EDU

CSDN学院　IT实战派

Python 深度学习

深度学习基础篇：神经网络基本概念与发展

讲师： 彭靖田

本章目录

- 激荡63年：人工智能简史
- 脱颖而出的深度学习
- 神经网络是“照猫画虎”
- 神经网络的血液：张量
- 神经网络的齿轮：张量运算
- 神经网络的引擎：梯度优化

EDU

CSDN学院　IT实战派

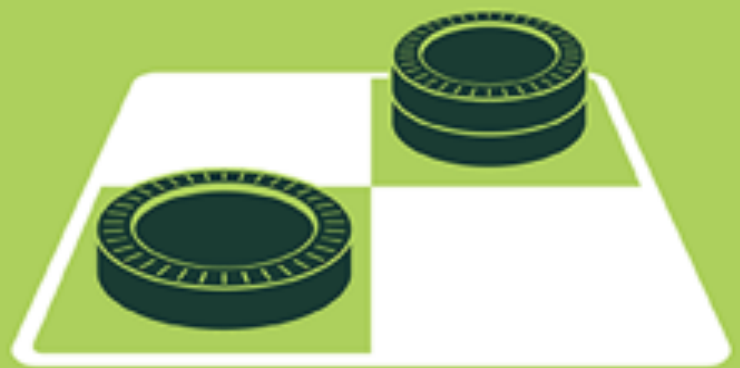
Python 深度学习

激荡63年：人工智能简史

讲师： 彭靖田

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



MACHINE LEARNING

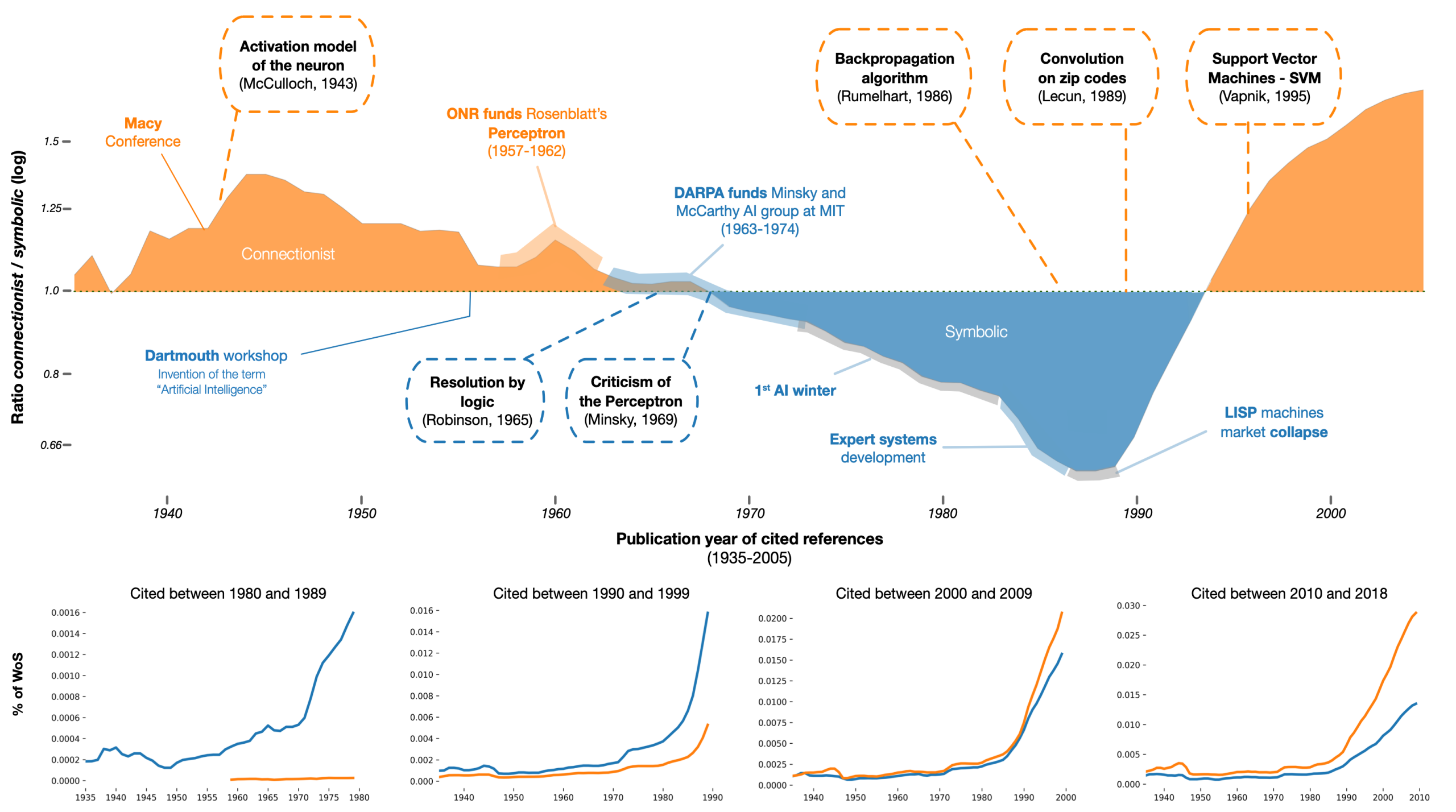
Machine learning begins to flourish.

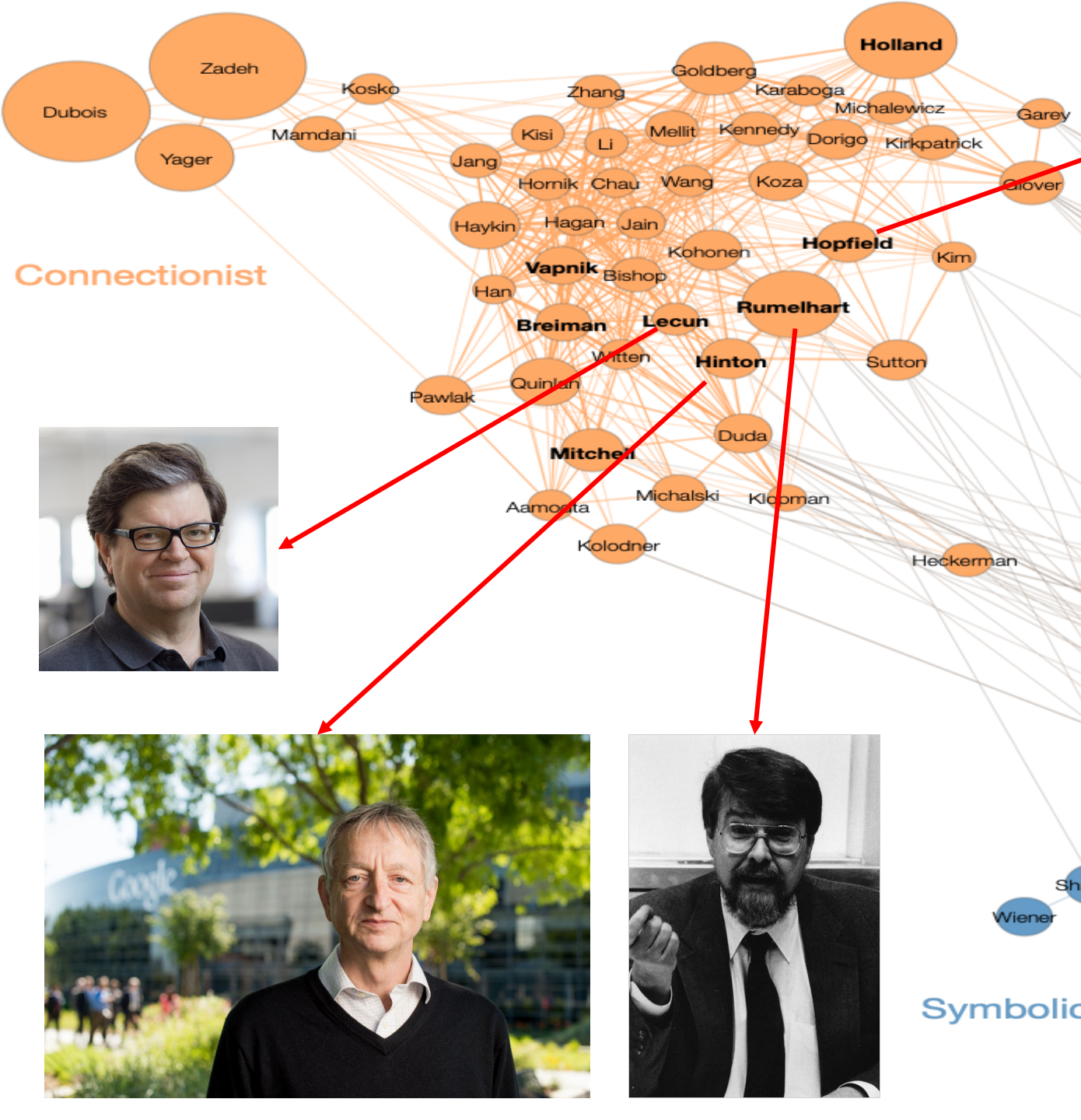
DEEP LEARNING

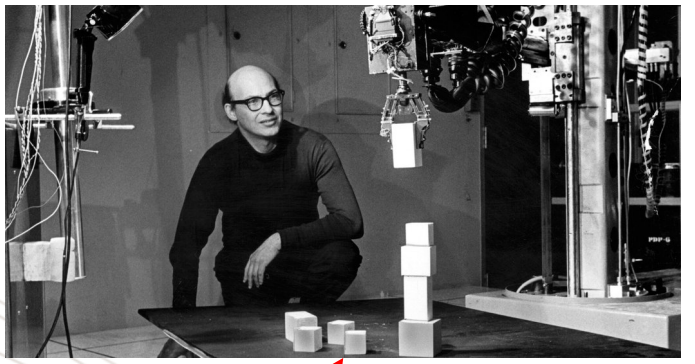
Deep learning breakthroughs drive AI boom.

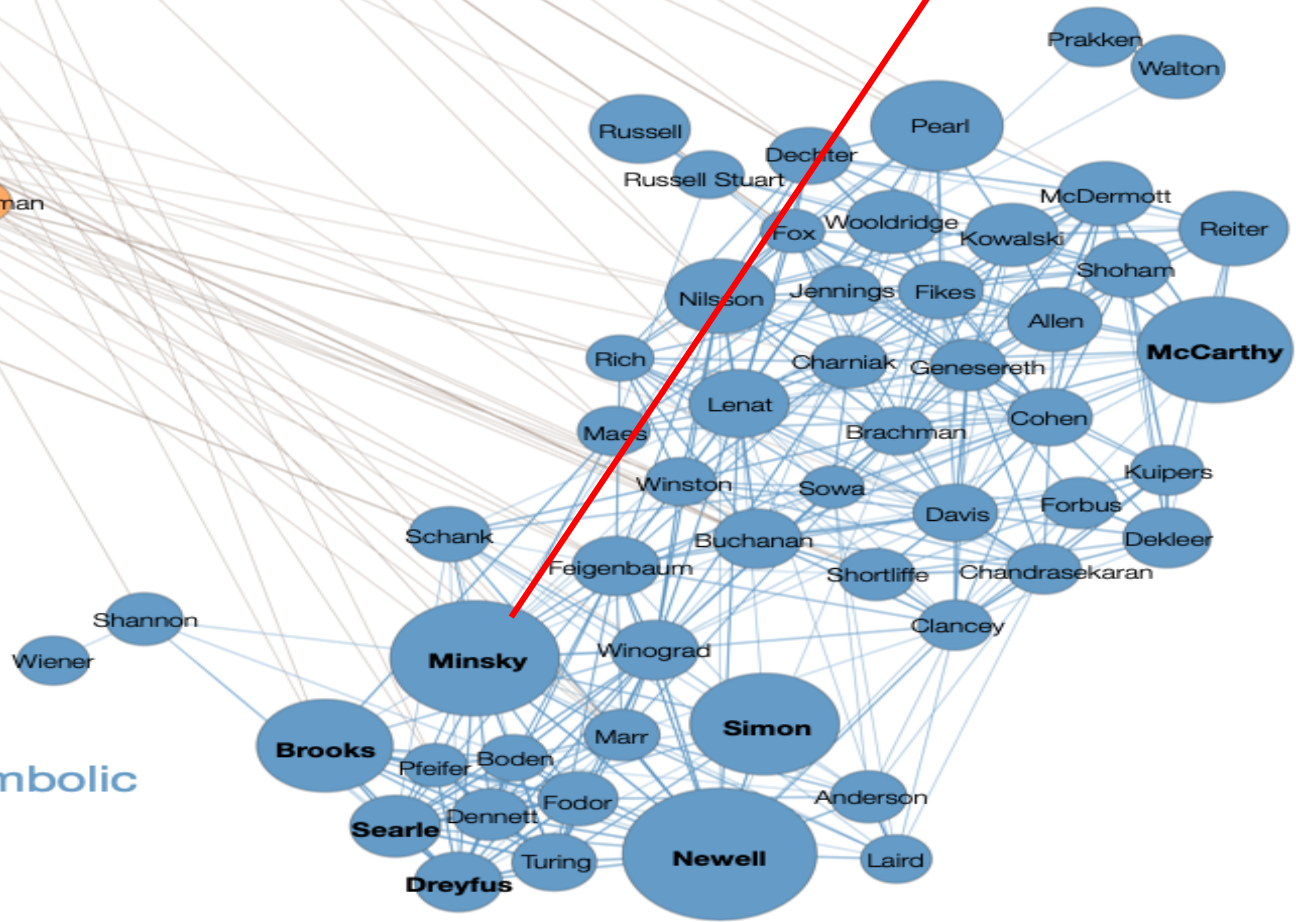
1950's 1960's 1970's 1980's 1990's 2000's 2010's

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.


Activation model of the neuron (McCulloch, 1943)
Macy Conference
ONR funds Rosenblatt's Perceptron (1957-1962)
Backpropagation algorithm (Rumelhart, 1986)
Convolution on zip codes (Lecun, 1989)
Support Vector Machines - SVM (Vapnik, 1995)
DARPA funds Minsky and McCarthy AI group at MIT (1963-1974)
Connectionist
Symbolic
Dartmouth workshop
Invention of the term "Artificial Intelligence"
Resolution by logic (Robinson, 1965)
Criticism of the Perceptron (Minsky, 1969)
1st AI winter
Expert systems development
LISP machines market collapse
Ratio connectionist / symbolic (log)
1.5
1.25
1.0
0.8
0.66
1940
1950
1960
1970
1980
1990
2000
Publication year of cited references
(1935-2005)
% of WoS
Cited between 1980 and 1989
Cited between 1990 and 1999
Cited between 2000 and 2009
Cited between 2010 and 2018


Zadeh
Dubois
Yager
Kosko
Mamdani
Holland
Goldberg
Zhang
Karaboga
Michalewicz
Garey
Kisi
Li
Mellit
Kennedy
Dorigo
Kirkpatrick
Jang
Hornik
Chau
Wang
Koza
Glover
Haykin
Hagan
Jain
Kohonen
Hopfield
Kim
Connectionist
Vapnik
Bishop
Han
Rumelhart
Breiman
Lecun
Witten
Hinton
Sutton
Quinlan
Pawlak
Duda
Mitchell
Aamodta
Michalski
Klopman
Kolodner
Heckerman




Prakken
Walton
Russell
Pearl
Dechter
Russell Stuart
McDermott
Fox
Wooldridge
Kowalski
Reiter
Shoham
Nilsson
Jennings
Fikes
Allen
McCarthy
Rich
Charniak
Genesereth
Lenat
Maes
Brachman
Cohen
Kuipers
Winston
Sowa
Davis
Forbus
Schank
Buchanan
Dekleer
Feigenbaum
Shortliffe
Chandrasekaran
Clancey
Shannon
Wiener
Minsky
Winograd
Simon
Brooks
Marr
Pfeifer
Boden
Anderson
Fodor
Searle
Dennett
Newell
Laird
Turing
Dreyfus
Symbolic


EDU
CSDN学院 IT实战派

EDU

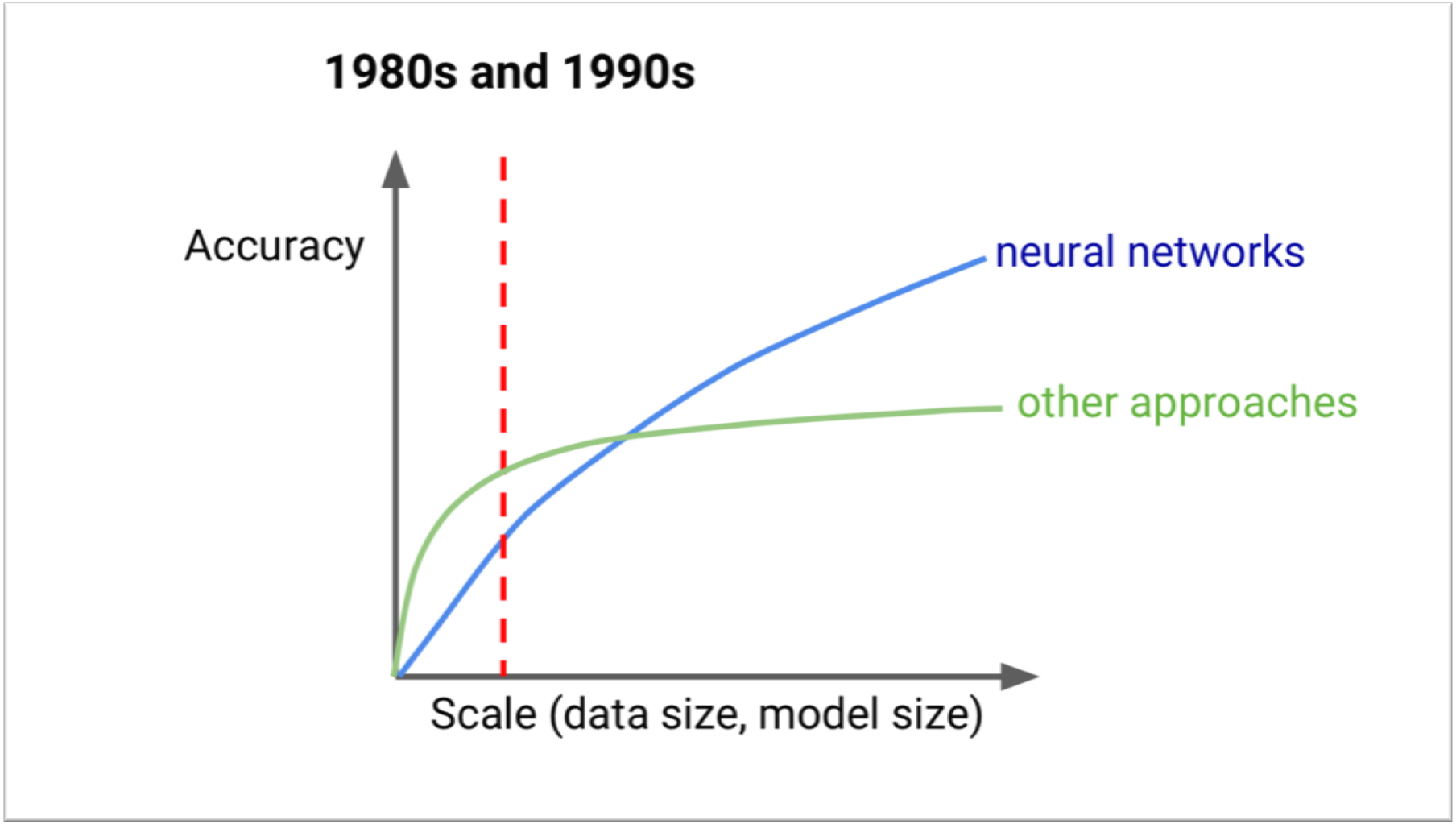
CSDN学院　IT实战派

Python 深度学习

脱颖而出的深度学习

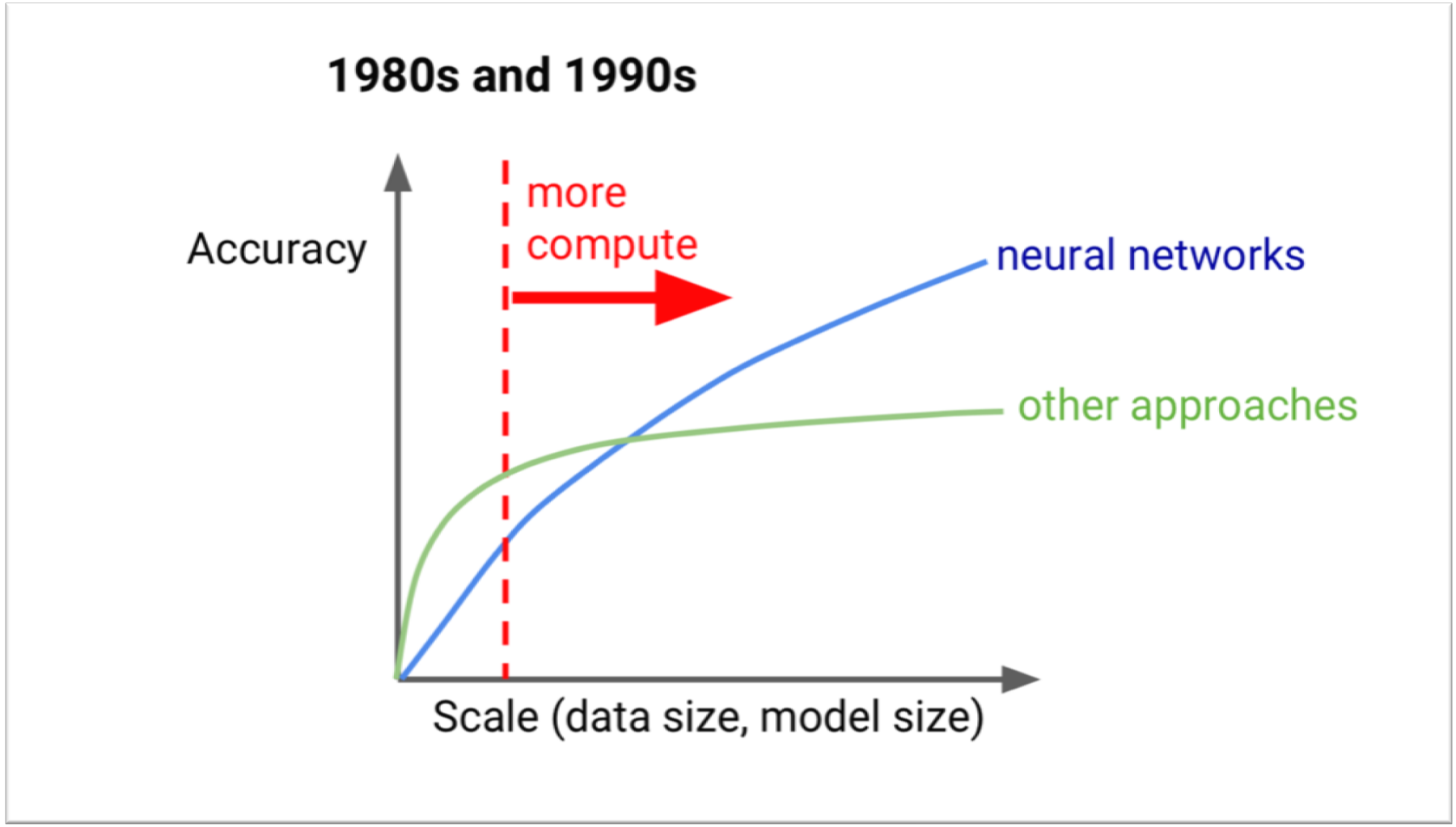
讲师： 彭靖田

1980s：感知机与特征工程



Jeff Dean, Google Brain Team, Building Intelligent Systems with Large Scale Deep Learning

1990s：神经网络在图像和语音领域发力



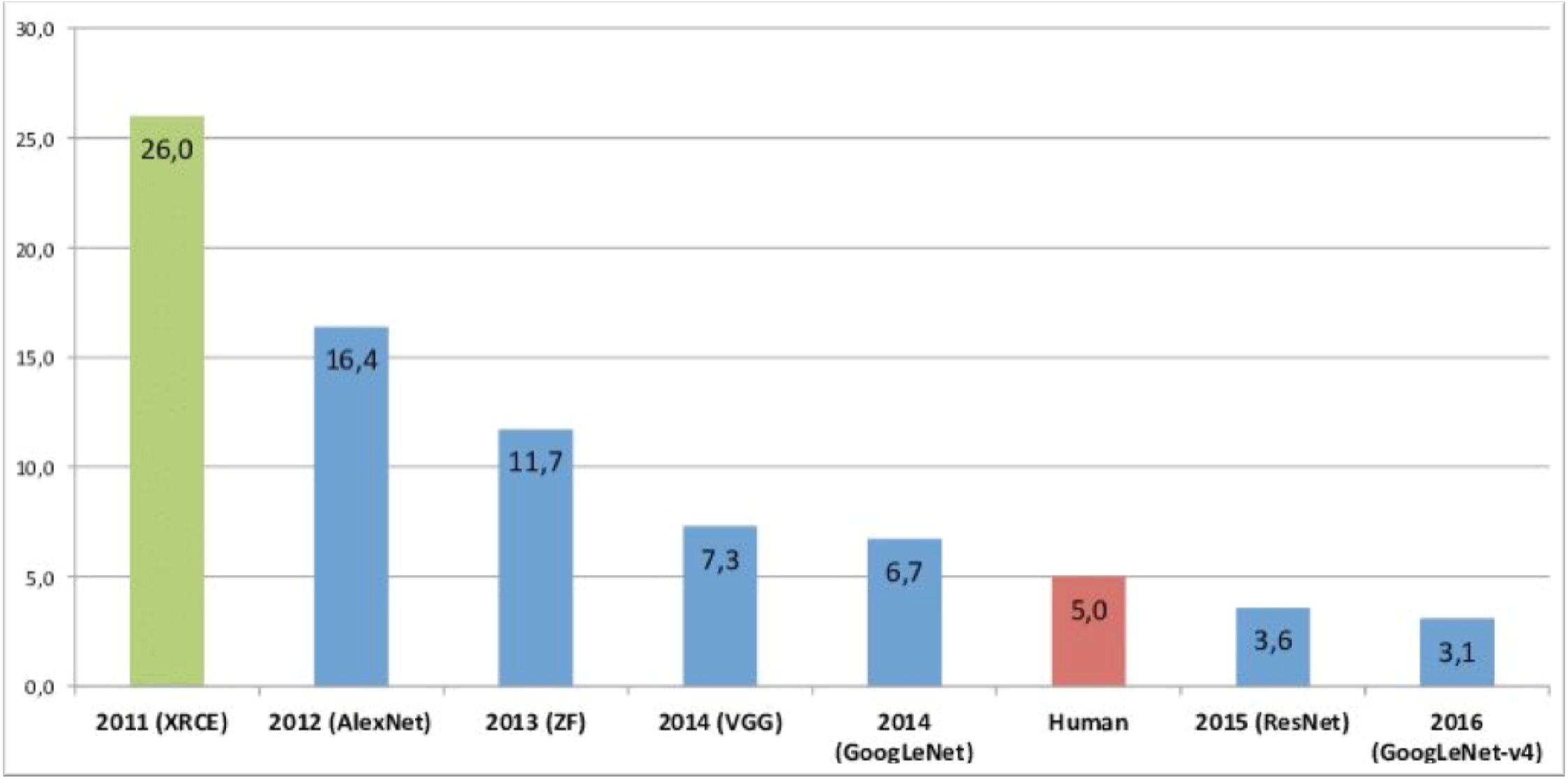
Jeff Dean, Google Brain Team, Building Intelligent Systems with Large Scale Deep Learning


Compute

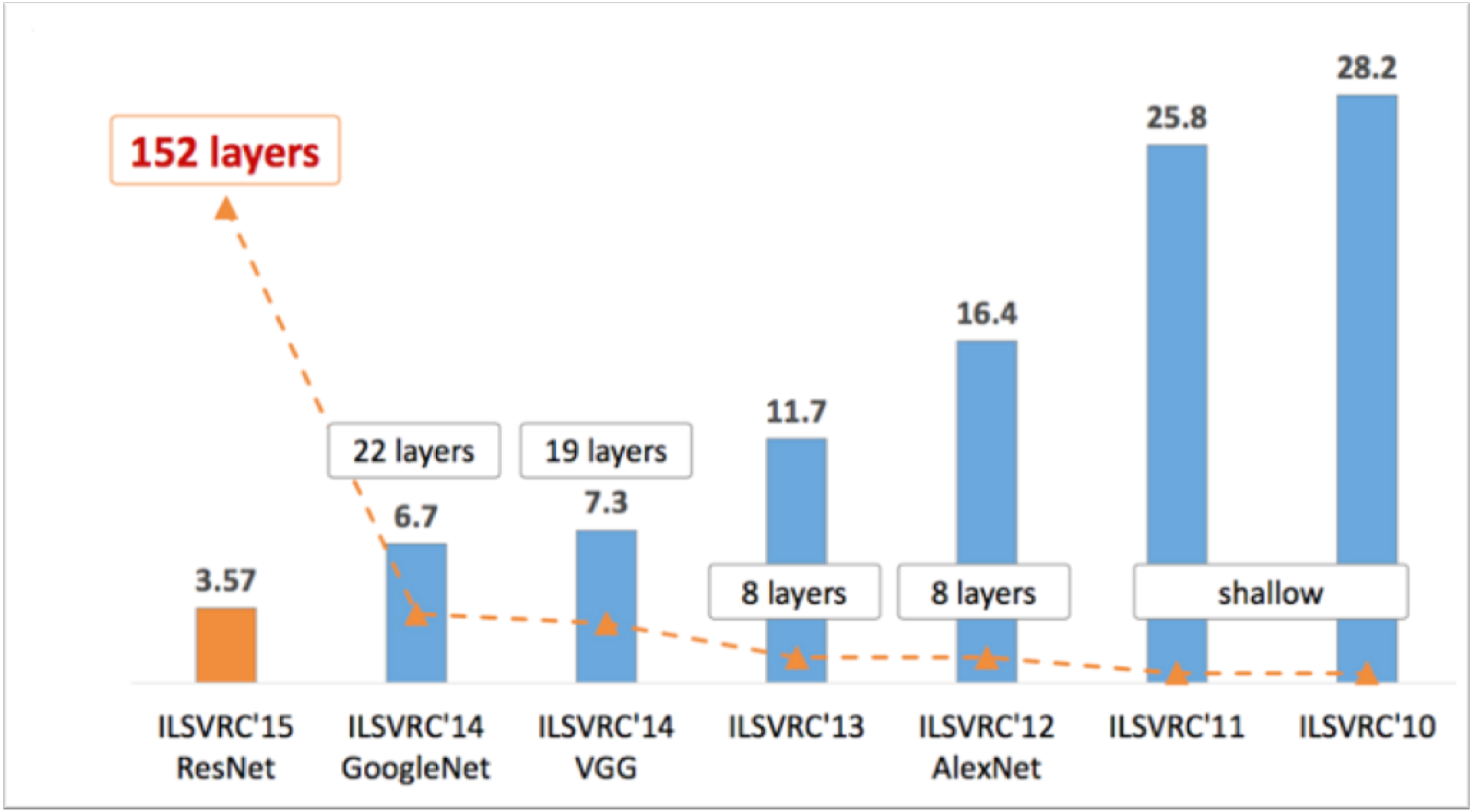

Animal
Skyscraper
Aircraft
Data

IMAGENET 数据集与竞赛

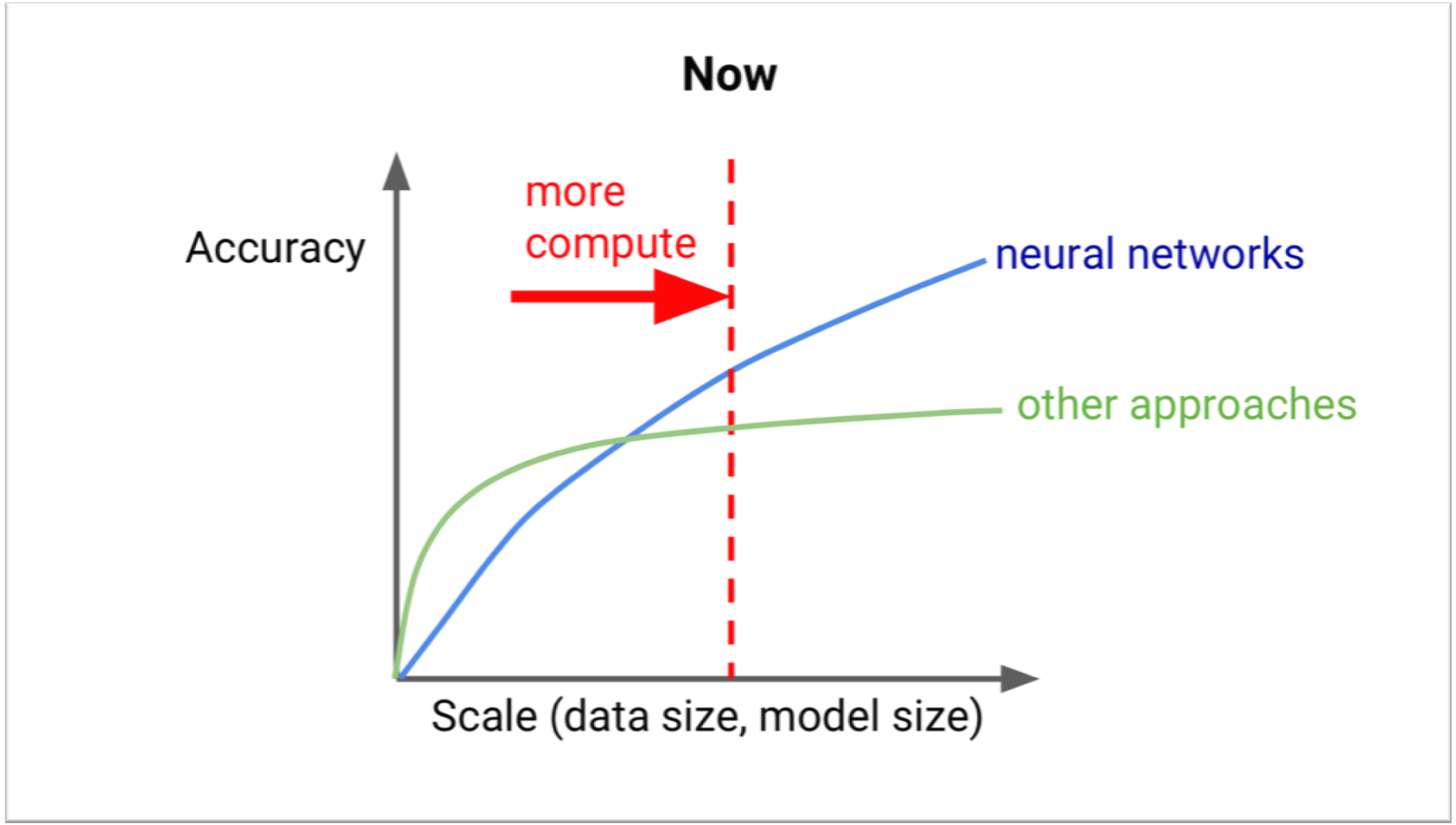
IMAGENET 分类任务错误率（Top 5）



IMAGENET 分类任务 模型大小



算力和数据增长推动深度学习的繁荣发展



Jeff Dean, Google Brain Team, Building Intelligent Systems with Large Scale Deep Learning


EDU
CSDN学院 IT实战派

EDU

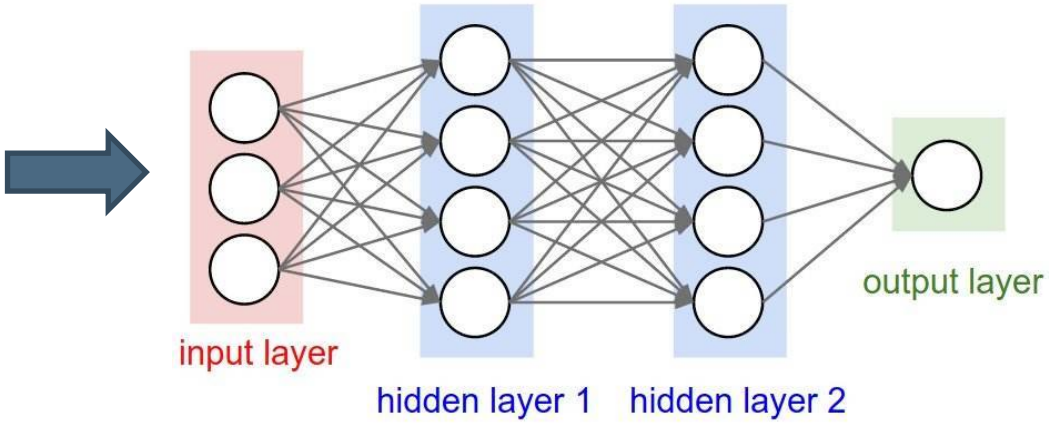
CSDN学院　IT实战派

Python 深度学习

神经网络是“照猫画虎”

讲师： 彭靖田

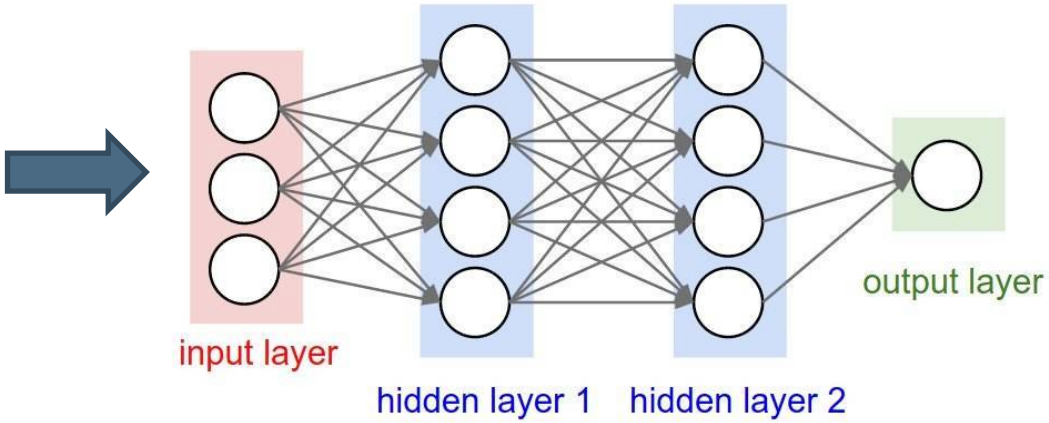
“照猫画虎”学习法 – 训练



虎
（预测）

猫
（标签）

“照猫画虎”学习法 – 训练

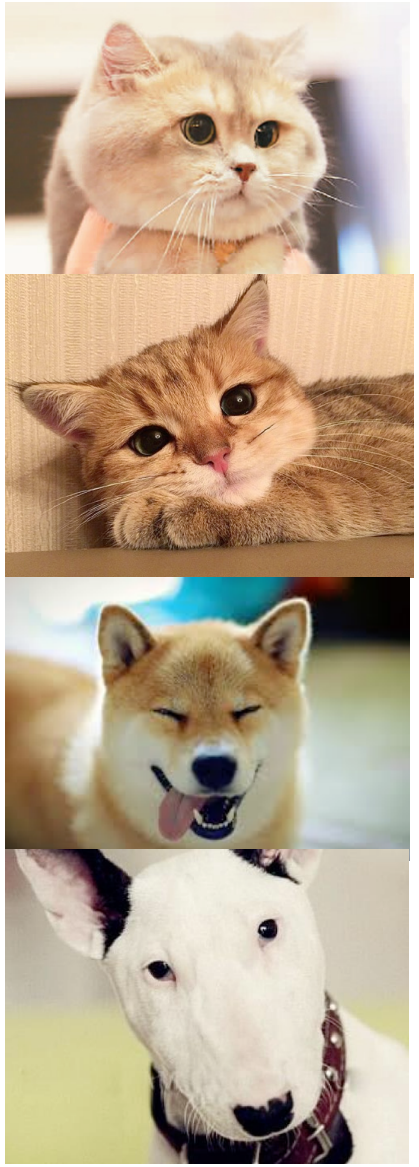


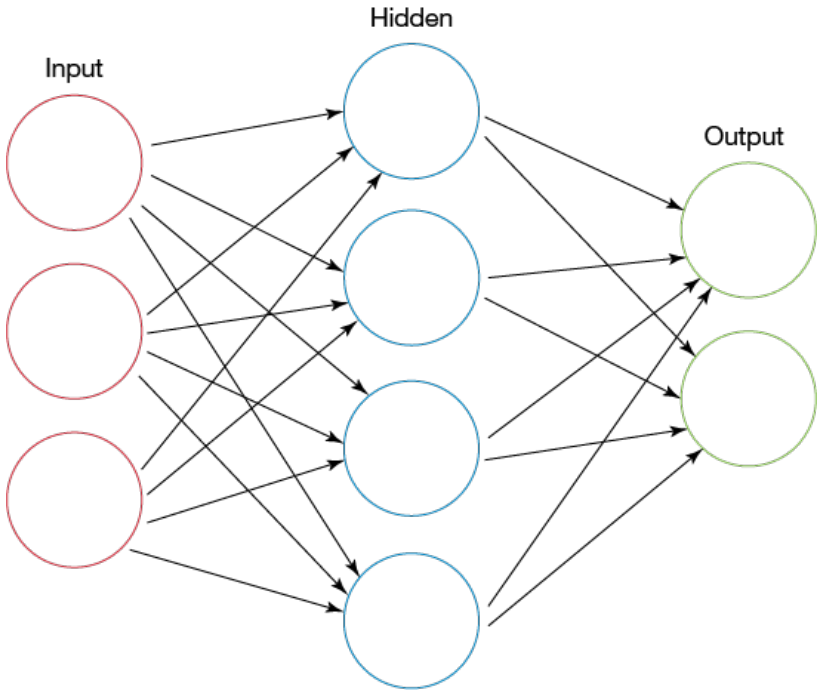
猫
（预测）

猫
（标签）

√

“照猫画虎”学习法 – 训练

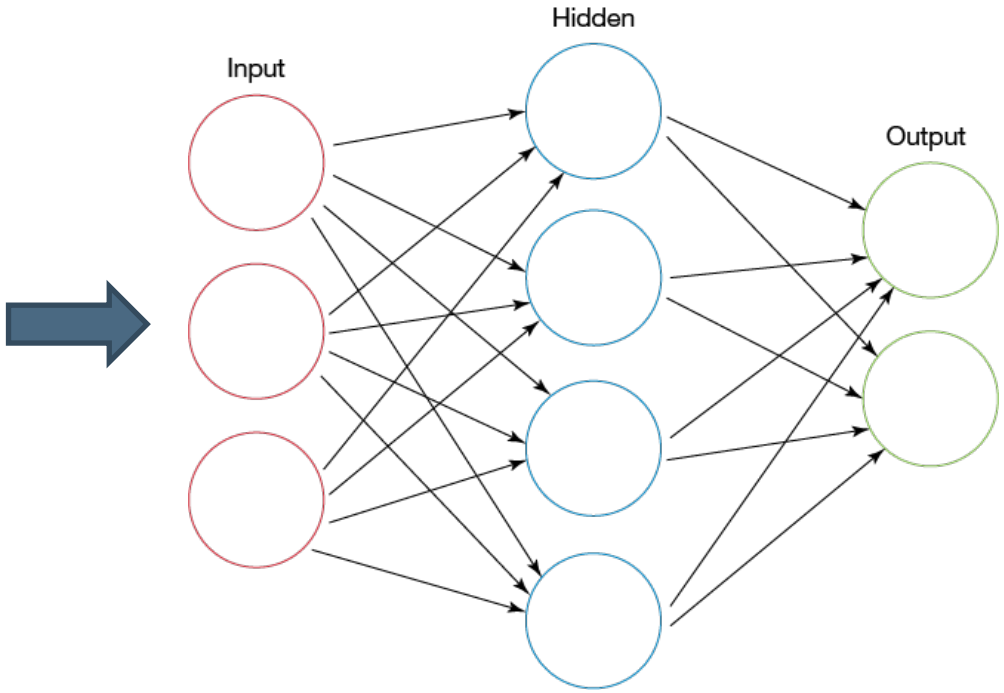




猫（预测） 猫（标签） √

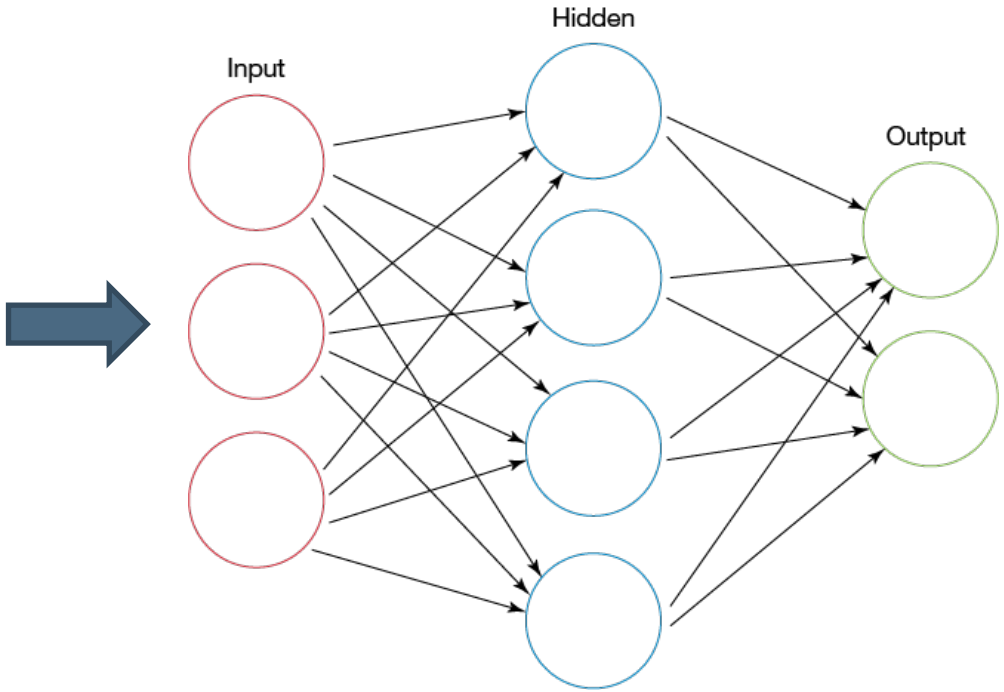
狗（预测） 狗（标签） √

“照猫画虎”学习法 – 推理



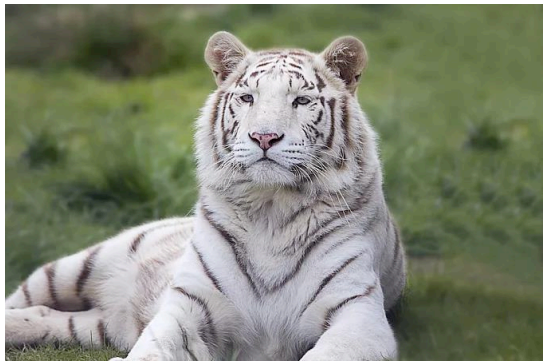
猫 √

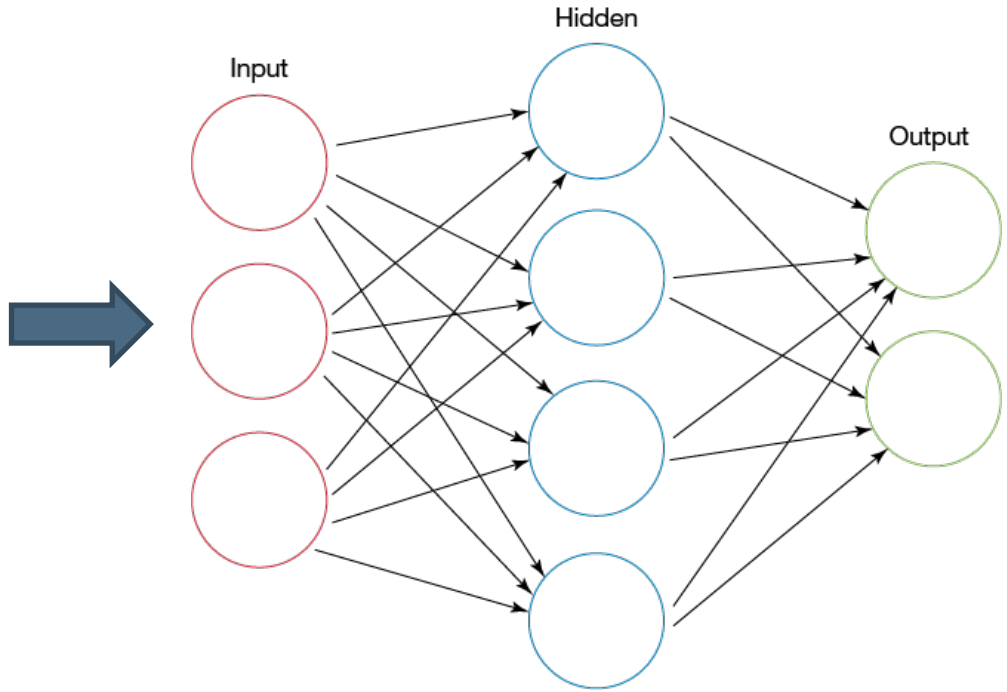
“照猫画虎”学习法 – 推理



猫

“照猫画虎”学习法 – 推理





猫


EDU
CSDN学院 IT实战派

EDU

CSDN学院　IT实战派

Python 深度学习

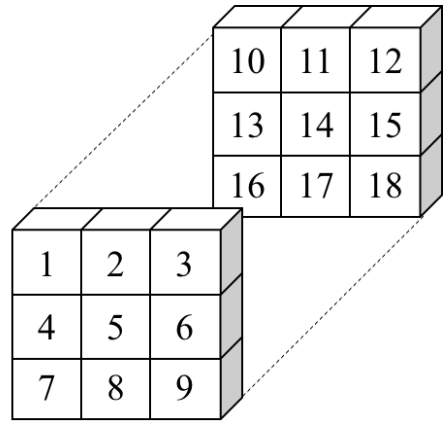
神经网络的血液：张量

讲师： 彭靖田

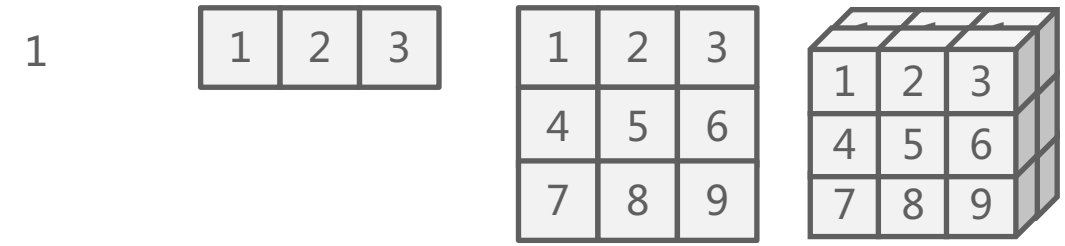
张量

在数学里，张量是一种几何实体，广义上表示任意形式的“数据”。张量可以理解为标量、向量和矩阵在高维空间上的推广。

阶（Dim）	数据实体	Python 样例
0	标量	scalar = 1
1	向量	vector = [1, 2, 3]
2	矩阵（数据表）	matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
3	数据立方	tensor = [[[1], [2], [3]], [[4], [5], [6]], [[7], [8], [9]],...]
n	n阶张量	....



3阶



0阶　1阶　2阶　3阶

张量

标量（0D张量）

```
In [1]: import numpy as np

In [2]: x = np.array(1)

In [3]: x
Out[3]: array(1)

In [4]: x.ndim
Out[4]: 0
```

向量（1D张量）

```
In [5]: y = np.array([1, 2, 3])

In [6]: y
Out[6]: array([1, 2, 3])

In [7]: y.ndim
Out[7]: 1
```

矩阵（2D张量）

```
In [8]: z = np.array([[1,2,3],
                      [4,5,6],
                      [7,8,9]])

In [9]: z
Out[9]: array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]])

In [10]: z.ndim
Out[10]: 2
```

3D张量与更高维度张量

```
In [8]: z = np.array([[1,2,3],
                      [4,5,6],
                      [7,8,9]])

In [9]: z

Out[9]: array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]])

In [10]: z.ndim

Out[10]: 2
```

```
In [11]: n_z = np.array([z, z*2, z*3])

In [12]: n_z

Out[12]: array([[[ 1,  2,  3],
                 [ 4,  5,  6],
                 [ 7,  8,  9]],

                [[ 2,  4,  6],
                 [ 8, 10, 12],
                 [14, 16, 18]],

                [[ 3,  6,  9],
                 [12, 15, 18],
                 [21, 24, 27]]])

In [13]: n_z.ndim

Out[13]: 3
```

张量-关键属性

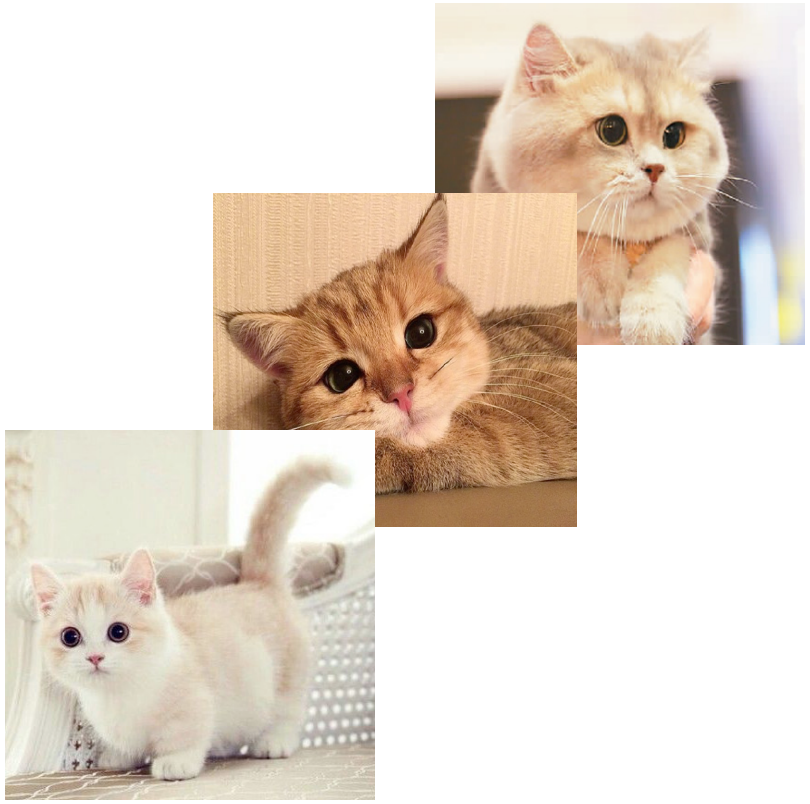
- 阶（rank）
- 形状（shape）
- 数据类型（dtype）

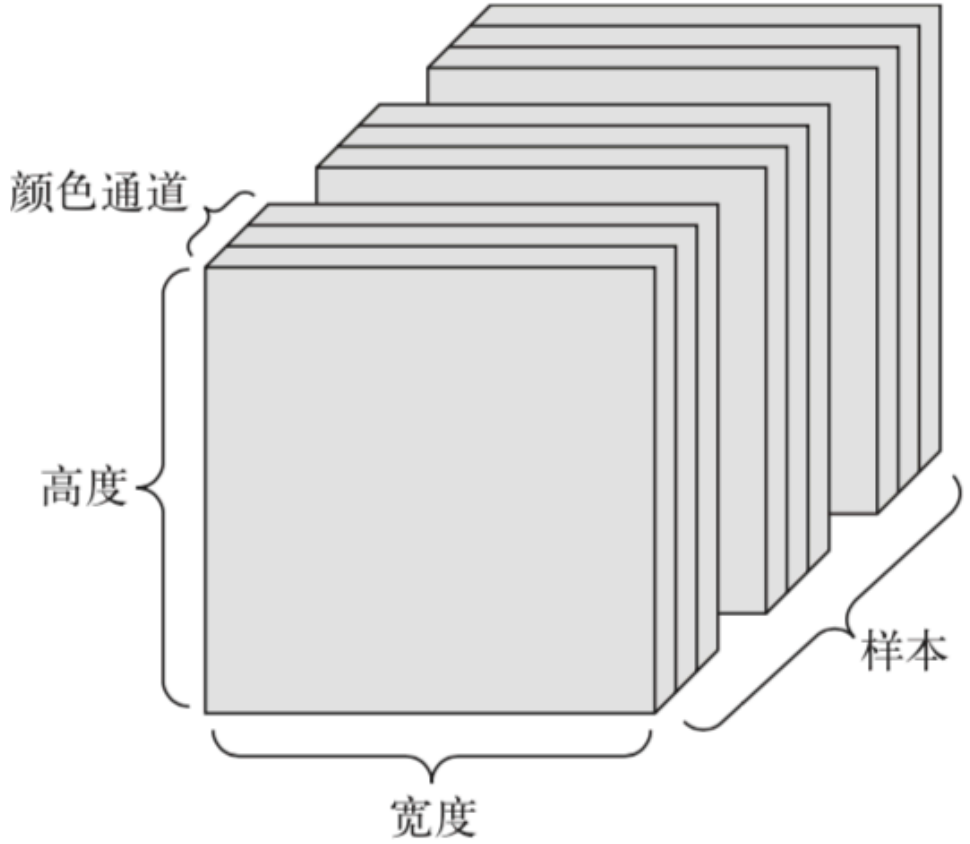
```
In [12]: n_z

Out[12]: array([[[ 1,  2,  3],
                 [ 4,  5,  6],
                 [ 7,  8,  9]],

                [[ 2,  4,  6],
                 [ 8, 10, 12],
                 [14, 16, 18]],

                [[ 3,  6,  9],
                 [12, 15, 18],
                 [21, 24, 27]]])

In [13]: n_z.ndim

Out[13]: 3

In [14]: n_z.shape

Out[14]: (3, 3, 3)

In [15]: n_z.dtype

Out[15]: dtype('int64')
```

深度学习处理的张量示例

数据类别	阶	形状
向量数据	2D张量	(samples, features)
时间序列数据	3D张量	(samples, timesteps, features)
图像	4D张量	(samples, height, width, channels)
视频	5D张量	(samples, frames, height, width, channels)

图像张量






EDU
CSDN学院 IT实战派

EDU

CSDN学院　IT实战派

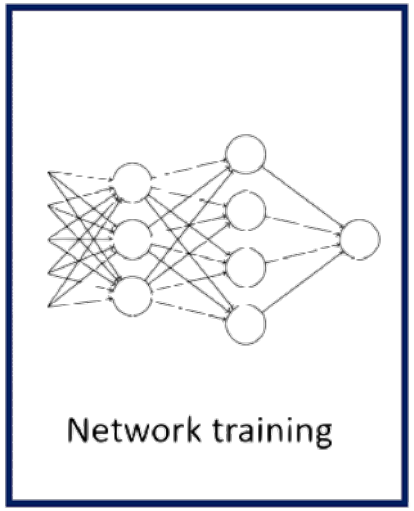
Python 深度学习

神经网络的齿轮：张量运算

讲师： 彭靖田

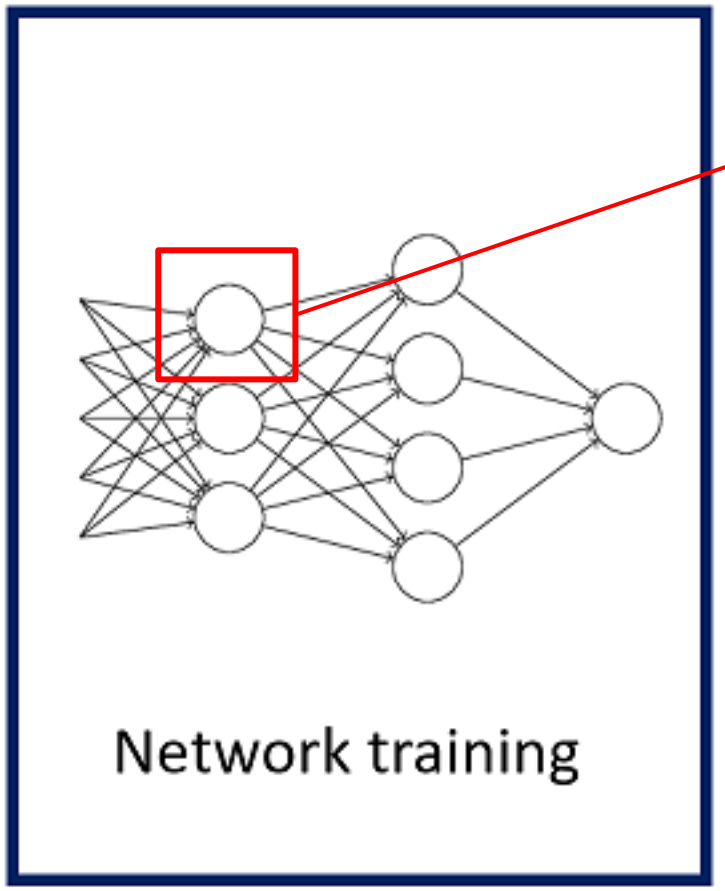
神经网络的计算

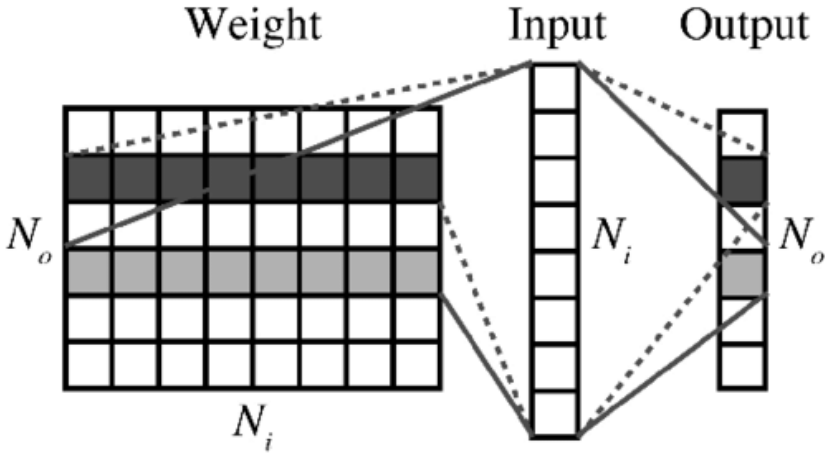
Data & Labels

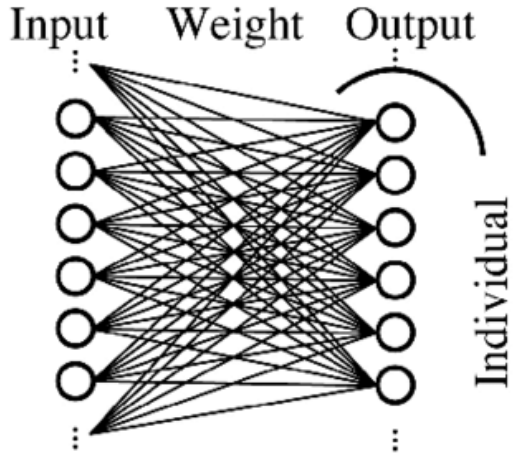


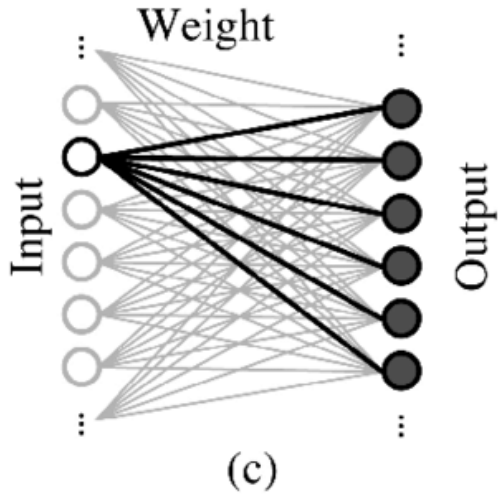
0
1
2
3
4
5
6
7
8
9

神经网络的计算


Network training

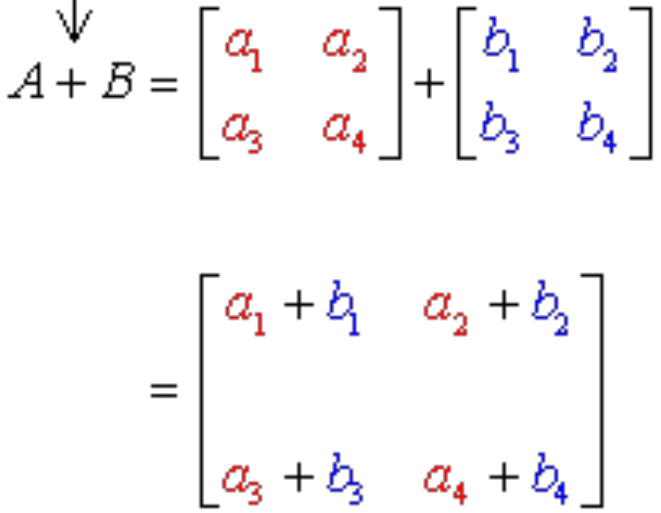

Weight
Input
Output
N_o
N_i
N_i
N_o
(a)


Input
Weight
Output
Individual
(b)


Weight
Input
Output
(c)

张量计算：逐元素运算

ADD



In [1]:
```
import numpy as np
```

In [2]:
```
def my_add(a, b):
    a = a.copy()
    for i in range(a.shape[0]):
        for j in range(a.shape[1]):
            a[i, j] += b[i, j]
    return a
```

In [3]:
```
a = np.array([[1,1],
              [1,1]])

b = np.array([[1,2],
              [3,4]])
```

In [4]:
```
my_add(a, b)
```

Out[4]:
```
array([[2, 3],
       [4, 5]])
```

张量计算：逐元素运算

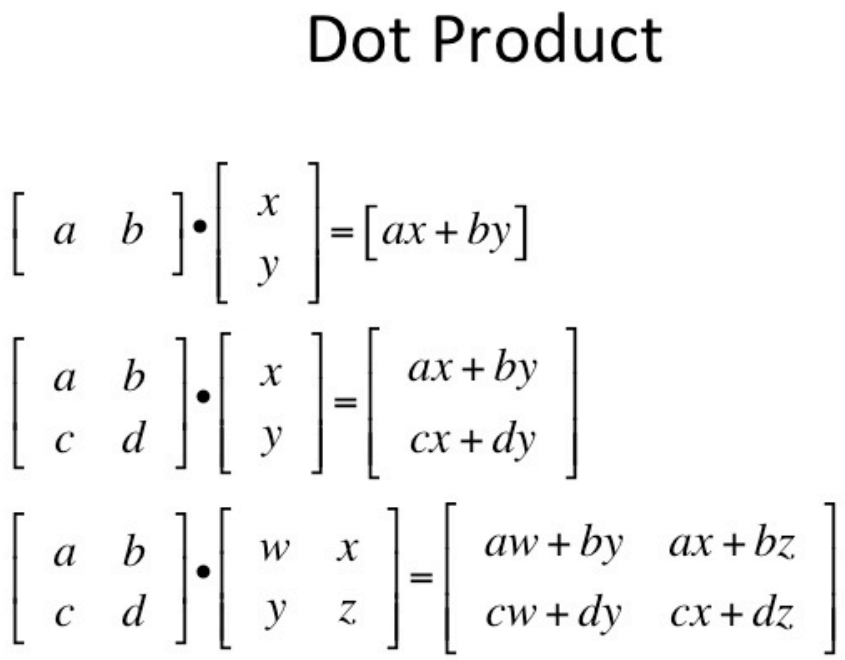
ADD

$$A + B = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} + \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix}$$

$$= \begin{bmatrix} a_1 + b_1 & a_2 + b_2 \\ a_3 + b_3 & a_4 + b_4 \end{bmatrix}$$

```
In [1]: import numpy as np

In [2]: def my_add(a, b):
            a = a.copy()
            for i in range(a.shape[0]):
                for j in range(a.shape[1]):
                    a[i, j] += b[i, j]
            return a

In [3]: a = np.array([[1,1],
                      [1,1]])

        b = np.array([[1,2],
                      [3,4]])

In [4]: my_add(a, b)

Out[4]: array([[2, 3],
               [4, 5]])

In [5]: a + b

Out[5]: array([[2, 3],
               [4, 5]])
```

张量计算：张量点积



In [9]:
```
a = np.array([1, 2])
b = np.array([3, 4])
# a*b = 1*3 + 2*4 = 11
np.dot(a, b)
```

Out[9]:
```
11
```

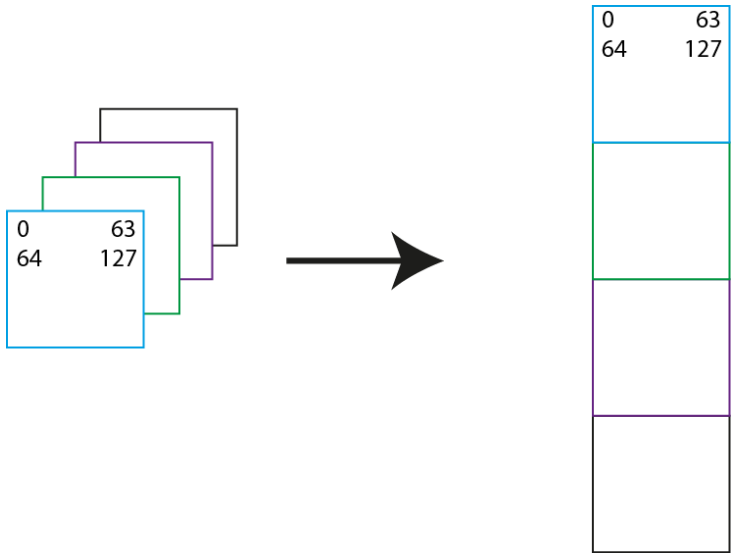
In [10]:
```
a = np.array([[1,1],
              [2,2]])
b = np.array([3,4])
# a*b = [1*3+1*4, 2*3+2*4] = [7, 14]
np.dot(a, b)
```

Out[10]:
```
array([ 7, 14])
```

In [11]:
```
a = np.array([[1,2],
              [3,4]])

b = np.array([[1,2],
              [3,4]])
# a*b = [[1*1+2*3, 1*2+2*4], = [[ 7, 10],
#        [3*1+4*3, 3*2+4*4]]    [15, 22]]
np.dot(a, b)
```

Out[11]:
```
array([[ 7, 10],
       [15, 22]])
```

张量运算：张量变形



$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}^{\mathrm{T}}$$

```
In [9]: a = np.array([[ 7, 10],
                      [15, 22]])

In [10]: a.shape

Out[10]: (2, 2)

In [11]: a = a.reshape(1, 4)
         a

Out[11]: array([[ 7, 10, 15, 22]])

In [12]: a = a.reshape(4,1)
         a

Out[12]: array([[ 7],
                [10],
                [15],
                [22]])

In [13]: # a[i, :] --- transpose ---> a[:, i]
         np.transpose(a)

Out[13]: array([[ 7, 10, 15, 22]])
```


EDU
CSDN学院 IT实战派

EDU

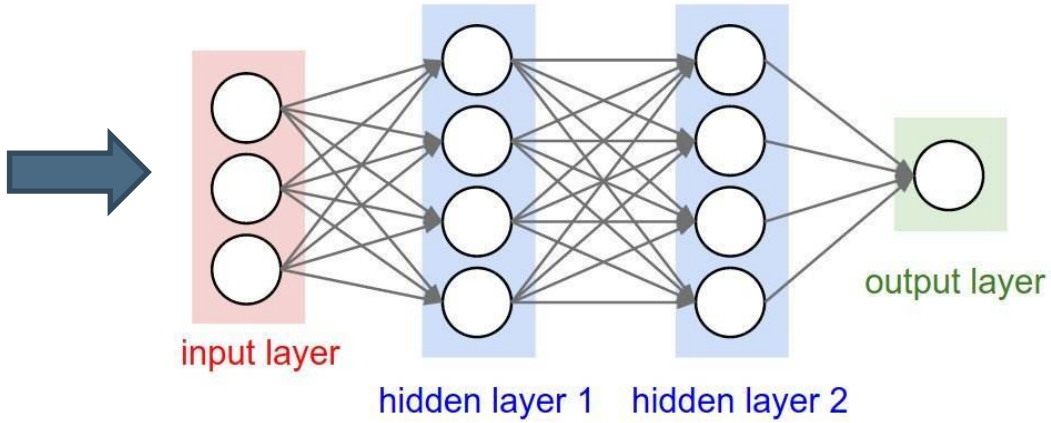
CSDN学院　IT实战派

Python 深度学习

神经网络的引擎：梯度优化

讲师： 彭靖田

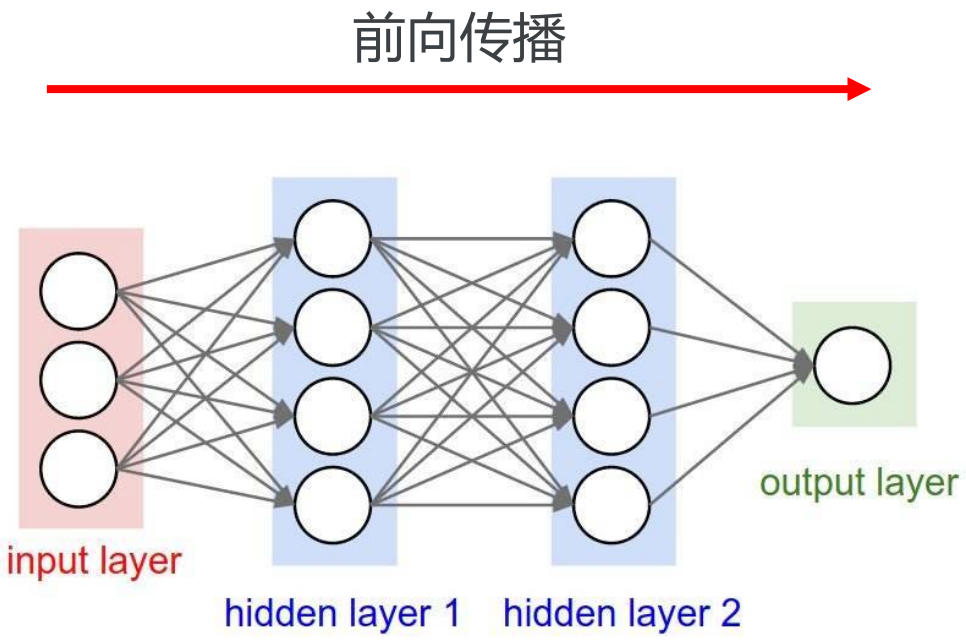
“照猫画虎”学习法 – 训练

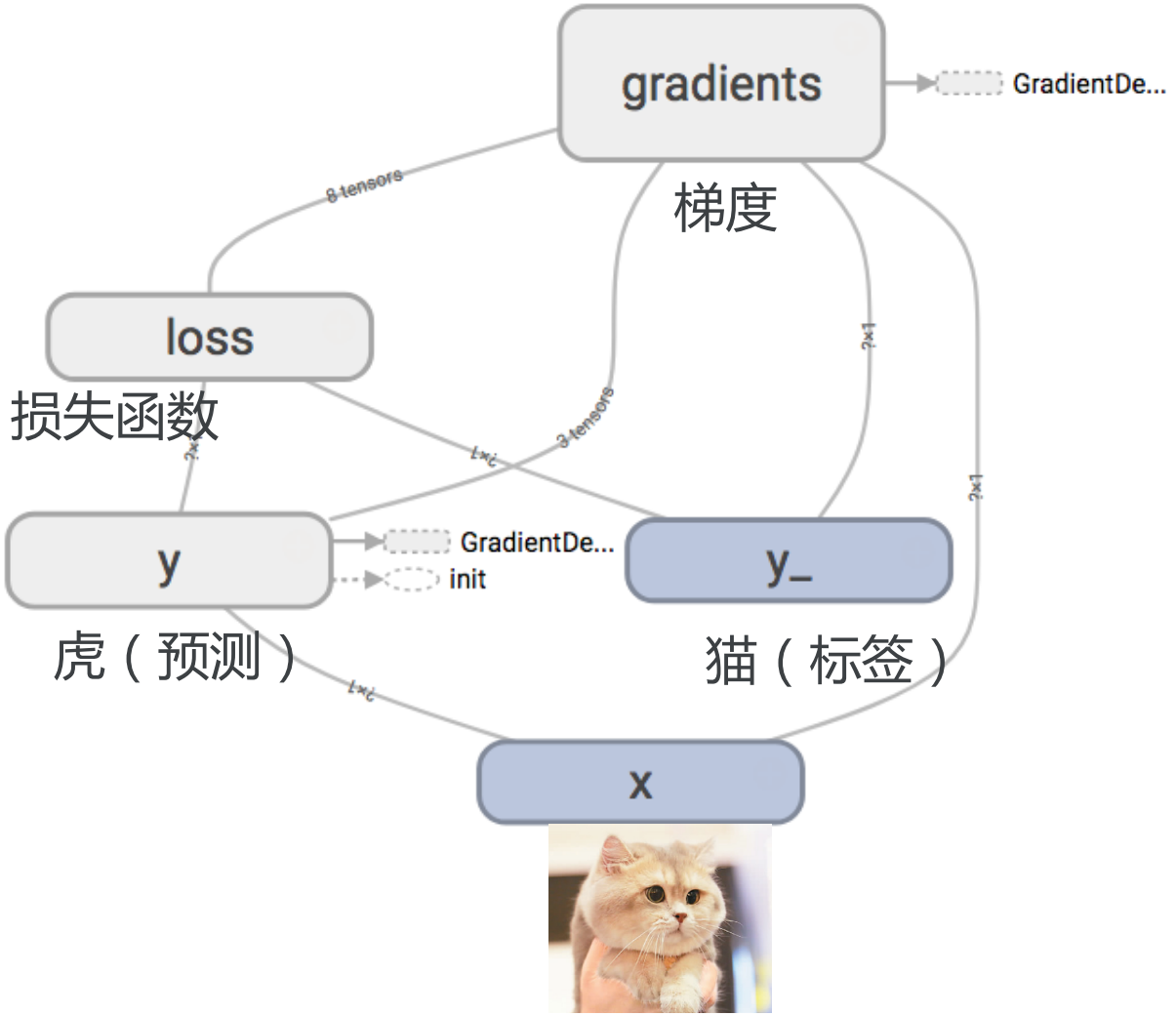


虎
（预测）

猫
（标签）

神经网络：前向传播


前向传播
input layer
hidden layer 1
hidden layer 2
output layer

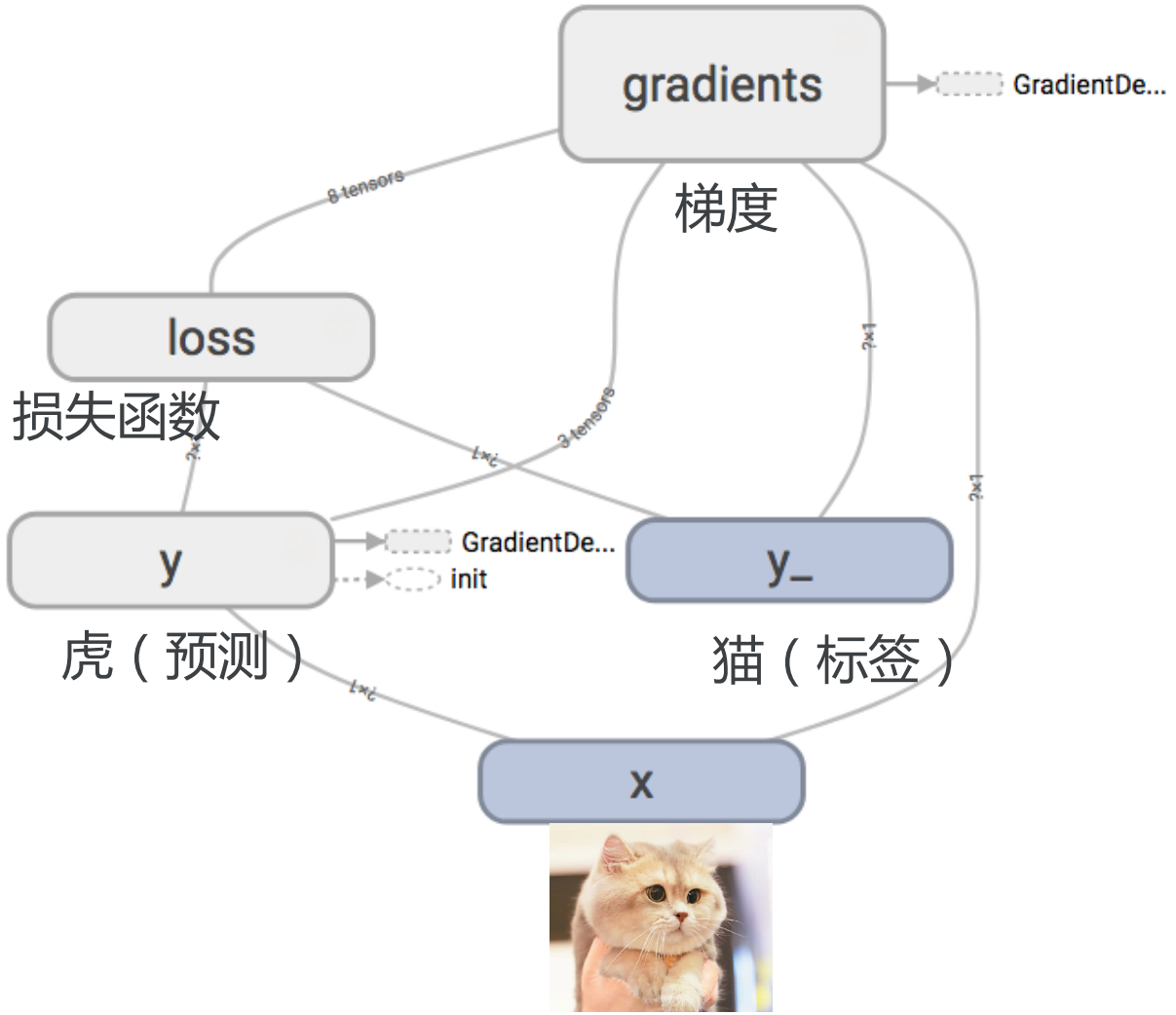

gradients
GradientDe...
8 tensors
梯度
loss
损失函数
3 tensors
y
GradientDe...
init
y_
虎（预测）
猫（标签）
x

神经网络：训练机制

理想函数： $y = b + wx$

假设函数：$h_\theta(x) = \theta_0 + \theta_1 x_1 = \theta^T x$

损失值（误差）： $loss = y - h_\theta(x)$



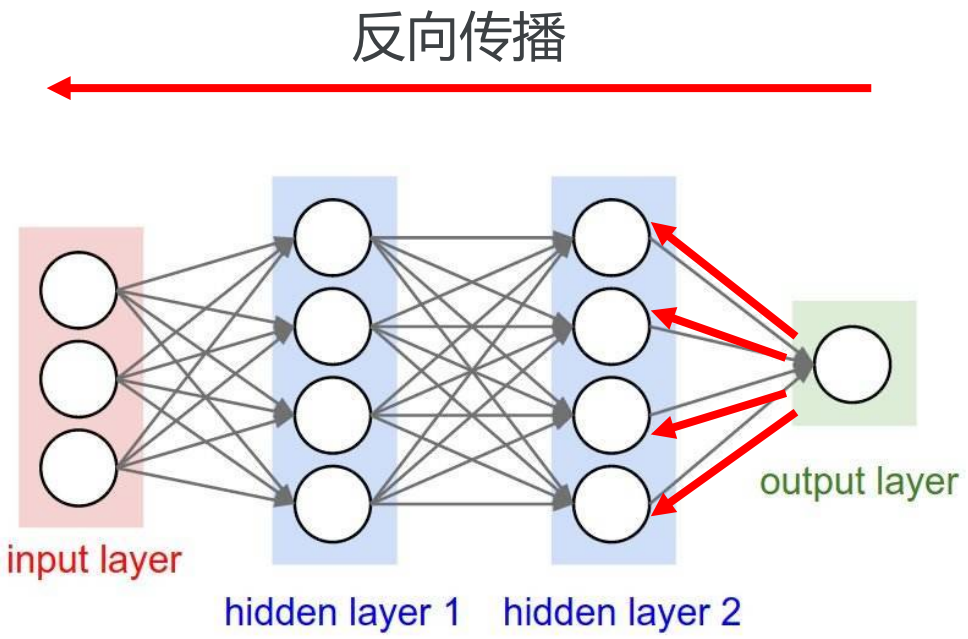
神经网络：反向传播

目标函数：

$$J(\theta) = \frac{1}{2n}\sum_{i}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

迭代模型参数：

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}\mathrm{J}(\theta)$$



优化算法：随机梯度下降（SGD）

迭代模型参数：

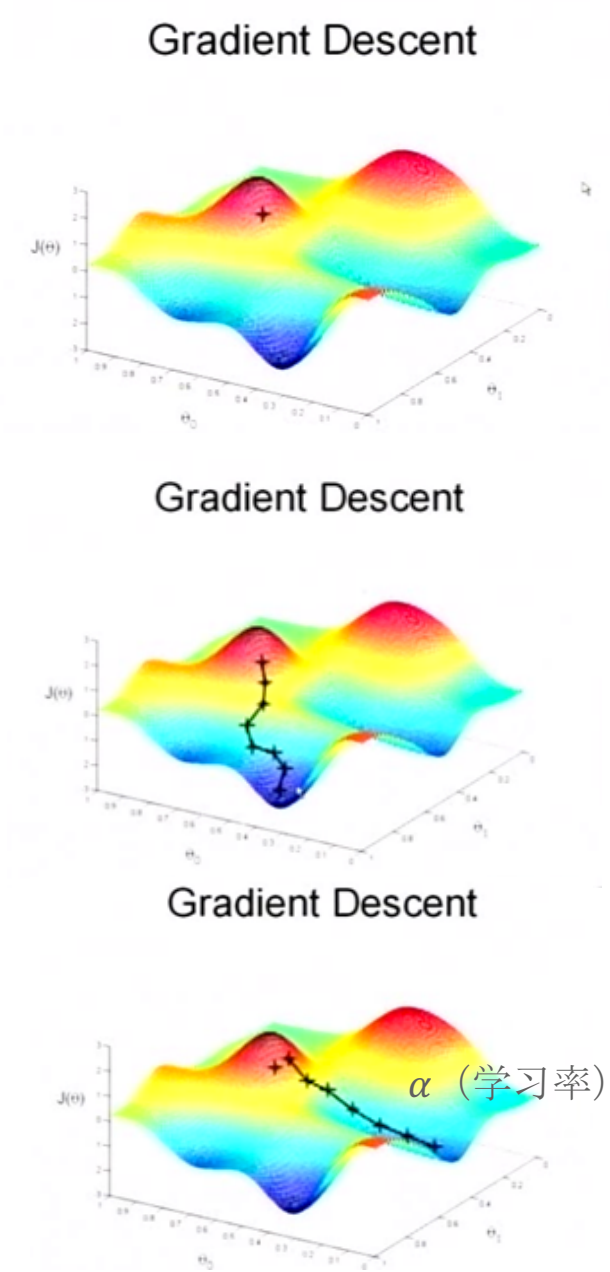
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathrm{J}(\theta)$$

代入 $\mathrm{J}(\theta)$ 求导，结果如下：

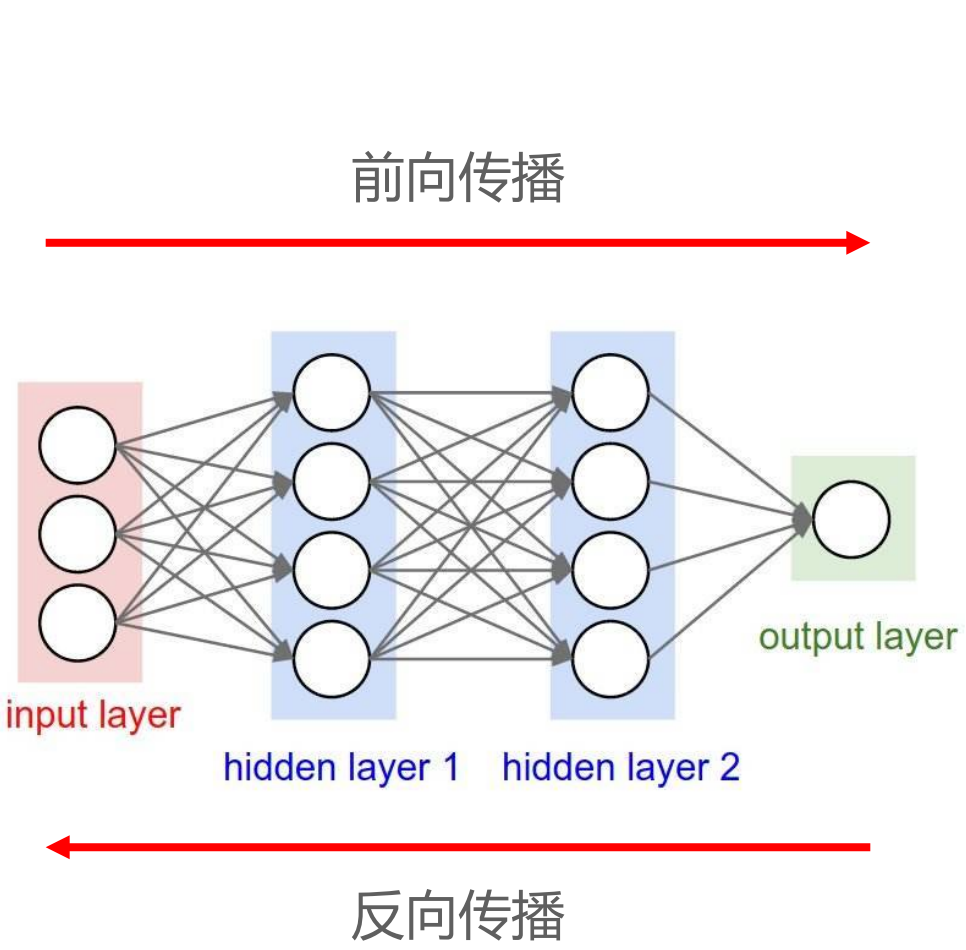
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_i^n \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

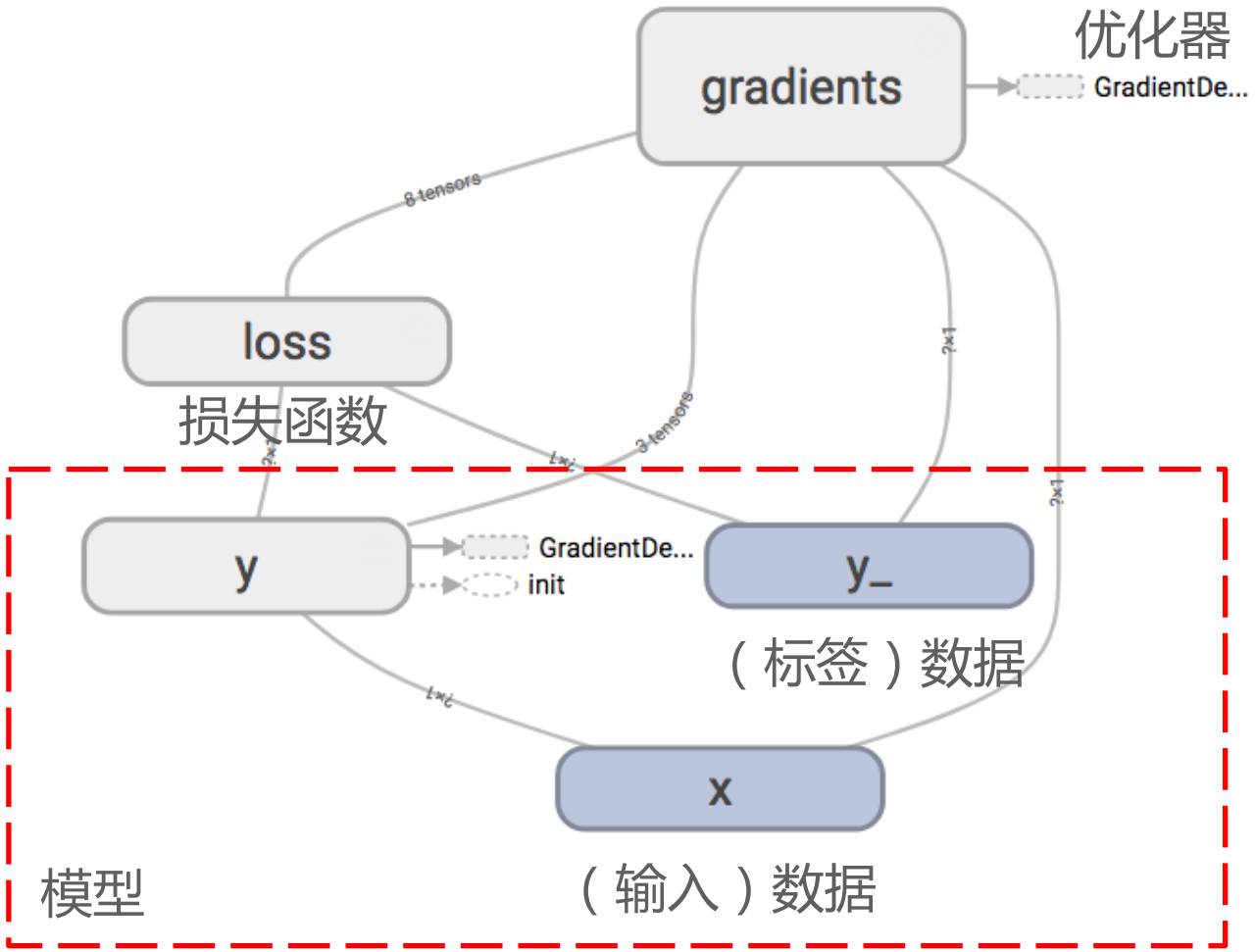
$$\theta_j := \theta_j - \alpha \frac{1}{2n} \sum_i^n 2\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)\left(x_j^{(i)}\right)$$

$$\theta_j := \theta_j - \alpha \frac{1}{n} \sum_i^n \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)\left(x_j^{(i)}\right)$$



神经网络四个核心概念间的关系






EDU
CSDN学院 IT实战派