中国科学院大学
University of Chinese Academy of Sciences

卷积神经网络

AI DISCOVERY

目录

AI DISCOVERY

AI DISCOVERY

目录

AI DISCOVERY

AI DISCOVERY

概述

AI DISCOVERY

深层神经网络问题导入

卷积神经网络概念引出

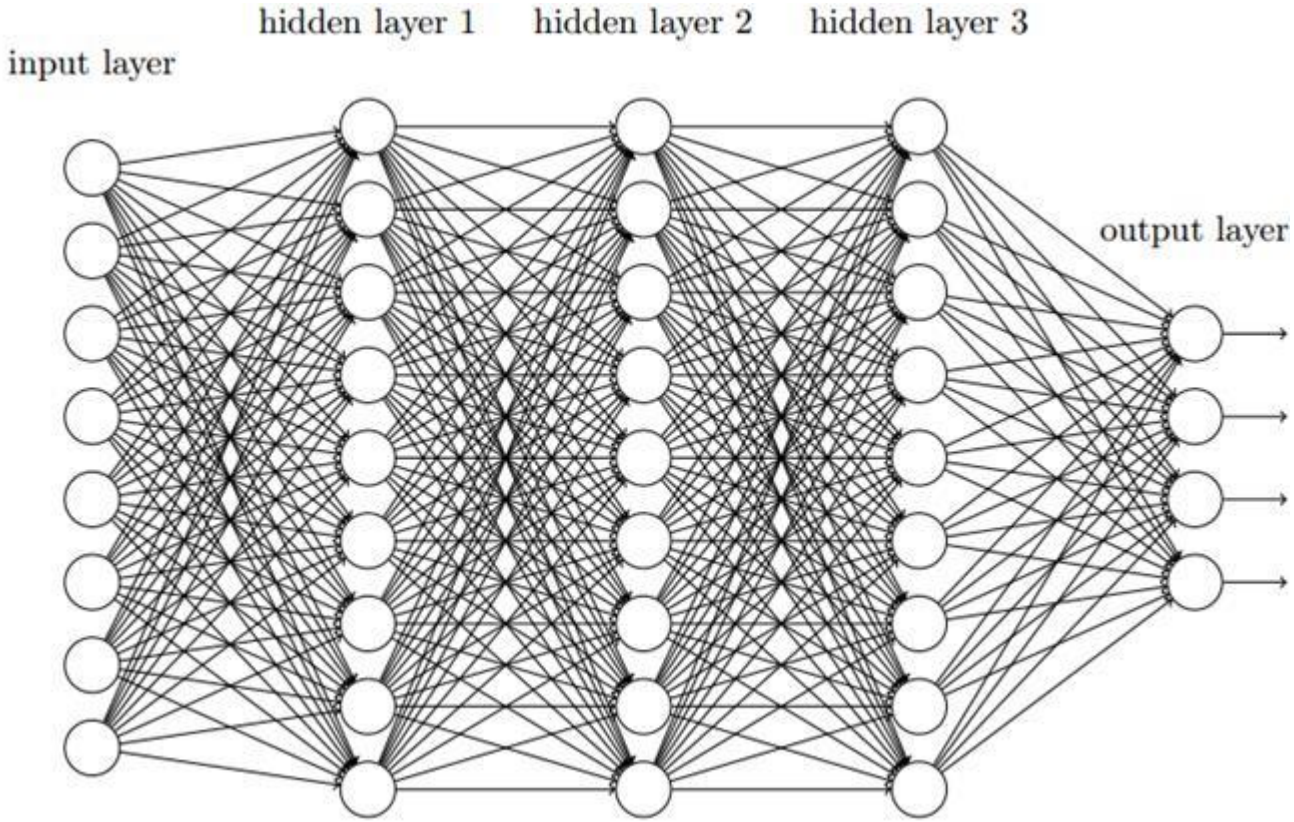
AI DISCOVERY

神经网络的问题

AI DISCOVERY

导入

在语音识别等一维信号处理领域取得成功的网络结构



- 考虑一种情形：

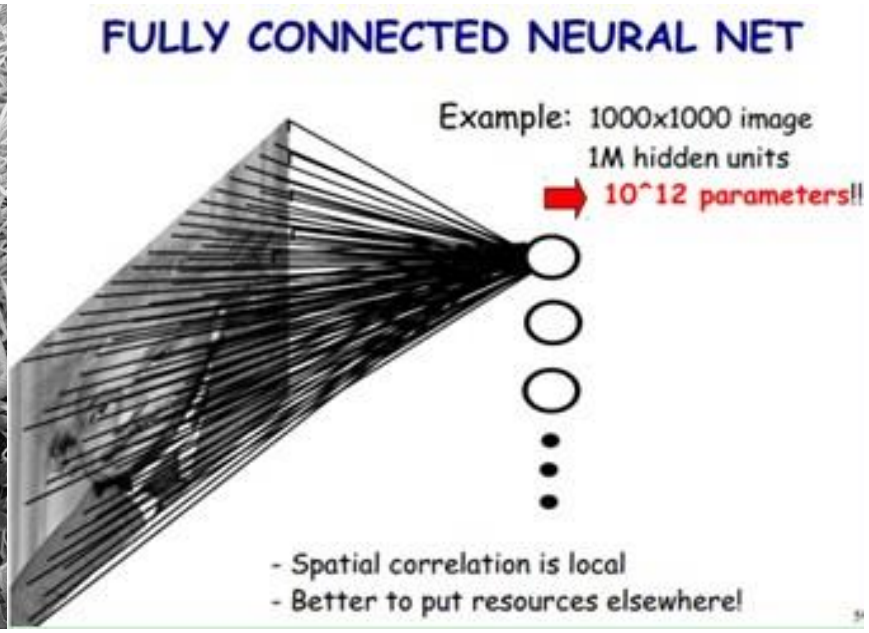
  ✓ 如果我们应用右图那样的网络来处理图像问题会发生什么？

AI DISCOVERY

神经网络的问题

导入

- 考虑一种情形：
  - ✓ 以1000*1000的灰度图像作为输入层
  - ✓ 希望隐层有和输入相同的神经单元
- 全连接：
  - ✓ 10^6 * 10^6 = 1M * 1M = 1T个参数，一个参数以浮点数4Byte的方式存储
  - ✓ 需要不小于4T内存



神经网络的问题

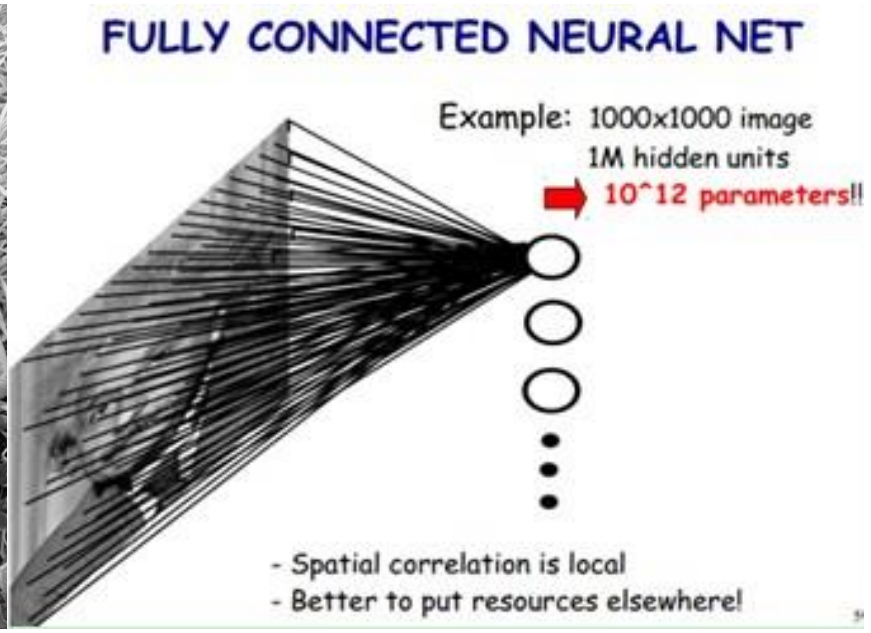
AI DISCOVERY

导入

- 考虑一种情形：
  - ✓ 以1000*1000的灰度图像作为输入层
  - ✓ 希望隐层有和输入相同的神经单元
- 全连接：
  - ✓ 10^6 * 10^6 = 1M * 1M = 1T个参数，一个参数以浮点数4Byte的方式存储
  - ✓ 需要不小于4T内存

：这才是第一个隐层，有什么机器能受得了这种计算？

：从入门到放弃系列。。



AI DISCOVERY

神经网络的问题

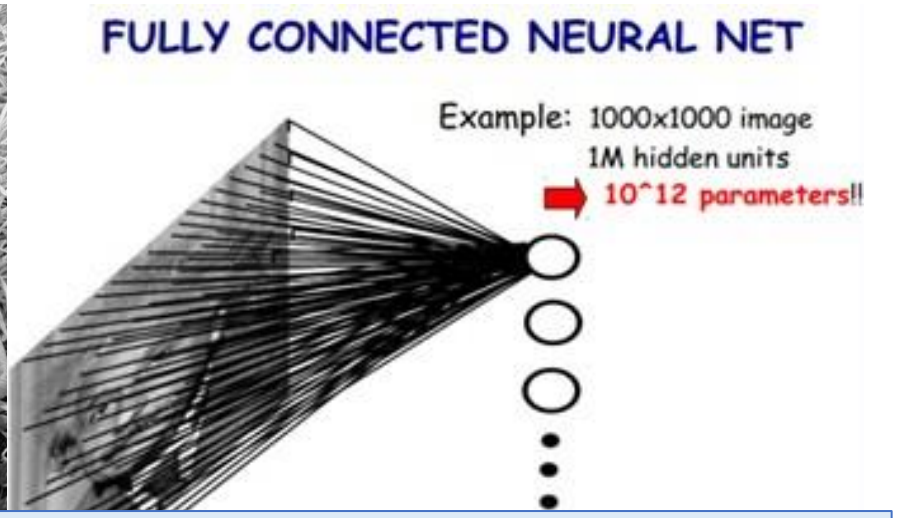
AI DISCOVERY

导入

- 考虑一种情形：
  - ✓ 以1000*1000的灰度图像作为输入层
  - ✓ 希望隐层有和输入相同的神经单元
- 全连接：
  - ✓ 10^6 * 10^6 = 1M * 1M = 1T个参数，一个参数以浮点数4Byte的方式存储
  - ✓ 需要不小于4T内存

：这才是第一个隐层，有什么机器能受得了这种计算？

：从入门到放弃系列。。



对于图像处理，我们是不是可以利用图像的某些模式或者说是特点，简化每一层的计算过程？

AI DISCOVERY

图像模式的特征

图像模式的特性一

- 第一个发现：

  鸢尾花仅出现在图像局部区域

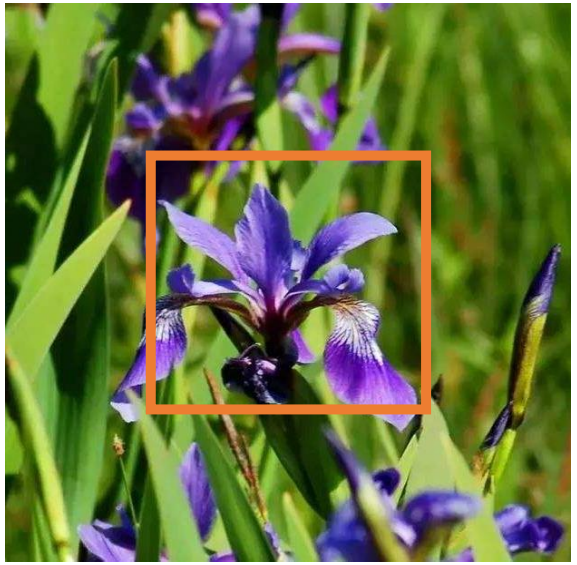
  并不是所有具有相似形态特征的

  鸢尾花都位于图像的同一个位置

- 如何应用这个发现?

- 可能的做法：

  1. 定义一种提取局部的特征的方法，可有效响应特定局部模式
  2. 用这种方法遍历整张图片



图像模式的特征

AI DISCOVERY

图像模式的特性一

- 第一个发现：

  鸢尾花仅出现在图像局部区域

  并不是所有具有相似形态特征的

  鸢尾花都位于图像的同一个位置

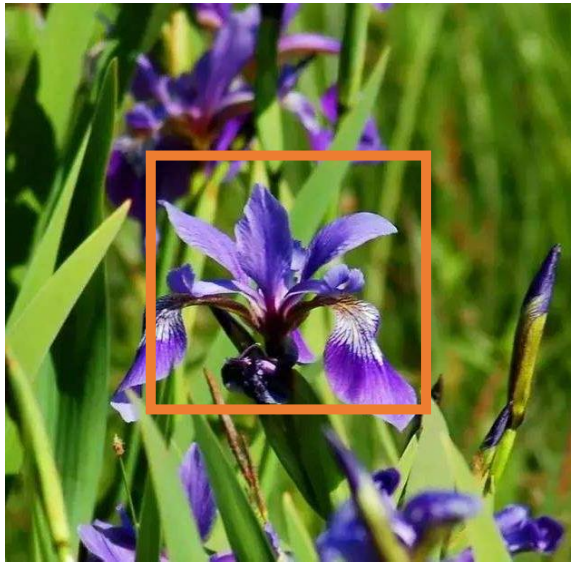
- 如何应用这个发现?

- 可能的做法：

  1. 定义一种提取局部的特征的方法，可有效响应特定局部模式
  2. 用这种方法遍历整张图片

：应用一次该方法只能提取一个特征

：所以对应同一张图片输入，应该应用多次该方法



AI DISCOVERY

图像模式的特征

AI DISCOVERY

图像模式的特性二

- 第二个发现：

  大小改变，鸢尾花仍然可以有效区分

- 如何利用这个特性？

- 可能的做法：

  1. 在神经网络逐层累加的过程中，可以直接对图像进行缩放
  2. 缩放到适当大小后，可以在特征提取过程中得到有效响应

AI DISCOVERY

概述

AI DISCOVERY

深层神经网络问题导入

卷积神经网络概念引出

AI DISCOVERY

卷积神经网络的诞生

AI DISCOVERY

图像模式的特性——小结

- 第一个发现对应的可能的做法：

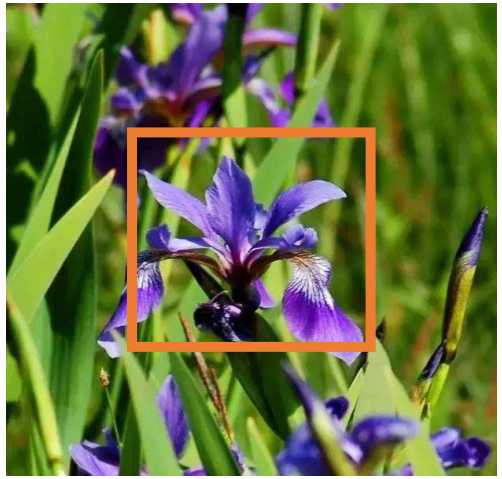
1. 定义一种提取局部的特征的方法，可有效响应特定局部模式
2. 用这种方法遍历整张图片

卷积：平移不变模式

- 第二个发现对应的可能的做法：

在神经网络逐层累加的过程中，可以直接对图像进行缩放

池化：下采样被检测物体不变模式



卷积神经网络的诞生

AI DISCOVERY

图像模式的特性——小结

- 第一个发现对应的可能的做法：

1. 定义一种提取局部的特征的方法，可有效响应特定局部模式
2. 用这种方法遍历整张图片

卷积：平移不变模式

- 第二个发现对应的可能的做法：

在神经网络逐层累加的过程中，可以直接对图像进行缩放

池化：下采样被检测物体不变模式

那是不是卷积和池化操作就够了？

我们是不是还需要更深的模型？

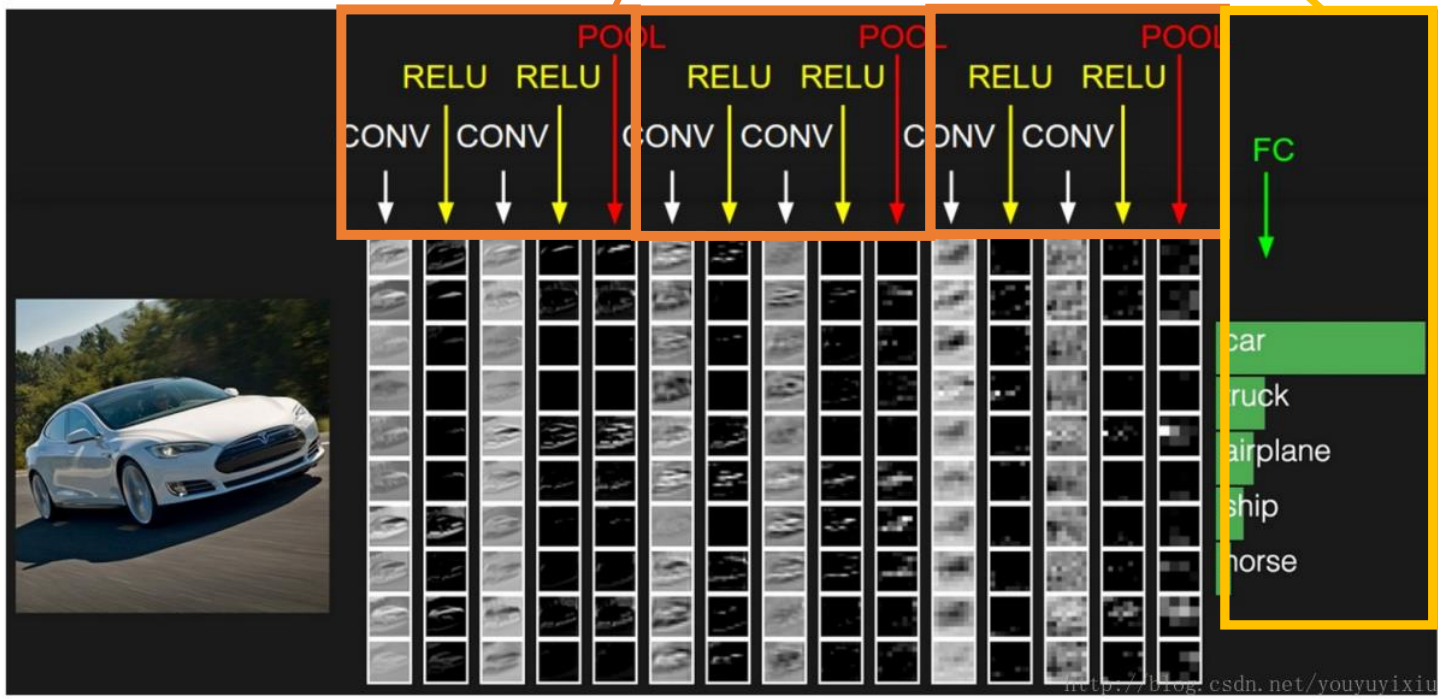
CNN基础结构

AI DISCOVERY

CNN应用图像模式的一般框架（以分类为例）

卷积层+激活函数+池化层+全连接层　　出现多次，用于提取特征

在最后出现一次或多次，用于做分类



目录

AI DISCOVERY

AI DISCOVERY

自己动手搭CNN

AI DISCOVERY

CNN基本结构

输入层

卷积层

激活层

池化层

全连接层

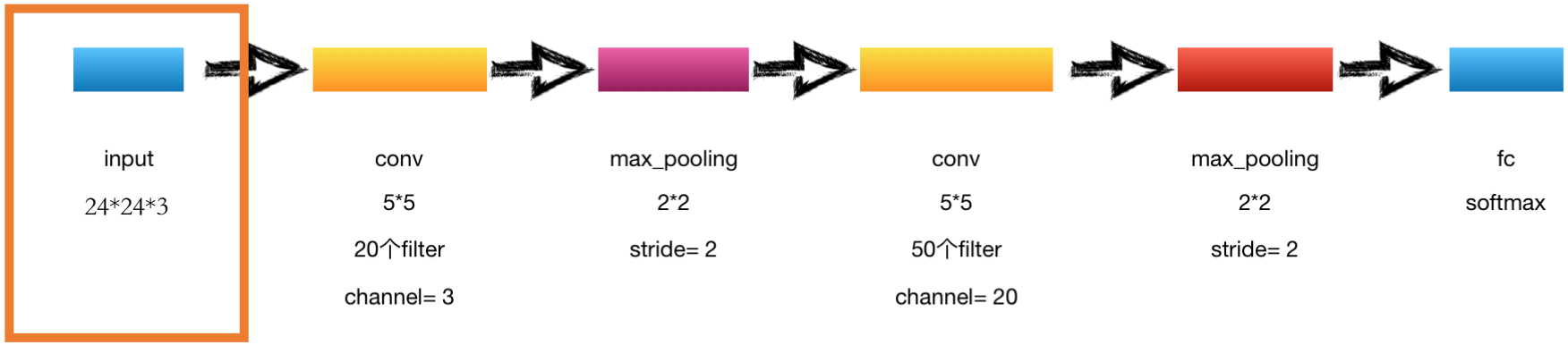
CNN网络训练

第一个CNN

AI DISCOVERY

输入层

输入层



simple_cnn网络结构

自己动手搭CNN

AI DISCOVERY

CNN基本结构

输入层

卷积层

激活层

池化层

全连接层

卷积、stride、padding
如何理解卷积?

CNN网络训练

第一个CNN

AI DISCOVERY

卷积层

AI DISCOVERY

卷积的直觉

计算机如何能知道图上有什么物体?

检测图像的边缘

卷积计算 = 特征抽取

灰度图像

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

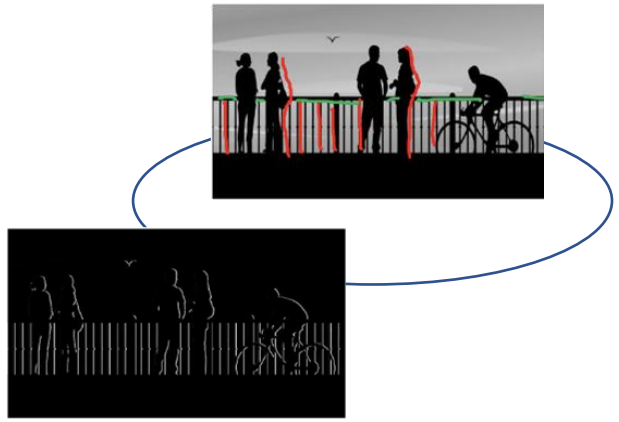
垂直边缘滤波器

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

结果图像

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

卷积：抽取局部模式



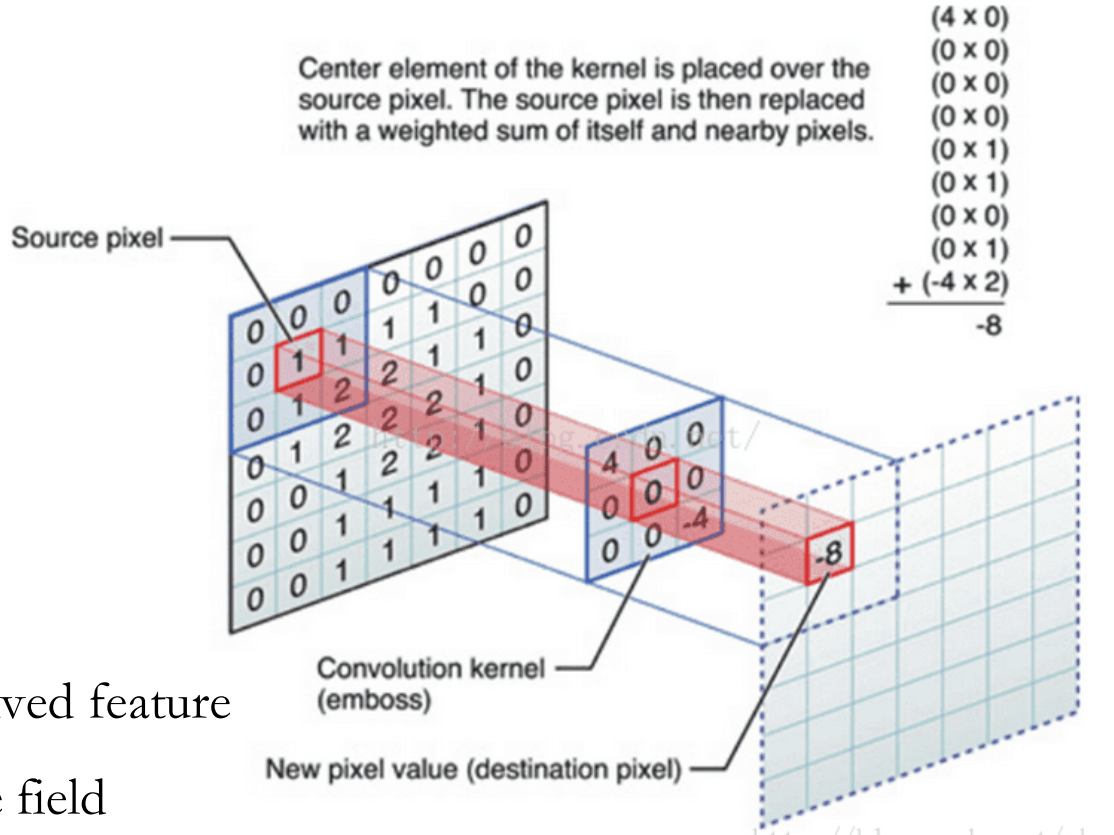
vertical edges

卷积层

AI DISCOVERY

灰度图像上使用单卷积核：单个特征的抽取

- **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
- **术语**
  - ✓ x：输入，input，image
  - ✓ w：卷积核，filter，kernel
  - ✓ s：特征映射，feature map，activation map，convolved feature
  - ✓ 感受野：一个神经元连接到的输入区域，receptive field



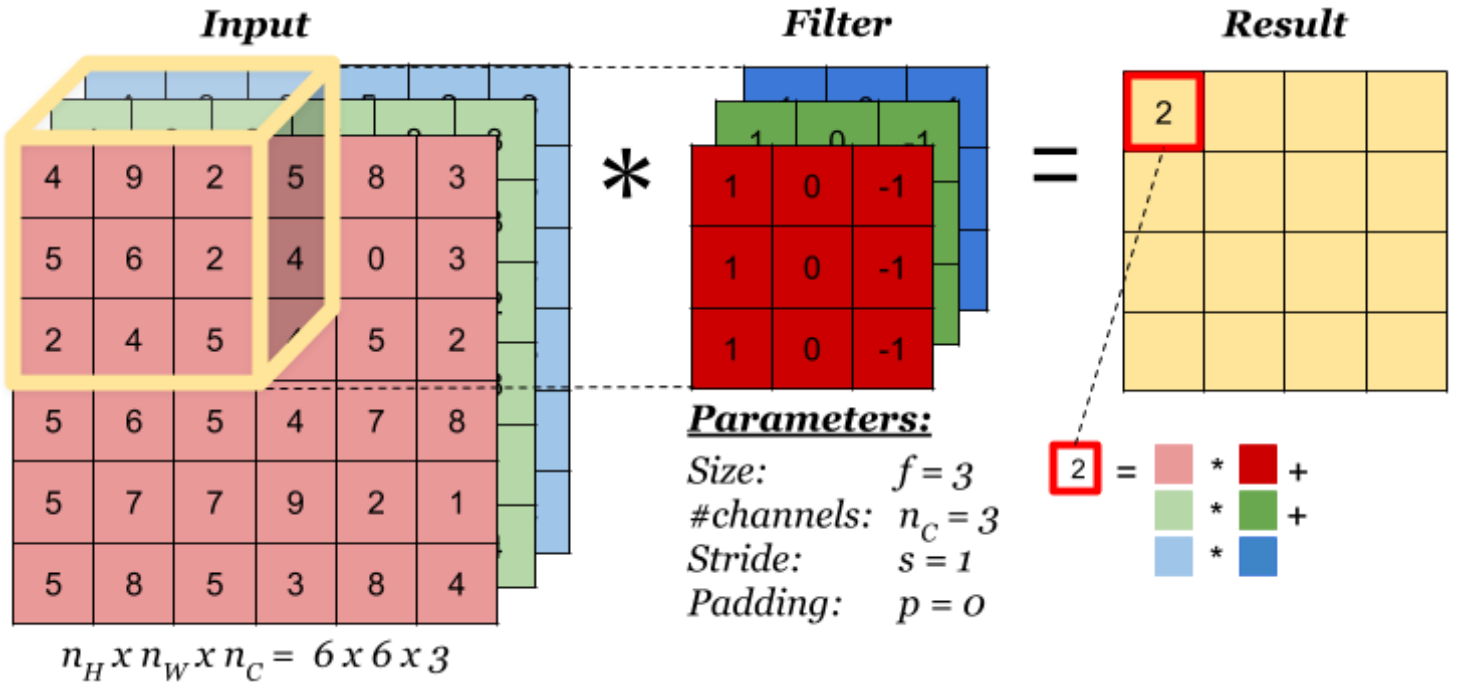
AI DISCOVERY

卷积层

AI DISCOVERY

RGB图像上使用单卷积核：单个特征的抽取

- **Notes 1**
  - ✓ RGB图像channel = 3
  - ✓ 对RGB图像进行操作的卷积核的深度为3
  - ✓ 右图以3 * 3 的卷积核为例
- **Notes 1 update**
  - ✓ 卷积核有第三个维度：Depth
  - ✓ 卷积核的深度 = 上一层数据输入的深度（channel数）



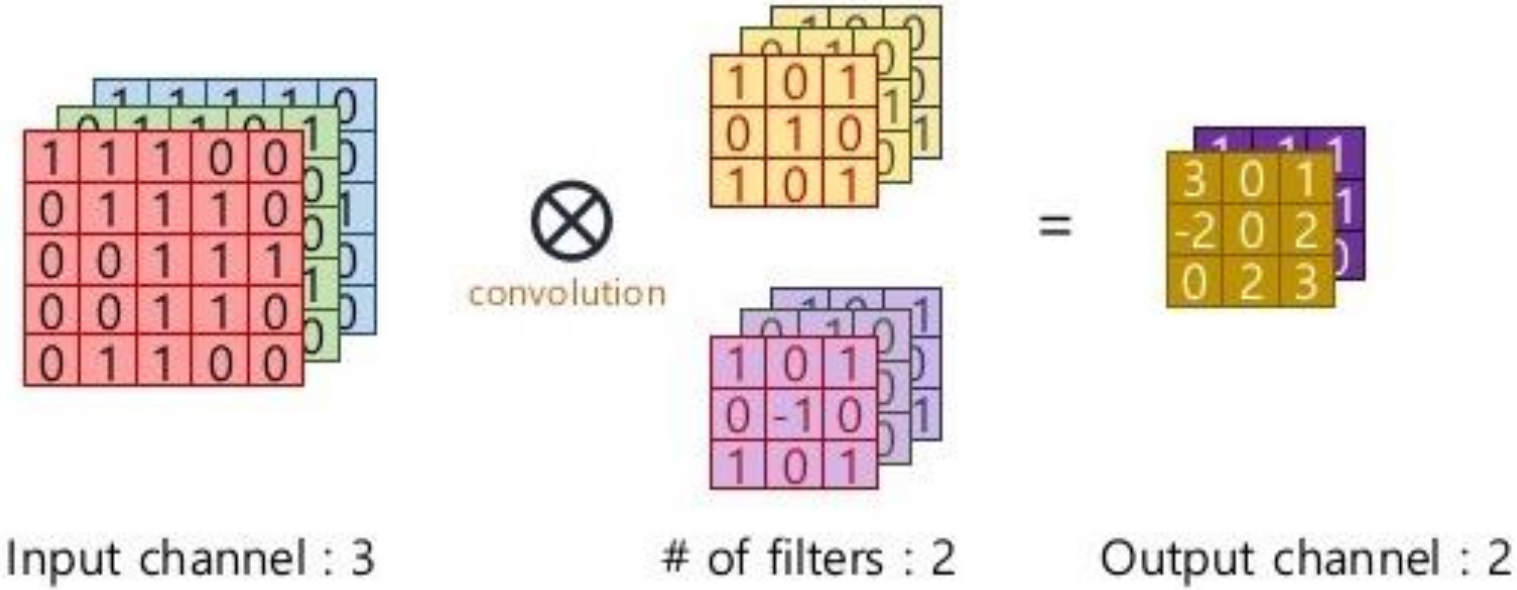
AI DISCOVERY

卷积层

AI DISCOVERY

RGB图像上使用多卷积核：多个不同特征的提取

- **多个卷积核**
  - ✓ 一个卷积核提取一种局部模式
  - ✓ 在对每一层的卷积操作中，都要同时使用多个卷积核，提取多种不同的局部模式

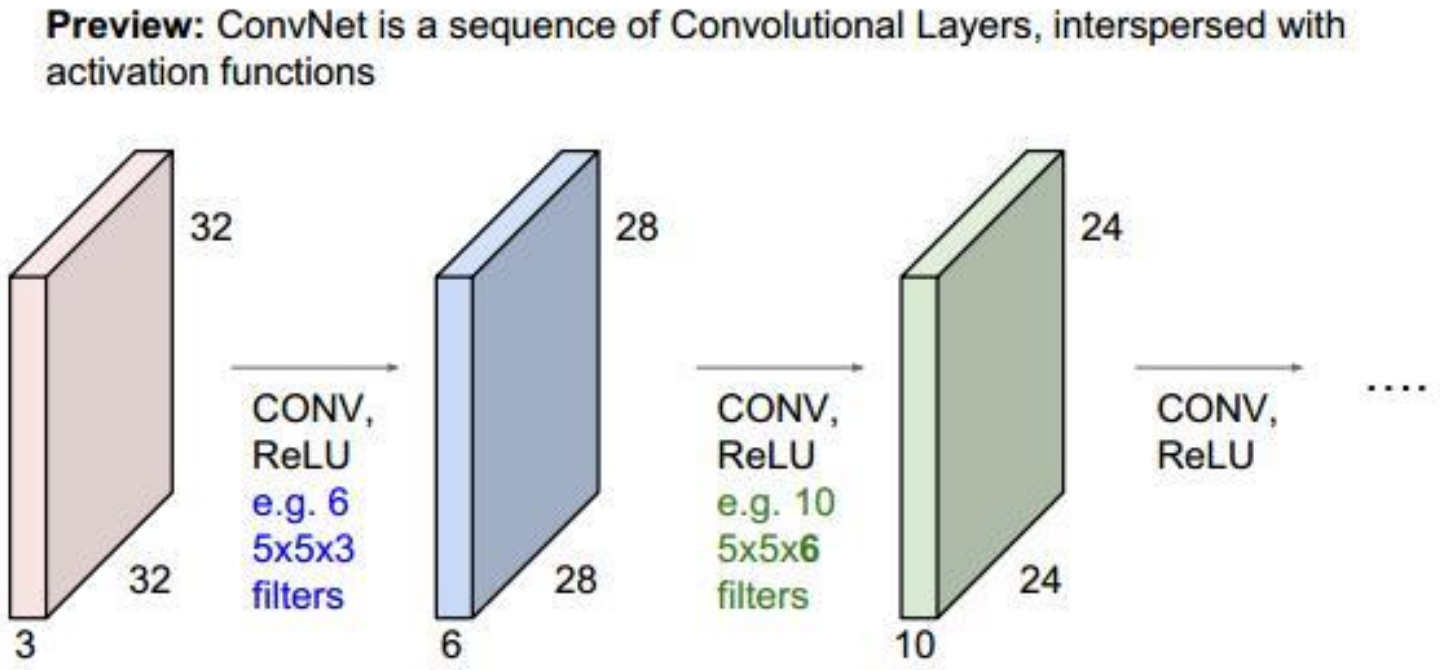


卷积层

AI DISCOVERY

卷积隐层的堆叠

- **Notes 2**
  - ✓ 卷积核的个数 = 下一层数据的深度 = 下一卷积层 卷积核的深度
  - ✓ 卷积核的个数 = 提取特征的数量，超参数，可以自己调节



卷积层

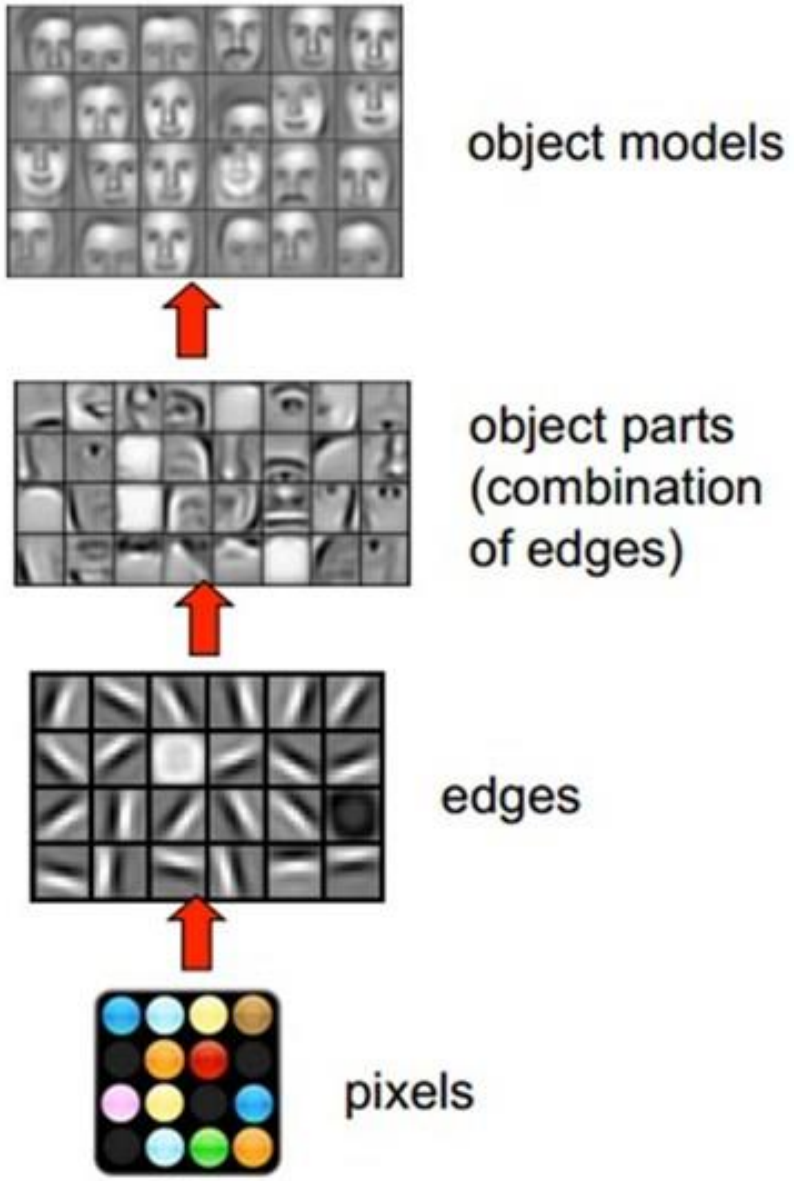
AI DISCOVERY

隐层的卷积：特征组合

✓ **多层卷积：**

一层卷积得到的特征往往是局部的

层数越高，学到的特征就越全局化



卷积层

AI DISCOVERY

需要注意的参数：stride

- **Stride**
  - ✓ 一次滑动的步长
  - ✓ 有height上的stride和width上的stride
  - ✓ 图片中的stride = 2,指在两个维度上的stride都为2

10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0

Stride =1： 一次滑动1格

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

0	30	30
0	30	30
0	30	30

10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0

Stride =2： 一次滑动2格

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

0	30
0	30

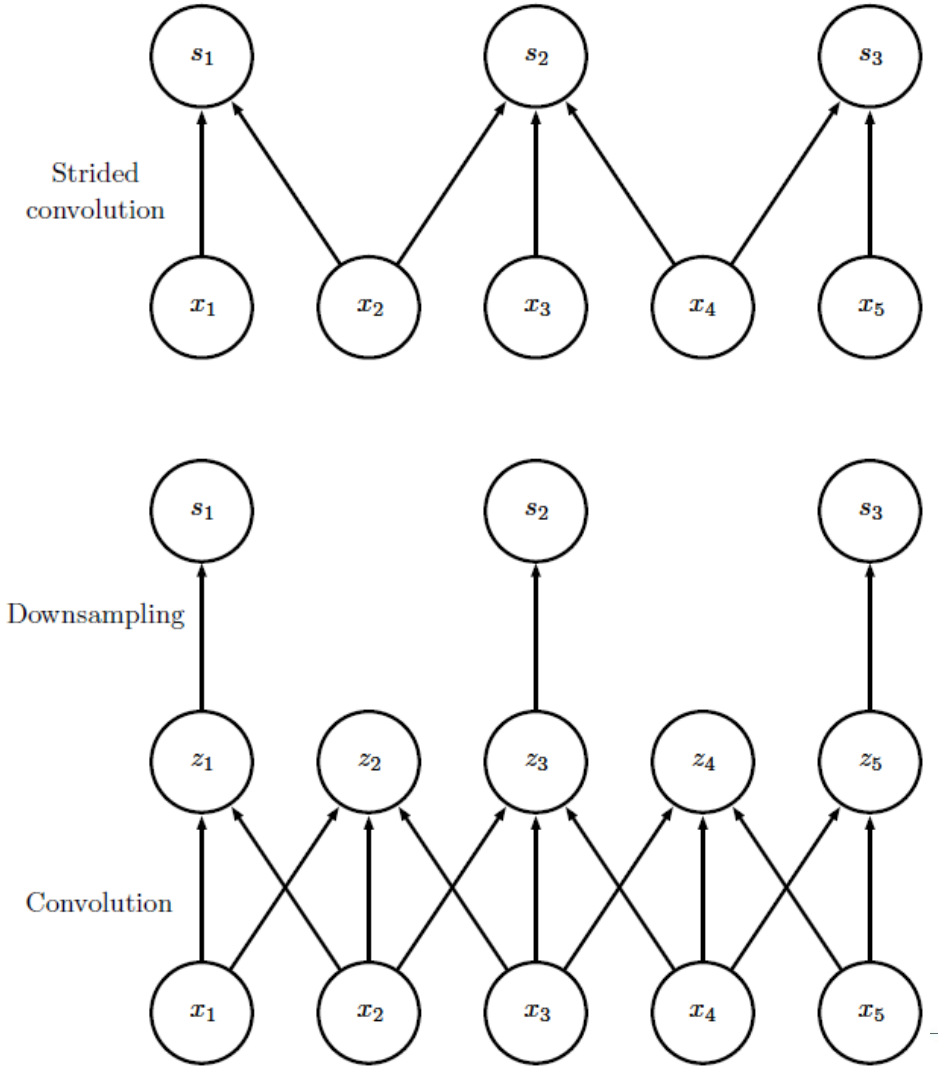
AI DISCOVERY

卷积层

AI DISCOVERY

需要注意的参数：stride

- **Stride**
  - ✓ stride设置为超过1的参数，就相当于在stride=1的卷积结果中作了下采样
  - ✓ 实际上是跳过去不计算，能够成倍减少计算量



卷积层

AI DISCOVERY

需要注意的参数：padding

- **Padding = valid**
  - ✓ 不进行补零操作，s=1时，每卷积一次，宽和高方向的数据维度下降F-1，其中F为卷积核大小
- **Padding = same**
  - ✓ 在输入的周围进行0或复制填充
  - ✓ 卷积前width=卷积后width，卷积前height=卷积后height
  - ✓ F =3，stride =1，pad = 1

10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0
10	10	10	0	0

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

padding：valid

0	30	30
0	30	30
0	30	30

0	0	0	0	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	10	10	10	0	0	0
0	0	0	0	0	0	0

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

padding：same

-20	0	20	20	0
-30	0	30	30	0
-30	0	30	30	0
-30	0	30	30	0
-20	0	20	20	0

卷积层

AI DISCOVERY

小结

✓ 输入：$W_1 \times H_1 \times D_1$

✓ 超参数：

the number of filters: K

the dimension of filters: F

stride步长 S

padding P

✓ 输出： $W_2 \times H_2 \times D_2$

$$W_2 = \frac{W_1+2P-F}{S}+1, \quad H_2 = \frac{H_1+2P-F}{S}+1, \quad D_2=K$$

✓ 参数：

$$(F \times F \times D_1 + 1) \times K$$

AI DISCOVERY

自己动手搭CNN

AI DISCOVERY

CNN基本结构

输入层

卷积层

激活层

池化层

全连接分层

激活函数

CNN网络训练

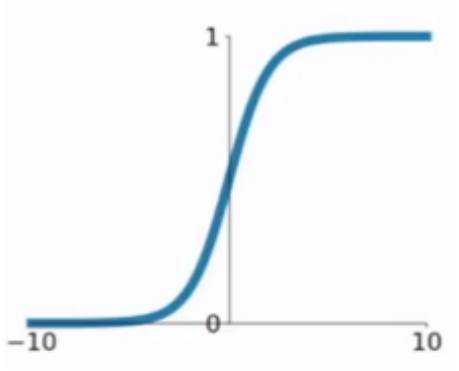
第一个CNN

AI DISCOVERY

激活函数简述

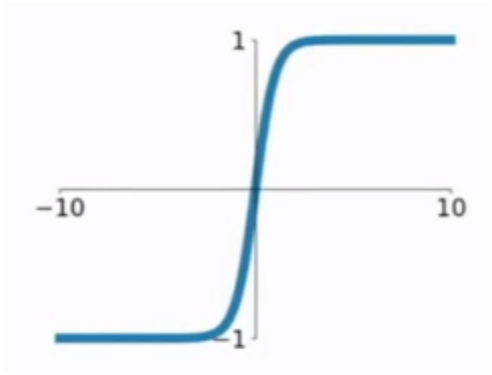
往模型中加入非线性元素，能表示更大范围的函数

一般不在同一个网络中使用多种激活函数



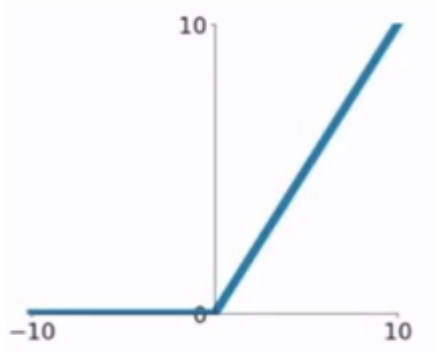
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

指数运算，效率低



$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

指数运算，效率低



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

线性运算，效率极高，最常用

自己动手搭CNN

AI DISCOVERY

CNN基本结构

输入层

卷积层

激活层

池化层

全连接层

一般意义上的池化
典型的池化：平均池化、最大池化

CNN网络训练

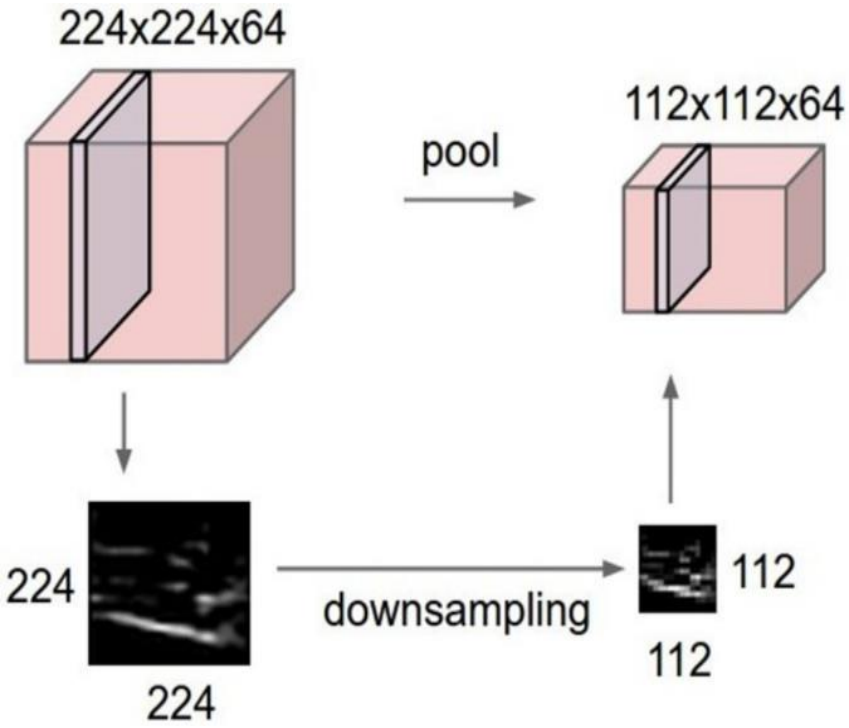
第一个CNN

AI DISCOVERY

池化层

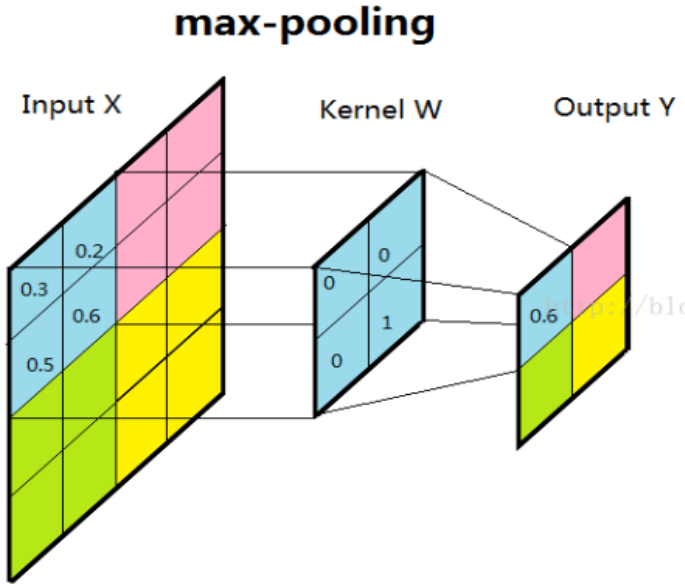
➢ 池化

- ✓ 在width和height维度上进行下采样，不改变depth的维度
- ✓ 右图相当于对输入数据使用了2*2， stride = 2的卷积核，但是该卷积核不是通过学习获得，而是人为定义的卷积核（不算做参数）
- ✓ 能够成倍减少计算量
- ✓ 相比stride，池化层可以选择进行下采样的方式



池化层

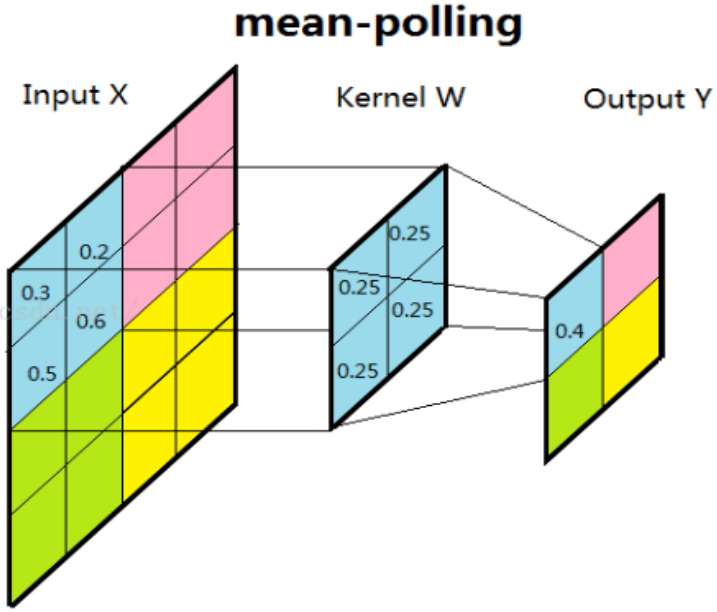
AI DISCOVERY



Max-Pooling:

对邻域内特征点取最大作为最后的特征值

$$\max(0.3,0.2,0.5,0.6) = 0.6$$



Mean-Pooling:

对邻域内特征点取平均作为最后的特征值

$$\frac{0.3 + 0.2 + 0.5 + 0.6}{4} = 0.4$$

池化层

AI DISCOVERY

小结

- ✓ 输入：$W_1 \times H_1 \times D_1$
- ✓ 超参数：

  the dimension of filters: F

  stride步长 S

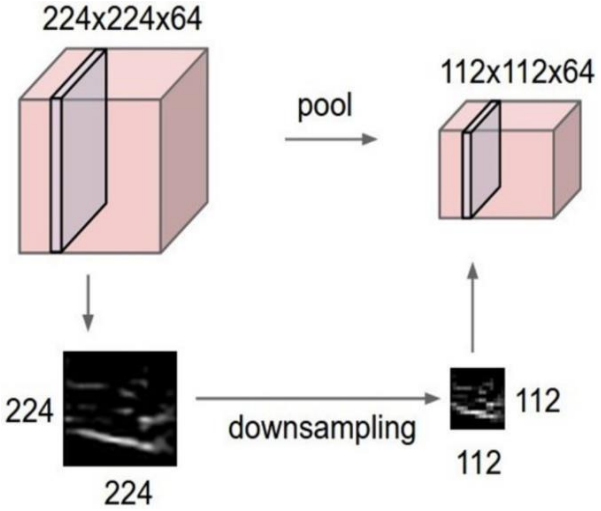
- ✓ 输出： $W_2 \times H_2 \times D_2$

$$W_2 = \frac{W_1 - F}{S} + 1, \quad H_2 = \frac{H_1 - F}{S} + 1, \quad D_2 = D_1$$

- ✓ 参数：

  一些池化方式中是有参数的

  max-pooling和mean-pooling没有参数



AI DISCOVERY

自己动手搭CNN

AI DISCOVERY

CNN基本结构

输入层

卷积层

激活层

池化层

全连接层

CNN中全连接的使用

CNN网络训练

第一个CNN

AI DISCOVERY

全连接层分类

➢ Notes

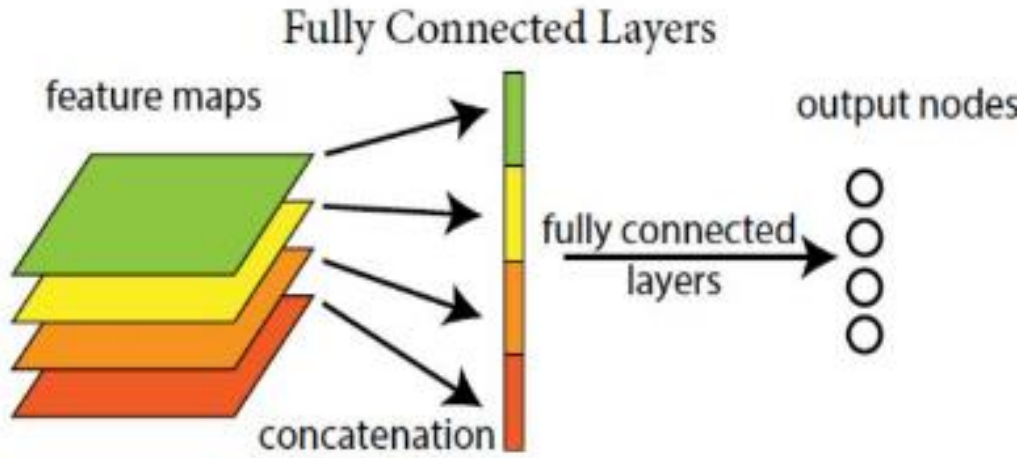
- ✓ 将多层的特征映射抻直成一个一维的向量
- ✓ 采用全连接的方式将向量连接向输出层，

  打破卷积特征的空间限制

  对卷积层获得的不同的特征进行加权

  最终目的是得到一个可以对不同类别进行区分的得分

- ✓ 输出层就是对应每个类别的得分

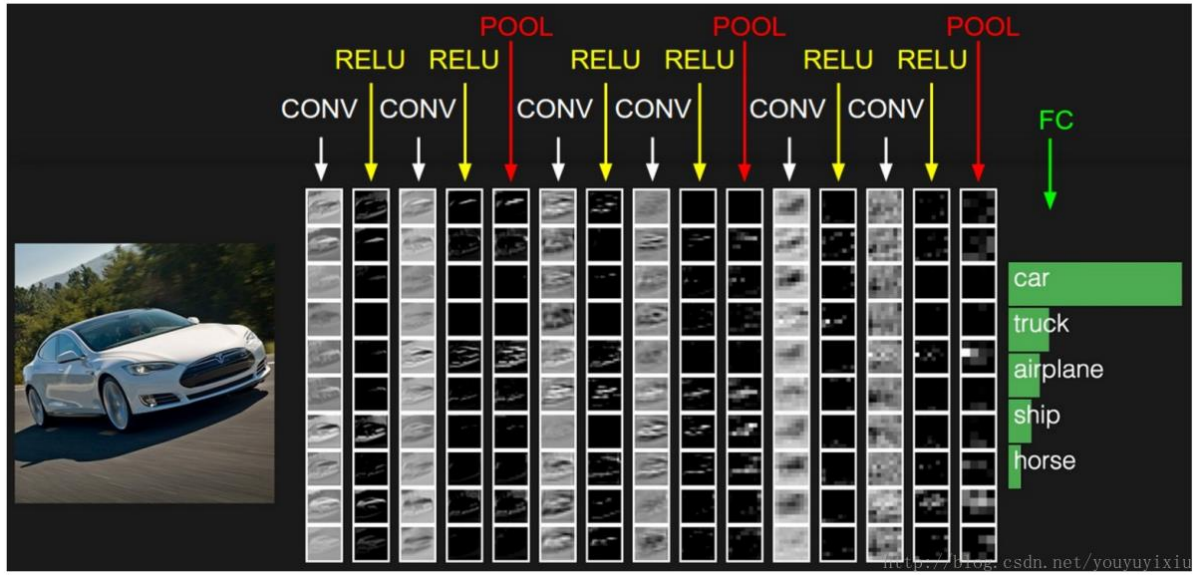


网络搭建小结

AI DISCOVERY

卷积神经网络的一般结构

1. 卷积层+ReLU和 池化层 的组合多次出现　　提取特征
2. 多个全连接 或 特殊的CNN结构 作为输出层　　作分类器/检测器/分割器



AI DISCOVERY

自己动手搭CNN

AI DISCOVERY

CNN基本结构

CNN网络训练

损失与误差的反向传播

模型评估与正则化

第一个CNN

AI DISCOVERY

损失与误差反向传播

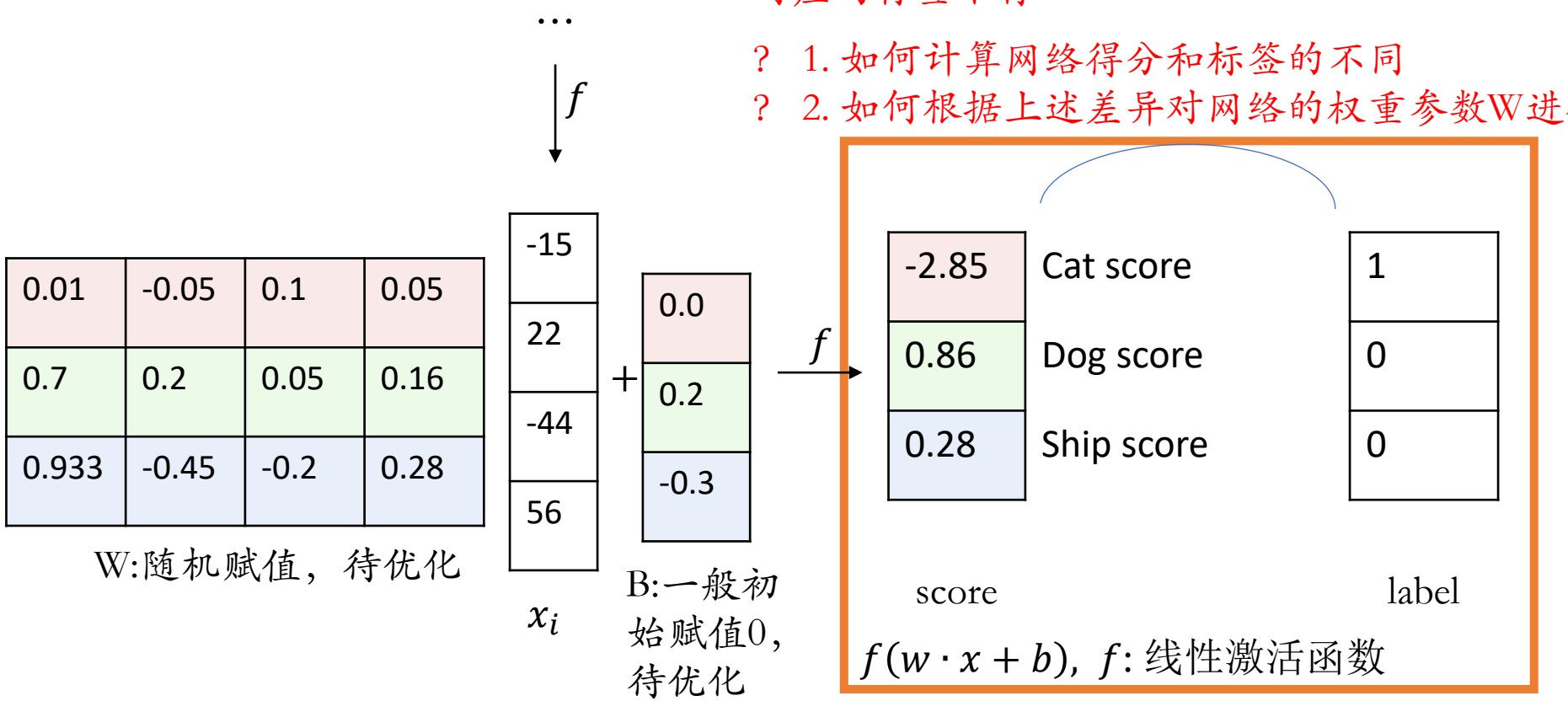
AI DISCOVERY

多分类（打标）损失导入

：在随机赋值w,b的情形下，模型的初始计算结果必然和对应的标签不符

？ 1. 如何计算网络得分和标签的不同

？ 2. 如何根据上述差异对网络的权重参数W进行更新



CNN的最后一层全连接层：作为分类结果

AI DISCOVERY

损失函数

AI DISCOVERY

- **交叉熵损失 & SoftMax 概率归一化**

$$\mathrm{H}(p,q) = -\sum p(x) log q(x)$$

用来衡量两个概率分布间的差异性

标签是个one-hot向量，是个概率分布

网络直接输出的结果不是概率分布

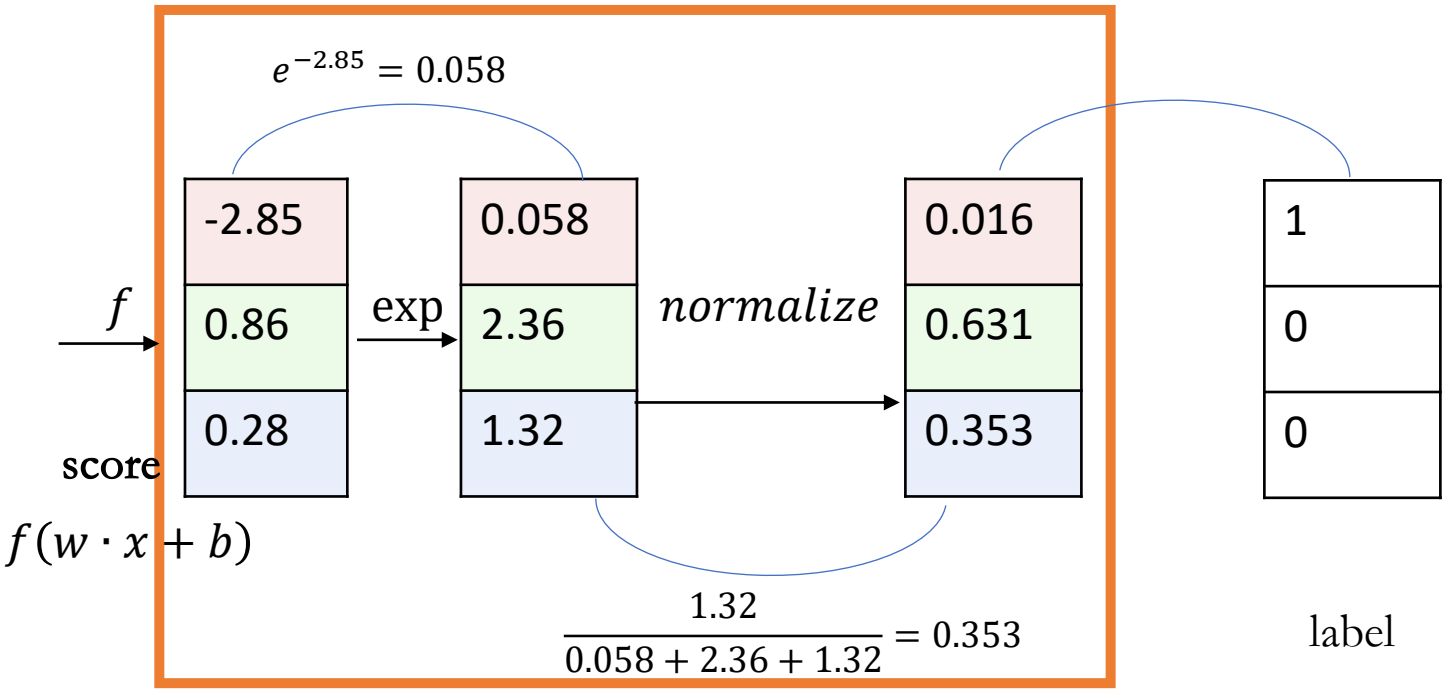
1. 所以在网络中先将得分结果归一化为概率分布→ SoftMax

$$\text{SoftMax的计算：}\ \mathrm{S}_i = \frac{e^i}{\sum_j e^j}$$

？ 1. 如何计算网络得分和标签的不同

Step1: 将得分转化为概率分布

概率分布：$1 > y_i > 0, \sum_i y_i = 1$



AI DISCOVERY

损失函数

AI DISCOVERY

？ 1. 如何计算网络得分和标签的不同

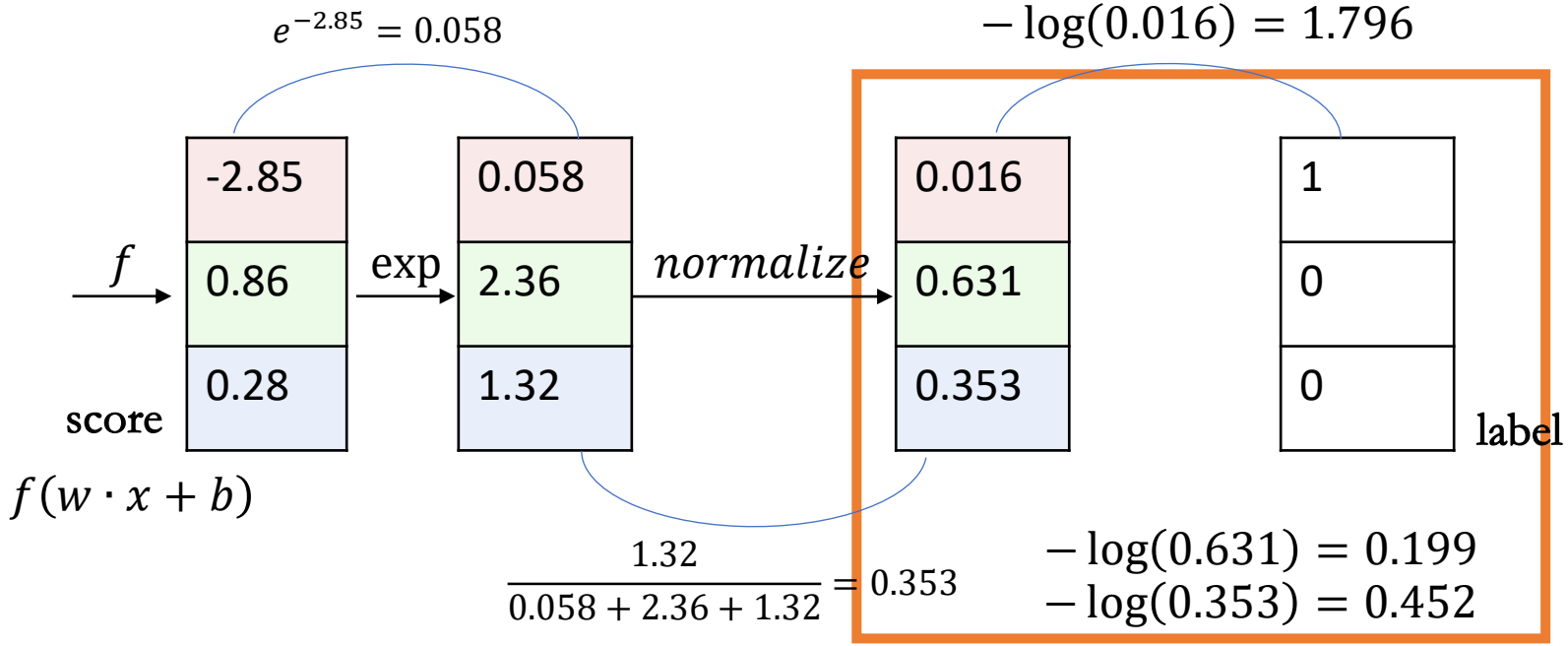
Step1: 使用概率分布间的差异度量

- **交叉熵损失 & SoftMax 概率归一化**

$$H(p,q) = -\sum p(x)logq(x)$$

用来衡量两个分布间的差异性

2. 根据输出和真实的标签间的概率分布计算交叉熵，来度量输出结果和标签的差异情况



- ✓ p(x): 标签代表的真实概率分布
- ✓ q(x): 输出代表的模型概率分布

AI DISCOVERY

梯度下降

AI DISCOVERY

? 2. 如何根据上述差异对网络的权重参数W进行更新

Step2: 权重根据差异方向更新

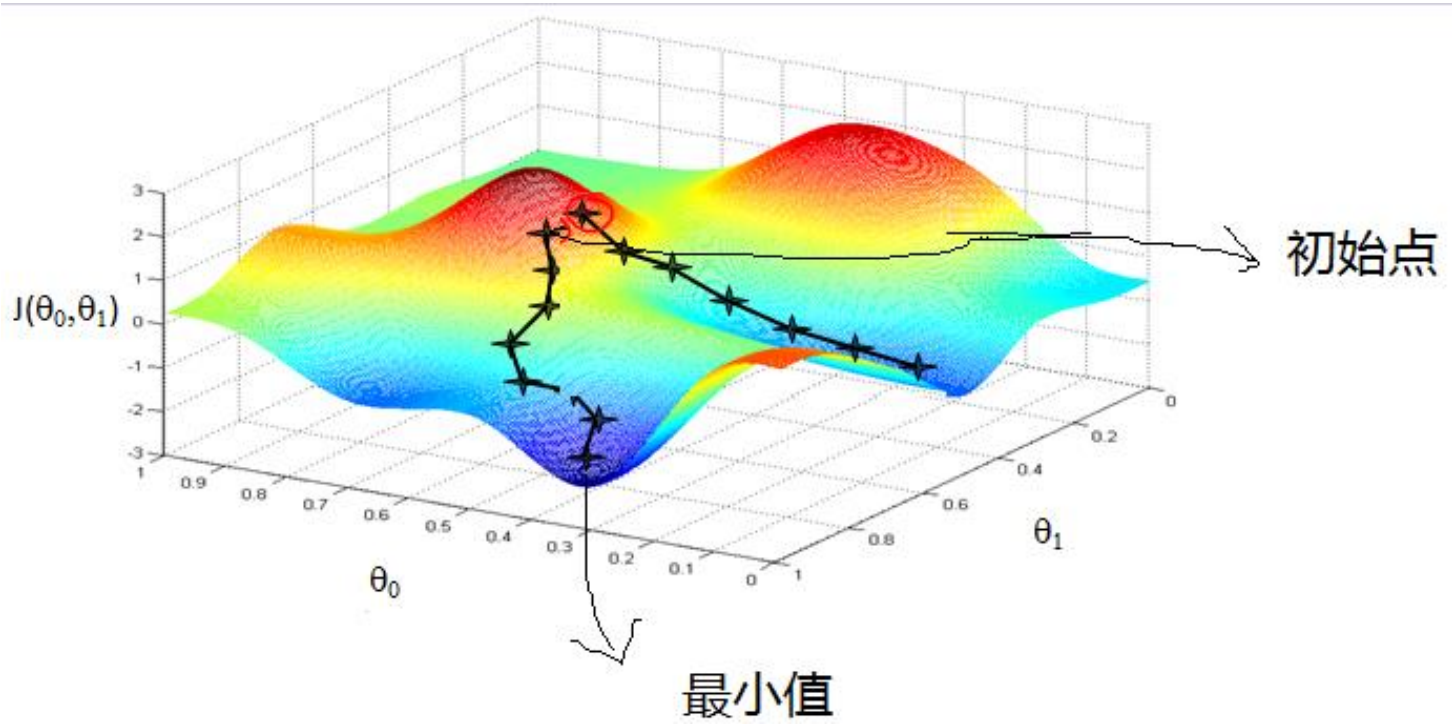
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

梯度：参数更新的方向
损失下降最快的方向

步长：参数每次更新多大

➢ 梯度下降的直觉

- ✓ 每一步都沿着损失下降最快的方向进行
- ✓ 一步一步走下去，直到所在的位置就是最低点，没办法再下降
- ✓ 超参：每次更新的步幅大小，更新参数的方式



梯度下降

AI DISCOVERY

？ 2. 如何根据上述差异对网络的权重参数W进行更新

Step2: 权重根据差异方向更新

➢ 梯度下降使用样本方式的变种

- ✓ 批量梯度下降（Batch gradient descent）

  每次迭代时使用所有样本来进行梯度的更新

- ✓ 随机梯度下降（Stochastic gradient descent）

  每次迭代时使用一个样本来进行梯度的更新

- ✓ 小批量梯度下降（Mini-batch gradient descent）

  批量梯度下降和随机梯度下降的折中

  每次迭代时使用batch-size个样本来进行梯度的更新

AI DISCOVERY

梯度下降

AI DISCOVERY

？ 2. 如何根据上述差异对网络的权重参数W进行更新

Step2: 权重根据差异方向更新

➢ 梯度下降方式的优化

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

- ✓ Momentum法

  动量项在梯度指向方向相同的方向逐渐增大，对梯度指向改变的方向逐渐减小

- ✓ Nesterov加速梯度法

  给予梯度项上述「预测」功能的方法

- ✓ Adagrad法

  对低频出现的参数进行大的更新，对高频出现的参数进行小的更新，适合处理稀疏数据

- ✓ Adadelta法
- ✓ RMSprop法
- ✓ Adam法
- ✓ …

更新梯度的具体方式上有所差异

AI DISCOVERY

反向传播

AI DISCOVERY

？ 2. 如何根据上述差异对网络的权重参数W进行更新

Step2: 将差异告诉权重——求损失对权重的梯度

➢ 反向传播算法

✓ 求解损失对所有参数的梯度

✓ 两个过程：

Forward pass: 逐层计算，并保存每一层的参数，方便反向传播计算梯度时使用

Backward pass：根据最后的损失倒序逐层计算每一个参数的梯度，为参数的更新提供依据

【基本原理：链式法则】

✓ 卷积神经网络的backward pass详情涉及更复杂的数学推导，出于时间考虑不再详述

✓ 下页仅描述单个神经元的反向传播过程

AI DISCOVERY

神经元中梯度的计算

AI DISCOVERY

？ 2. 如何根据上述差异对网络的权重参数W进行更新

Step2: 将差异告诉权重——求损失对权重的梯度

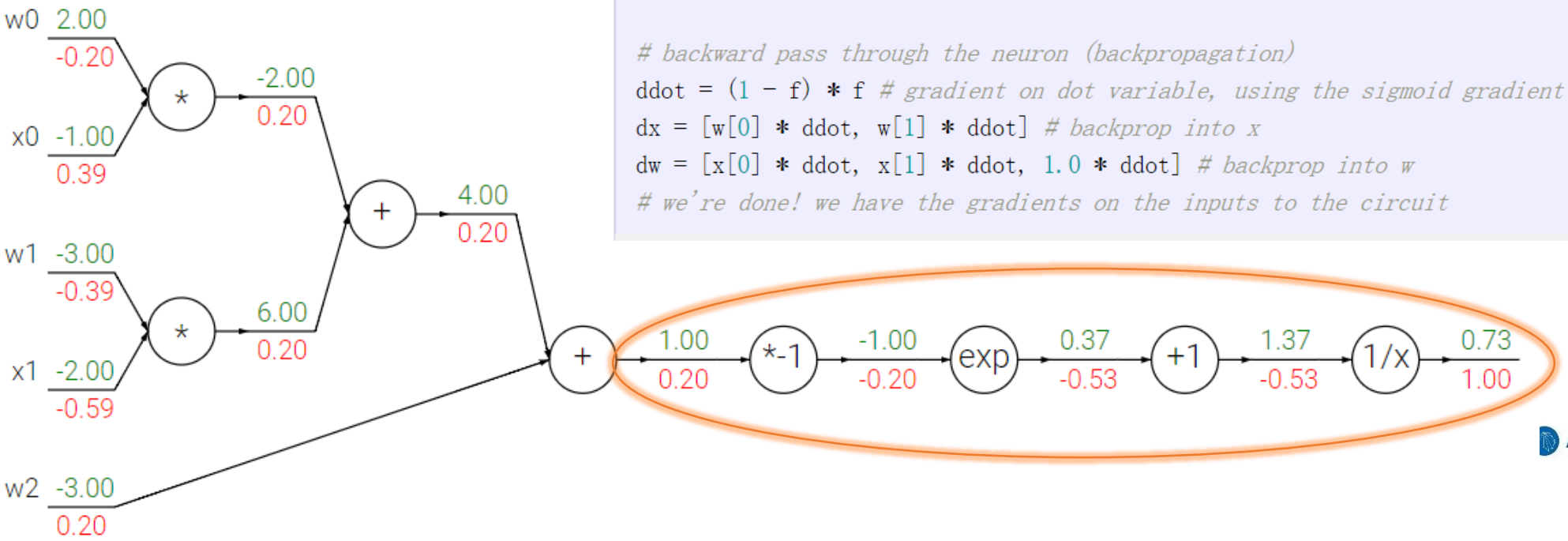
以sigmoid损失为例：

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

```
w = [2,-3,-3] # assume some random weights and data
x = [-1, -2]

# forward pass
dot = w[0]*x[0] + w[1]*x[1] + w[2]
f = 1.0 / (1 + math.exp(-dot)) # sigmoid function

# backward pass through the neuron (backpropagation)
ddot = (1 - f) * f # gradient on dot variable, using the sigmoid gradient derivation
dx = [w[0] * ddot, w[1] * ddot] # backprop into x
dw = [x[0] * ddot, x[1] * ddot, 1.0 * ddot] # backprop into w
# we're done! we have the gradients on the inputs to the circuit
```



AI DISCOVERY

自己动手搭CNN

AI DISCOVERY

CNN基本结构

CNN网络训练

损失与误差的反向传播

模型评估与正则化

第一个CNN

AI DISCOVERY

模型的泛化

AI DISCOVERY

参数越多，越容易过拟合

1）学习算法的基本假设

用来训练模型的数据（训练集）和真实数据（测试集）间是独立同分布的

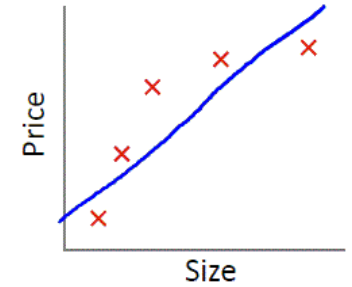
2）泛化能力

机器学习算法对未知样本的适应能力

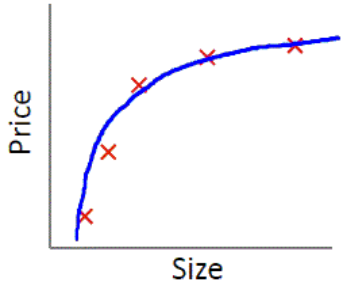
I. 学到隐含在数据集背后的规律

II. 针对具有同一规律的训练集以外的数据

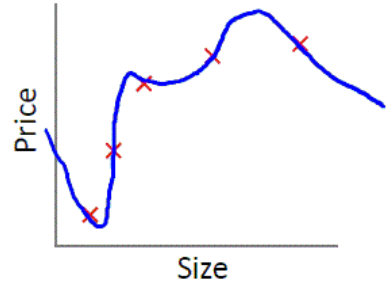
III. 经过训练的网络也能给出合适的输出



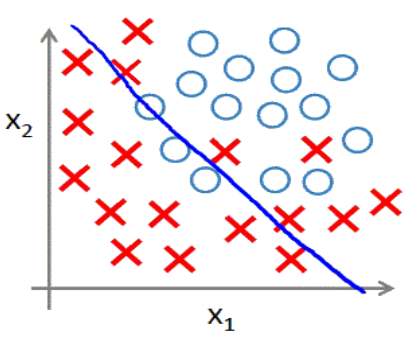
$\theta_0 + \theta_1 x$



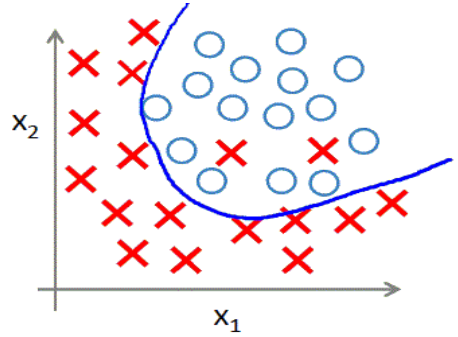
$\theta_0 + \theta_1 x + \theta_2 x^2$



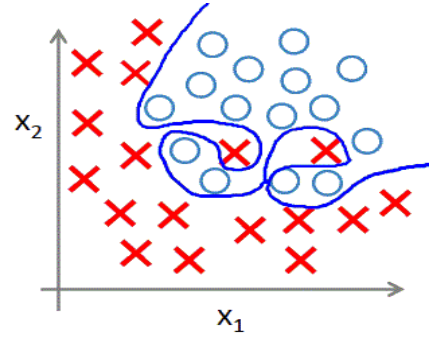
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$



欠拟合



正确拟合



过拟合

AI DISCOVERY

模型的泛化

- **如何提高学习算法效果?**

1）降低训练误差

训练误差高：欠拟合

表现：训练集和测试集上精确度都低

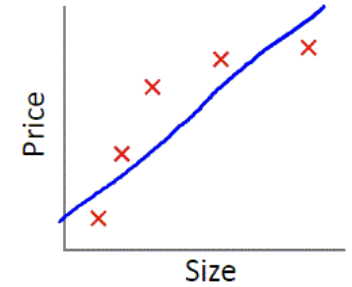
实质：模型的表示能力不够

2）缩小训练误差和测试误差的差距

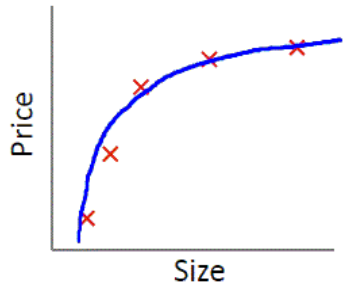
训练误差和测试误差间差距大：过拟合

表现：训练集上精确度高，测试集上精确度低

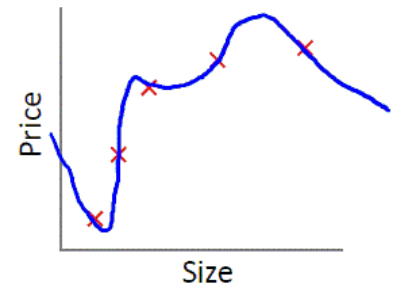
实质：模型模拟了训练数据独有的噪声



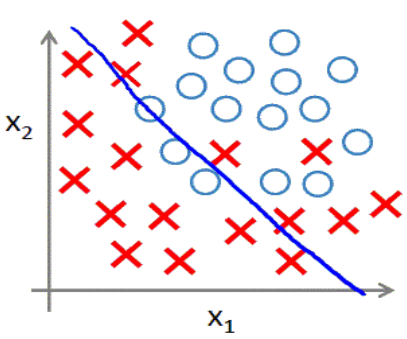
$\theta_0 + \theta_1 x$



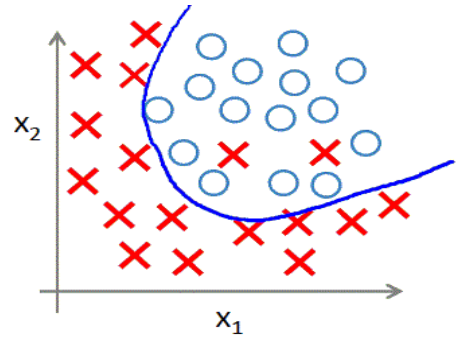
$\theta_0 + \theta_1 x + \theta_2 x^2$



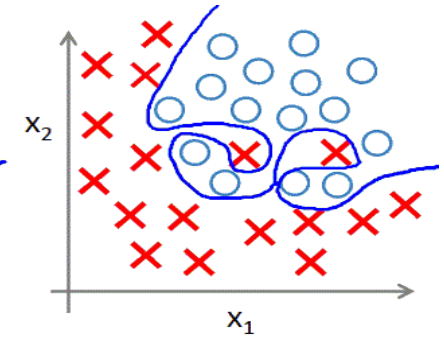
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$



欠拟合



正确拟合



过拟合

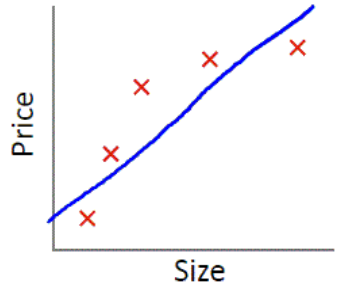
模型的泛化

AI DISCOVERY

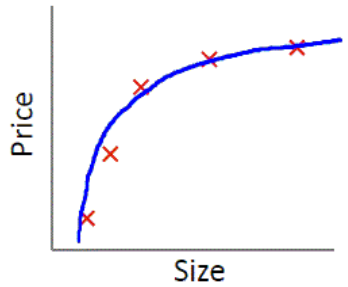
◆ 深度神经网络的泛化能力

1）高模型容量——高拟合各种函数的能力，模型偏向于过拟合

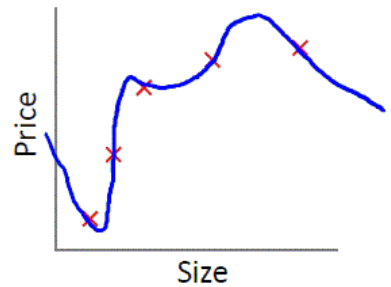
2）正则化——对学习算法的修改，为了减少测试误差（泛化误差）而不是训练误差



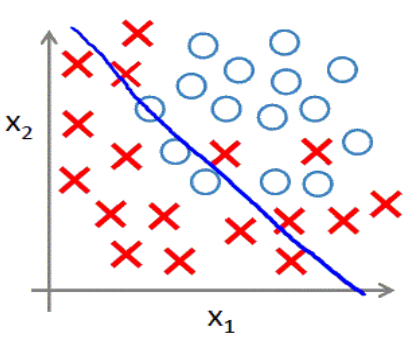
$\theta_0 + \theta_1 x$



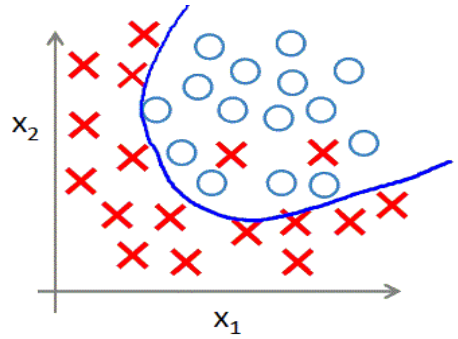
$\theta_0 + \theta_1 x + \theta_2 x^2$



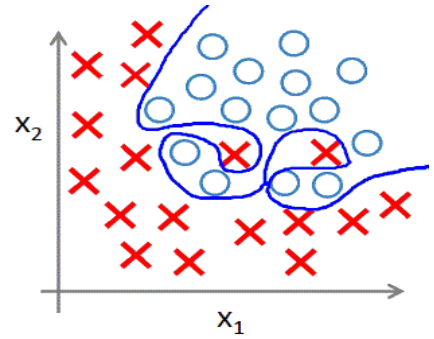
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$



欠拟合



正确拟合



过拟合

AI DISCOVERY

模型的正则化

AI DISCOVERY

➢ 正则化

✓ Early-stopping（早停法）：

检测训练集和验证集上的精确度，

训练集的精确度下降 but 验证集上的精确度上升→则停止训练

✓ 权重正则化：

噪声相比于正常信号而言，通常会在某些点出现较大的峰值，

保证权重系数在绝对值意义上足够小，就能够保证噪声不会被过度响应（模型不应过于复杂）。

L1正则：$J = J_0 + \lambda||w||_1$ ，J代表损失函数，$||w||_1$ 代表参数向量w的L_1范数

L2正则（weight decay）：$J = J_0 + \frac{\lambda}{2}||w||^2$ ，J代表损失函数，$||w||$ 代表参数向量w的L_2范数

将损失函数改写成上述方式，在最小化损失函数的过程中就会限制权重w不会过大。

AI DISCOVERY

✓ 数据增强/ dropout：AlexNet中会进行讲解

自己动手搭CNN

AI DISCOVERY

CNN基本结构

CNN网络训练

第一个CNN

AI DISCOVERY

MNIST

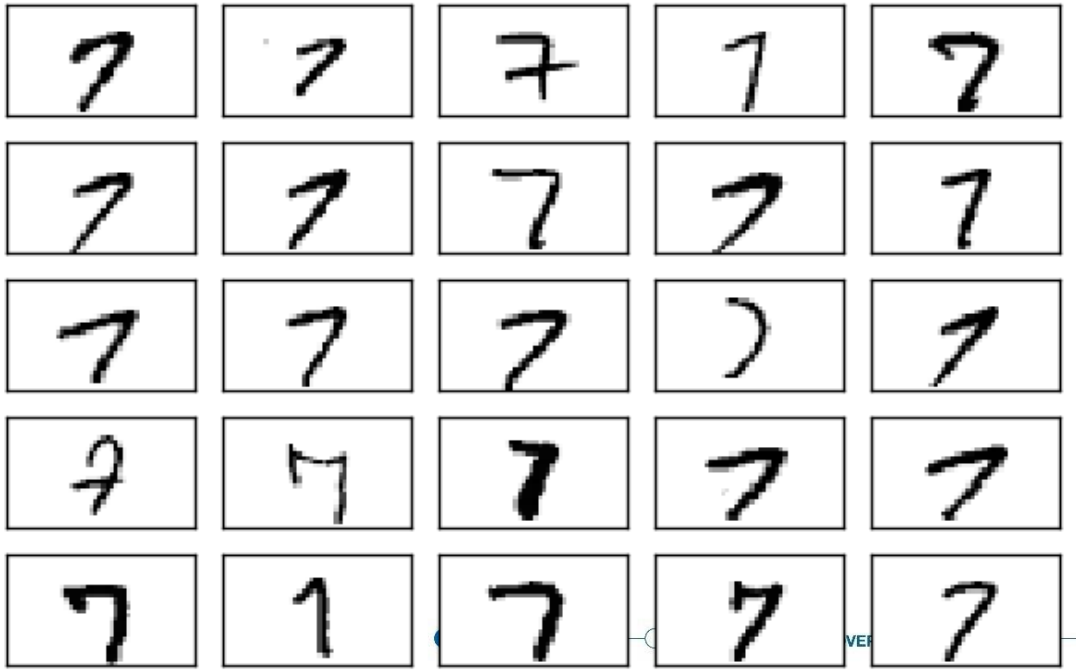
AI DISCOVERY

http://yann.lecun.com/exdb/mnist/

- ✓ 手写数字
- ✓ 10个互斥的类别，28*28的灰度图像
- ✓ 美国国家标准与技术研究所, National Institute of Standards and Technology (NIST)
- ✓ 250个人的手写数字。50%来自人口普查局工作人员，50%来自高中学生。
- ✓ 训练：60,000个样本
- ✓ 测试：10,000个样本

Four files are available on this site:

train-images-idx3-ubyte.gz: training set images (9912422 bytes)
train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)



CIFAR

AI DISCOVERY

http://www.cs.toronto.edu/~kriz/cifar.html

✓ 数据集的类别分布见右图

✓ 60000张32*32 的RGB图像

✓ 10个互斥的类别，每类6000张图片

✓ 训练：50000张，

5个训练批，每批10000张

每类图像随机抽取，张数不平均

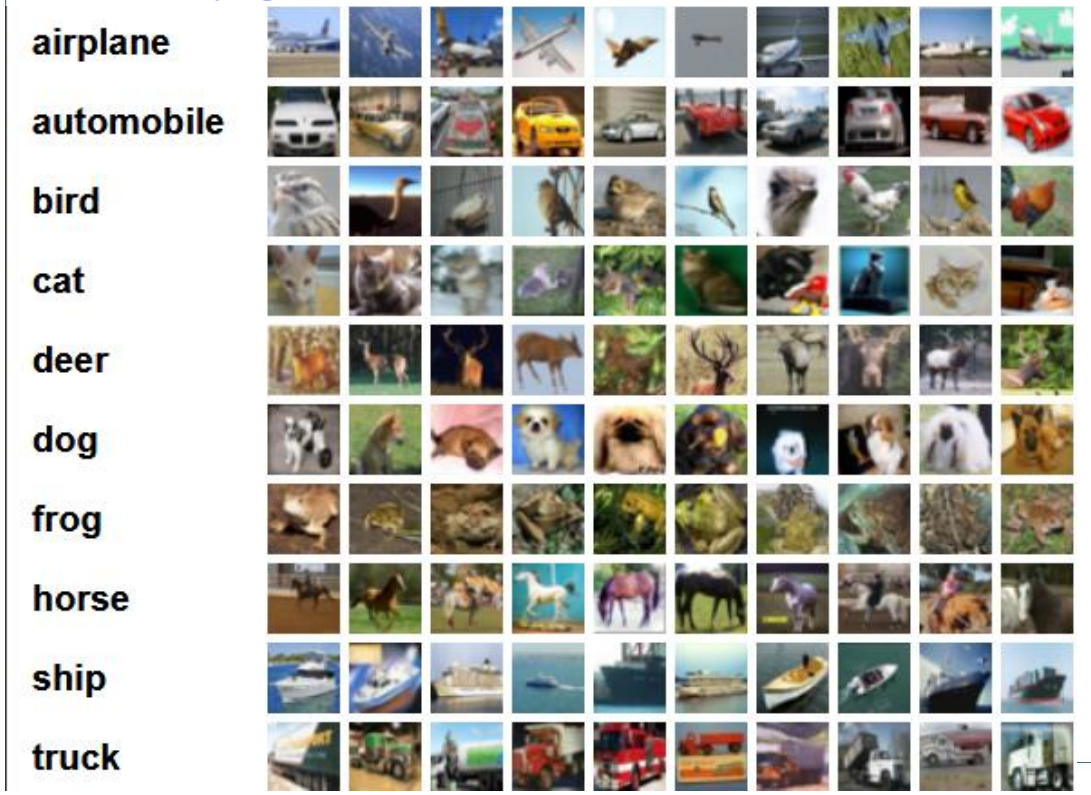
✓ 测试：10000张，

单独构成一批，每类1000张

Download

If you're going to use this dataset, please cite the tech report at the bottom of this page.

Version	Size	md5sum
CIFAR-10 python version	163 MB	c58f30108f718f92721af3b95e74349a
CIFAR-10 Matlab version	175 MB	70270af85842c9e89bb428ec9976c926
CIFAR-10 binary version (suitable for C programs)	162 MB	c32a1d4ab5d03f1284b67883e8d87530



ImageNet

AI DISCOVERY

http://image-net.org/

1）Total number of non-empty synsets: **21841**

2）Total number of images: **14,197,122**

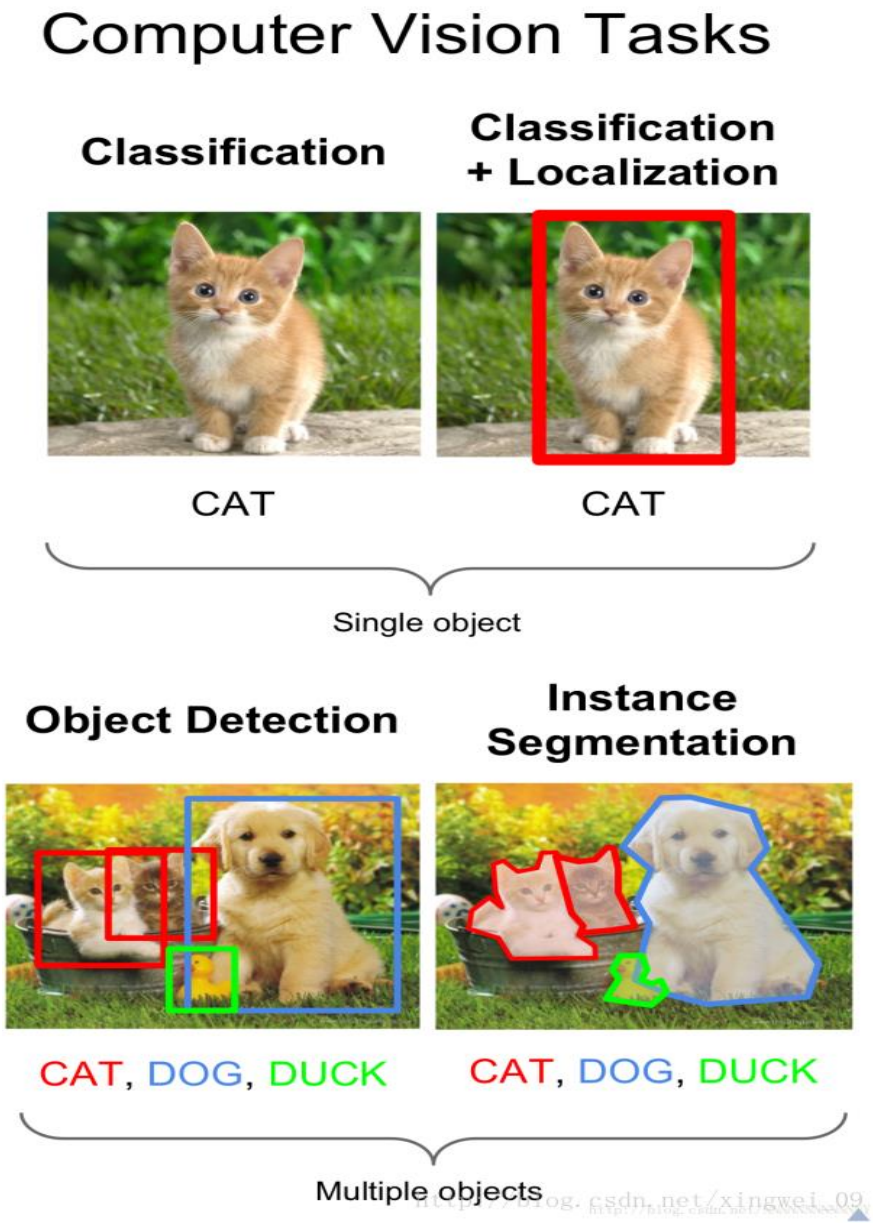
3）Number of images with bounding box annotations: **1,034,908**

4）Number of synsets with SIFT features: **1000**

5）Number of images with SIFT features: **1.2 million**

✓ 可供用于图像分类、目标定位、目标检测、实例分割等多个计算机视觉任务。

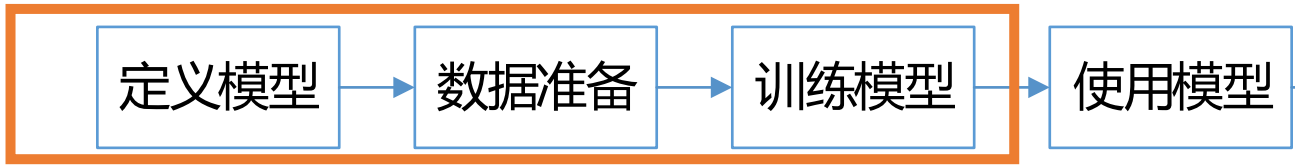
✓ ImageNet国际计算机视觉挑战赛(ILSVRC)



第一个CNN

AI DISCOVERY

通常前三步会迭代多次，以寻找最佳模型



➢ 基础的训练步骤

1. 定义模型的整体框架

1）定义网络结构——神经网络前向传播过程

2）定义优化算法——梯度下降使用方式、参数更新方式、步长设置

3）定义输出日志——

打印训练过程中得到的模型在验证集上的精确度和在训练集上的精确度，获知模型是否过（欠）拟合

4）定义最终获得的模型参数的保存路径

2. 导入数据和预处理

处理数据集的数据，使之适合模型使用

3. 训练模型

把所有的代码交给平台运行，我们就可以喝茶加观察了~

4. 测试模型

使用新数据看一看模型是不是好用

AI DISCOVERY

第一个CNN

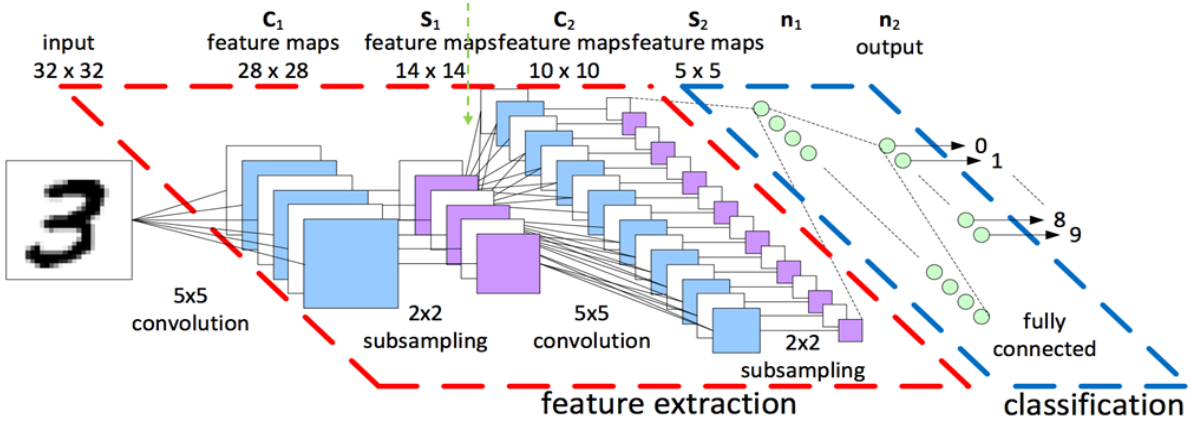
AI DISCOVERY

✓ CNN的模型

以简化版的LeNet-5为例

模型如图所示

详细的参数如下表所示



Layer	Kernels	Stride	Pad	Output	Parameters
Input				32*32*1	
Conv1	5*5*1 filters * 20	1	Valid	28*28*20	(5*5*1+1)*20
Pool2	2*2 filter	2		14*14*20	
Conv3	5*5*20 filters * 50	1	Valid	10*10*50	(5*5*20+1)*50
Pool4	2*2 filter	2		5*5*50	
Output				10	(5*5*50)*10

第一个CNN 前向传播过程

AI DISCOVERY

✓ 定义模型——输入层

以MNIST数据集为例，手写数字是灰度图像

所以模型的输入为32 * 32 =1024维的数据，channel = 1

```
# 定义数据模型，数据大小是32*32 即 1024
img = paddle.layer.data(name="pixel",
                        type=paddle.data_type.dense_vector(1024))          # 手写数字是灰度图像，所以深度（通道数）只有1
```

✓ 定义模型——卷积+池化+激活 的运算组合

```
# 第一个卷积--池化层
conv_pool_1 = paddle.networks.simple_img_conv_pool(input=img,                  # 本层输入的数据：  图像的数据
                                                   filter_size=5,              # 卷积核的大小：5 * 5
                                                   num_filters=20,             # 卷积核的个数 = 特征图的深度 =
                                                                               # 下一层的卷积核的深度 = 想提取的特征数
                                                   num_channel=1,              # 本层输入的深度（通道数）= 图像的深度（通道数）
                                                   pool_size=2,                # 池化核的大小：2 * 2
                                                   pool_stride=2,              # 池化的步长
                                                   act=paddle.activation.Relu())   # 激活函数：ReLU
```

第一个CNN 前向传播过程

AI DISCOVERY

```
# 第二个卷积--池化层
conv_pool_2 = paddle.networks.simple_img_conv_pool(input=conv_pool_1,              # 本层输入的数据：上一层的输出
                                                   filter_size=5,                  # 卷积核的大小：5 * 5
                                                   num_filters=50,                 # 卷积核的个数：50 = 第二个卷积操作组想要提取的特征数
                                                   num_channel=20,                 # 本层输入的深度（通道数） = 上一层特征图的深度（通道数）
                                                   pool_size=2,                    # 池化核的大小： 2 * 2
                                                   pool_stride=2,                  # 池化的步长
                                                   act=paddle.activation.Relu())   # 激活函数： ReLU
```

✓ 定义全连接输出层

对输出结果归一化成概率分布，所以要用Softmax激活函数

```
# 以softmax为激活函数的全连接输出层，输出层的大小必须为数字的个数10
predict = paddle.layer.fc(input=conv_pool_2,
                          size=10,
                          act=paddle.activation.Softmax())
```

AI DISCOVERY

第一个CNN 损失函数、反向传播、正则化

AI DISCOVERY

```
# ********************获取训练器**********************************
def get_trainer(self):
    # 获取分类器
    out = convolutional_neural_network()
    # 定义标签
    label = paddle.layer.data(name="label", type=paddle.data_type.integer_value(10))
    # 获取损失函数
    cost = paddle.layer.classification_cost(input=out, label=label)
    # 获取要训练和优化的参数
    parameters = paddle.parameters.create(layers=cost)
    # 定义方向传播算法优化权重参数的方式
    optimizer = paddle.optimizer.Momentum(learning_rate=0.1 / 128.0,
                                          momentum=0.9,
                                          regularization=paddle.optimizer.L2Regularization(rate=0.0005 * 128))
    # 定义训练三要素——损失、要优化的参数、优化的算法
    trainer = paddle.trainer.SGD(cost=cost,
                                 parameters=parameters,
                                 update_equation=optimizer)
    return trainer
```

✓ 训练过程的定义

根据标签和网络计算出的值来定义损失函数

找到需要优化的参数，损失函数，和反向传播算法的优化策略，即可定义一个优化算法，我们定义的优化算法就是对这个模型的优化训练器

OVERY

第一个CNN 日志的打印，模型的保存

```
# 定义训练事件
def event_handler(event):
    lists = []
    if isinstance(event, paddle.event.EndIteration):
        if event.batch_id % 100 == 0:
            print "\nPass %d, Batch %d, Cost %f, %s" % (
                event.pass_id, event.batch_id, event.cost, event.metrics)
        else:
            sys.stdout.write('.')
            sys.stdout.flush()
    if isinstance(event, paddle.event.EndPass):
        # 保存训练好的参数
        model_path = '../model'
        if not os.path.exists(model_path):
            os.makedirs(model_path)
        with open(model_path + "/model.tar", 'w') as f:
            trainer.save_parameter_to_tar(f=f)
        # 使用测试进行测试
        result = trainer.test(reader=paddle.batch(paddle.dataset.mnist.test(), batch_size=128))
        print "\nTest with Pass %d, Cost %f, %s\n" % (event.pass_id, result.cost, result.metrics)
        lists.append((event.pass_id, result.cost, result.metrics['classification_error_evaluator']))
```

✓ 训练日志的输出

每100个batch输出一次模型在训练集上的损失，每一轮训练结束后，统一输出一次整体的损失，也就是模型能达到的精度值——可以判断模型拟合程度

✓ 模型的保存

每一轮训练结束后，都保存一次参数

第一个CNN 读取数据，开始训练~

AI DISCOVERY

```
# 获取数据
reader = paddle.batch(paddle.reader.shuffle(paddle.dataset.mnist.train(), buf_size=20000), batch_size=128)
trainer.train(reader=reader, num_passes=100, event_handler=event_handler)
```

- ✓ 读取训练数据
  - ✓ Paddle-paddle 在reader中定义读取数据的方式，包括设置缓冲区的大小
  - ✓ 这里是采用批读取的方式，每批中有128张图像
- ✓ 调用trainer.train开始训练过程
  - ✓ 这里定义算法的迭代轮数，一共为100轮

第一个CNN

AI DISCOVERY

✓ 数据的预处理

主要是对测试数据的处理：

RGB空间 → Grey 空间

将图像调整为 32*32 大小

将像素值从[0, 255] 转换到[0, 1]范围内

```
# ******************获取你要预测的参数**********************************
def get_TestData(self, path):
    def load_images(file):
        # 对图进行灰度化处理
        im = Image.open(file).convert('L')
        # 缩小到跟训练数据一样大小
        im = im.resize((32,32), Image.ANTIALIAS)
        im = np.array(im).astype(np.float32).flatten()
        im = im / 255.0
        return im
    test_data = []
    test_data.append((load_images(path),))
    return test_data
```

AI DISCOVERY

第一个CNN 检测过程/使用过程

AI DISCOVERY

```
class TestMNIST:
    def __init__(self):
        # 该模型运行在CPU上，CPU的数量为2
        paddle.init(use_gpu=False, trainer_count=2)

    # ********************获取参数**********************************
    def get_parameters(self):
        with open("../model/model.tar", 'r') as f:
            parameters = paddle.parameters.Parameters.from_tar(f)
        return parameters
    # ********************获取你要预测的参数************************
    def get_TestData(self, path):
```

```
    # ********************使用训练好的参数进行预测**********************************
    def to_prediction(self, out, parameters, test_data):
        # 开始预测
        probs = paddle.infer(output_layer=out,
                             parameters=parameters,
                             input=test_data)
        # 处理预测结果并打印
        lab = np.argsort(-probs)
        print "预测结果为: %d" % lab[0][0]
```

I. 从存储的本地参数文件中读取参数

II. 读取定义好的分类网络

III. 读取测试数据

✓ 调用paddle.infer进行预测

✓ 最后的分类结果就是概率最高的概率所对应的类别

```
if __name__ == "__main__":
    testMNIST = TestMNIST()
    # 开始预测
    out = convolutional_neural_network()
    parameters = testMNIST.get_parameters()
    test_data = testMNIST.get_TestData('../images/infer_3.png')
    testMNIST.to_prediction(out=out, parameters=parameters, test_data=test_data)
```

目录

AI DISCOVERY

AI DISCOVERY

经典CNN

AI DISCOVERY

早期尝试

Hubel & Wiesel

LeNet

历史突破

Dropout
ReLU

AlexNet

发展和演化

网络加深

VGG

增强卷积模块功能

NIN

GoogleNet

Inception V3,V4

增加新的功能单元

Inception V2，BN

融合

ResNet

AI DISCOVERY

经典CNN

AI DISCOVERY

早期尝试

Hubel & Wiesel

LeNet

使用MNIST数据集，
这是最早用于数字识别的CNN

历史突破

2012 ILSVRC远超第2名

ZF Net，基于AlexNet，2013 ILSVRC冠军

AlexNet

Dropout
ReLU

发展和演化

网络加深

VGG

2014 ILSVRC，图像识别略差于GoogLeNet，但是在很多图像分析问题(比如object detection)上效果好

增强卷积模块功能

NIN

GoogLeNet

Inception V3,V4

增加新的功能单元

Inception V2，BN

2014 ILSVRC冠军

融合

ResNet

2015年ILSVRC，
Classification获得第一名

AI DISCOVERY

经典CNN结构探索

AI DISCOVERY

AlexNet

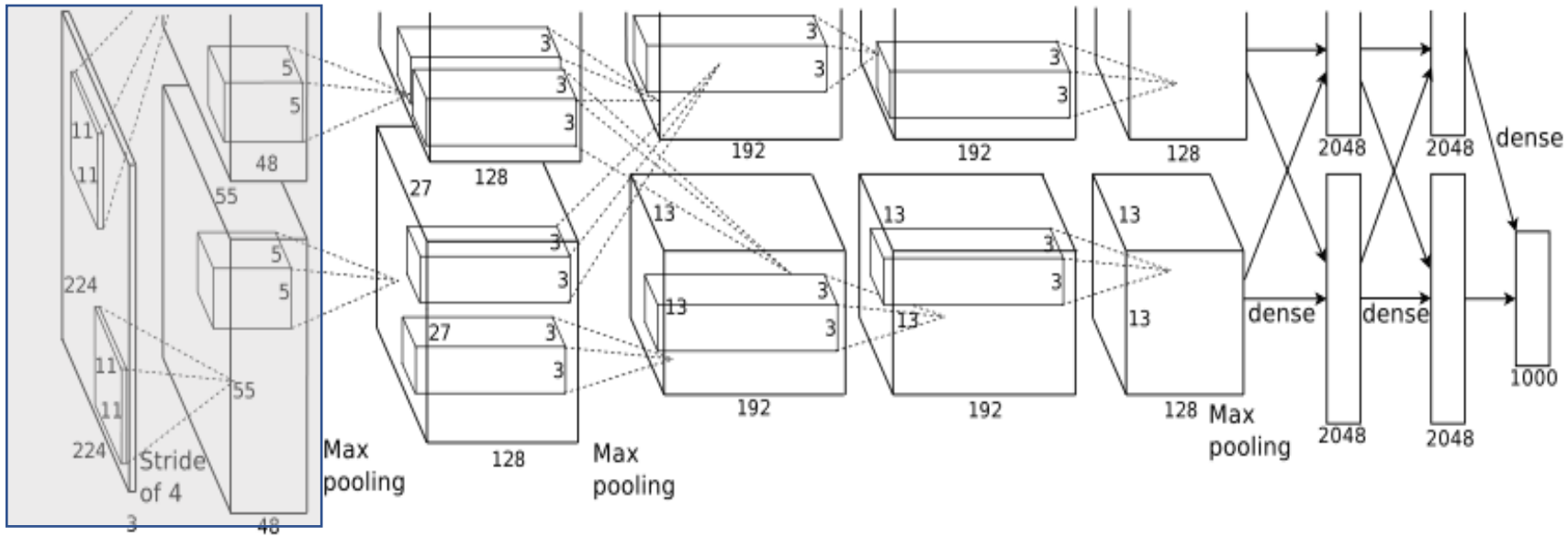
VGG

GoogLeNet/Inception

ResNet

AI DISCOVERY

AlexNet



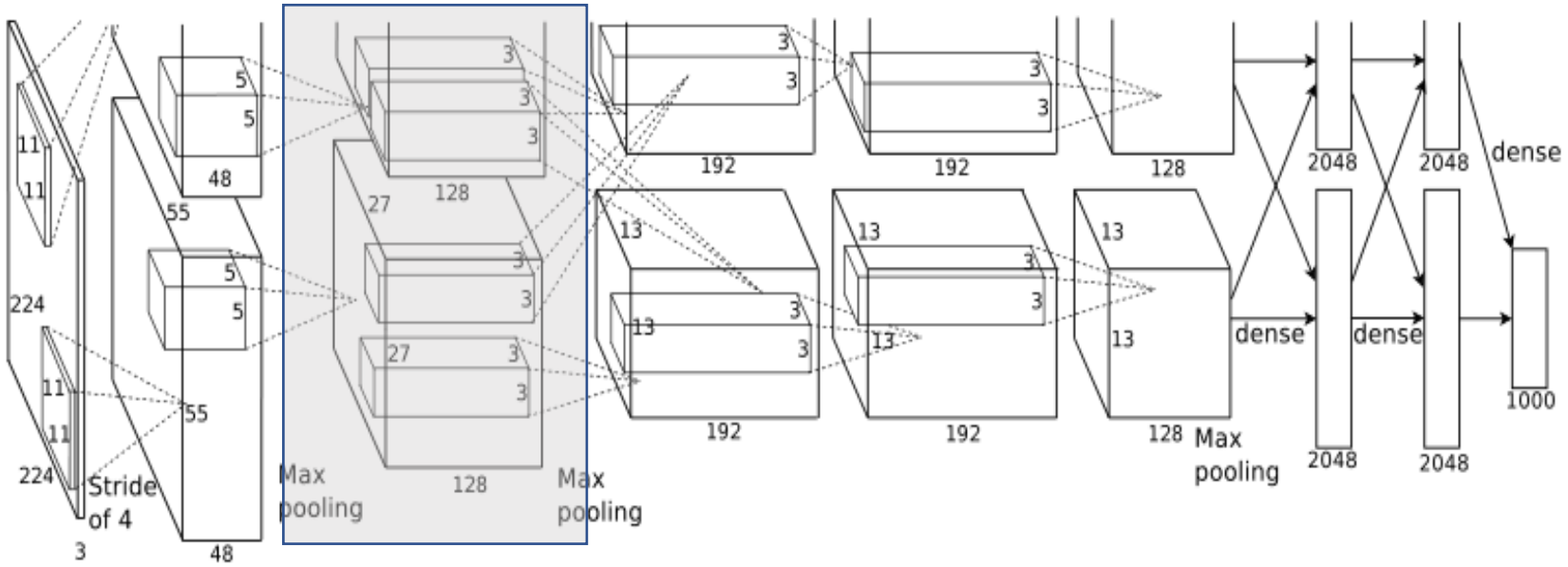
输入图像 （大小 227×227×3）

卷积层 1 227×227×3	卷积核大小 11×11，数量 48 个，步长 4	卷积核大小 11×11，数量 48 个，步长 4
	激活函数（relu）	激活函数（relu）
	池化（kernel size=3, stride=2）	池化（kernel size=3, stride=2）
	标准化	

两台 GPU 同时训练。即共 96 个核。
输出特征图像大小：（227 - 11）/ 4 + 1 = 55，即 55×55×96

输出特征图像大小：（ 55 - 3）/ 2 + 1 = 27，即 27×27×96

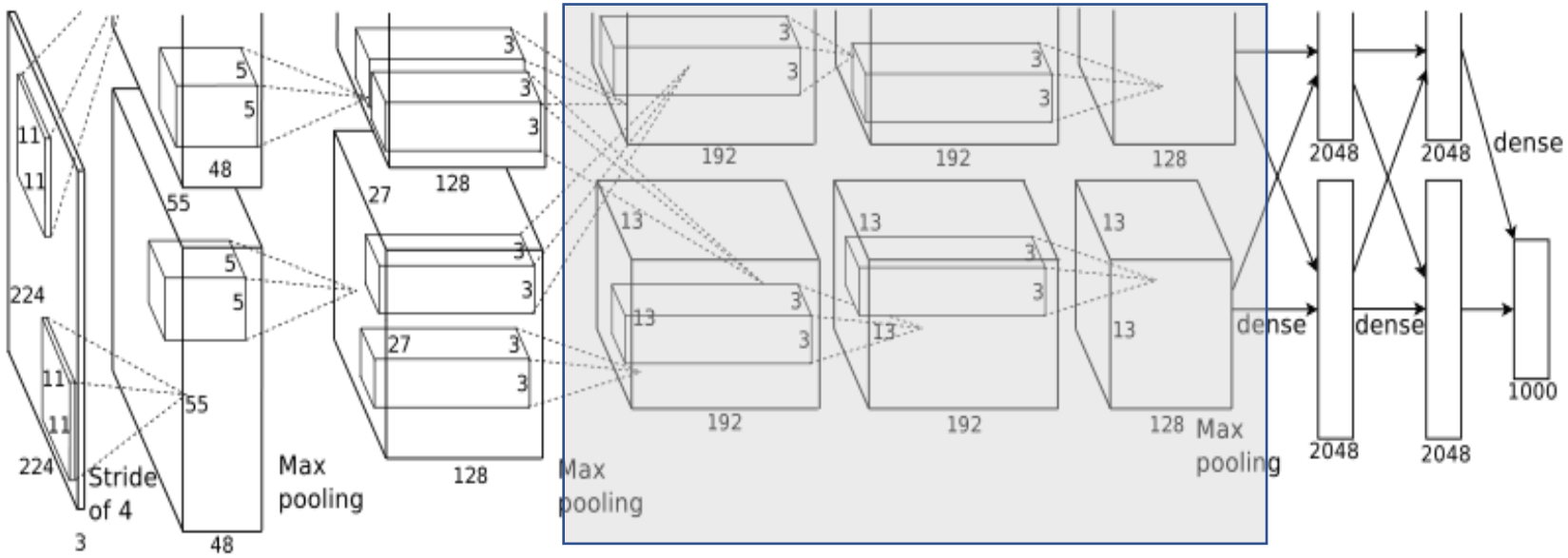
AlexNet



卷积层 2 27×27×96	卷积核大小 5×5，数量 128 个，步长 1	卷积核大小 5×5，数量 128 个，步长 1	输入特征图像先扩展 2 个像素，即大小 31×31 输出特征图像大小：（ 31 - 5）/ 2 + 1 = 27，即 27×27×256
	激活函数（relu）	激活函数（relu）	
	池化（kernel size=3, stride=2）	池化（kernel size=3, stride=2）	输出特征图像大小：（ 27 - 3）/ 2 + 1 = 13，即 13×13×256
	标准化		

AlexNet

AI DISCOVERY



卷积层 3 13×13×256	卷积核大小 3×3，数量 192 个，步长 1	卷积核大小 3×3，数量 192 个，步长 1
	激活函数（relu）	激活函数（relu）
卷积层 4 13×13×384	卷积核大小 3×3，数量 192 个，步长 1	卷积核大小 3×3，数量 192 个，步长 1
	激活函数（relu）	激活函数（relu）
卷积层 5 13×13×384	卷积核大小 3×3，数量 128 个，步长 1	卷积核大小 5×5，数量 128 个，步长 1
	激活函数（relu）	激活函数（relu）
	池化（kernel_size=3, stride=2）	池化（kernel_size=3, stride=2）

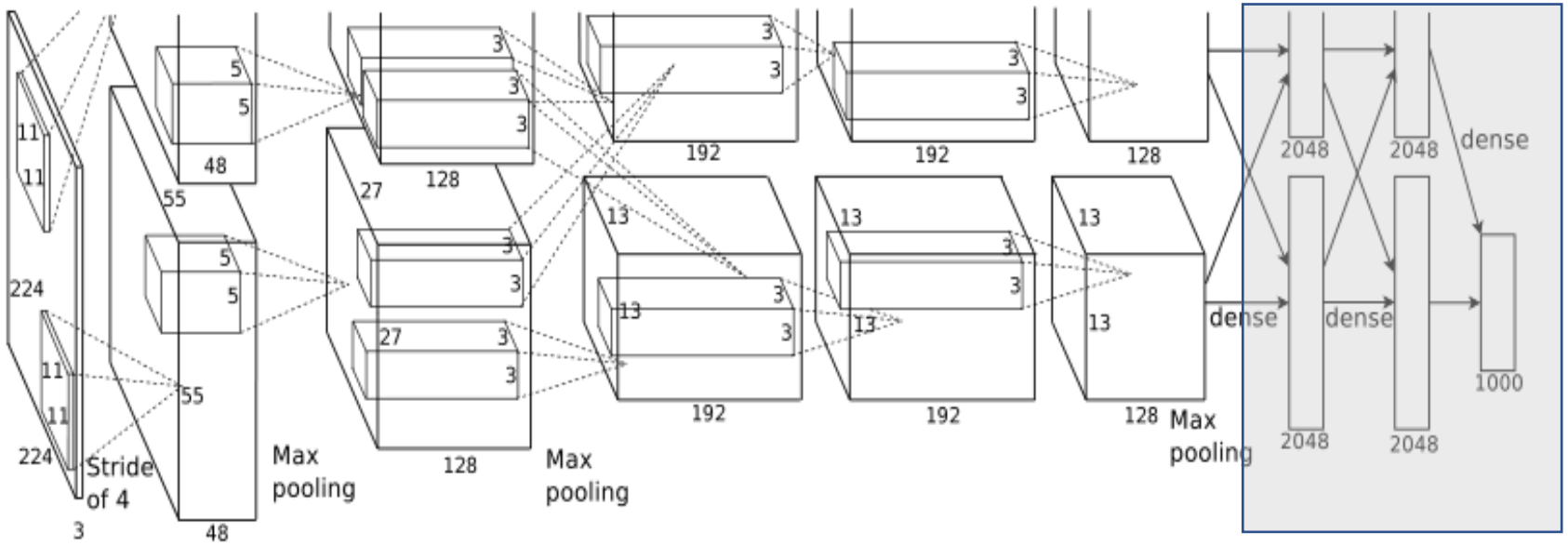
输入特征图像先扩展 1 个像素，即大小 15×15
输出特征图像大小：（ 15 - 3）/1 + 1 = 13，即 13×13×384

输入特征图像先扩展 1 个像素，即大小 15×15
输出特征图像大小：（ 15 - 3）/1 + 1 = 13，即 13×13×384

输入特征图像先扩展 1 个像素，即大小 15×15
输出特征图像大小：（ 15 - 3）/ 1 + 1 = 13，即 13×13×256

输出特征图像大小：（ 13 - 3）/ 2 + 1 = 6，即 6×6×256

AlexNet

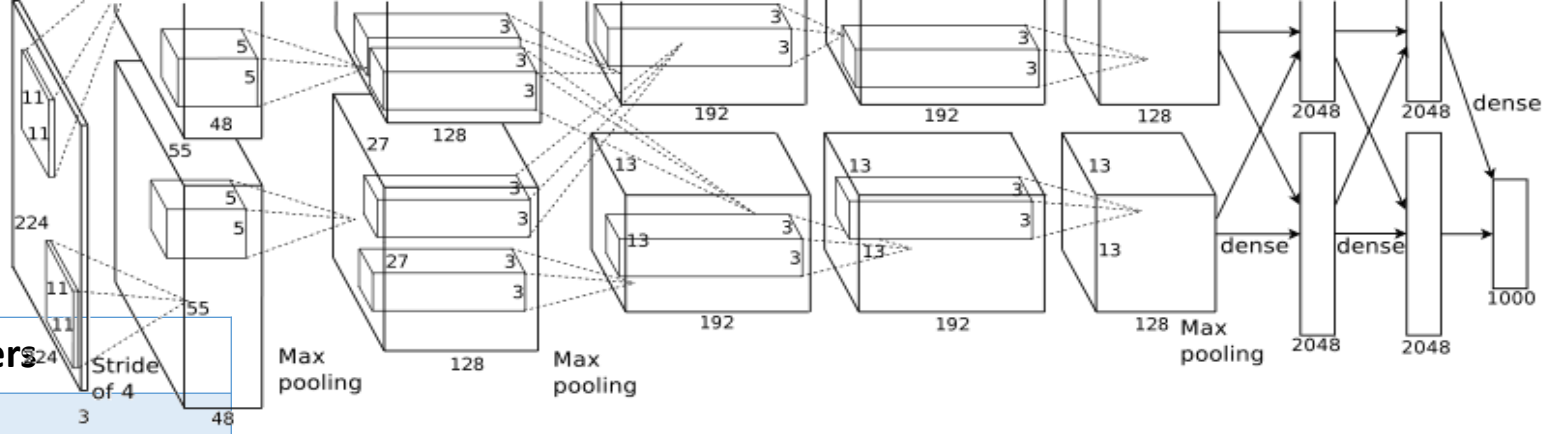


全连接 6 6×6×256	2048 个神经元	2048 个神经元	共 4096 个神经元
	dropout	dropout	输出 4096×1 的向量
全连接 7 4096×1	2048 个神经元	2048 个神经元	共 4096 个神经元
	dropout	dropout	输出 4096×1 的向量
全连接 8 4096×1	1000 个神经元		输出 1000×1 的向量

AlexNet

AI DISCOVERY

Layer	Output	Parameters
Input	227*227*3	
Conv1	55*55*96	(11*11*3+1)*96=34944
MaxPool1	27*27*96	0
Norm1		
Conv2	27*27*256	(5*5*48+1)*128*2
MaxPool2	13*13*256	0
Norm2		
Conv3	13*13*384	(3*3*256+1)*384
Conv4	13*13*384	(3*3*192+1)*192*2
Conv5	13*13*256	(3*3*192+1)*128*2
MaxPool3	6*6*256	0
Fc6+Dropout	4096	(6*6*128*2+1)*4096
Fc7+Dropout	4096	(4096+1)*4096
Fc8	1000	(4096+1)*1000



✓ Details:

分组卷积

第一次使用ReLU

data augmentation

dropout = 0.5

batch-size = 128

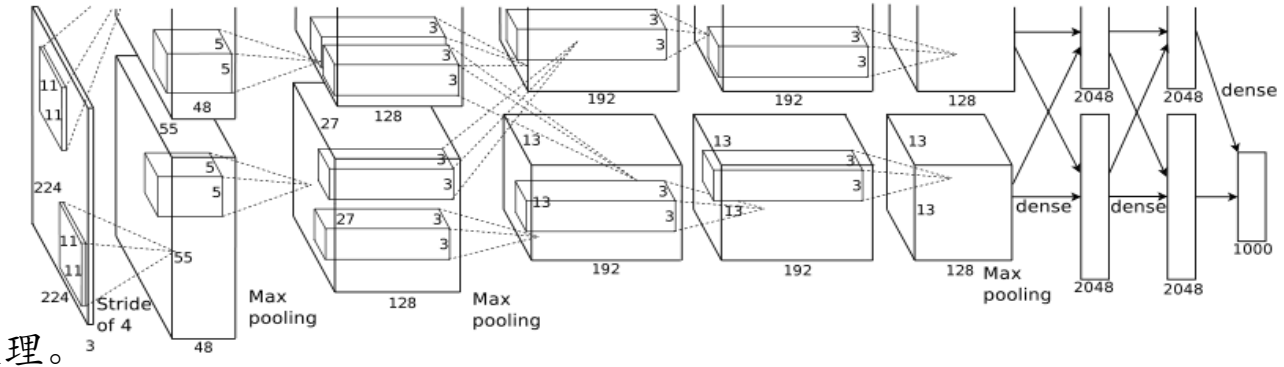
SGD momentum = 0.9

learning-rate = 1e-2

L2 weight-decay = 5e-4

AlexNet

AI DISCOVERY



✓ Group Convolution（分组卷积）：

最早出现于AlexNet

硬件资源有限，卷积操作不能放在同一个GPU进行处理。

Intuition：把feature maps分给多个GPU，对多个GPU的结果进行融合。

✓ Group Convolution的计算量评估：

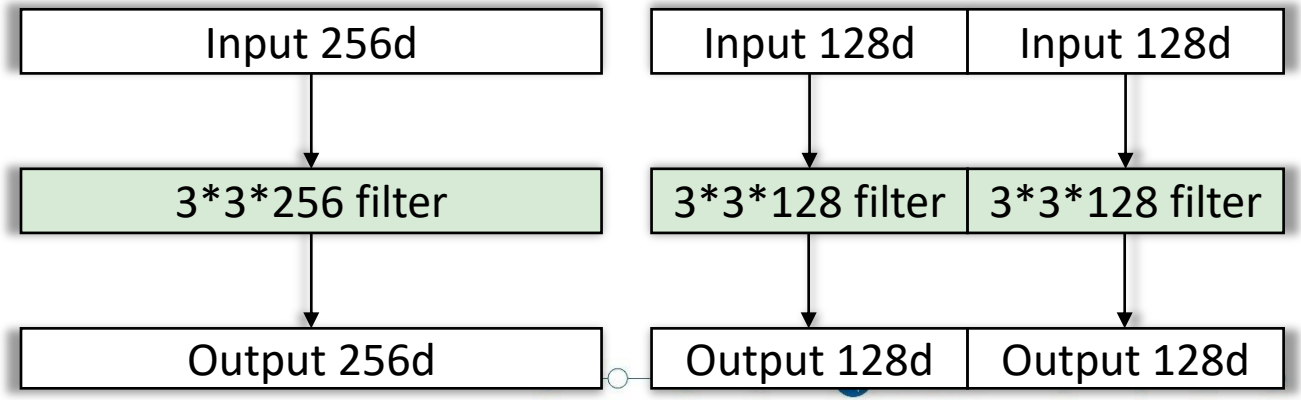
每个GPU的计算量会降低到整个模型的1/groups^2

整体GPU的计算量会下降为1/groups

Input channel = 256，Output channel = 256，kernel = 3*3

不分组的卷积：256*3*3*256 parameters

分为2组的卷积：128*3*3*128 parameters * 2



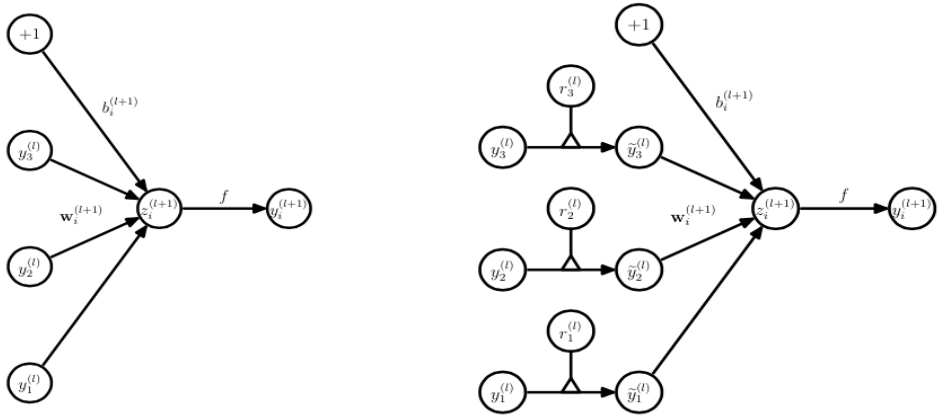
AlexNet

AI DISCOVERY

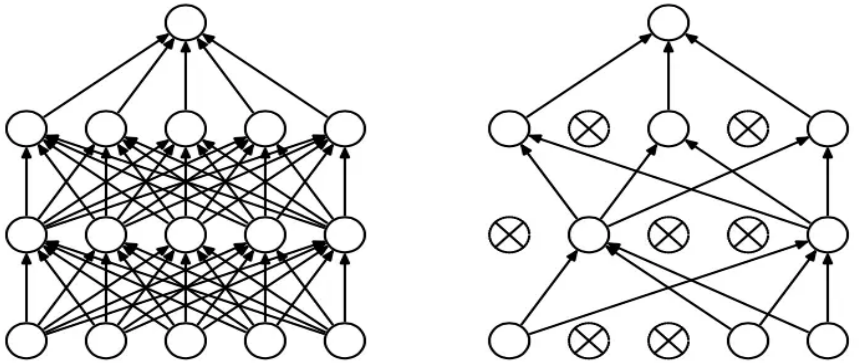
✓ Dropout原理和使用：

1 引入概率的思想，训练时节点是否存在于网络中是由随机概率决定的

2 训练时加入随机性，进行反向传播算法。测试时，只是用训练好的参数



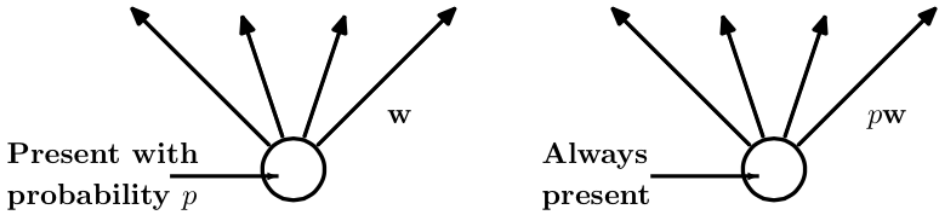
(a) Standard network (b) Dropout network



N个节点的网络，Dropout = 同时训练2^n个网络

是ensemble在分类性能上的一个近似，减弱了神经元节点间的联合适应性，在不大幅增加计算量的情形下防止过拟合，增强了泛化能力

dropout = 0.5效果最好，随机生成的网络结构最多



(a) At training time (b) At test time

AlexNet

AI DISCOVERY

✓ Data Augmentation：

拟合自然界中常见的噪声

主要用于数据集较小的时候，可以丰富图像训练集、防止过拟合

✓ 常用的数据增强方式有：

对颜色的数据增强

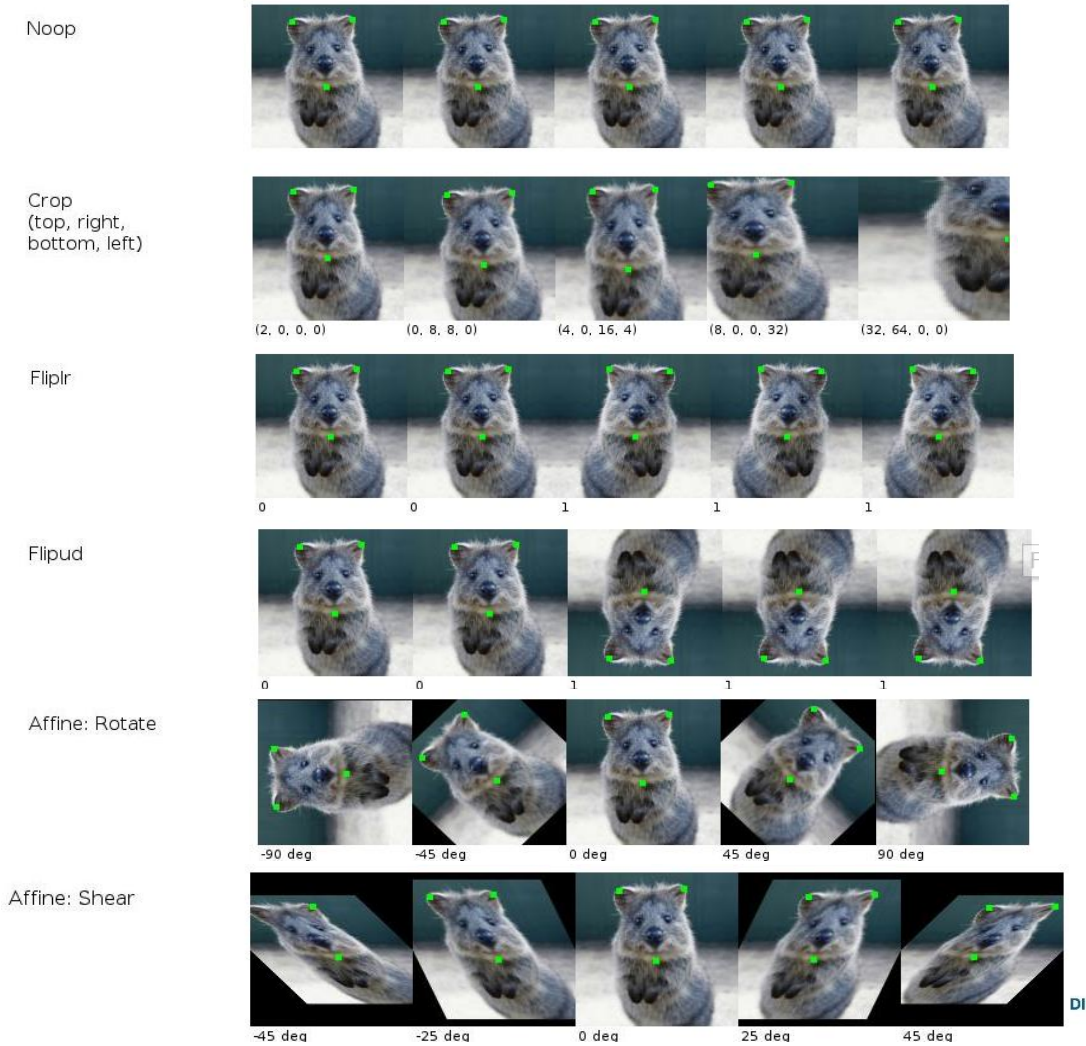
色彩的饱和度、亮度、对比度

加噪声（高斯噪声）

水平翻转、垂直翻转

随机旋转、随机裁剪（crop）

https://github.com/aleju/imgaug



DISCOVERY

经典CNN结构探索

AlexNet

VGG

GoogLeNet/Inception

ResNet

VGG

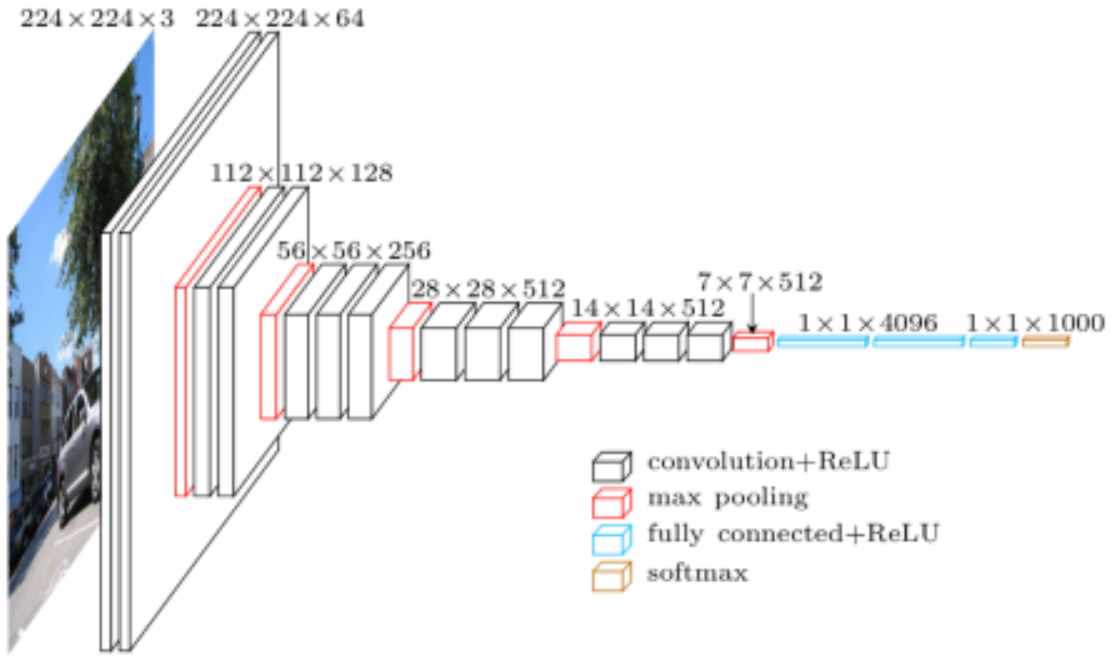
AI DISCOVERY

牛津大学计算机视觉组

深度增加 & 小卷积核 → 对网络最后的分类识别效果有很大作用

小卷积核： 3*3：表示上下、左右、中心这些概念的最小卷积核尺寸

深度：AlexNet 8 层 → VGG最深19层



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG

AI DISCOVERY

✓ Details:

3*3 conv-filter

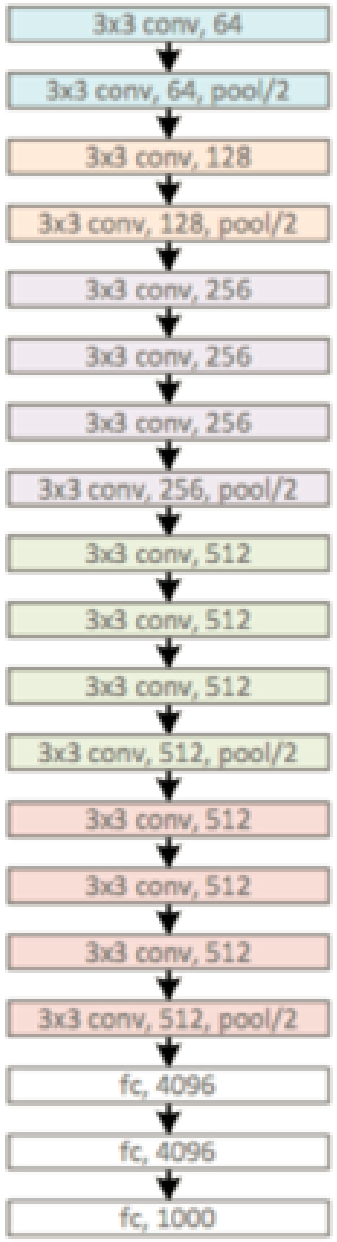
conv-stride = 1

pad = 1, Same

2*2 pool-filter

pool-stride = 2

ILSVRC 2014



Layer	Output	Parameters
Input	224*224*3	Note:未计算偏置
Conv3-64	**224*224*64**	(3*3*3)*64 = 1728
Conv3-64	**224*224*64**	(3*3*64)*64 = 36864
Pool2	112*112*64	0
Conv3-128	112*112*128	(3*3*64)*128 = 73728
Conv3-128	112*112*128	(3*3*128)*128 = 147456
Pool2	56*56*128	0
Conv3-256	56*56*256	(3*3*128)*256 = 294912
Conv3-256	56*56*256	(3*3*256)*256 = 589824
Conv3-256	56*56*256	(3*3*256)*256 = 589824
Pool2	28*28*256	0
Conv3-512	28*28*512	(3*3*256)*512 = 1179648
Conv3-512	28*28*512	(3*3*512)*512 = 2359296
Conv3-512	28*28*512	(3*3*512)*512 = 2359296
Pool2	14*14*512	0
Conv3-512	14*14*512	(3*3*512)*512 = 2359296
Conv3-512	14*14*512	(3*3*512)*512 = 2359296
Conv3-512	14*14*512	(3*3*512)*512 = 2359296
Pool2	7*7*512	0
Fc	1*1*4096	**(7*7*512)*4096**
Fc	1*1*4096	4096*4096
Fc	1*1*1000	4096*1000

VGG

AI DISCOVERY

✓ Small Convolution filter：

最早用于VGG，Inception2中明确指出该设计准则

AlexNet用到一些非常大的卷积核，比如11*11、5*5。

Intuition：感受野越大、看到的图片信息就越多，获得的特征会越好

参数和计算量的增加，如何衡量该使用多大的卷积核?

Methods：使用2个3*3的卷积核的组合比用1个5*5的卷积核

效果更佳 && 参数量降低

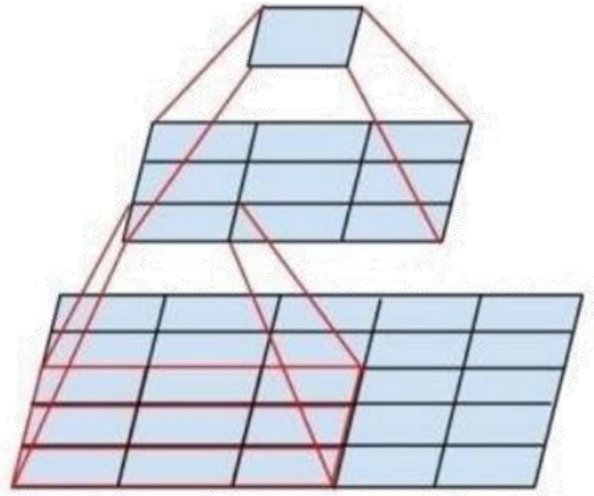


Figure 1. Mini-network replacing the 5 × 5 convolutions.

✓ 参数量的评估：

以256通道的隐层数据为例：

2个3*3的卷积核 == 1个5*5的卷积核

2*3*3*256 params 5*5*256 params

3个3*3的卷积核 == 1个7*7的卷积核

3*3*3*256 params 7*7*256 params

经典CNN结构探索

AI DISCOVERY

AlexNet

VGG

GoogLeNet/Inception

ResNet

AI DISCOVERY

GoogLeNet/Inception

AI DISCOVERY

Google提出的神经网络，一共有V1-V4共4个不同的模型

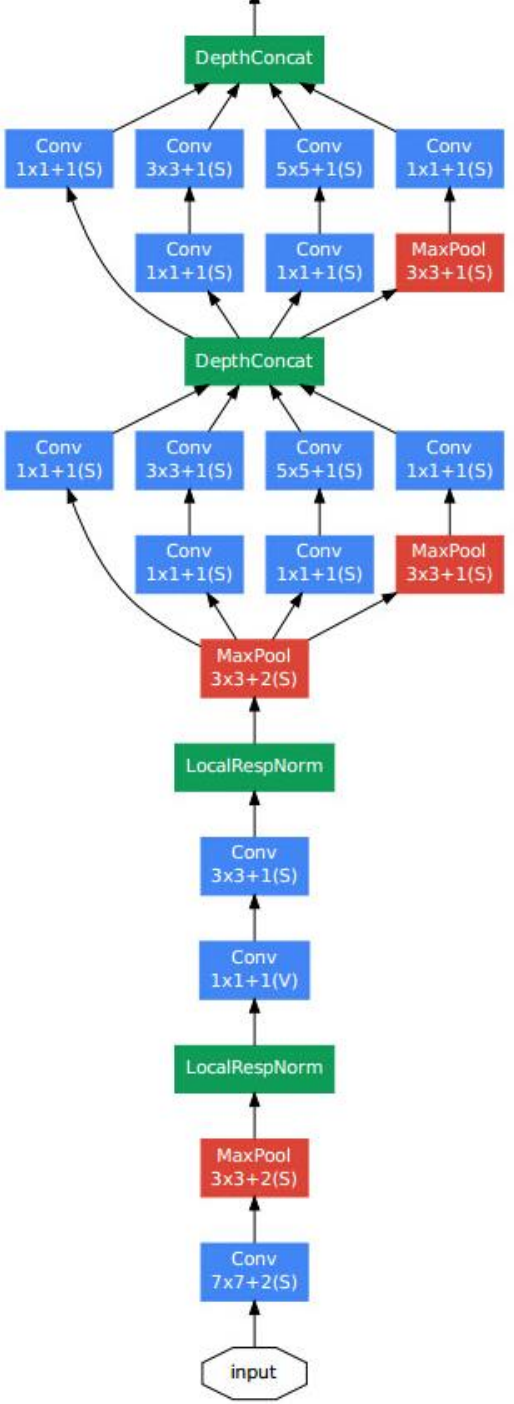
✓ **使用了Inception模块**

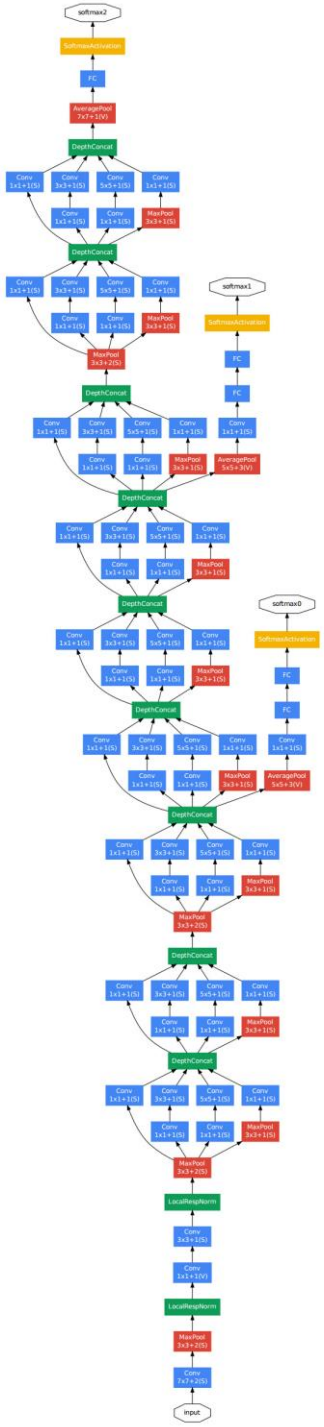
- 可以并行执行多个具有不同尺度的卷积运算或池化操作

  将多个卷积核卷积的结果拼接成一个非常深的特征图

✓ **使用了大量的trick提高网络性能**

- Bottleneck（瓶颈）：1*1的卷积核，借鉴NIN
- 使用全局平均池化GAP代替全连接
- 在v2中，采用Batch Normalization（批归一化）
- 在v3中，采用非对称卷积降低运算量
- 在v4中，结合了ResNet中的思想，发现Residual Connections貌似只能加快网络收敛速度，是更大的网络规模提高了精度





GoogLeNet/Inception

AI DISCOVERY

✓ Multi-size filters in one layer（Inception Block）：

最早出现于Inception-v1/GoogleNet

传统的堆叠式网络，每层仅用一个尺寸的卷积核。

VGG每层只使用3*3的卷积核。

Intuition：信息位置的巨大差异，

使得不同尺度的特征结合起来可以得到更好的特征表示

Methods：卷积核设为1，3，5，stride设为1

可以使用 pad = 0, 1, 2 很方便的对齐

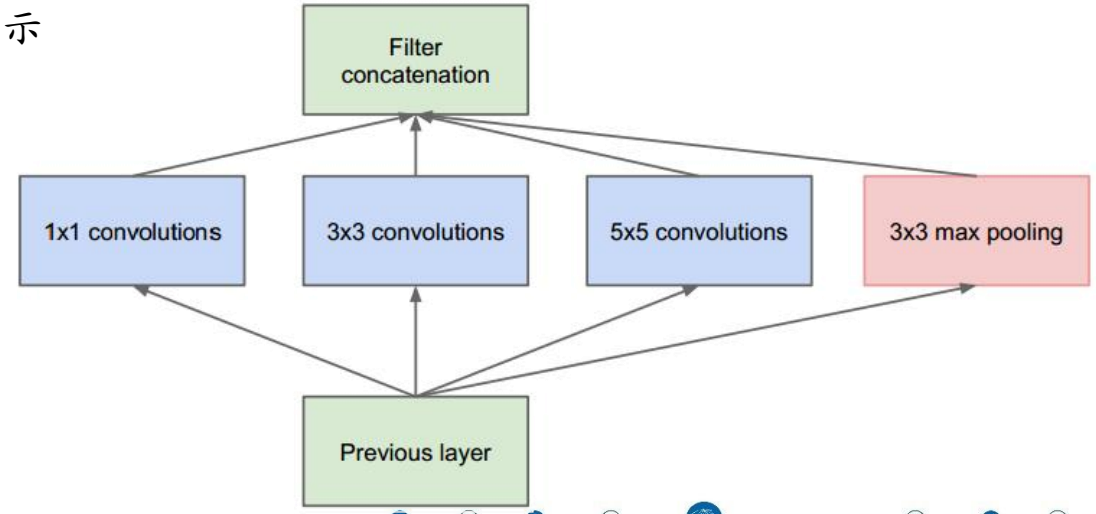
结合的方式：在depth维度上Concatenation（拼接）

Disadvantage：

参数量比使用单个卷积核要多

庞大的计算量 = 模型效率低下

https://unsplash.com/



AI DISCOVERY

(a) Inception module, naïve version

GoogLeNet/Inception

AI DISCOVERY

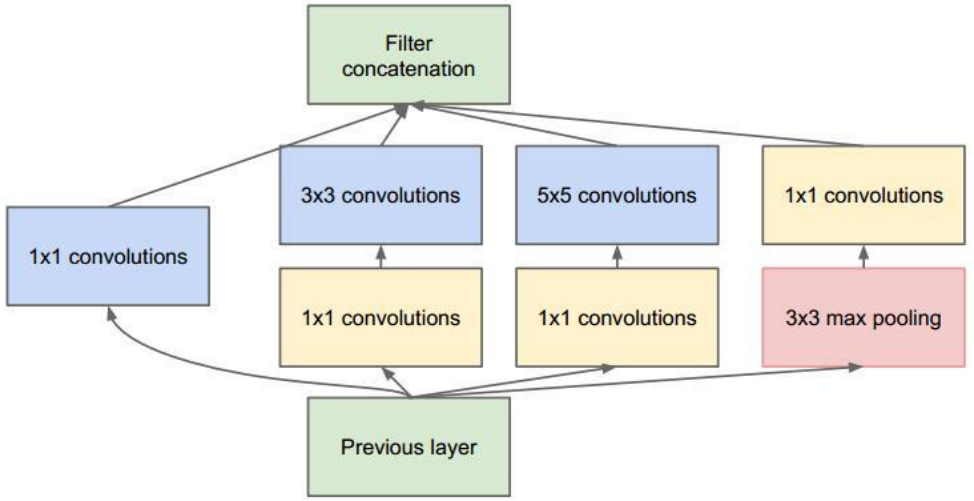
✓ Bottleneck：

最早出现于Network in Network

在同一层使用多个不同尺寸的卷积核会导致参数过多

Intuition：Network in Network中1*1的卷积核的启发

在原来的Inception结构中加入1*1的卷积核



(b) Inception module with dimension reductions

✓ Bottleneck参数量评估：

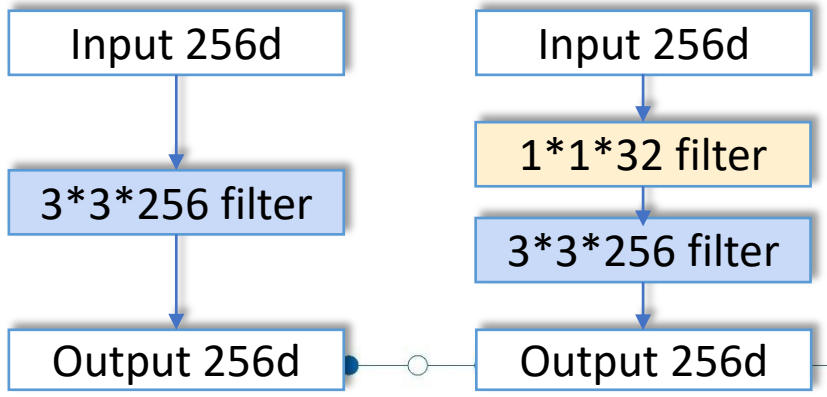
Input channel = 256，Output channel = 256

卷积核 = 3*3*256

参数量：256*3*3*256 = 589,824

卷积核：1*1*32，3*3*256

参数量：256*1*1*32+32*3*3*256=81,920



GoogLeNet/Inception

AI DISCOVERY

✓ 使用全局平均池化GAP代替全连接：

多设计几个全连接层可以提升神经网络的分类性能，全连接层一度成为标配

Intuition：全连接层参数量巨大，

特别是与最后一个卷积层相连接的全连接层

Methods：全局平均池化 Global Average Pooling

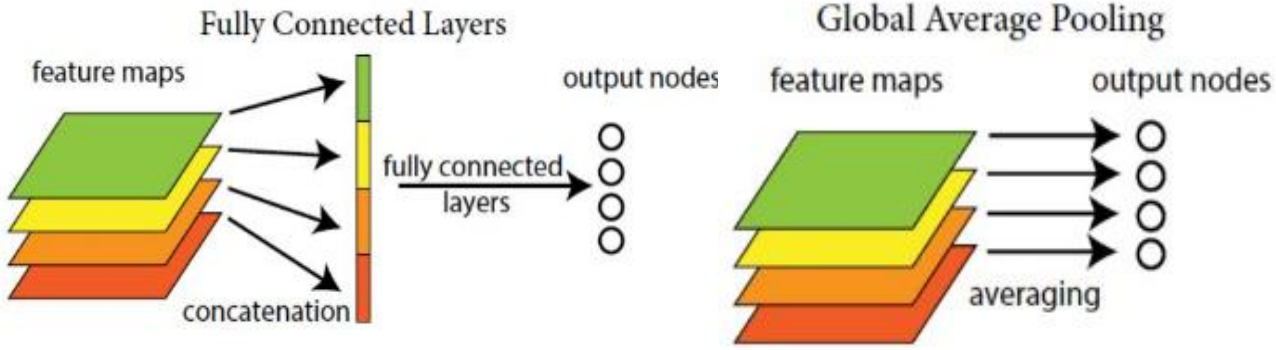
代替全连接

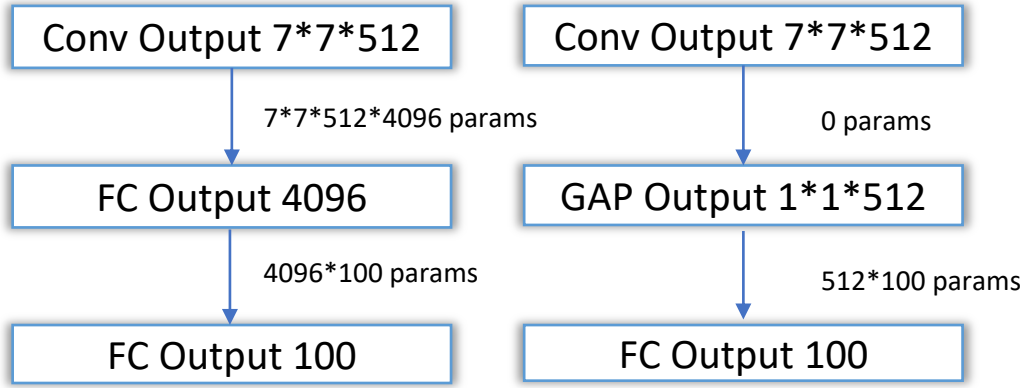
Advantage：

可以实现任意图像大小的输入

Disadvantage：

可能会导致收敛速度变慢

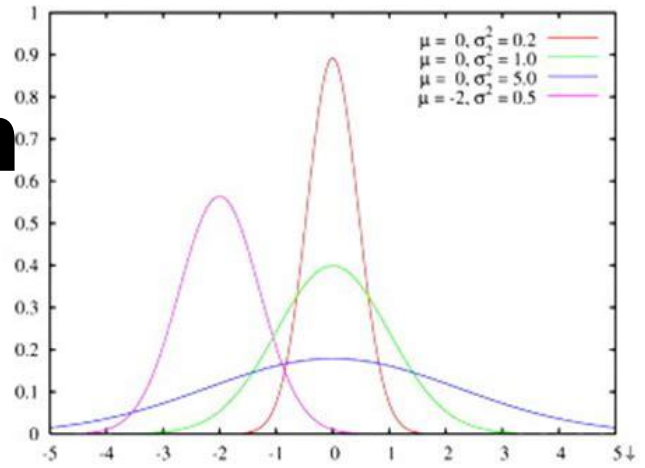




GAP参数量（计算量）评估

GoogLeNet/Inception

AI DISCOVERY



- ✓ Batch Normalization（批归一化）：

  最早出现于Inception v2

  训练的过程中，隐层输入的分布一直在变化

  学习的过程又要使每一层适应输入的分布，

  会导致反向传播时低层神经网络梯度的减少甚至消失

  这是训练深层神经网络收敛越来越慢的**本质原因**

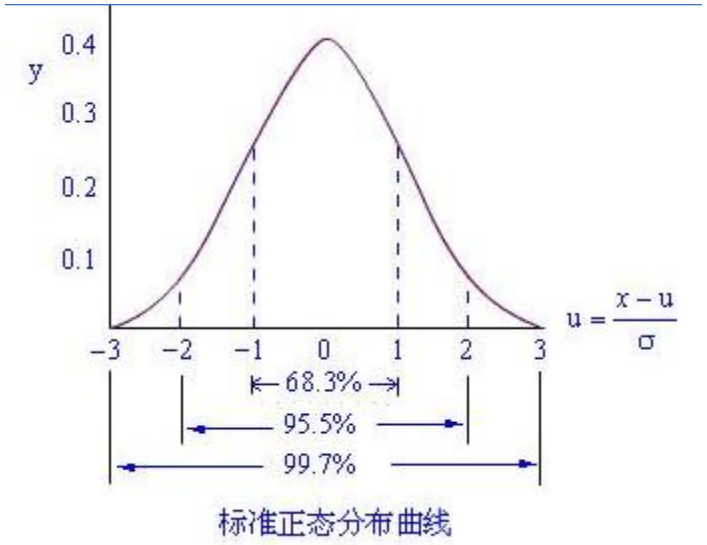
  Intuition：在训练过程中强行将每一层的输入

  拉回到比较标准的相同的分布

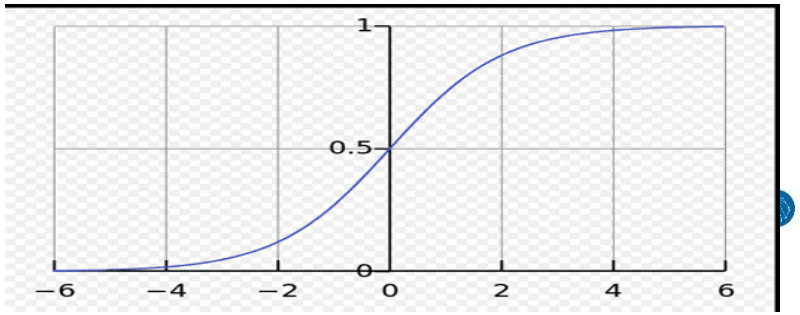
  [白化操作能尽快使神经网络收敛的启发]

  可以防止梯度消失或爆炸

  可以加快训练速度



标准正态分布曲线



AI DISCOVERY

经典CNN结构探索

AI DISCOVERY

AlexNet

VGG

GoogLeNet/Inception

ResNet

AI DISCOVERY

ResNet

AI DISCOVERY

微软提出的神经网络，CVPR当年的最佳论文

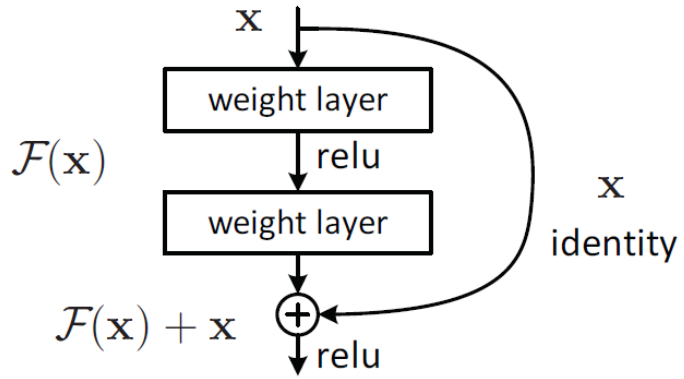
✓ **使用了恒等映射**

传统神经网络训练的函数为F(x)

添加恒等映射后，神经网络训练的函数变为F(x)+x

作者认为这样训练出来的网络，相当于是在对x作修正，修正的幅度就是F(x)， F(x)在数学上称为残差

所以作者提出的网络称为残差网络



method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

ResNets @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

ResNet

AI DISCOVERY

✓ 两个相同深度网络的比较（训练误差方面）

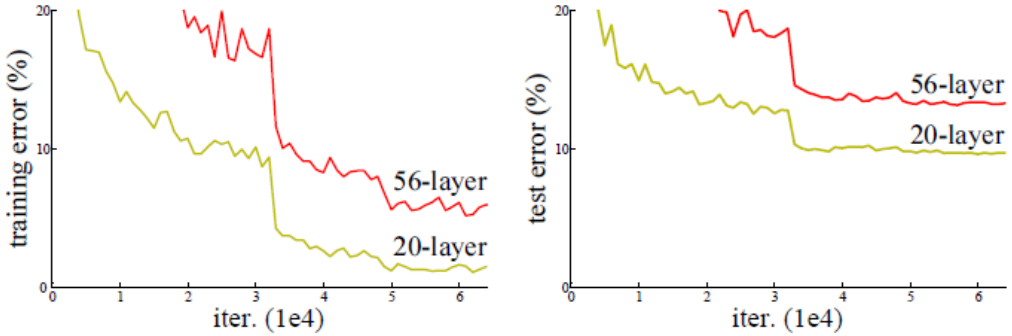
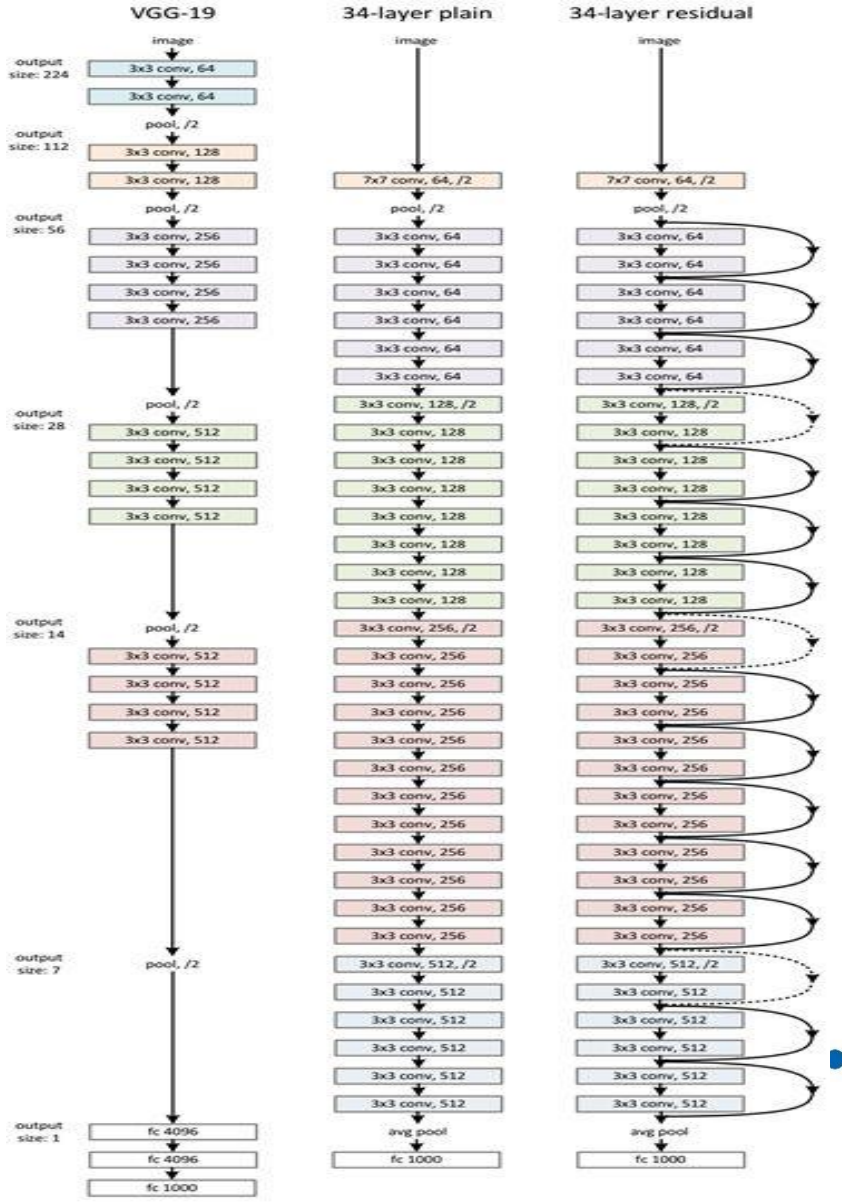


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.



ResNet

AI DISCOVERY

➢ 一点直觉：

平凡网络中，为什么深度增加到一定程度后，训练的效果反而下降了？

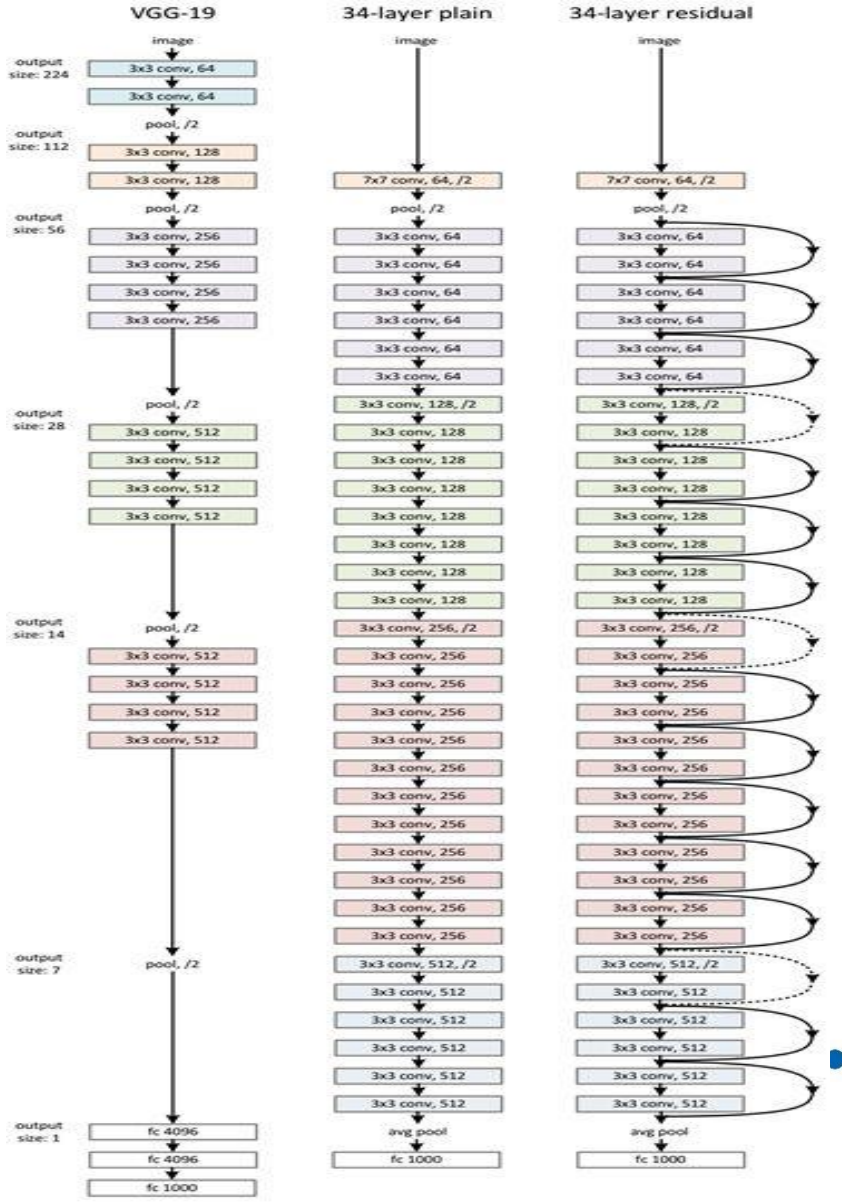
梯度下降算法本身的缺陷

深度增加后，相比浅一些的网络来说梯度减小。误差传播过程变慢，网络的优化速度就慢

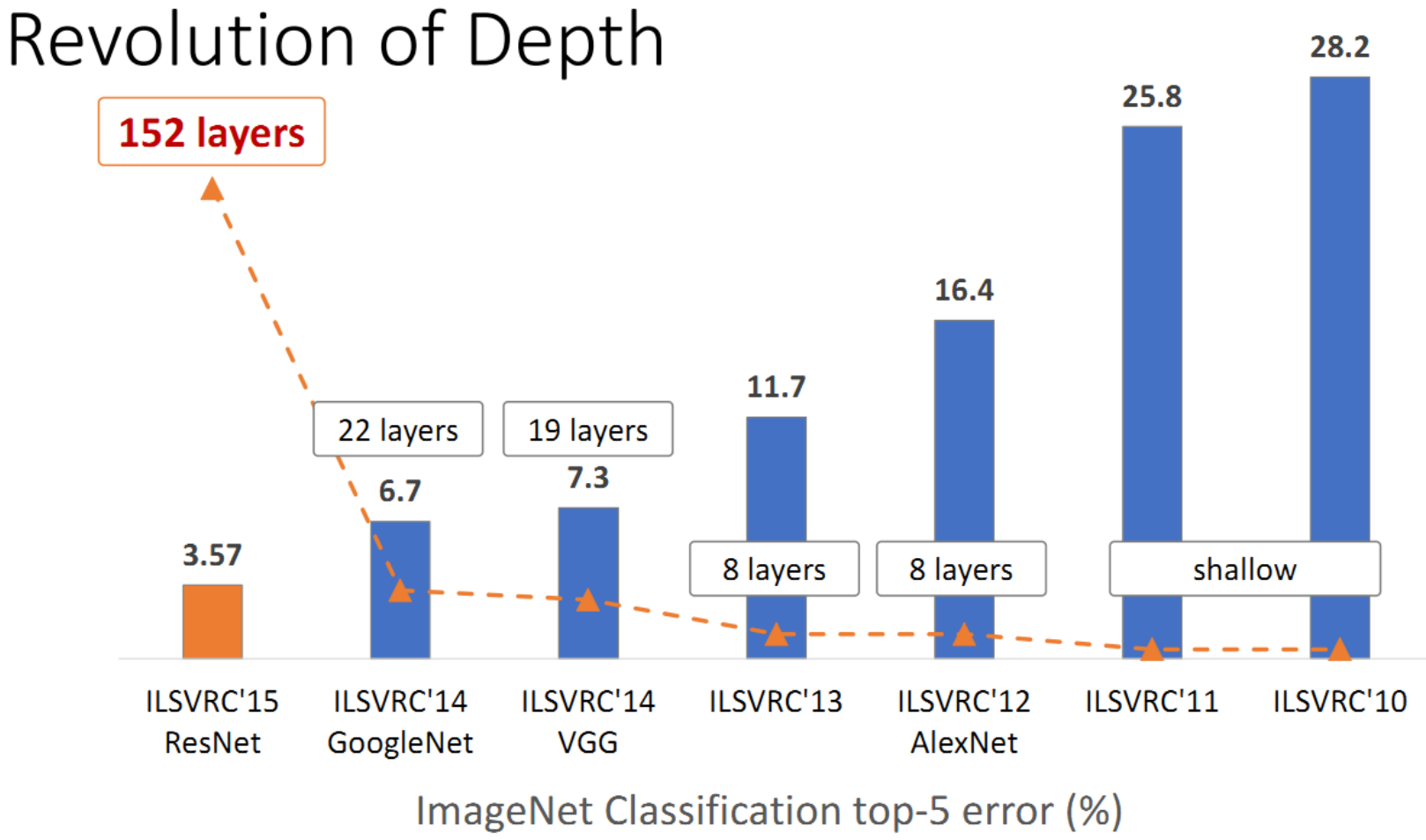
残差网络中，为什么能够避免这种现象？

恒等映射下，虽然网络的深度加深，但是每层中都会有足够多的由梯度承载的信息量，梯度不会太小

加快了深层网络的收敛速度



小结


Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

小结

PaddlePaddle

如何提升一个神经网络的性能？

Aspect 1：　修改网络：　增加深度，增加宽度

减少参数量，防止过拟合，解决梯度消失问题

Aspect 2：　数据集：　尽可能多的数据防止过拟合

- ✓ **AlexNet：**

  Group Convolution(分组卷积)，Dropout，Data Augmentation（数据增强）

- ✓ **VGG：**

  深度，小卷积核

- ✓ **Inception（Google-Net）：**

  同一层使用多个类型的卷积核，Bottleneck，Batch Normalization（批归一化）

- ✓ **ResNet：**

  skip/identity connection 残差的引入

目录

AI DISCOVERY

1 概述

深层神经网络问题导入、
卷积神经网络概念引出

2 自己动手搭CNN

CNN网络结构、CNN网络训练、
如何用Paddle实现CNN

3 经典CNN结构探索

AlexNet，VGG，GoogLeNet
/Inception, ResNet

4 课程实践

实践：猫狗分类

AI DISCOVERY

应用与实践

AI DISCOVERY

实践：猫狗分类