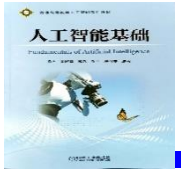
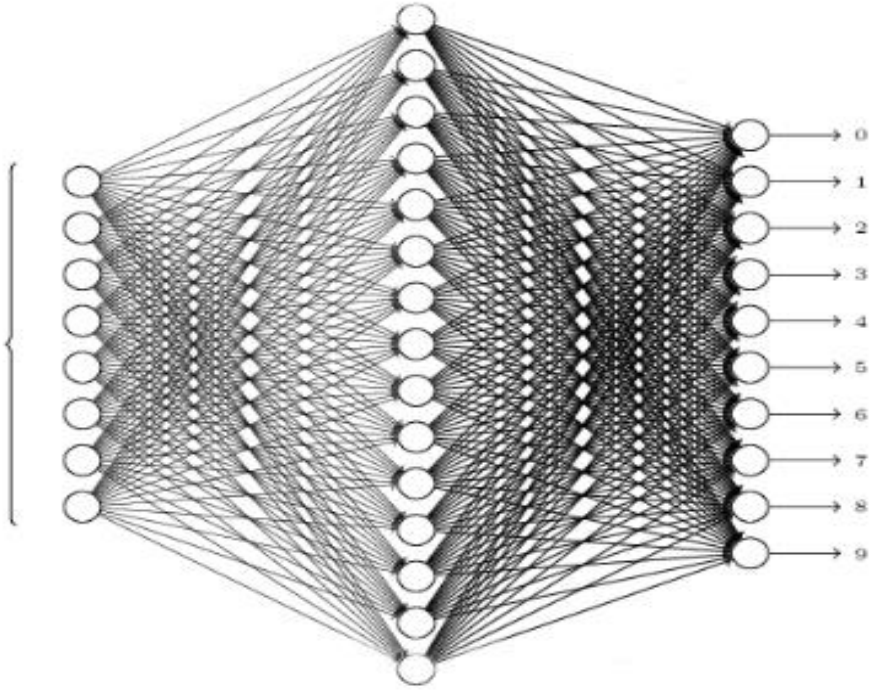
第7章　深度学习



全连接神经网络

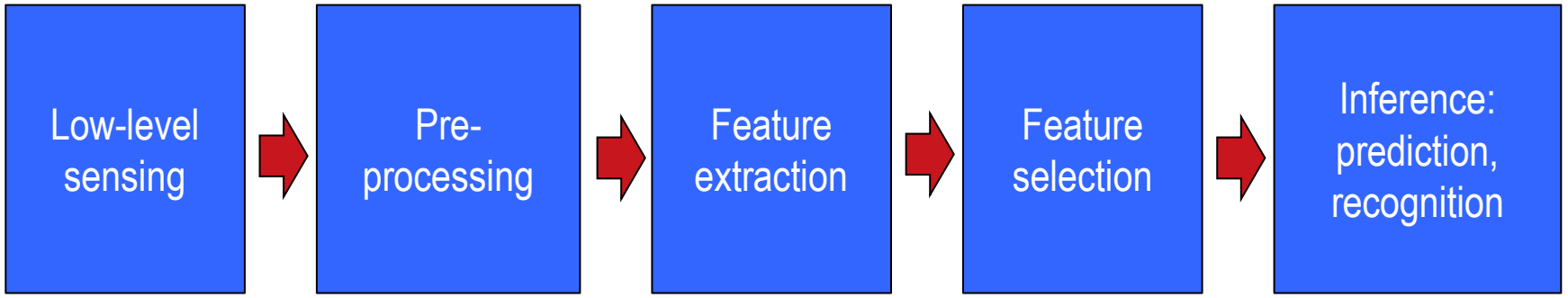
- 每相邻两层之间的每个神经元之间都是相连的，连接权重就是需要训练的参数
- 当输入层的特征维度很高时，全连接网络需要训练的参数就会增加很多，计算速度就会变得很慢的



深度学习概述

- 卷积神经网络
- 循环神经网络
- 长短期记忆网络

深度学习概述



- 良好的特征表达，对最终算法的准确性起了非常关键的作用
- 识别系统主要的计算和测试工作耗时主要集中在特征提取部分
- 特征的样式目前一般都是人工设计的，靠人工提取特征。

深度学习概述

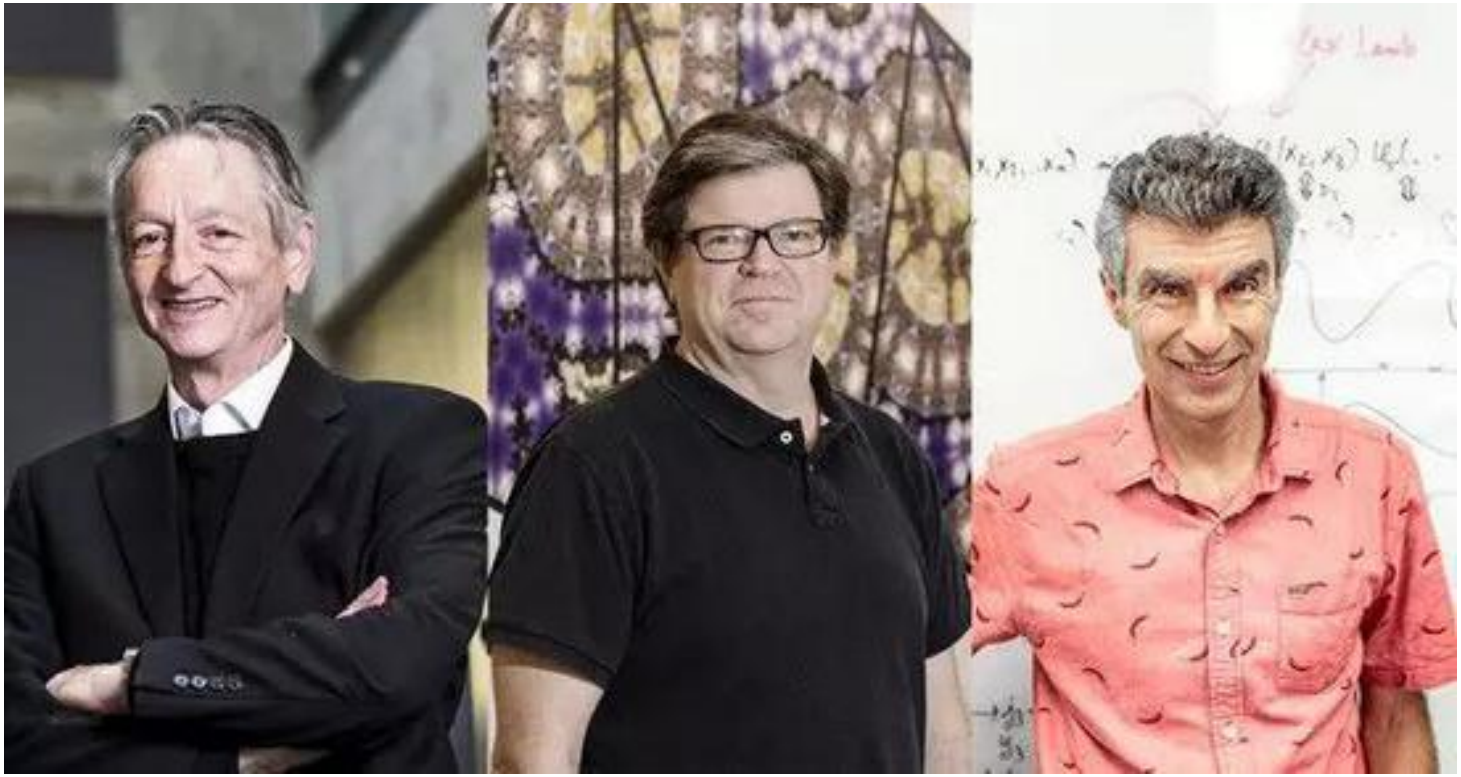
- 机器学习中，获得好的特征是识别成功的关键
- 目前存在大量人工设计的特征，不同研究对象特征不同，特征具有多样性，如：SIFT, HOG, LBP等
- 手工选取特征费时费力，需要启发式专业知识，很大程度上靠经验和运气
- 是否能自动地学习特征？

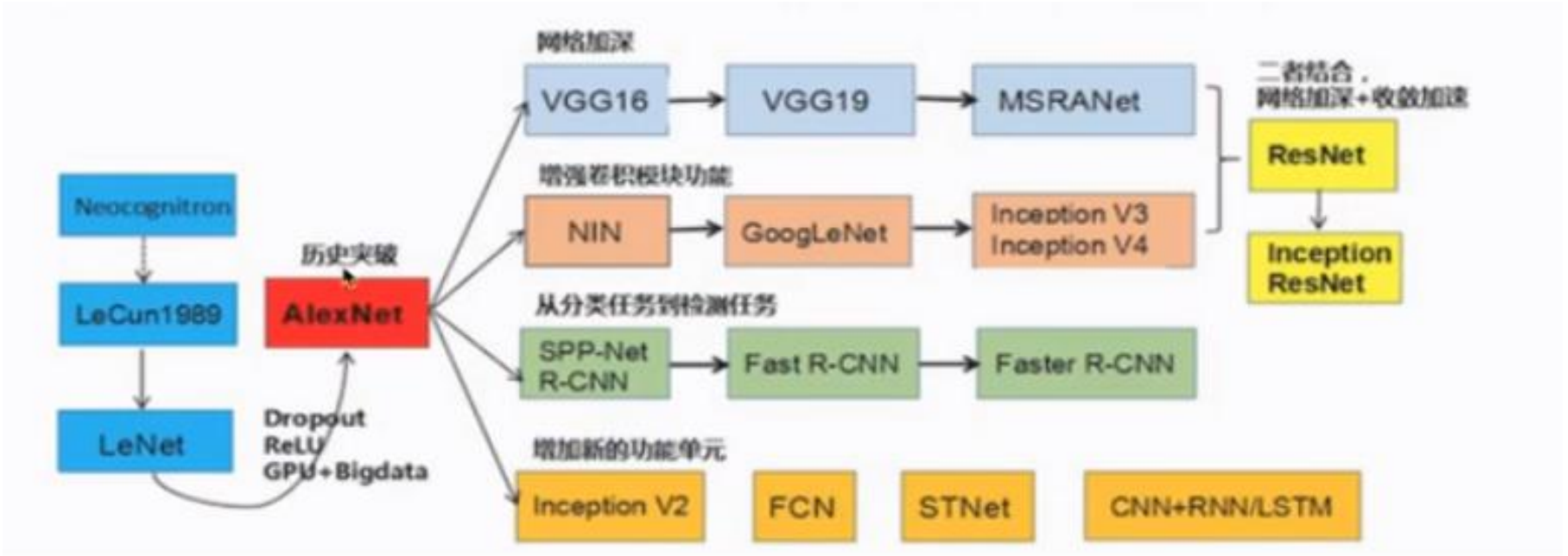
深度学习概述

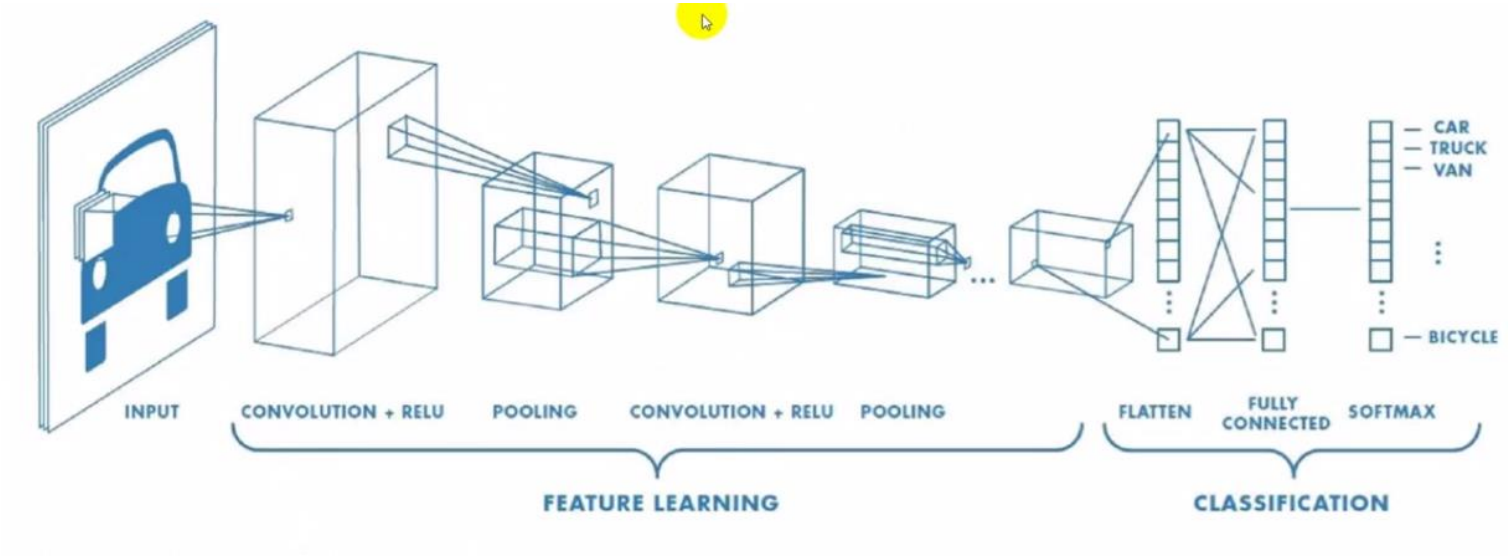
单层感知器 ▶ 多层感知器 ▶ 深度神经网络

深度学习概述

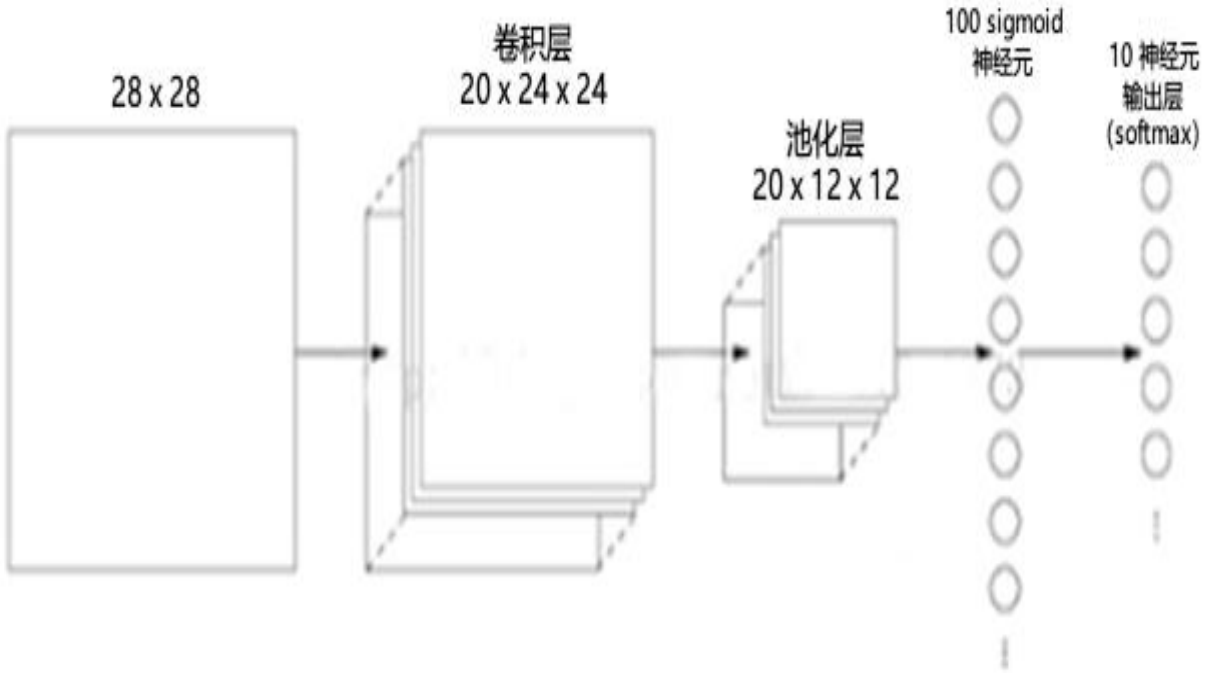
- 三位深度学习之父 Geoffrey Hinton， Yann LeCun，和Yoshua Bengio共同获得了2019年的图灵奖



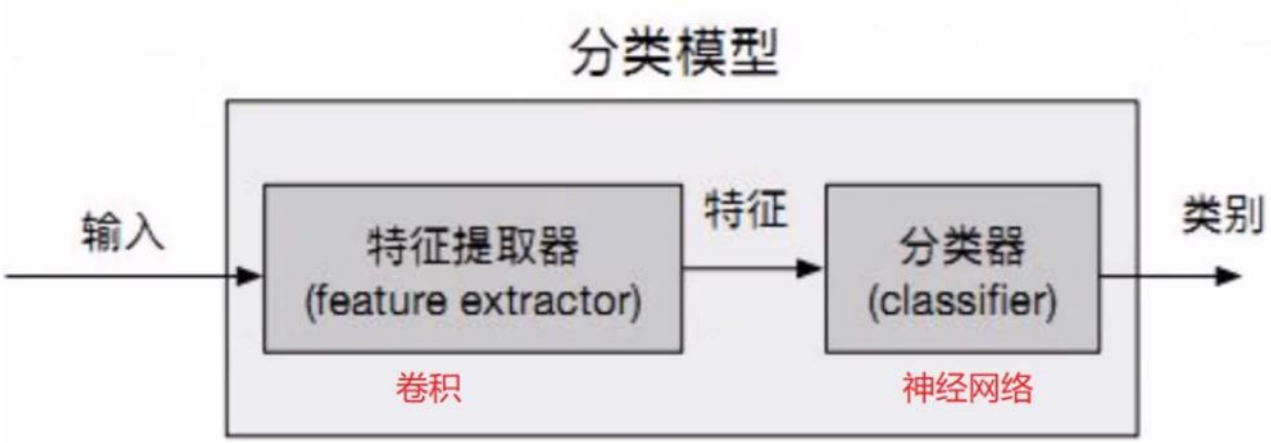

网络加深
VGG16
VGG19
MSRANet
二者结合，
网络加深+收敛加速
ResNet
Inception
ResNet
增强卷积模块功能
NIN
GoogLeNet
Inception V3
Inception V4
Neocognitron
LeCun1989
LeNet
历史突破
AlexNet
Dropout
ReLU
GPU+Bigdata
从分类任务到检测任务
SPP-Net
R-CNN
Fast R-CNN
Faster R-CNN
增加新的功能单元
Inception V2
FCN
STNet
CNN+RNN/LSTM


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

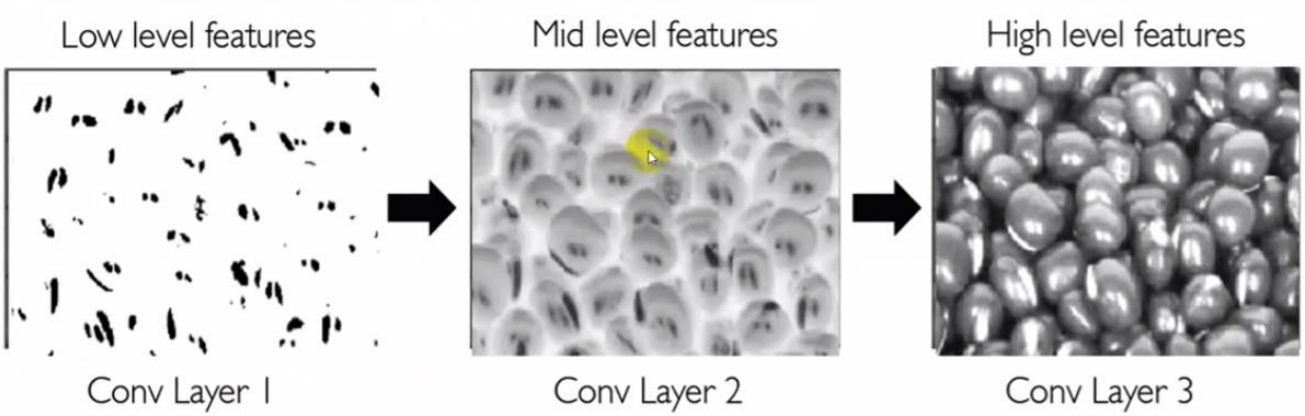
一、卷积神经网络


28 x 28
卷积层
20 x 24 x 24
池化层
20 x 12 x 12
100 sigmoid
神经元
10 神经元
输出层
(softmax)

一、卷积神经网络



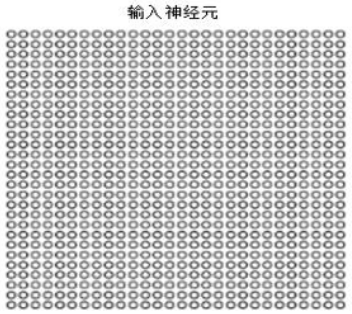
分类模型的结构



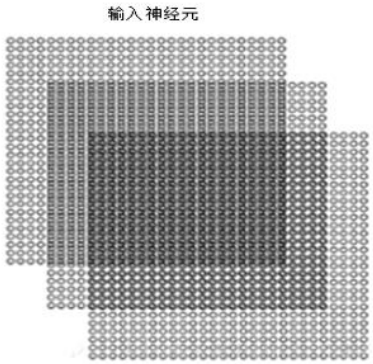
一、卷积神经网络

1、输入层

- CNN输入层的输入格式保留了图片本身的结构。。
  - ✓ 对于黑白的 28×28 的图片，CNN的输入是一个28×28的二维神经元。
  - ✓ 对于RGB格式的28×28图片，CNN的输入则是一个3×28×28的三维神经元（RGB中的每一个颜色通道都有一个 28×28 的矩阵）。



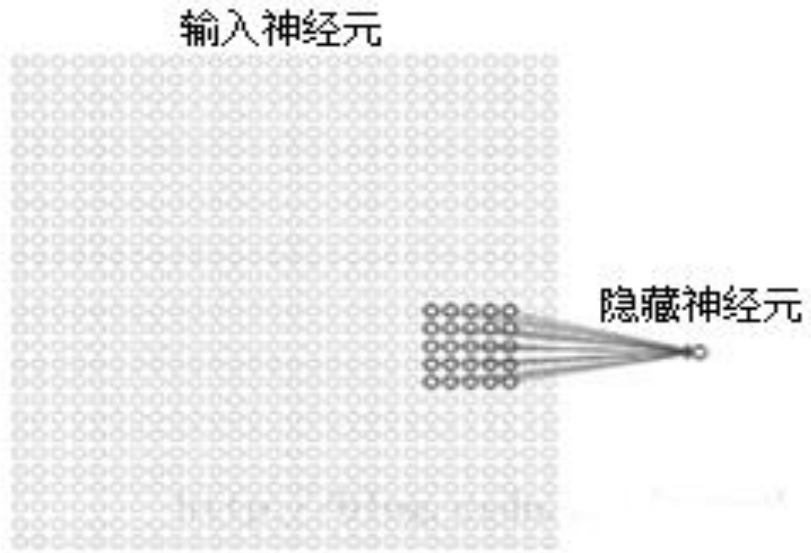
28×28的输入层

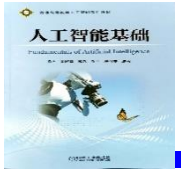


3×28×28的输入层

一、卷积神经网络

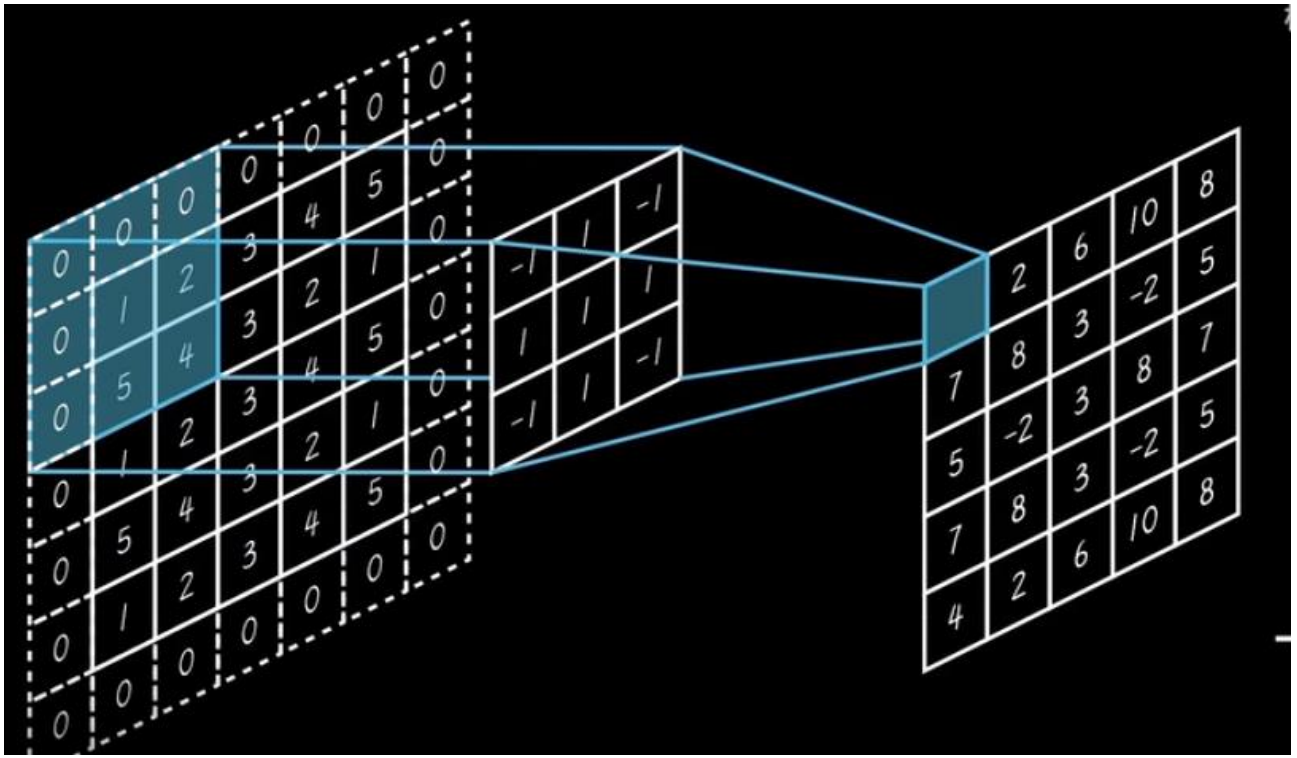
- ✓ 隐藏层的神经元与输入层的5×5个神经元相连，这个5×5的区域就称为**感受视野**。
- ✓ 隐藏层中的神经元具有一个固定大小的感受视野去感受上一层的部分特征
- ✓ 一个感受视野对应一个**卷积核**，感受视野中权重为w的矩阵称为卷积核，感受视野对输入的扫描间隔称为**步长**





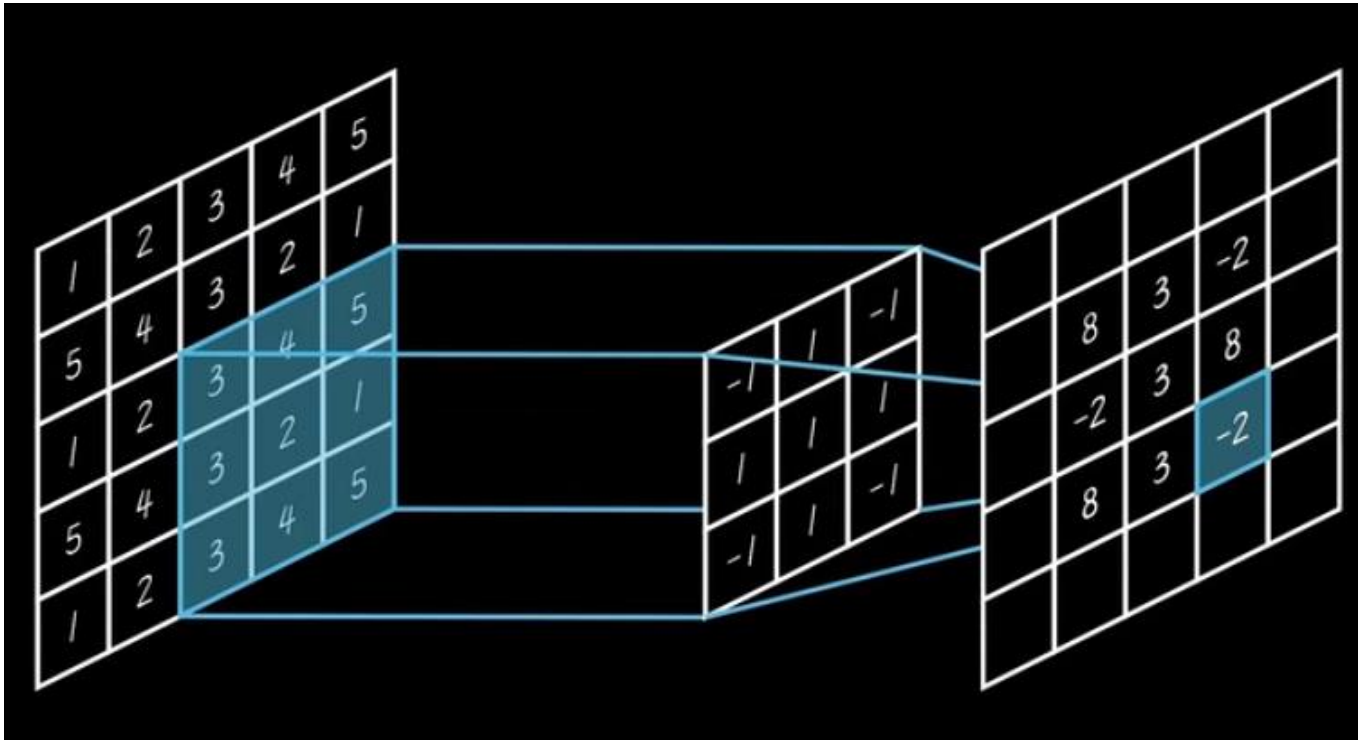
一、卷积神经网络

2、卷积层



- **特征图**：通过一个带有卷积核的感受视野扫描生成的下一层神经元矩阵称为一个特征图（feature map）
- **权值共享**：在同一个特征图上的神经元使用的卷积核是相同的，即共享卷积核中的权值和附带的偏移
- 一个特征图对应一个卷积核

一、卷积神经网络

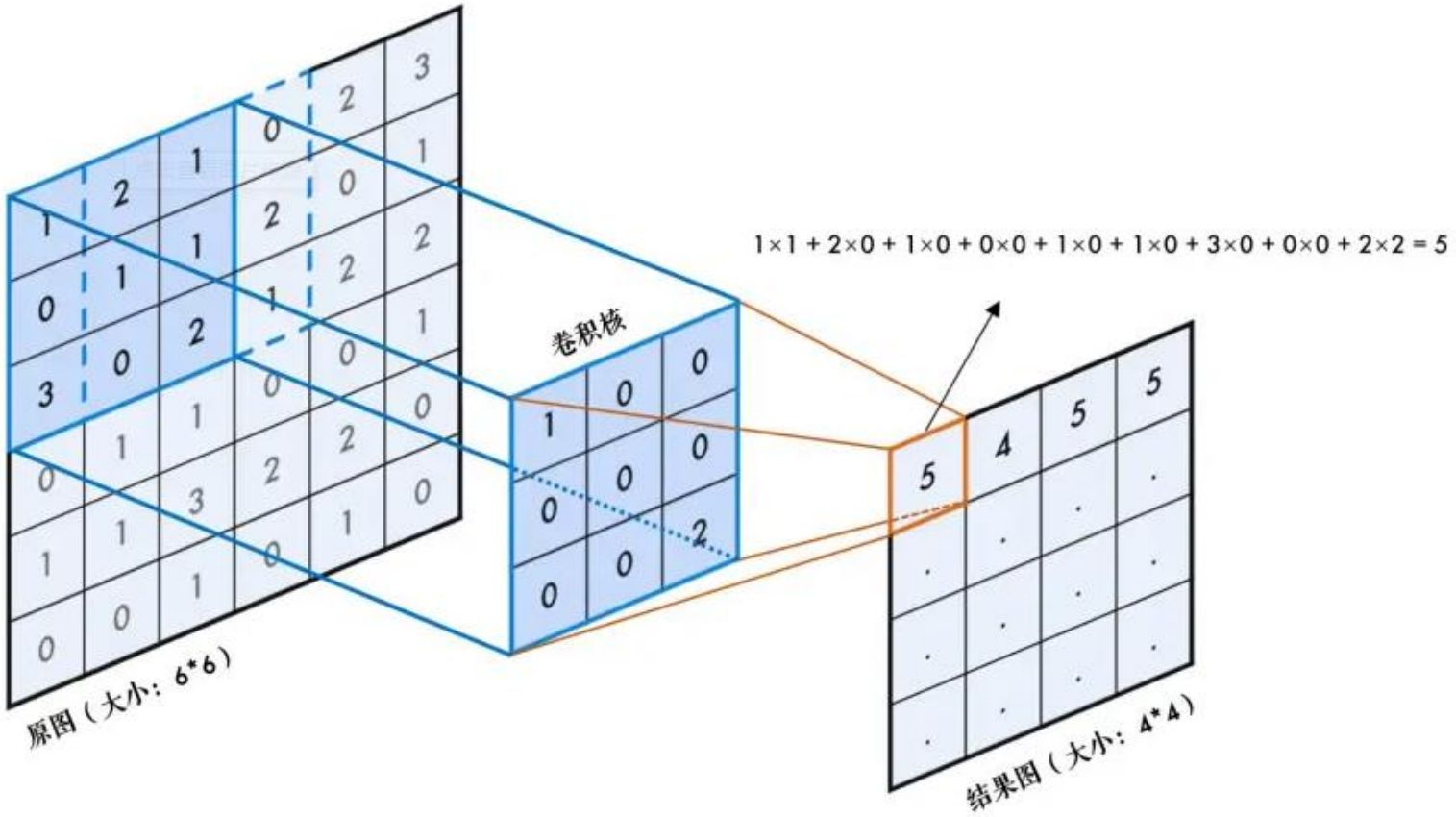


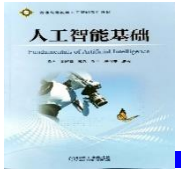
一、卷积神经网络

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

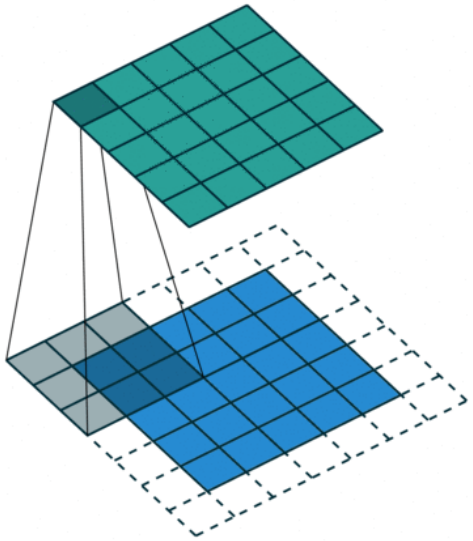
4		

一、卷积神经网络

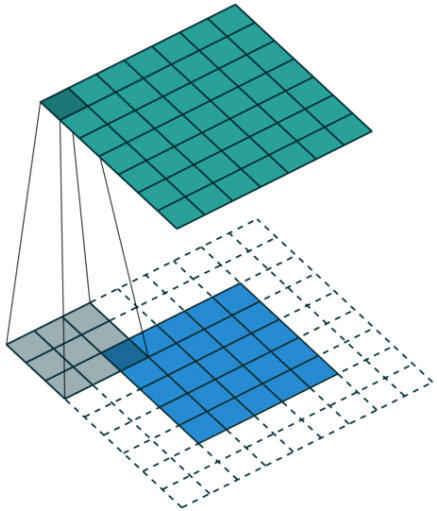




一、卷积神经网络

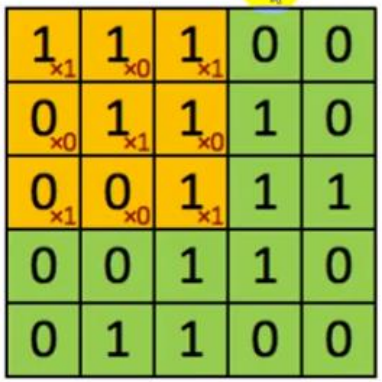


$$W_2 = (W_1 - F + 2P)/S + 1$$
$$H_2 = (H_1 - F + 2P)/S + 1$$

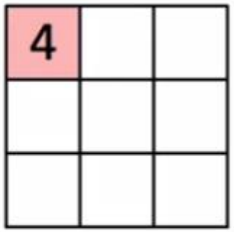


$$P = \frac{F-1}{2}$$

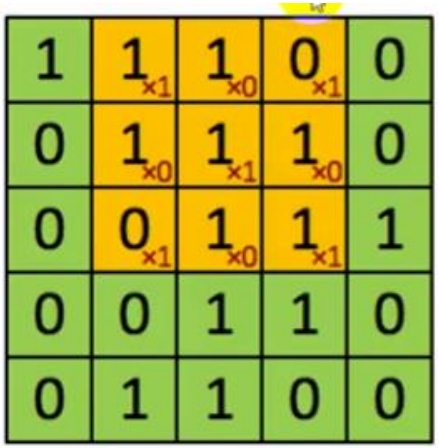
$1\times1+1\times0+1\times1+0\times0+1\times1+1\times0+0\times1+0\times0+1\times1=4$



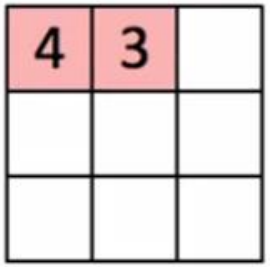
Image



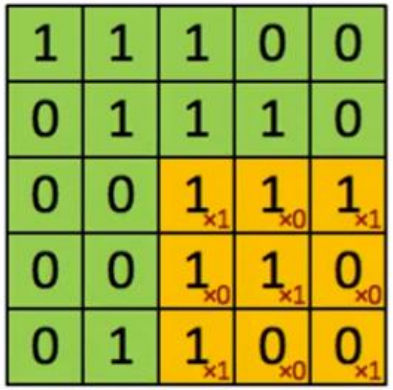
Convolved Feature



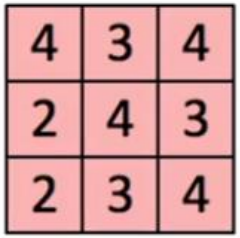
Image



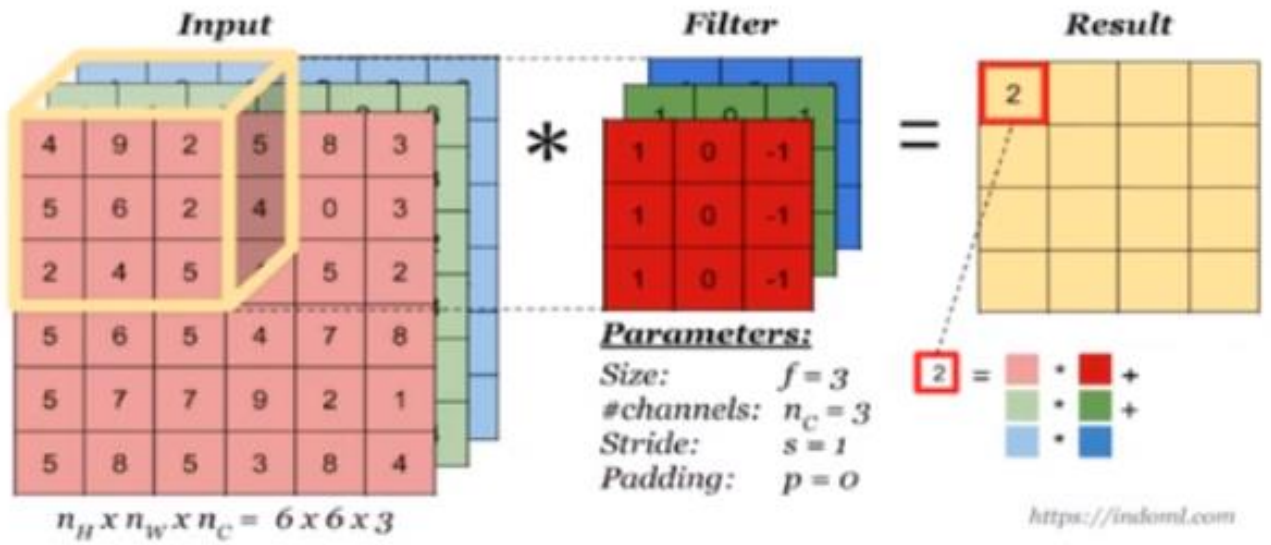
Convolved Feature



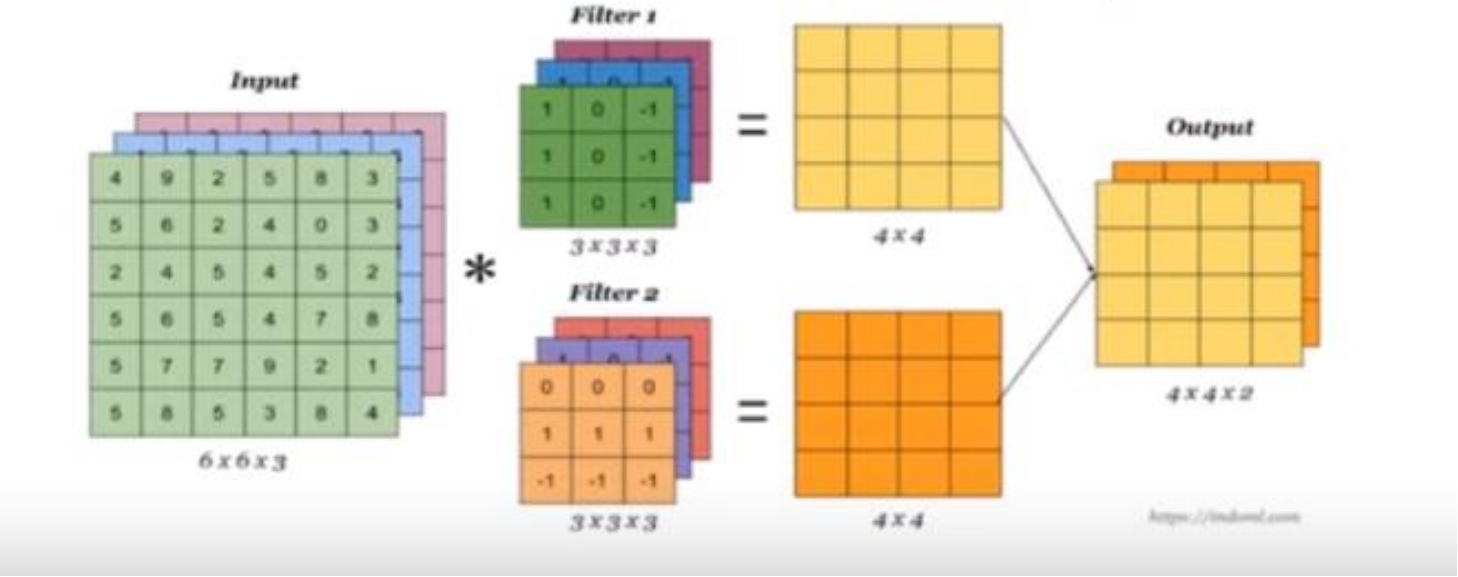
Image



Convolved Feature



当输入有多个通道（channel）时(例如图片可以有 RGB 三个通道)，卷积核需要拥有相同的channel数,每个卷积核 channel 与输入层的对应 channel 进行卷积，将每个 channel 的卷积结果按位相加得到最终的 Feature Map。

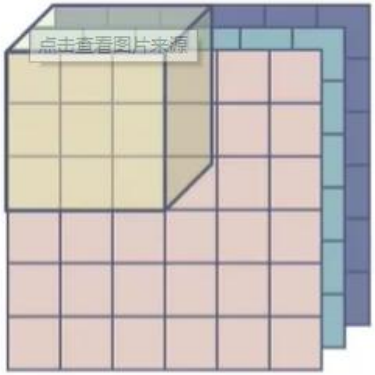


当有多个卷积核时，可以学习到多种不同的特征，对应产生包含多个 channel 的 Feature Map，例如上图有两个 filter，所以 output 有两个 channel。

- 假设我们有10 个Filter，每个Filter3 X 3 X 3（计算RGB图片），并且只有一层卷积，那么参数有多少?

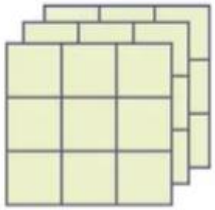
计算：每个Filter参数个数为：3 3 3 + 1 bias = 28个权重参数，总共28 * 10 = 280个参数，即使图片任意大小，我们这层的参数也就这么多。

一、卷积神经网络



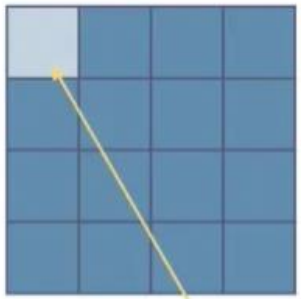
6x6x3

*



3x3x3

=



4x4

C = 0

0	0	0
0	2	2
0	1	2

*

-1	1	1
-1	1	-1
1	-1	1

C = 1

0	0	0
0	0	2
0	1	2

*

1	-1	-1
-1	0	-1
-1	0	1

C = 2

0	0	0
0	1	1
0	0	2

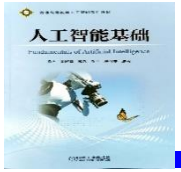
1	-1	-1
-1	-1	0
-1	1	1

=

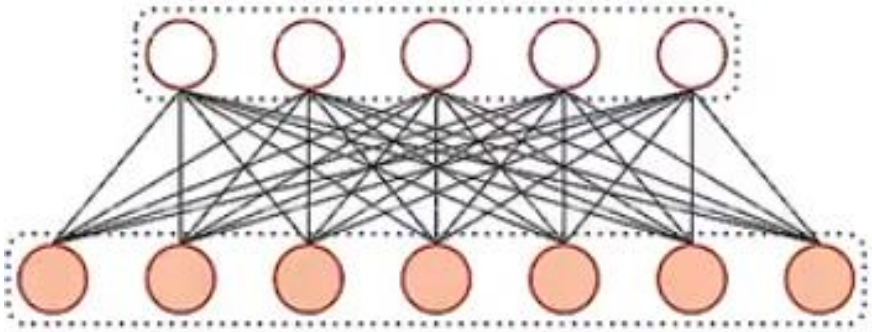
2-2-1+2
+
0-2+0+2
+
(-1)+0+0+2

=

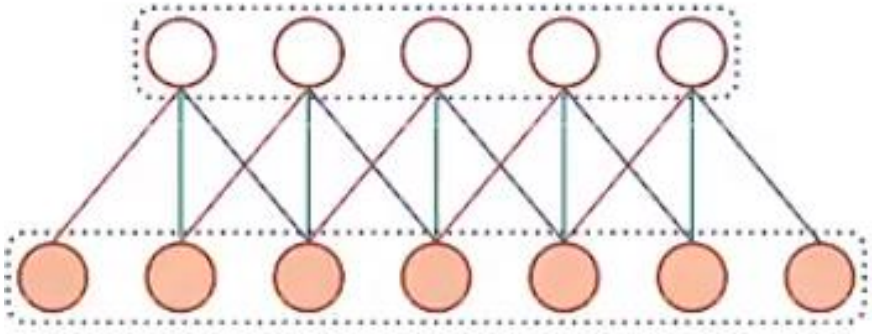
2



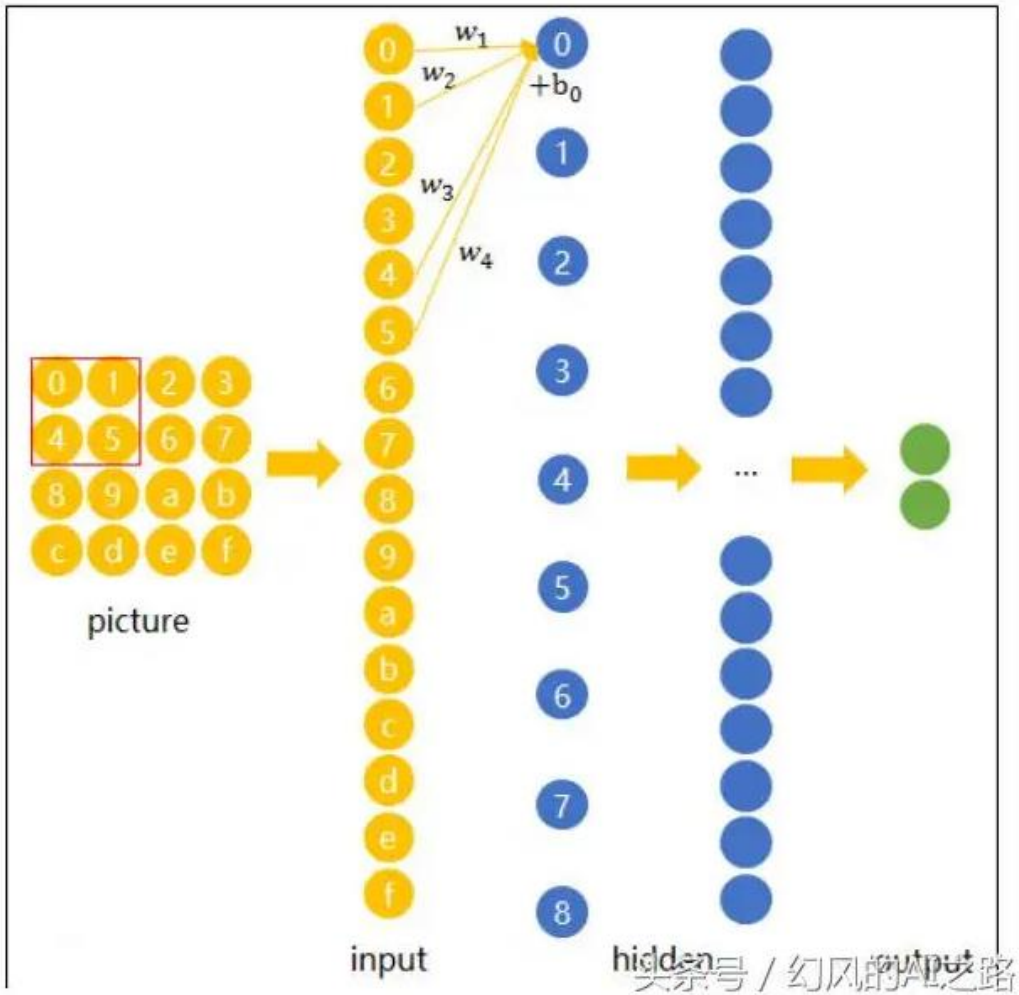
一、卷积神经网络



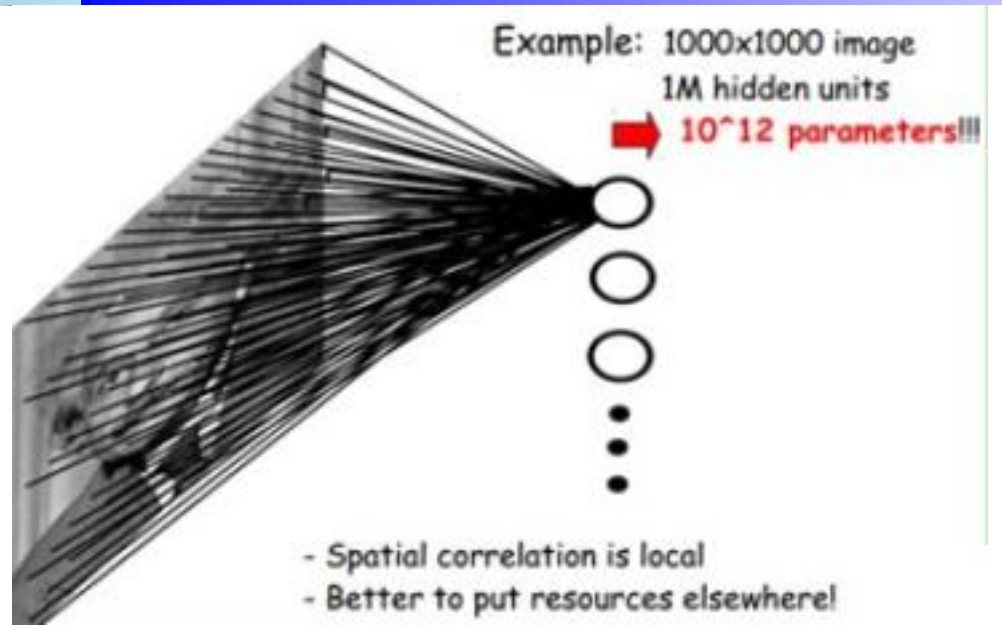
(a) 全连接层

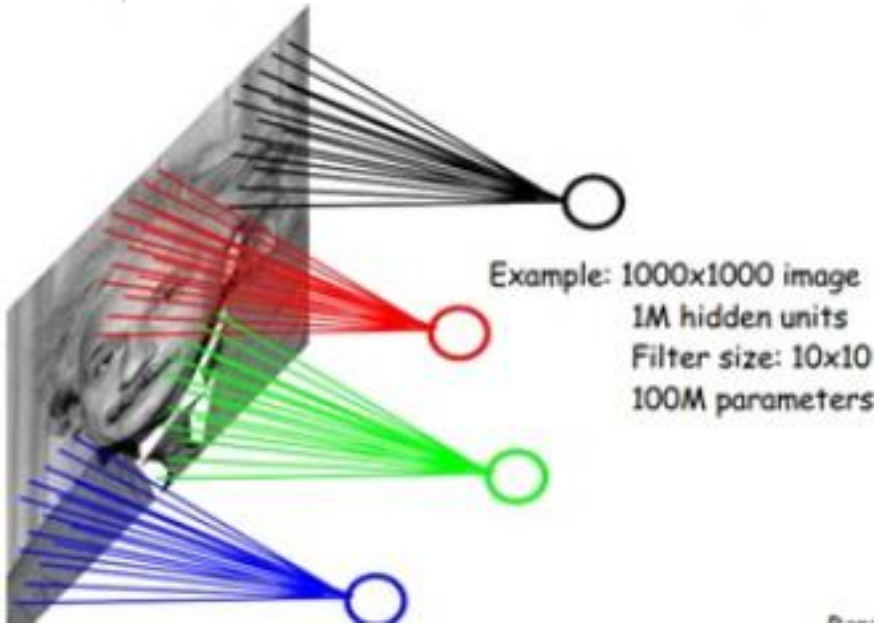


(b) 卷积层

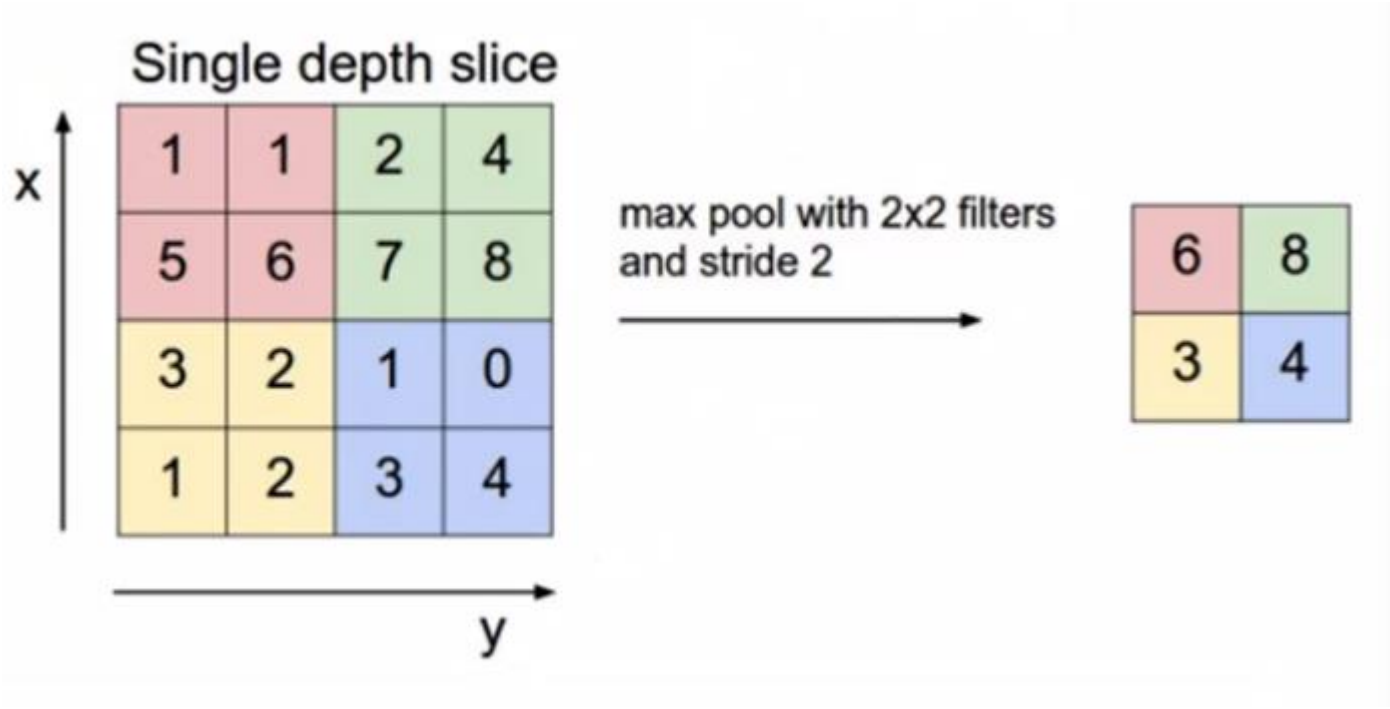


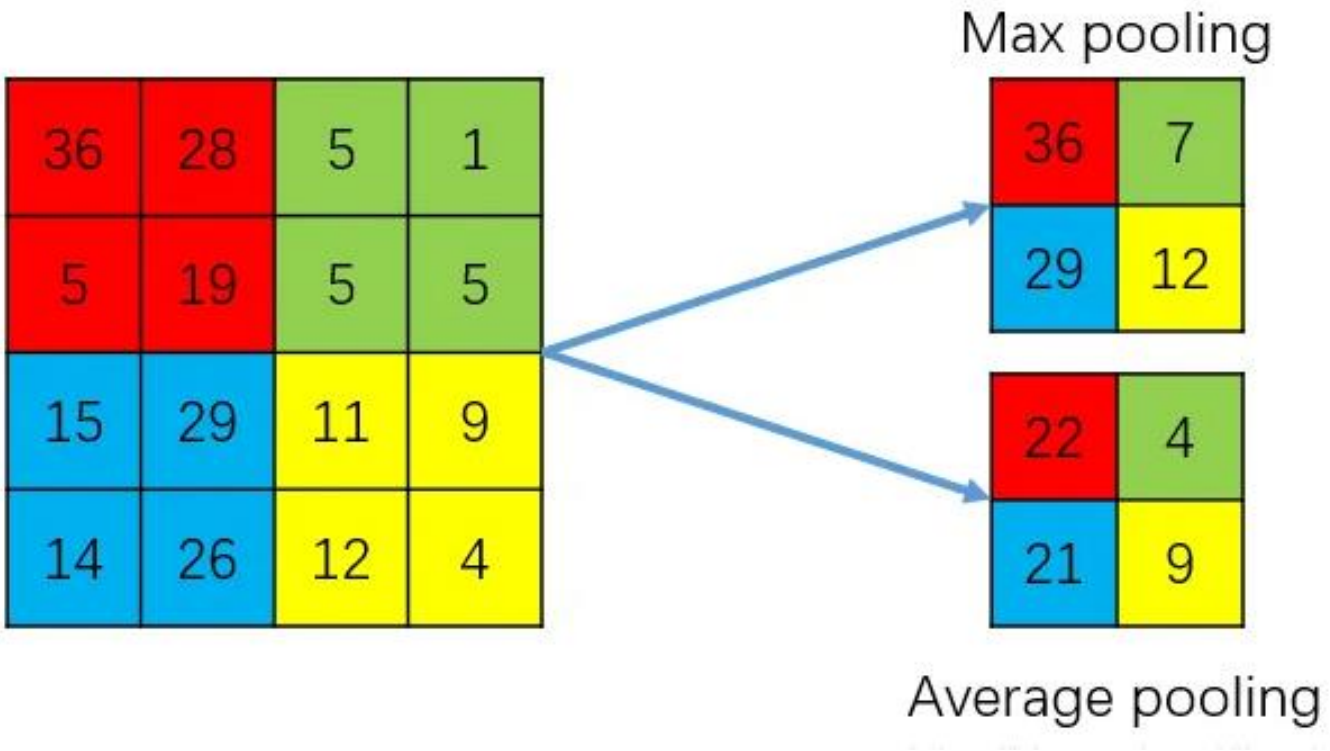
一、卷积神经网络


Example: 1000x1000 image
1M hidden units
10^12 parameters!!!
- Spatial correlation is local
- Better to put resources elsewhere!


Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

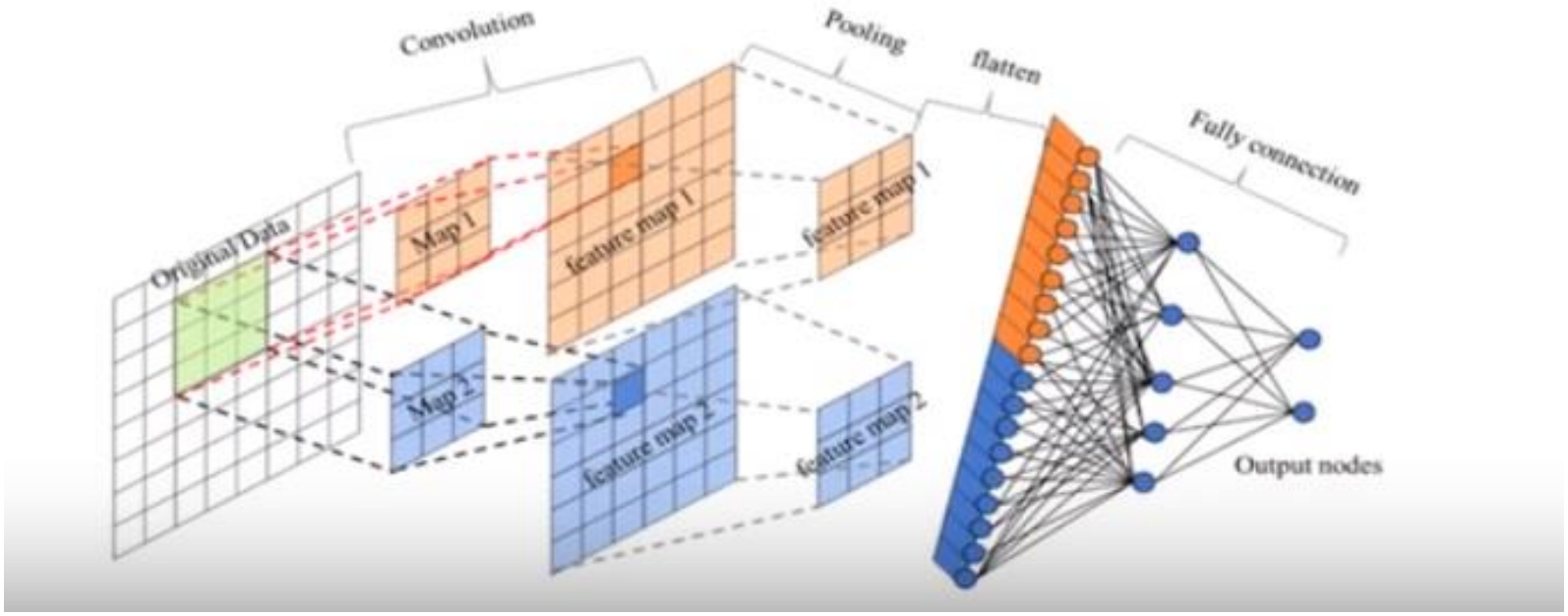
3、池化层

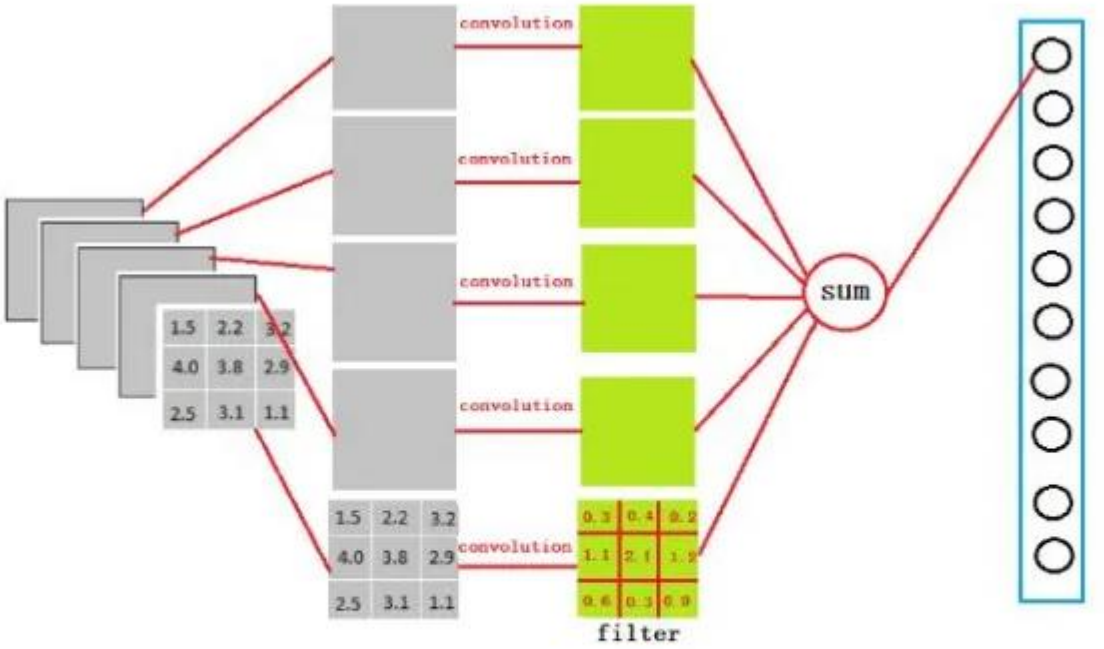

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4



对于一个输入的图片，我们使用一个区域大小为2 2，*步长为2的参数进行求最大值操作。同样池化也有一组参数，* f, s，*得到*2 2的大小。当然如果我们调整这个超参数，比如说3 * 3，那么结果就不一样了，通常选择默认都是$f = 2 * 2, s = 2$

4、全连接层

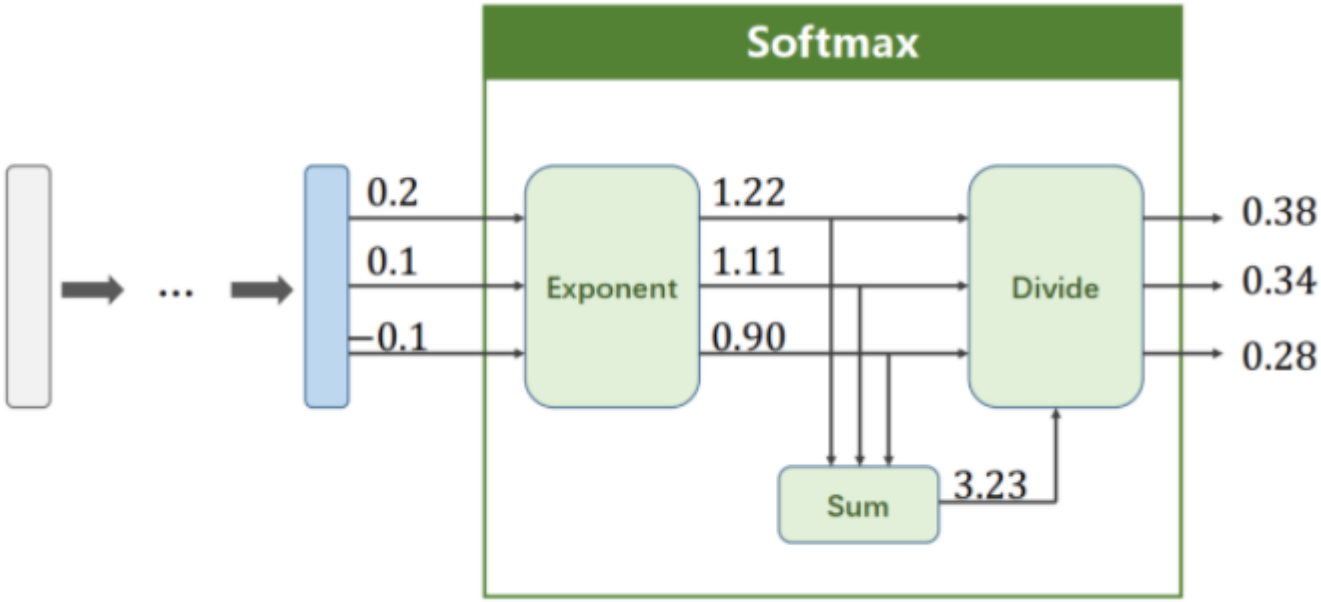

Convolution
Pooling
flatten
Fully connection
Original Data
Map 1
Map 2
feature map 1
feature map 2
feature map 1
feature map 2
Output nodes



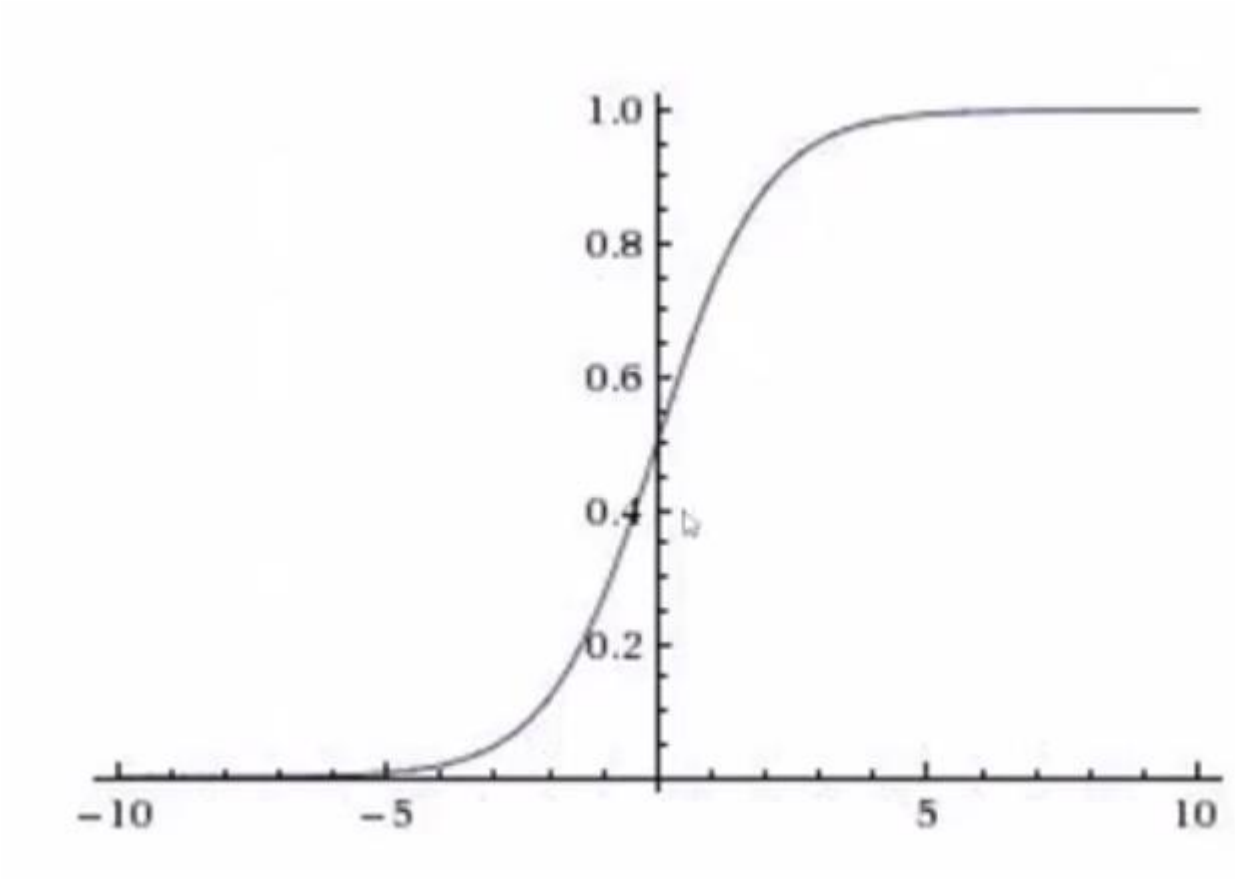
对224x224x3的输入，最后一层卷积可得输出为7x7x512，如后层是一层含4096个神经元的FC，则可用卷积核为7x7x512x4096的全局卷积来实现这一全连接运算过程。

5、输出层

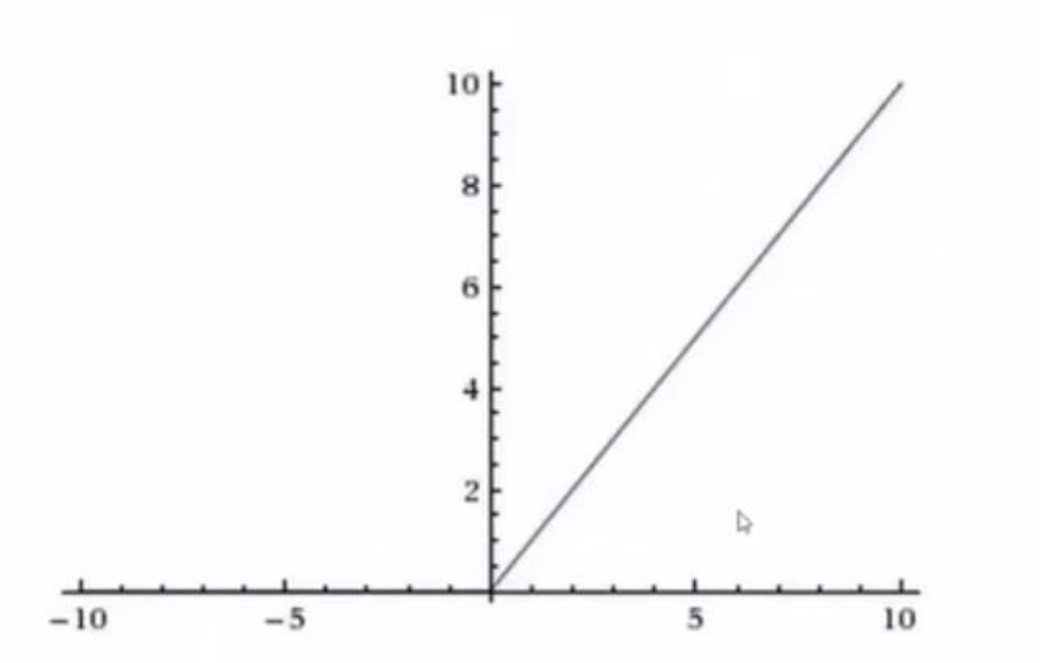
$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \ for\ j = 1, \cdots, K$$



6、激活函数



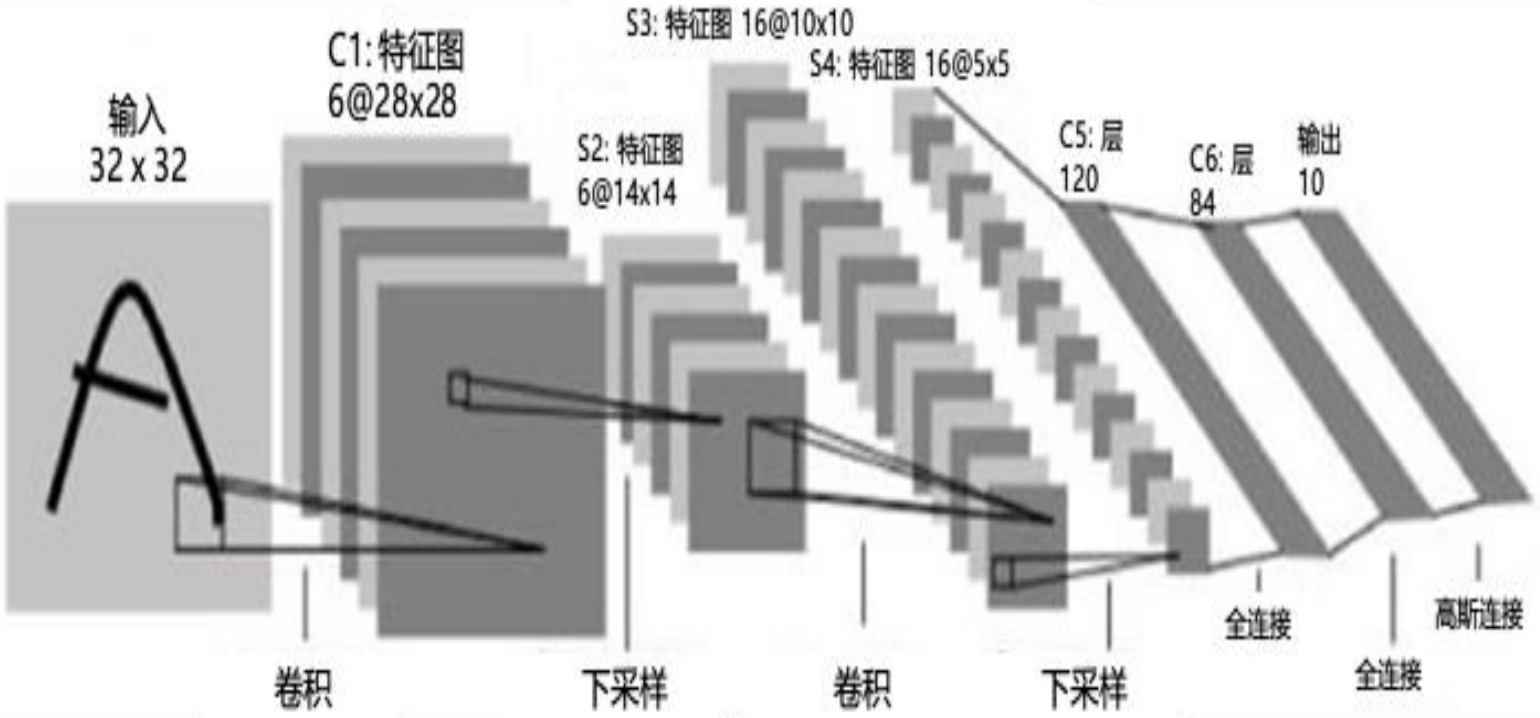
$$f(x) = \max(x, 0)$$

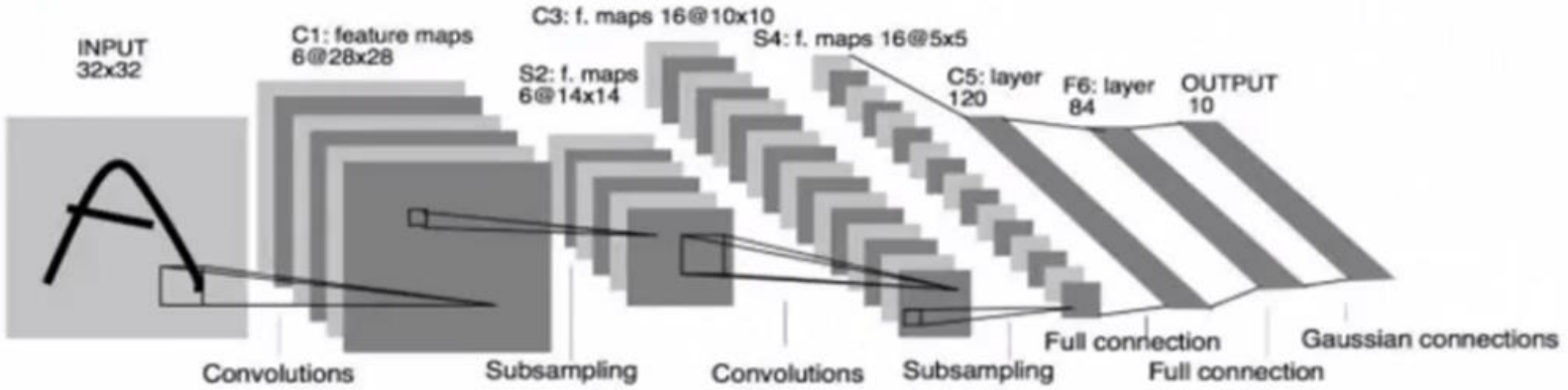


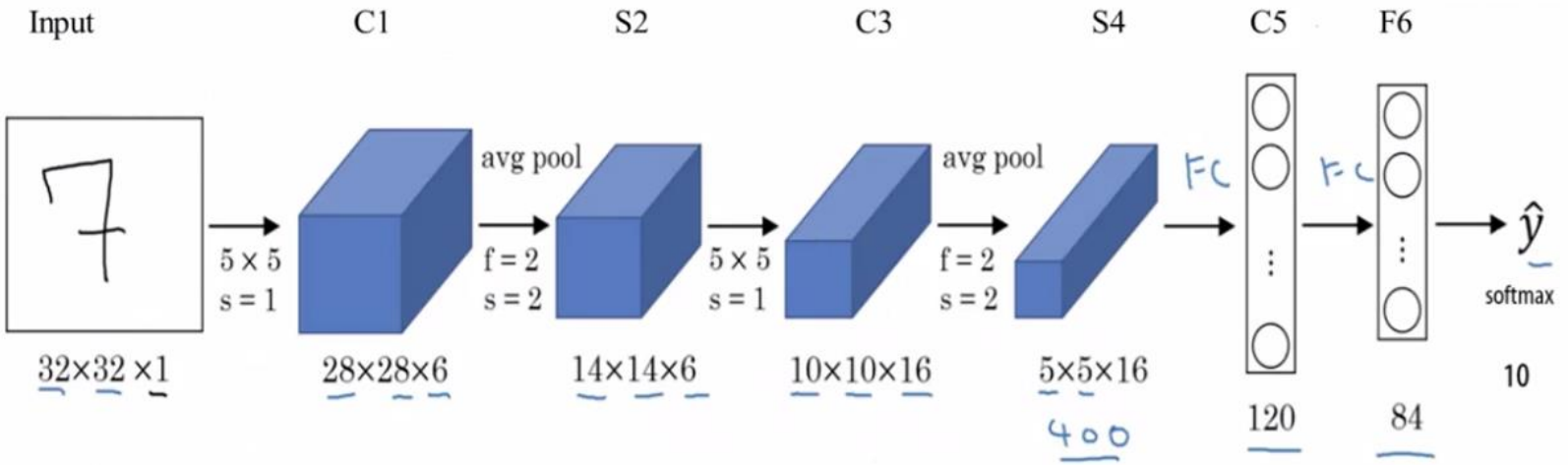
7、典型卷积神经网络

LeNet

- 由LeCun在1989年提出，被用于手写字符的识别
- 第一个真正意义上的卷积神经网络，是当前各种深度卷积神经网络的鼻祖

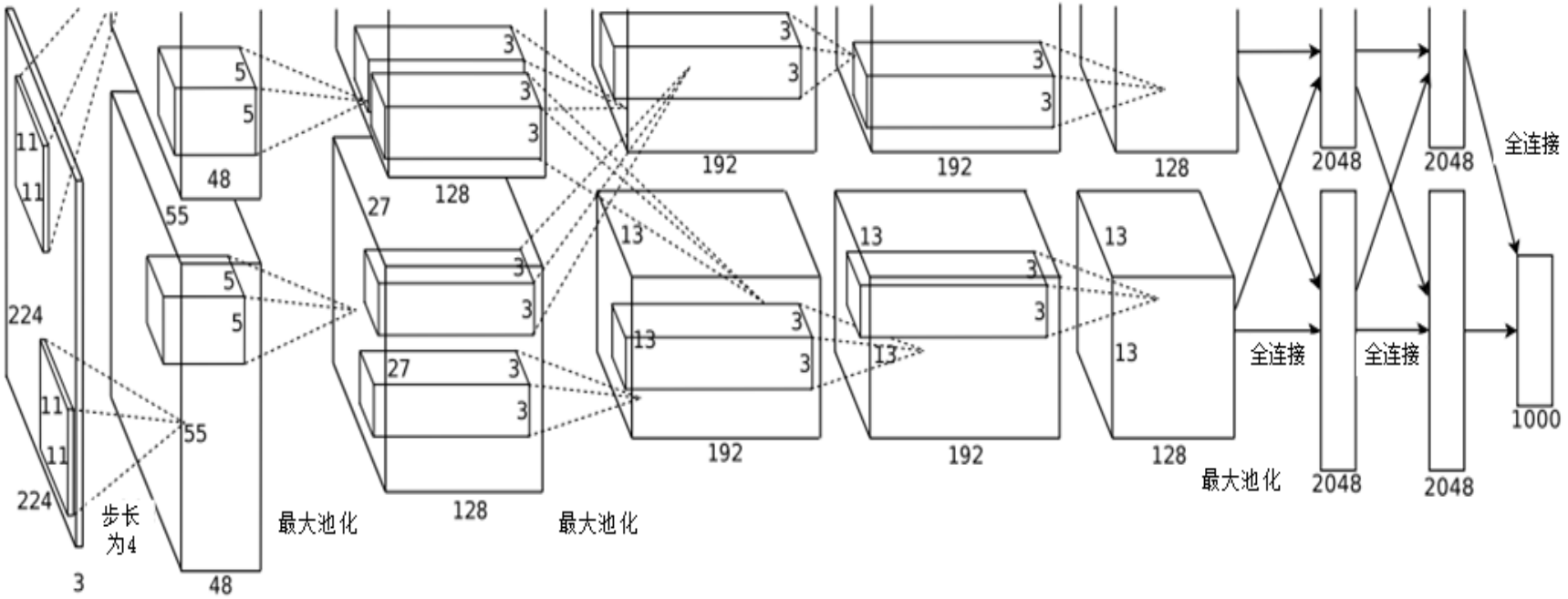



INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

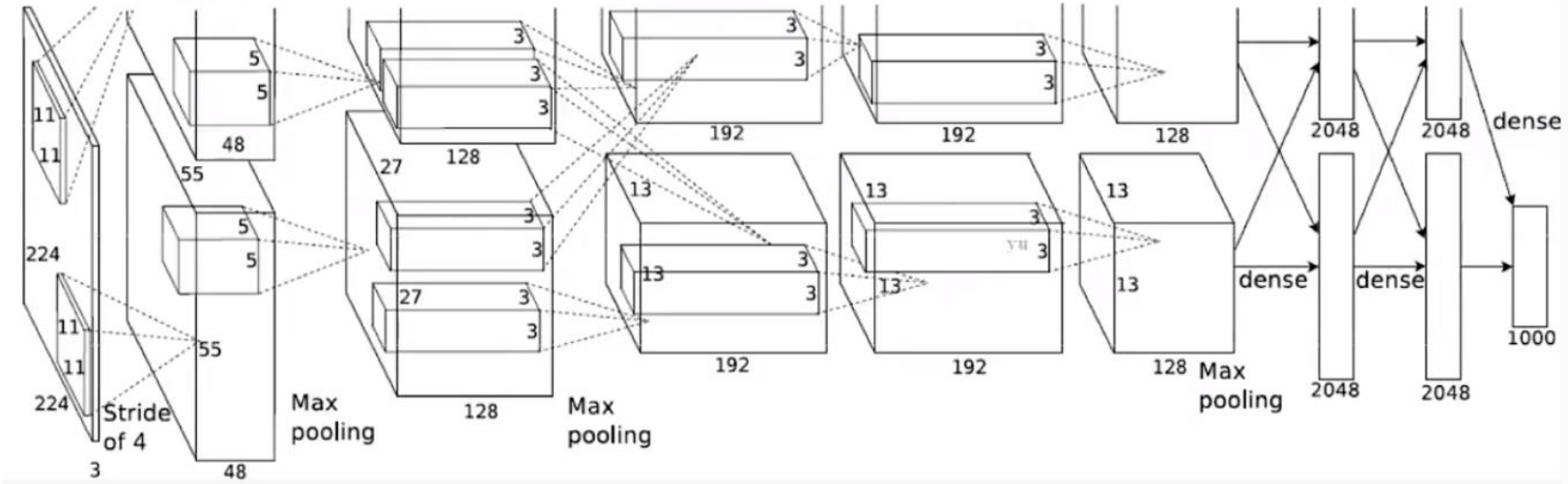

Input
C1
S2
C3
S4
C5
F6
5 × 5
s = 1
avg pool
f = 2
s = 2
5 × 5
s = 1
avg pool
f = 2
s = 2
FC
FC
softmax
32×32 ×1
28×28×6
14×14×6
10×10×16
5×5×16
400
120
84
10

AlexNet

- 现代意义上的深度卷积神经网络起源于AlexNet网络，是深度卷积神经网络的鼻祖
- 与之前的卷积网络相比，最显著的特点是层次加深，参数规模变大



网络结构

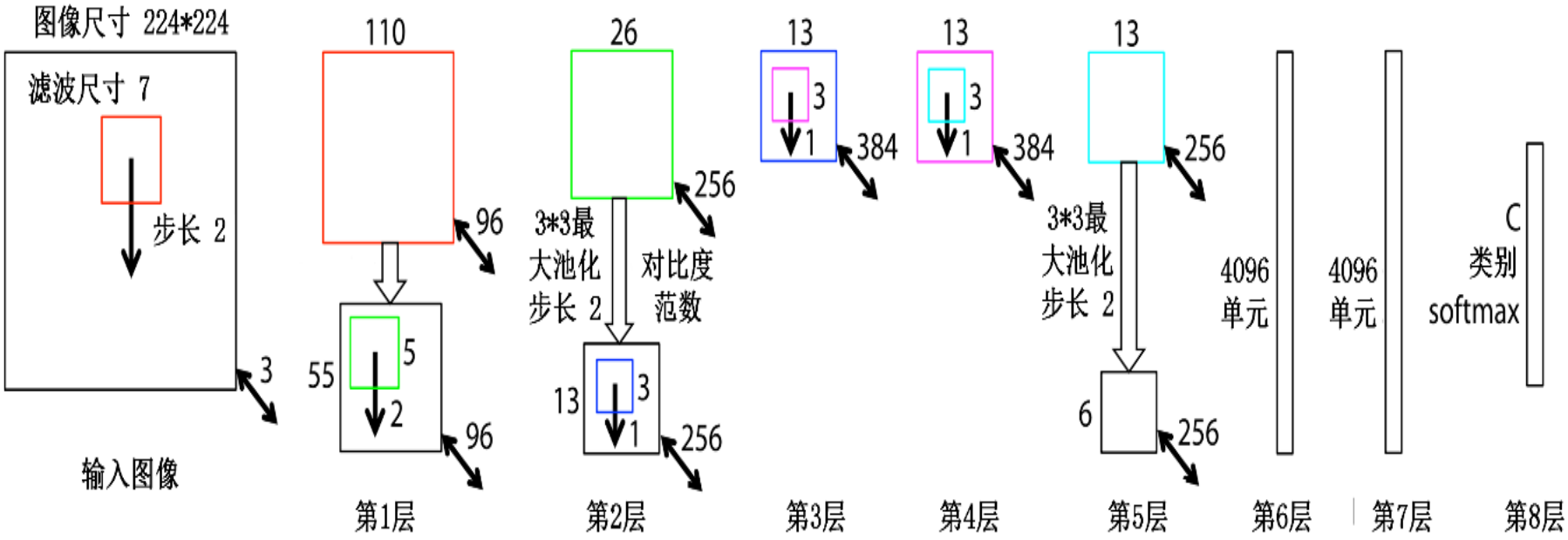


网络结构

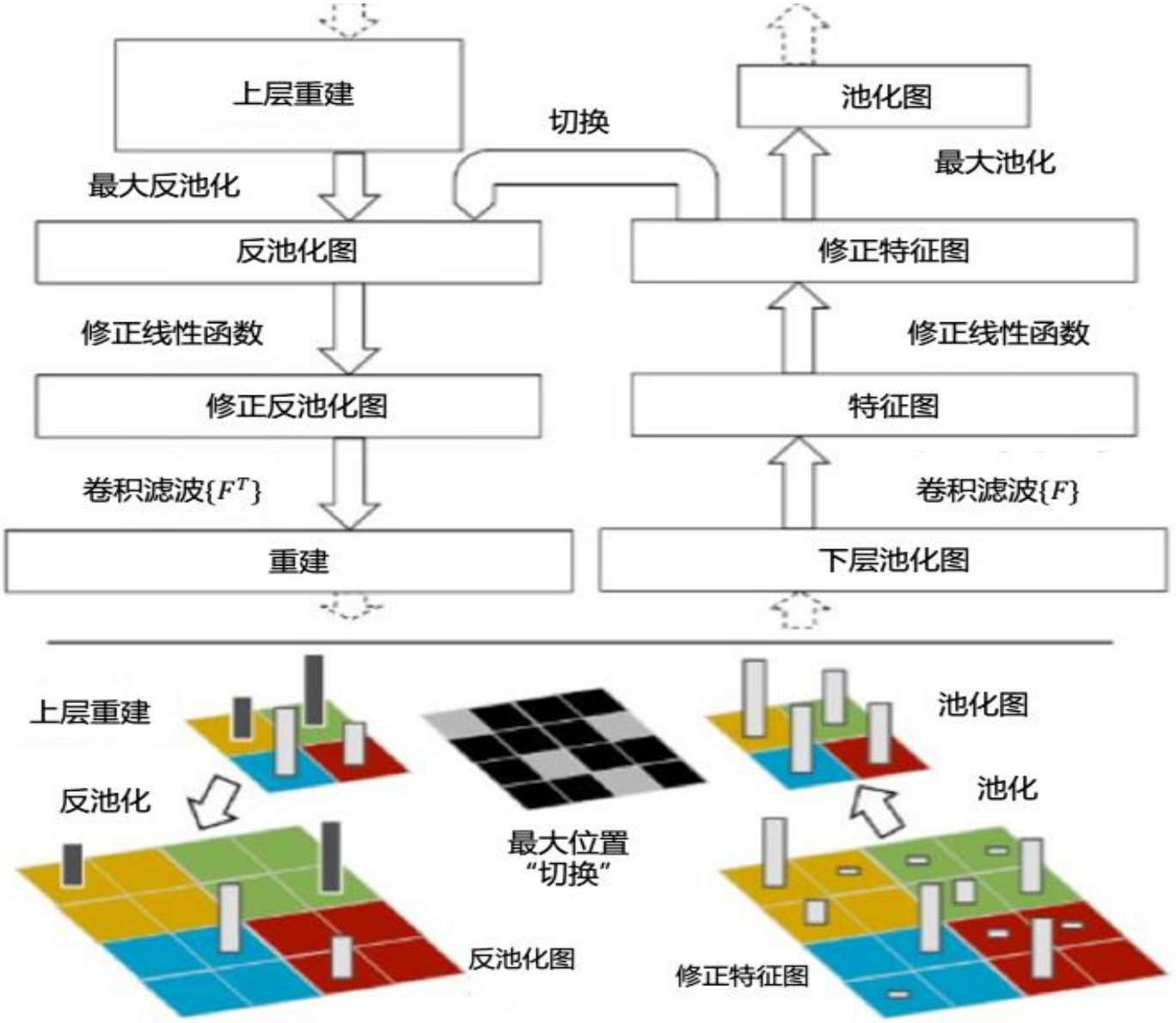
- 输入: 224x224, 3通道
- 第一层卷积: 11x11的卷积核, 96个, stride=4
- 第二层卷积: 5x5的卷积核256个
- 第二层max_pooling: 2x2
- ……
- 第一层全连接:4096
- 第二层全连接:4096
- Softmax: 1000,概率值

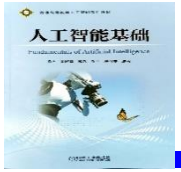
ZFNet

- 通过反卷积（转置卷积）进行卷积网络层可视化的方法，分析卷积网络的效果并指导网络的改进
- 最大贡献在于通过使用可视化技术揭示了神经网络各层到底在干什么，起到了什么作用



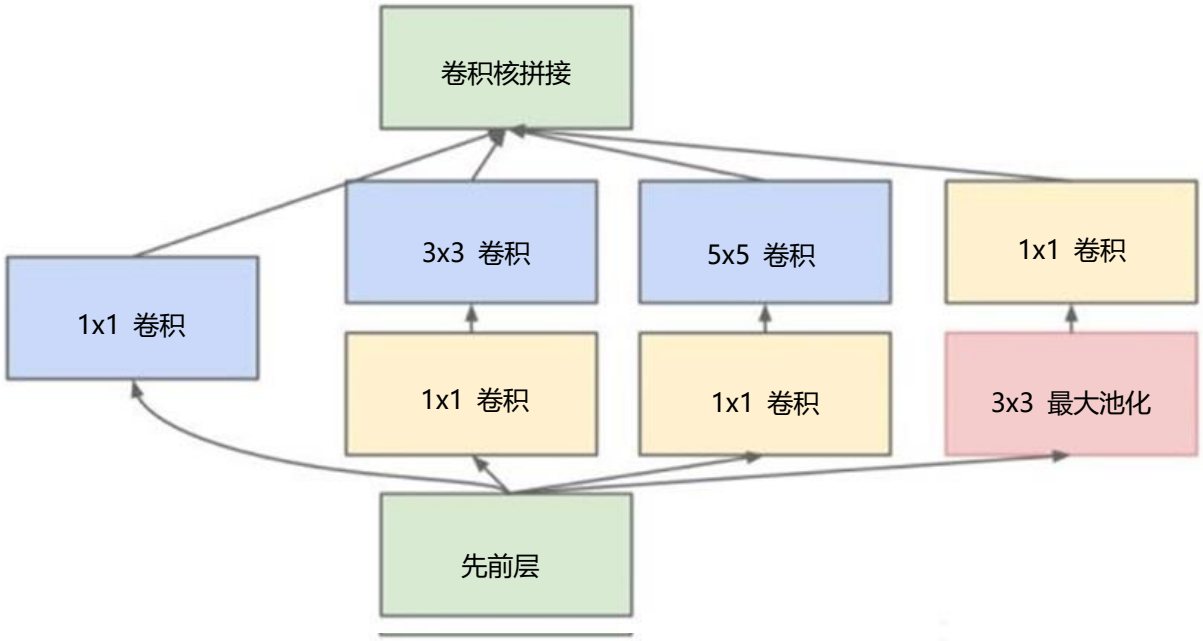
反卷积网络


上层重建
最大反池化
反池化图
修正线性函数
修正反池化图
卷积滤波{F^T}
重建
切换
池化图
最大池化
修正特征图
修正线性函数
特征图
卷积滤波{F}
下层池化图
上层重建
反池化
反池化图
最大位置
“切换”
池化图
池化
修正特征图



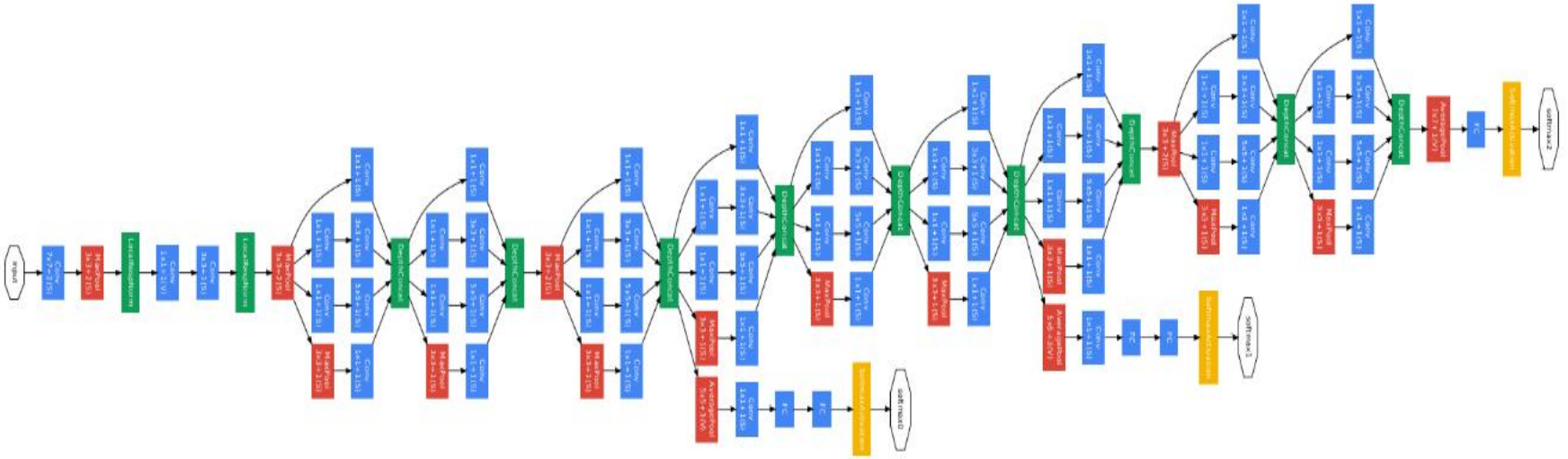
GoogLeNet网络

- Inception机制：将多个不同尺度的卷积核，池化层进行整合，形成一个Inception模块
- Inception模块：由3组卷积核以及一个池化单元组成，共同接受来自前一层的输入图像，并行的对输入图像进行处理，然后将输出结果按照通道拼接起来



GoogLeNet网络

- 为了降低网络参数，GoogLeNet网络去除了最后的全连接层，用全局平均池化替代
- 全连接层几乎占据了AlexNet中90%的参数量，而且会引起过拟合
- 去除全连接层后模型训练更快并且减轻了过拟合




系安全带
没系安全带
结果
1，传统算法最好的结果是74.9%
2，深度学习的结果90%以上

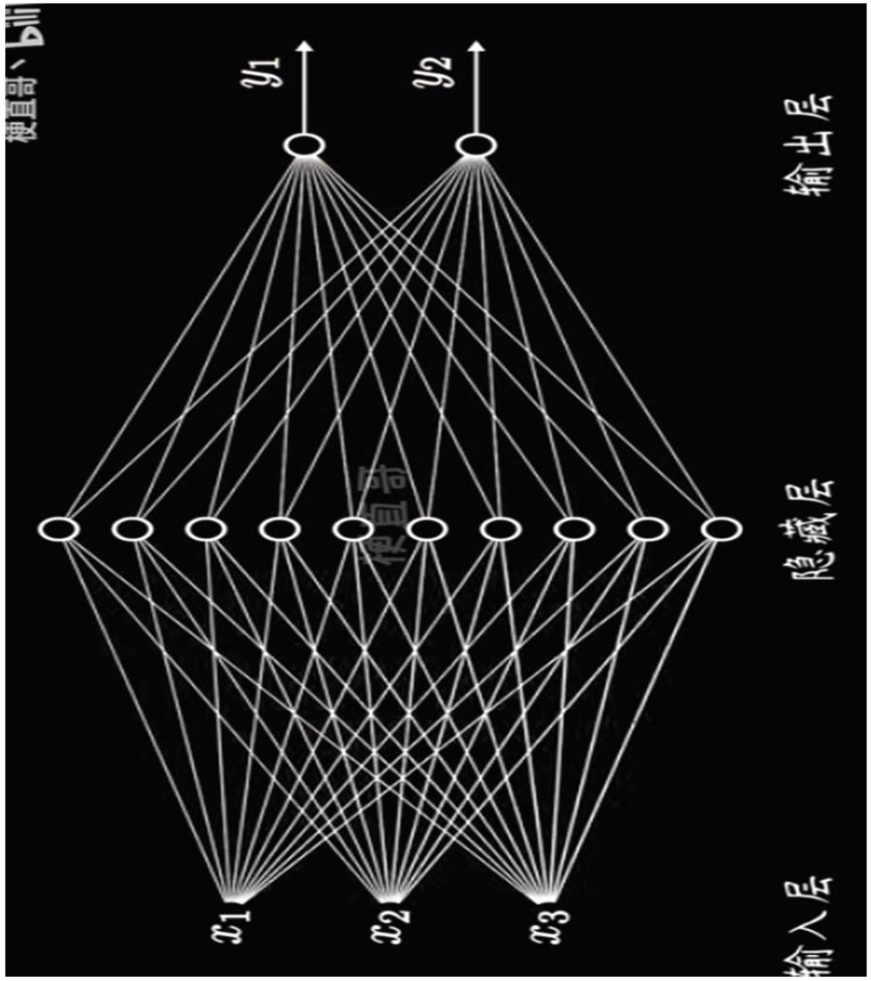
循环神经网络

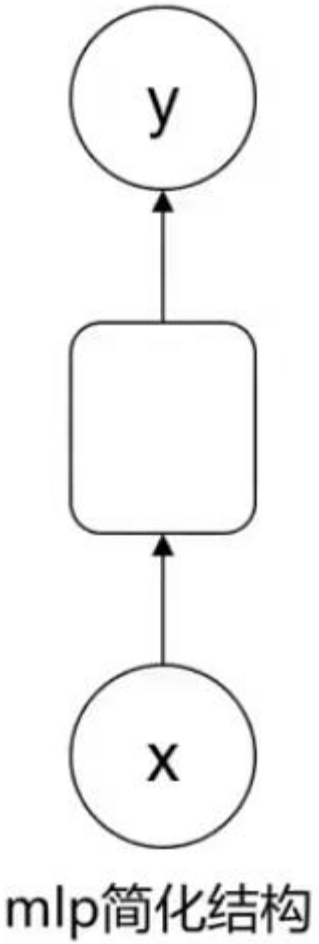
1、如何处理时间序列数据？

2、多层感知器与卷积神经网络适合吗？

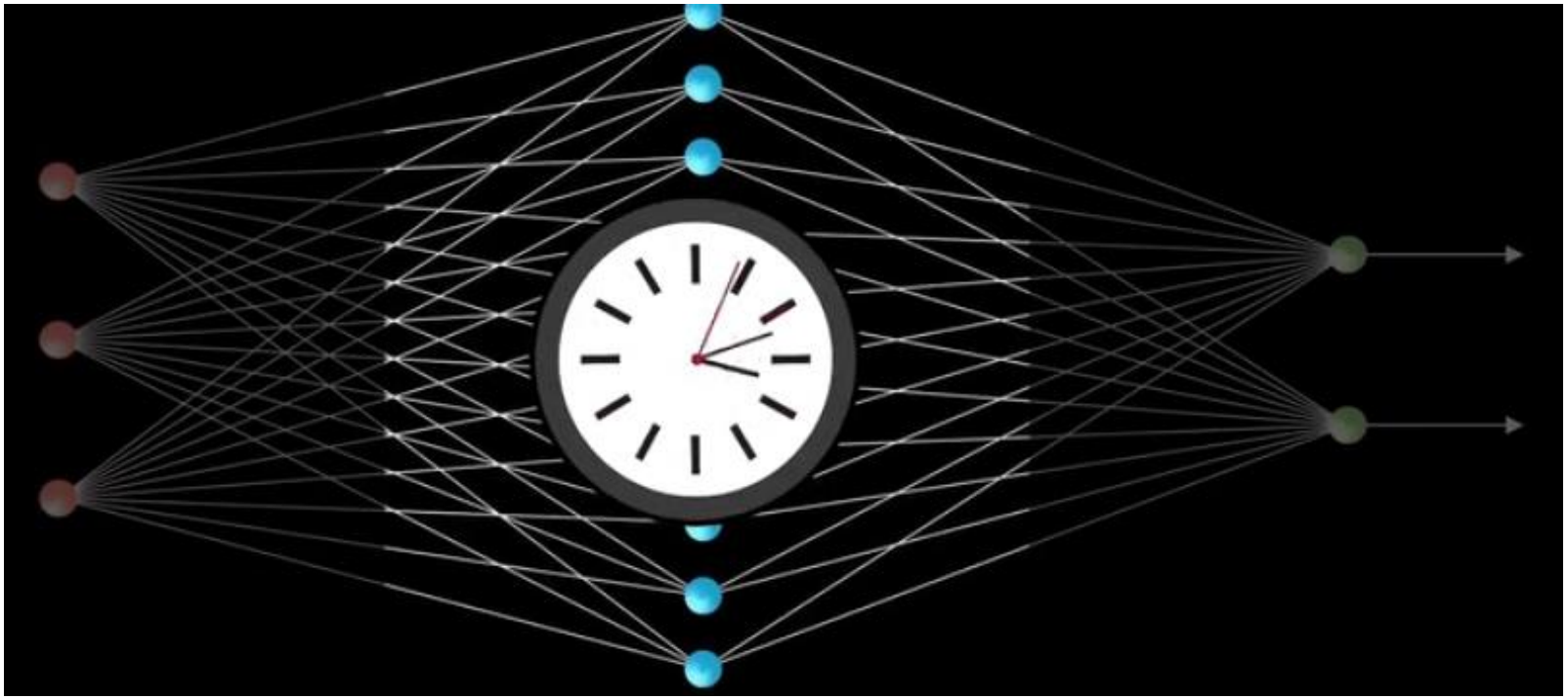
3、循环神经网络（RNN）怎么处理时序数据？

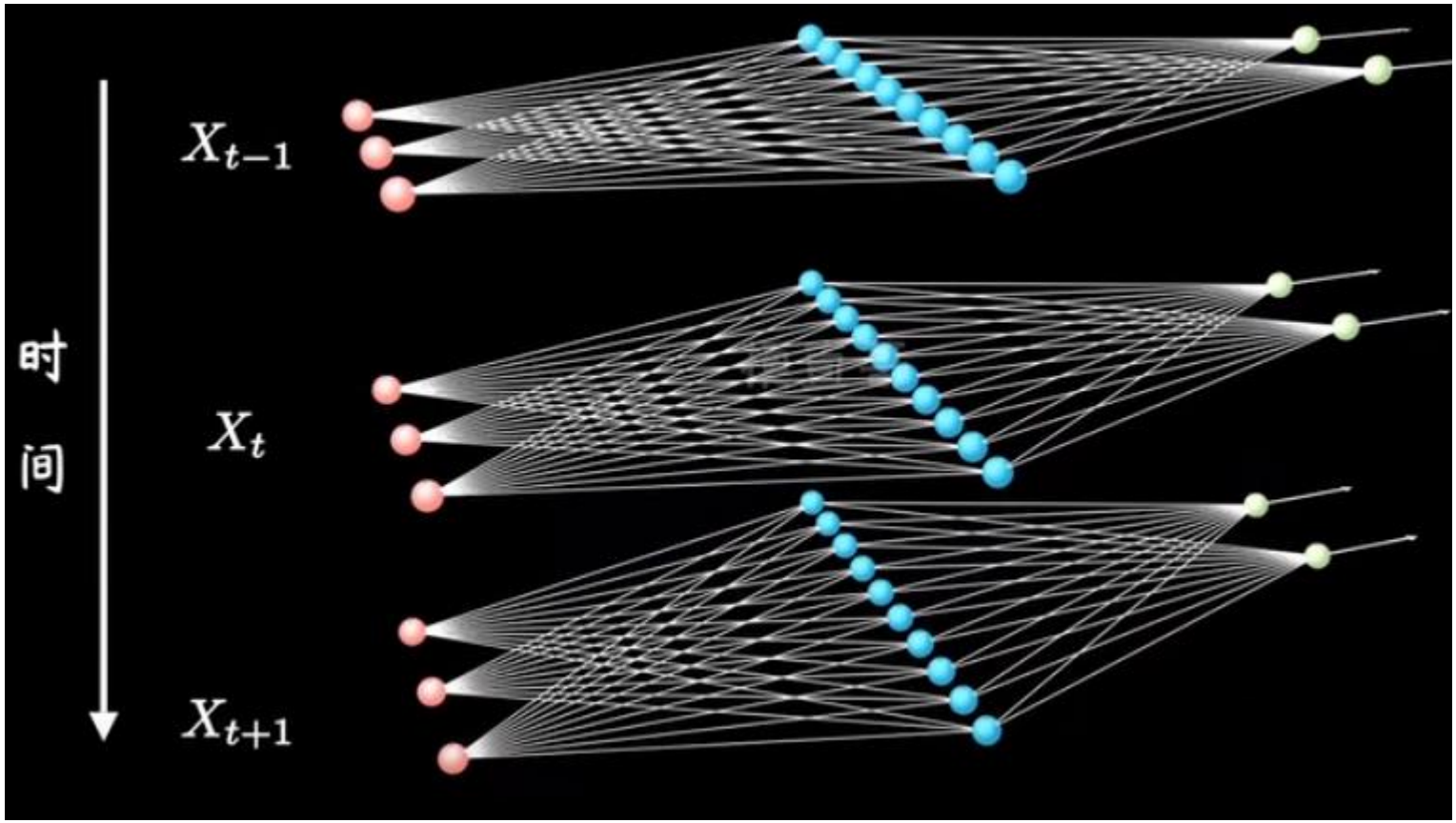
二、循环神经网络

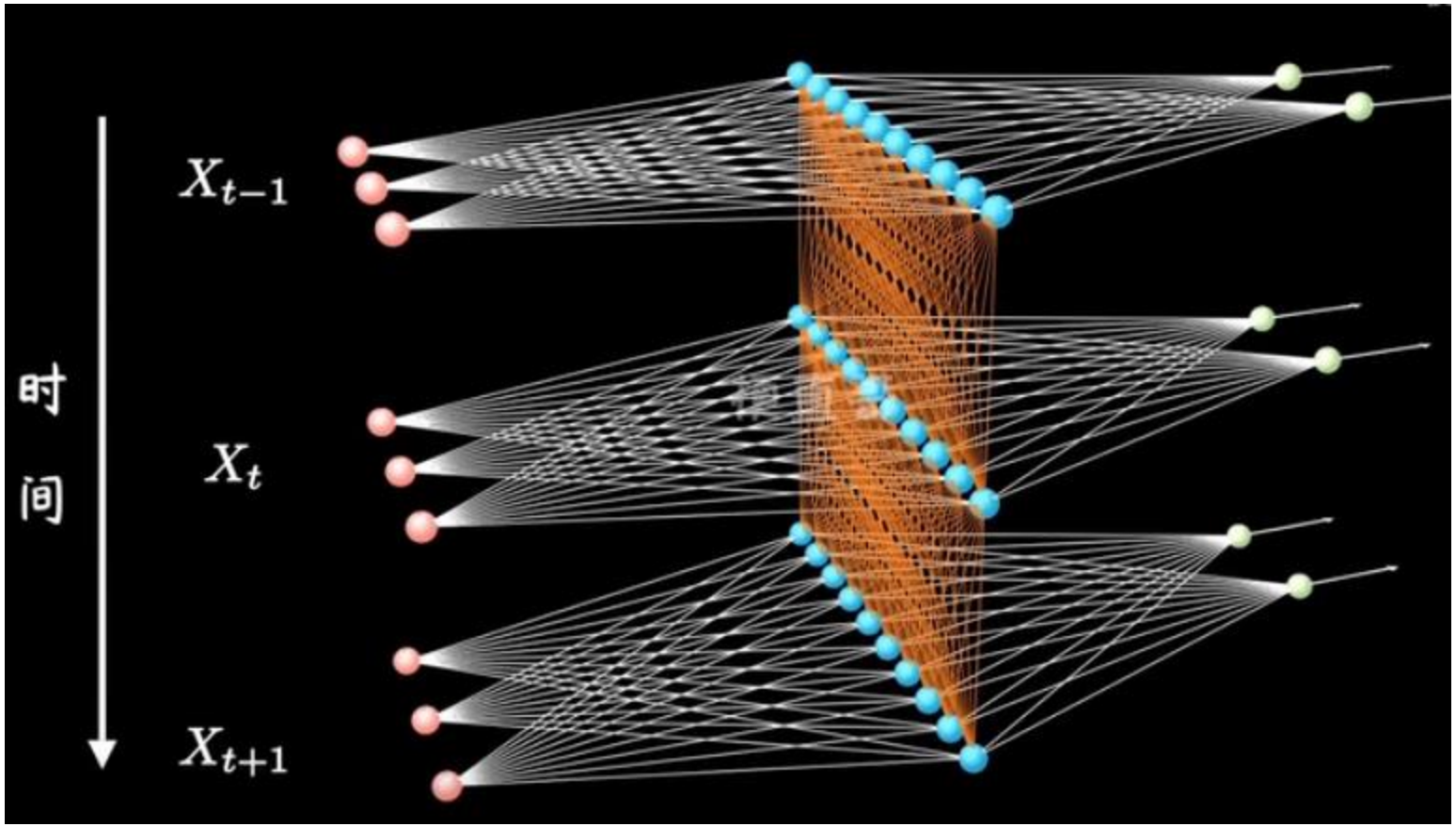


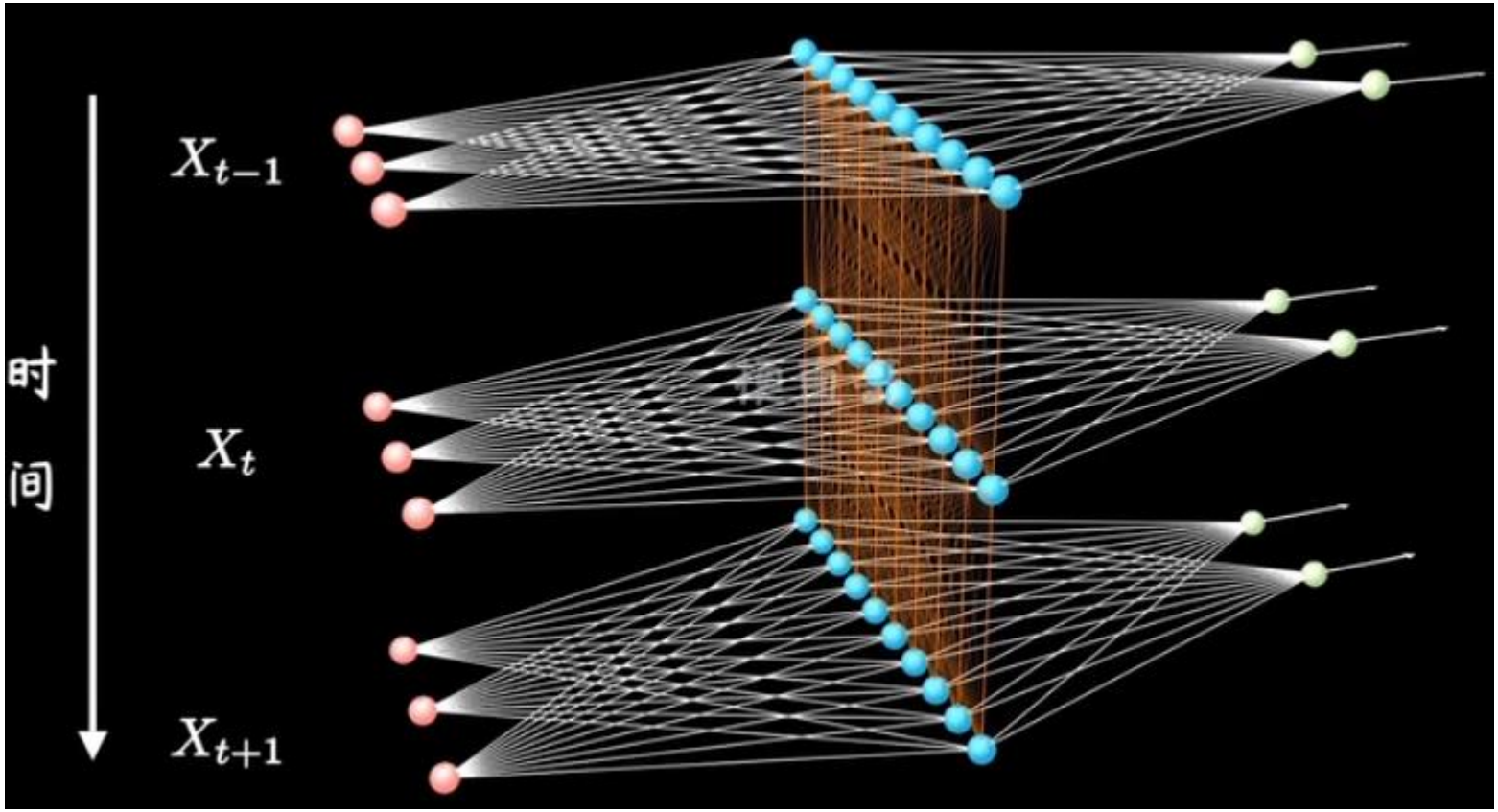


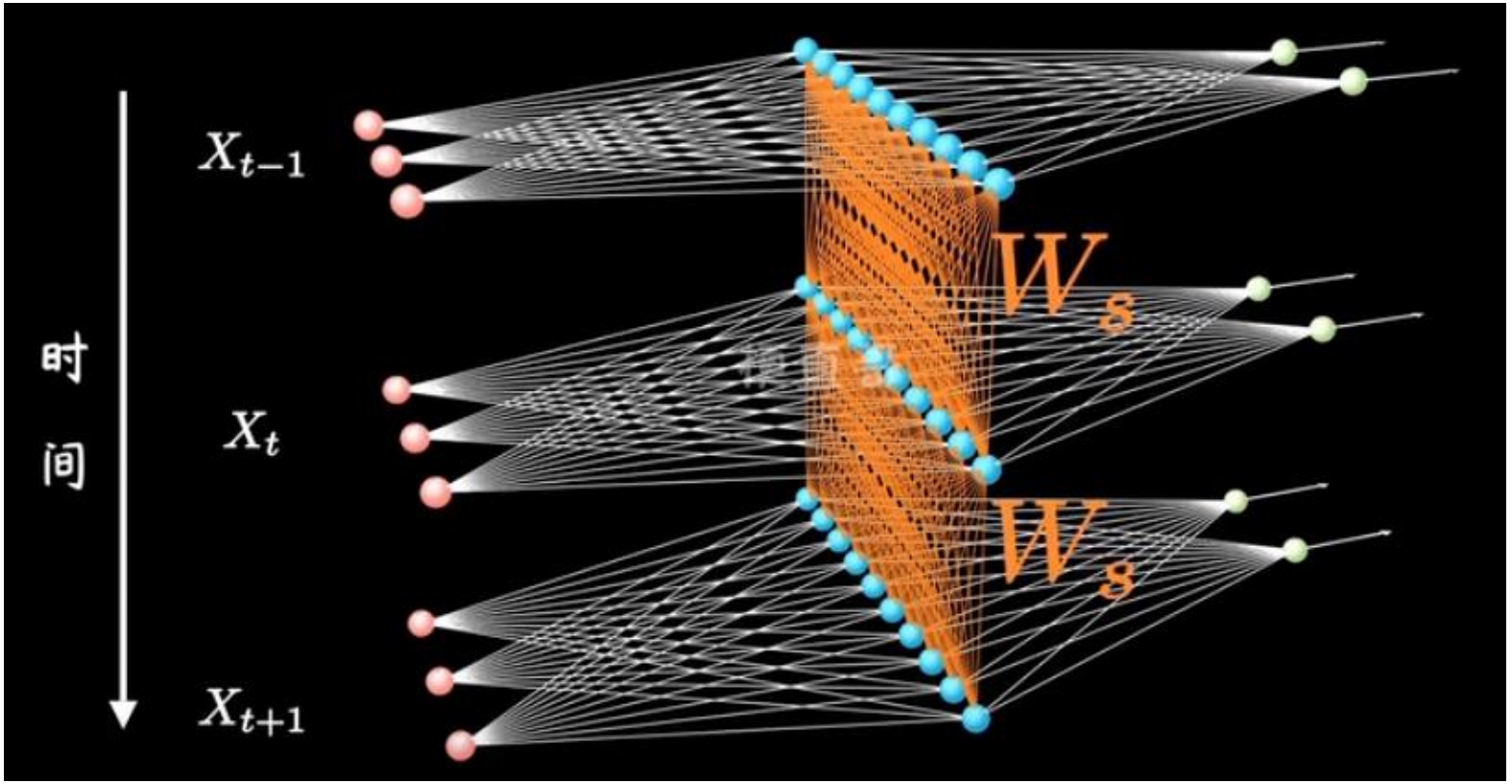
mlp简化结构

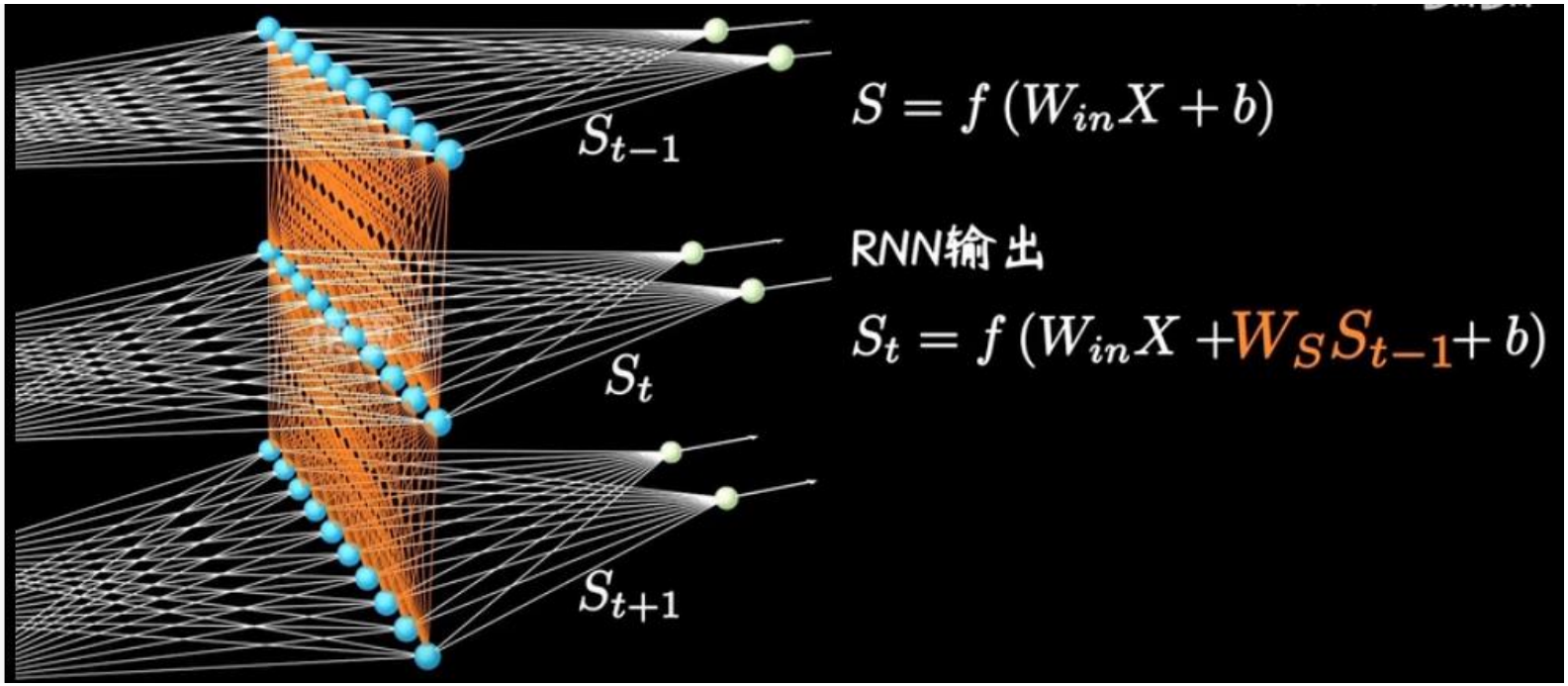


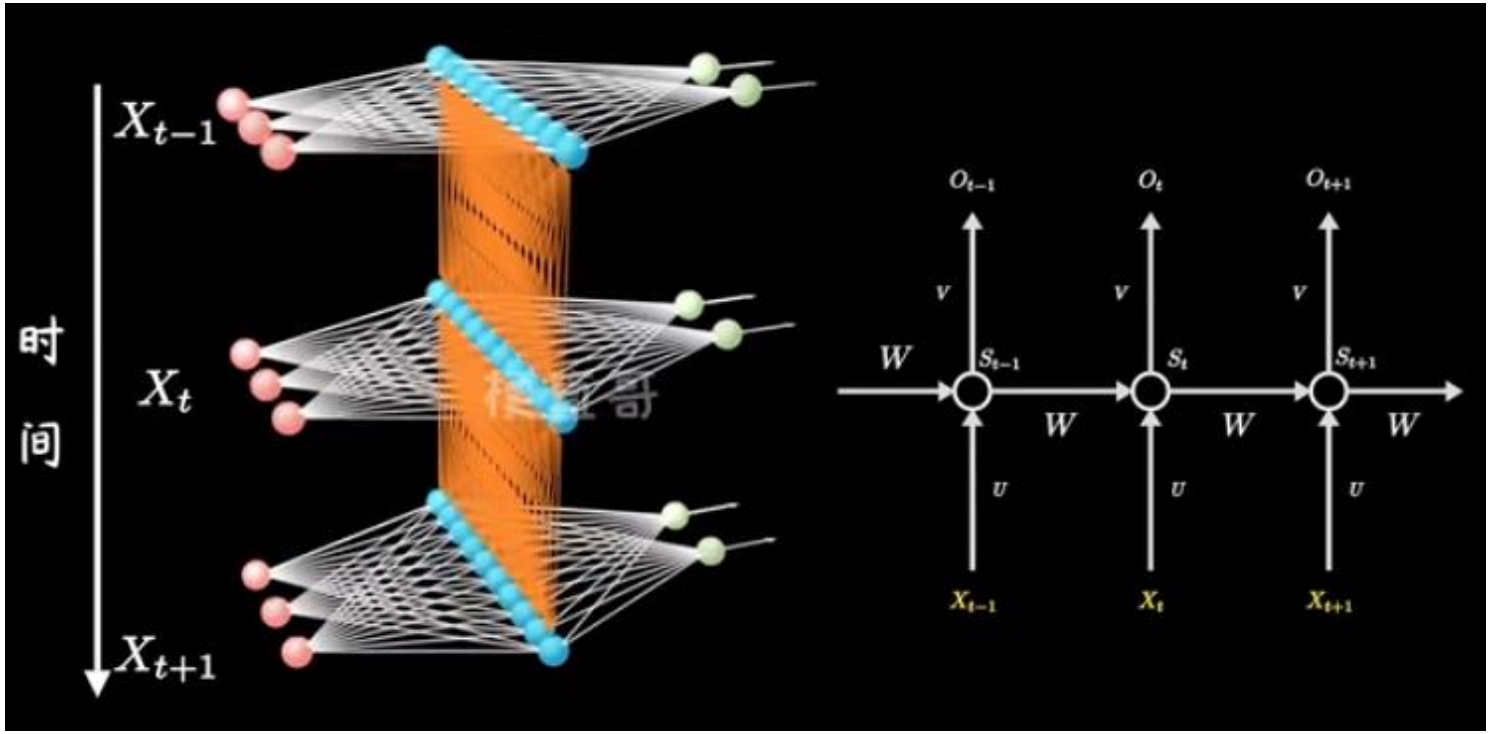

X_{t-1}
时间
X_t
X_{t+1}

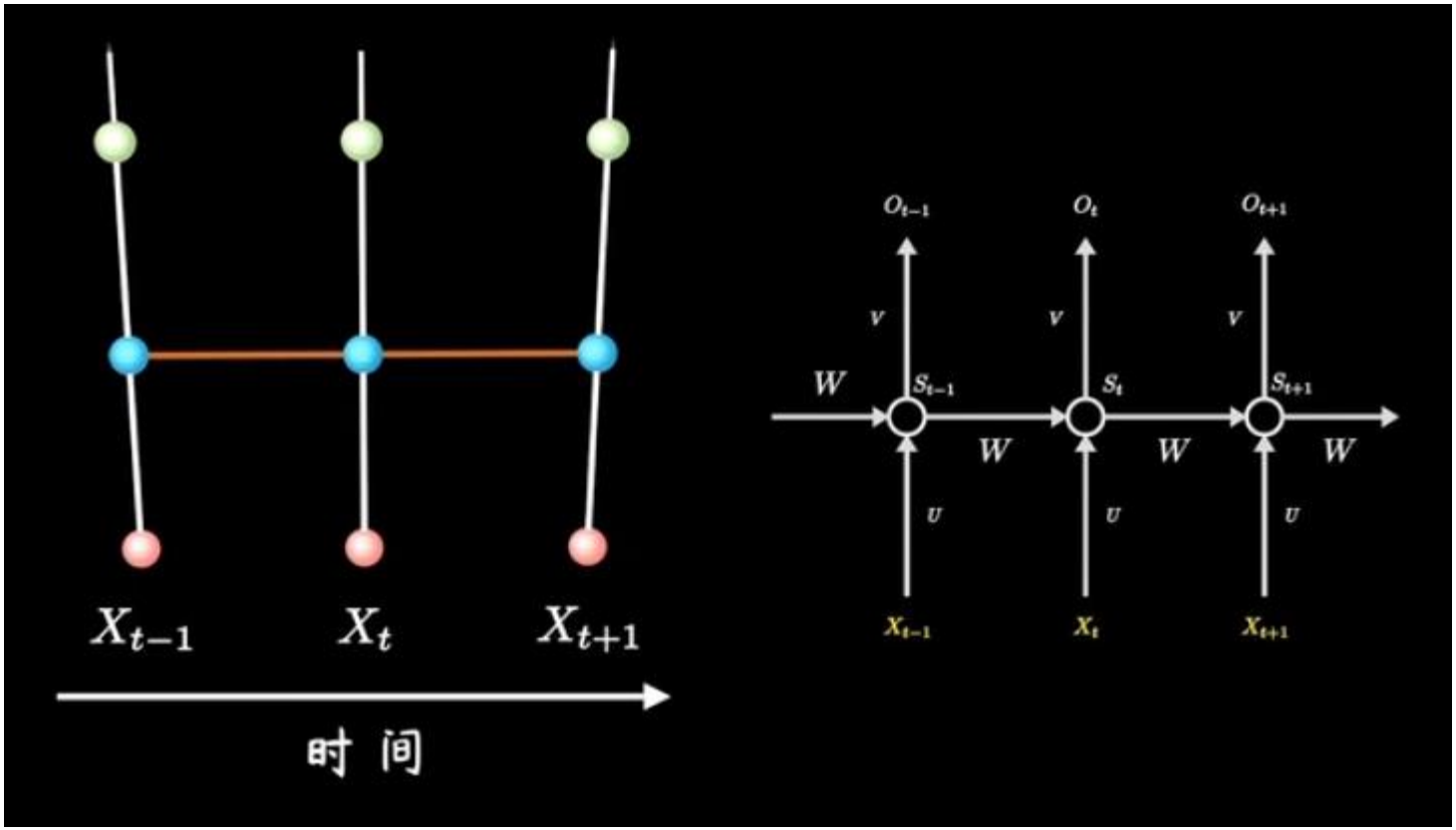

X_{t-1}
时
间
X_t
X_{t+1}

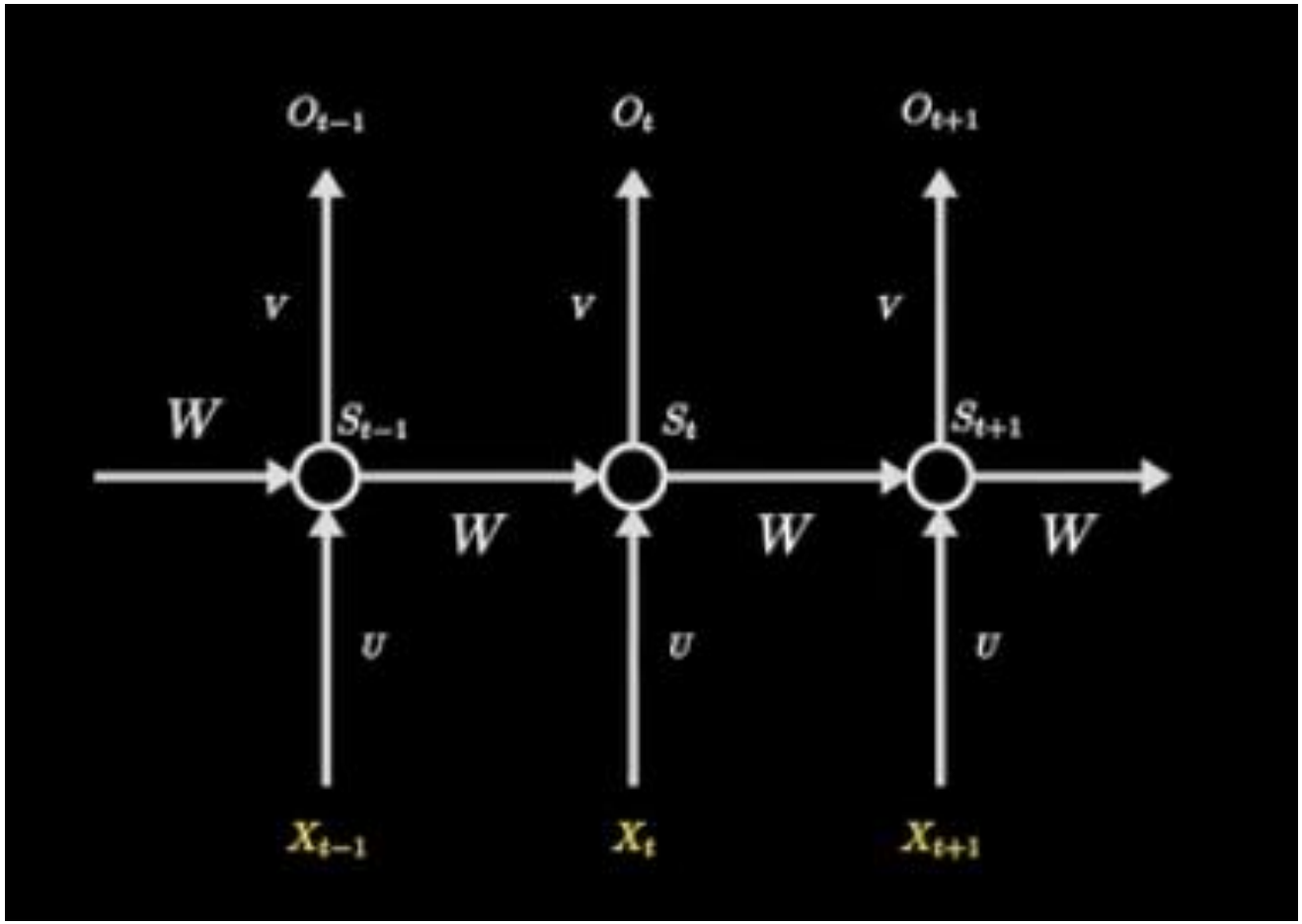

X_{t-1}
时
间
X_t
X_{t+1}

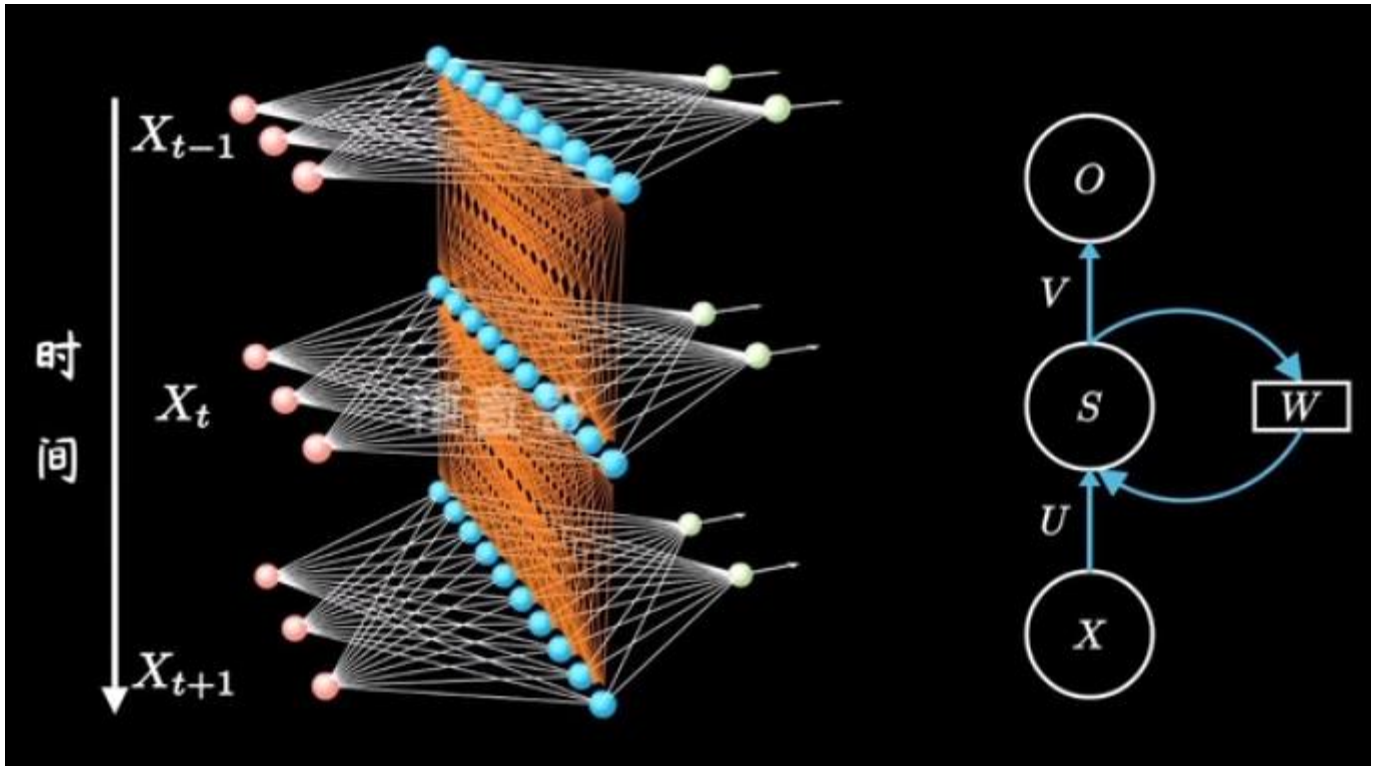

时间
X_{t-1}
X_t
X_{t+1}
W_s
W_s

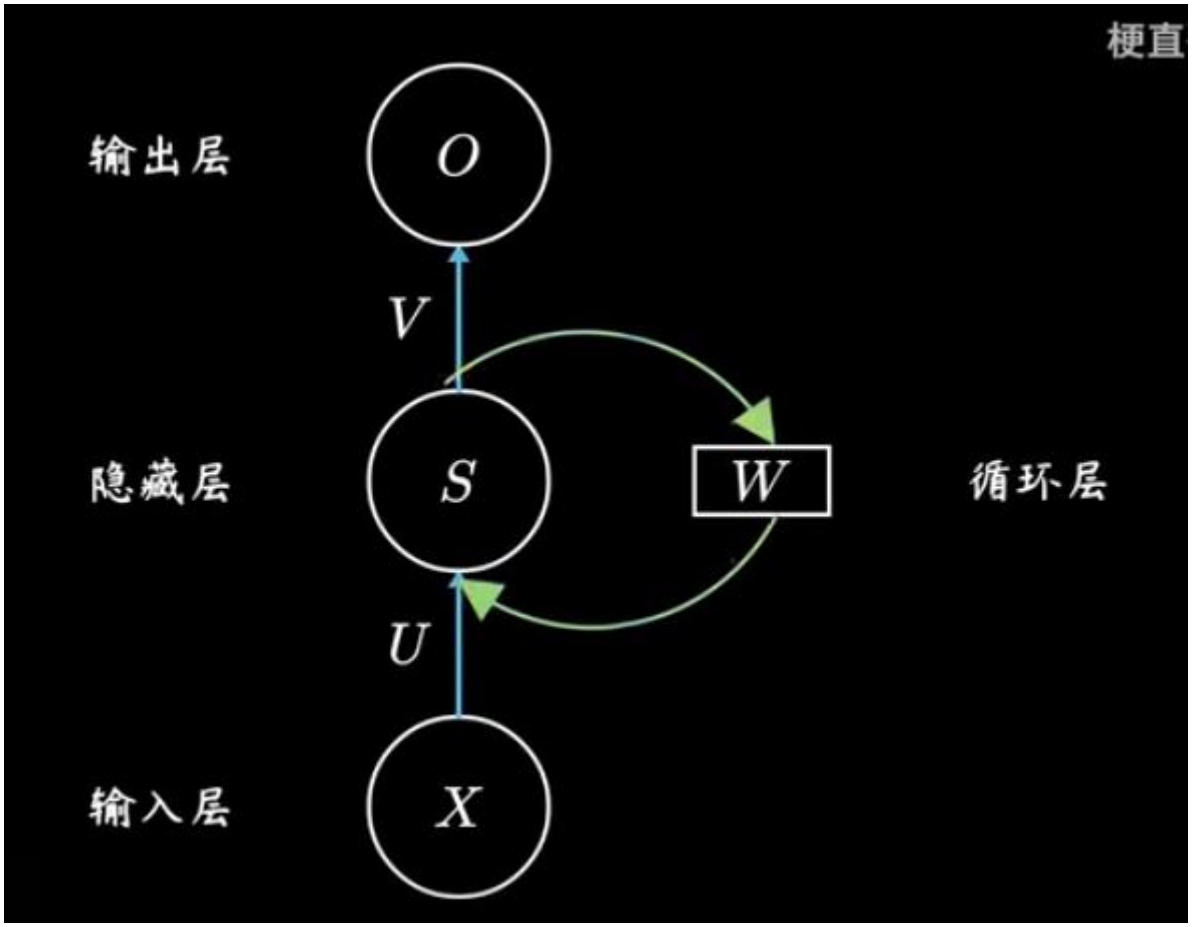

S_{t-1}
S_t
S_{t+1}
$S = f(W_{in}X + b)$
RNN输出
$S_t = f(W_{in}X + W_S S_{t-1} + b)$


时间
X_{t-1}
X_t
X_{t+1}
O_{t-1}
O_t
O_{t+1}
V
W
U
S_{t-1}
S_t
S_{t+1}
X_{t-1}
X_t
X_{t+1}

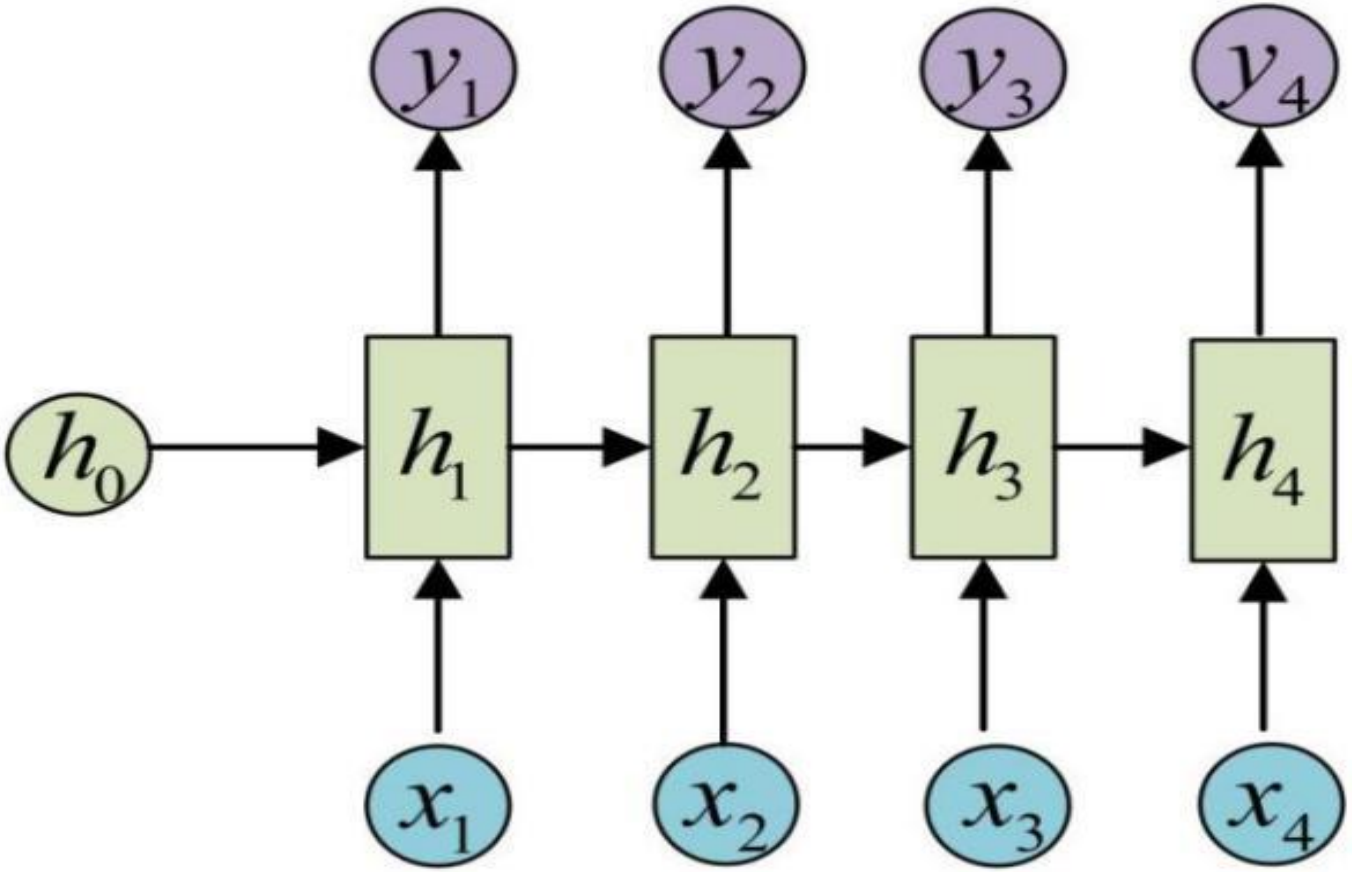

X_{t-1}
X_t
X_{t+1}
时 间
O_{t-1}
O_t
O_{t+1}
V
W
S_{t-1}
S_t
S_{t+1}
U
X_{t-1}
X_t
X_{t+1}


O_{t-1}
O_t
O_{t+1}
V
V
V
W
S_{t-1}
S_t
S_{t+1}
W
W
W
U
U
U
X_{t-1}
X_t
X_{t+1}


X_{t-1}
时间
X_t
X_{t+1}
O
V
S
W
U
X

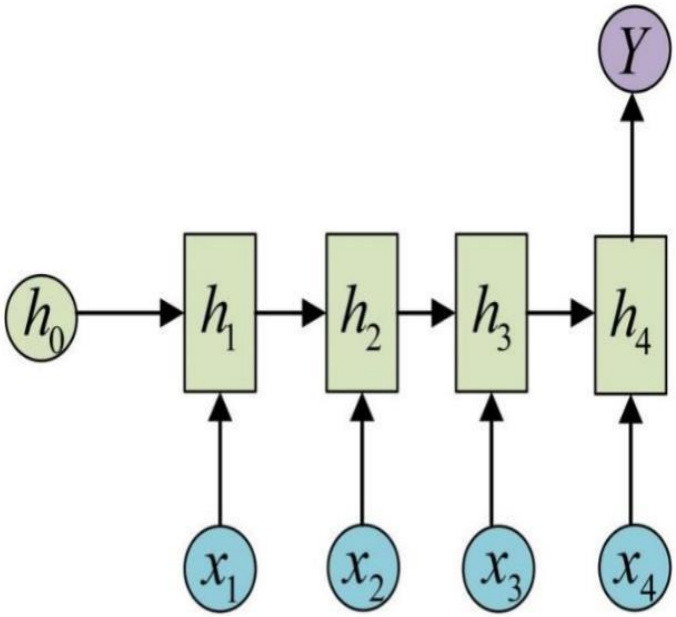

梗直
输出层
O
V
隐藏层
S
W
循环层
U
输入层
X

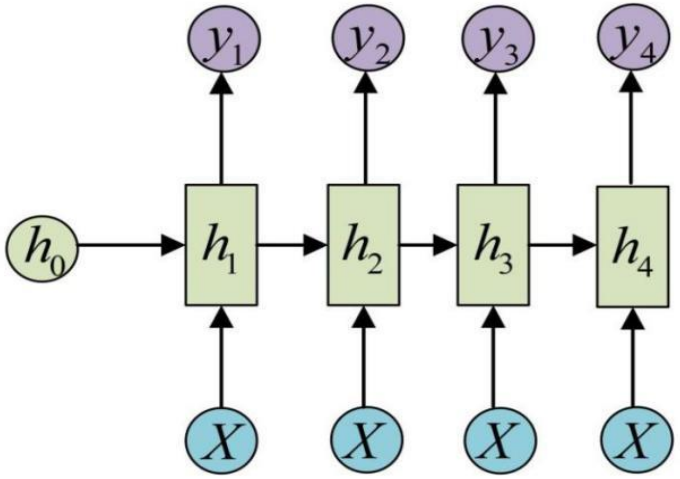
1、循环神经网络结构



RNN典型结构

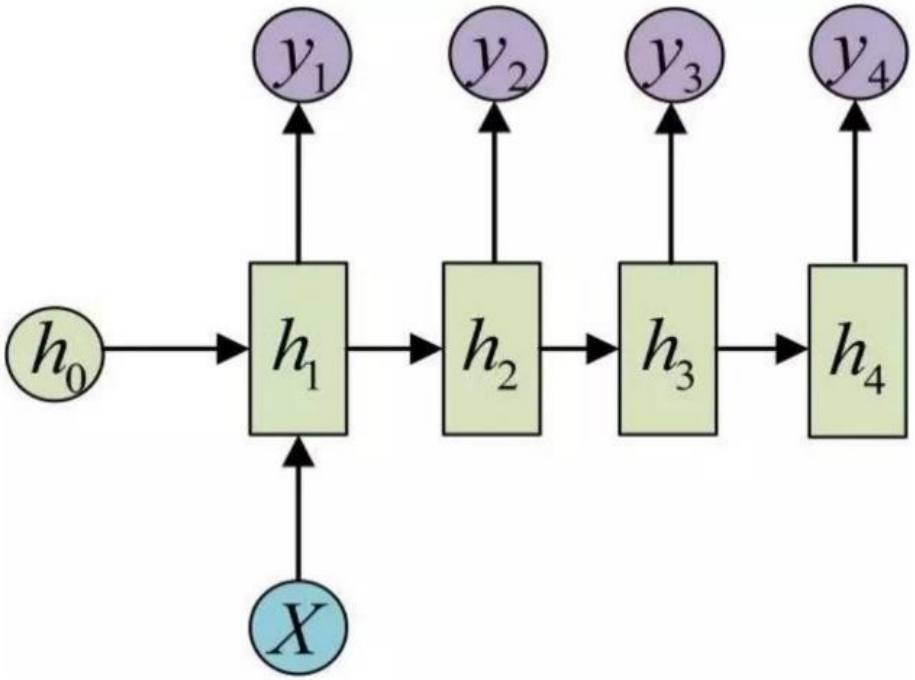
1、循环神经网络结构





N vs 1结构的RNN

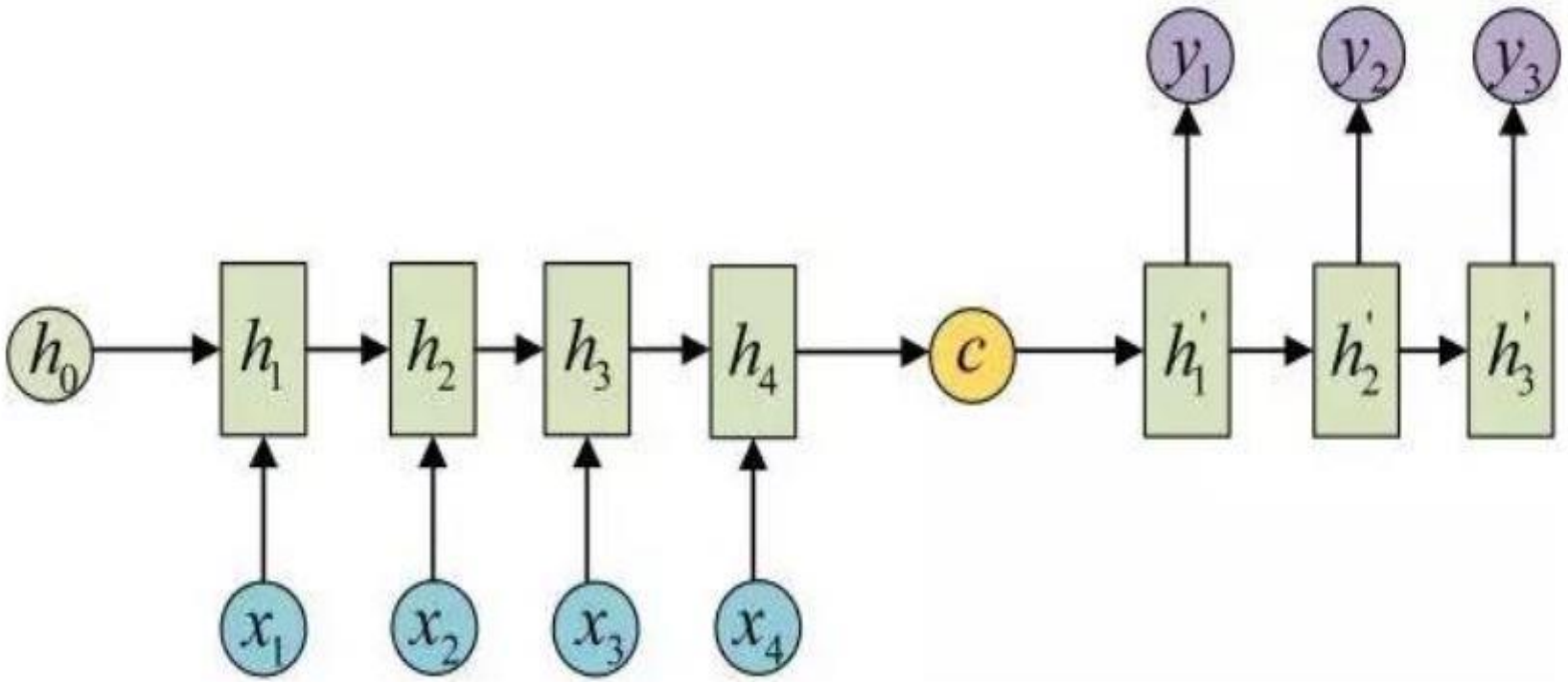
1、循环神经网络结构



1 vs N结构的RNN

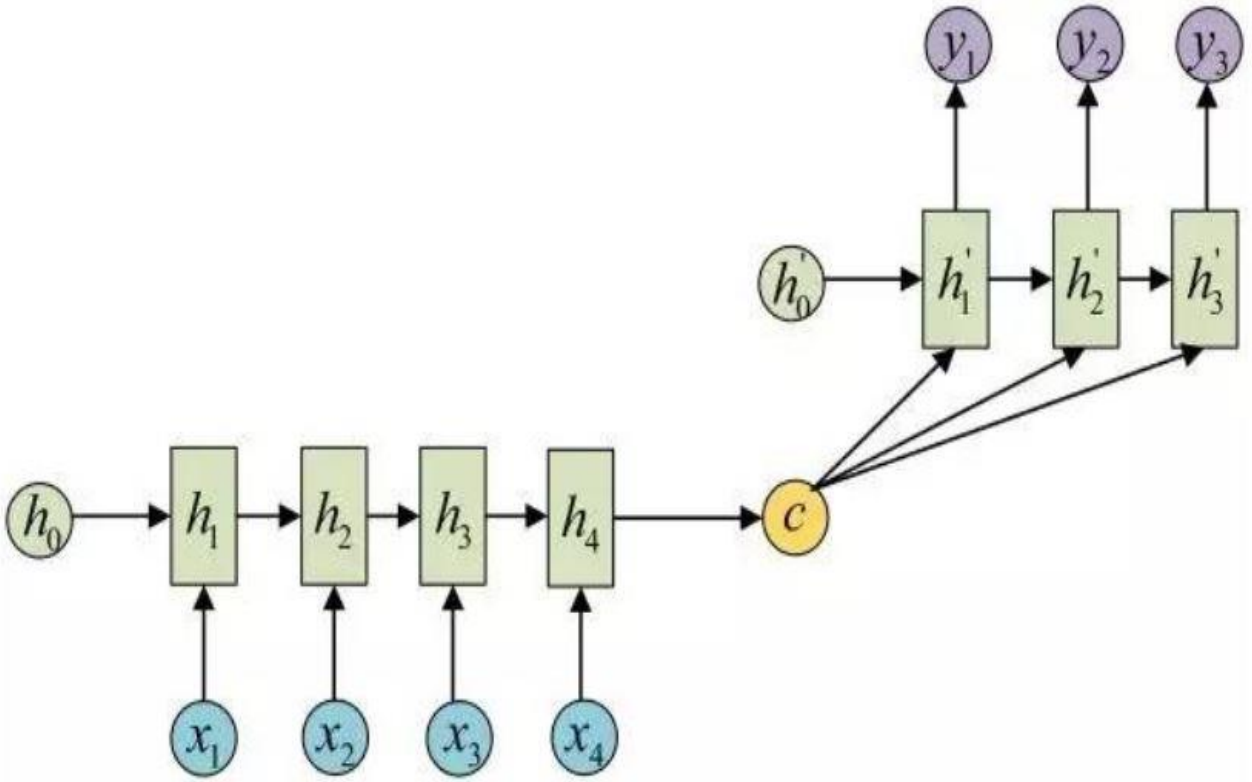
Encoder-Decoder模型

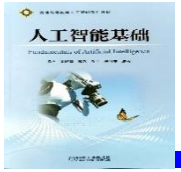
- 称为Seq2Seq模型，不要求输入和输出的序列等长度
- 将输入数据通过RNN网络编码成一个上下文向量c，随后用另一个RNN网络将c作为初始输入，从而得到结果序列
- N vs 1和1 vs M的串联
- 用于机器翻译、文本摘要、语音识别和阅读理解等问题



Encoder-Decoder模型

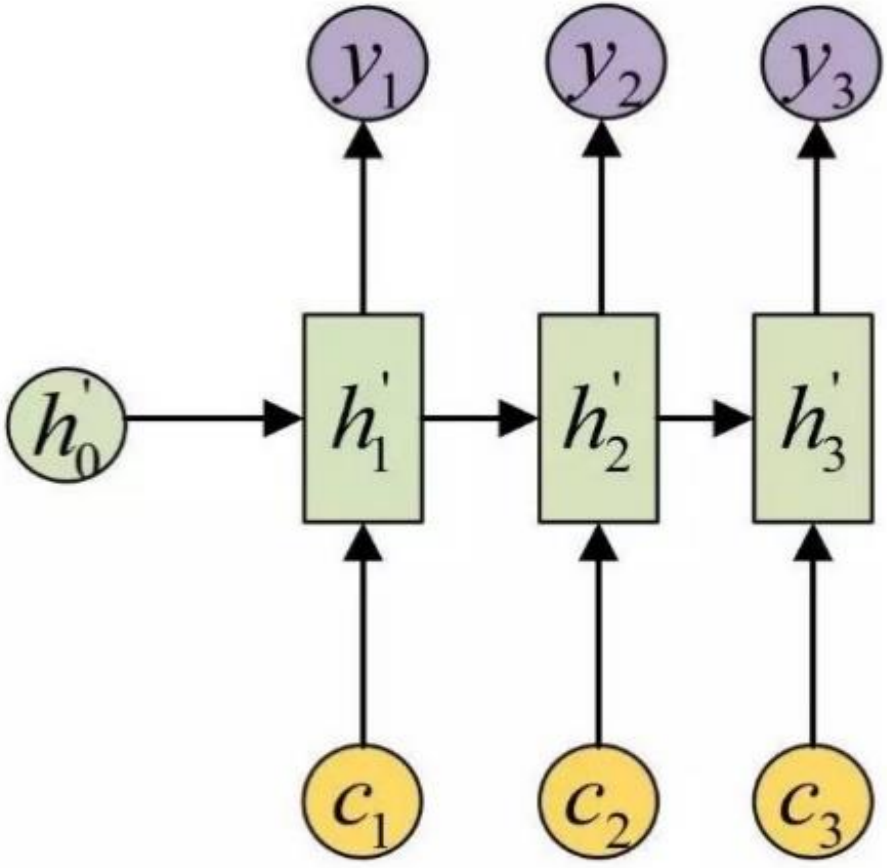
- 一个上下文向量c，在每个阶段输入N vs 1和1 vs M的串联

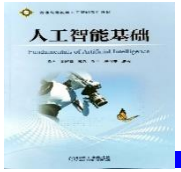




Attention机制

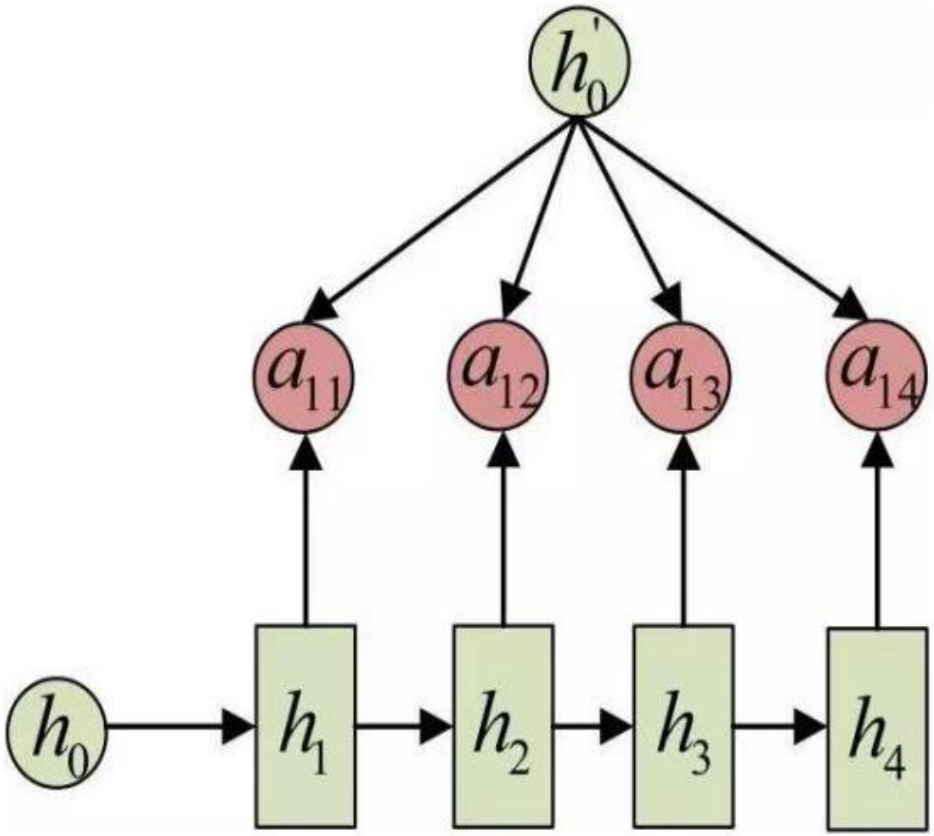
- Attention机制通过在每个时间输入不同的c来解决这个问题
- 每一个c都会自动去选取与当前所要输出的y最合适的上下文信息，相当于对所有信息进行了重要性的加权



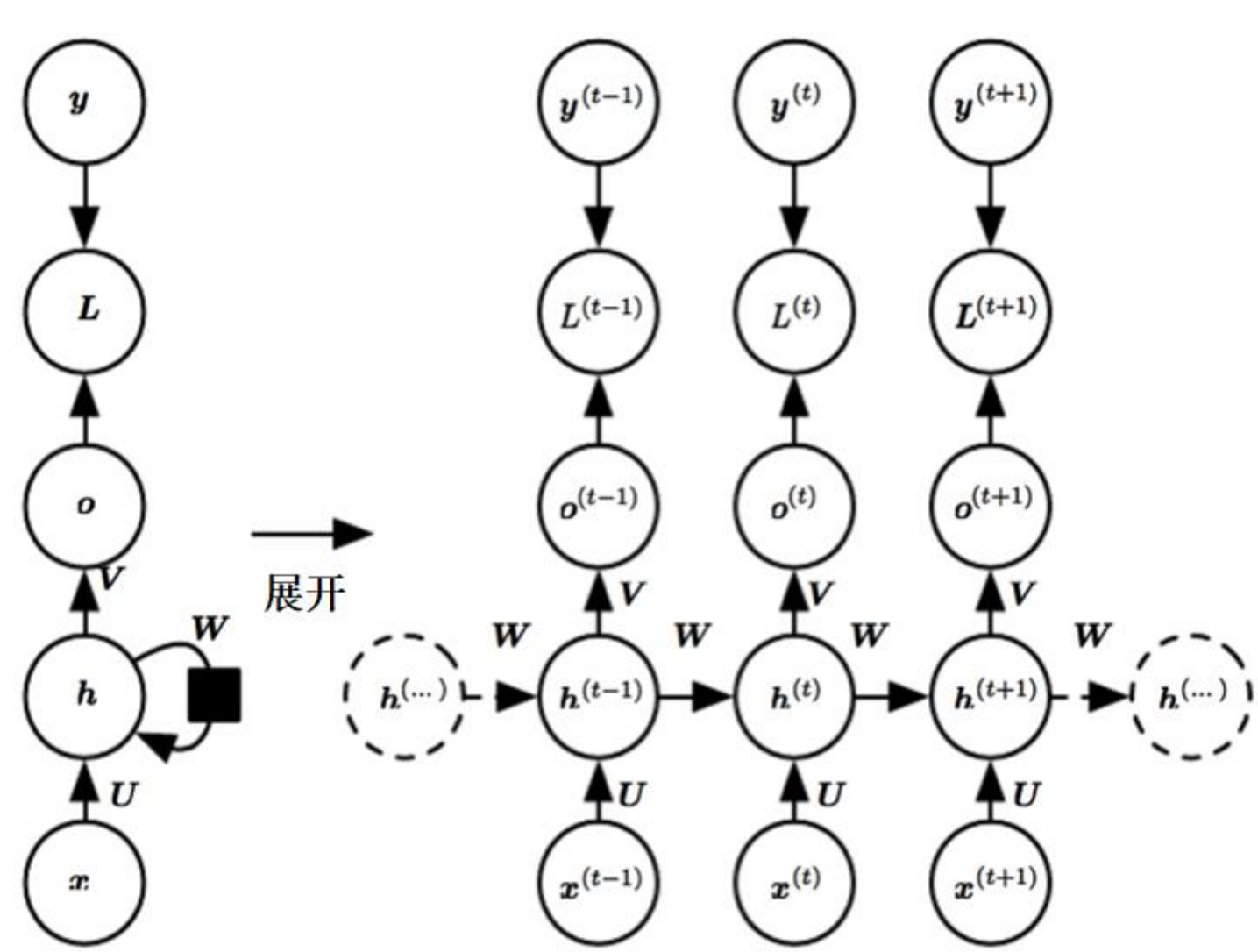


Attention机制

- a_{ij}衡量Encoder第j阶段的h_j和Decoder第i阶段的相关性
- Decoder中第i阶段输入的上下文信息c_i就来自于所有h_j对a_{ij}的加权和。



RNN的前向传播


y
L
o
V
W
h
U
x
展开
$y^{(t-1)}$
$y^{(t)}$
$y^{(t+1)}$
$L^{(t-1)}$
$L^{(t)}$
$L^{(t+1)}$
$o^{(t-1)}$
$o^{(t)}$
$o^{(t+1)}$
$h^{(\cdots)}$
$h^{(t-1)}$
$h^{(t)}$
$h^{(t+1)}$
$h^{(\cdots)}$
$x^{(t-1)}$
$x^{(t)}$
$x^{(t+1)}$

RNN的前向传播

- 前向传播的算法十分简单，在t时刻的隐层神经元有

$$h^{(t)} = \phi\left(Ux^{(t)} + Wh^{(t-1)} + b\right)$$

其中$\phi(\cdot)$是激活函数，可以选择tanh函数等，b为偏置。

- 在t时刻的输出为

$$o^{(t)} = Vh^{(t)} + c$$

c为偏置。

- 最终模型的预测输出为

$$\hat{y}^{(t)} = \sigma\left(o^{(t)}\right)$$

- 其中$\sigma(\cdot)$是激活函数。
- 在分类问题中，通常使用softmax函数作为最终的激活函数

RNN的逆向传播

- 逆向传播算法是BPTT(back-propagation through time)算法
- RNN的损失随时间累加，在t时刻求之前全部时刻的偏导和

$$\frac{\partial L^{(t)}}{\partial \mathrm{V}} = \sum_{t=1}^{n} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial V}$$

- 隐层激活函数的选择
  - ✓ 当隐层激活函数为tanh函数时，

$$\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} = \prod_{j=k+1}^{t} \tanh' \cdot W_s$$

  - ✓ 当隐层激活函数是sigmoid函数时，

$$\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} = \prod_{j=k+1}^{t} \mathrm{sigmoid}' \cdot W_s$$

双向RNN

- **Encoder**

$$\vec{h}_j = \begin{cases} 0 & \text{if } j = 0 \\ tanh\left(\vec{W}_x E_x x_j + \vec{U}_x h_{j-1}\right) & \text{if } j > 0 \end{cases}$$

$$\overleftarrow{h}_j = \begin{cases} 0 & \text{if } j = T_x + 1 \\ \tanh\left(\overleftarrow{W}_x E_x x_j + \overleftarrow{U}_x h_{j+1}\right) & \text{if } j \le T_x \end{cases}$$

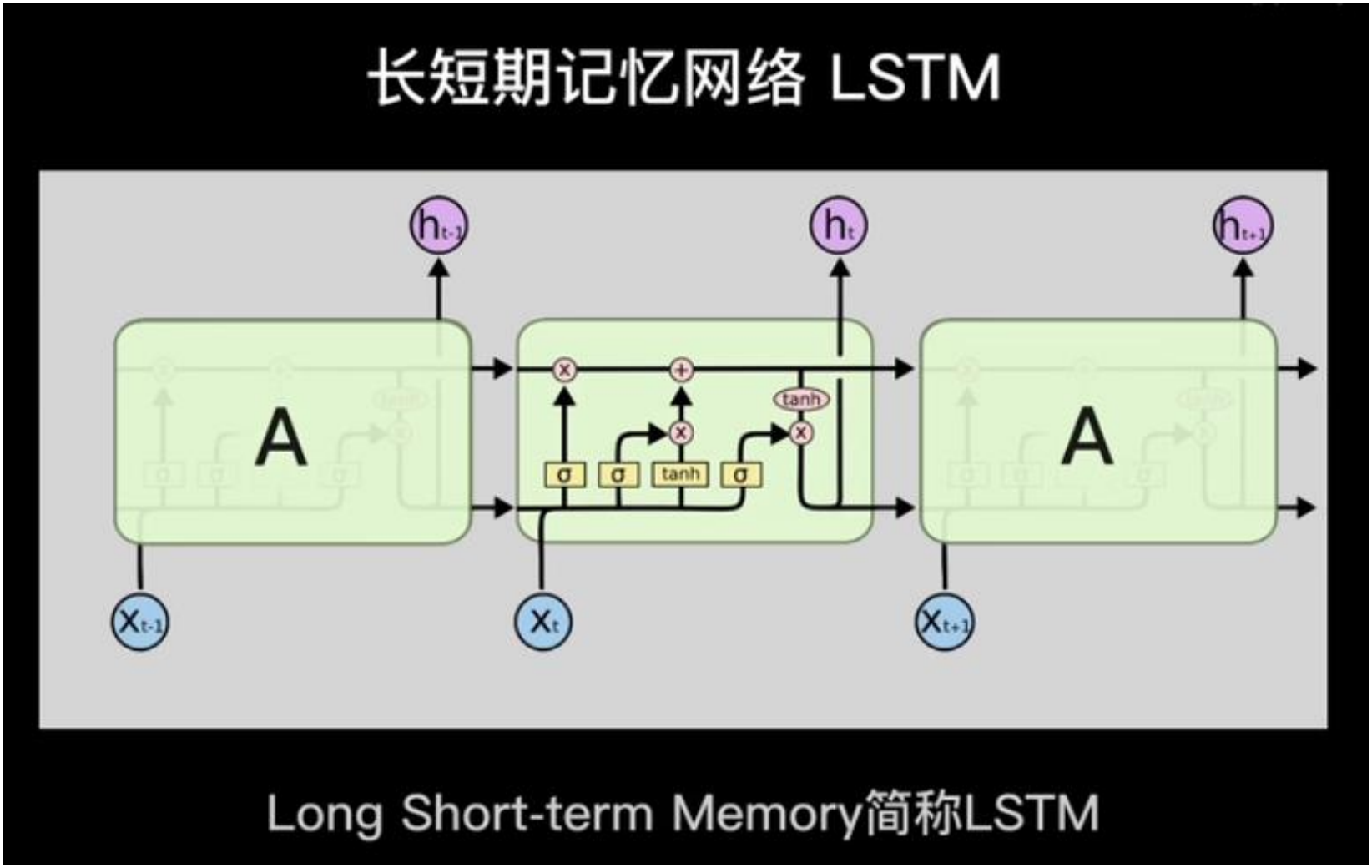
$$h_j = \left(\vec{h}_j, \overleftarrow{h}_j\right)$$

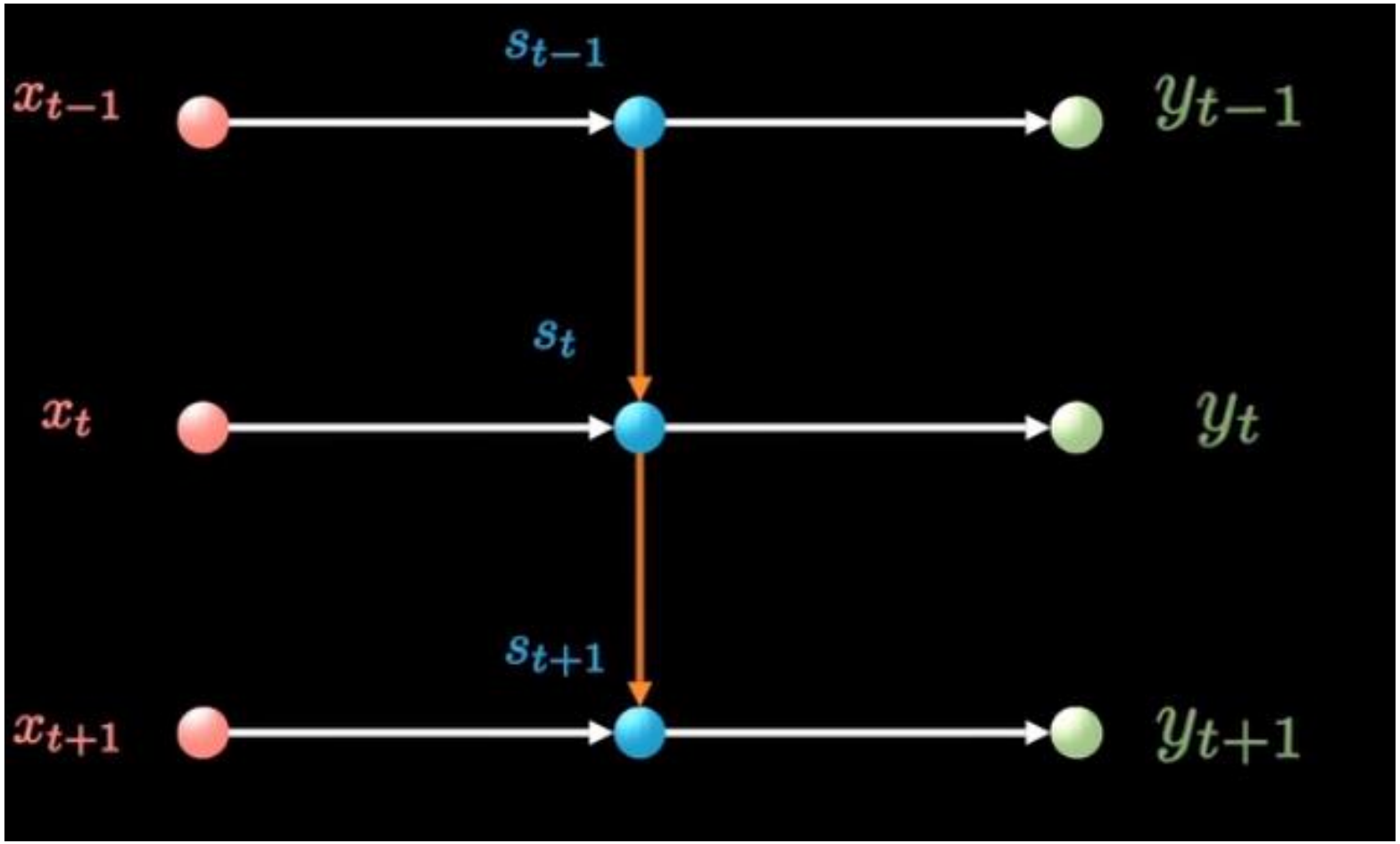
- **Attention**

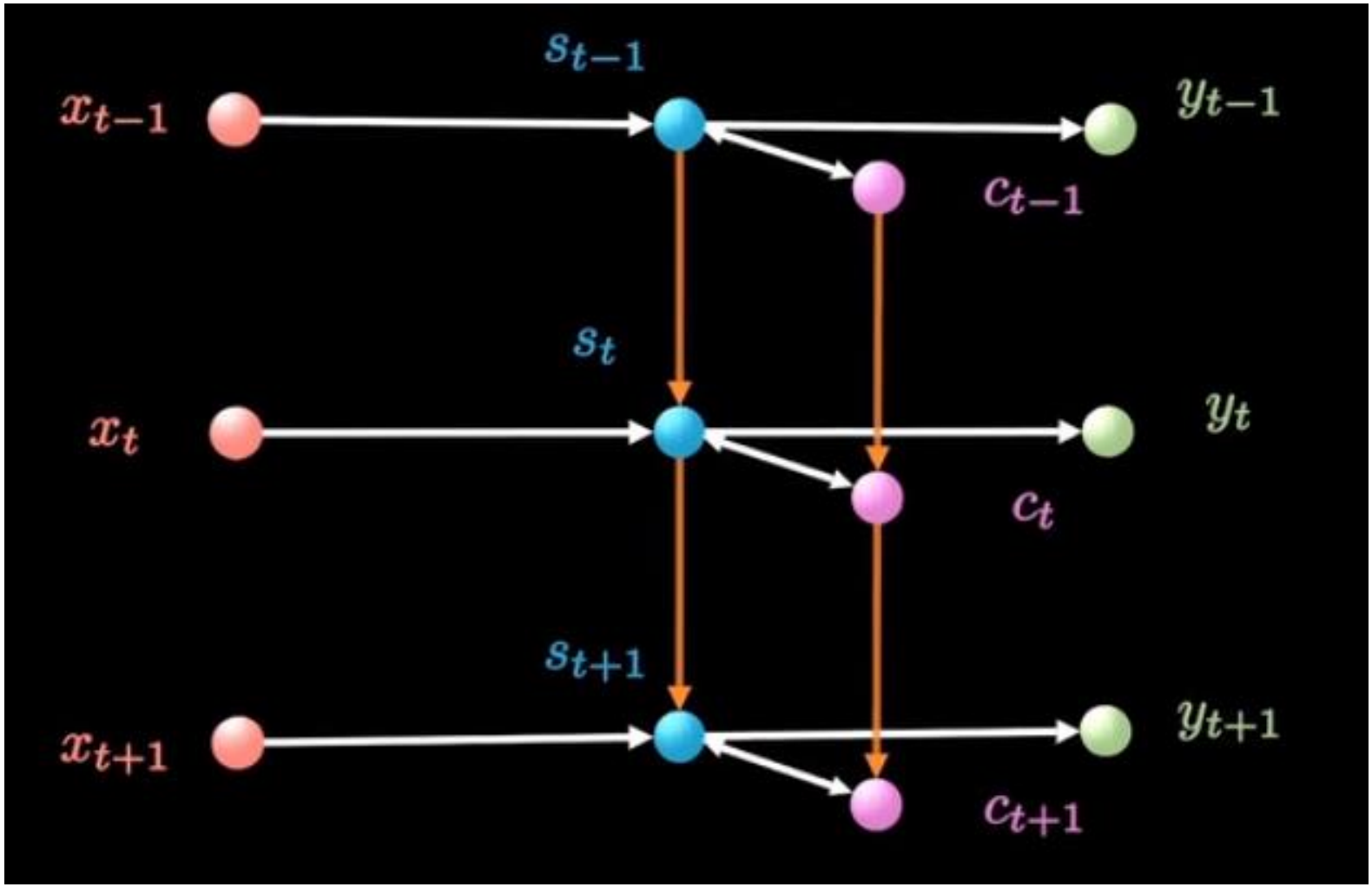
$$e_{ij} = v_a^{\mathrm{T}} \tanh\left(W_a s_{i-1} + U_a h_j\right)$$
$$\alpha_{ij} = \mathrm{softmax}\left(e_{ij}\right)$$
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

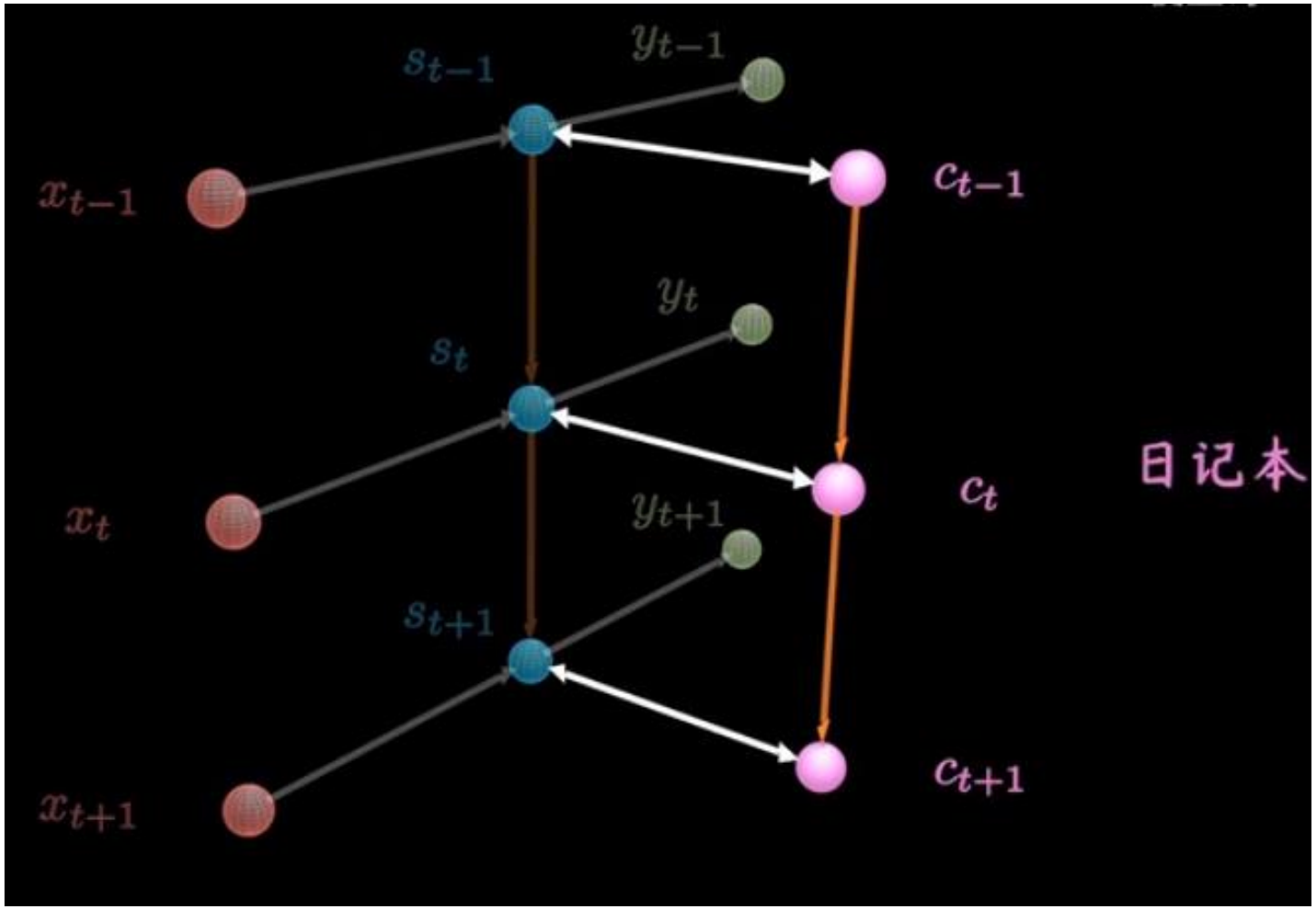
- **Decoder：单向的RNN网络，有3个输入，输出为概率向量**

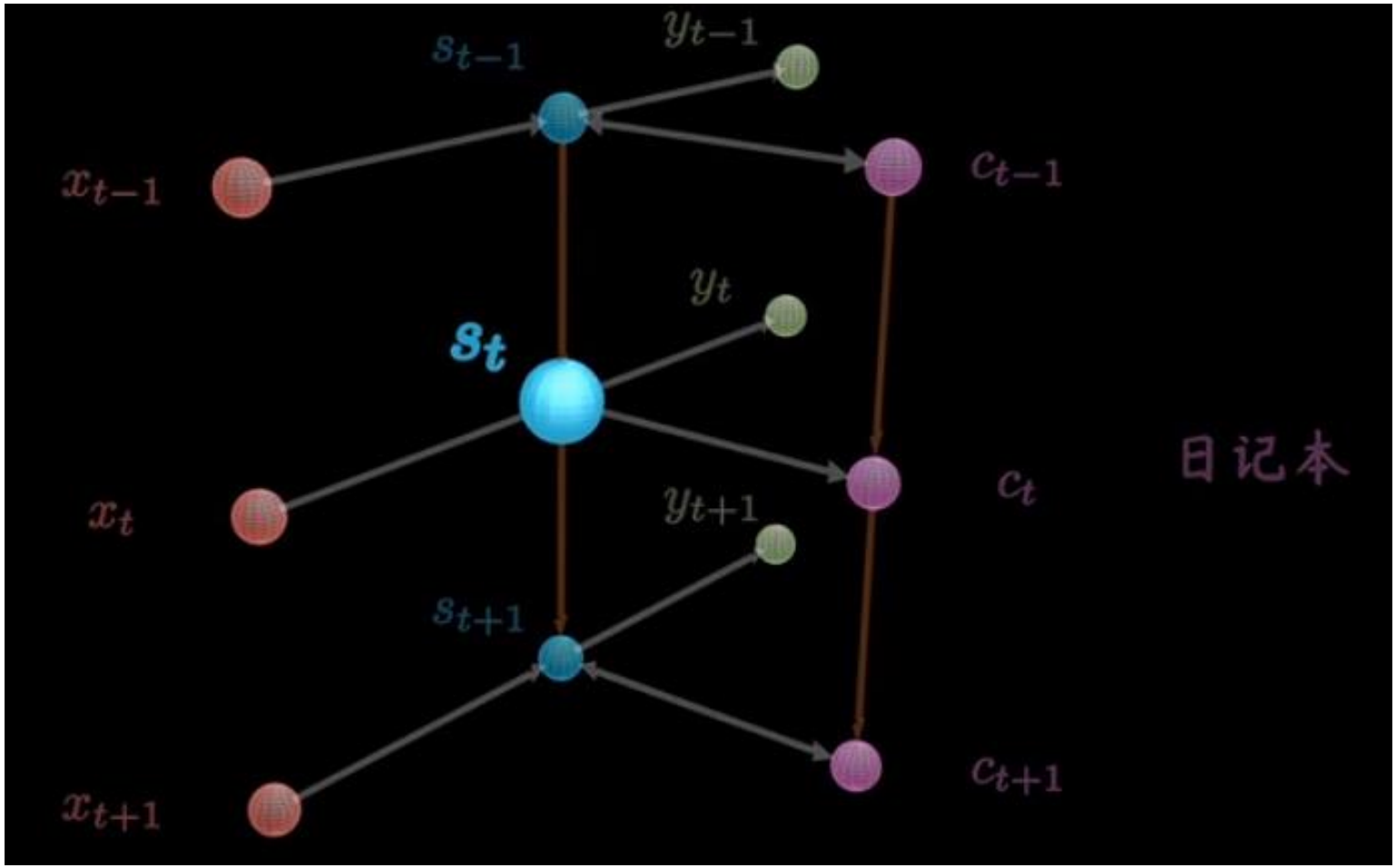
长短期记忆网络

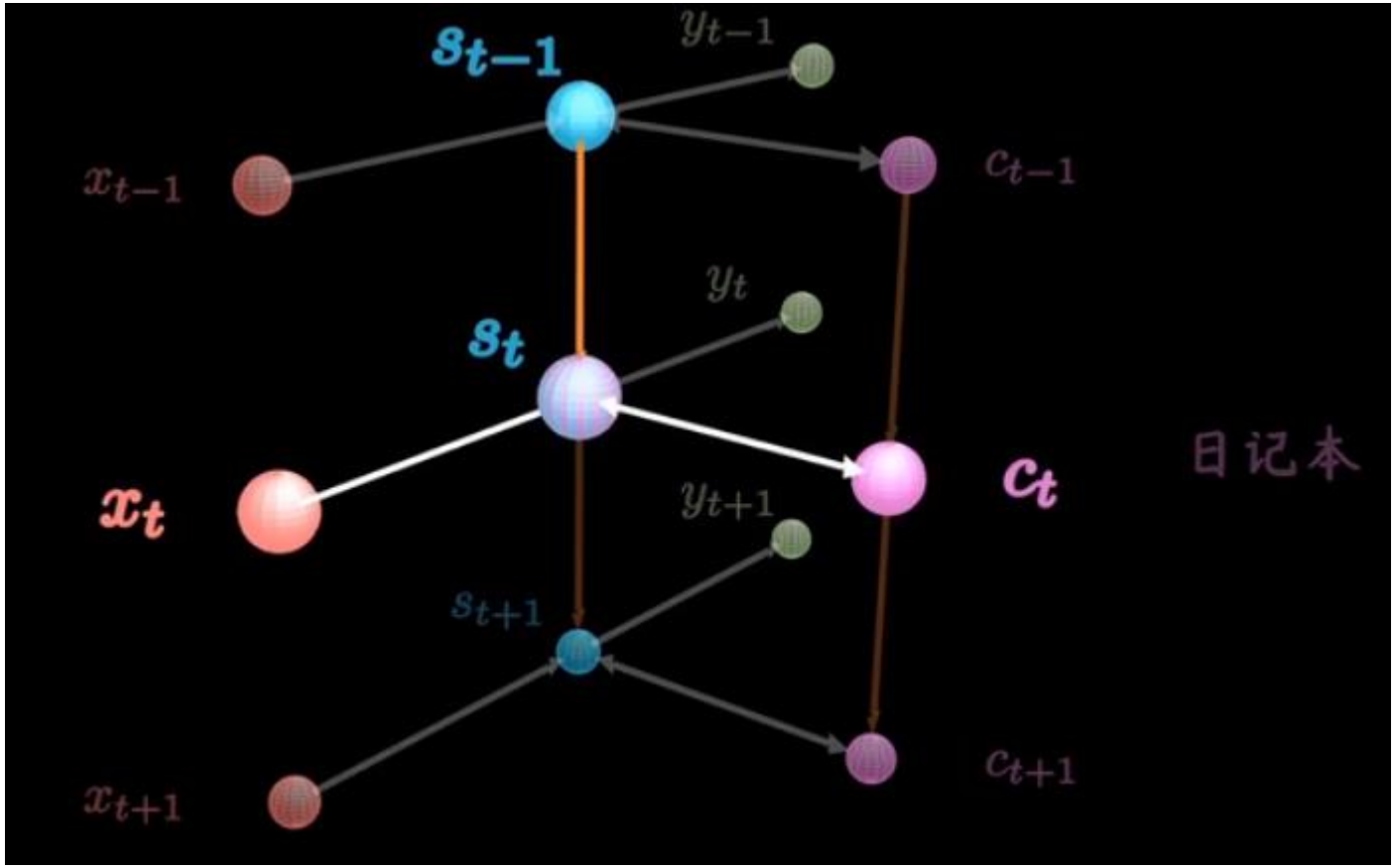


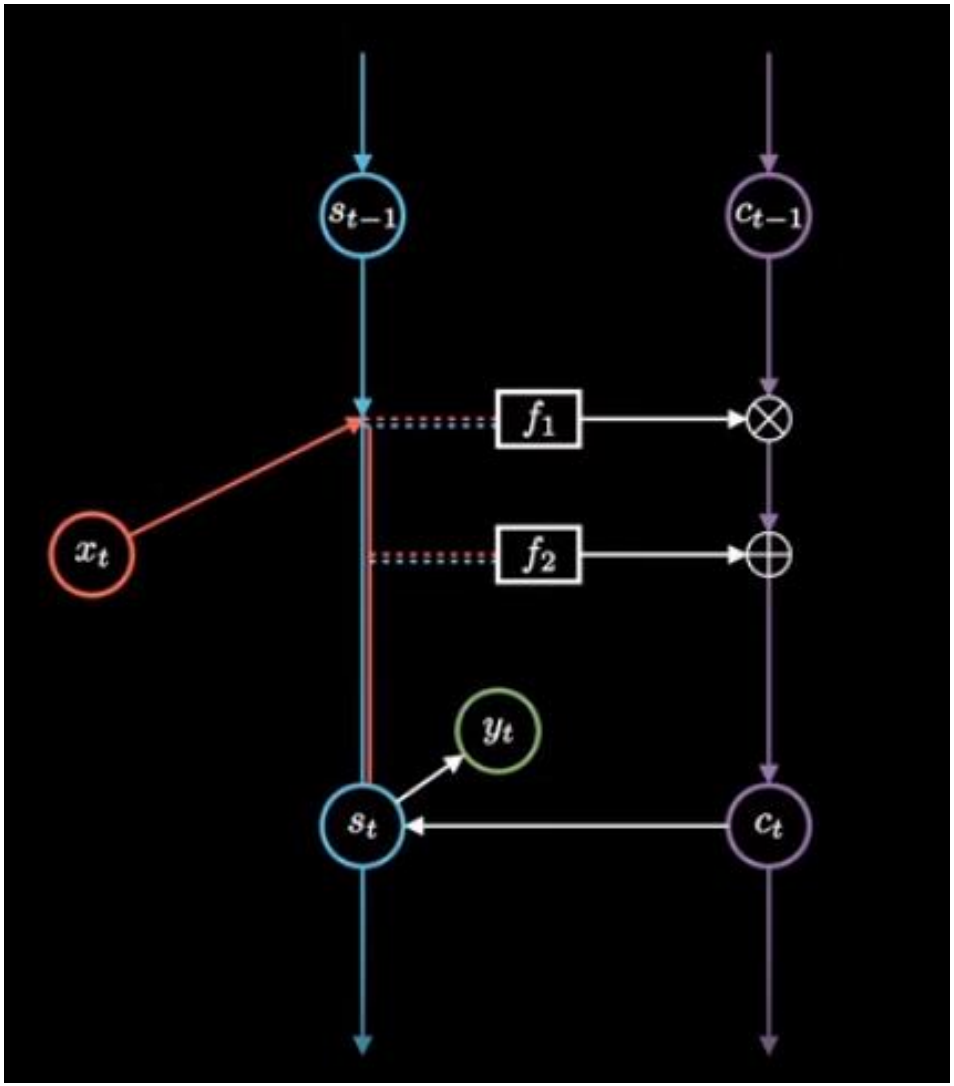

s_{t-1}
x_{t-1}
y_{t-1}
s_t
x_t
y_t
s_{t+1}
x_{t+1}
y_{t+1}

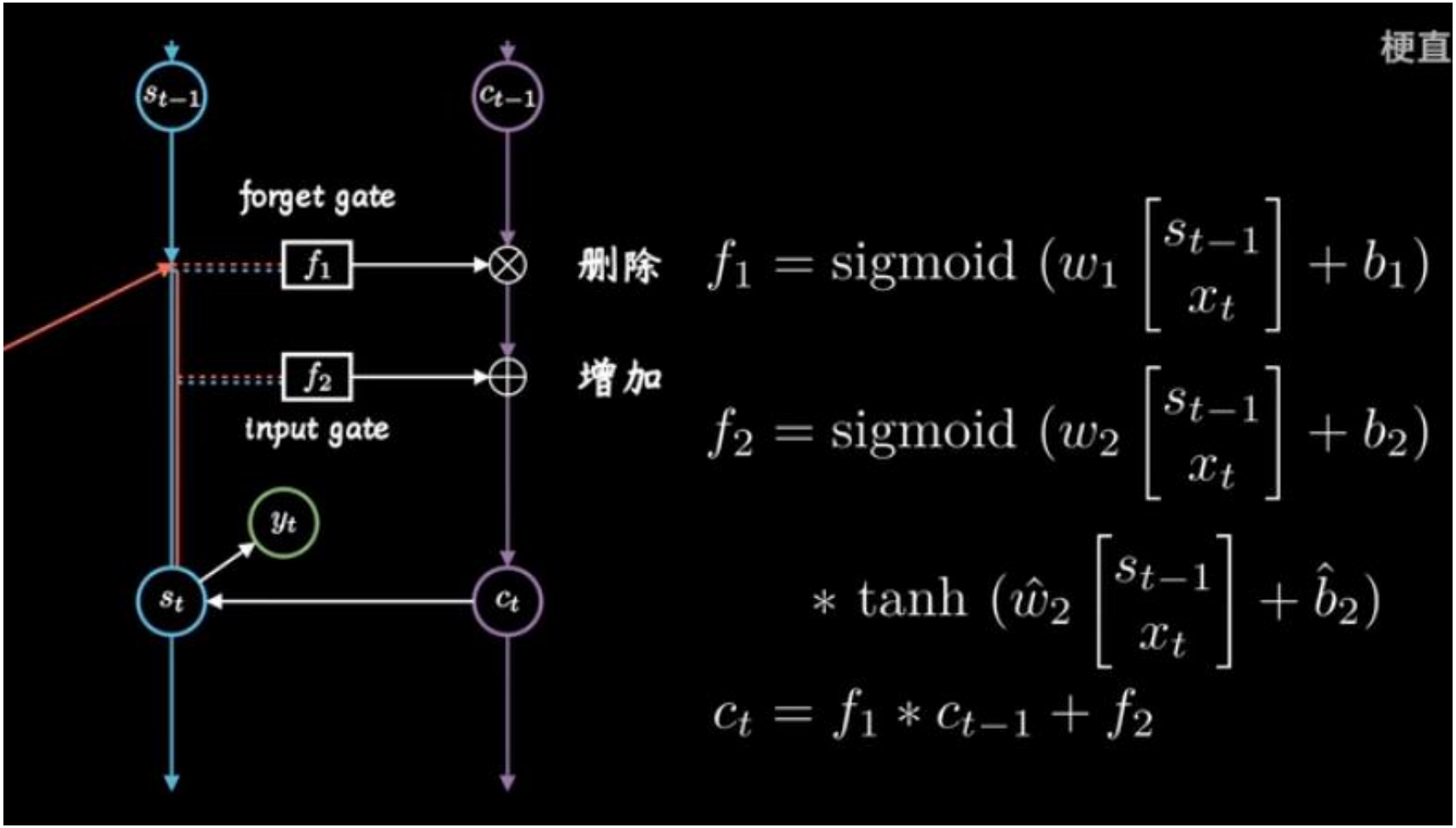

s_{t-1}
x_{t-1}
y_{t-1}
c_{t-1}
s_t
x_t
y_t
c_t
s_{t+1}
x_{t+1}
y_{t+1}
c_{t+1}

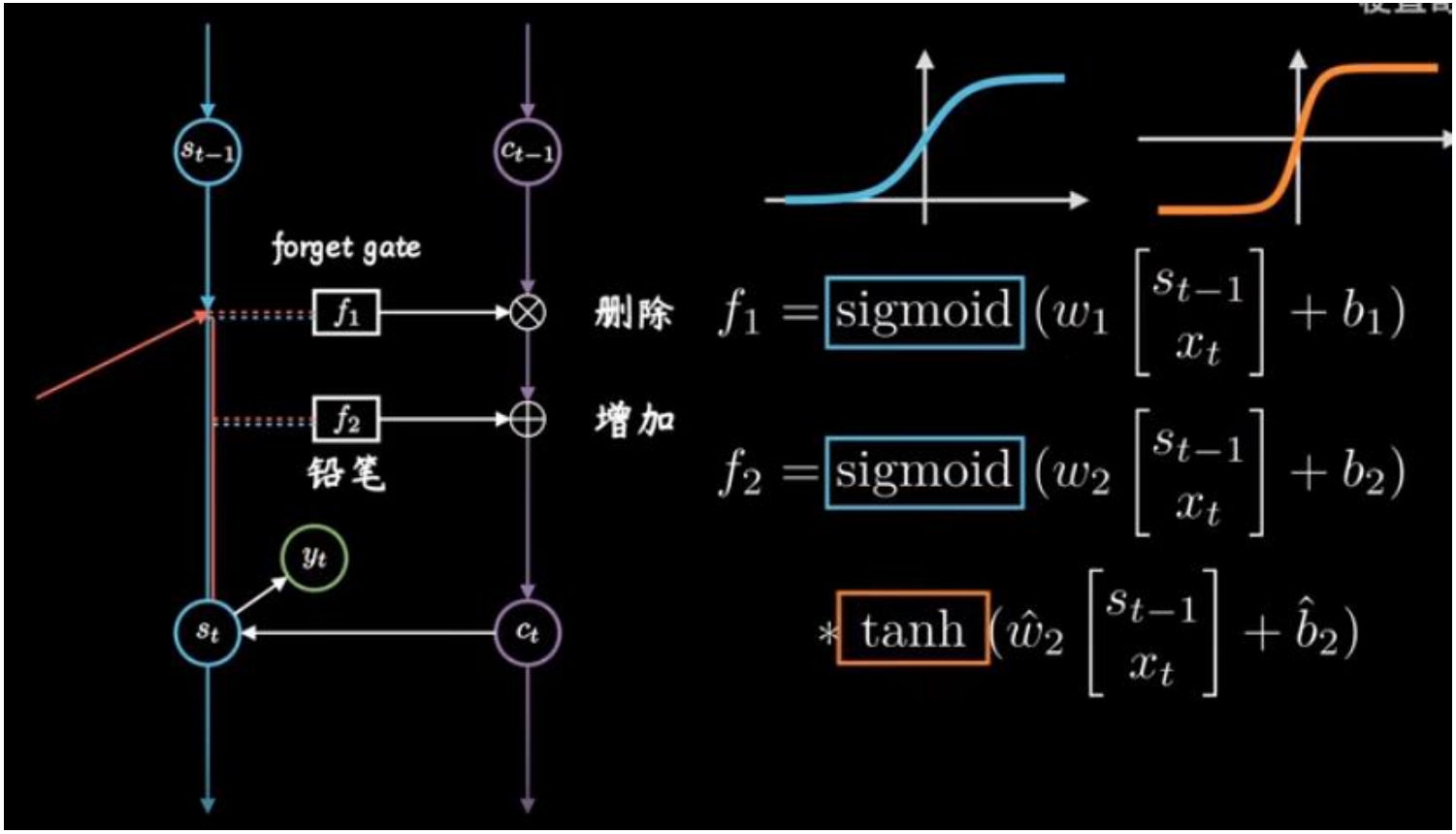

s_{t-1}
y_{t-1}
x_{t-1}
c_{t-1}
y_t
s_t
x_t
c_t
日记本
y_{t+1}
s_{t+1}
x_{t+1}
c_{t+1}

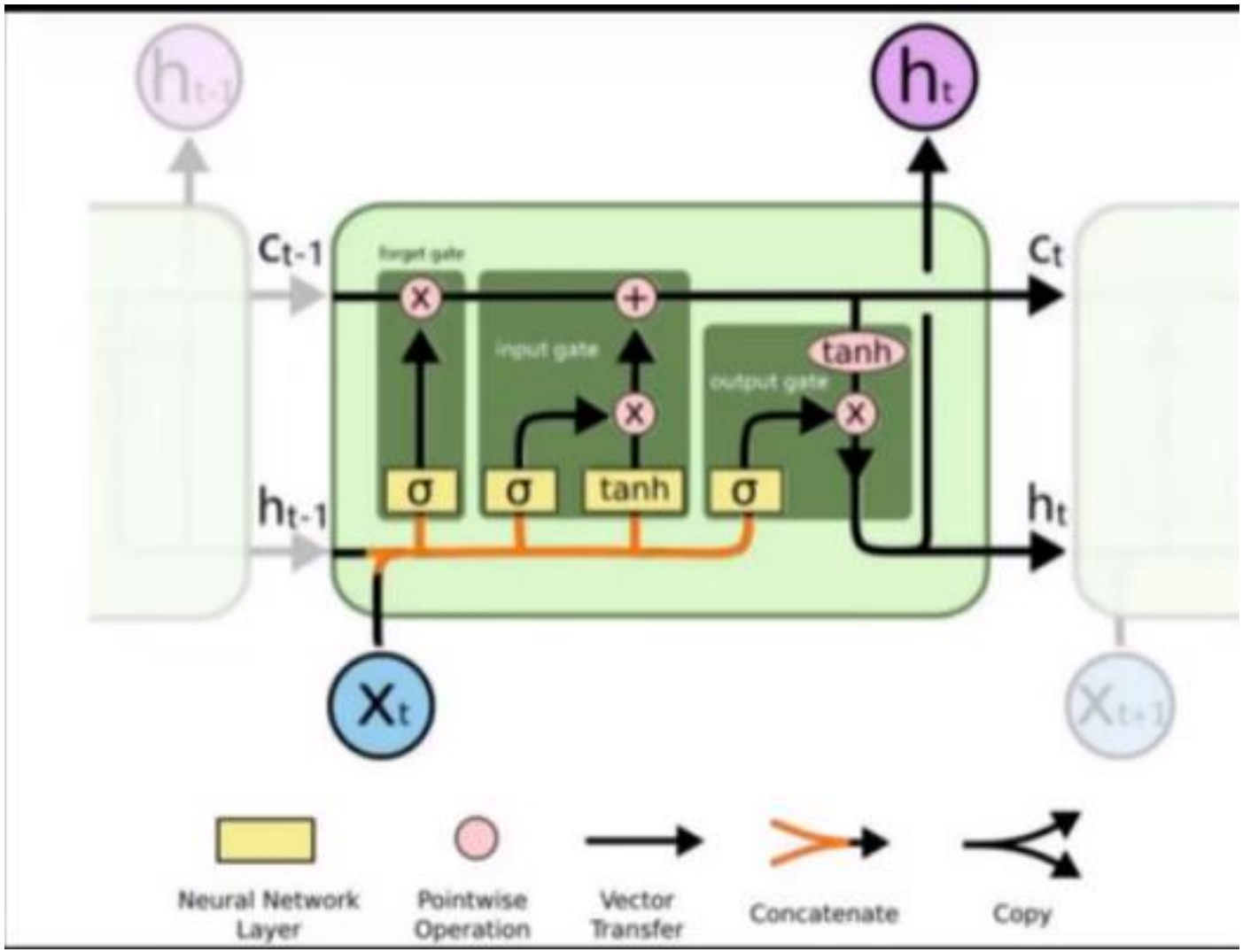

y_{t-1}
s_{t-1}
x_{t-1}
c_{t-1}
y_t
s_t
x_t
c_t
日记本
y_{t+1}
s_{t+1}
x_{t+1}
c_{t+1}

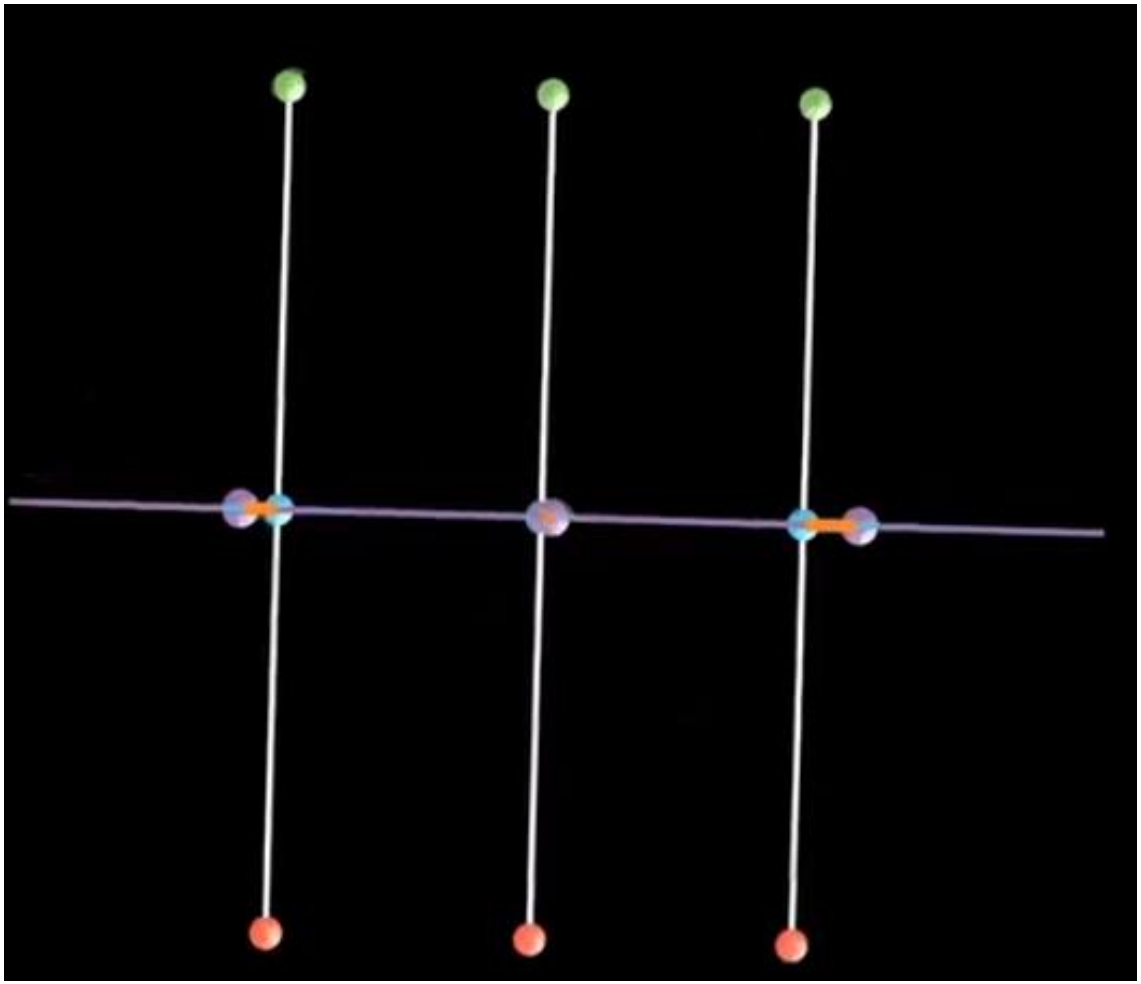

s_{t-1}
y_{t-1}
x_{t-1}
c_{t-1}
y_t
s_t
x_t
c_t
日记本
y_{t+1}
s_{t+1}
x_{t+1}
c_{t+1}

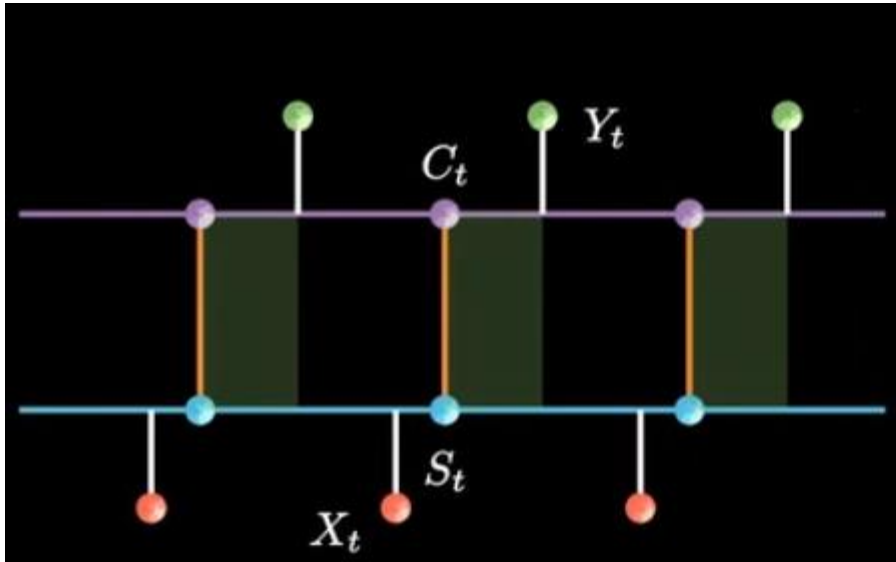

s_{t-1}
c_{t-1}
f_1
x_t
f_2
y_t
s_t
c_t

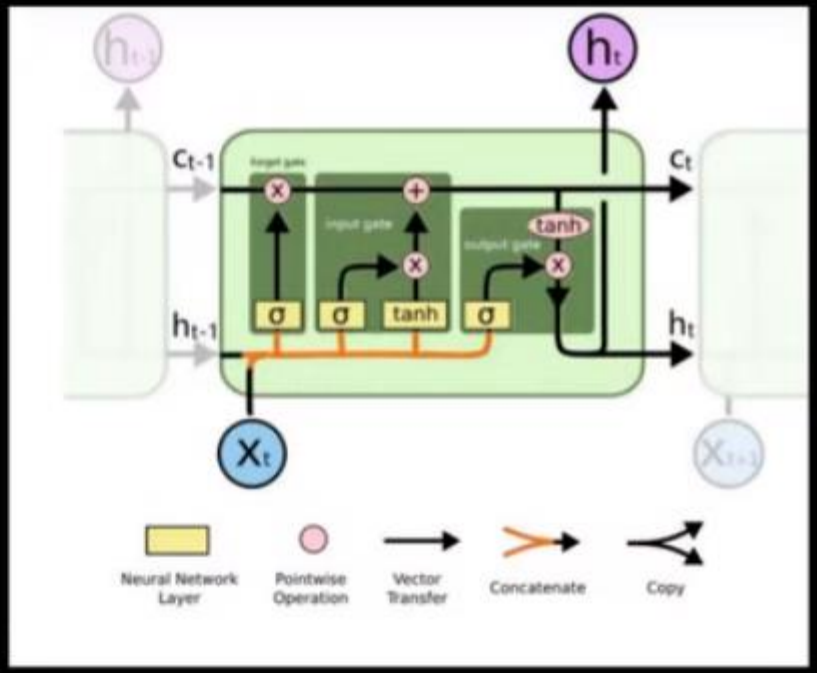

梗直
s_{t-1}
c_{t-1}
forget gate
f_1
删除
f_2
增加
input gate
y_t
s_t
c_t
$f_1 = \text{sigmoid}\ (w_1 \begin{bmatrix} s_{t-1} \\ x_t \end{bmatrix} + b_1)$
$f_2 = \text{sigmoid}\ (w_2 \begin{bmatrix} s_{t-1} \\ x_t \end{bmatrix} + b_2)$
$* \tanh\ (\hat{w}_2 \begin{bmatrix} s_{t-1} \\ x_t \end{bmatrix} + \hat{b}_2)$
$c_t = f_1 * c_{t-1} + f_2$


s_{t-1}
c_{t-1}
forget gate
f_1
f_2
铅笔
y_t
s_t
c_t
删除
增加
$f_1 = \boxed{\text{sigmoid}}\left(w_1 \begin{bmatrix} s_{t-1} \\ x_t \end{bmatrix} + b_1\right)$
$f_2 = \boxed{\text{sigmoid}}\left(w_2 \begin{bmatrix} s_{t-1} \\ x_t \end{bmatrix} + b_2\right)$
$* \boxed{\tanh}\left(\hat{w}_2 \begin{bmatrix} s_{t-1} \\ x_t \end{bmatrix} + \hat{b}_2\right)$

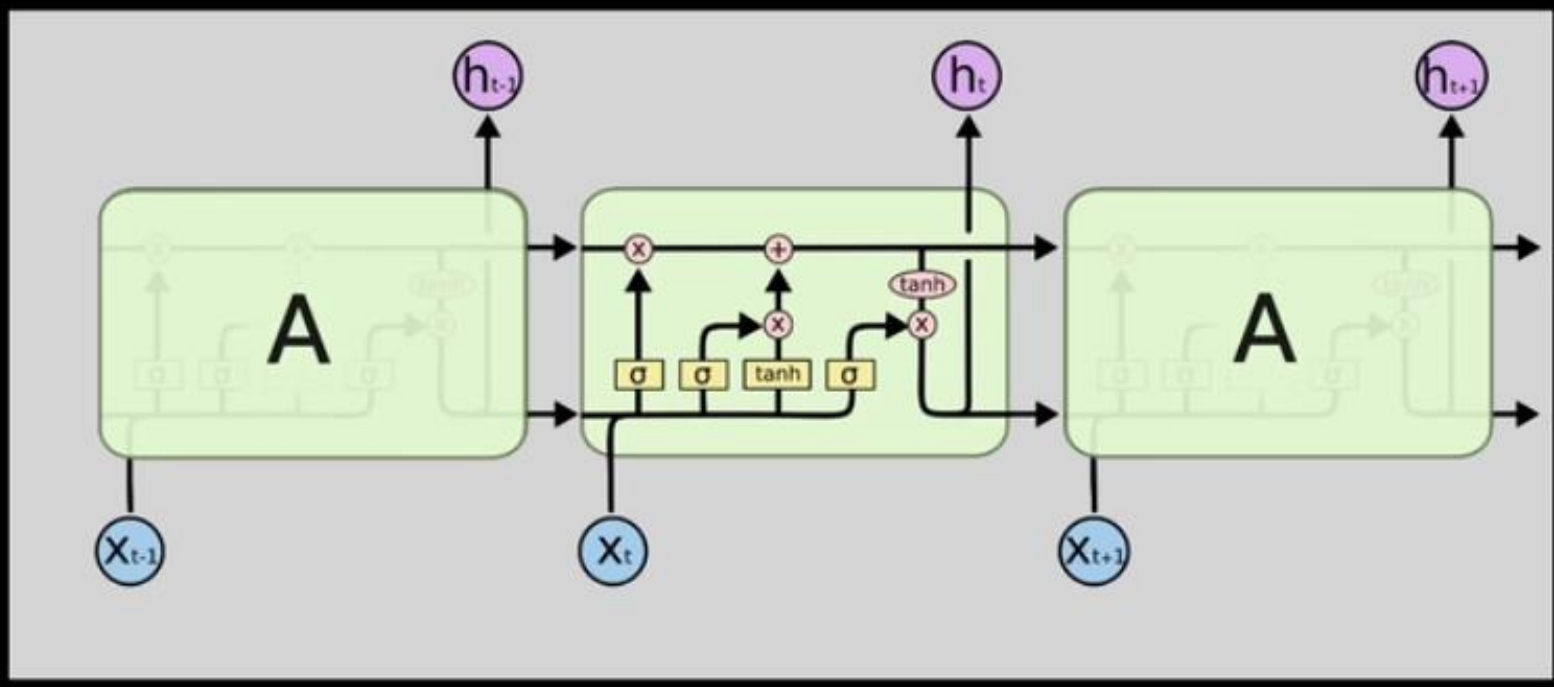

h_{t-1}
h_t
C_{t-1}
C_t
forget gate
input gate
output gate
tanh
σ
σ
tanh
σ
h_{t-1}
h_t
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy




C_t
Y_t
S_t
X_t


h_{t-1}
h_t
C_{t-1}
C_t
forget gate
input gate
output gate
tanh
σ
σ
tanh
σ
h_{t-1}
h_t
X_t
X_{t-1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

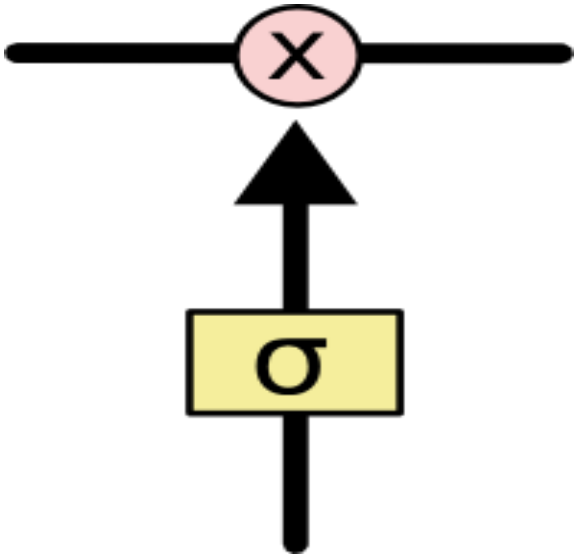
长短期记忆网络 LSTM

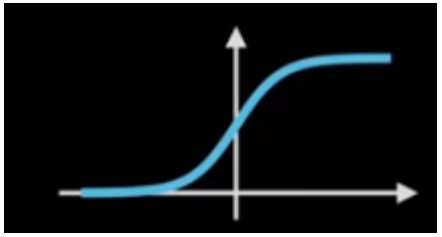


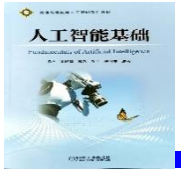
Long Short-term Memory简称LSTM

LSTM门结构

0 代表“不许任何信息通过”，1 就指“允许所有信息通过”，0.5表示“信息可以通过，但只有原来权重的一半”



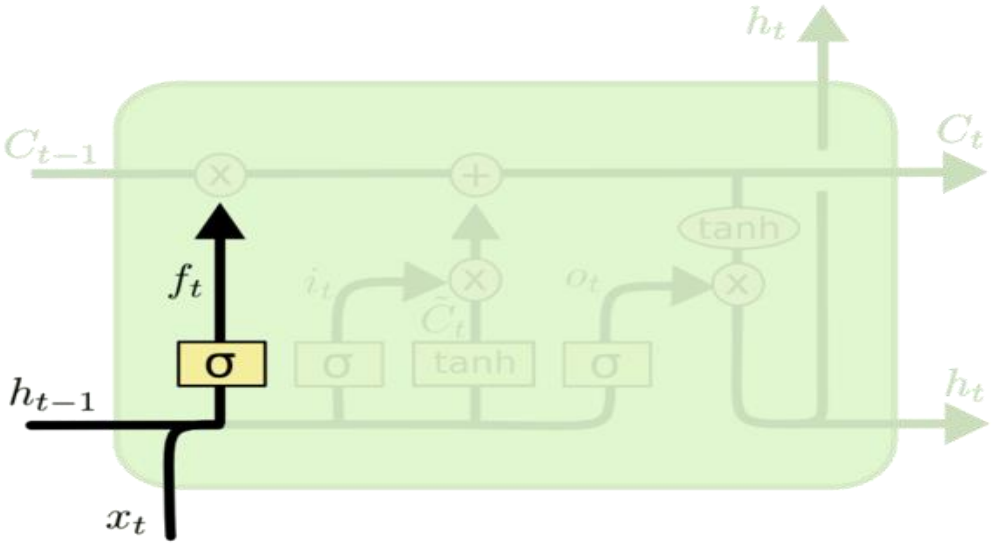


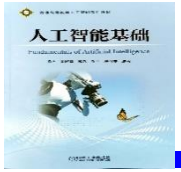


遗忘门

- 遗忘门：
  - ✓ 将细胞状态中的信息选择性的遗忘
- 遗忘权重系数 f_t
  - ✓ 读取 h_{t-1}和 x_t，输出一个在0到1之间的遗忘权重系数 f_t
  - ✓ 提供前一个细胞状态C_{t-1}在本细胞中继续传输的权重系数
  - ✓ 0表示 C_{t-1} 完全放弃（遗忘），1表示 C_{t-1} 完全保留（记忆）

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





输入门

- 输入门：
  - ✓ 将新的信息选择性的记录到细胞状态中
- 输入权重系数 i_t：决定信息的输入权重系数

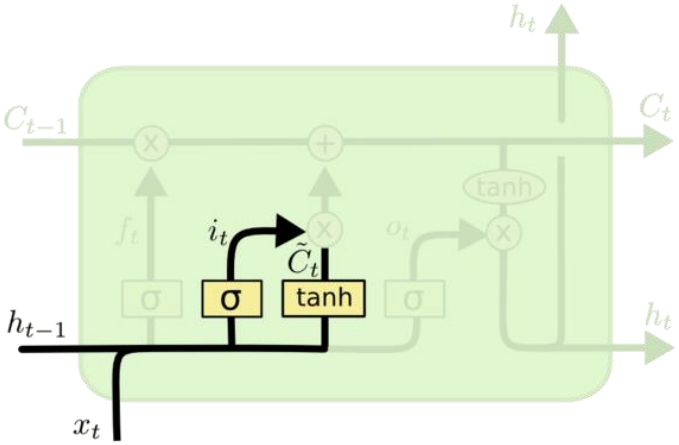
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

- 输入信息来源：需要加入的新信息来源

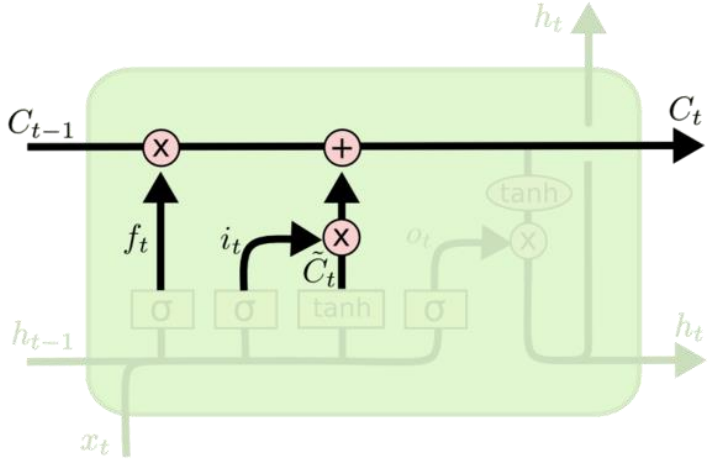
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- 细胞状态C_t的更新：由记忆信息和输入信息组成

$$C_t = f_t \times C_{t-1} + i_t \times \check{C}_t$$



输入门结构图



更新细胞状态

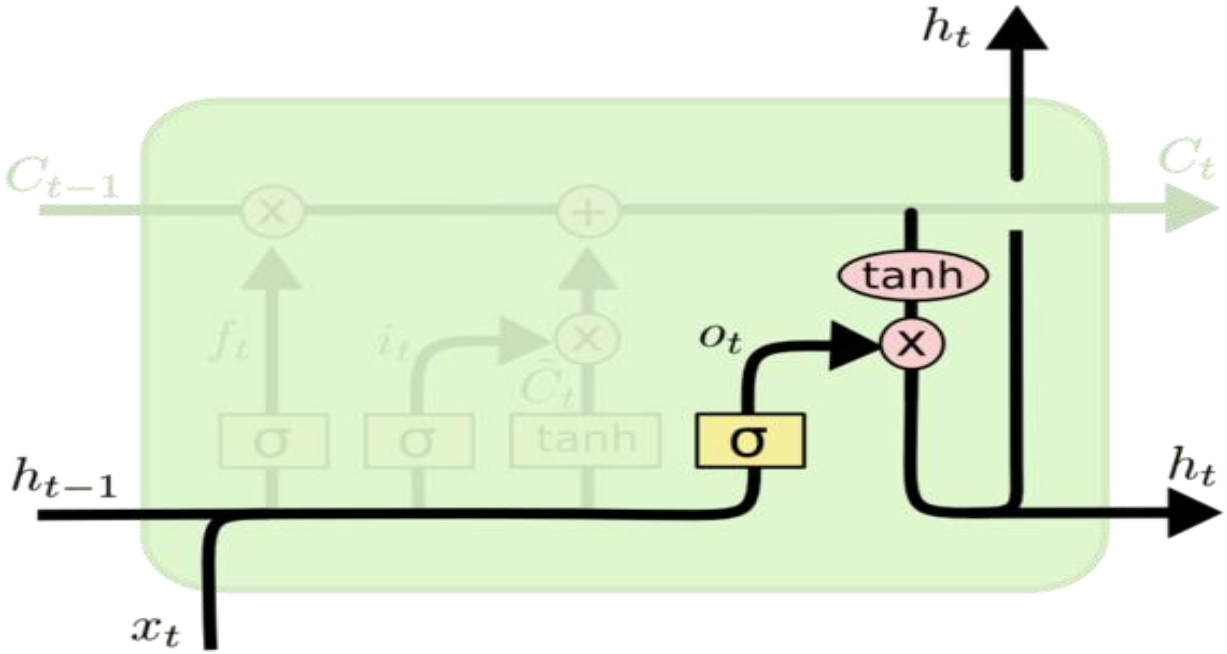
输出门

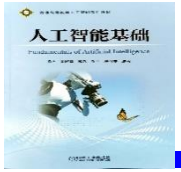
- 输出门：
  - ✓ 将更新细胞状态C_t选择性的记录到隐层输出h_t中
- 输出权重系数 o_t：更新细胞状态C_t的输出权重系数

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

- 输出信息：更新细胞状态C_t通过tanh函，和输出权重o_t相乘

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

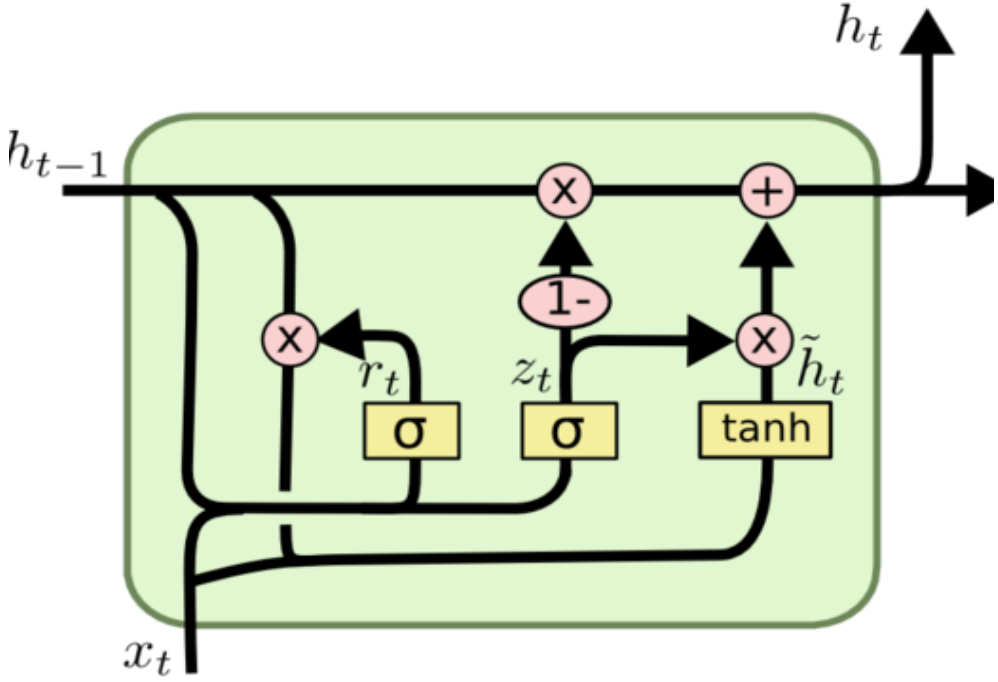




Gated Recurrent Unit

- Gated Recurrent Unit（GRU）：
  - ✓ LSTM的变体，通过一个更新门来控制细胞状态

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



Peephole LSTM

- **Peephole LSTM：**
  - ✓ **允许当前时刻的门Gate也接受前一时刻细胞状态C_{t-1}**

$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o)$$