪些能力才能适应未来社会。这个问题挺常见的，但需要仔细分析。首先，用户可能是家长或者教育工作者，关心孩子的未来竞争力。他们可能已经意识到传统教育可能不够，但不确定具体该怎么做。

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

5.1 Transformer核心组成

- 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

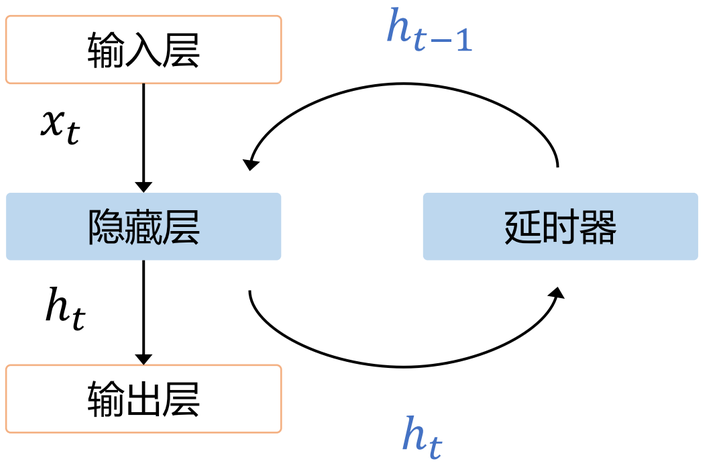
Google BERT

OpenAI GPT

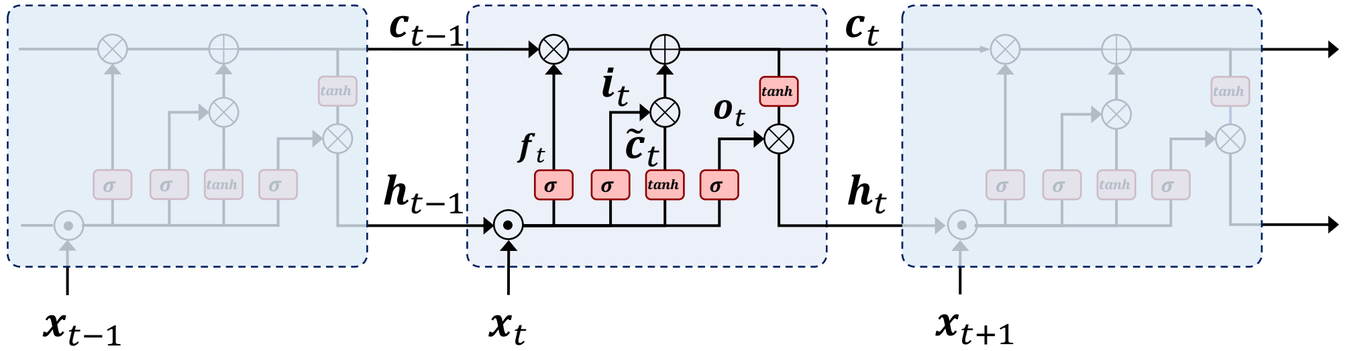
搜索引擎

5.1 Transformer核心组成

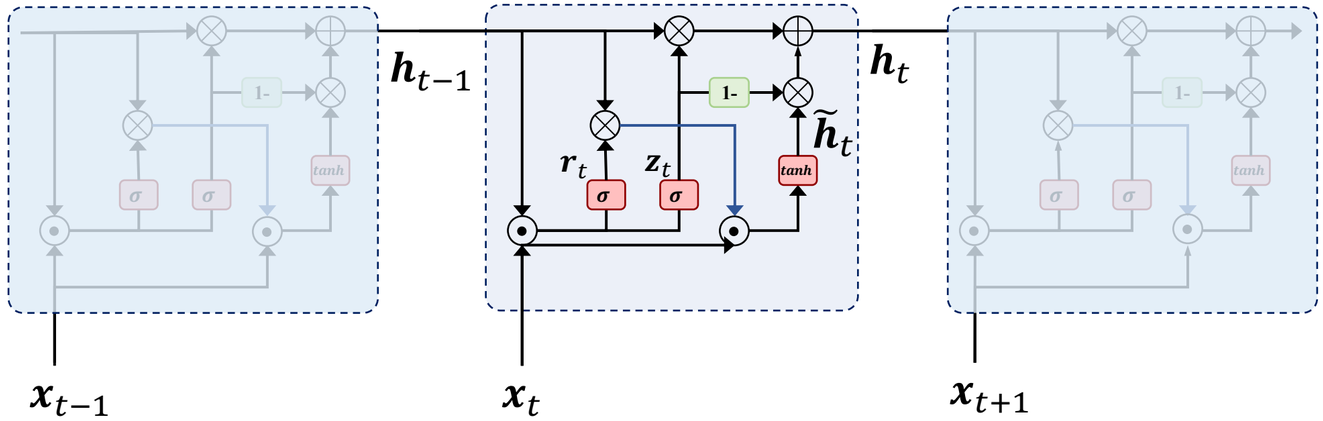
为什么Transformer能够迅速取代RNN、LSTM、GRU等一系列序列模型呢?



RNN（Recurrent Neural Network，循环神经网络）

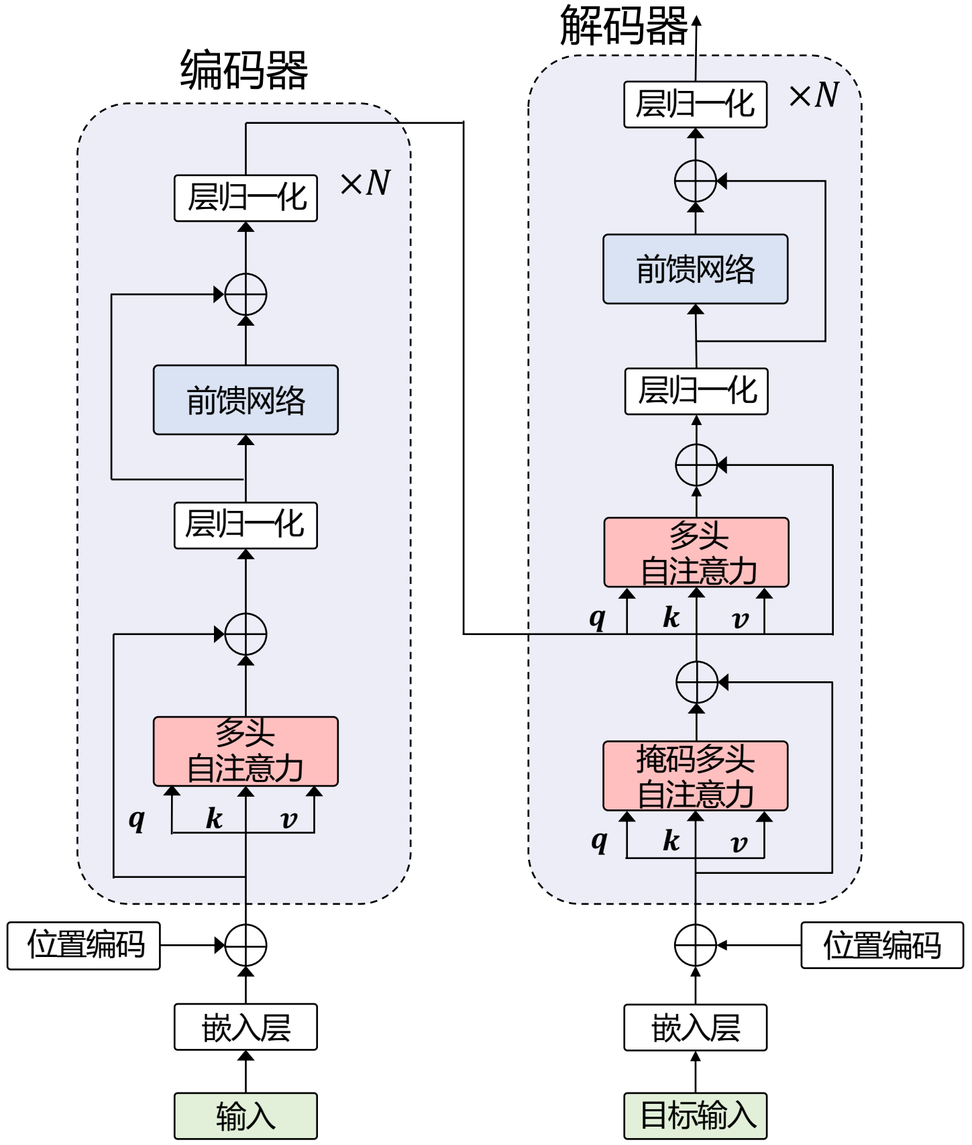


LSTM （Long Short-Term Memory，长短期记忆）



GRU（Gated Recurrent Unit，门循环单元）

5.1 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。

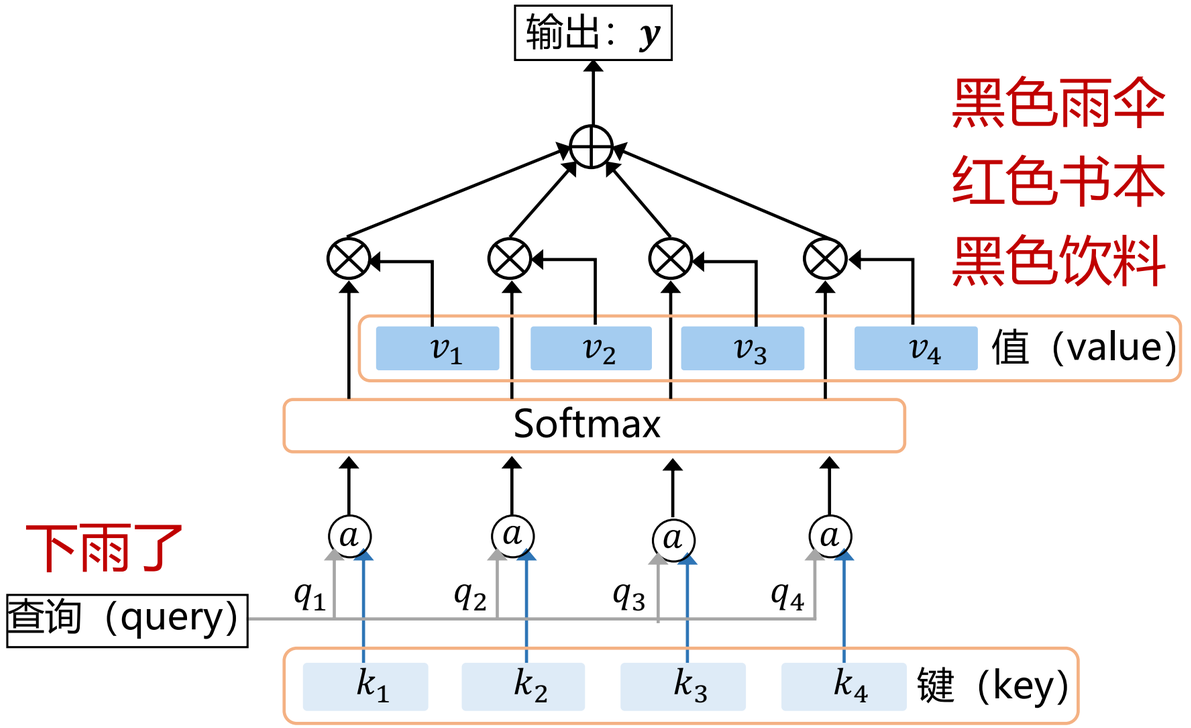
5.1 Transformer核心组成

能遮雨的黑色雨伞

可阅读的红色书本

可解渴的黑色饮料

下雨了

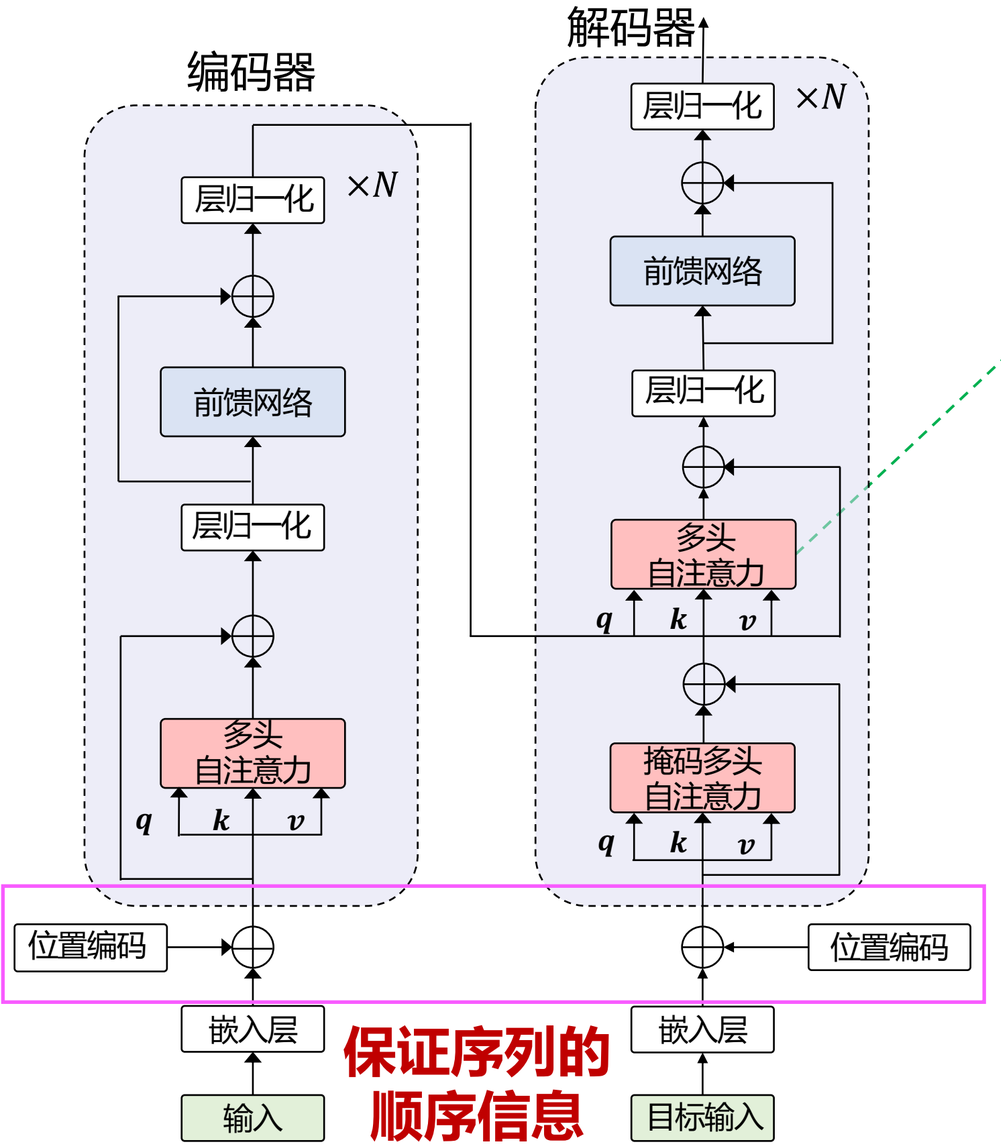


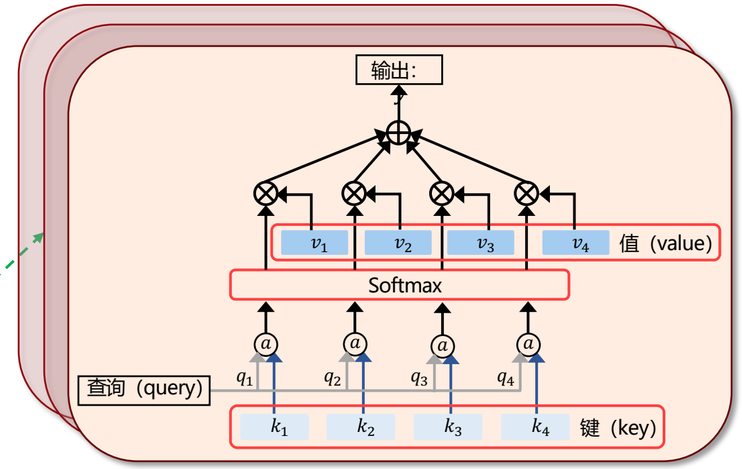
能遮雨　可阅读　可解渴

$$a(\boldsymbol{q},\boldsymbol{k}) = \frac{\boldsymbol{q}^{\mathrm{T}}\boldsymbol{k}}{\sqrt{d}} \qquad (1)$$

$$Attn(\boldsymbol{q},\boldsymbol{k},\boldsymbol{v}) = Softmax(a(\boldsymbol{q},\boldsymbol{k})) \otimes \boldsymbol{v} \qquad (2)$$

5.1 Transformer核心组成





多头注意力

$$MHA(q,k,v) = Concat(Head_1, Head_2, \ldots, Head_H)$$

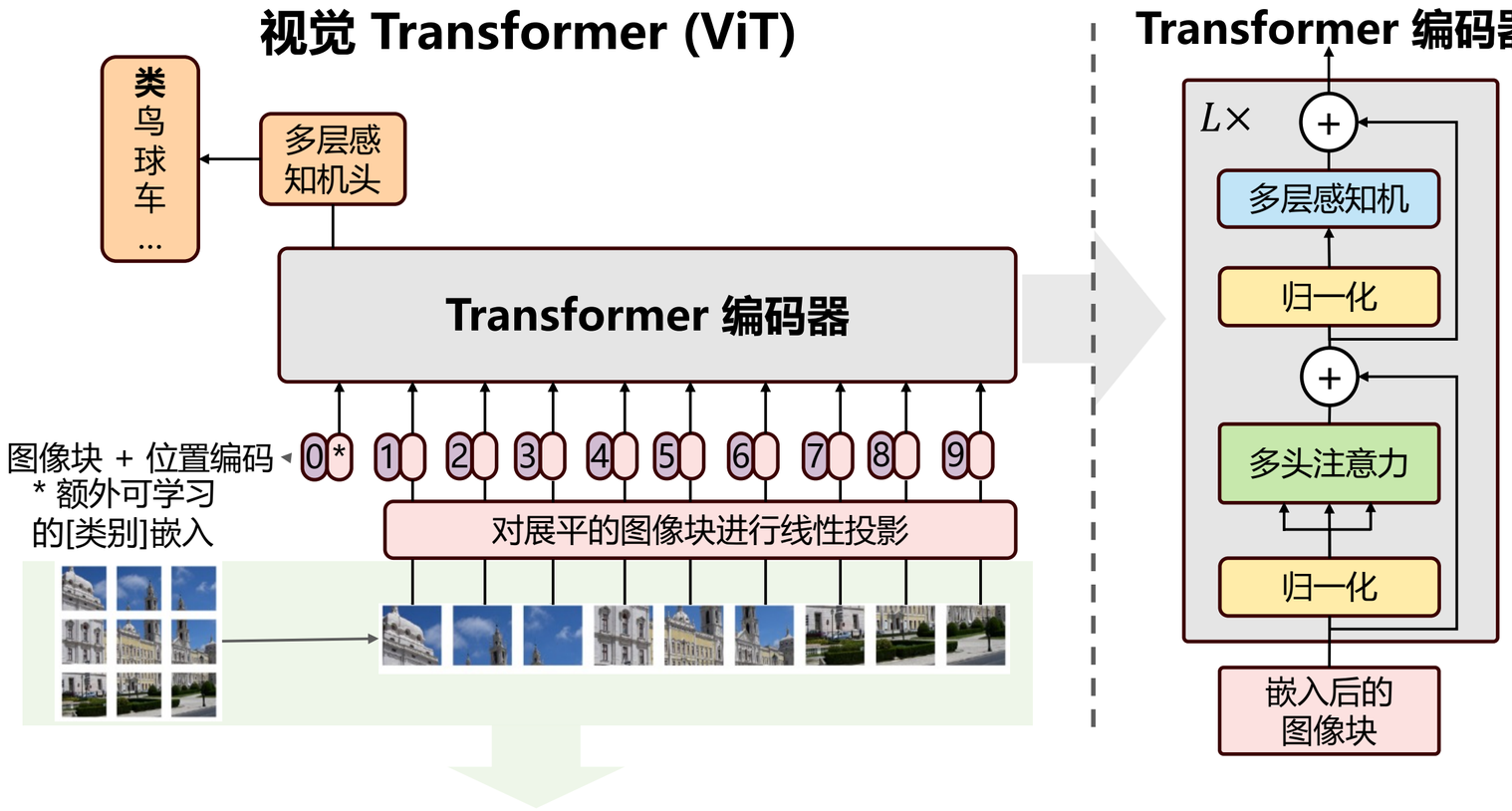
正弦余弦固定位置编码

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

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

5.2 从语言Transformer到视觉 Transformer

- 视觉Transformer（Vision Transformer）将NLP领域中广泛应用的Transformer架构引入到计算机视觉领域。



- ✓ 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）**。

5.2 从语言Transformer到视觉 Transformer

■ Swin Transformer通过其创新的**分层金字塔结构、窗口字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。

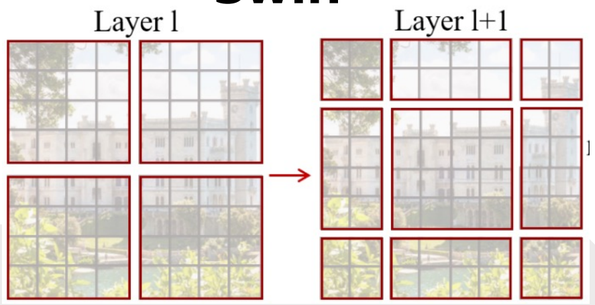
- ViT将图像分割成固定大小的非重叠小块，直接进行全局自注意力计算。在处理高分辨率图像时，计算复杂度会随着图像尺寸的增加而显著增加。
- ViT的全局自注意力机制，不同窗口之间的信息无法直接交互，这可能导致信息隔离。

全局自注意力

ViT

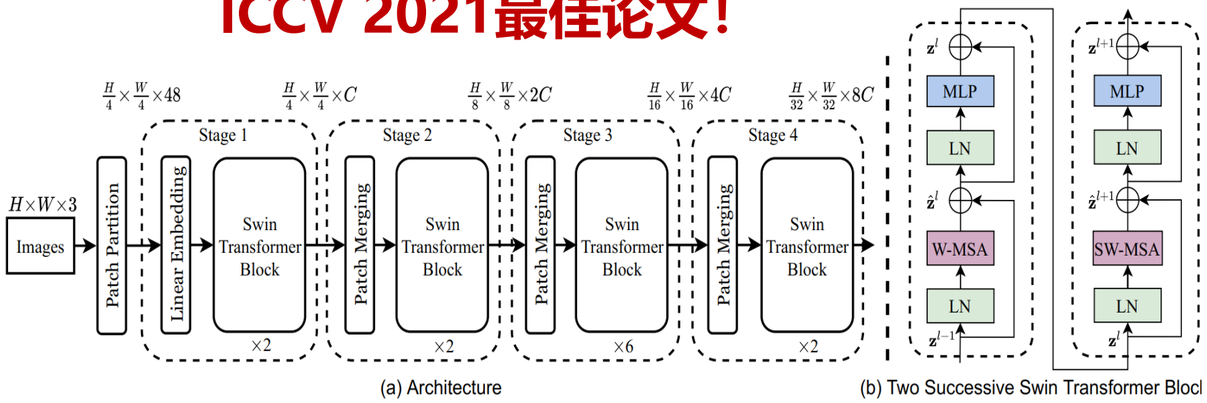
VS

Swin



窗口自注意力

ICCV 2021最佳论文！



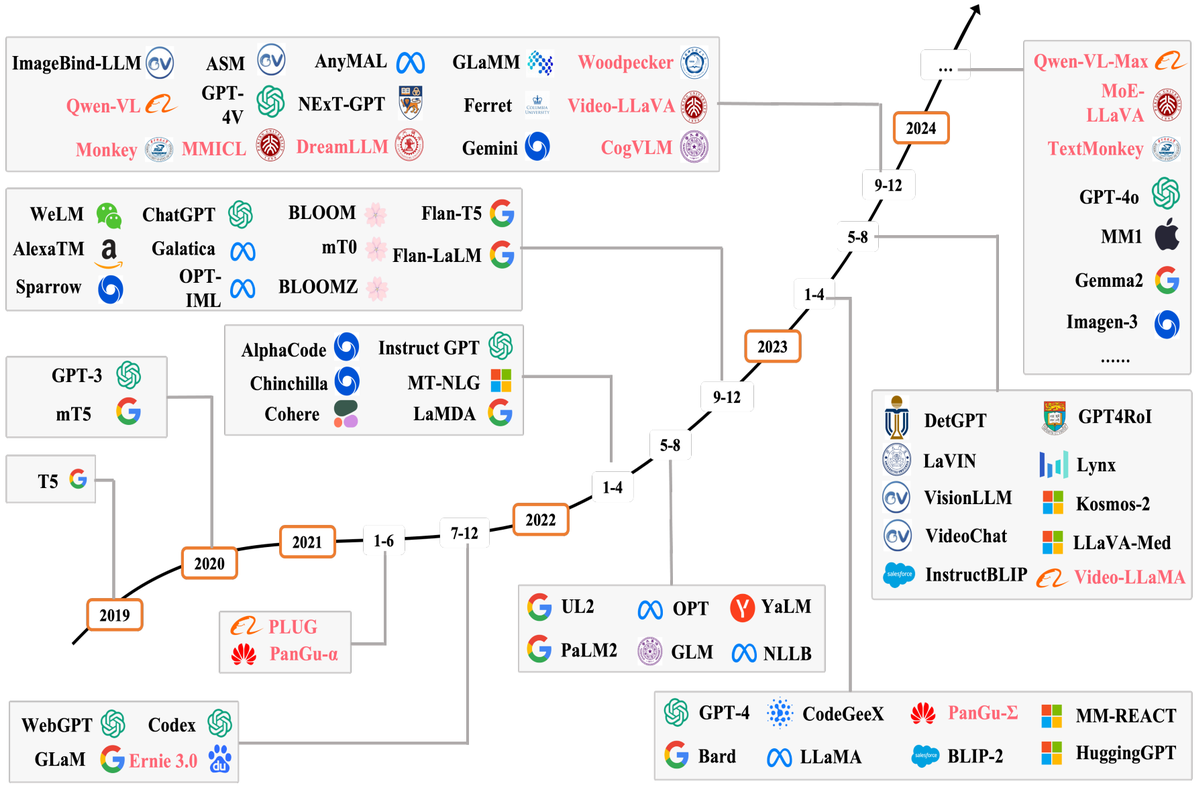
- 在每个窗口内进行自注意力计算。将计算复杂度降低到与窗口大小线性相关，显著减少了计算量。
- 移位窗口的设计使得相邻窗口之间能够进行信息交互。

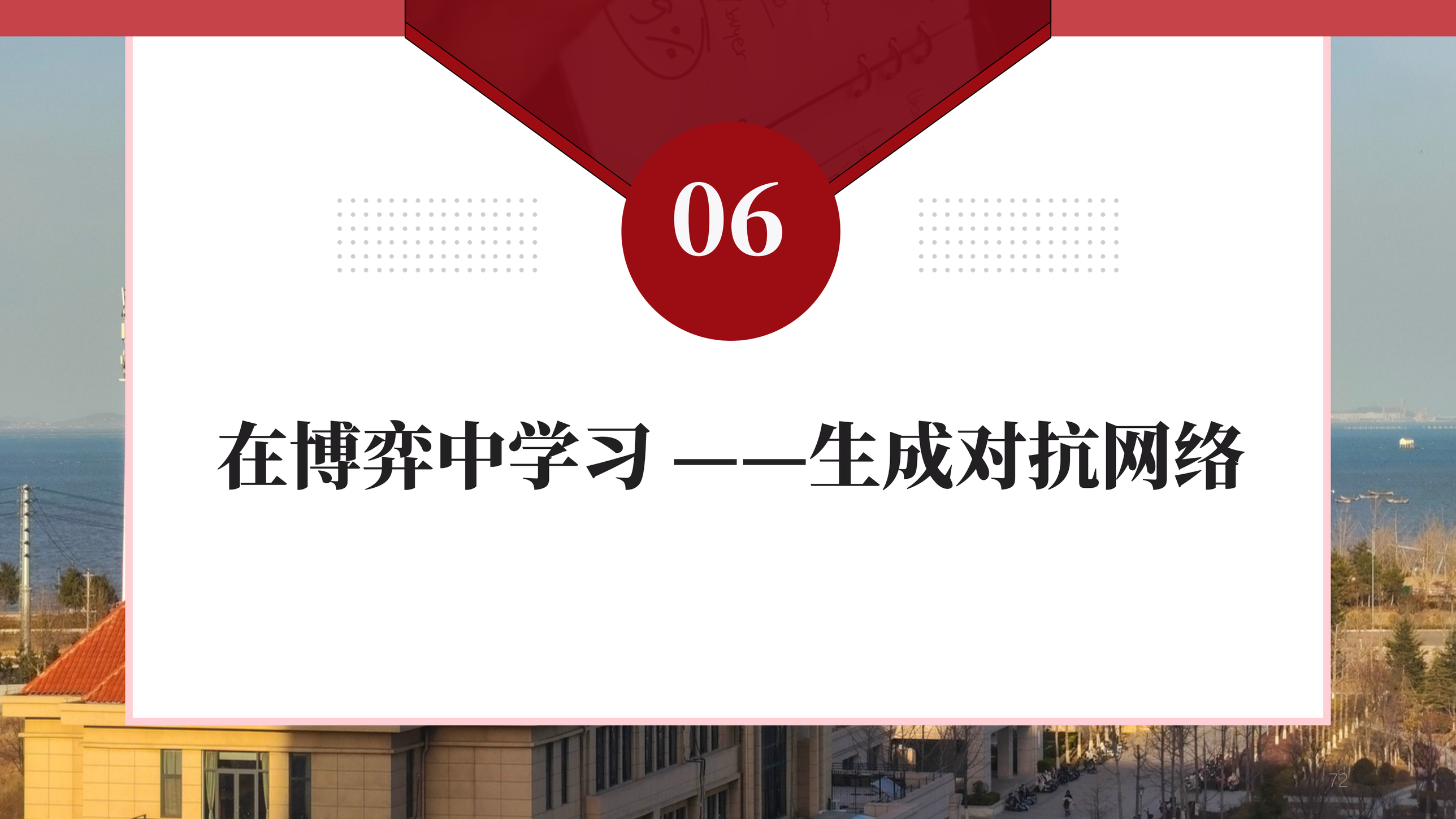
5.2 从语言Transformer到视觉 Transformer

RNN
CNN
VS
Transformer

五大优势

- ✅ 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

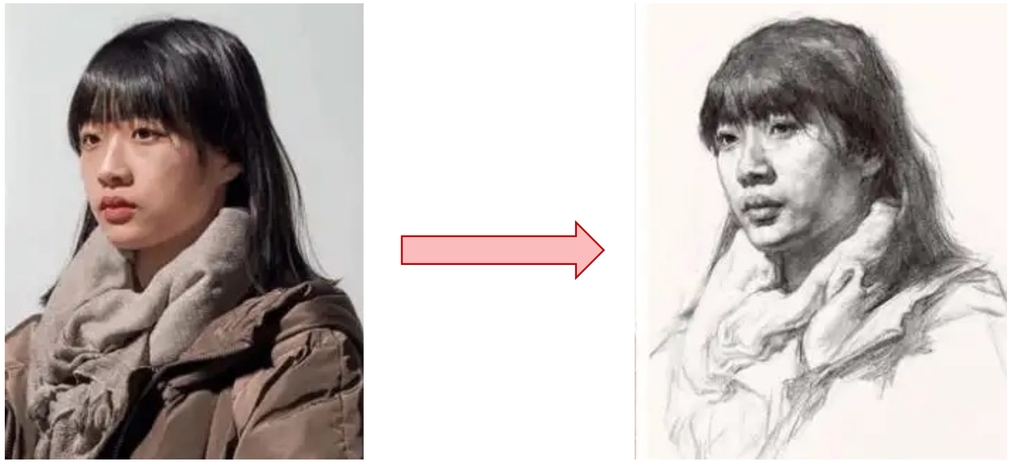


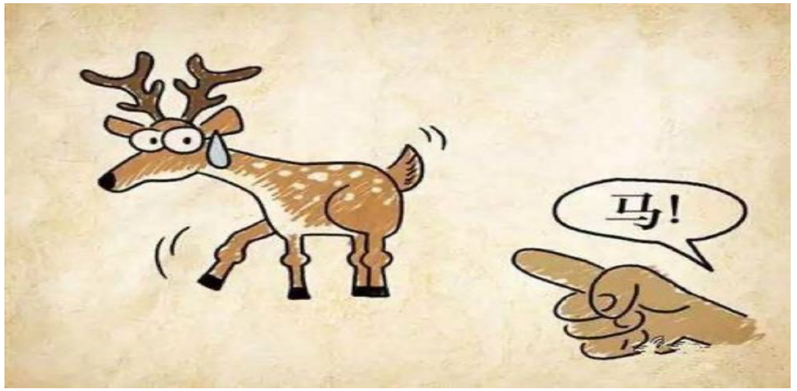


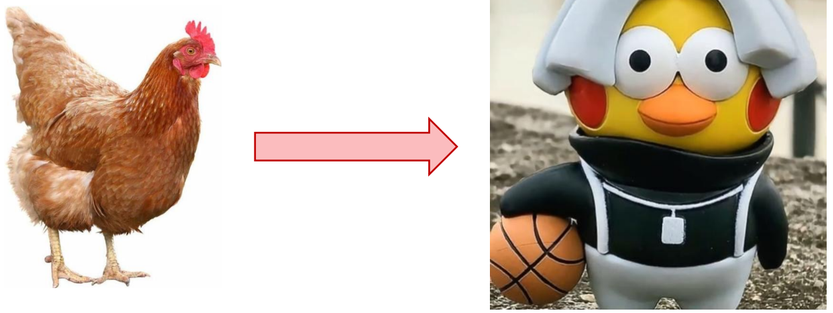
06

在博弈中学习 ——生成对抗网络

6.1 生成模型概述



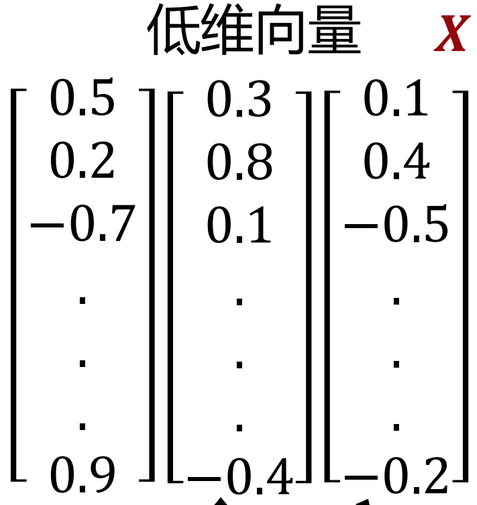


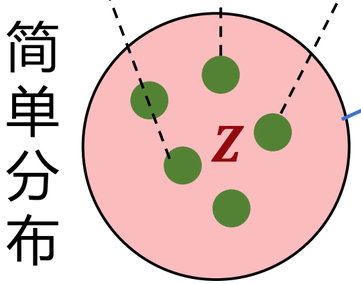


6.1 生成模型概述

早期生成模型主要**基于概率统计方法**，并常通过**引入隐变量**来建模数据的生成过程。

随着深度学习的兴起，生成模型开始**结合神经网络**，显著提升了生成能力和灵活性。





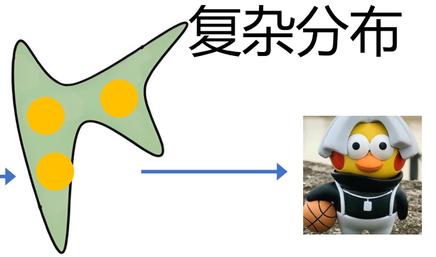
隐马尔可夫模型
(Hidden Markov Model, HMM)

建模时间序列数据的生成模型（语音信号和自然语言处理）。

高斯混合模型
(Gaussian Mixture Model, GMM)

基于高斯分布的生成模型，它能够捕捉数据的多模态特性。（聚类）

生成模型



篮球和鸡

变分自编码器
(Variational Auto-encoder，VAE)

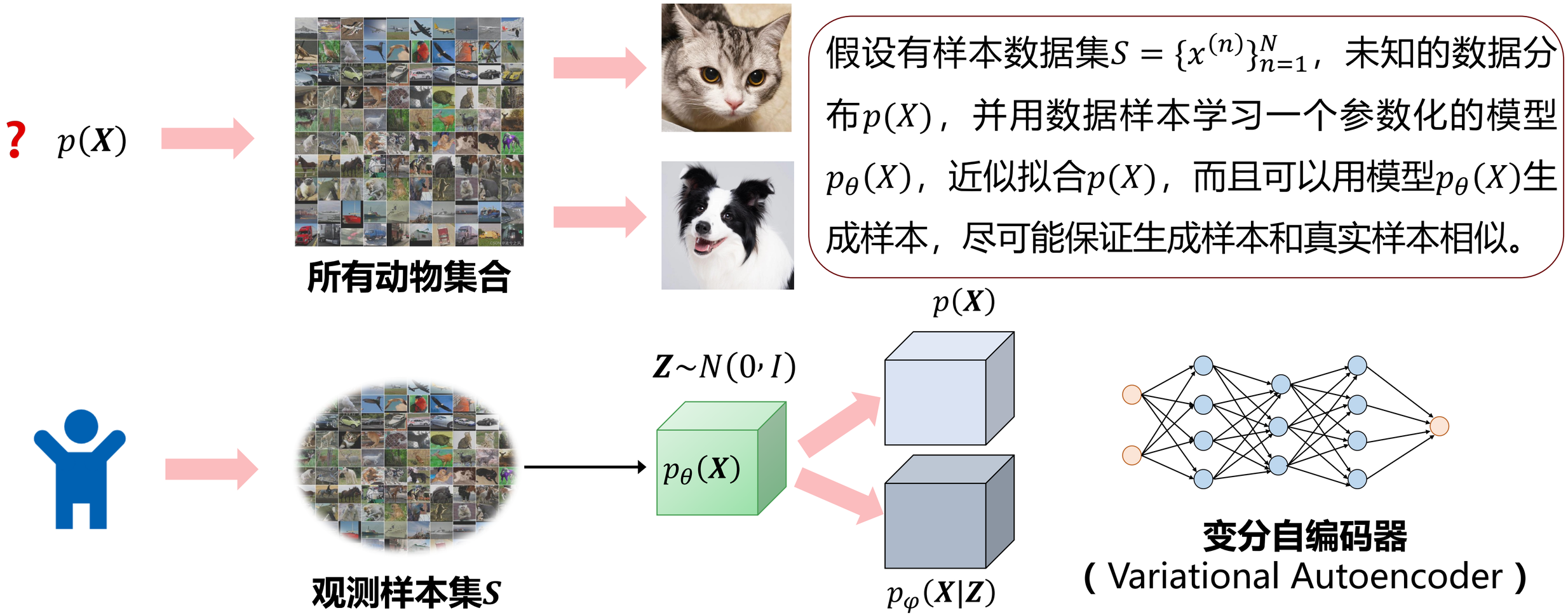
基于变分推断的生成模型，生成的样本质量相对稳定（图像、文本生成）。

生成对抗网络
(Generative Adversarial Network，GAN)

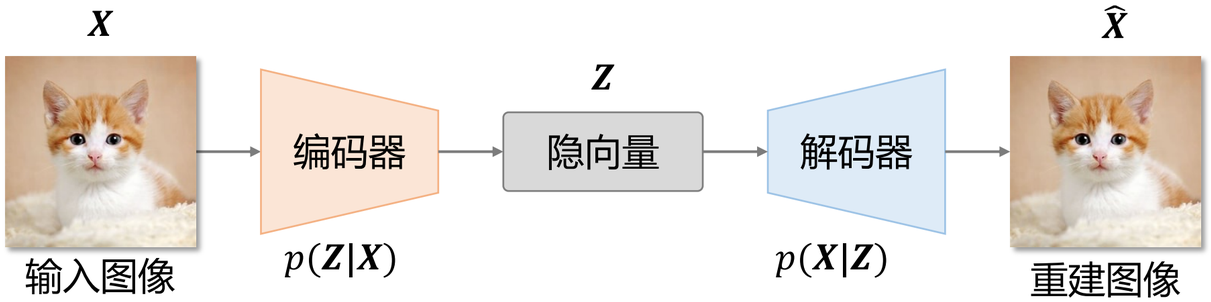
通过生成器与判别器的对抗训练实现数据生成，生成模型的里程碑式框架。

6.1 生成模型概述

■ 概率生成模型（Probabilistic Generative Models）是一类基于概率论的模型，它试图通过拟合数据的概率分布来生成新样本。



6.1 生成模型概述

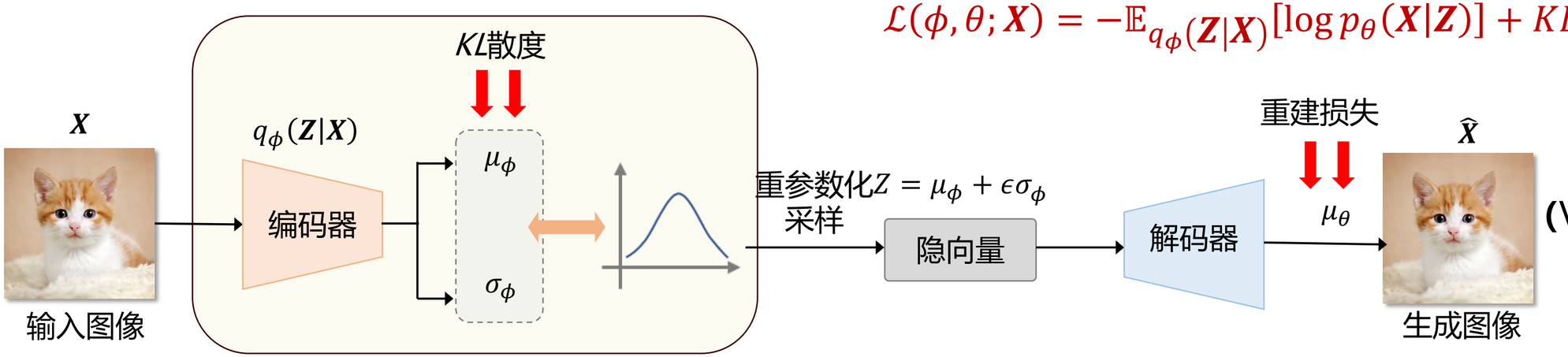


自编码器（Autoencoder）是一种无监督学习技术，通常用于学习数据得高效编码。本质上是对图像信息进行压缩，消除冗余信息，得到不丢失原始信息的低维表达。

编码器：负责将输入数据X转换成一个紧凑的表示Z（通常称为编码向量或隐向量）

解码器：从Z中重建原始输入数据

$$\mathcal{L}(\phi,\theta;\boldsymbol{X}) = -\mathbb{E}_{q_\phi(\boldsymbol{Z}|\boldsymbol{X})}[\log p_\theta(\boldsymbol{X}|\boldsymbol{Z})] + KL\left(q_\phi\left|(\boldsymbol{Z}|\boldsymbol{X})\right|q(\boldsymbol{Z})\right)$$



变分自编码器

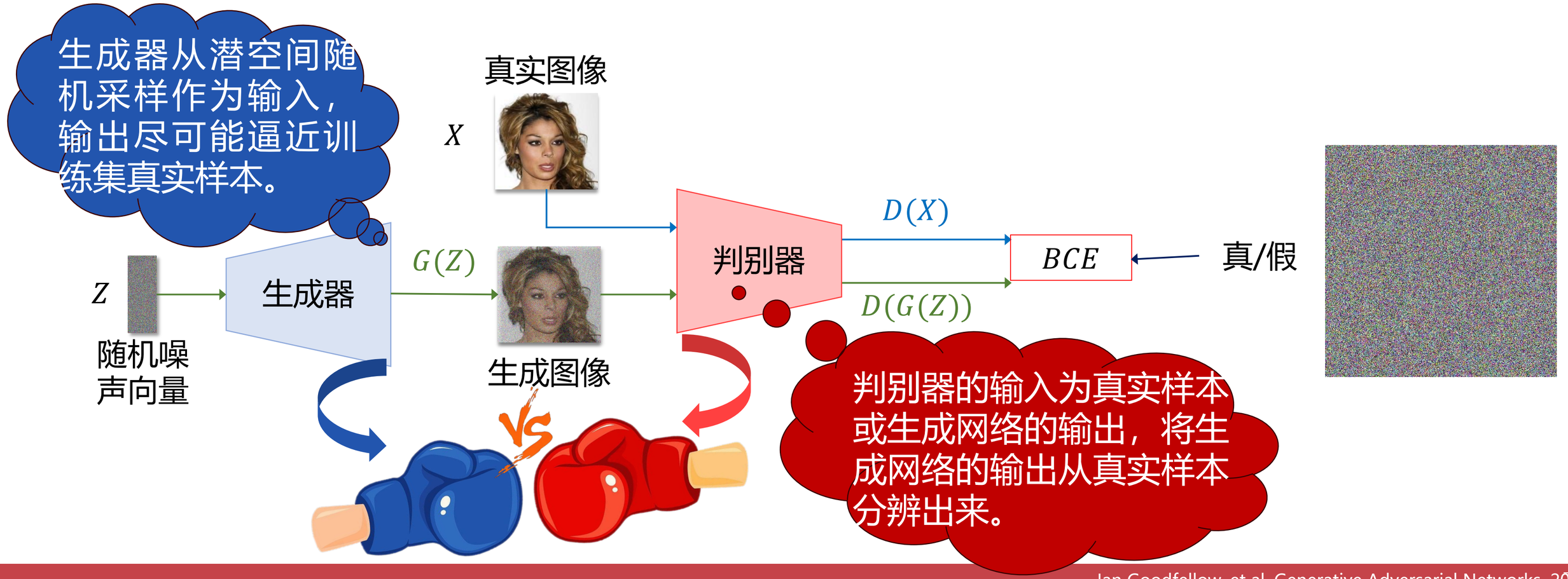
(Variational Autoencoder)

概率生成模型

VAE 对编码进行概率建模 ⇒ **VAE 泛化与生成能力强，能从潜在分布采样生成新样本**

6.2 博弈启发的生成对抗网络

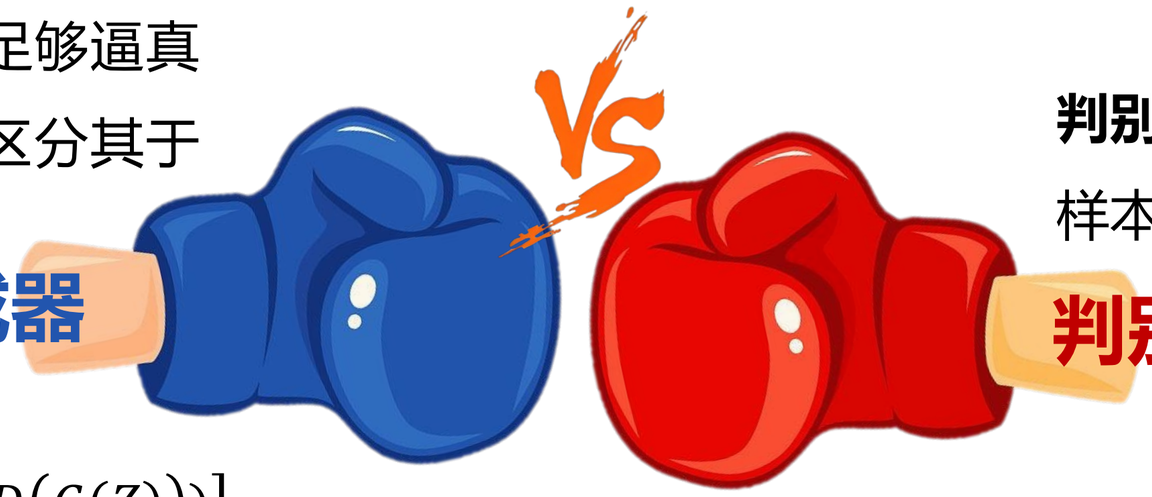
生成对抗网络（Generative Adversarial Networks, GAN）启发自博弈论中的二人零和博弈，包含一个生成模型和一个判别模型，前者负责捕捉样本数据的分布，而后者一般情况下是一个二分类器，判别输入是真实数据还是生成的样本。



Ian Goodfellow, et al. Generative Adversarial Networks. 2014.

6.2 博弈启发的生成对抗网络

生成器训练目标是使生成足够逼真的样本，使得判别器难于区分其于真实训练的数据。



判别器训练目标是最大化对真实样本的识别概率和最小化对生成样本的识别概率。

生成损失：

$$L_G = -E_{z\sim p_z(Z)}\left[\log\left(D\left(G(Z)\right)\right)\right]$$

判别损失：

$$L_D = -E_{X\sim P_{data}(X)}\left[\log\left(D(X)\right)\right] - E_{zp_z(Z)}\left[\log\left(1 - D\left(G(Z)\right)\right)\right]$$

GAN的训练过程通常分为以下几个步骤：

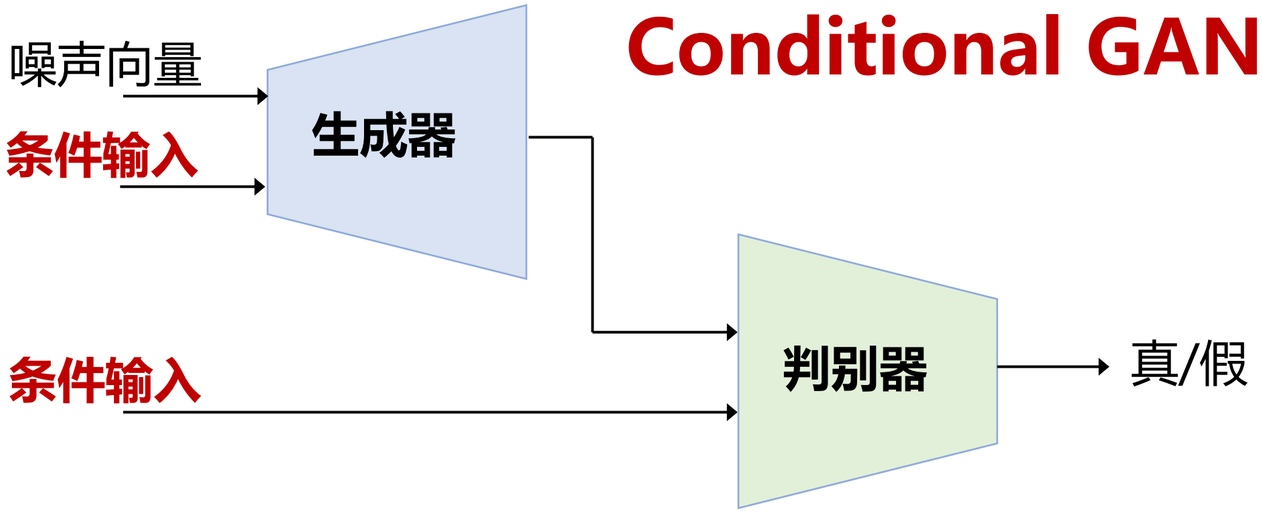
（1）初始化生成器G和判别器D

（2）固定生成器G，训练判别器D

（3）固定判别器D，训练生成器G

纳什均衡
(Nash Equilibrium)

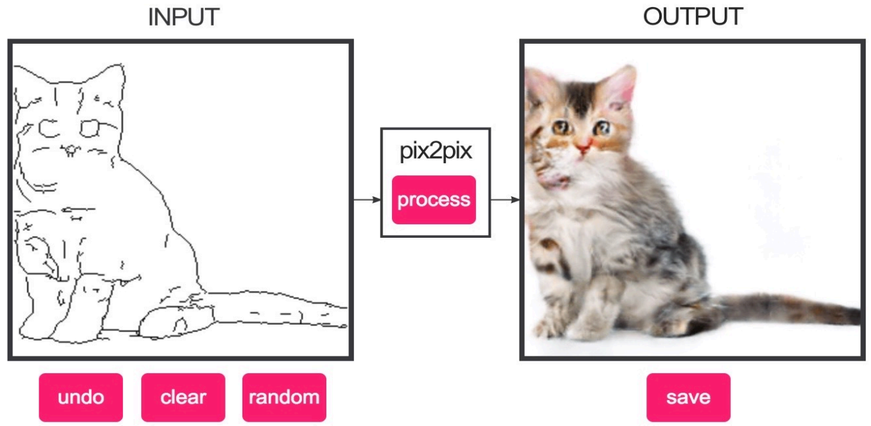
6.2 博弈启发的生成对抗网络



Conditional GAN

- 创新性地提出了条件对抗网络来学习**输入图像到输出图像的映射**。
- **允许指定生成样本的特定属性**，提高了生成任务的针对性和灵活性。
- 条件变量可以是不同模态数据，如文本描述，实现了**跨模态的联合学习**。

提供了一种通过条件信息控制生成过程的方法!

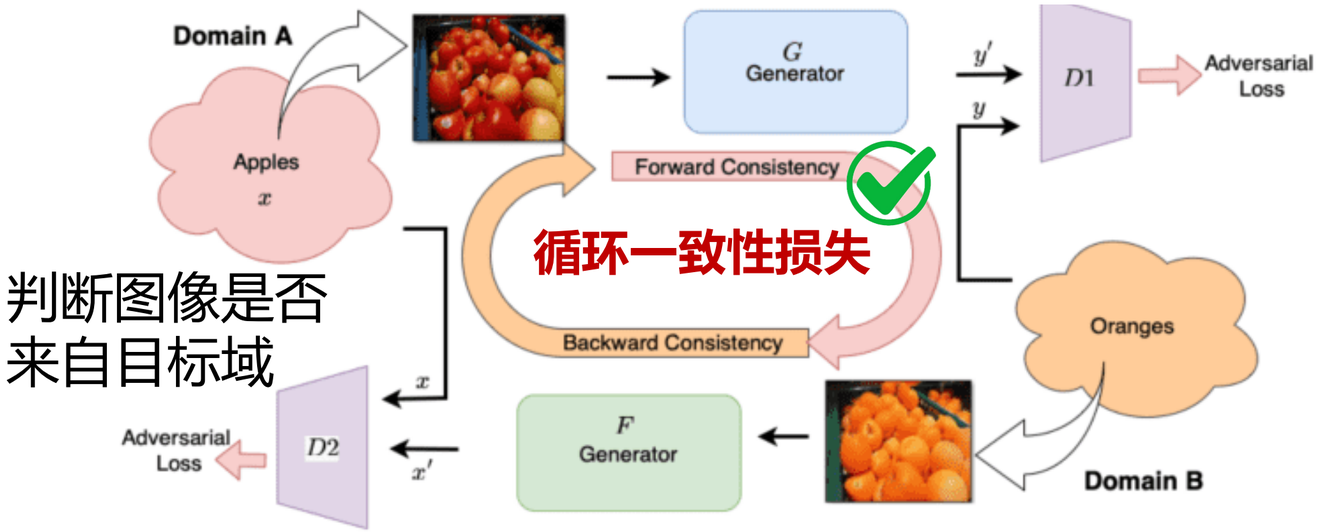


Mehdi Mirza, Simon Osindero. Conditional Generative Adversarial Nets, 2014.

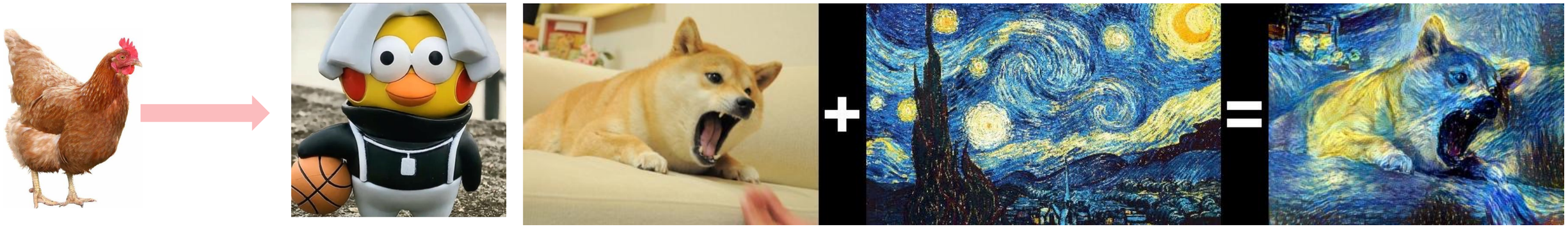
6.2 博弈启发的生成对抗网络

- Conditional GAN依赖于成对数据和条件信息，训练不稳定且生成样本多样性不足。
- 为解决这些问题，CycleGAN通过引入**循环一致性损失**确保翻译的准确性和稳定性，**无需成对数据**便可以将图像从一种风格或域转换到另一种风格或域。

CycleGAN 生成从源域到目标域的映射

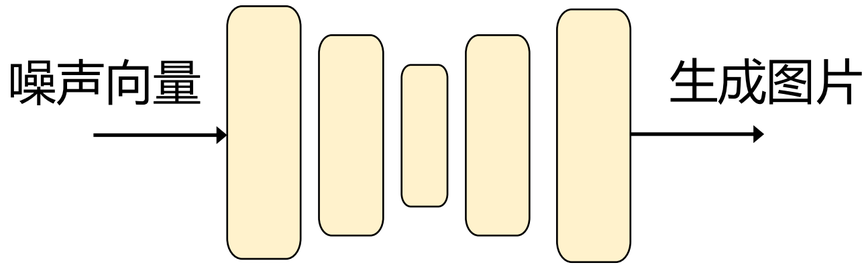


提供了一种无监督图像到图像翻译的方法!

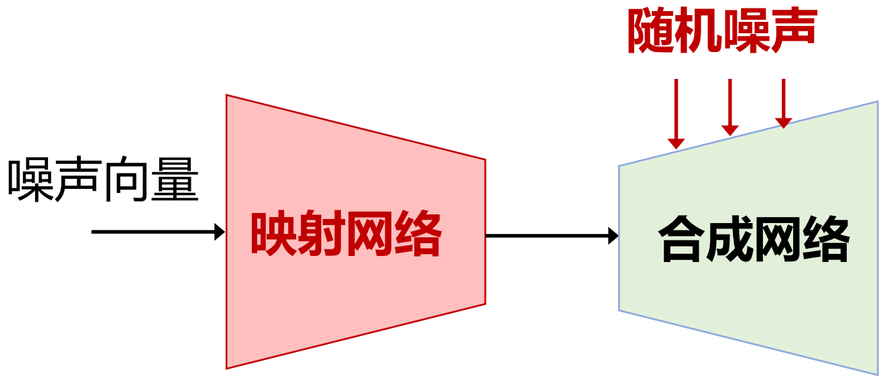


Jun-Yan Zhu, et al. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks, 2017.

6.2 博弈启发的生成对抗网络



传统生成器



StyleGAN生成器

- 映射网络将**随机噪声映射到风格向量**。在生成器的每一层引入风格信息，实现了对生成图像的多尺度风格控制。
- 合成网络采用**噪声注入机制**来增强生成图像的细节和随机性。
- StyleGAN采用**渐进式训练方法**，从较低分辨率开始训练，逐步增加分辨率，直到最终达到目标分辨率。

StyleGAN引入“风格”向量来控制生成图像的不同层次特征，实现对生成图像的精细控制。

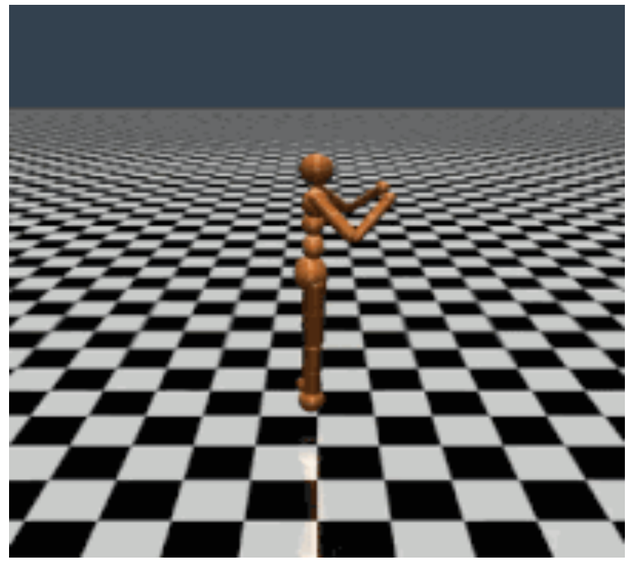
Tero Karras, et al. A Style-Based Generator Architecture for Generative Adversarial Networks, 2019.

6.2 博弈启发的生成对抗网络

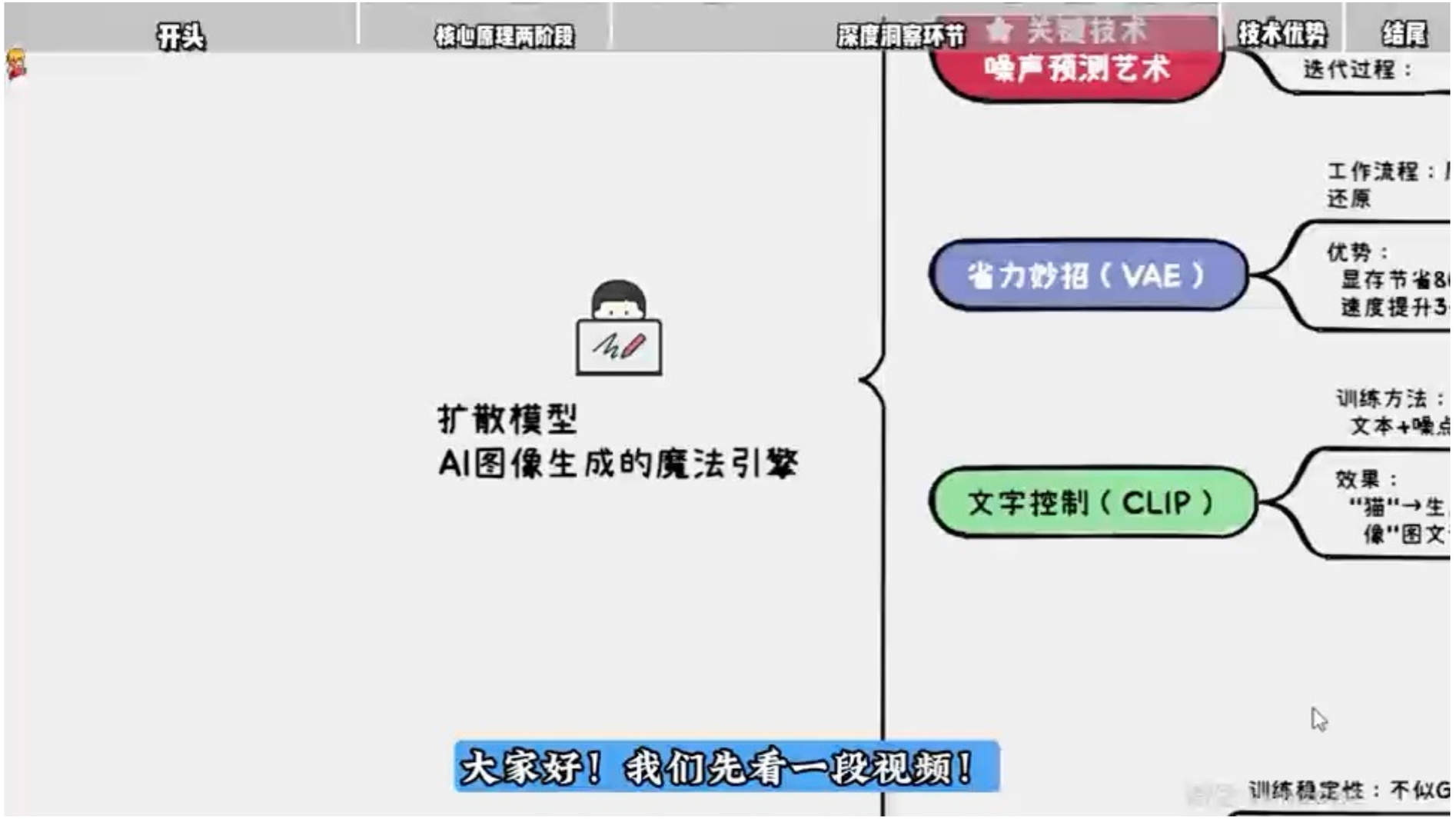








6.3 生成模型的新宠——Diffusion模型



山东大学 SHANDONG UNIVERSITY

第4讲
深度学习