未|来|教|育|高|精|尖|创|新|中|心
Advanced Innovation Center for Future Education
AICFE

深度学习入门及实战

刘杰飞

aic-fe.bnu.edu.cn

目录

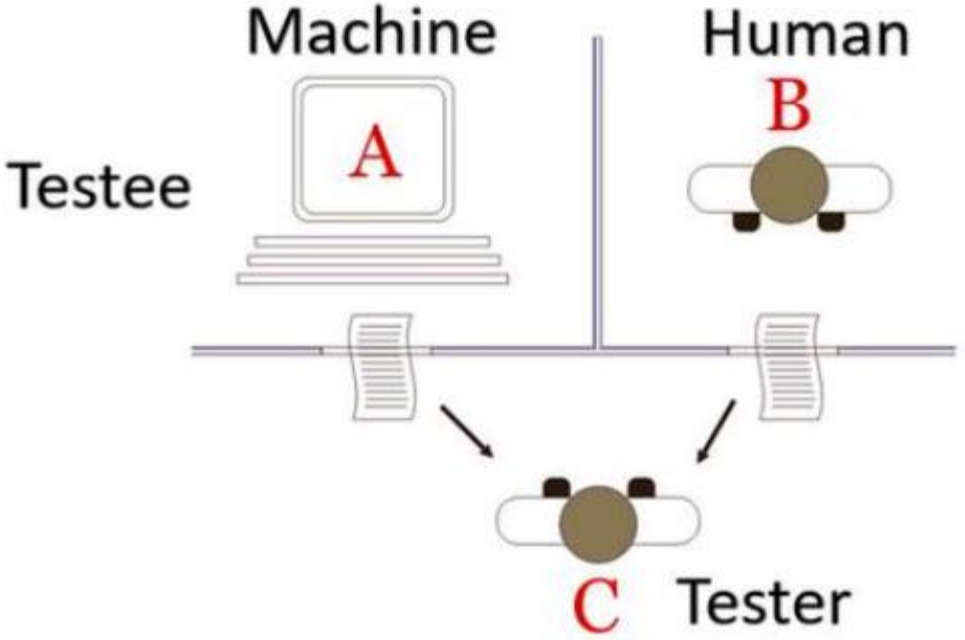
- 深度学习基础
- 典型模型介绍
- 深度学习框架
- 计算机视觉项目实战
- 自然语言处理项目实战

什么是深度学习

Definition of Deep Learning

人工智能与深度学习


人工智能
能够感知、推理、行动和适应的程序
机器学习
能够随着数据量的增加不断改进性能的算法
深度学习
机器学习的一个子集：利用多层神经网络从大量数据中进行学习


Machine
Human
B
Testee
A
C
Tester

机器学习与深度学习

- 语音识别

  $f($  $)=$ “你好”

- 图像识别

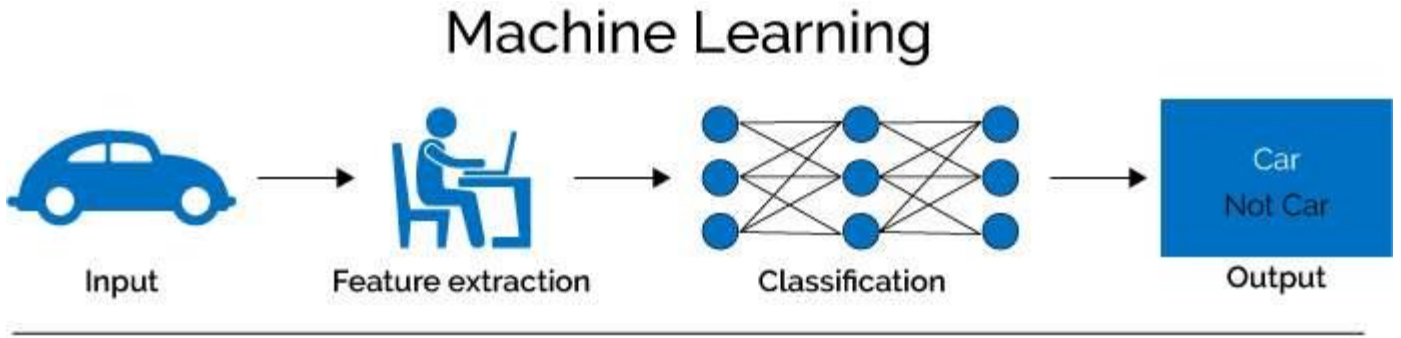
  $f($  $)=$ “9”

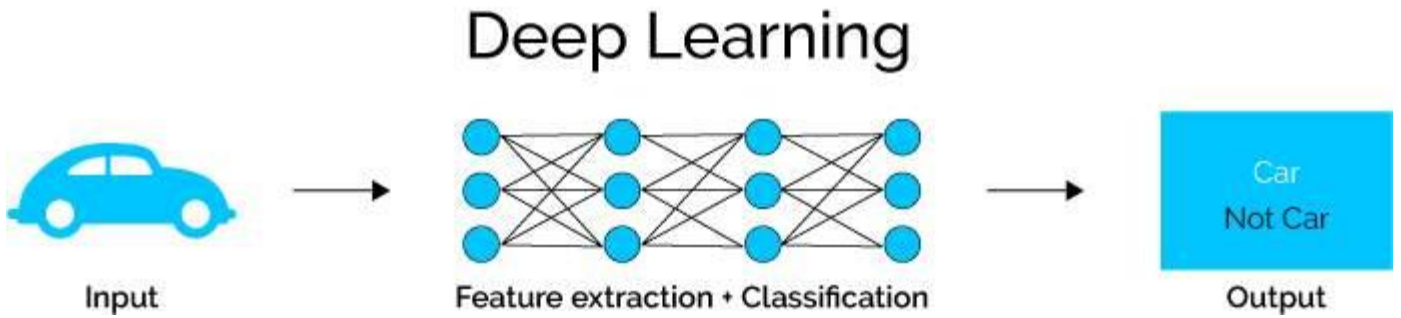
- 围棋

  $f($  $)=$ “6-5” （落子位置）

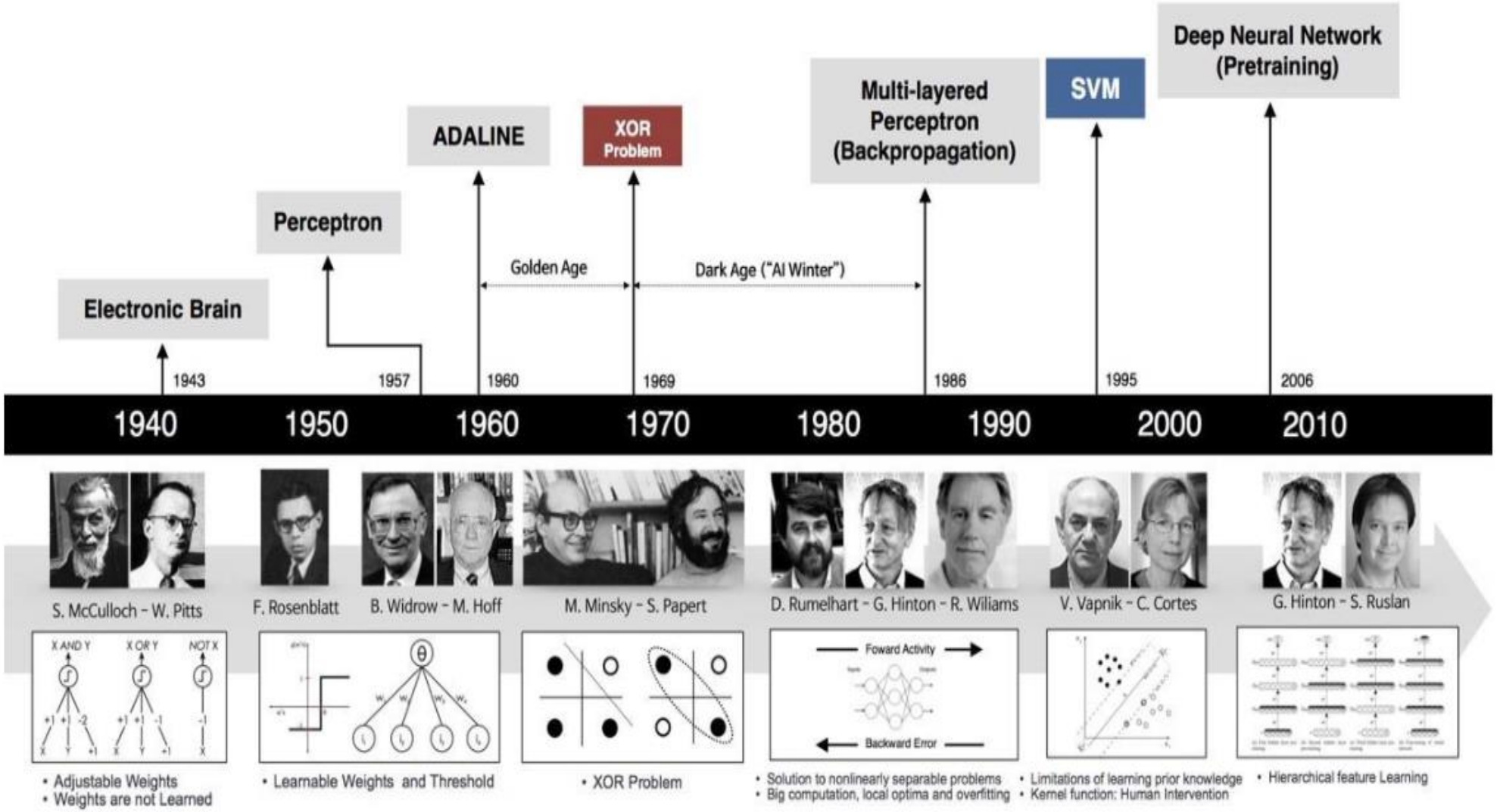
- 机器翻译

  $f($ “你好！” $)=$ “Hello!”



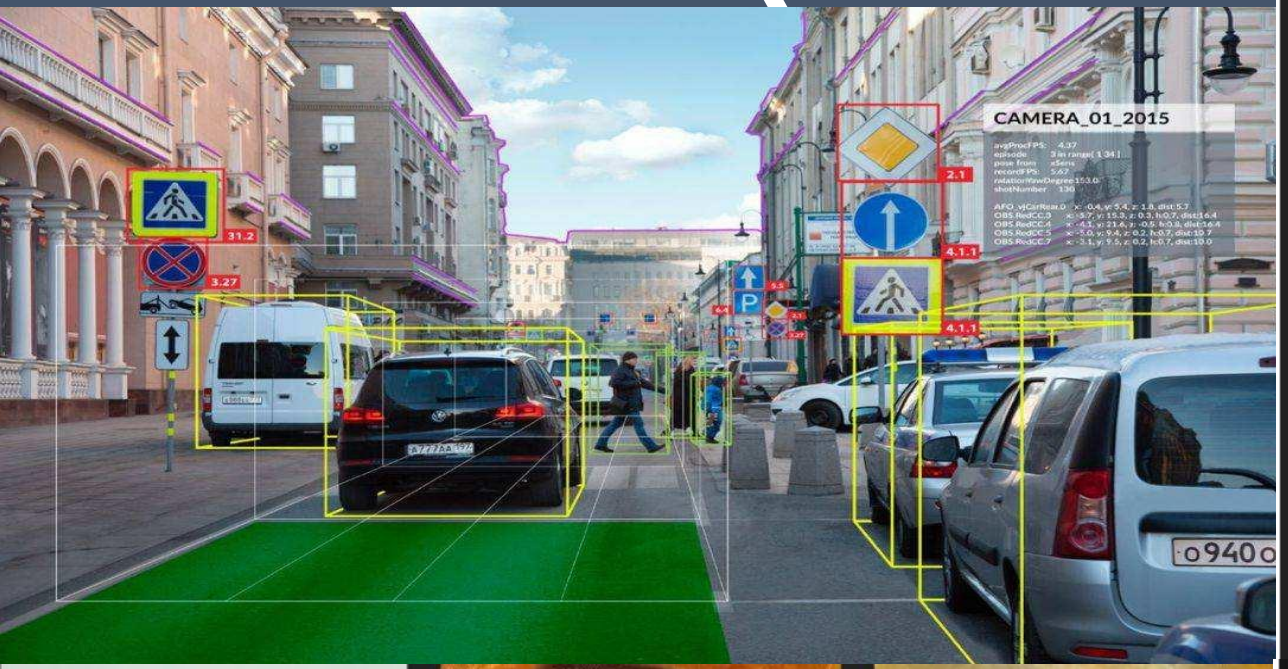


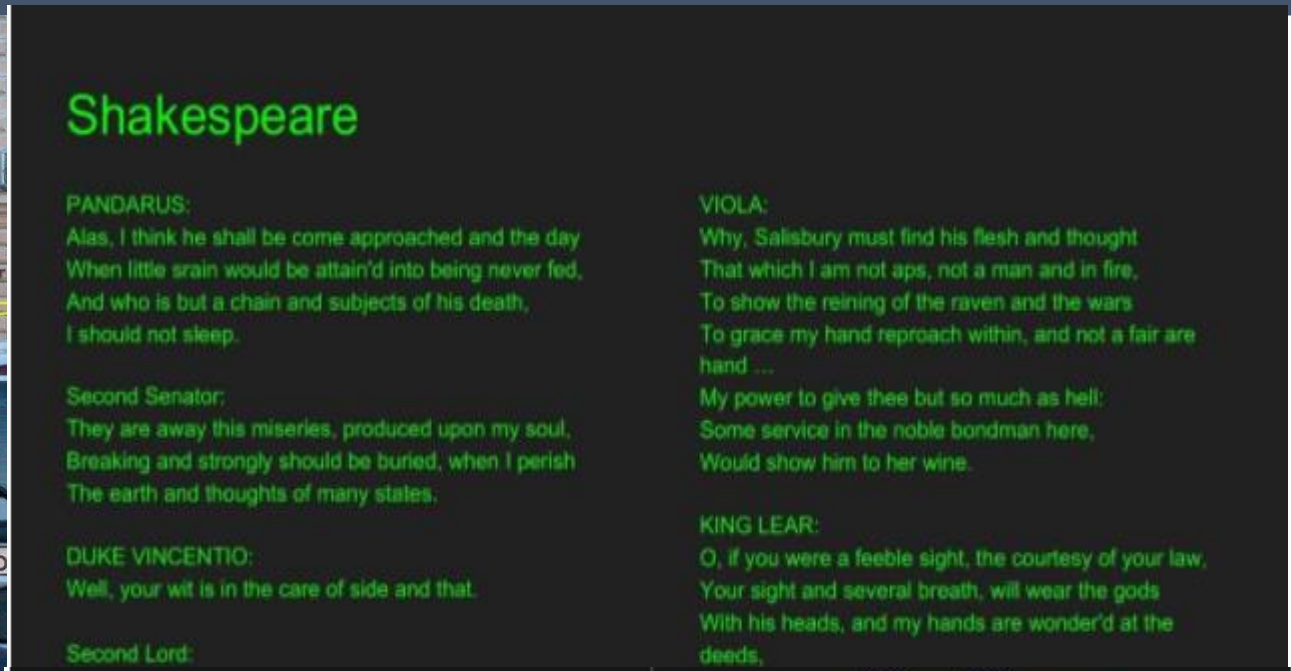
深度学习的发展历程

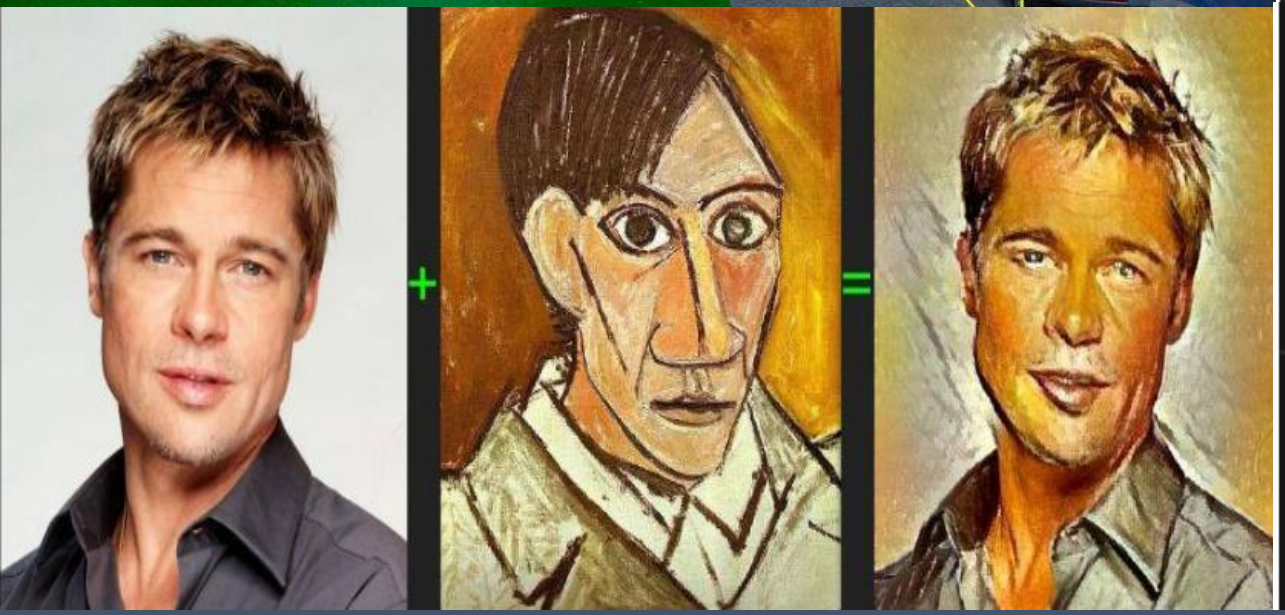

Deep Neural Network
(Pretraining)
SVM
Multi-layered
Perceptron
(Backpropagation)
XOR
Problem
ADALINE
Perceptron
Electronic Brain
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

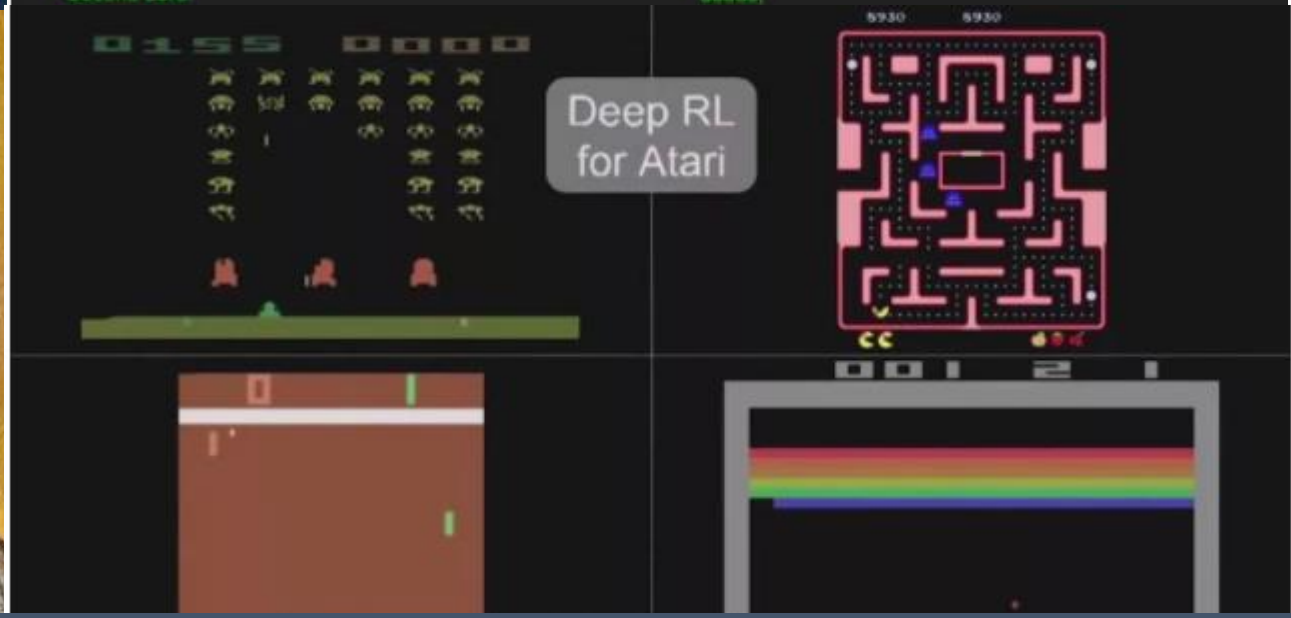

NVIDIA TESLA

深度学习的应用


CAMERA_01_2015


Shakespeare
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.
Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.
DUKE VINCENTIO:
Well, your wit is in the care of side and that.
Second Lord:
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are
hand ...
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.
KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the
deeds,




Deep RL
for Atari

深度学习的应用

消费娱乐

新零售货架商品检测

图片、视频审核

人脸检测

智慧交通

行人检测

车辆检测

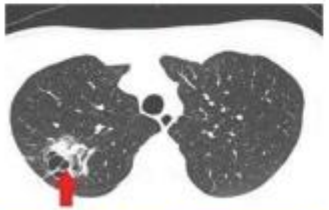
卫星遥感

地块检测

遥感目标检测

智慧医疗

眼底病变检测



肺炎检测

生产质检

零件计数

产品缺陷检测

设备巡检

表计巡检

设备状态监控

厂区安防

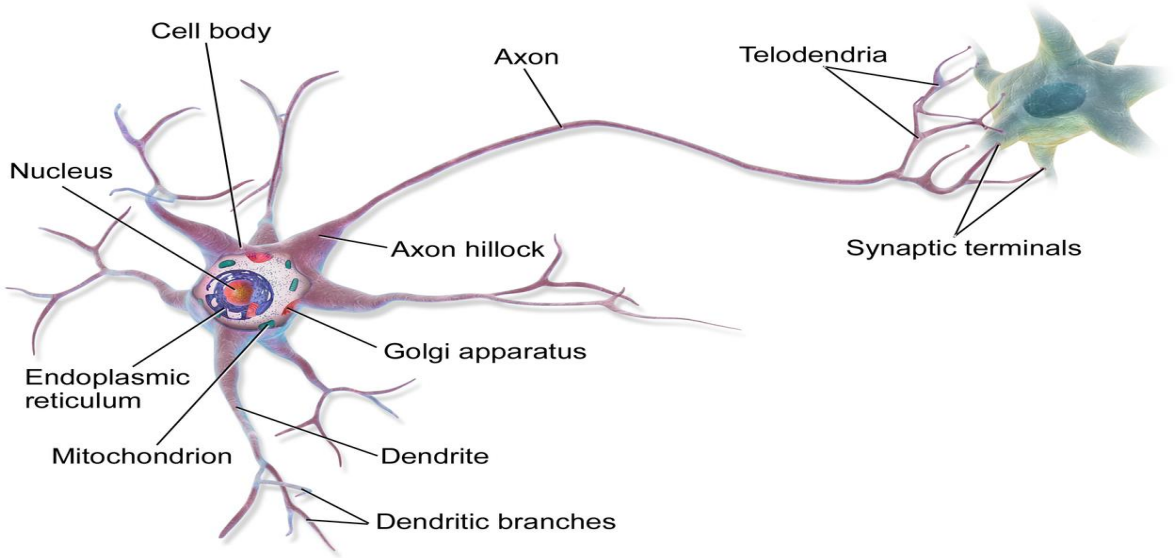
工服安全帽识别

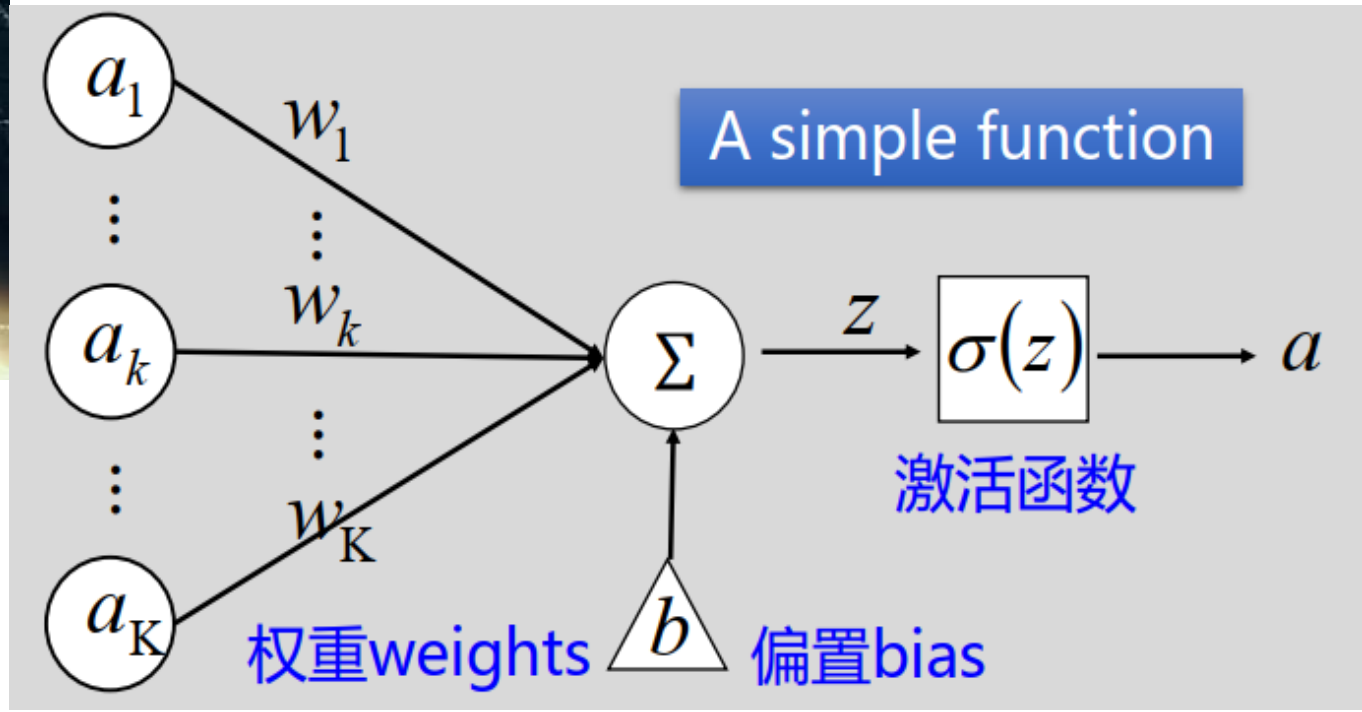
烟火异常检测

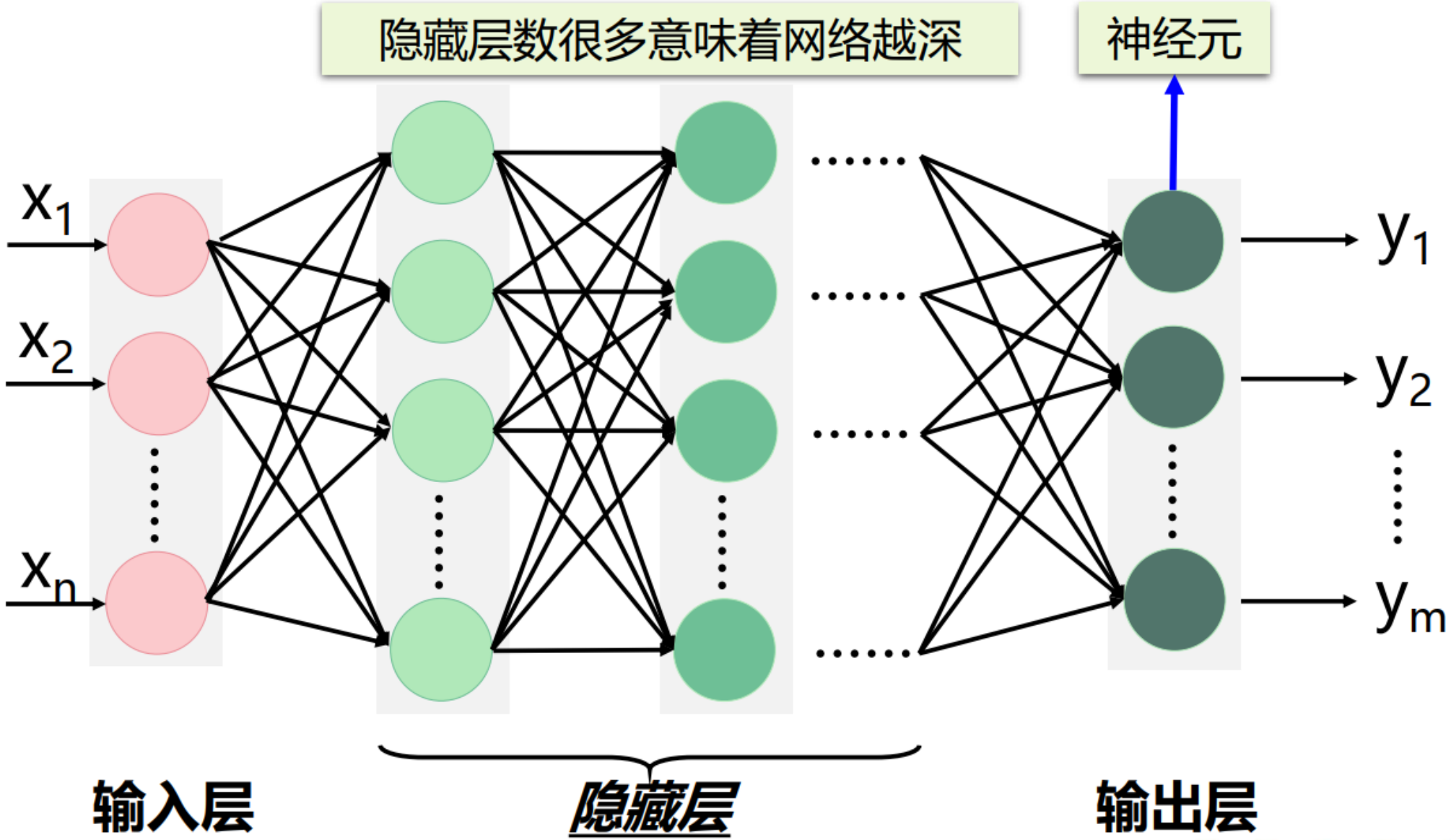
从神经元开始

Start with Neurons

神经元

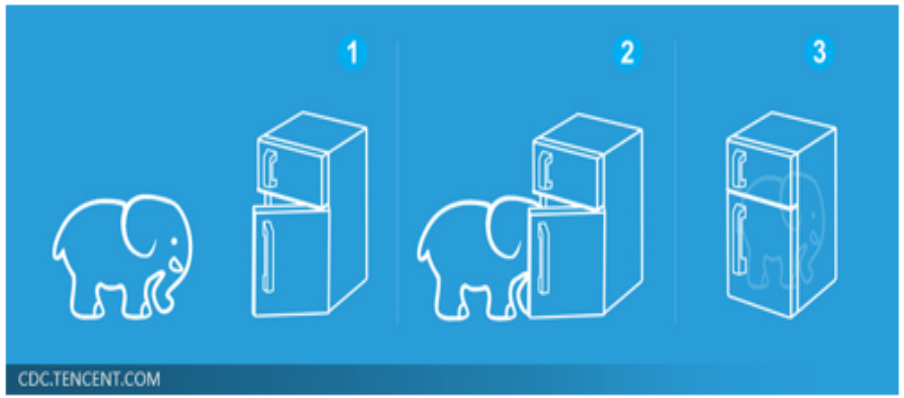

Cell body
Axon
Telodendria
Nucleus
Axon hillock
Synaptic terminals
Golgi apparatus
Endoplasmic reticulum
Mitochondrion
Dendrite
Dendritic branches


a_1
a_k
a_K
w_1
w_k
w_K
A simple function
Σ
z
$\sigma(z)$
a
激活函数
权重weights
b
偏置bias


隐藏层数很多意味着网络越深
神经元
x_1
x_2
x_n
y_1
y_2
y_m
输入层
隐藏层
输出层

Three Steps for Deep Learning

Deep Learning is so simple ……



建立模型

- 选择什么样的网络结构
- 选择多少层数，每层选择多少神经元

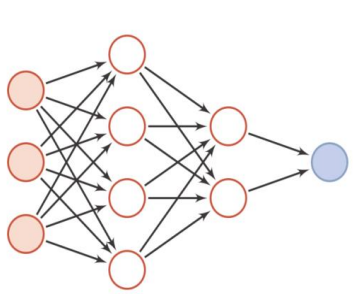
损失函数

- 选择常用损失函数，平方误差，交叉熵....

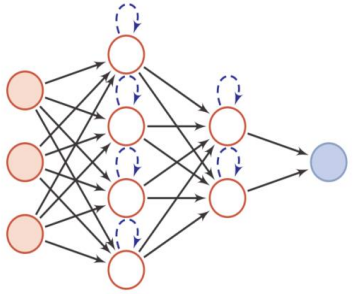
参数学习

- 梯度下降
- 反向传播算法

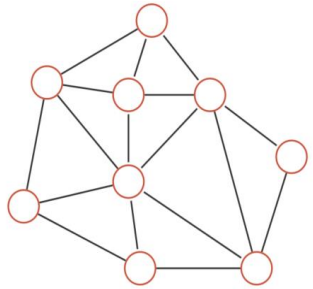
Deep Convolutional Neural Network



(a) 前馈网络

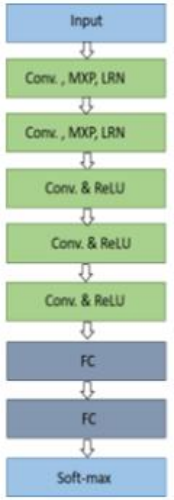


(b) 反馈网络



(c) 图网络

8 layers



Inaccuracy: 16.4%

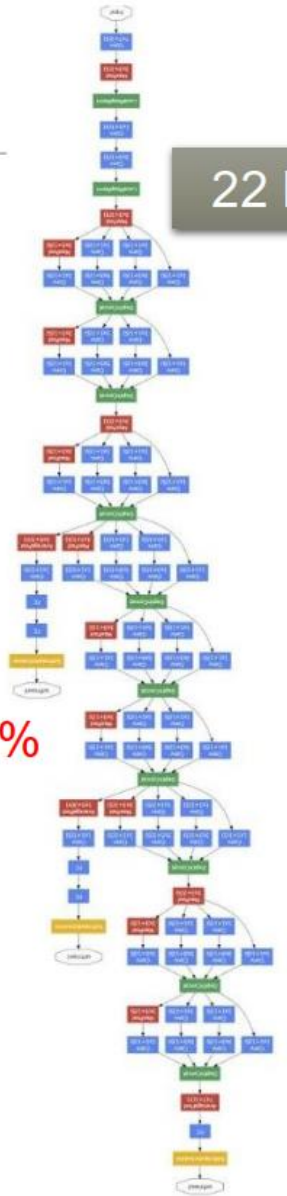
AlexNet (2012)

19 layers

7.3%

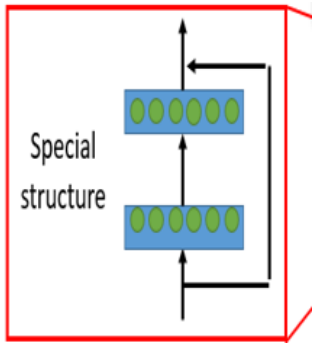
VGG (2014)

22 layers



6.7%

GoogleNet (2014)



3.57%

152 layers

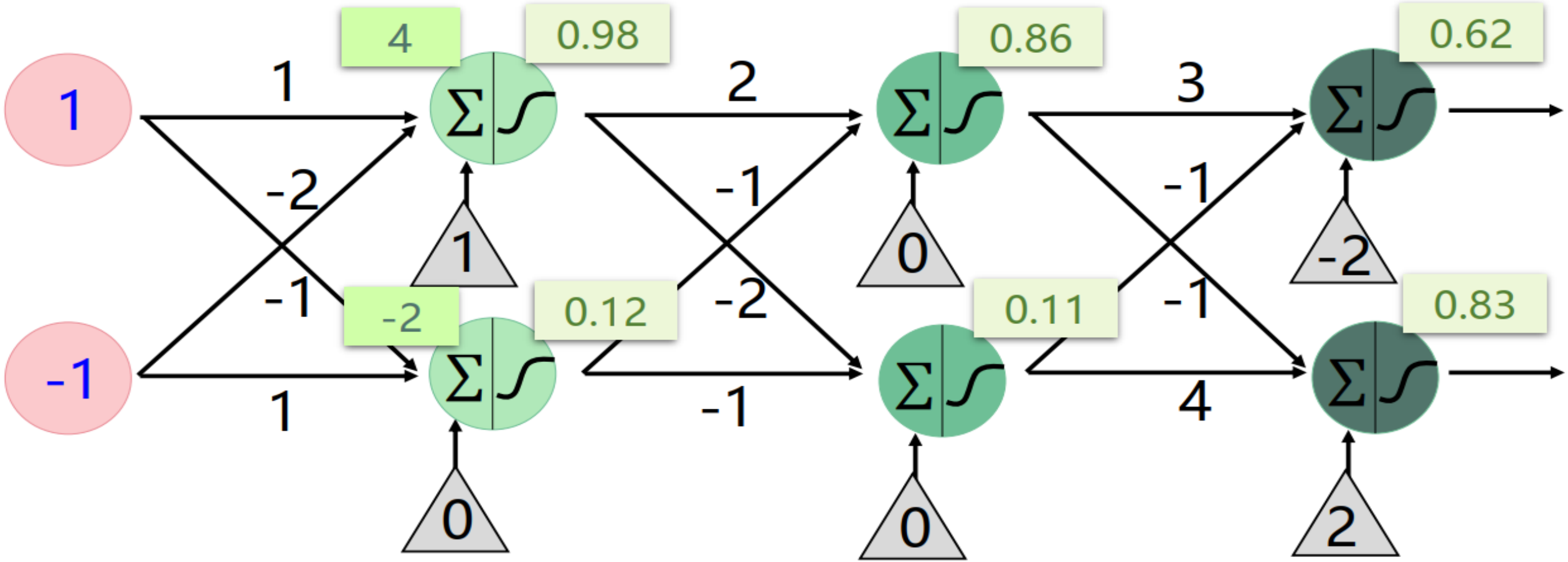
Residual Net (2015)

◆ 为什么引入激活函数

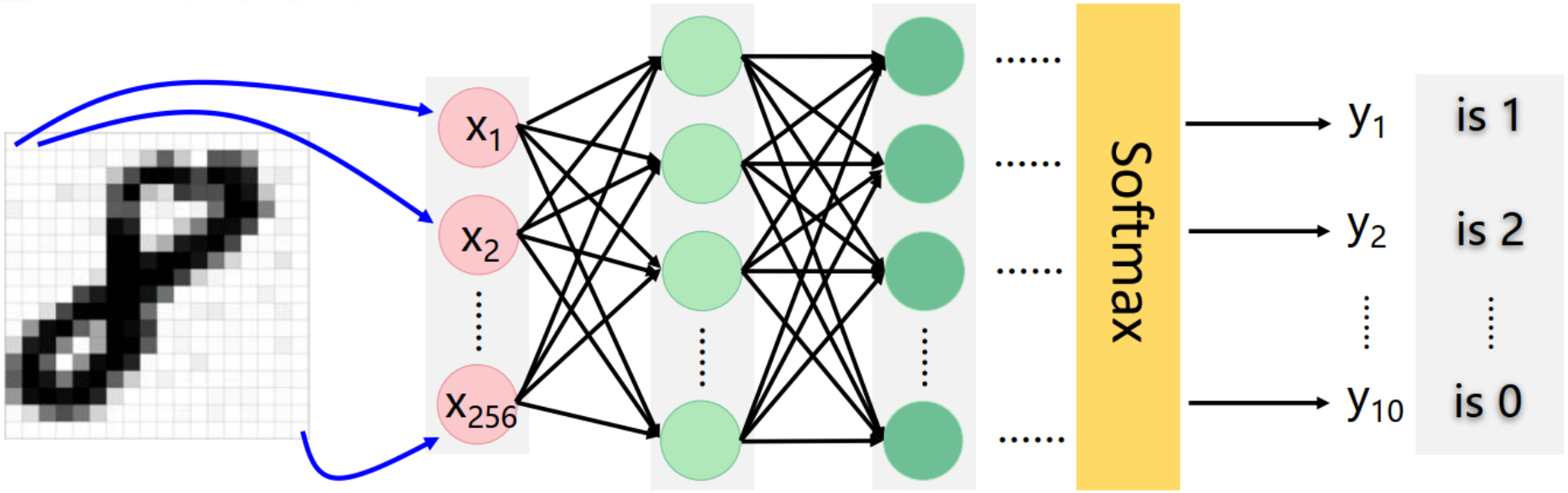
- ➢ 为了增强网络的表达能力，我们需要激活函数来将线性函数->非线性函数
- ➢ 非线性的激活函数需要有连续性。因为连续非线性激活函数可以可导的，所以可以用最优化的方法来求解

Logistic (a.k.a Soft step)	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH	$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan	$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)[7]	$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[8]	$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

常用激活函数示例

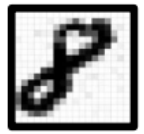


函数：$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)=\begin{bmatrix}0.62\\0.83\end{bmatrix}$



16 x 16 = 256

对于数字识别任务，设计用于分类的损失函数，使得学习目标变为：

输入：  ➡ y_1 值最大；输入：  ➡ y_8 值最大 ……

平方损失函数:

$$L(Y|f(X)) = \sum_N (Y - f(X))^2$$

交叉熵损失函数

$$loss = -\frac{1}{n}\sum_i y_i ln a_i$$

$$loss = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$$

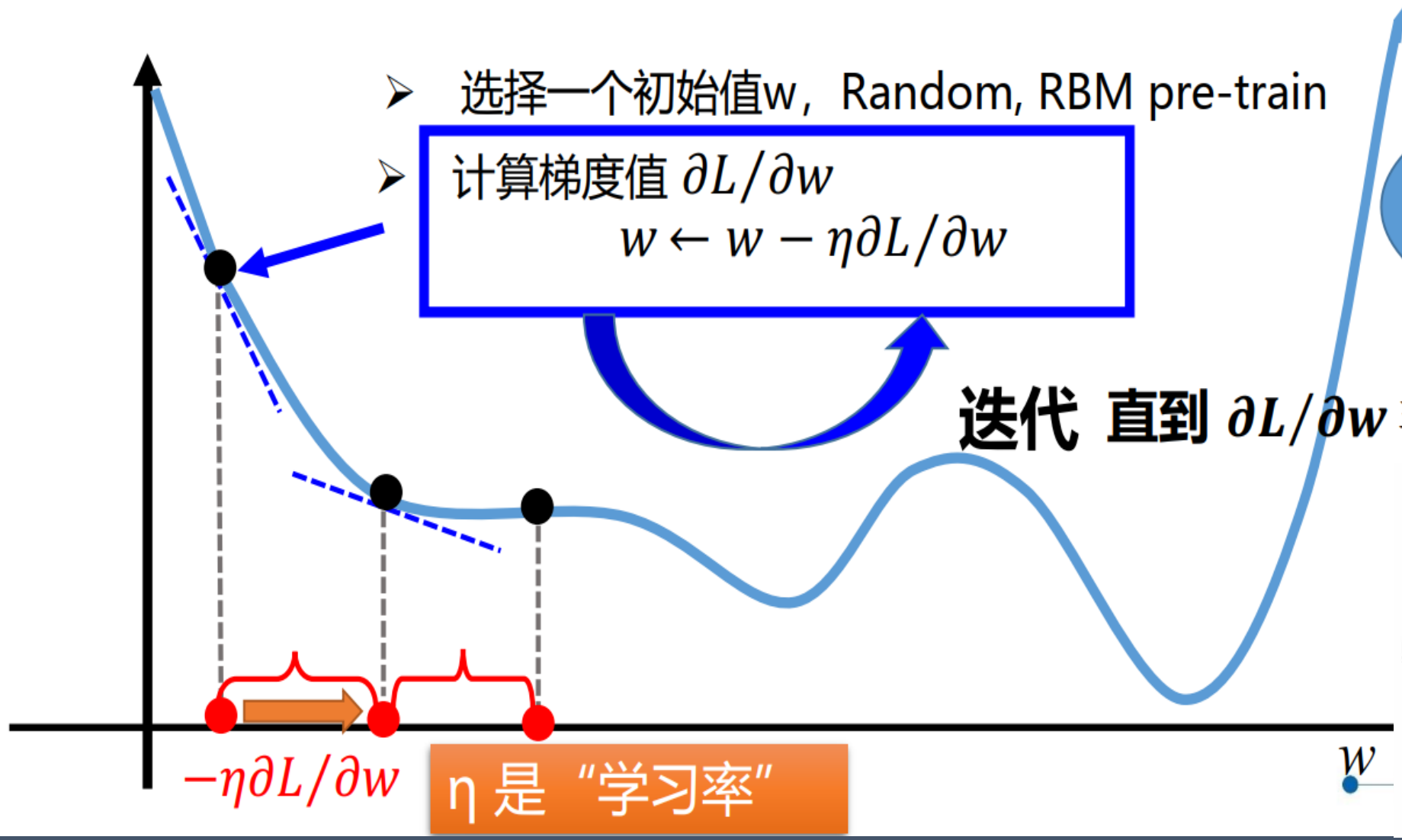
梯度下降法

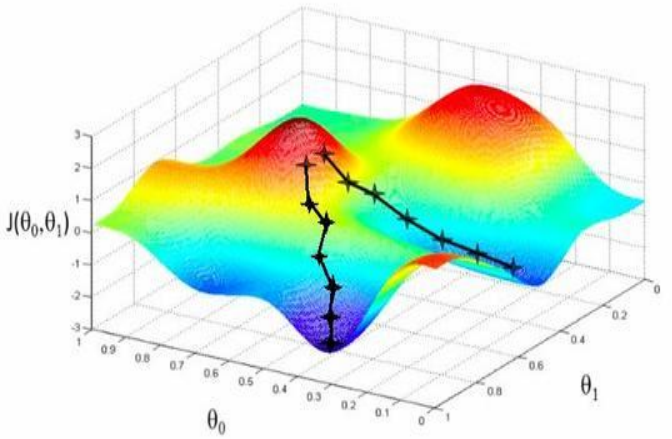
- 选择一个初始值w，Random, RBM pre-train
- 计算梯度值 $\partial L/\partial w$

$$w \leftarrow w - \eta \partial L/\partial w$$

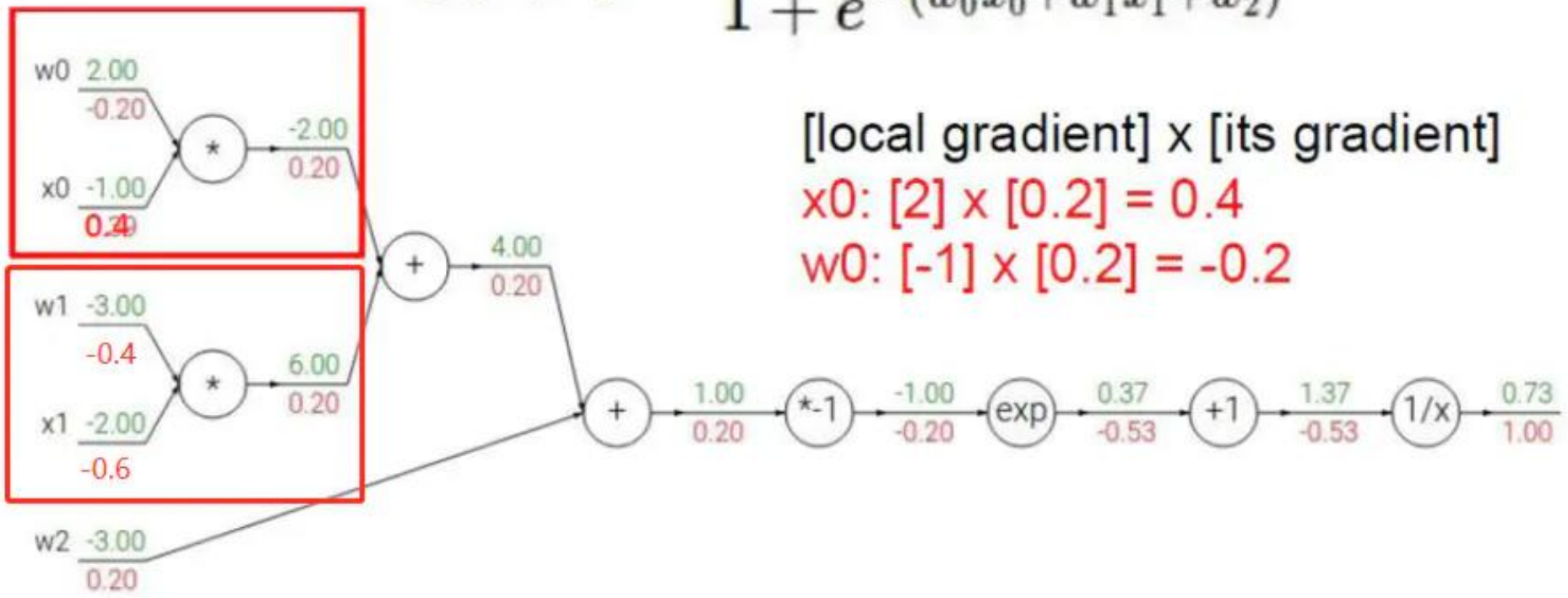
神经网络参数$\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

迭代 直到 $\partial L/\partial w$ 非常小（更新很小）





$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



[local gradient] x [its gradient]
x0: [2] x [0.2] = 0.4
w0: [-1] x [0.2] = -0.2

$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x$$

$$f_a(x) = ax \rightarrow \frac{df}{dx} = a$$

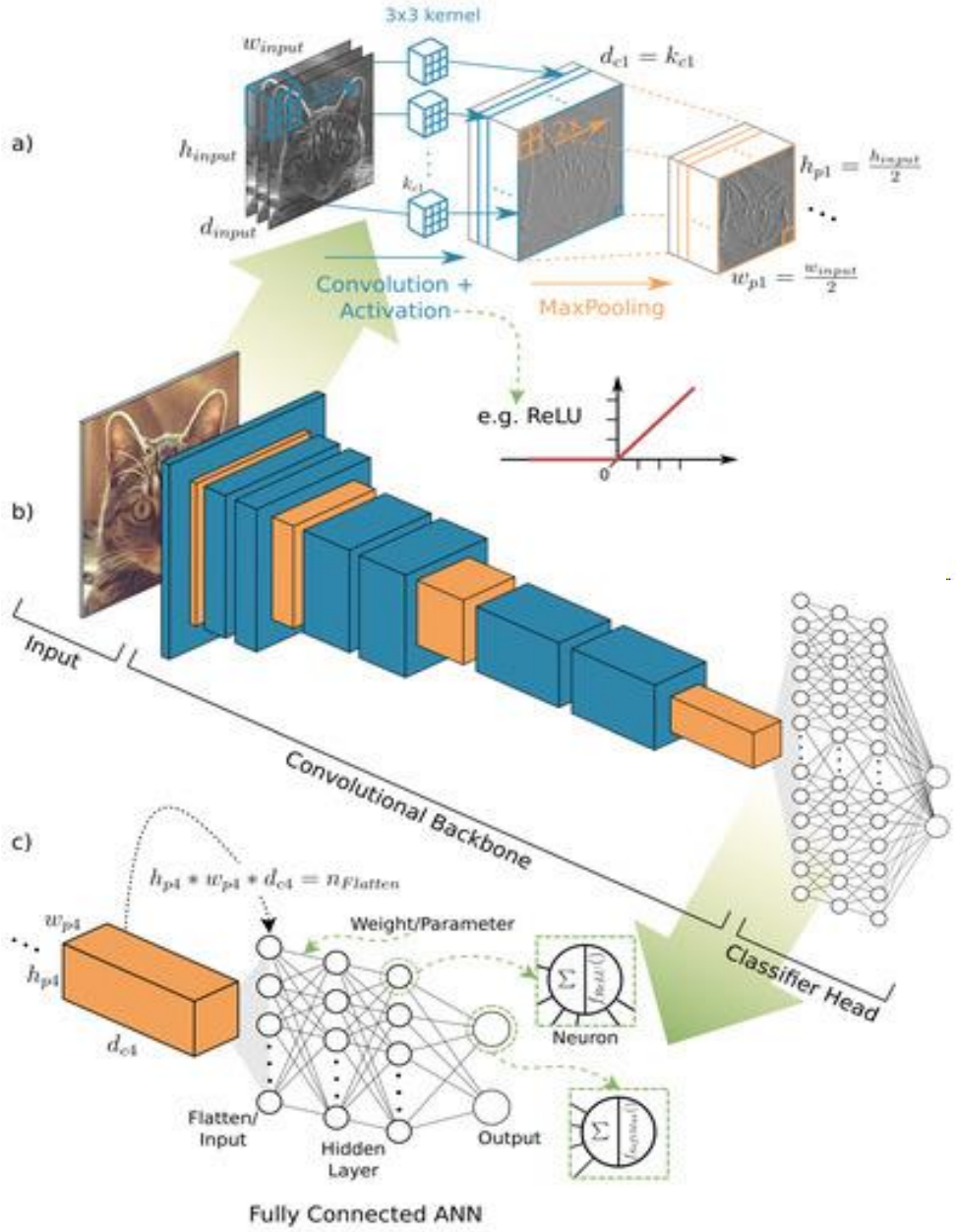
$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -1/x^2$$

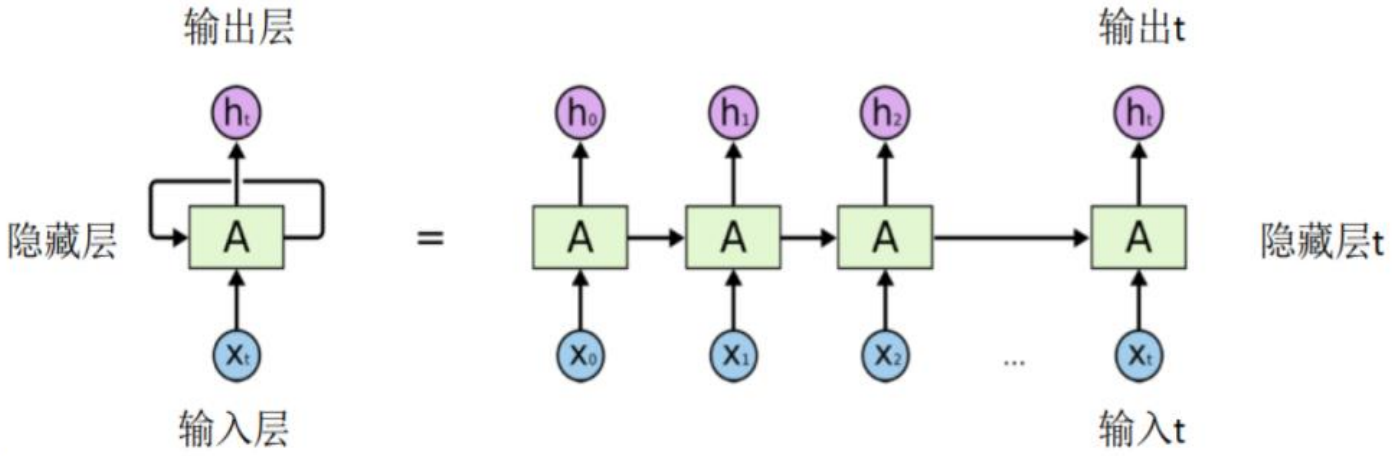
$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1$$

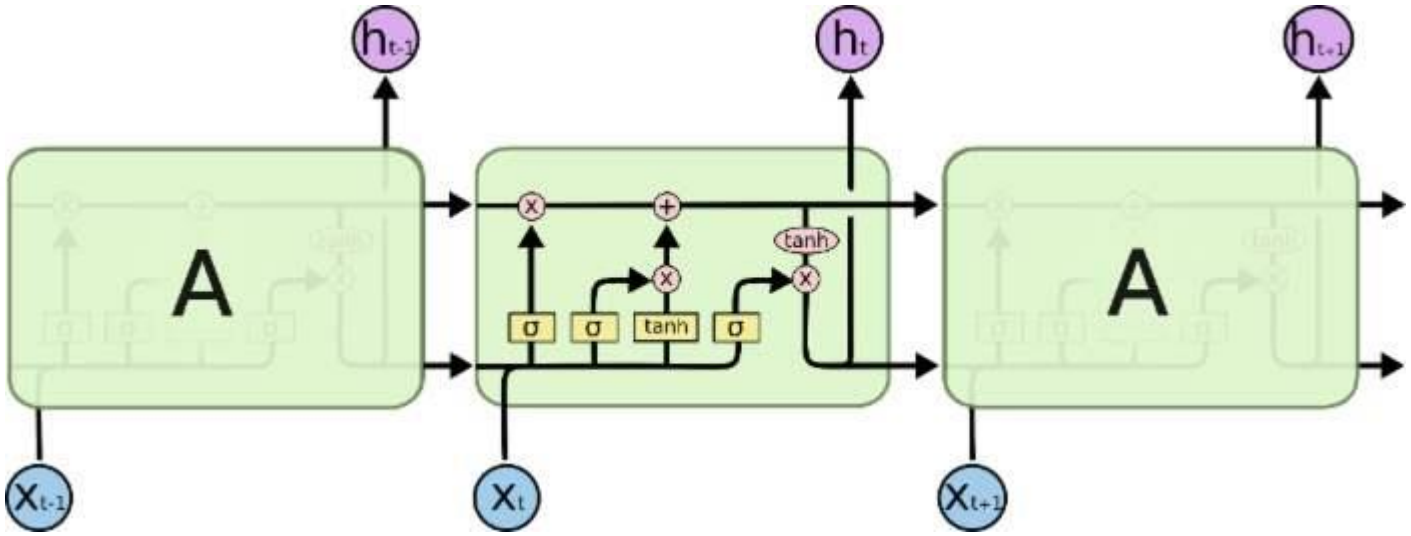
典型神经网络模型

Typical Deep Model

常用深度网络结构

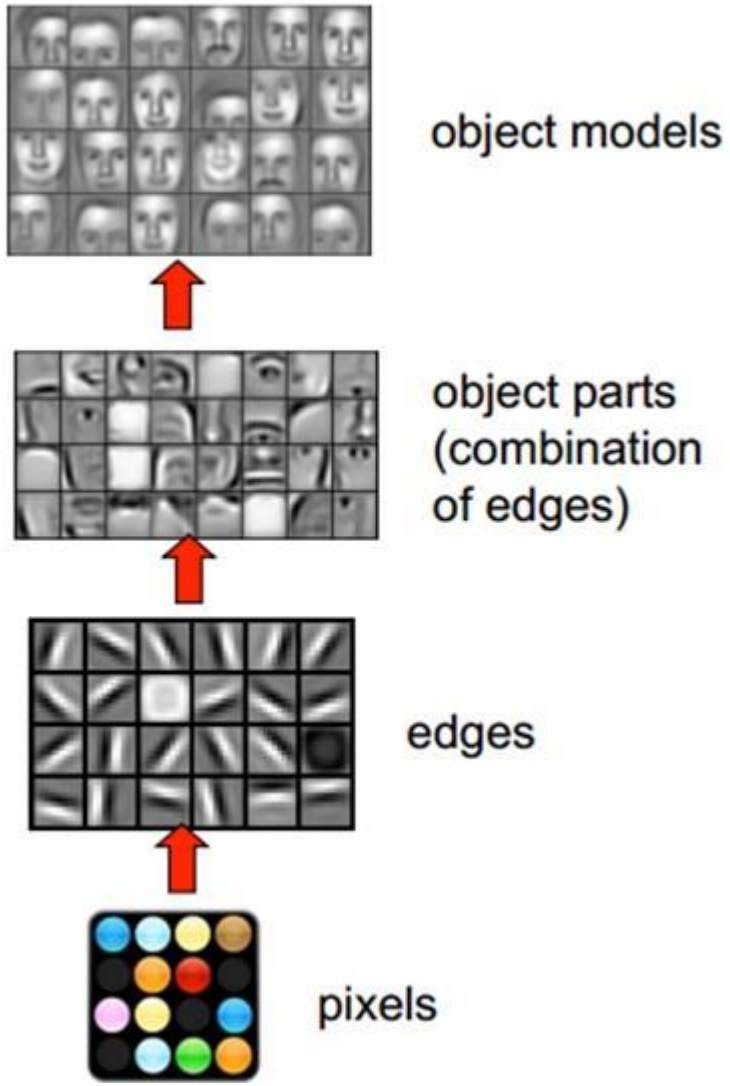

3x3 kernel
a)
Convolution + Activation
MaxPooling
e.g. ReLU
b)
Input
Convolutional Backbone
Classifier Head
c)
Weight/Parameter
Neuron
Flatten/ Input
Hidden Layer
Output
Fully Connected ANN

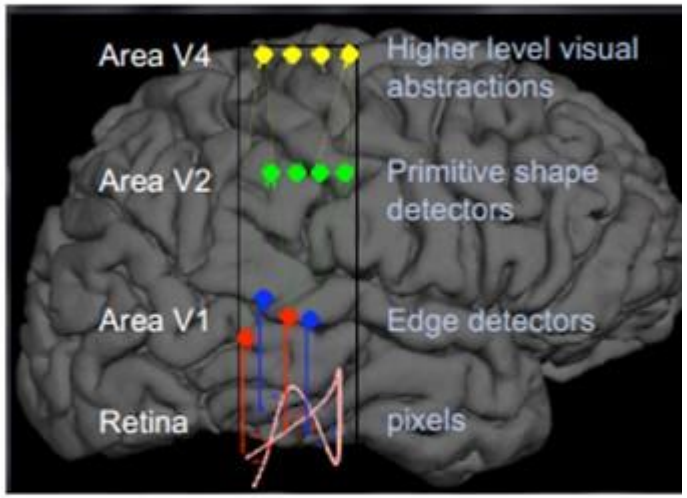

输出层
输出t
隐藏层
=
隐藏层t
输入层
输入t

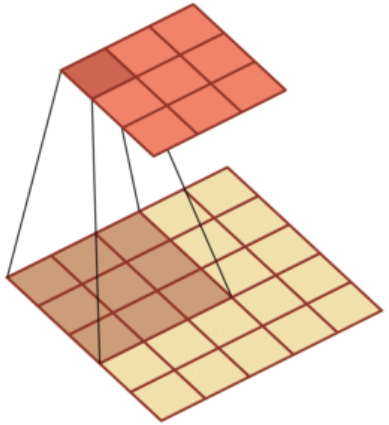

A
A
tanh
σ
σ
tanh
σ

卷积神经网络

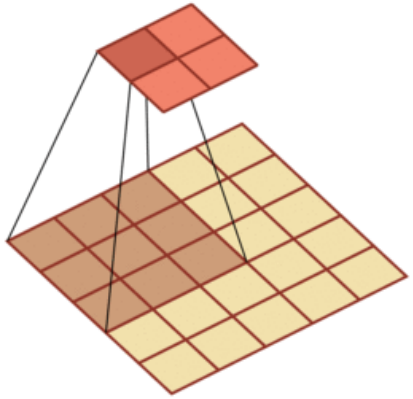
CNN


object models
object parts
(combination
of edges)
edges
pixels

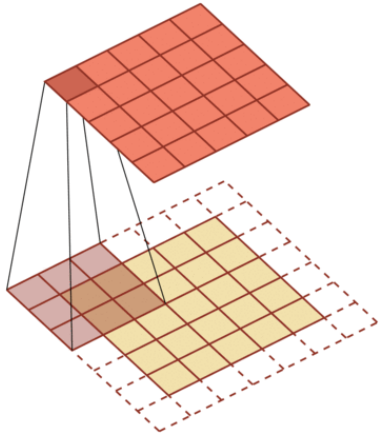

Area V4
Higher level visual
abstractions
Area V2
Primitive shape
detectors
Area V1
Edge detectors
Retina
pixels



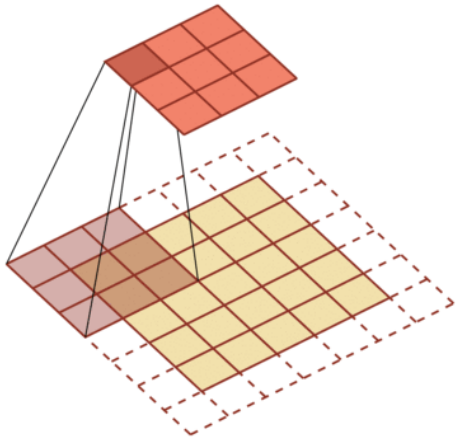
步长1，零填充0



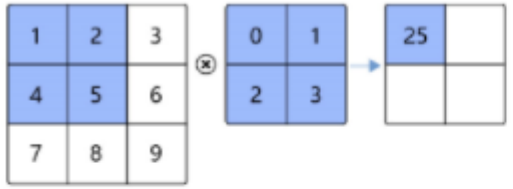
步长2，零填充0



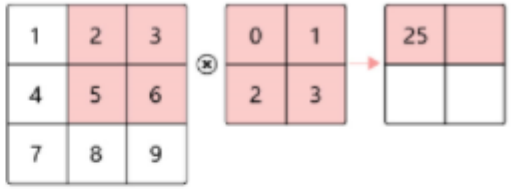
步长1，零填充1



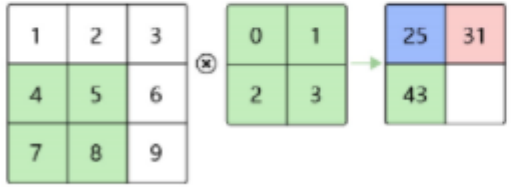
步长2，零填充1



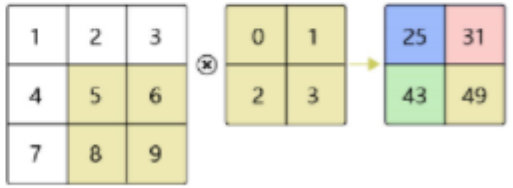
(a) 0x1+1x2+2x4+3x5=25



(b) 0x2+1x3+2x5+3x6=31



(c) 0x4+1x5+2x7+3x8=43

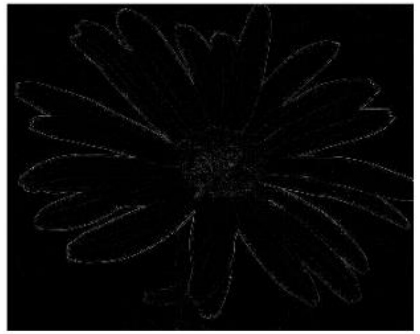


(d) 0x5+1x6+2x8+3x9=49

- **卷积运算**
  - ✓ 用同一个卷积核从左到右Z字形滑动遍历每一个可能的局部位置
  - ✓ 在每个位置计算卷积核和局部数据的点积值

原始图像

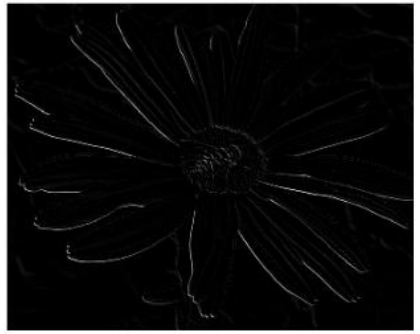
$$\otimes \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} =$$



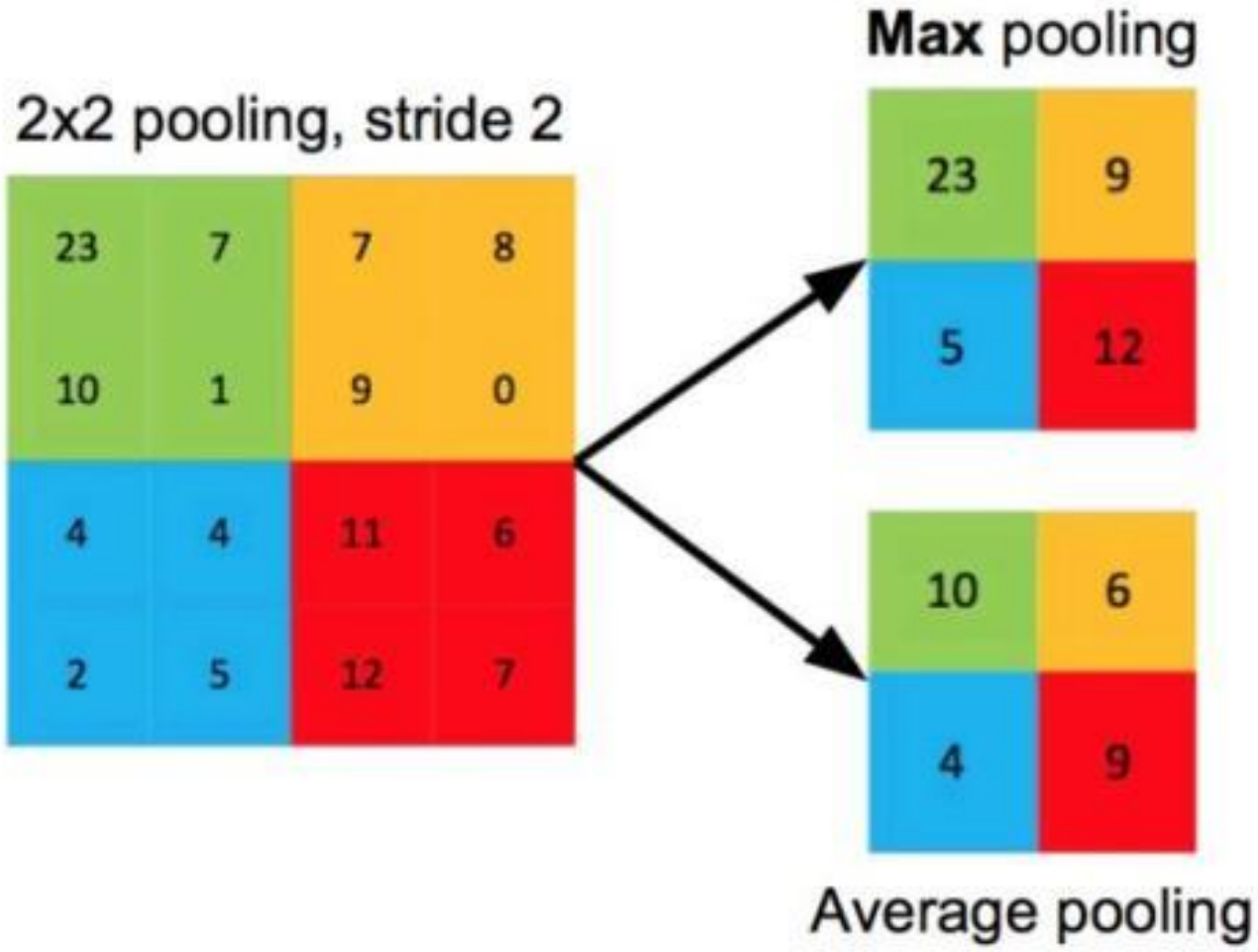
$$\begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix} =$$

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

滤波器



输出特征映射



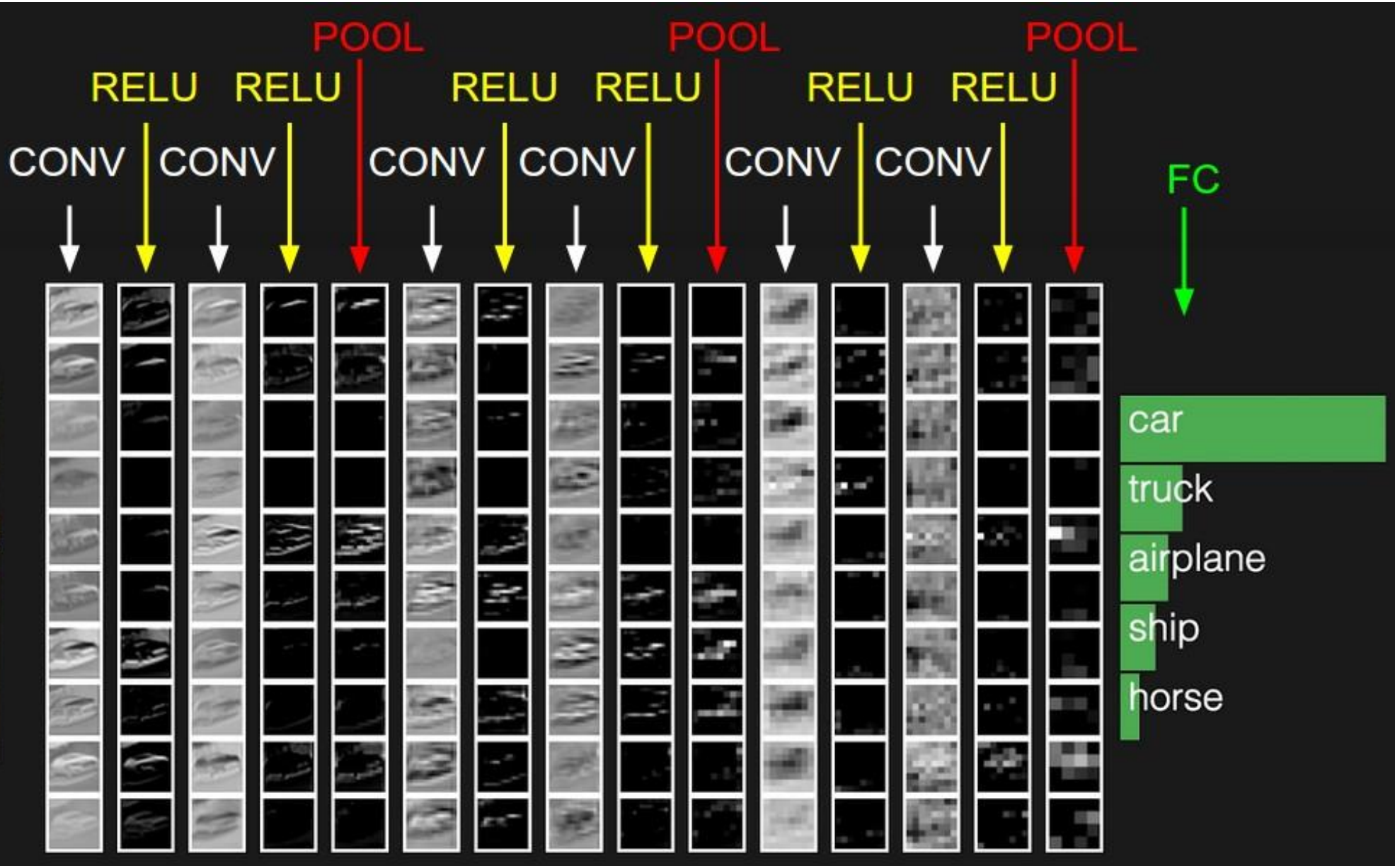
Feature Map

6	4	8	5
5	4	5	8
3	6	7	7
7	9	7	2

Max-Pooling

卷积网络

卷积网络识别图像例子



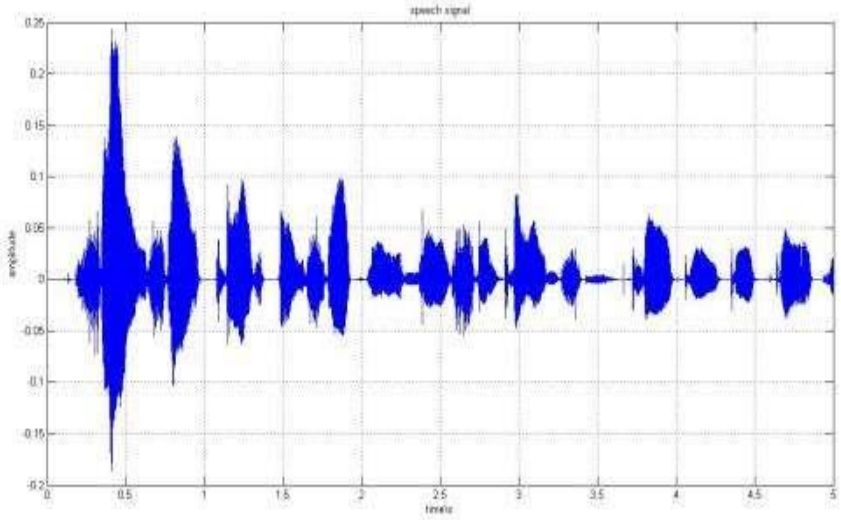
循环神经网络

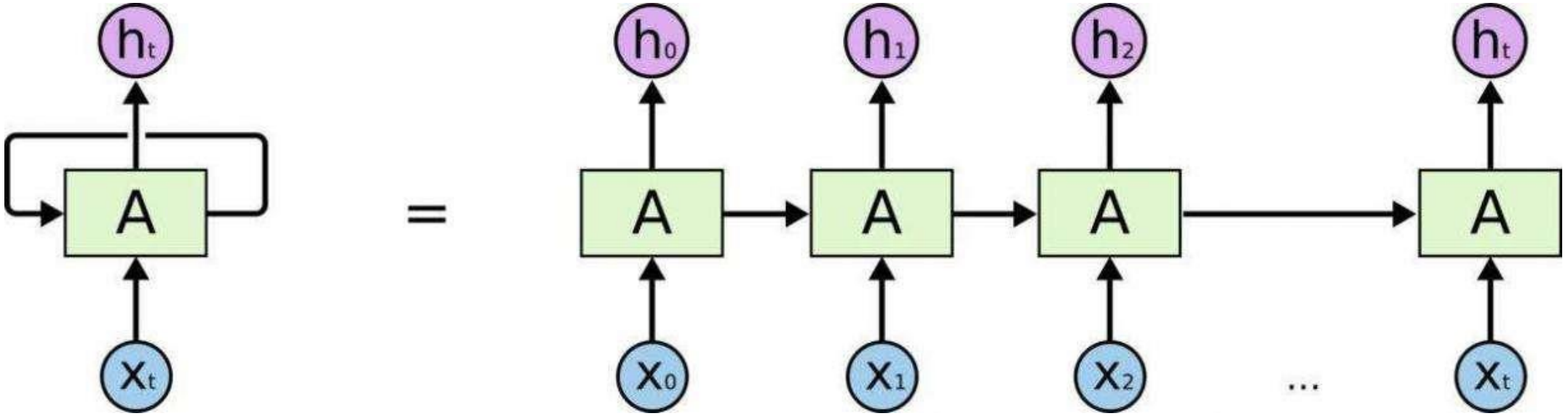
RNNs

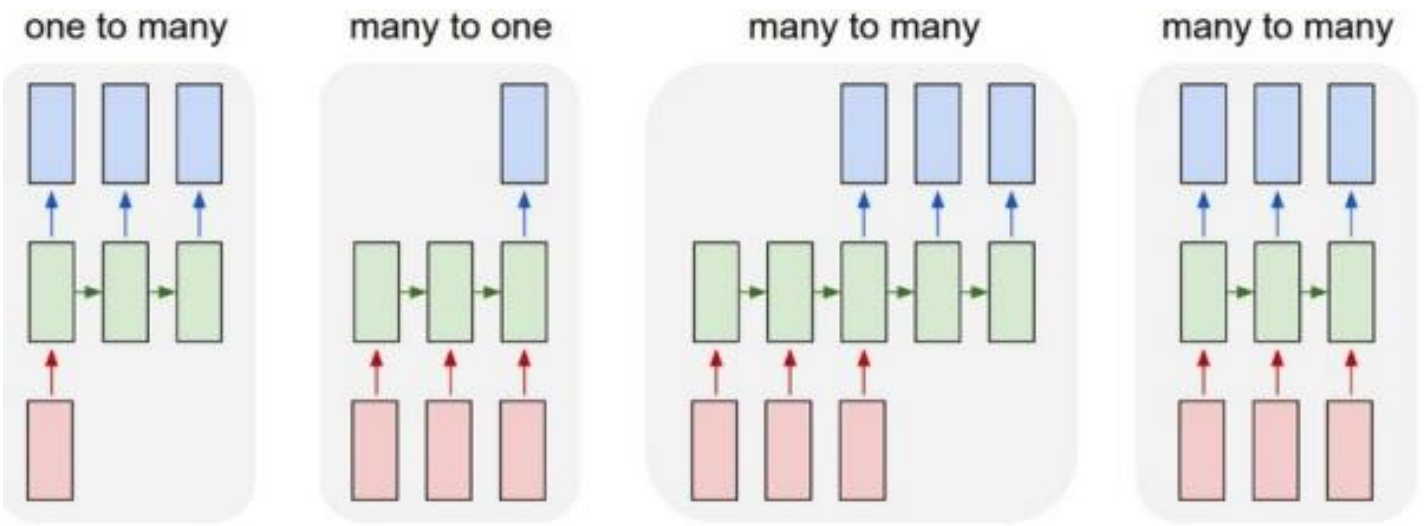
循环网络

序列数据


8942.08
8525.08
7691.08
7274.08
6857.08
6440.08
6023.08
5606.08
4772.08
4355.08
11.47
13.12
14.37
16.02
17.27
玉树芝兰


speech signal


h_t
A
X_t
=
h_0
h_1
h_2
h_t
A
A
A
A
X_0
X_1
X_2
...
X_t

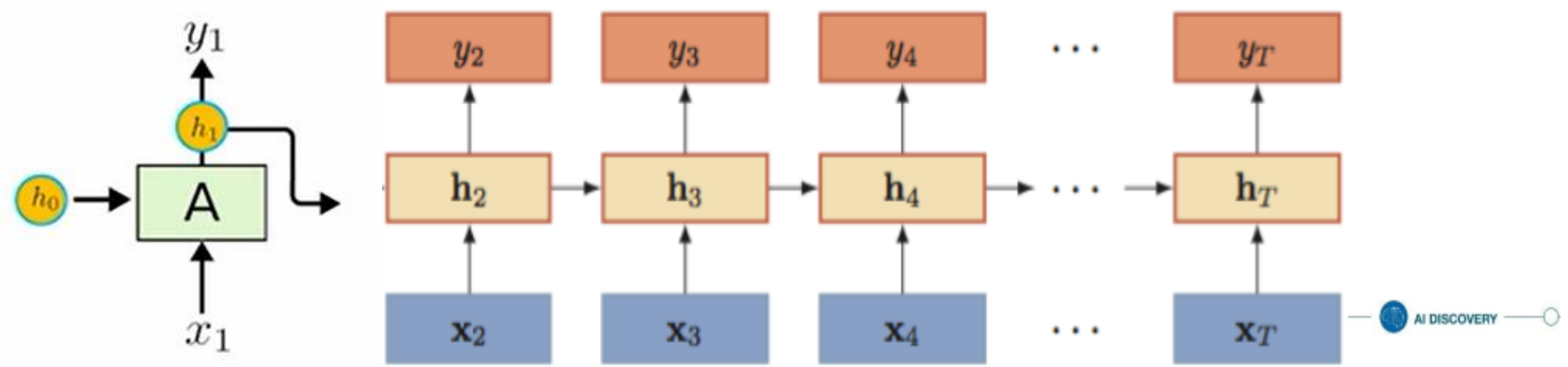

one to many
many to one
many to many
many to many

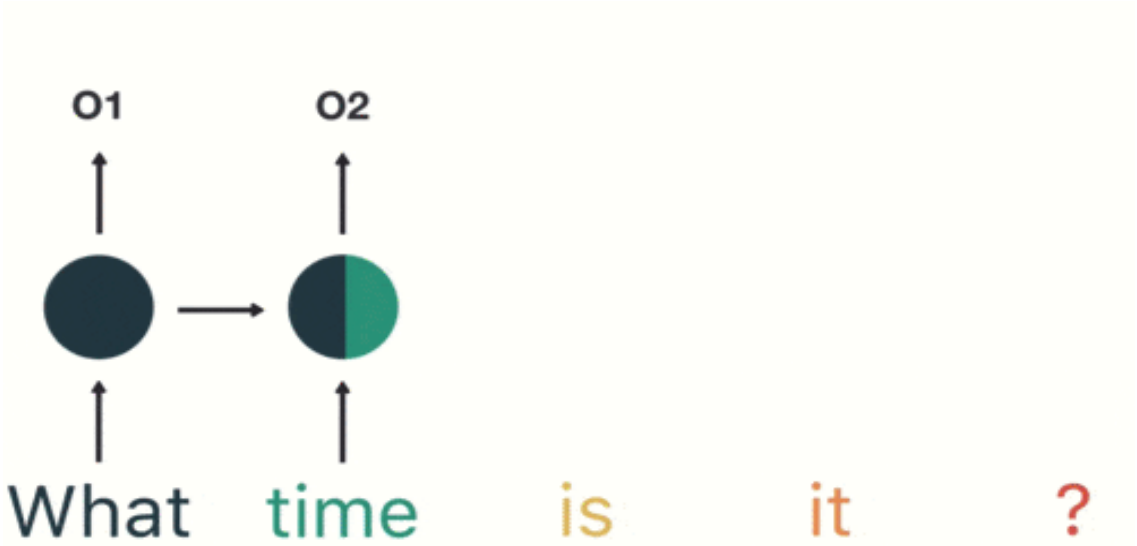
✓假设时刻 t 时，输入为 $\mathbf{x_t}$，隐层状态（隐层神经元活性）为 $\mathbf{h_t}$。$\mathbf{h_t}$ 不仅和当前时刻的输入相关，也和上一个时刻的隐层状态相关。

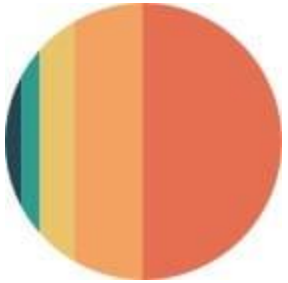
✓一般我们使用如下函数：

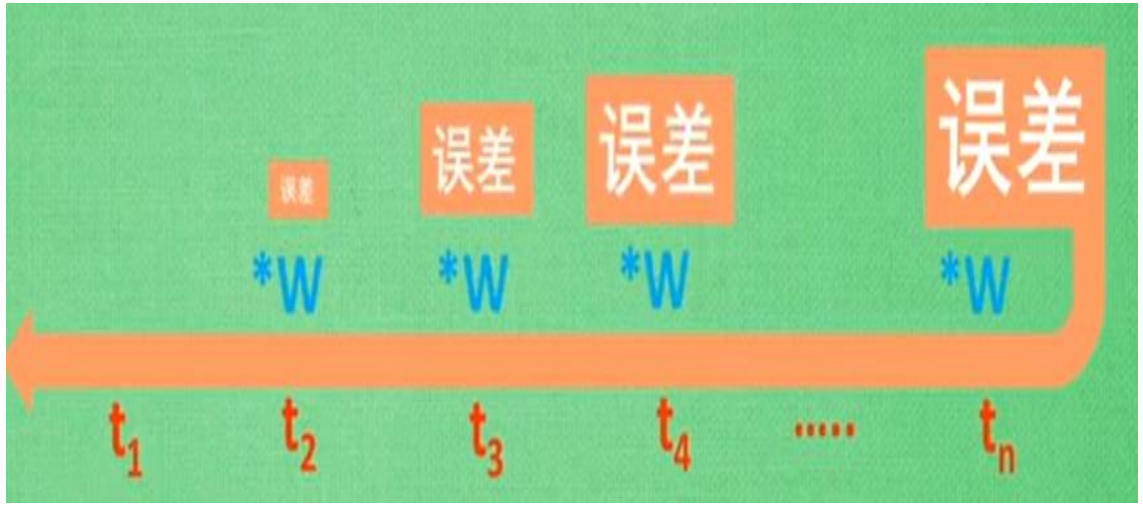
$$\boldsymbol{h}_t = f(\boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{W}\boldsymbol{x}_t + b) \qquad \boldsymbol{y}_t = softmax(\boldsymbol{W}^{(S)}\boldsymbol{h}_t)$$

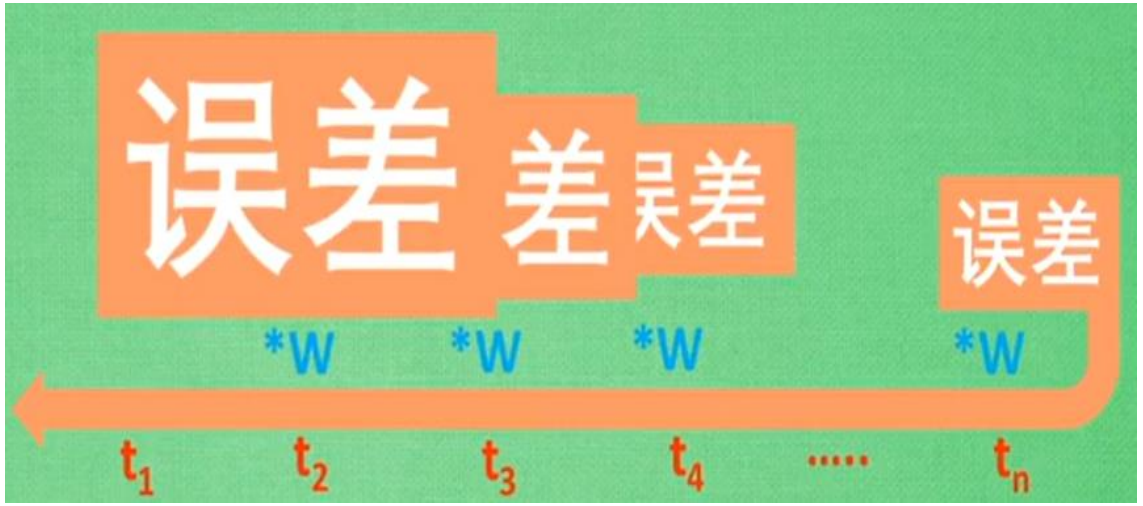
✓ 这里，f 是非线性函数，通常为 *sigmod*函数或 *tanh* 函数。

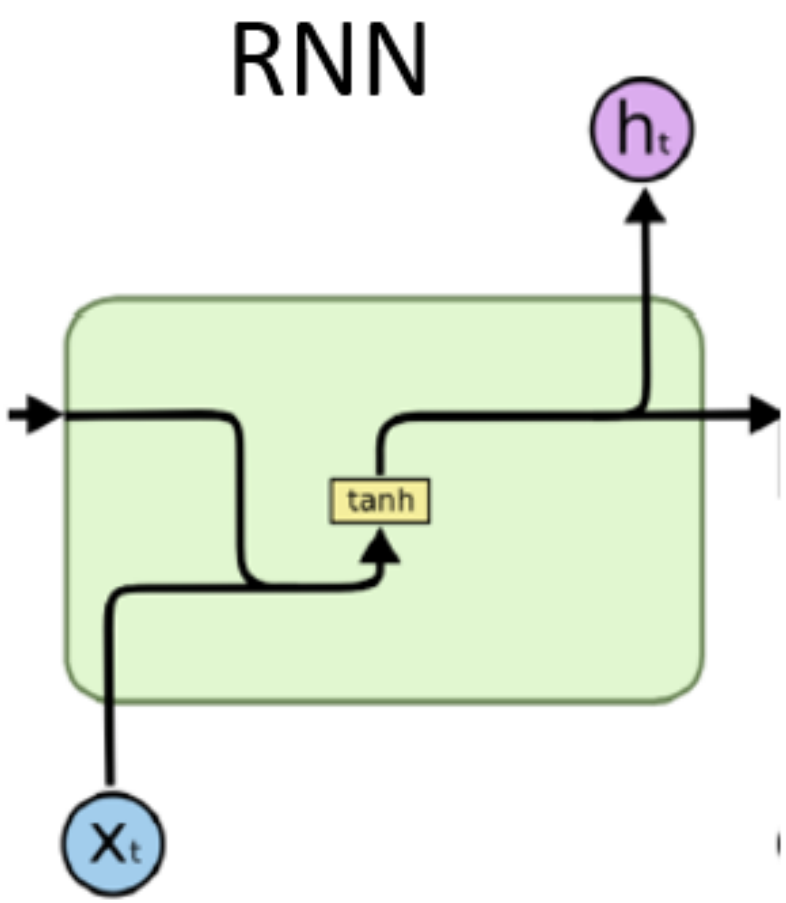


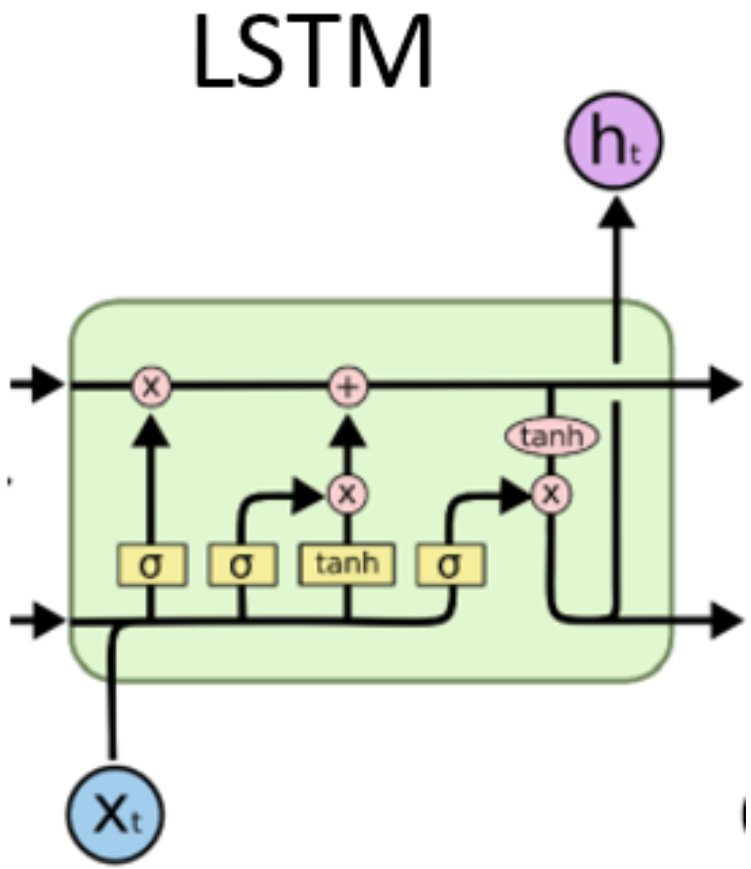

O1
O2
What
time
is
it
?

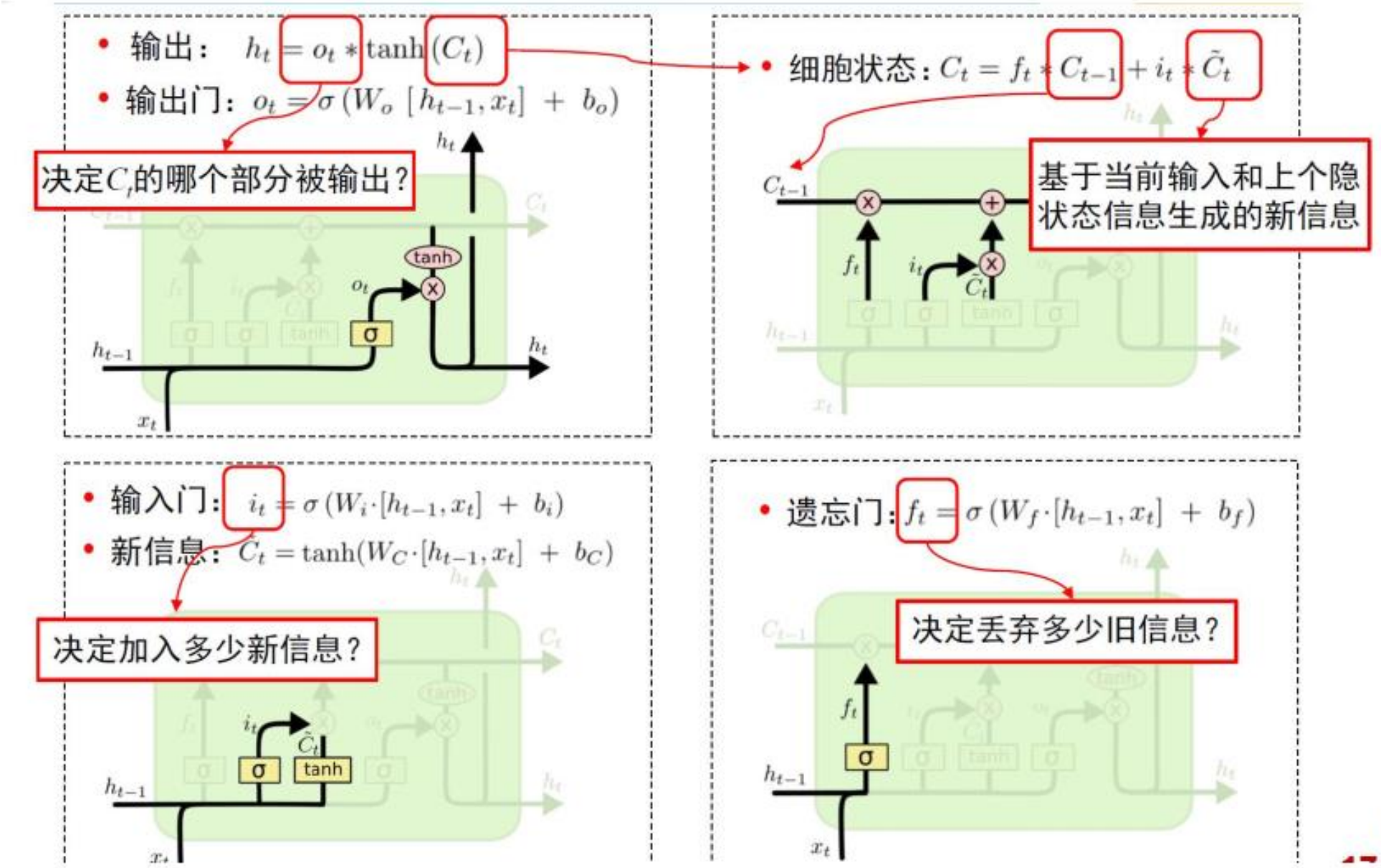



误差
误差
误差
误差
*W
*W
*W
*W
t1
t2
t3
t4
.....
tn


误差
差
误差
误差
*W
*W
*W
*W
t1
t2
t3
t4
.....
tn


RNN
h_t
tanh
X_t


LSTM
h_t
tanh
σ
σ
tanh
σ
X_t


• 输出： $h_t = o_t * \tanh(C_t)$
• 输出门： $o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$
决定C_t的哪个部分被输出？
• 细胞状态： $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
基于当前输入和上个隐状态信息生成的新信息
• 输入门： $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
• 新信息： $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
决定加入多少新信息？
• 遗忘门： $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
决定丢弃多少旧信息？
σ
tanh

深度学习框架

Deep Learning Frameworks

常用深度学习框架


TensorFlow


K
Keras


PYTORCH


Caffe2


Microsoft
CNTK


theano


PaddlePaddle

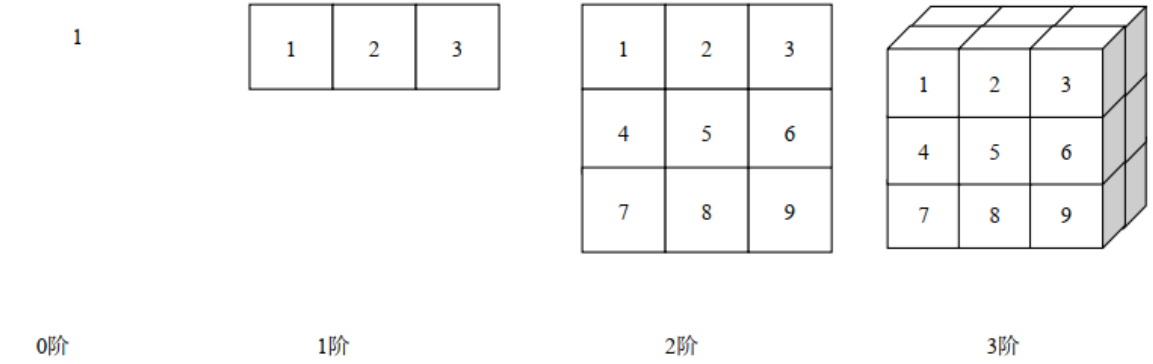

Chainer

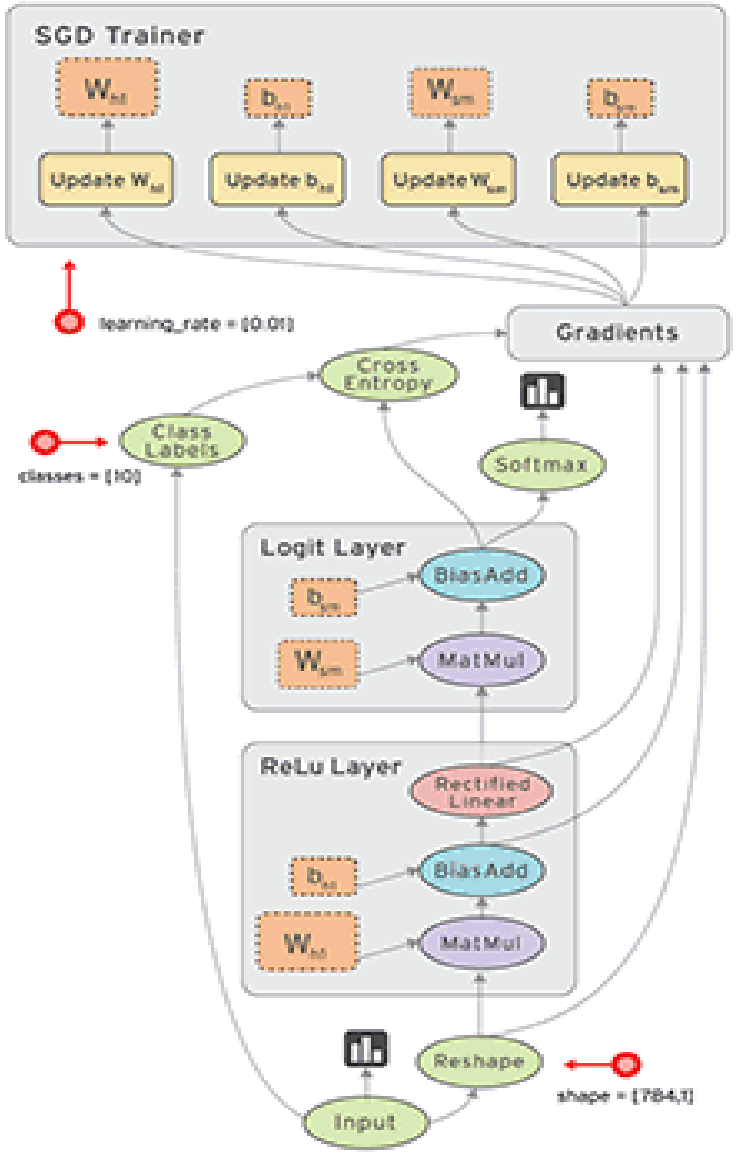
- **Tensorflow 是一个编程系统，使用图 (graphs) 来表示计算任务，图中的节点称为operation，一个operation获得多个Tensor执行计算，产生多个Tensor，图必须在会话session里启动。**

在 TensorFlow 中，张量（`Tensor`）表示某种相同**数据类型**的**多维数组**。

因此，张量有两个重要属性：

1. 数据类型（如浮点型、整型、字符串）
2. 数组形状（各个维度的大小）





```python
import tensorflow as tf
a = tf.constant([1.0, 2.0])  # constant()代表定义常数
b = tf.constant([3.0, 4.0])
result = tf.add(a,b)
print(result)

m1 = tf.constant([[3.0,3.0]])  # constant()代表定义常数
m2 = tf.constant([[2.0], [3.0]])
product = tf.matmul(m1,m2)#创建一个矩阵乘法op，把m1和m2传入
print(product)

# 定义会话的方式（常用）
with tf.Session() as sess:
    add_result=sess.run(result)
    print(add_result)

# 旧版本定义一个会话 启动默认图
sess=tf.Session()
product_res =sess.run(product)
print(product_res)
sess.close()
```

输出结果：

Tensor("Add:0", shape=(2,), dtype=float32)

节点名 第0个输出 维度 一维数组长度2 数据类型

Tensor("MatMul:0", shape=(1, 1), dtype=float32)

[4. 6.]

[[15.]]

操作类型	典型操作
基础算术	add/multiply/mod/sqrt/sin/trace/fft/argmin
数组运算	size/rank/split/reverse/cast/one_hot/quantize
梯度裁剪	clip_by_value/clip_by_norm/clip_by_global_norm
逻辑控制和调试	identity/logical_and/equal/less/is_finite/is_nan
数据流控制	enqueue/dequeue/size/take_grad/apply_grad/
初始化操作	zeros_initializer/random_normal_initializer/orthogonal_initializer
神经网络运算	convolution/pool/bias_add/softmax/dropout/erosion2d
随机运算	random_normal/random_shuffle/multinomial/random_gamma
字符串运算	string_to_hash_bucket/reduce_join/substr/encode_base64
图像处理运算	encode_png/resize_images/rot90/hsv_to_rgb/adjust_gamma

```python
import tensorflow as tf

x = tf.Variable([1, 2])
a = tf.constant([3, 3])

sub = tf.subtract(x, a)  # 增加一个减法op
add = tf.add(x, sub)  # 增加一个加法op

# 注意变量再使用之前要再sess中做初始化，但是下边这种初始化方法不会指定变量的初始化顺序
init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)
    print(sess.run(sub))
    print(sess.run(add))


Output:
[-2 -1]
[-1  1]
```

```python
# 创建一个名字为'counter'的变量 初始化0
state = tf.Variable(0, name='counter')
new_value = tf.add(state, 1)  # 创建一个op，作用是使state加1
update = tf.assign(state, new_value)  # 赋值op
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(state))
    for _ in range(5):
        sess.run(update)
        print(sess.run(state))
Output:
0
1
2
3
4
5
```

```
# Fetch概念 在session中同时运行多个op
input1 = tf.constant(3.0)  #
constant()是常量不用进行init初始化
input2 = tf.constant(2.0)
input3 = tf.constant(5.0)

add = tf.add(input2, input3)
mul = tf.multiply(input1, add)

with tf.Session() as sess:
    result = sess.run([mul, add])
    # 这里的[]就是Fetch操作
    print(result)

Output:
[21.0, 7.0]
```

```
# Feed
# 创建占位符
input1 = tf.placeholder(tf.float32)
input2 = tf.placeholder(tf.float32)
# 定义乘法op，op被调用时可通过Feed的方
式将input1、input2传入
output = tf.multiply(input1,
input2)

with tf.Session() as sess:
      # feed的数据以字典的形式传入
       print(sess.run(output,feed_di
ct={input1: [7.], input2: [2.]}))

Output:
[14.0]
```

```python
import tensorflow as tf
import numpy as np

# 使用numpy生成100个随机点
x_data = np.random.rand(100)
y_data = x_data * 0.1 + 0.2  # 这里我们设定已知直线的k为0.1 b为0.2得到y_data

# 构造一个线性模型
b = tf.Variable(0.) # 初始化为0
k = tf.Variable(0.)
y = k * x_data + b

# MSE 的loss
loss = tf.reduce_mean(tf.square(y_data - y))
# 定义一个梯度下降法来进行训练的优化器
optimizer = tf.train.GradientDescentOptimizer(0.2)  # 这里的0.2是梯度下降的系数
# 最小化代价函数(训练的方式就是使loss值最小)
train = optimizer.minimize(loss)

# 初始化变量
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    for step in range(201):
        sess.run(train)
        if step % 20 == 0:
            print(step, sess.run([k, b]))
```

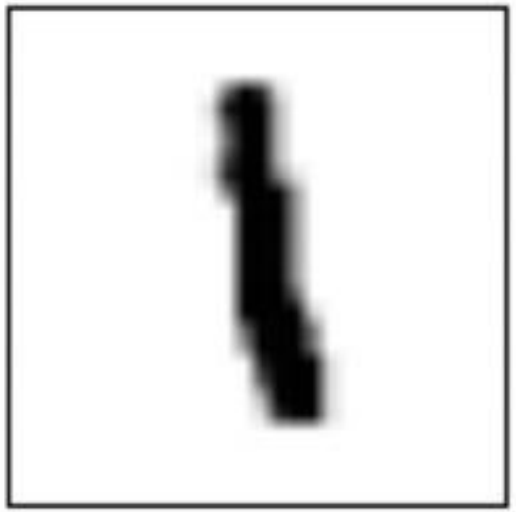
```
0 [0.044449475, 0.09701918]
20 [0.095833234, 0.20189862]
40 [0.097567454, 0.20110859]
60 [0.09857984, 0.2006472]
80 [0.099170886, 0.20037785]
100 [0.099515945, 0.20022058]
120 [0.099717416, 0.20012878]
140 [0.099835016, 0.20007518]
160 [0.09990368, 0.2000439]
180 [0.09994376, 0.20002563]
200 [0.09996716, 0.20001496]
```

深度学习项目-1

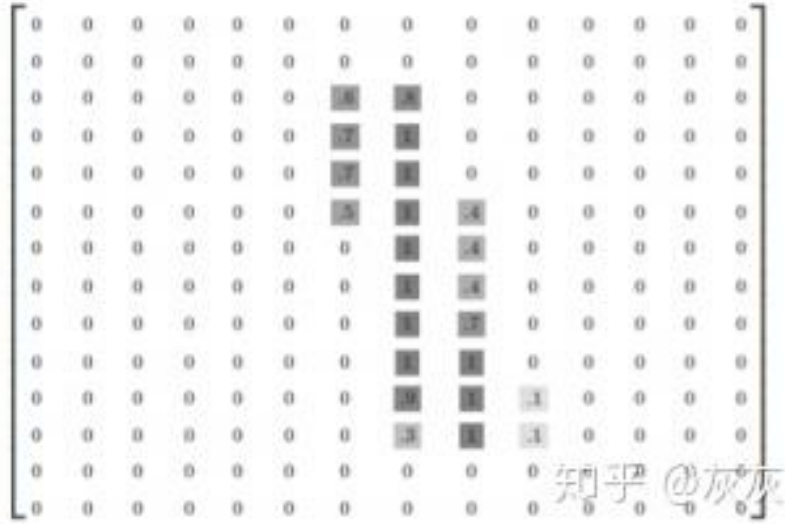
CV Project

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data", one_hot=True)
```



≃



```
# 为输入图像和目标输出类别创建节点
x = tf.placeholder(tf.float32, shape=[None, 784]) # 训练所需数据  占位符
y_ = tf.placeholder(tf.float32, shape=[None, 10]) # 训练所需标签数据  占位符

# 权重、偏置、卷积及池化操作初始化,以避免在建立模型的时候反复做初始化操作
def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1) # 取随机值，符合均值为0，标准差stddev为0.1
  return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)

# x 的第一个参数为图片的数量，第二、三个参数分别为图片高度和宽度，第四个参数为图片通道数。
# W 的前两个参数为卷积核尺寸，第三个参数为图像通道数，第四个参数为卷积核数量
# strides为卷积步长，其第一、四个参数必须为1，因为卷积层的步长只对矩阵的长和宽有效
# padding表示卷积的形式，即是否考虑边界。"SAME"是考虑边界，不足的时候用0去填充周围，"VALID"则不考虑
def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

# x 参数的格式同tf.nn.conv2d中的x，ksize为池化层过滤器的尺度，strides为过滤器步长
def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

设置网络结构

```
#把x更改为4维张量，第1维代表样本数量，第2维和第3维代表图像长宽，  第4维代表图像通道数
x_image = tf.reshape(x, [-1,28,28,1]) # -1表示任意数量的样本数,大小为28x28，深度为1的张量
# 第一层：卷积
W_conv1 = weight_variable([5, 5, 1, 32]) # 卷积在每个5x5的patch中算出32个特征。
b_conv1 = bias_variable([32])
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
# 第二层：池化
h_pool1 = max_pool_2x2(h_conv1)
# 第三层：卷积
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
# 第四层：池化
h_pool2 = max_pool_2x2(h_conv2)
# 第五层：全连接层
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])
h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
# 在输出层之前加入dropout以减少过拟合
keep_prob = tf.placeholder("float")
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
# 第六层：全连接层
W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])
# 第七层：输出层
y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

```
# ********** 训练和评估模型 ********** #
# 为训练过程指定最小化误差用的损失函数，即目标类别和预测类别之间的交叉熵
cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
# 使用反向传播，利用优化器使损失函数最小化
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
# 检测我们的预测是否真实标签匹配(索引位置一样表示匹配)
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
# 统计测试准确率， 将correct_prediction的布尔值转换为浮点数来代表对、错，并取平均值。
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
saver = tf.train.Saver() # 定义saver
# ********** 开始训练模型 ********** #
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(1000):
      batch = mnist.train.next_batch(50)
      if i%100 == 0:
        train_accuracy = accuracy.eval(feed_dict={x:batch[0], y_: batch[1], keep_prob: 1.0})
        print("step %d, training accuracy %g"%(i, train_accuracy))
      # 训练模型
      train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
    saver.save(sess, './save/model.ckpt') #模型储存位置
    print("test accuracy %g"%accuracy.eval(feed_dict={x: mnist.test.images [0:2000], y_:
mnist.test.labels [0:2000], keep_prob: 1.0}))
```

```
img = Image.open('./images/3.png')
reIm = img.resize((28, 28), Image.ANTIALIAS)
im_arr = np.array(reIm.convert('L'))
# 对图片做二值化处理（这样以滤掉噪声，另外调试中可适当调节阈值）
threshold = 50
# 模型的要求是黑底白字，但输入的图是白底黑字，所以需要对每个像素点的值改为255减去原值以得到互补的反色。
for i in range(28):
    for j in range(28):
        im_arr[i][j] = 255 - im_arr[i][j]
        if (im_arr[i][j] < threshold):
            im_arr[i][j] = 0
        else:
            im_arr[i][j] = 255
# 接着让现有的RGB图从0-255之间的数变为0-1之间的浮点数（因为要求像素点是0-1 之间的浮点数）
img_ready = np.multiply(im_arr, 1.0 / 255.0)
# 把图片形状拉成1行784列
result = img_ready.reshape([784])
. . . 模型结构相同 . . .
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    saver.restore(sess, "./save/model.ckpt")#这里使用了之前保存的模型参数
    prediction = tf.argmax(y_conv,1)
    predint = prediction.eval(feed_dict={x: [result],keep_prob: 1.0},
session=sess)
    print("recognize result: %d" %predint[0])
```

```
recognize result: 3
```

深度学习项目-2

NLP Project

文本分类

文本数据

Id sentiment review

"6666_4“ 0 "I really wanted to like this film, but the story is ridicules. I don't want to spoil this film, - don't worry right from the begin you know something bad is going to happen - but here's an example of how sloppy this film was put together. The Cowboy and \"Twig\" ride up the ridge. The Cowboy has a handle bar mustache. The Cowboy and \"Twig\" get into a shoot out and race half way down the ridge. The Cowboy is clean shaven through out the rest of the film. Sometime between the gun fight and the ride down the mountain the cowboy has had time to shave, in dark, on the back of a horse.

To be fair, the acting by the four main characters is solid."

*"10064_10“ 1 "The King of Masks is a beautifully told story that pits the familial gender preference towards males against human preference for love and companionship. Set in 1930s China during a time of floods, we meet Wang, an elderly street performer whose talents are magical and capture the awe of all who witness him. When a famous operatic performer sees and then befriends Wang, he invites Wang to join their troupe. However, we learn that Wang's family tradition allows him only to pass his secrets to a son. Learning that Wang is childless, Wang is encouraged to find an heir before the magic is lost forever. Taking the advice to heart, Wang purchases an 8 year old to fulfill his legacy; he would teach his new son, Doggie, the ancient art of silk masks. Soon, Wang discovers a fact about Doggie that threatens the rare and dying art.

Together, Wang and Doggie create a bond and experience the range of emotions that invariably accompany it. The story is absorbing. The setting is serene and the costuming simple. Summarily, it is an International Award winning art film which can't help but to move and inspire."*

```
train_clean, train_labels = clean_text(train.values, True) # 文本清理 去除无效字符

# tokenizer give each word a unique id
tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_clean)
word_index = tokenizer.word_index  # final id
train_seq = tokenizer.texts_to_sequences(train_clean)  # convert dataset to ids

max_review_length = 200
train_pad = pad_sequences(train_seq, maxlen=max_review_length)  # padded with max length 默认为前向填充
review_lengths_longer_than_pad = 0
for seq in train_seq:  # calculate how many reviews longer than pad length
    if len(seq) > max_review_length:
        review_lengths_longer_than_pad = review_lengths_longer_than_pad + 1
x_train, x_test, y_train, y_test = train_test_split(train_pad, train_labels, test_size=0.20, random_state=2)
x_test, x_valid, y_test, y_valid = train_test_split(x_test, y_test, test_size=0.20, random_state=2)

# RESULT DEMO
With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there,

stuff going moment mj ve started listening music watching odd documentary,

[415, 75, 431, 8839, 50, 519, 2478, 122, 56, 889, 528, 187, 18714, 187, 11270, 173, 83, 14, 676, 2479, 124, 97, 10, 511, 4097, 173, 22, 219,
595, 2354, 1210, 11270, 76, 4849, 76, 650, 2, 262, 75, 11, 312, 1682, 499, 1158, 3286, 8839, 422, 809, 3363, 17, 452, 614, 1519, 15, 52,
4446, 1871, 1013, 154, 353, 1459, 759, 2443, 4,,,, 1374]
[   10   511  4097   173    22   219   595  2354  1210 11270    76  4849
    76   650     2   262    75    11   312  1682   499  1158  3286  8839
   422   809  3363    17   452   614  1519    15    52  4446  1871  1013
   154   353  1459   759  2443     4  8839   429    75   652    74   245
   101  7449   614  3460  8839 37276  1883     1   136   353  1459   256
     3   880    16    42  1506  1012  2354    12   561    50   397   733
  6944    12    41    16   166   373  4416  3408    41    92   235   449
   216   263   124     3 18716 18717   327  1374]
```

```
# 构造模型
datas_placeholder = tf.placeholder(tf.int32, [None, max_document_length])
labels_placeholder = tf.placeholder(tf.int32, [None])

# 建立embeddings矩阵
embeddings = tf.get_variable("embeddings", [vocab_size, embedding_size],
initializer=tf.truncated_normal_initializer)
# 将词索引号转换为词向量[None, max_document_length] => [None, max_document_length,
embedding_size]
embedded = tf.nn.embedding_lookup(embeddings, datas_placeholder)
# 转换为LSTM的输入格式，要求是数组，数组的每个元素代表某个时间戳一个Batch的数据
rnn_input = tf.unstack(embedded, max_document_length, axis=1)

# 定义LSTM，20为输出神经元数量
lstm_cell = BasicLSTMCell(20, forget_bias=1.0)
rnn_outputs, rnn_states = static_rnn(lstm_cell, rnn_input, dtype=tf.float32)

#利用LSTM最后的输出进行预测，取最后一个时间戳的输出神经元，在其上加全连接层
logits = tf.layers.dense(rnn_outputs[-1], num_classes)

predicted_labels = tf.argmax(logits, axis=1)
```

```
# 定义损失和优化器
losses= tf.nn.softmax_cross_entropy_with_logits(
    labels=tf.one_hot(labels_placeholder, num_classes),
    logits=logits)
mean_loss = tf.reduce_mean(losses)
optimizer = tf.train.AdamOptimizer(learning_rate=1e-2).minimize(mean_loss)
with tf.Session() as sess:
    # 初始化变量
    sess.run(tf.global_variables_initializer())
    # 定义要填充的数据
    feed_dict = {
        datas_placeholder: datas,
        labels_placeholder: labels}
    print("开始训练")
    for step in range(3000):
        _, mean_loss_val = sess.run([optimizer, mean_loss], feed_dict=feed_dict)
        if step % 10 == 0:
            print("step = {}\tmean loss = {}".format(step, mean_loss_val))
    print("训练结束，进行预测")
    predicted_labels_val = sess.run(predicted_labels, feed_dict=feed_dict)
```

```
Use standard file APIs to check for files with this prefix.
Predicted 'truly masterful movie' as:
Good movie

Process finished with exit code 0
```

THANK YOU

欢迎各位老师指导

Q&A