人工智能原理与技术

卷积神经网络

提 纲

- 绪论
  1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络
- 基本组成结构
  1. 卷积 2.池化 3.全连接
- 卷积神经网络典型结构
  1. AlexNet 2. ZFNet 3. VGG 4. GoogleNet 5. ResNet

提 纲

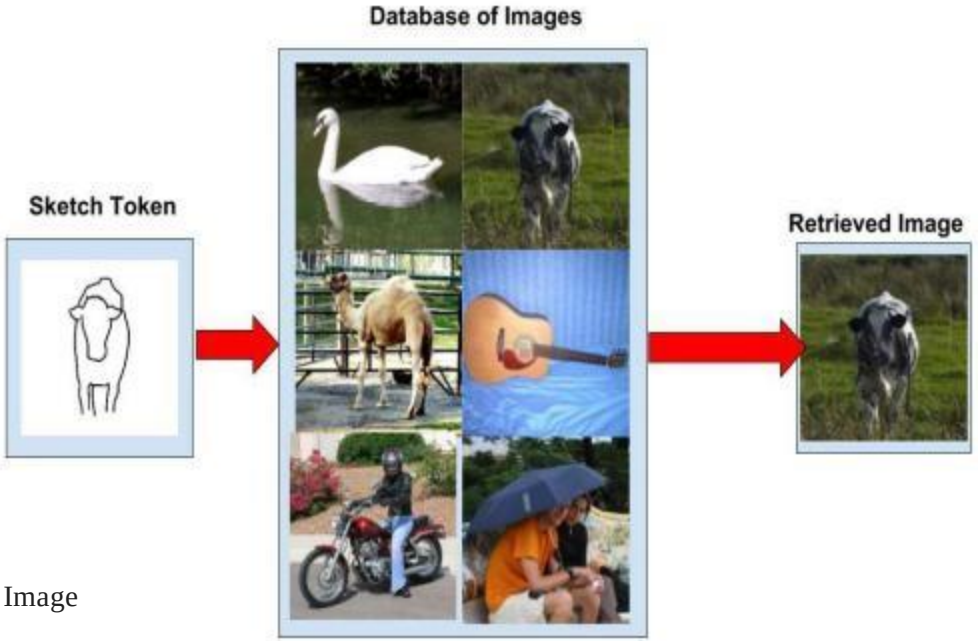
- 绪论
  - 1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络
- 基本组成结构
  - 1. 卷积 2.池化 3.全连接
- 卷积神经网络典型结构
  - 1. AlexNet 2. ZFNet 3. VGG 4. GoogleNet 5. ResNet

无处不在的卷积神经网络 – 基本应用

分类

Gebru, Timnit, Judy Hoffman, and Li Fei-Fei. "Fine-grained recognition in the wild: A multi-task domain adaptation approach." *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017.

检索



Yelamarthi, Sasi Kiran, et al. "A Zero-Shot Framework for Sketch Based Image Retrieval." European Conference on Computer Vision. 2018.

无处不在的卷积神经网络 – 基本应用

检测

https://github.com/arvention/STDN

Zhou, Peng, et al. "Scale-Transferrable Object Detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

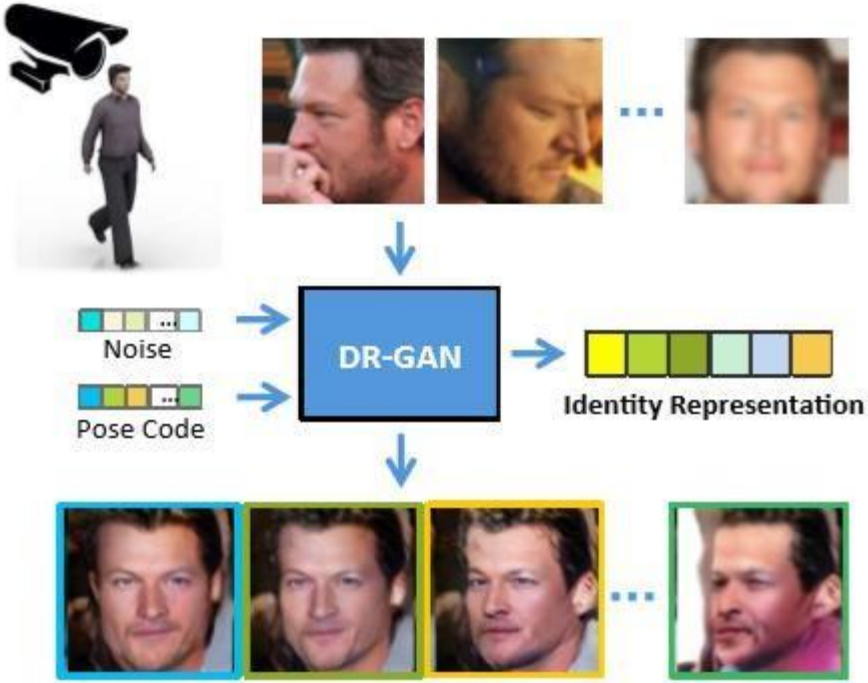
分割

Shen, Tong, et al. "Bootstrapping the Performance of Webly Supervised Semantic Segmentation." CVPR. 2018.

无处不在的卷积神经网络 – 人脸识别

人脸识别

人脸验证（face verification）
这个人脸是XXX吗?

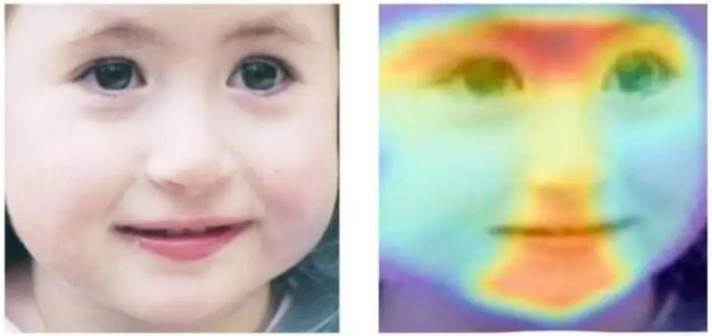


人脸识别（face identification/face recognition）
我是谁?

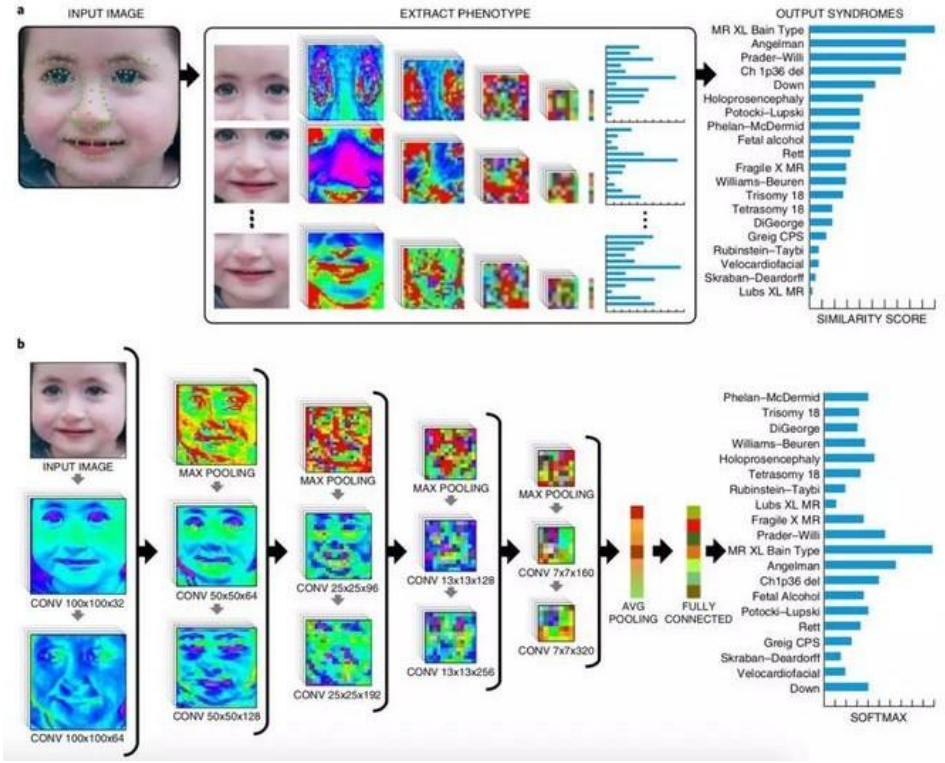
Tran L Q, Yin X, Liu X. Representation learning by rotating your faces.TPAMI, 2018. https://github.com/kayamin/DR-GAN

Tran L, Yin X, Liu X. Disentangled representation learning gan for pose-invariant face recognition. CVPR. 2017, 3(6): 7.

无处不在的卷积神经网络 – 人脸识别



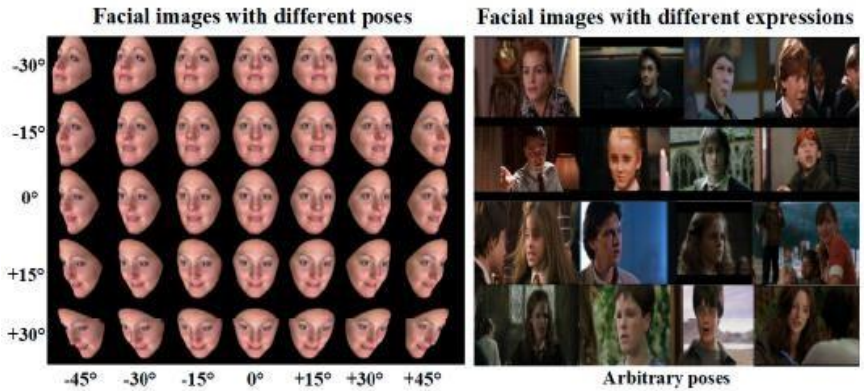
识别遗传病准确率达91%



Yaron Gurovich, Yar Hanani, et al. identifying facial phenotypes of genetic disorders using deep learning. *Nature Medicine.*

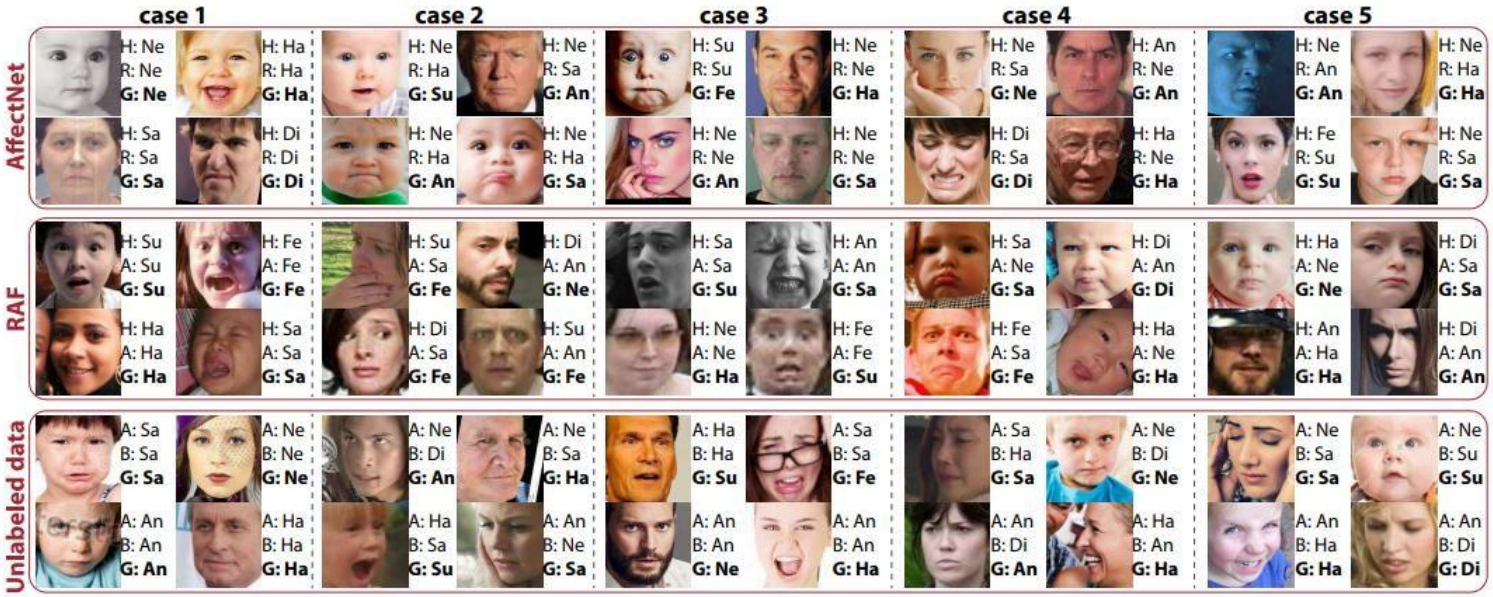
无处不在的卷积神经网络 – 人脸表情识别

人脸表情识别



https://github.com/FFZhang1231/Facial-expression-recognition

Zhang F, Zhang T, Mao Q, et al. Joint Pose and Expression Modeling for Facial Expression Recognition. CVPR. 2018: 3359-3368.



Zeng J, Shan S, Chen X. Facial expression recognition with inconsistently annotated datasets. ECCV. 2018: 222-37.

无处不在的卷积神经网络 – 图像生成

图像生成

Image to Image.

https://github.com/charliememory/Disentangled-Person-Image-Generation

Ma, Liqian, et al. "Disentangled person image generation." CVPR. 2018.

Text to Image.

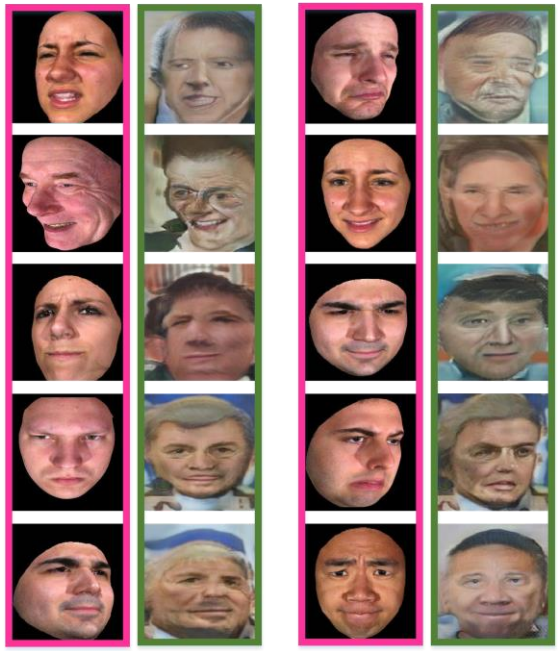
Johnson, Justin and Gupta, Agrim and Fei-Fei, Li. Image Generation From Scene Graphs.CVPR.2018. https://github.com/google/sg2im

无处不在的卷积神经网络 – 图像风格转化

图像风格转化

Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." CVPR. 2017.

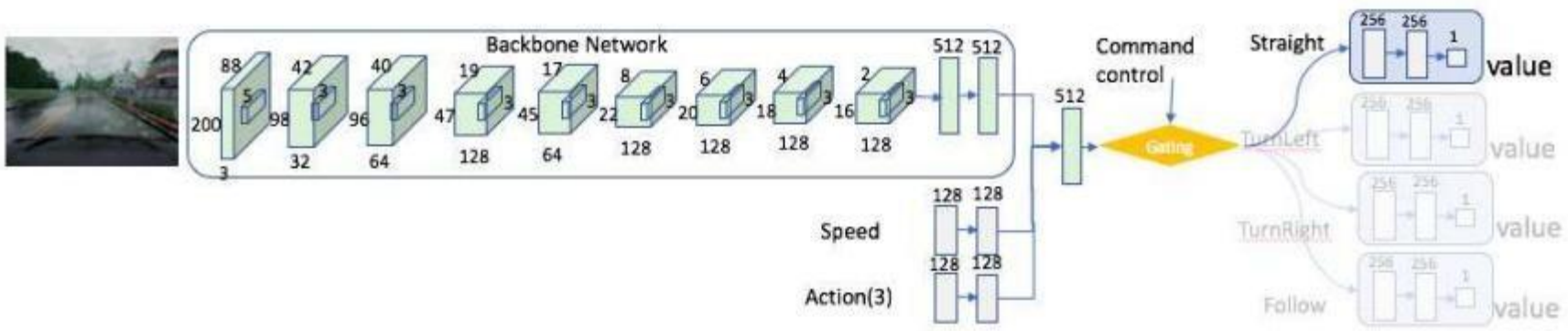
https://github.com/phillipi/pix2pix



Zhang, Feifei, et al. "Facial Expression Recognition in the Wild: A Cycle-Consistent Adversarial Attention Transfer Approach." 2018 ACM Multimedia Conference on Multimedia Conference. ACM, 2018.

无处不在的卷积神经网络 – 自动驾驶

自动驾驶



Liang X, Wang T, Yang L, et al. CIRL: Controllable imitative reinforcement learning for vision-based self-driving. ECCV, 2018, 1.

提 纲

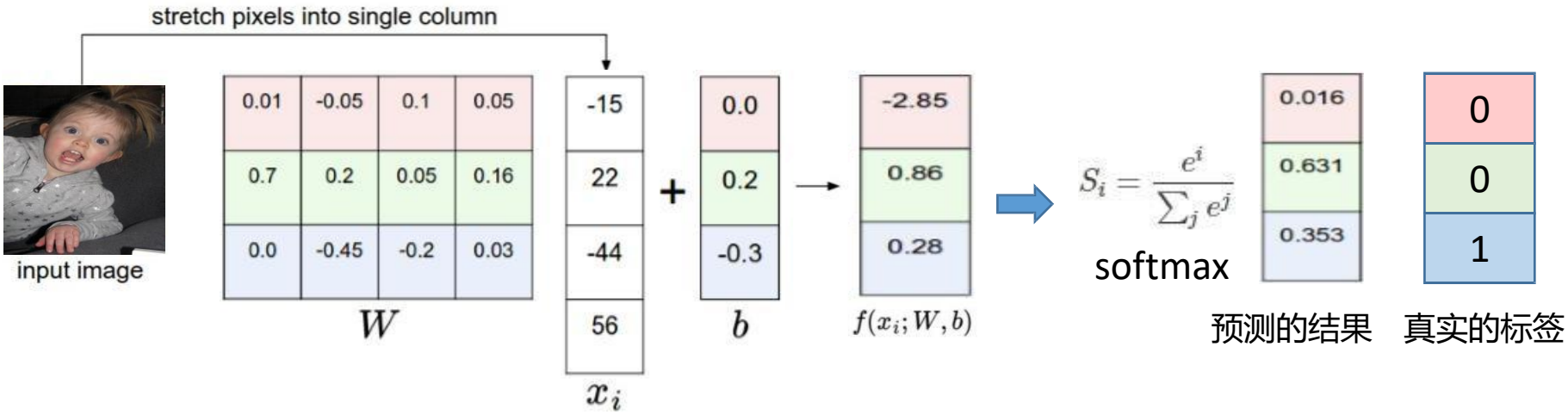
- 绪论
  - 1. 卷积神经网络的应用 **2. 传统神经网络vs卷积神经网络**
- 基本组成结构
  - 1. 卷积 2.池化 3.全连接
- 卷积神经网络典型结构
  - 1. AlexNet 2. ZFNet 3. VGG 4. GoogleNet 5. ResNet
- 代码实战
  - Tensorflow-CNN
- 总结
  - 1. 参考文献 2. 代码 3. 作业

传统神经网络vs卷积神经网络

深度学习三部曲

- **Step 1. 搭建神经网络结构**
- **Step 2. 找到一个合适的损失函数**
  - ✓ 交叉熵损失（cross entropy loss），均方误差（MSE），……
- **Step 3. 找到一个合适的优化函数，更新参数**
  - ✓ 反向传播（BP），随机梯度下降（SGD），……

损失函数

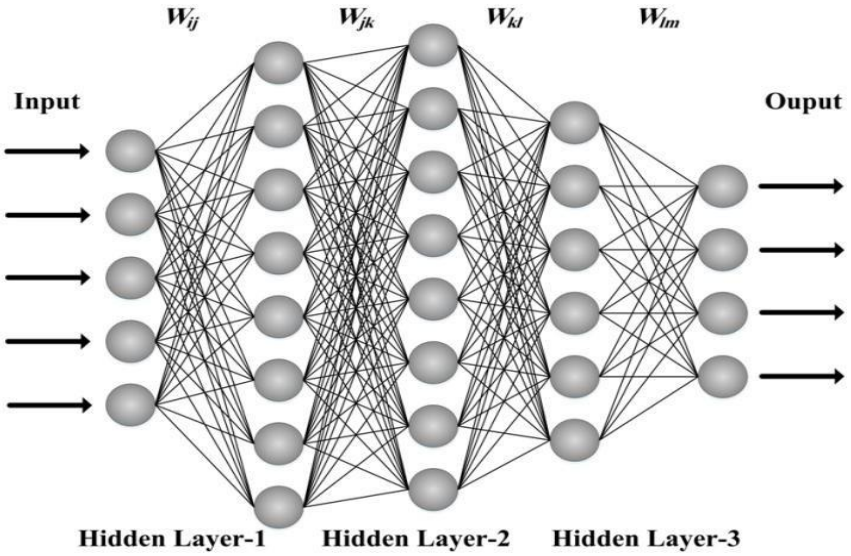


- 给定W，可以由像素映射到类目得分
- 损失函数是用来衡量吻合度的
- 可以调整参数/权重W，使得映射的结果和实际类别吻合
- 常用分类损失：

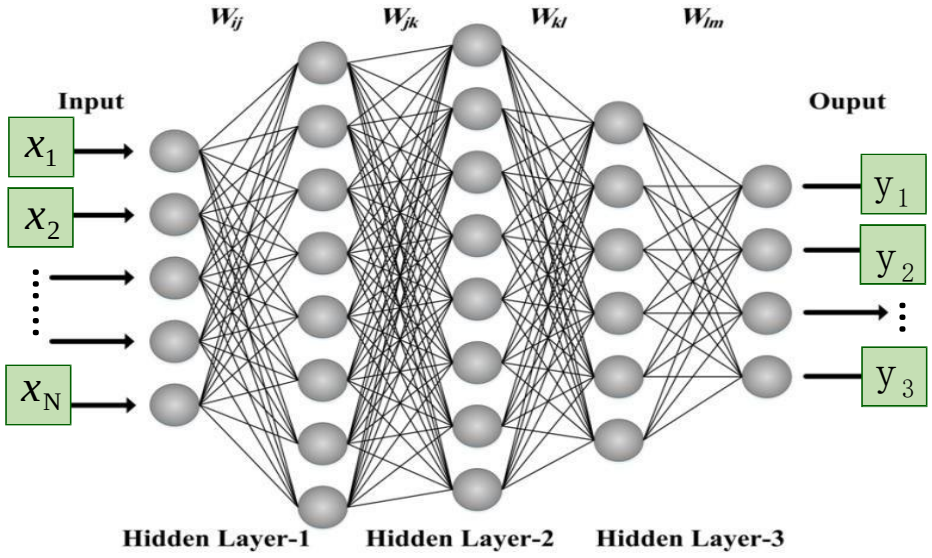
  交叉熵损失：$Loss = -\sum_i y_i \ln y_i^p$　　hinge loss：$L(y, f(x)) = max(0, 1 - yf(x))$　　.....
- 常用回归损失：

  均方误差：$MSE = \sum_{i=1}^{n} (y_i - y_i^p)^2$　　平均绝对值误差(L1损失)：$MAE = \sum_{i=1}^{n} |y_i - y_i^p|$　　.....

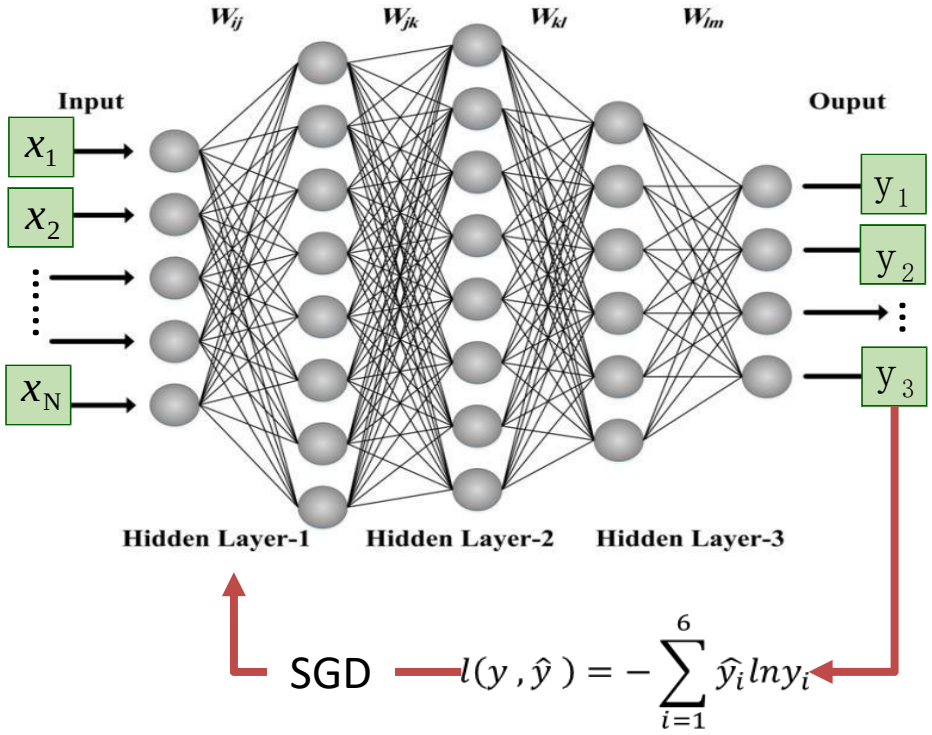
传统神经网络vs卷积神经网络



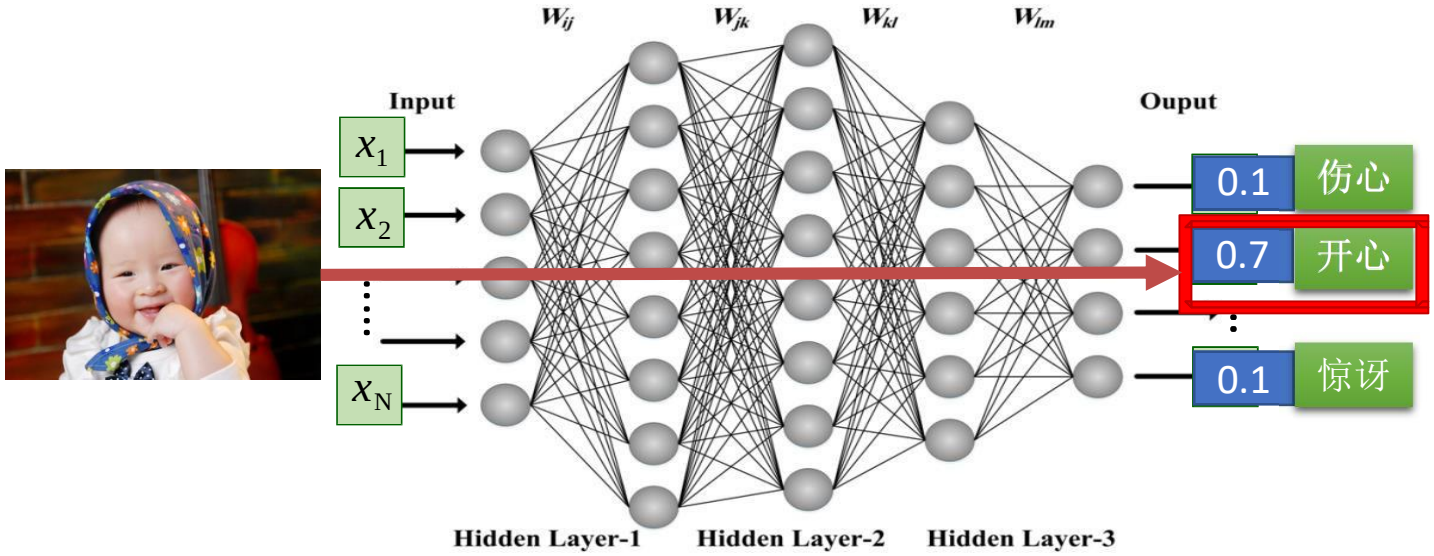
传统神经网络vs卷积神经网络



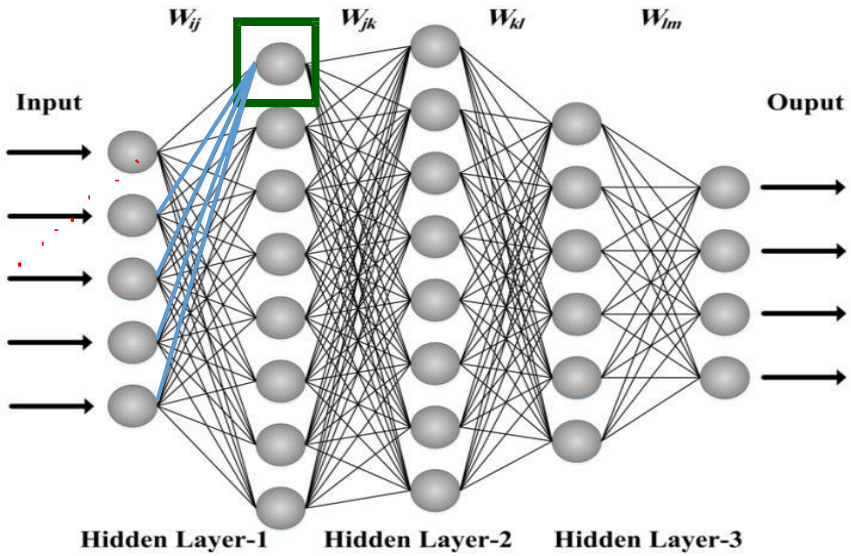
传统神经网络vs卷积神经网络



传统神经网络vs卷积神经网络



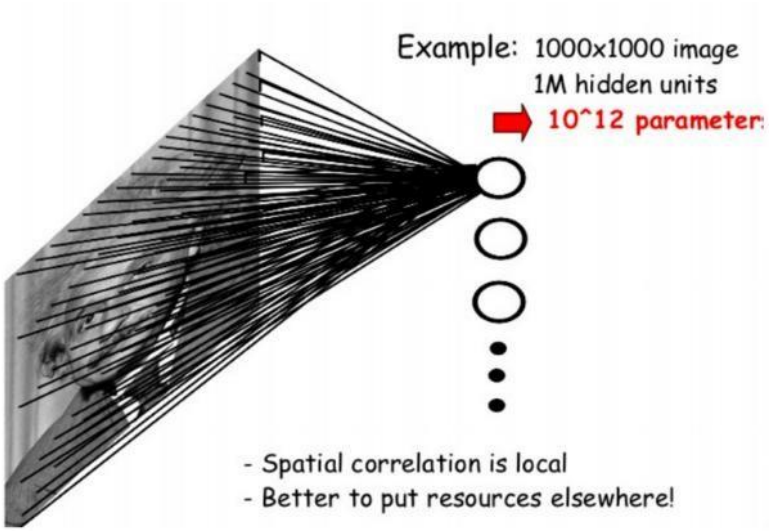
传统神经网络vs卷积神经网络



传统神经网络能用到计算机视觉上吗?

为什么还需要卷积神经网络呢?

传统神经网络vs卷积神经网络

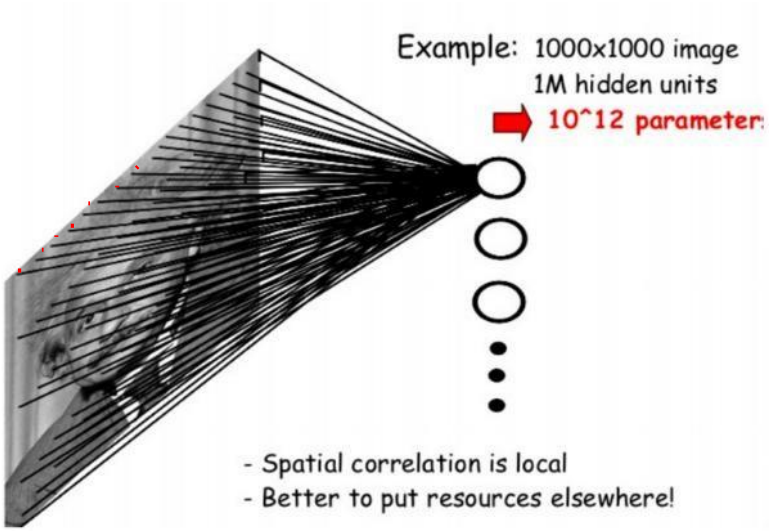


传统神经网络能用到计算机视觉上吗？

为什么还需要卷积神经网络呢？

- **全连接网络处理图像的问题**
  - **参数太多**：权重矩阵的参数太多 –> 过拟合

传统神经网络vs卷积神经网络



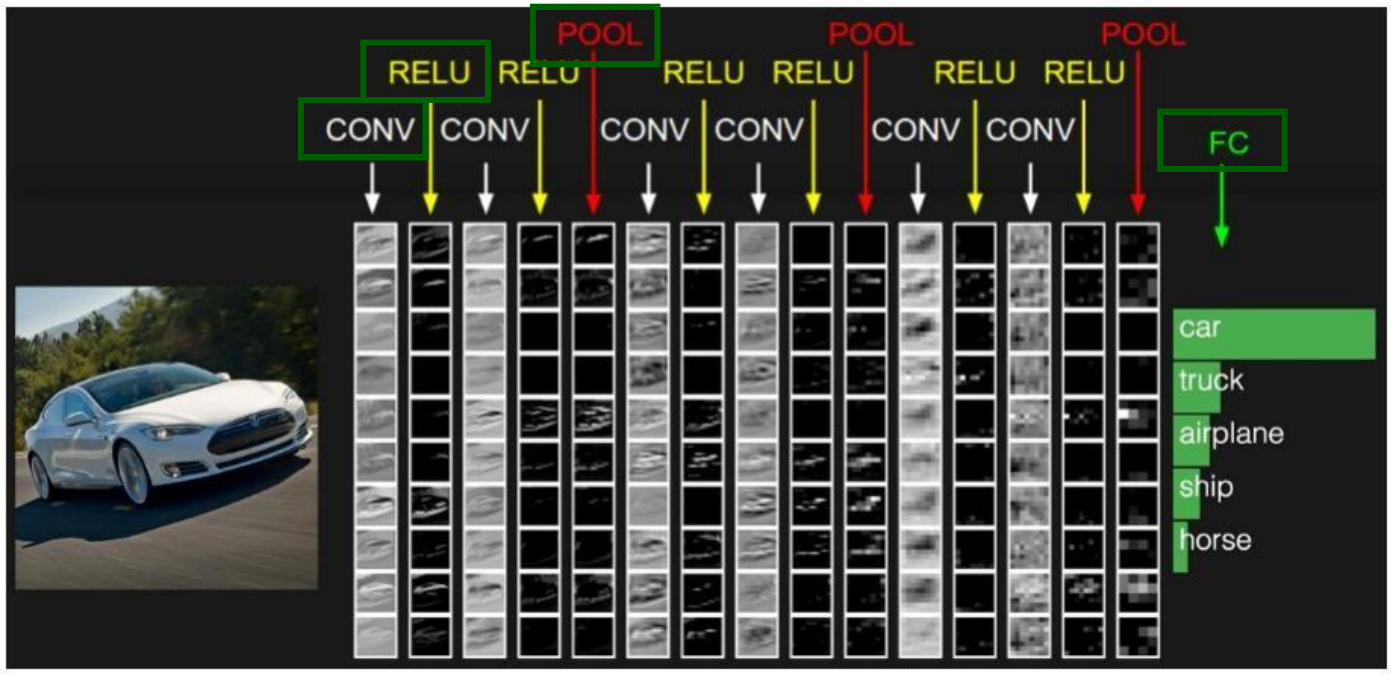
传统神经网络能用到计算机视觉上吗?

为什么还需要卷积神经网络呢?

- **全连接网络处理图像的问题**
  - **参数太多**：权重矩阵的参数太多 –> 过拟合
- **卷积神经网络的解决方式**
  - 局部关联，参数共享

传统神经网络vs卷积神经网络

- **相同之处**



- ✓ CONV layer: 卷积层
- ✓ RELU layer: RELU 激活层
- ✓ POOL layer: 池化层
- ✓ FC layer：全连接层

提纲

- 绪论
  1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络
- 基本组成结构
  1. 卷积 2.池化 3.全连接
- 卷积神经网络典型结构
  1. AlexNet 2. ZFNet 3. VGG 4. GoogleNet 5. ResNet

卷积 — Convolutional Layer

- 一维卷积
  - 一维卷积经常用在信号处理中，用于计算信号的延迟累积
  - 假设一个信号发生器在时刻t发出一个信号x_t，其信息的衰减率为f_k,即在k−1个时间步长后，信息衰减为原来的f_k倍
  - 设$f_1 = 1, f_2 = \frac{1}{2}, f_3 = 1/4$，在时刻t收到的信号$y_t$为当前时刻产生的信息和以前时刻延迟信息的叠加

$$\begin{aligned} y_t &= 1 \times x_t + 1/2 \times x_{t-1} + 1/4 \times x_{t-2} \\ &= f_1 \times x_t + f_2 \times x_{t-1} + f_3 \times x_{t-2} \\ &= \sum_{k=1}^{3} w_k \cdot x_{t-k+1}. \end{aligned}$$

  - 此处的f=[f_1 f_2 f_3]被称为滤波器（filter）或卷积核（convolutional kernel
  - 设滤波器f长度为m，它和一个信号序列x=[x_1,x_2,x_3,...]的卷积记为

$$y_t = \sum_{k=1}^{m} f_k \cdot x_{t-k+1},$$

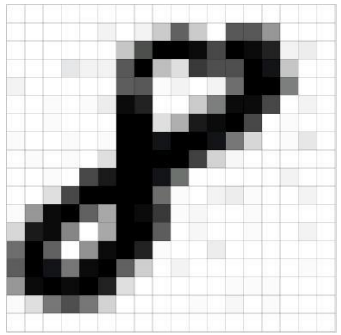
卷积 — Convolutional Layer

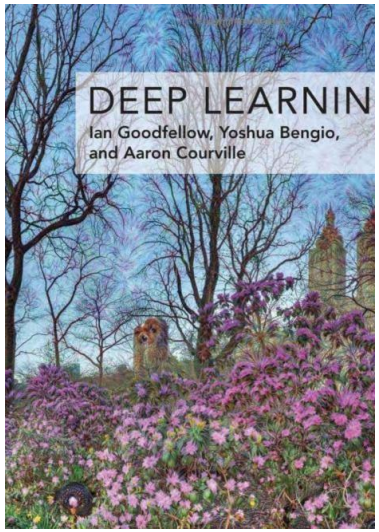
卷积是什么?

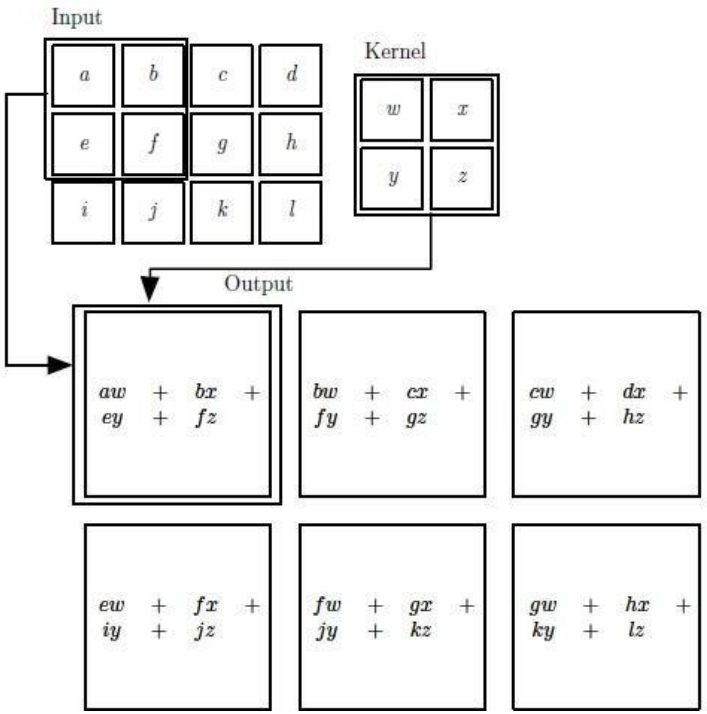
- ✓ convolution is an operation on two functions of a real-valued argument.

  卷积是对两个实变函数的一种数学操作。

- ✓ 在图像处理中，图像是以二维矩阵的形式输入到神经网络的，因此我们需要二维卷积。





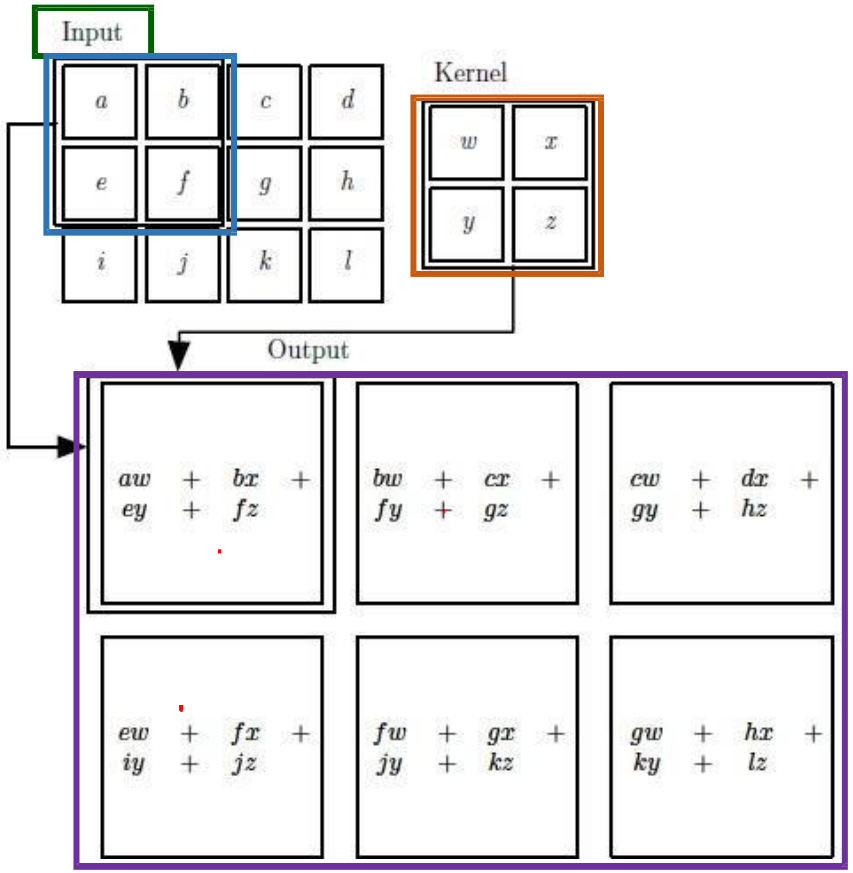


给定一个图像 $X \in \mathbb{R}^{M \times N}$，和滤波器 $W \in \mathbb{R}^{m \times n}$

$$y = WX + b$$

卷积 — Convolutional Layer

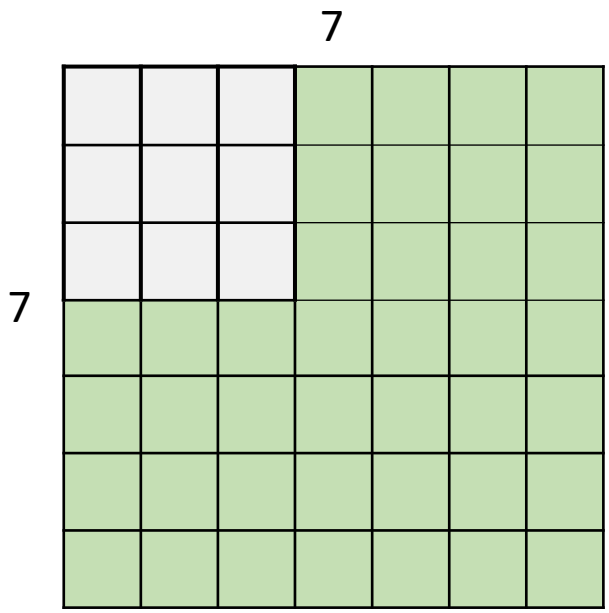
- **涉及到的基本概念?**



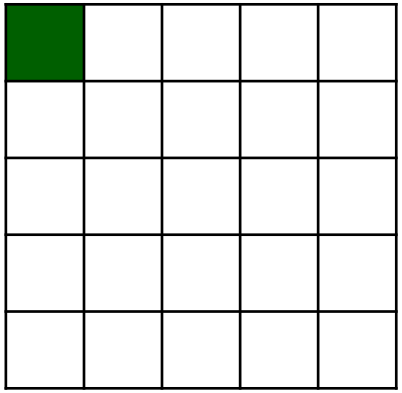
- ✓ input: 输入
- ✓ kernel/filter: 卷积核/滤波器
- ✓ weights:权重
- ✓ receptive field:感受野
- ✓ feature map: 特征图
- ✓ padding
- ✓ channel: 深度
- ✓ output: 输出

卷积 — Convolutional Layer

- **更具体的看一看……**

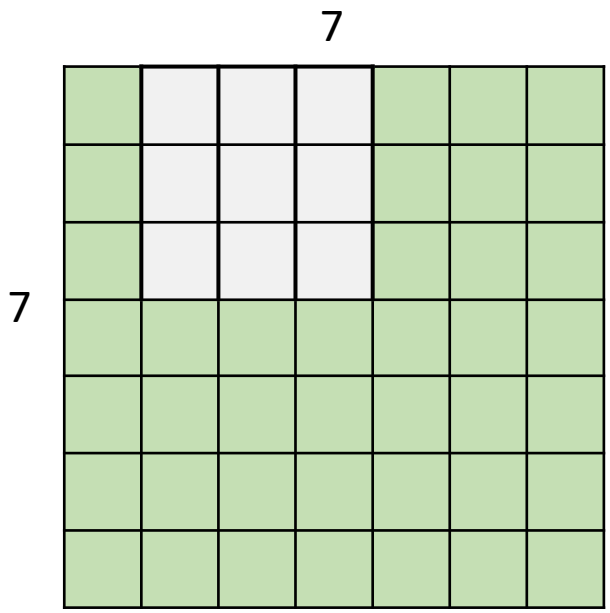


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：1

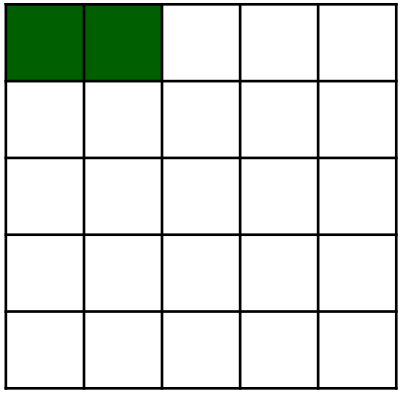


卷积 — Convolutional Layer

- **更具体的看一看……**

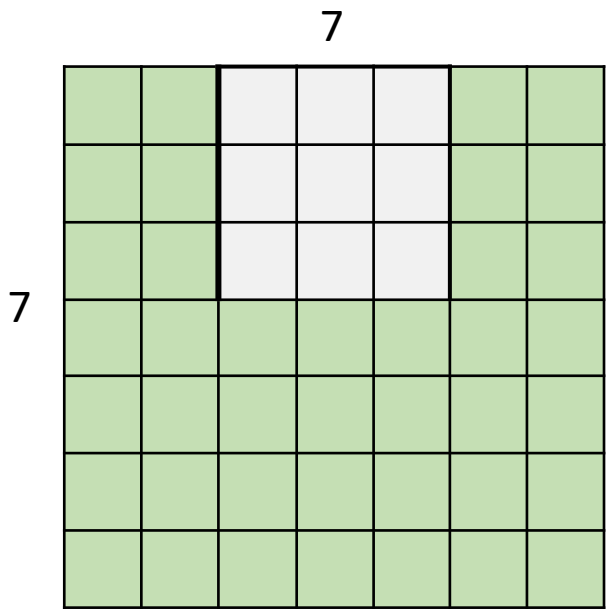


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：1

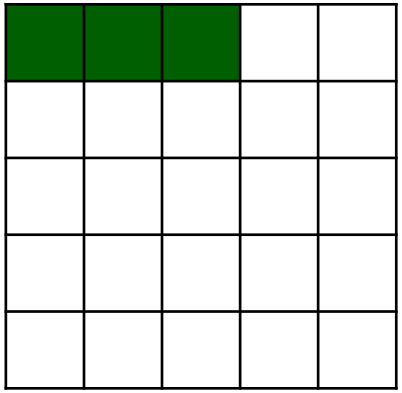


卷积 — Convolutional Layer

- **更具体的看一看……**

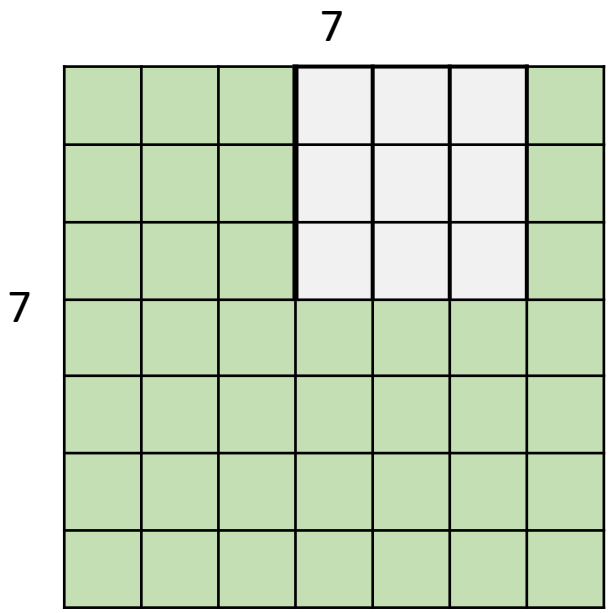


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：1

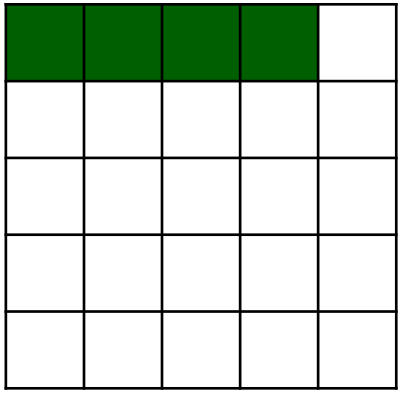


卷积 — Convolutional Layer

- **更具体的看一看……**

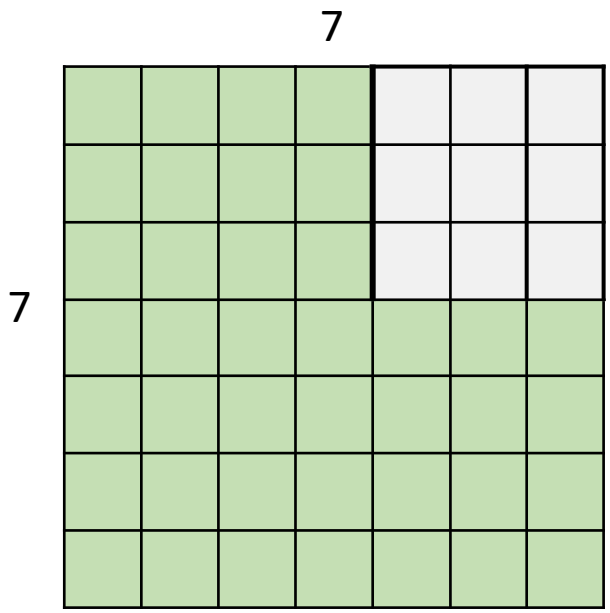


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：1

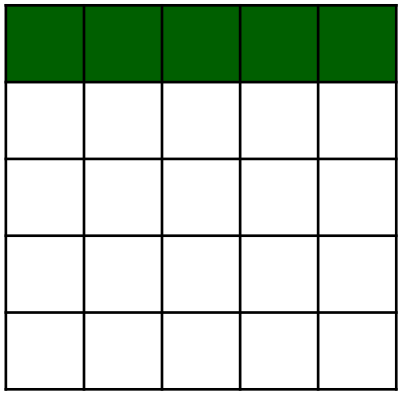


卷积 — Convolutional Layer

- **更具体的看一看……**

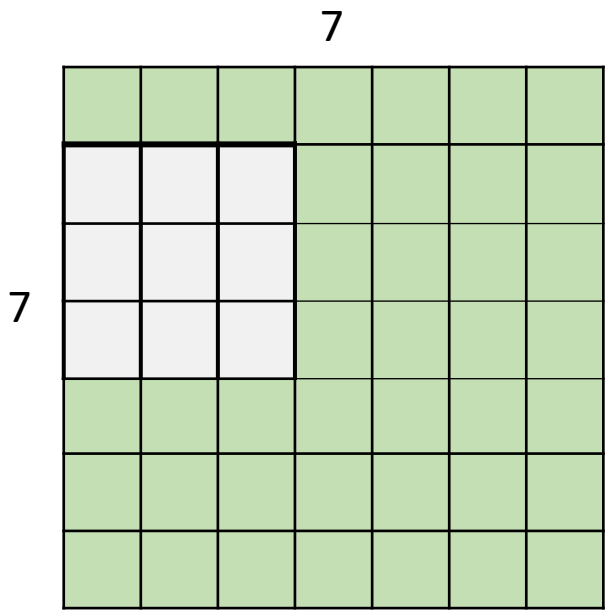


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：1

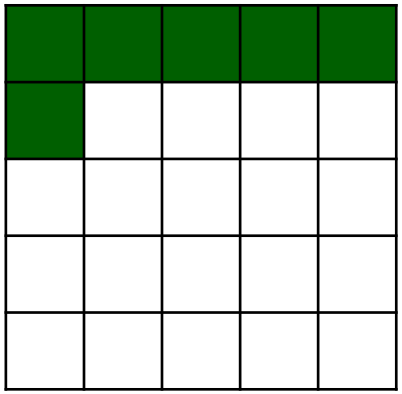


卷积 — Convolutional Layer

- **更具体的看一看……**

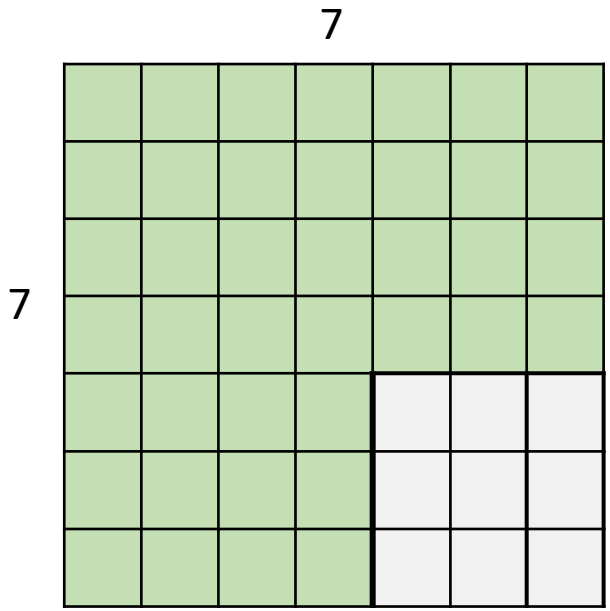


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：1

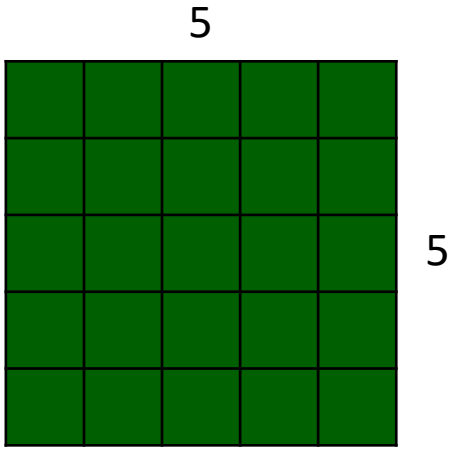


卷积 — Convolutional Layer

- **更具体的看一看……**



- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：1
- ✓ feature map： 5x5



卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

1*1+1*0+1*1+0*0+1*1
+1*0+0*1+0*0+1*1 =4

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4		

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2		

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2		

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2	4	

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2	4	

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2	4	3

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2	4	3

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2	4	3
2		

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2	4	3
2	3	

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2	4	3
2	3	

特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

- **更具体的看一看……**

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

输入

4	3	4
2	4	3
2	3	4

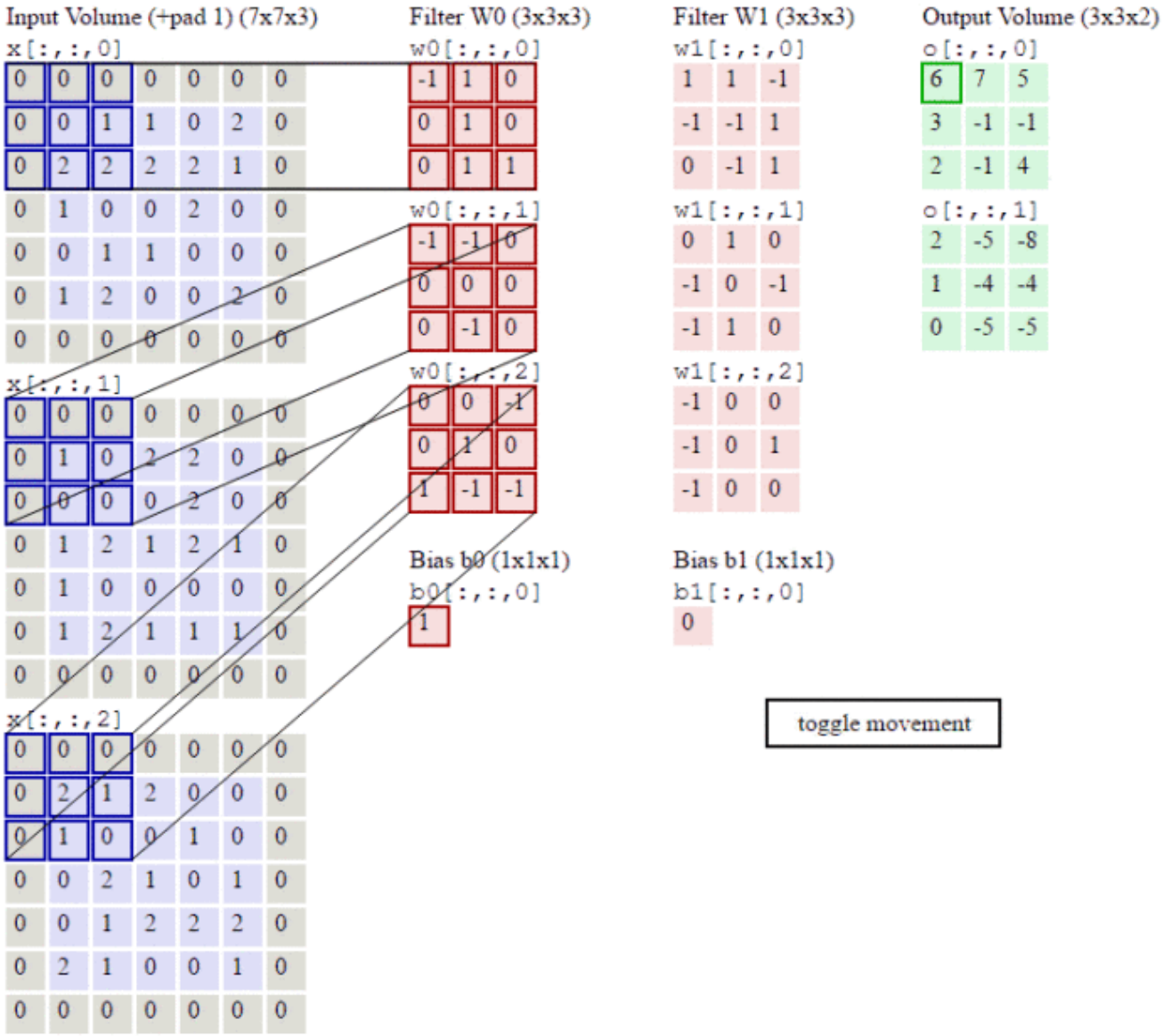
特征图

- ✓ 输入：5x5
- ✓ 卷积核：3x3

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

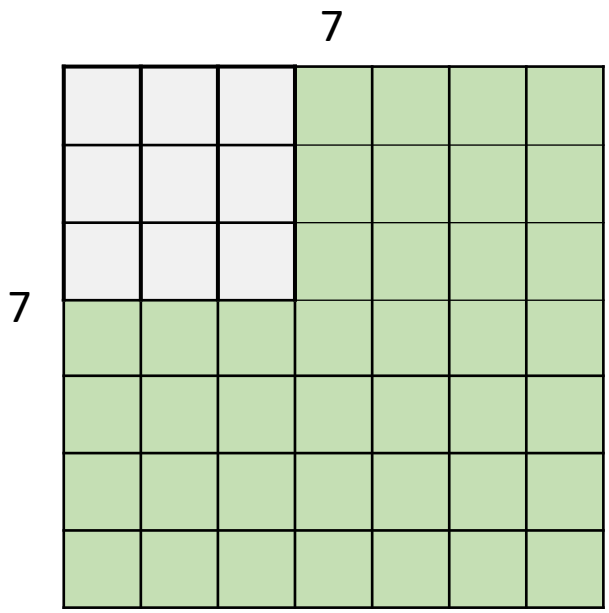
- ✓ 步长 stride：1
- ✓ 特征图： 3x3

卷积 — Convolutional Layer

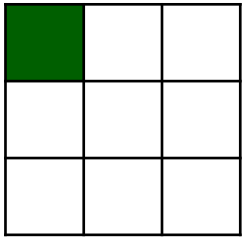


卷积 — Convolutional Layer

- **更具体的看一看……**

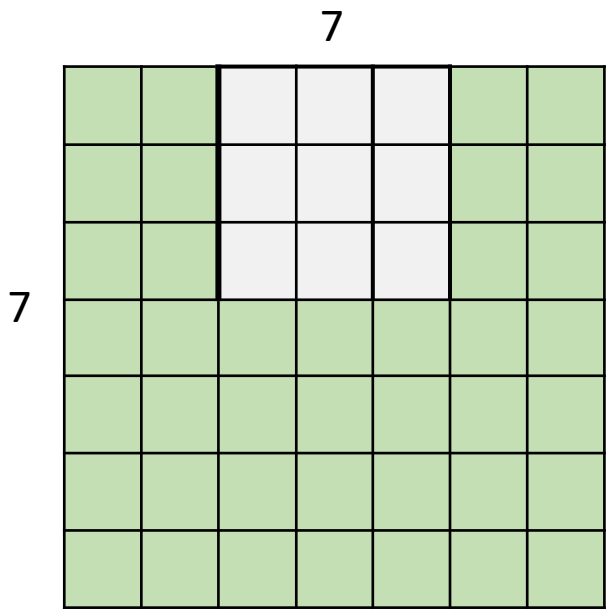


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ stride：2

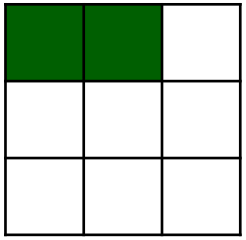


卷积 — Convolutional Layer

- **更具体的看一看……**

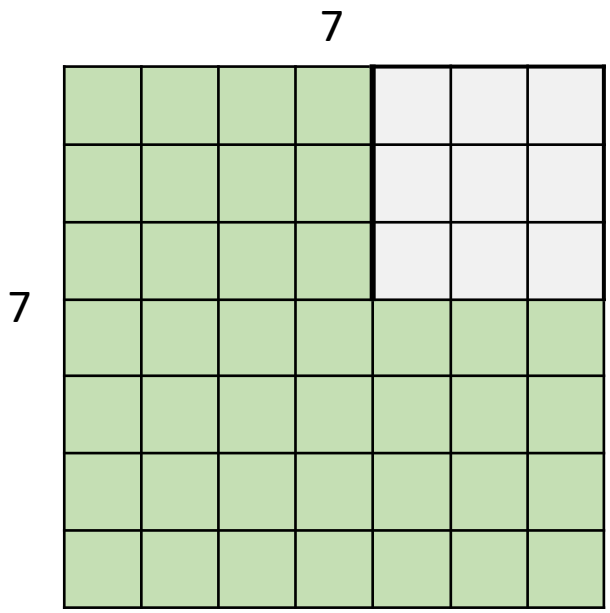


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ stride：2

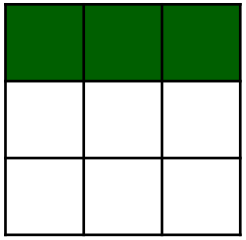


卷积 — Convolutional Layer

- **更具体的看一看……**

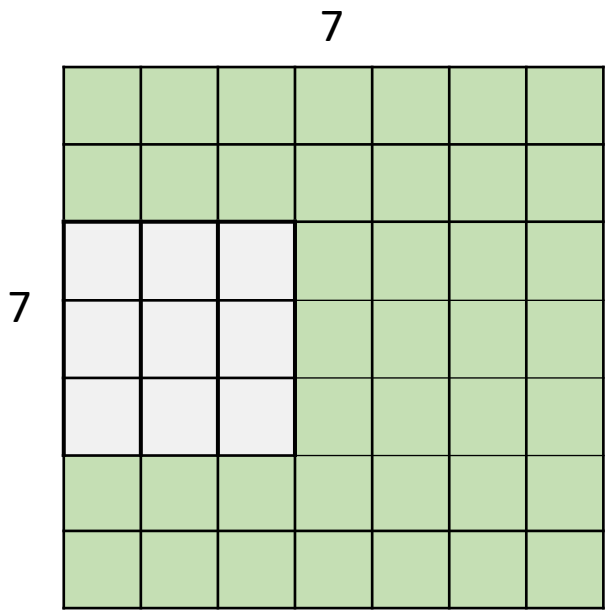


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ stride：2

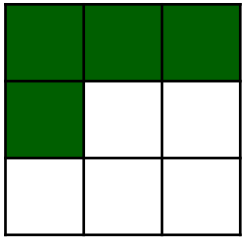


卷积 — Convolutional Layer

- **更具体的看一看……**

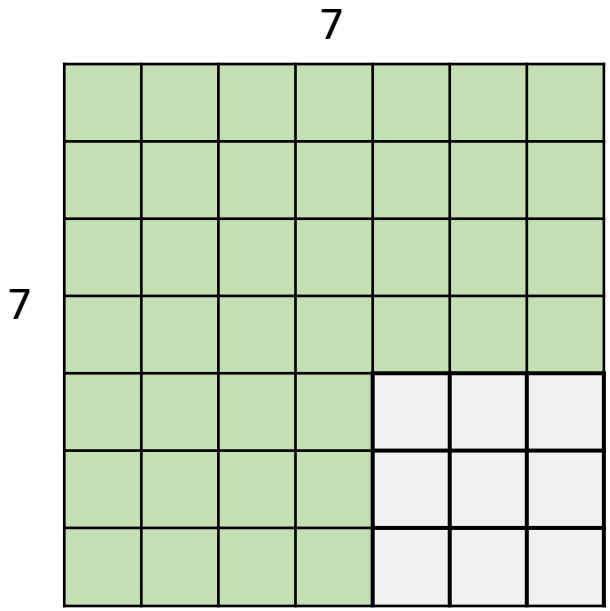


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ stride：2

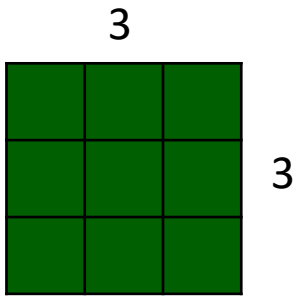


卷积 — Convolutional Layer

- **更具体的看一看……**

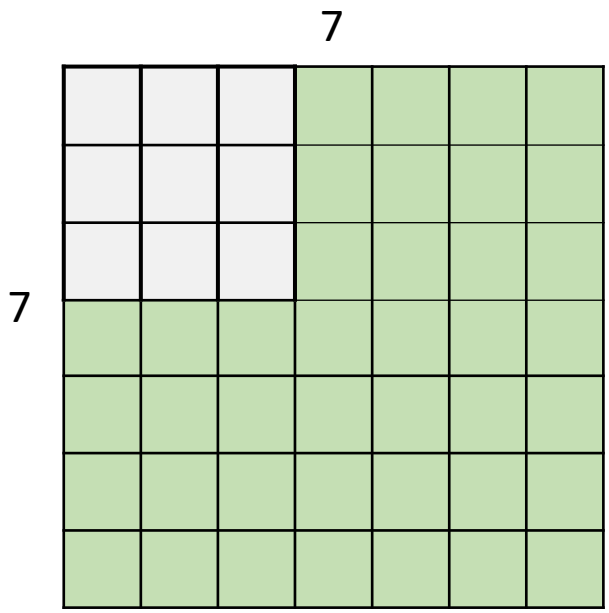


- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ stride：2
- ✓ feature map： 3x3



卷积 — Convolutional Layer

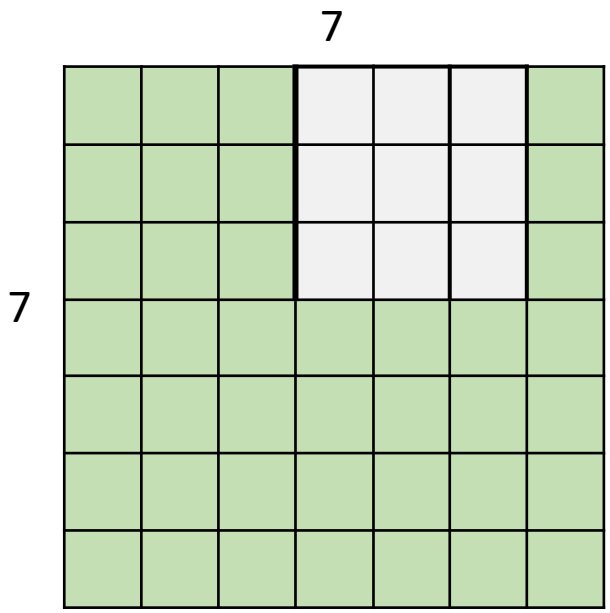
- **更具体的看一看……**



- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：3

卷积 — Convolutional Layer

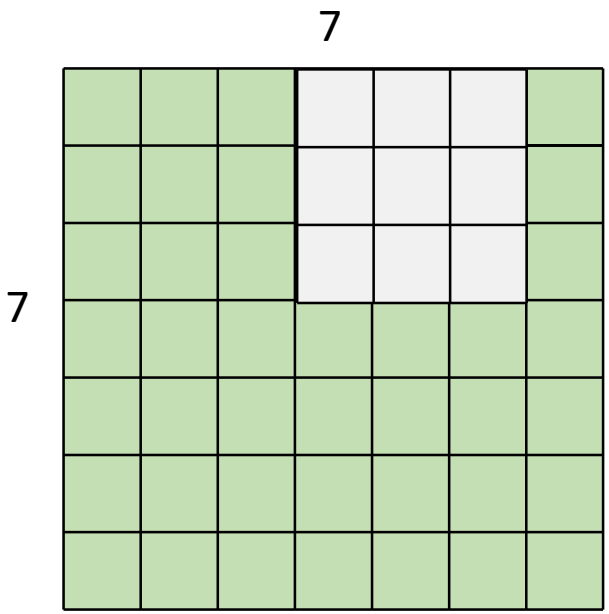
- **更具体的看一看……**



- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：3

卷积 — Convolutional Layer

- **更具体的看一看……**



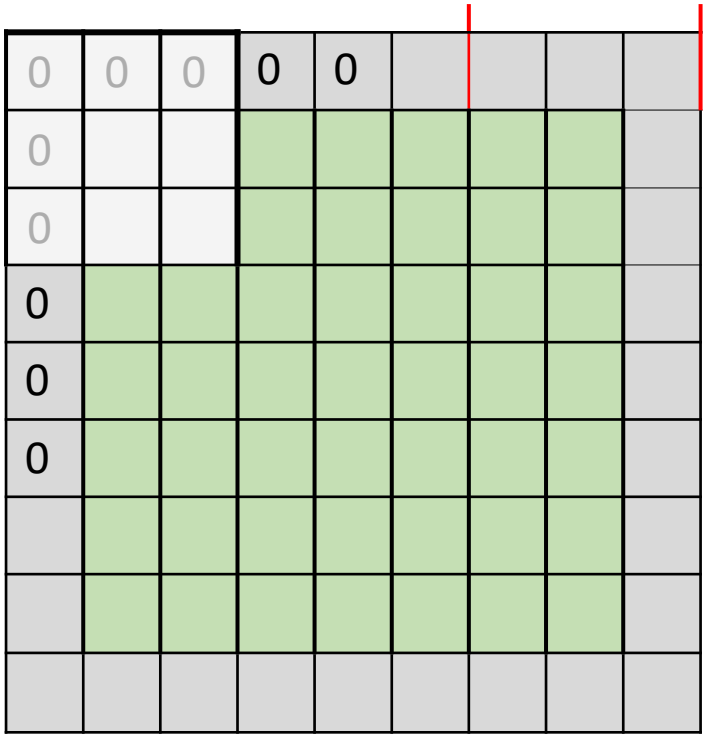
- ✓ 输入：7*7
- ✓ filter：3*3
- ✓ 步长 stride：3

大小不匹配！

卷积 — Convolutional Layer

- **更具体的看一看……**

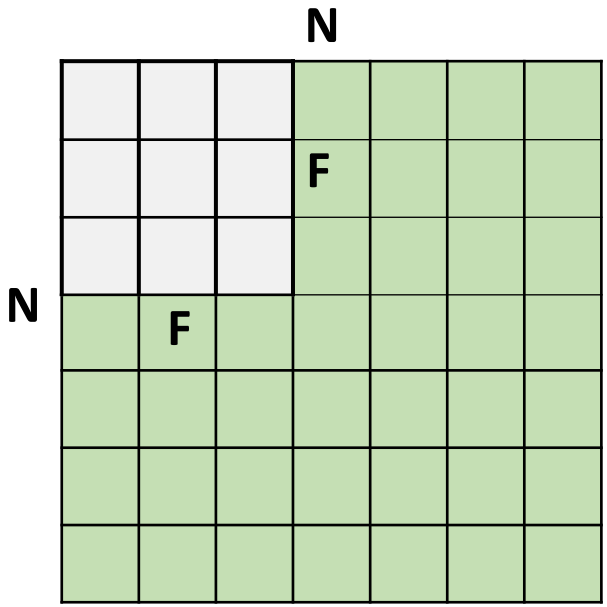
零填充



- ✓ 输入：7x7
- ✓ filter：3x3
- ✓ 步长 stride：3
- ✓ padding: 1
- ✓ feature map: 3x3

卷积 — Convolutional Layer

- 更具体的看一看……



输出的特征图大小：

(N-F)/stride+1 (未加padding)

- ✓ 例 如： N=7， F=3

  stride = 1, (7-3)/1+1=5

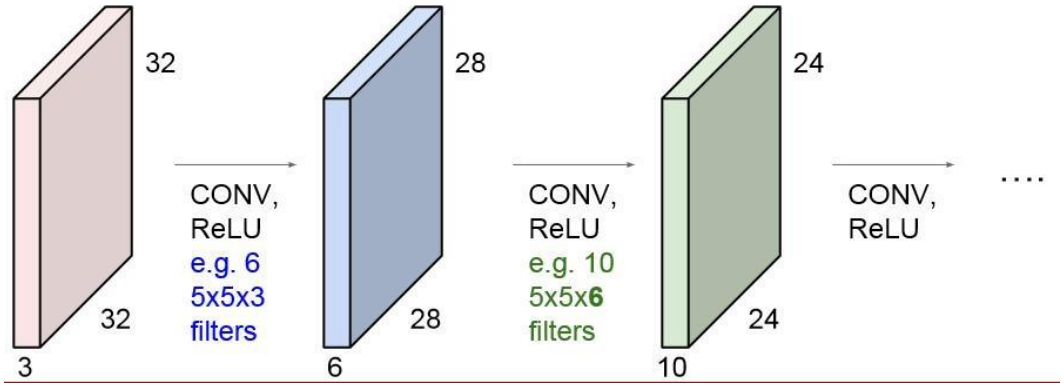
  stride = 2, (7-3)/2+1=3

  stride = 3, (7-3)/3+1=2.3333….

有padding时输出的特征图大小：

(N+padding*2-F)/stride+1

卷积 — Convolutional Layer

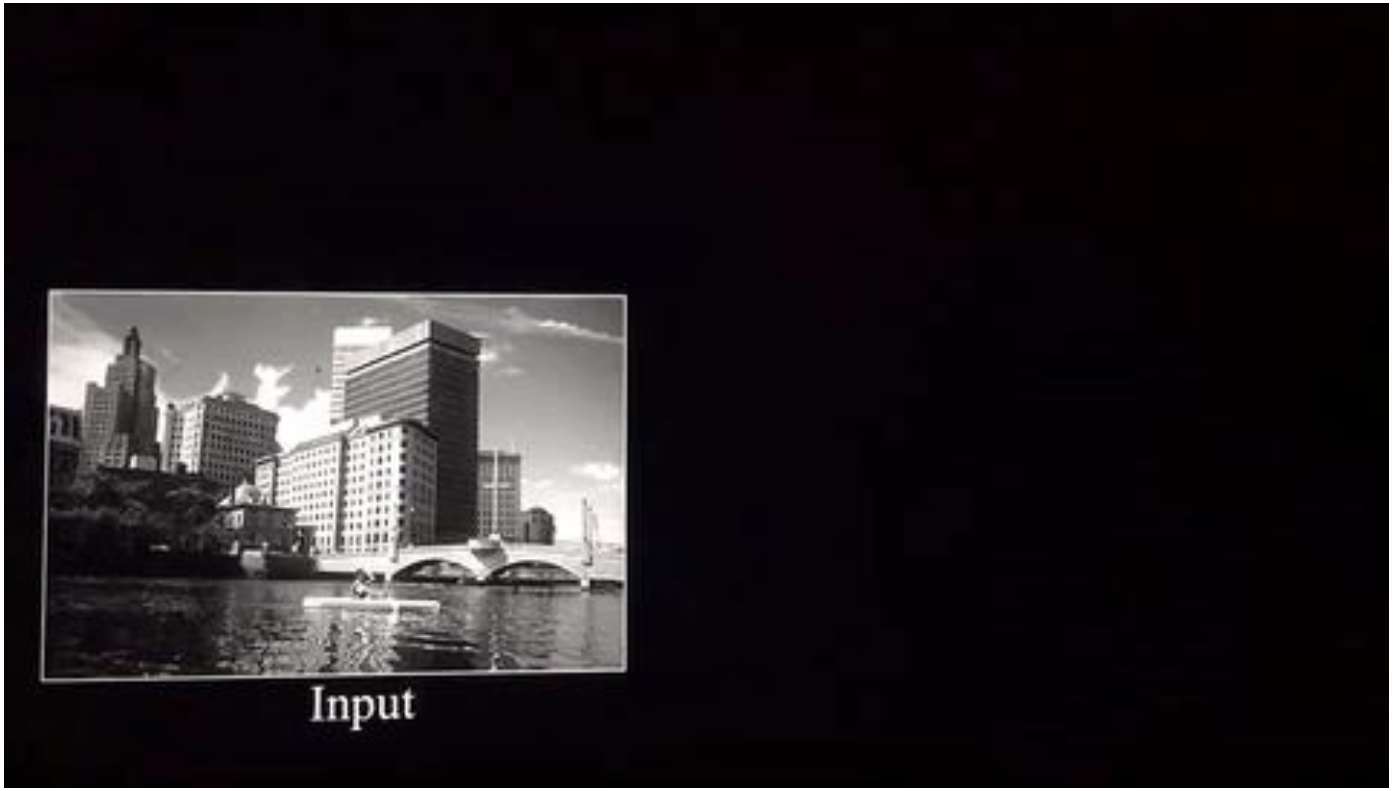


- ✓ 深度：depth/channel

- 思考一个问题：
  - ✓ Input: 32x32
  - ✓ filter：10, 5x5
  - ✓ 步长 stride：1
  - ✓ Padding: 2
  - ✓ 输出： **32x32x10（(32+2x2-5)/1+1=32)**
  - ✓ 参数量： **(5*5+1)*10=260**

卷积 — Convolutional Layer

特征图的产生过程：

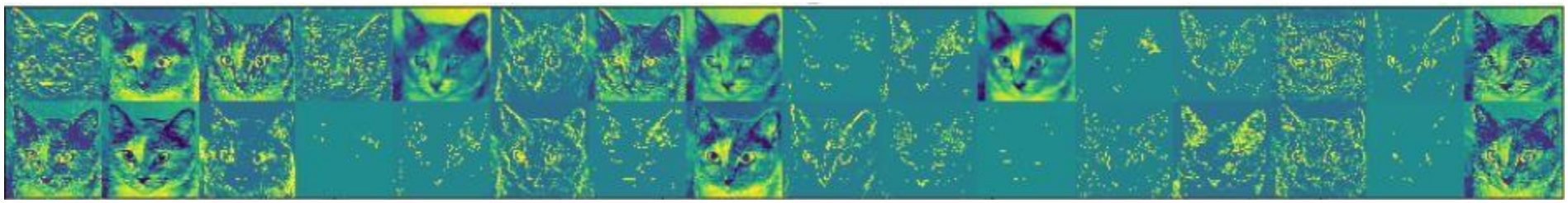


卷积的可视化理解

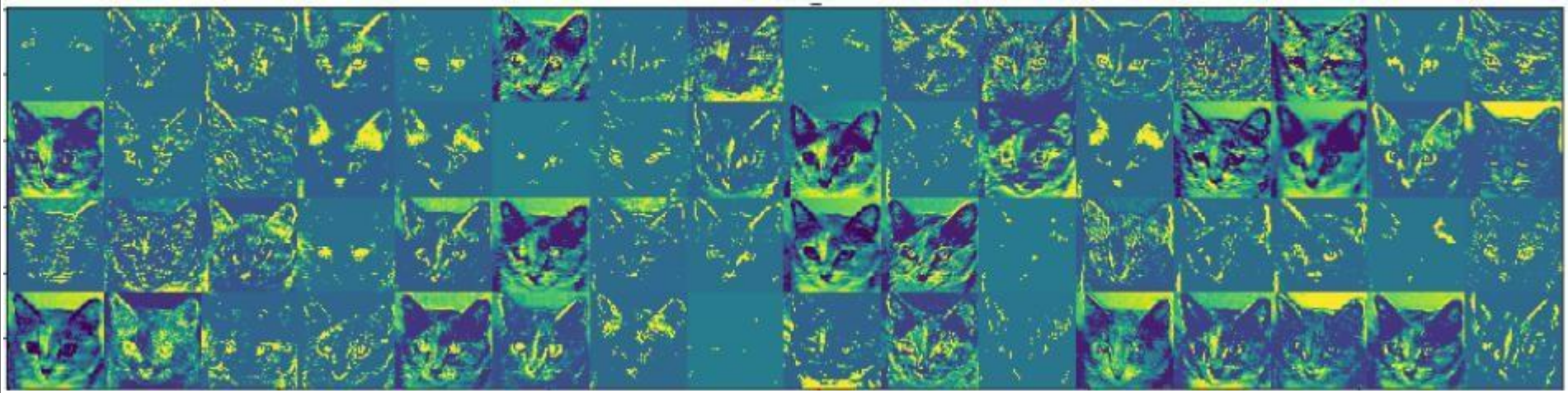
cats_and_dogs_small_2.h5

深度学习是一个黑盒子，我们看看到底每层在学什么

Layer5



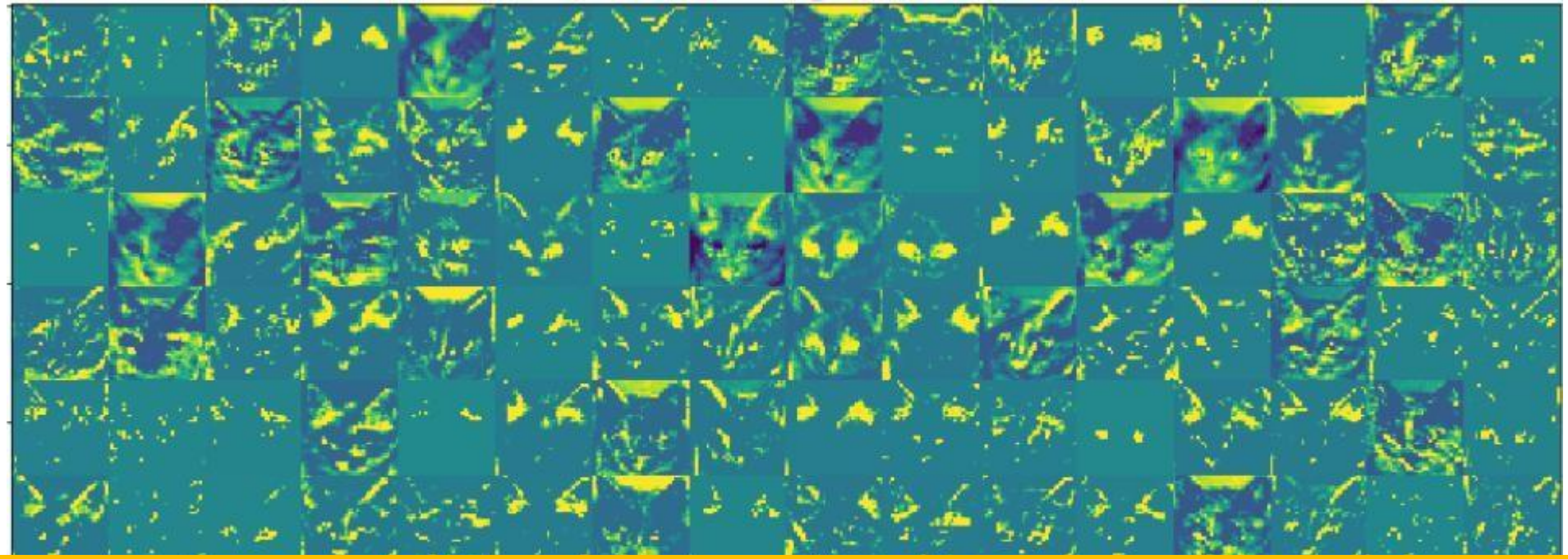
Layer6



卷积的可视化理解

深度学习是一个黑盒子，我们看看到底每层在学什么

Layer7



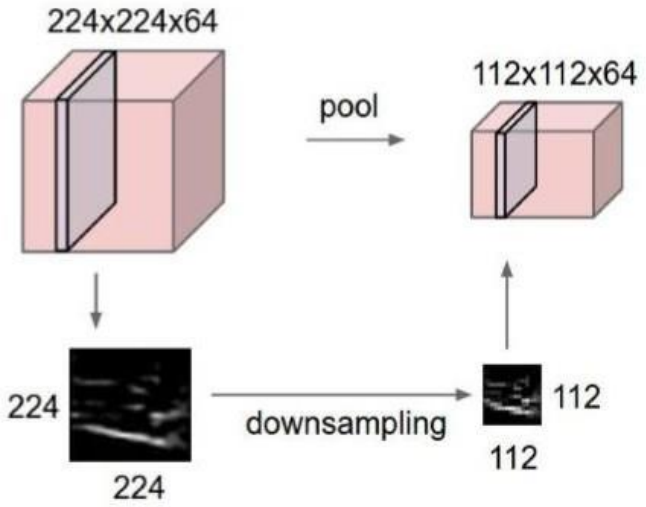
低层的网络多在关注细节，高层的多在关注语义概念

同时，不同的卷积核关注的东西也不太一样

提纲

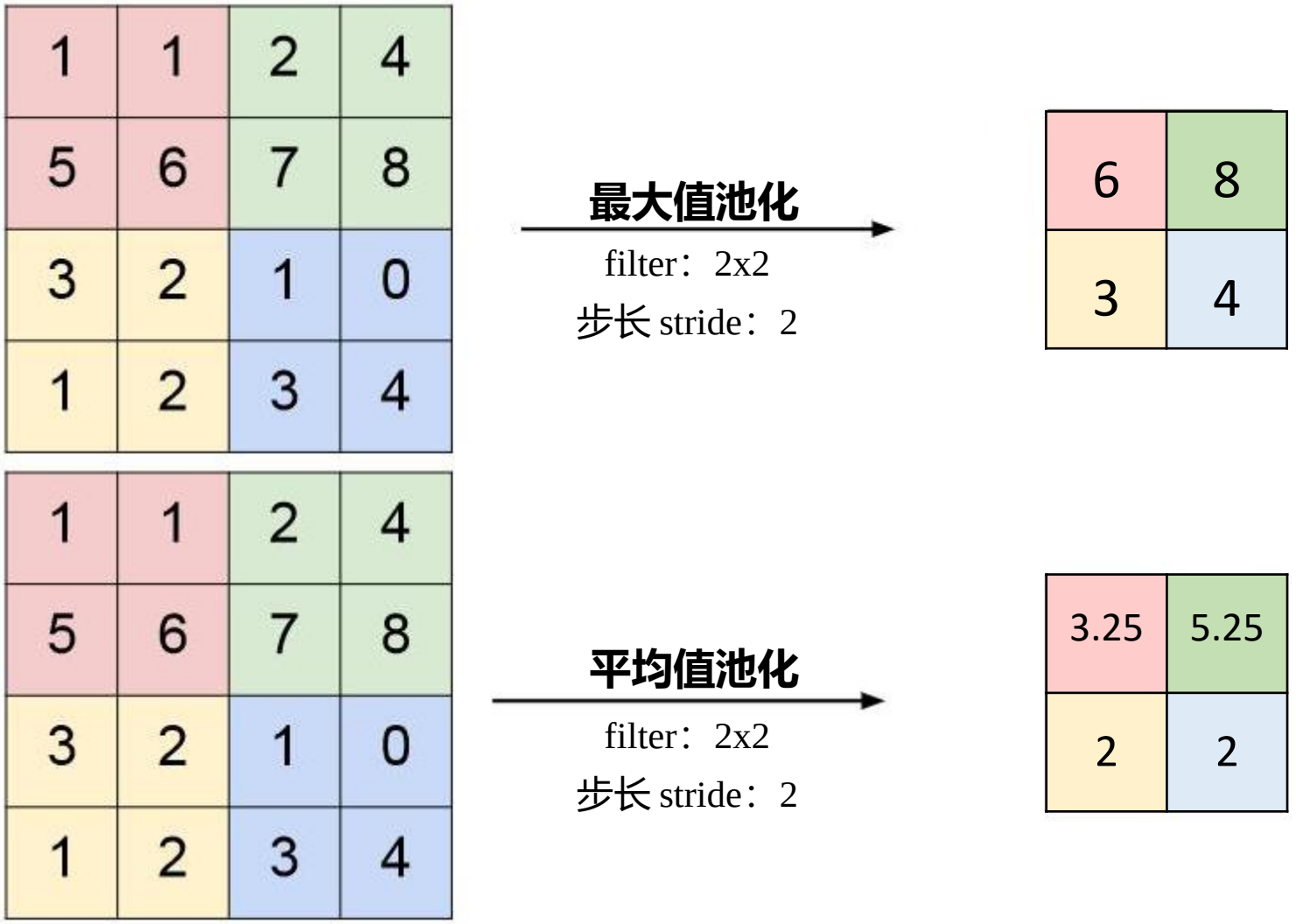
- 绪论
  1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络
- 基本组成结构
  1. 卷积 **2.池化** 3.全连接
- 卷积神经网络典型结构
  1. AlexNet 2. ZFNet 3. VGG 4. GoogleNet 5. ResNet

池化 — Pooling Layer



- **Pooling：**
  - 保留了主要特征的同时减少参数和计算量，防止过拟合，提高模型泛化能力。
  - 它一般处于卷积层与卷积层之间，全连接层与全连接层之间
- **Pooling的类型：**
  - Max pooling: 最大值池化
  - Average pooling: 平均池化

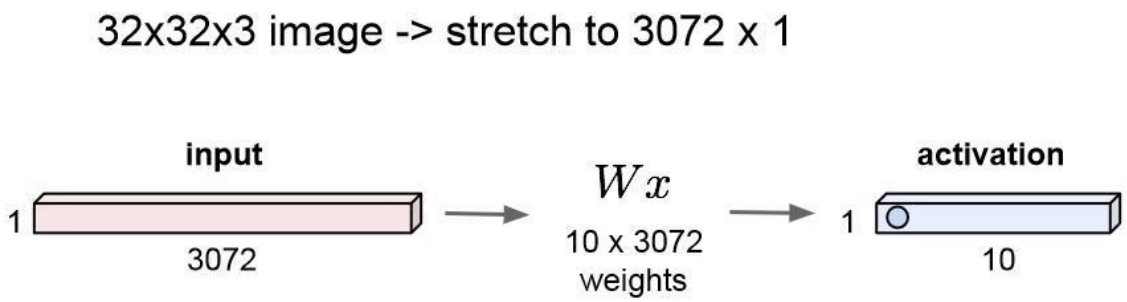
池化 — Pooling Layer



提 纲

- 绪论
  1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络
- 基本组成结构
  1. 卷积 2.池化 **3.全连接**
- 卷积神经网络典型结构
  1. AlexNet 2. ZFNet 3. VGG 4. GoogleNet 5. ResNet

全连接 — Fully Connected Layer



- **全连接层 / FC layer：**
  - 两层之间所有神经元都有权重链接
  - 通常全连接层在卷积神经网络尾部
  - 全连接层参数量通常最大

卷积神经网络

小结

- 一个典型的卷积网络是由卷积层、池化层、全连接层交叉堆叠而成
- 卷积是对两个实变函数的一种数学操作。
- 局部关联，参数共享
- 未加padding时输出的特征图大小：(N-F)/stride+1
- 有padding时输出的特征图大小：(N+padding*2-F)/stride+1
- Pooling的类型：Max pooling：最大值池化，Average pooling: 平均池化
- 全连接：通常全连接层在卷积神经网络尾部

提纲

- 绪论

  1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络

- 基本组成结构

  1. 卷积 2.池化 3.全连接

- 卷积神经网络典型结构

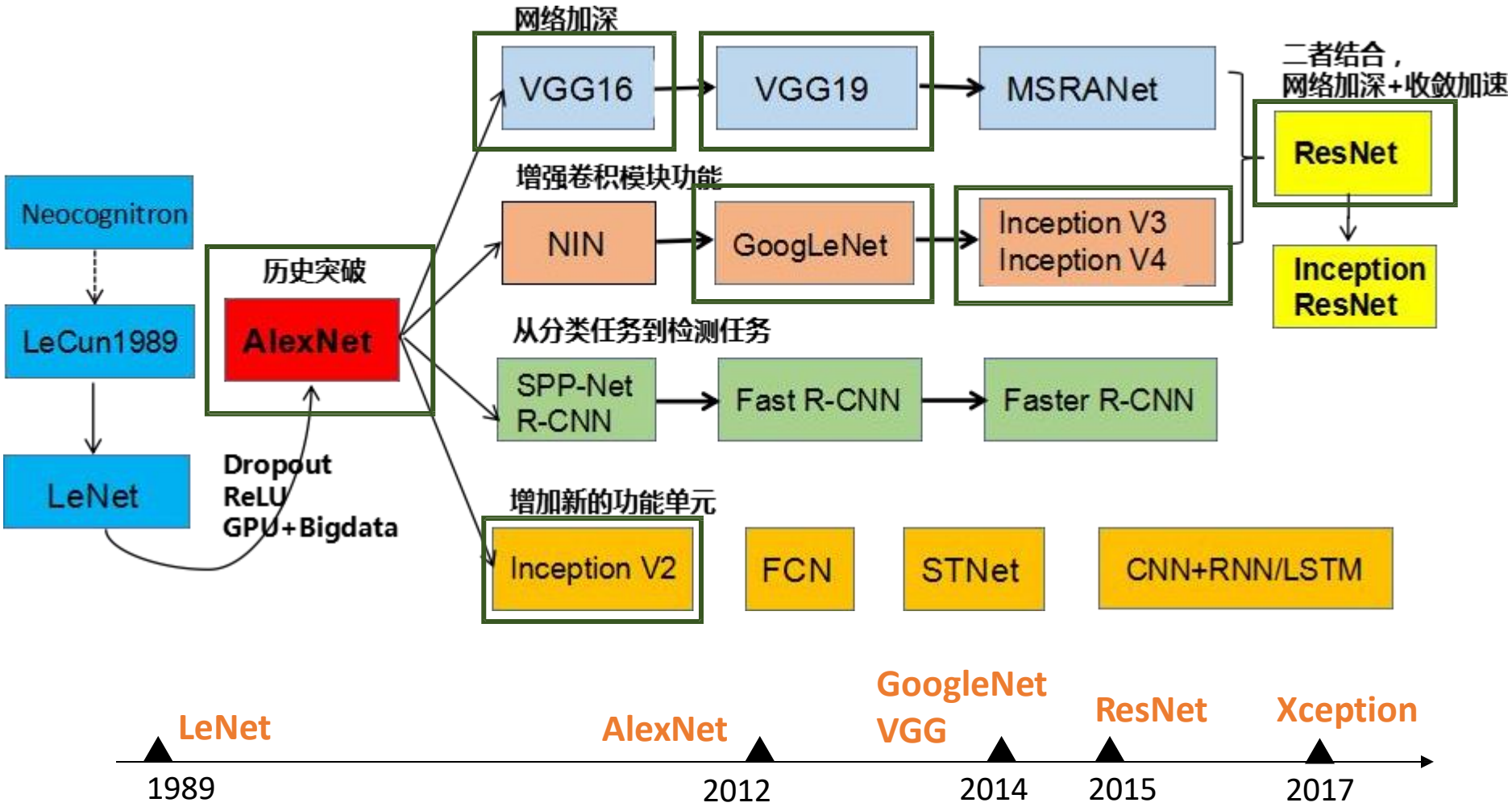
  1. AlexNet 2. ZFNet 3. VGG 4. GoogleNet 5. ResNet

卷积神经网络典型结构

2013年初，谷歌收购了 DNNresearch。该公司由Geoffrey Hinton教授于2012年建立，公司的核心员工包括Hinton教授的两个优秀的学生Alex Krizhevsky和 Ilya Sutskever

事实上，当时 Hinton 可以选择微软或IBM，之所以选择谷歌，Hinton说：“在谷歌，科学家和工程师没有明显区别。谁有先进的理论，谁就可以打造出具体的产品。”

卷积神经网络典型结构

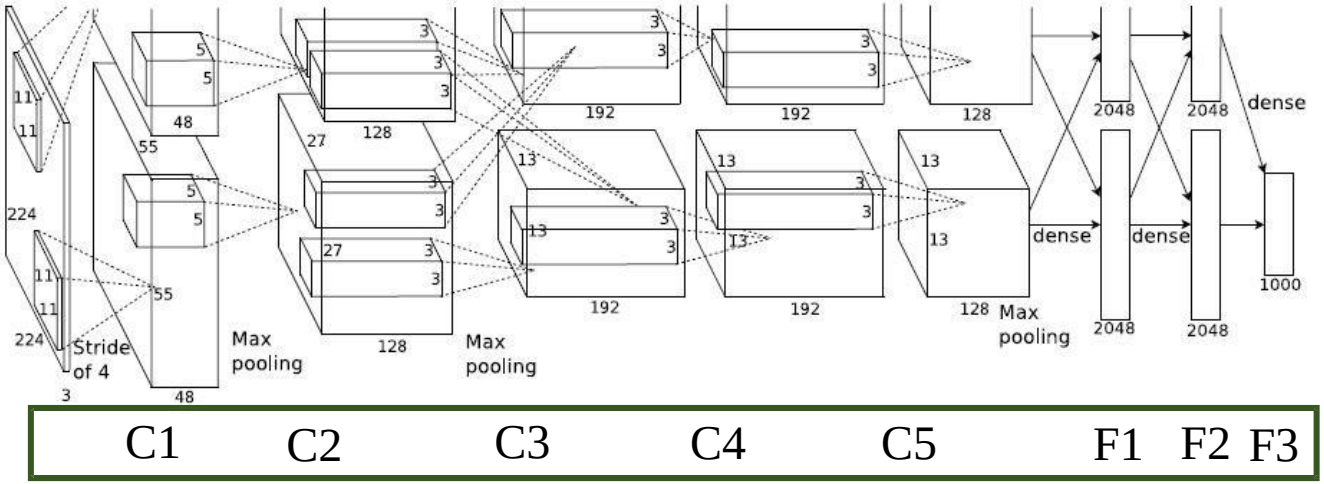

网络加深
VGG16
VGG19
MSRANet
二者结合，
网络加深+收敛加速
ResNet
Inception
ResNet
增强卷积模块功能
NIN
GoogLeNet
Inception V3
Inception V4
Neocognitron
LeCun1989
LeNet
历史突破
AlexNet
Dropout
ReLU
GPU+Bigdata
从分类任务到检测任务
SPP-Net
R-CNN
Fast R-CNN
Faster R-CNN
增加新的功能单元
Inception V2
FCN
STNet
CNN+RNN/LSTM
LeNet
AlexNet
GoogleNet
VGG
ResNet
Xception
1989
2012
2014
2015
2017

卷积神经网络典型结构 -- AlexNet

- AlexNet 是具有历史意义的一个网络结构，在AlexNet之前，深度学习已经沉寂了很久。
  - 历史的转折在2012年到来，AlexNet 在当年的ImageNet图像分类竞赛中，**错误率比上一年的冠军下降了十个百分点**，而且远远超过当年的第二名。

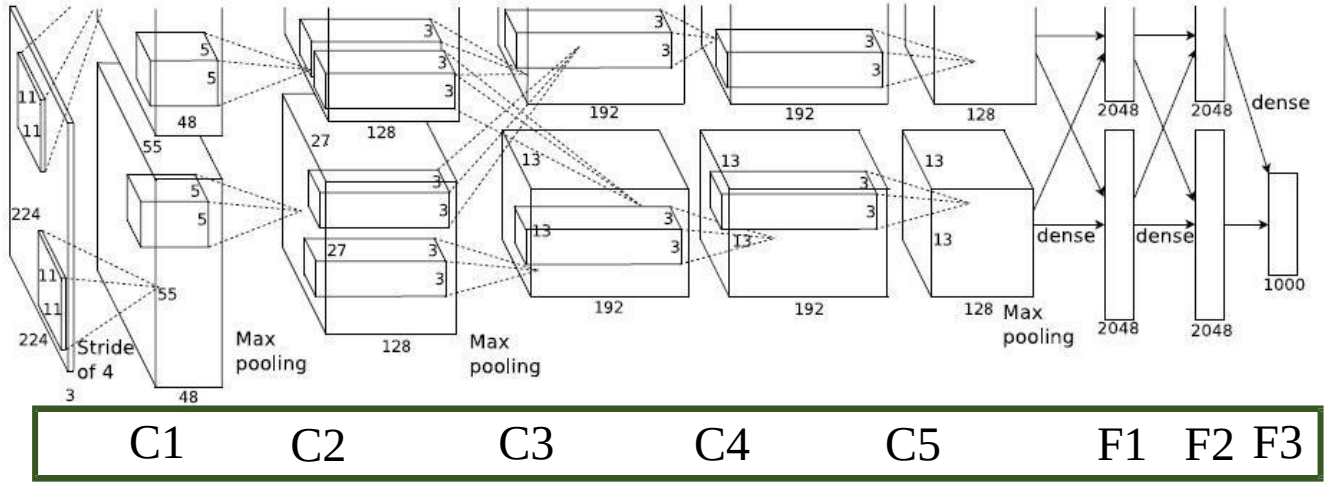
- 模型结构

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8



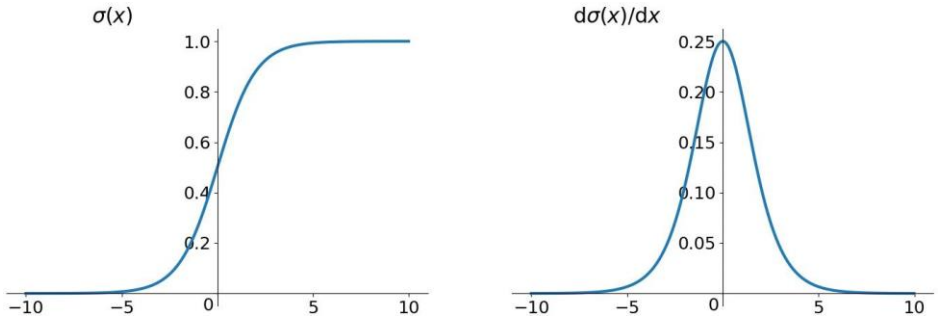
卷积神经网络典型结构 -- AlexNet

- **AlexNet 之所以能够成功，深度学习之所以能够重回历史舞台，原因在于**：
  - **大数据训练**：百万级ImageNet图像数据
  - **非线性激活函数**：ReLU
  - **防止过拟合**：Dropout, Data augmentation
  - **其他**：双GPU实现

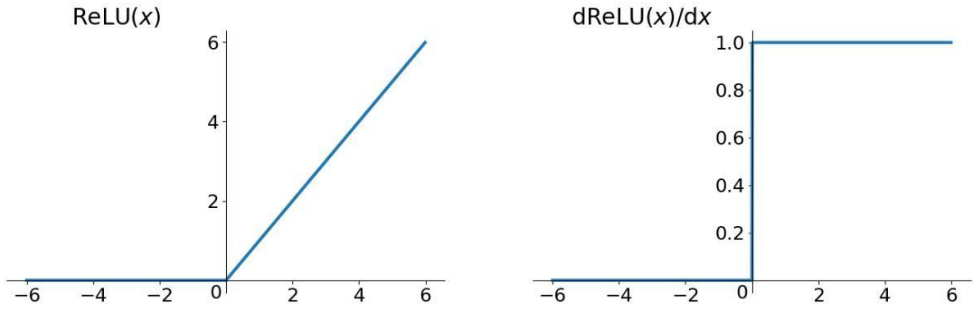


卷积神经网络典型结构 -- AlexNet

- **Sigmoid函数**



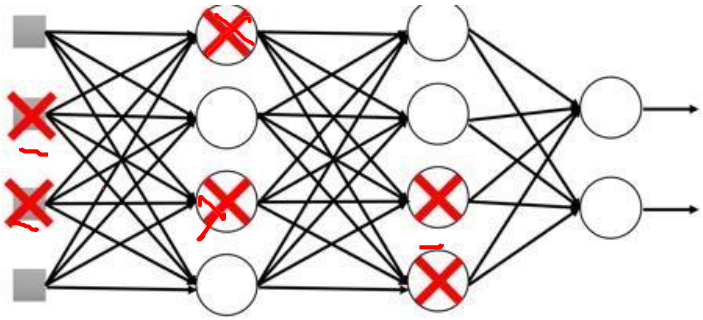
- **ReLU函数** $ReLU = max(0, x)$



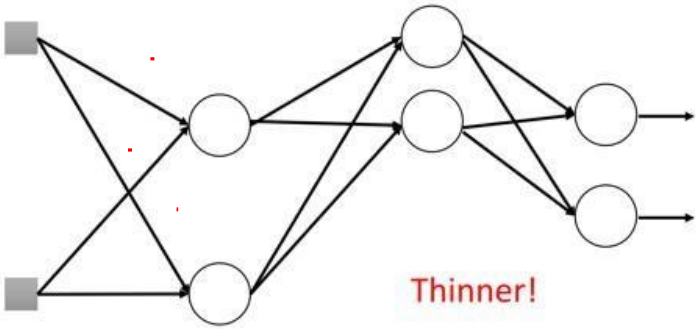
- 优点：
  - 解决了梯度消失的问题（在正区间）
  - 计算速度特别快，只需要判断输入是否大于0
  - 收敛速度远快于sigmoid

卷积神经网络典型结构 -- AlexNet

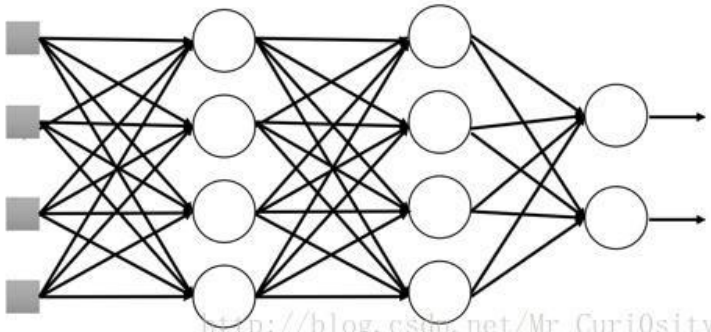
Training:



Training:

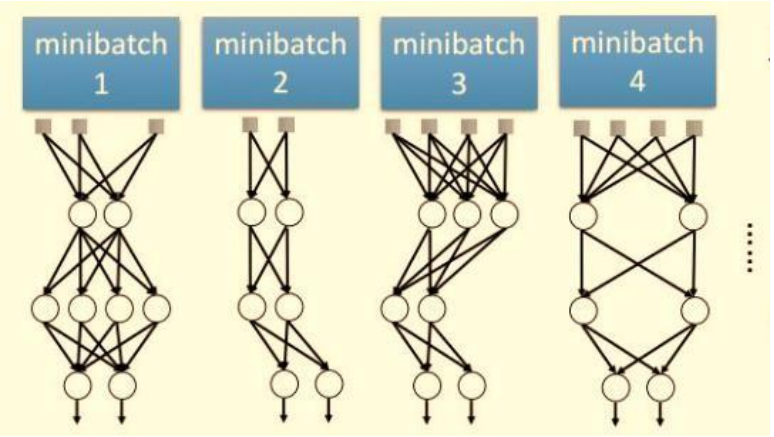


Testing:



Dropout（随机失活）

训练时随机关闭部分神经元
测试时整合所有神经元



隐式的模型集成

卷积神经网络典型结构 -- AlexNet

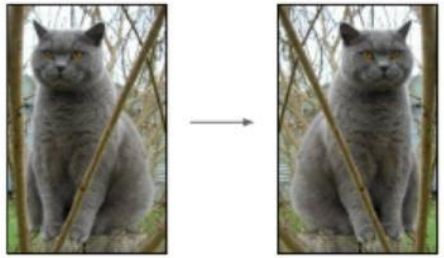
数据增强（data augmentation)

- 平移、翻转、对称
  - 随机crop。训练时候，对于256 * 256的图片进行随机crop到224 * 224。
  - 水平翻转，相当将样本倍增。
- 改变RGB通道强度
  - 对RGB空间做一个高斯扰动。

每个RGB图片的像素 $I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T$

$$I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T + [p_1, p_2, p_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

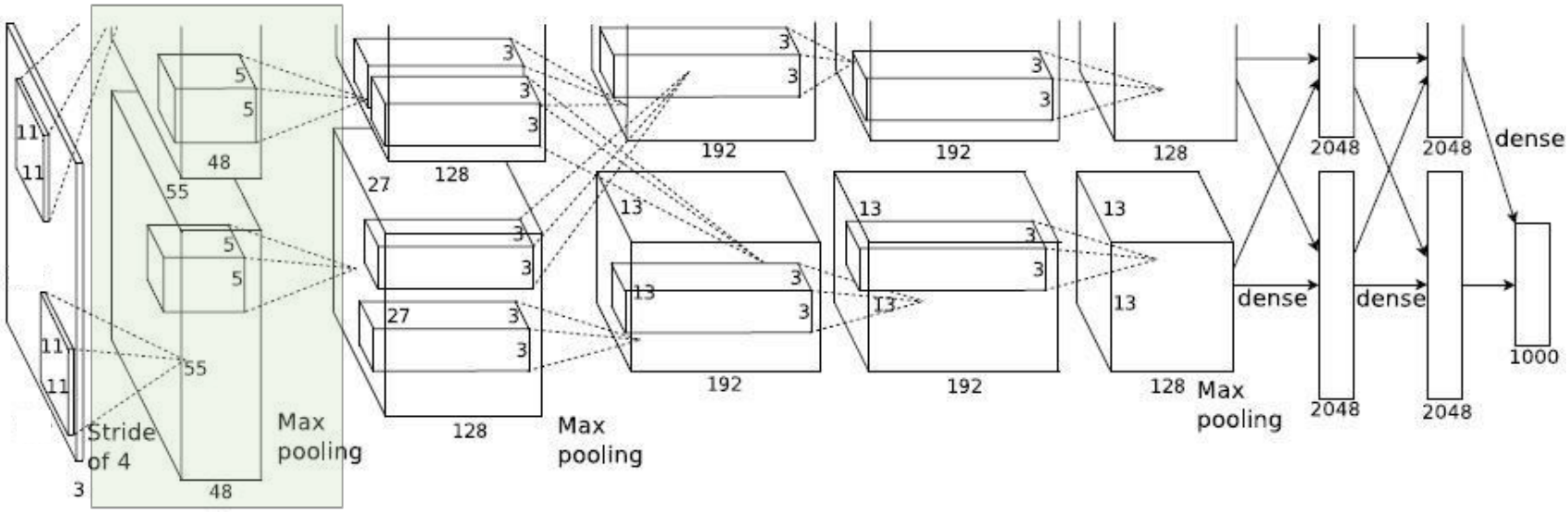
Random crops/scales



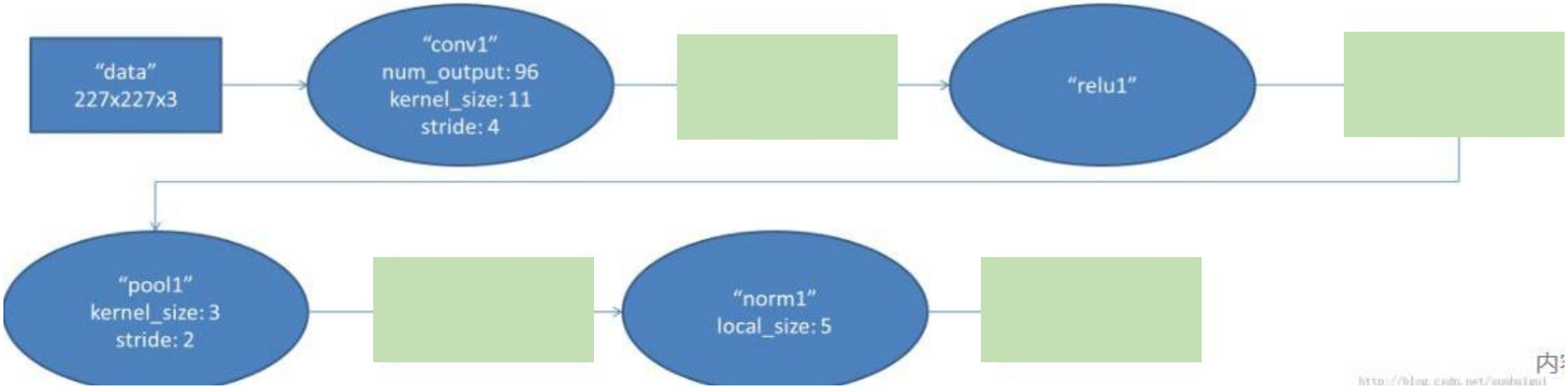
Flip horizontally

Color jittering

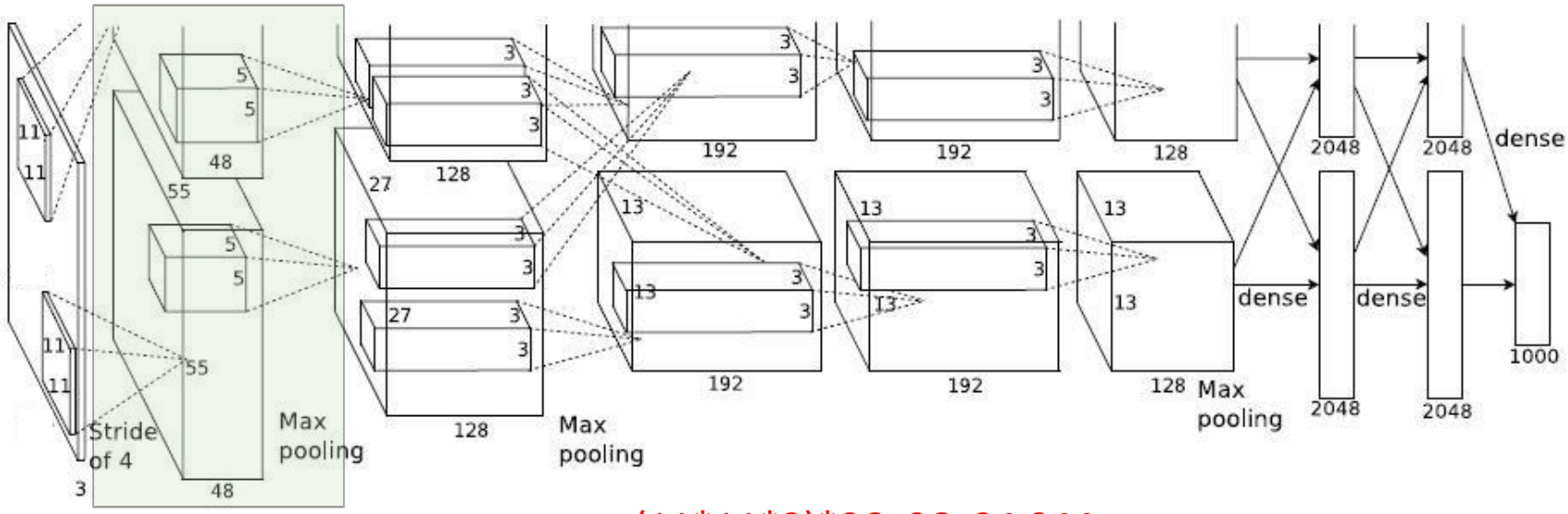
AlexNet分层解析



第一次卷积：卷积 - ReLU - 池化

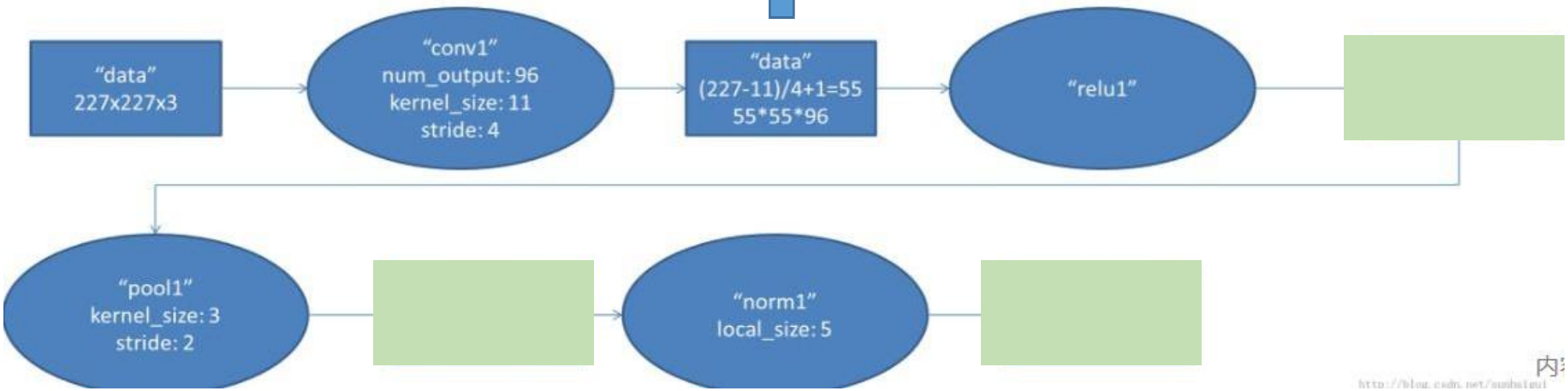


AlexNet分层解析

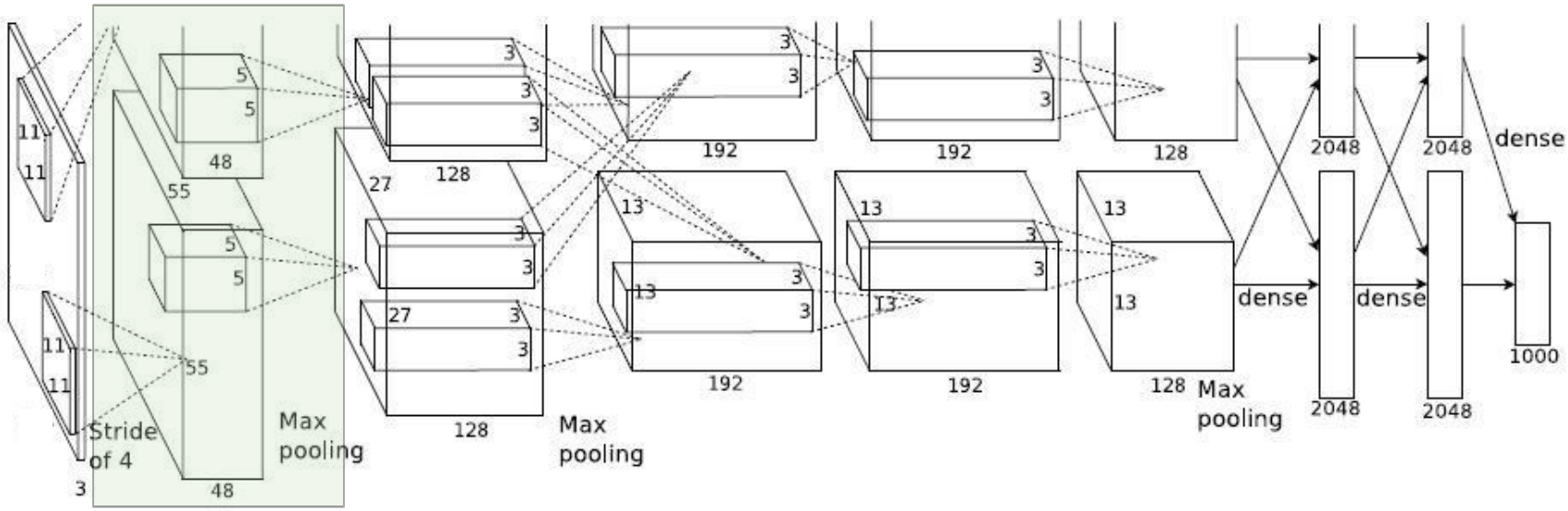


(11*11*3)*96+96=34,944

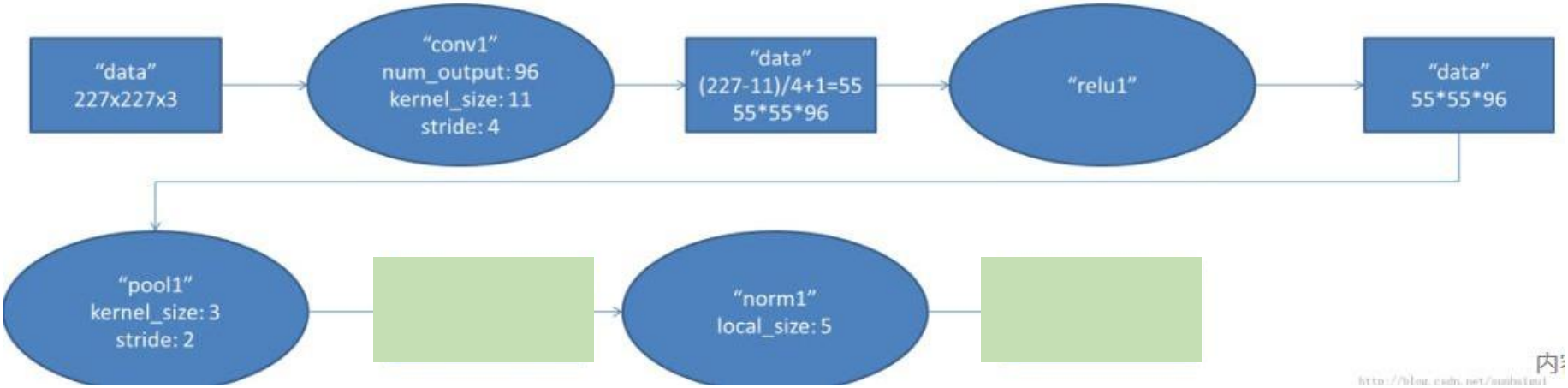
第一次卷积：卷积 - ReLU - 池化



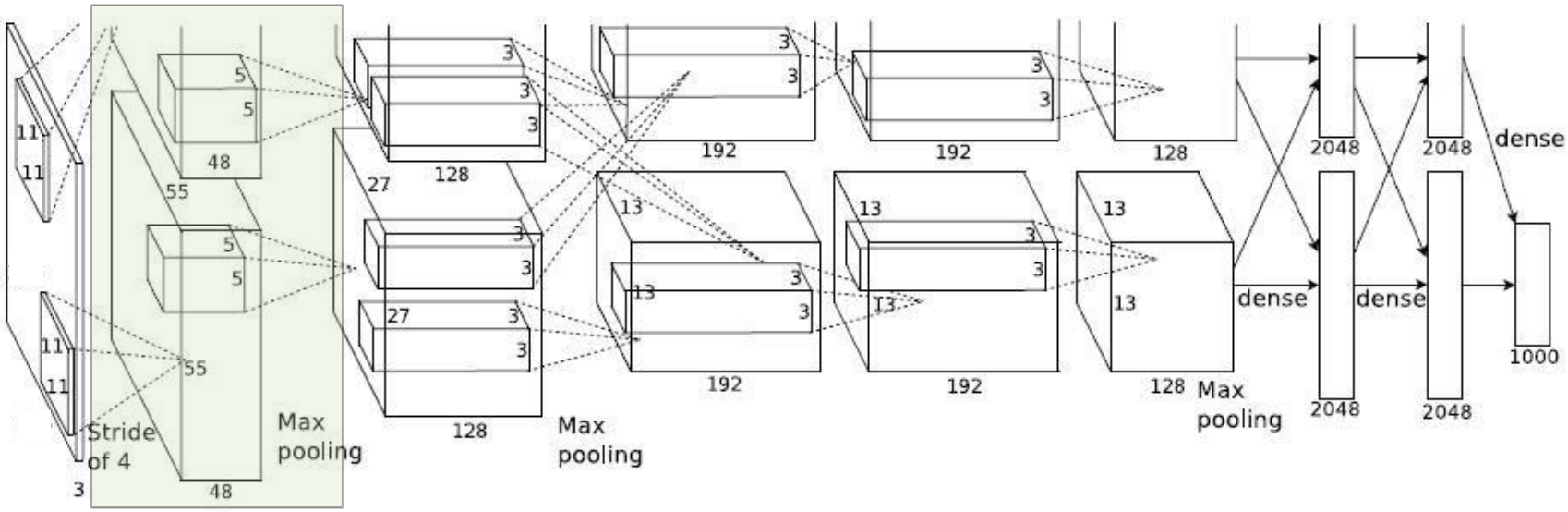
AlexNet分层解析



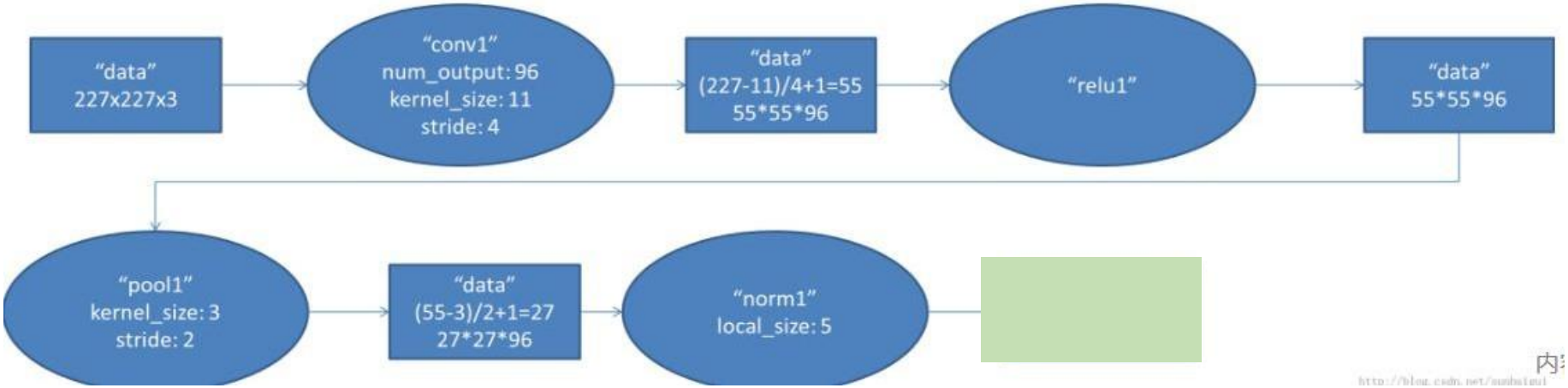
第一次卷积：卷积 - ReLU - 池化



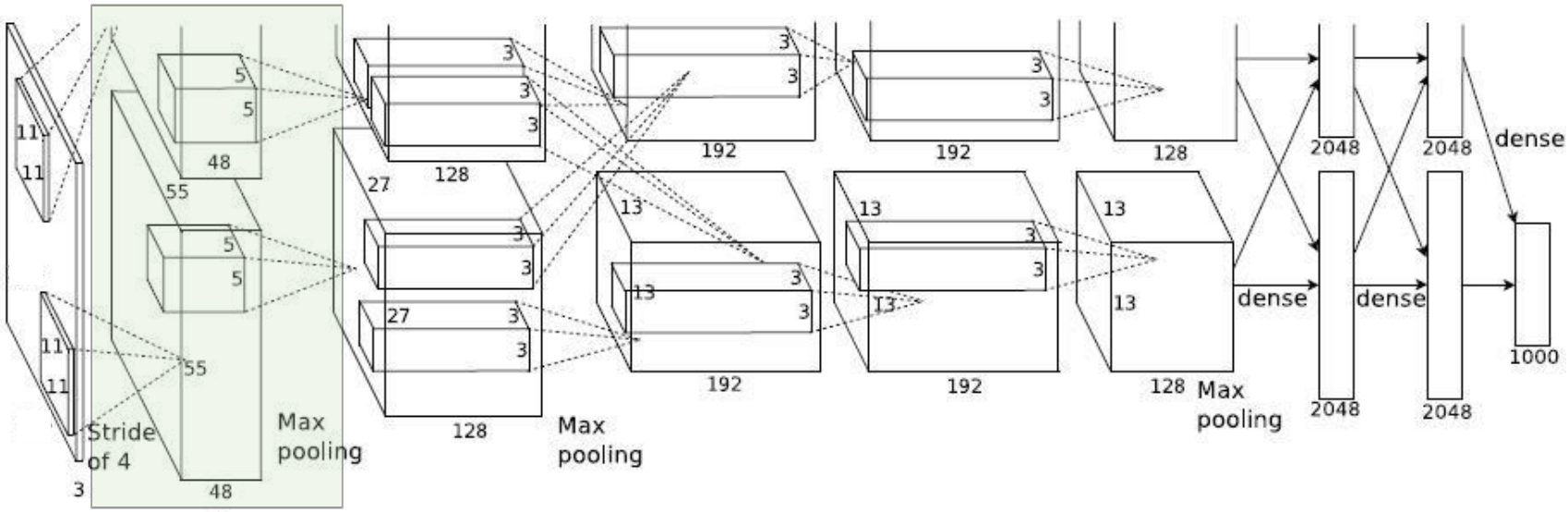
AlexNet分层解析



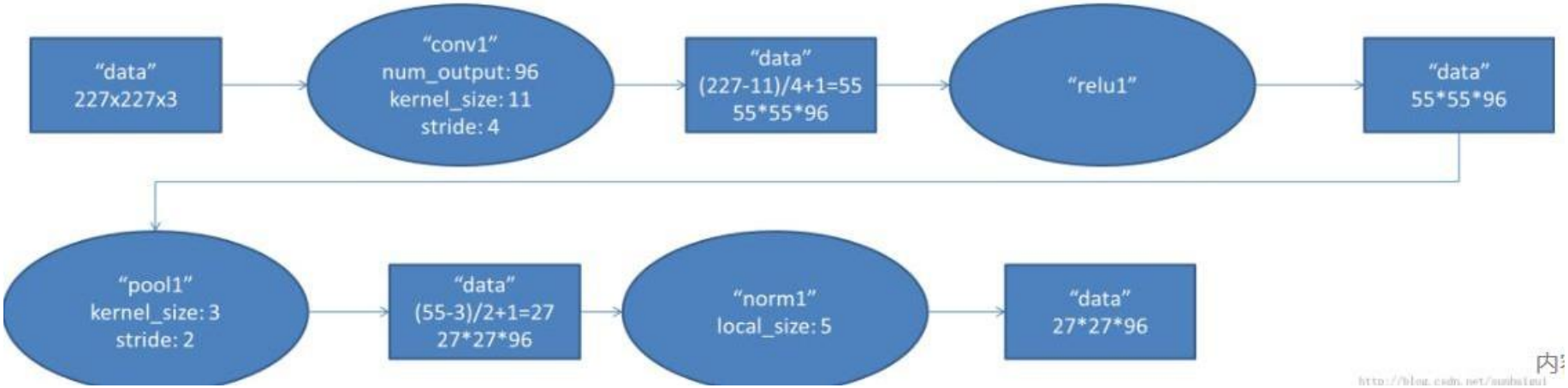
第一次卷积：卷积 - ReLU - 池化



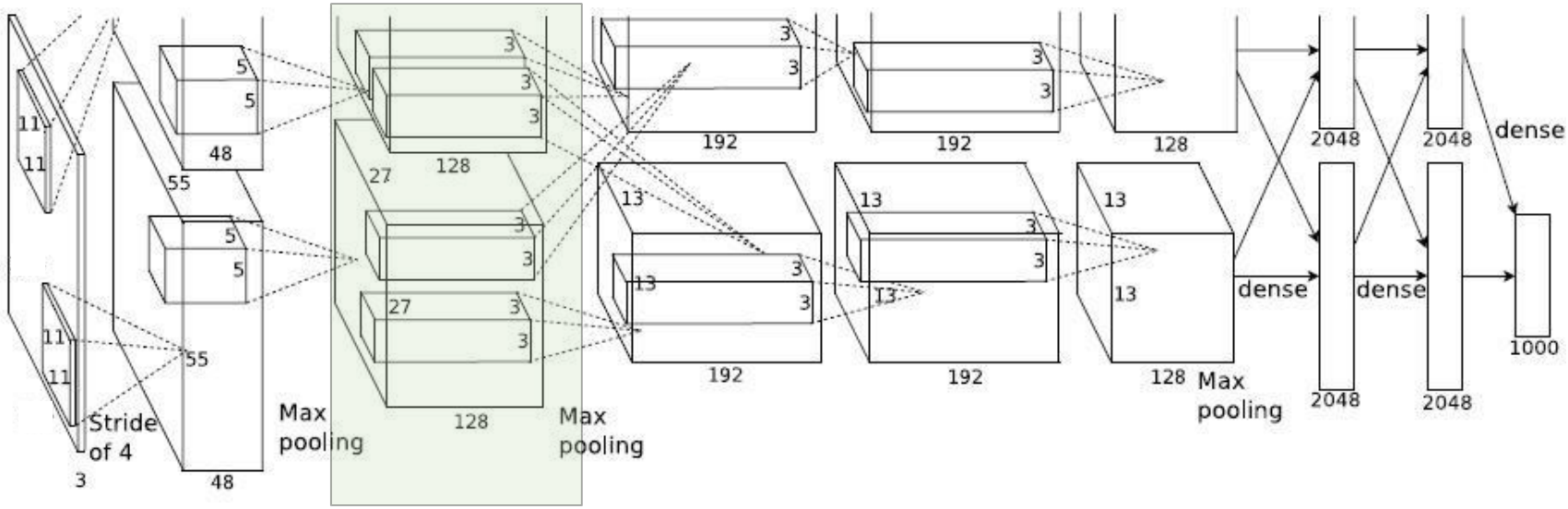
AlexNet分层解析



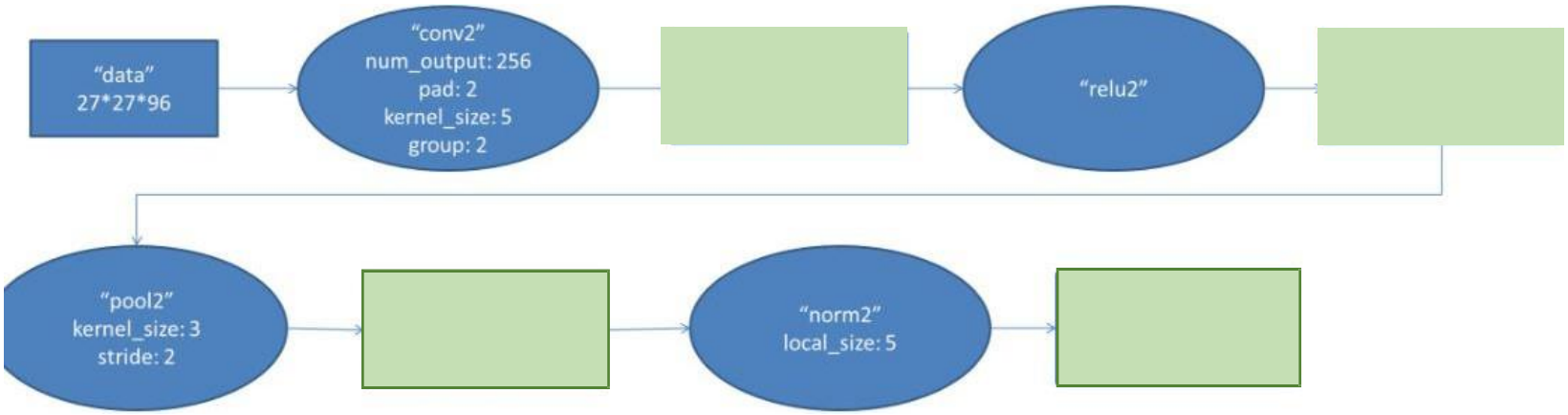
第一次卷积：卷积 - ReLU - 池化



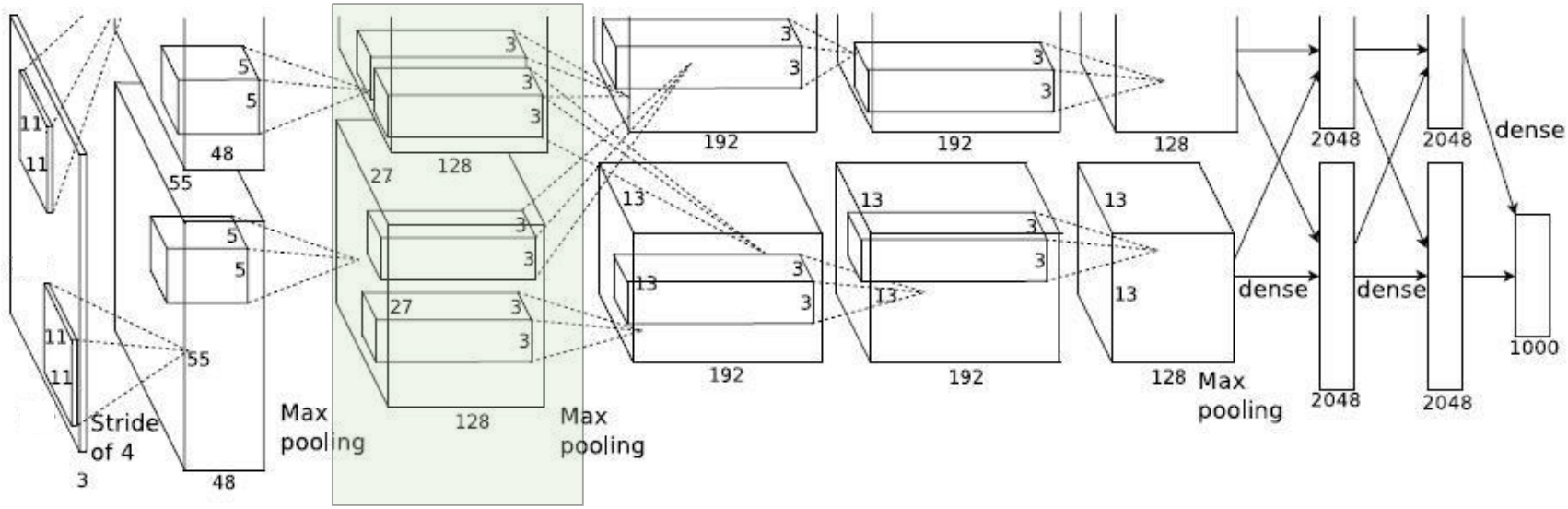
AlexNet分层解析



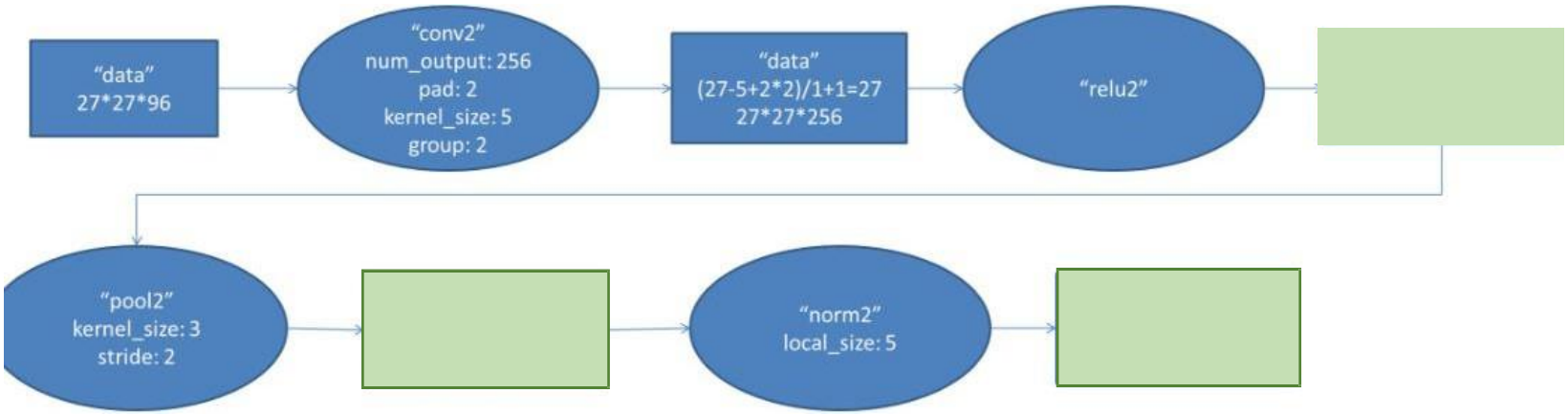
第二次卷积：卷积 – ReLU - 池化



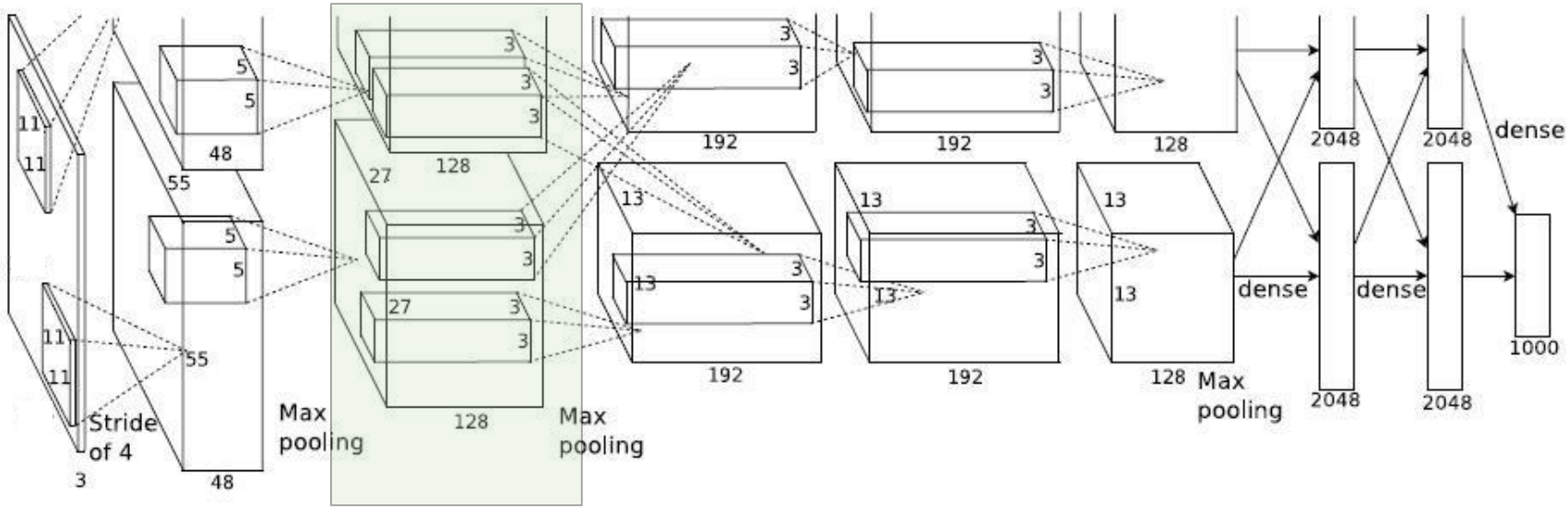
AlexNet分层解析



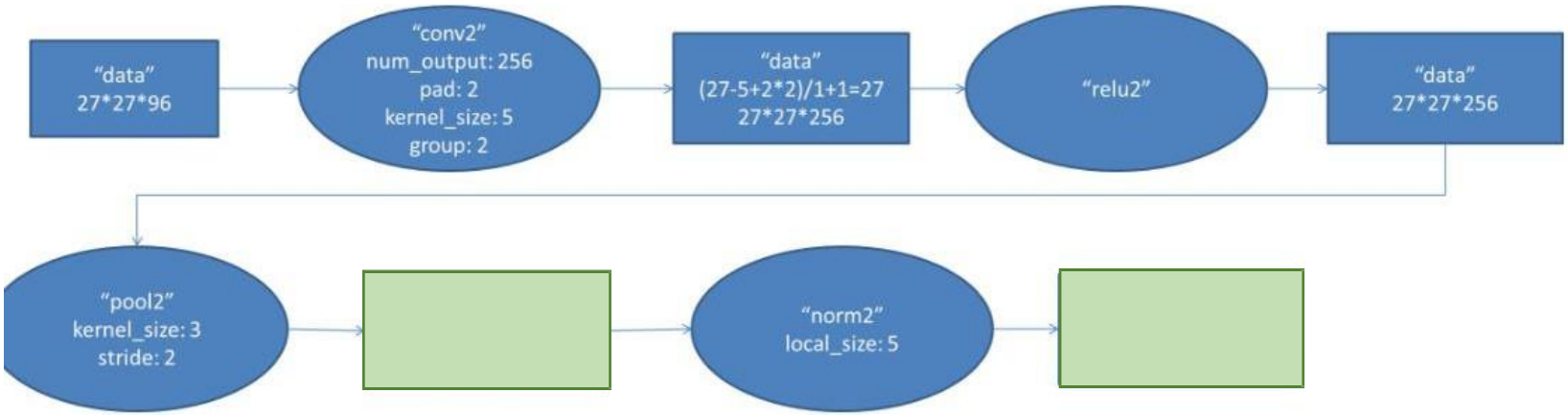
第二次卷积：卷积 – ReLU - 池化



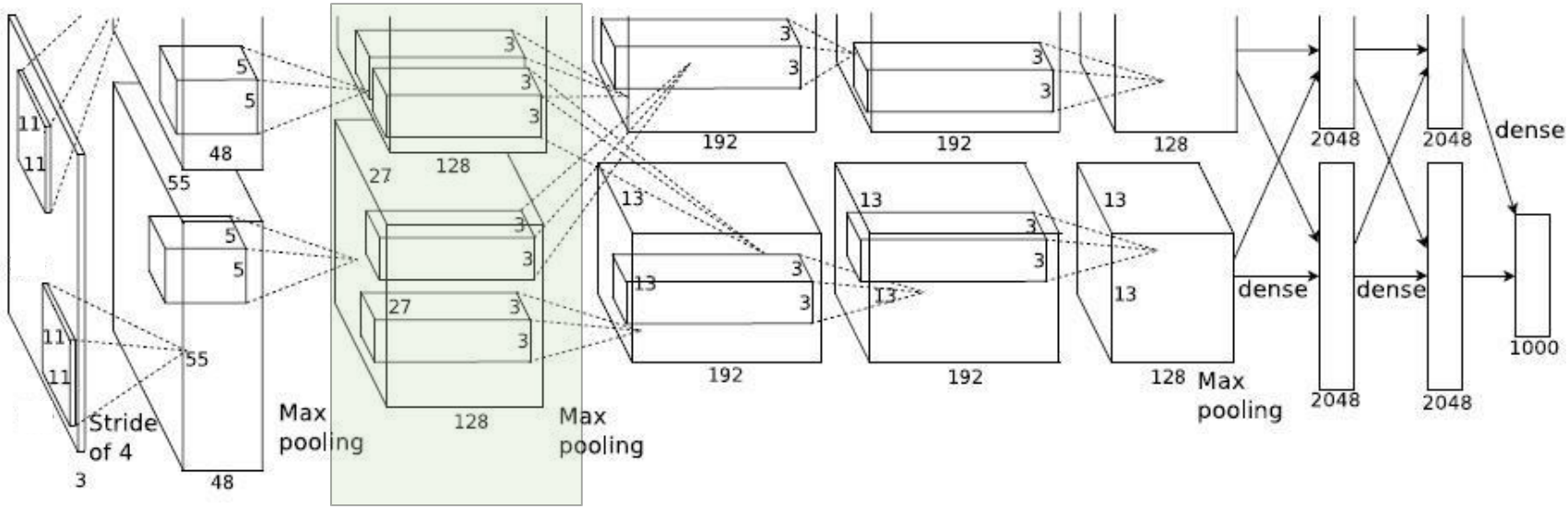
AlexNet分层解析



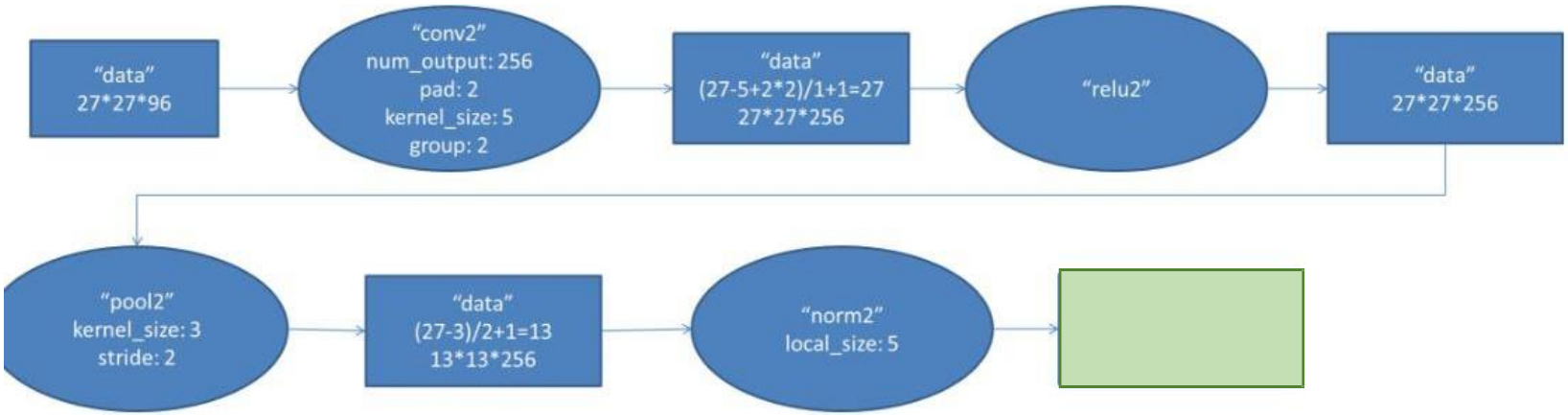
第二次卷积：卷积 – ReLU - 池化



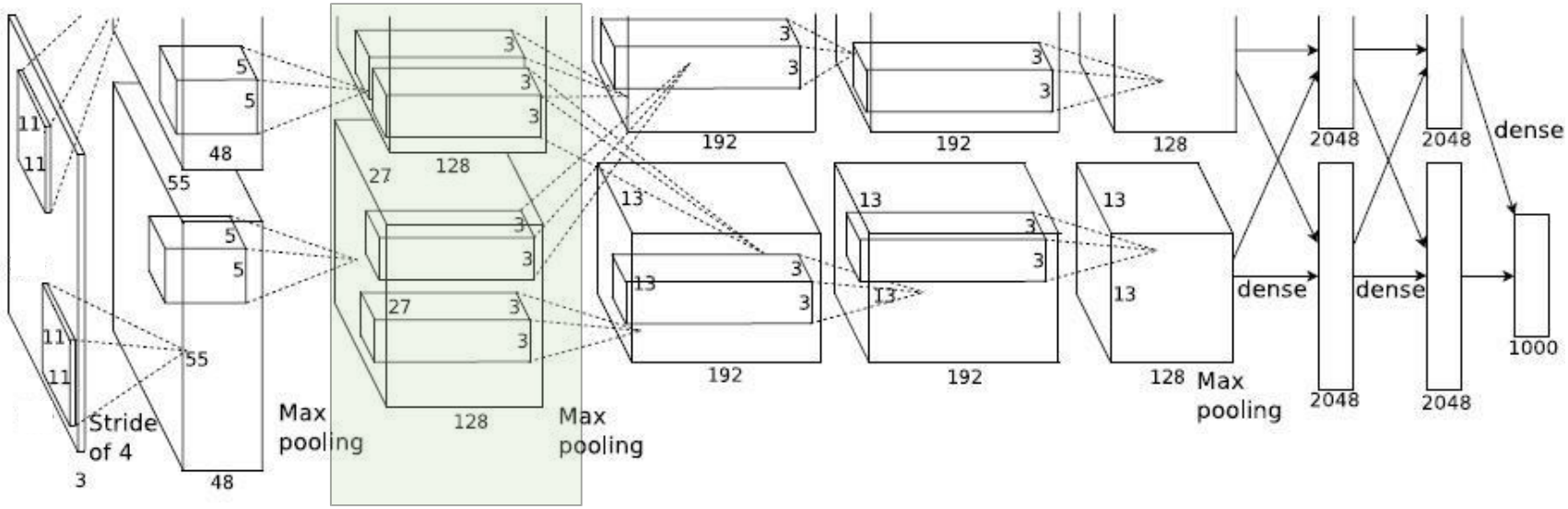
AlexNet分层解析



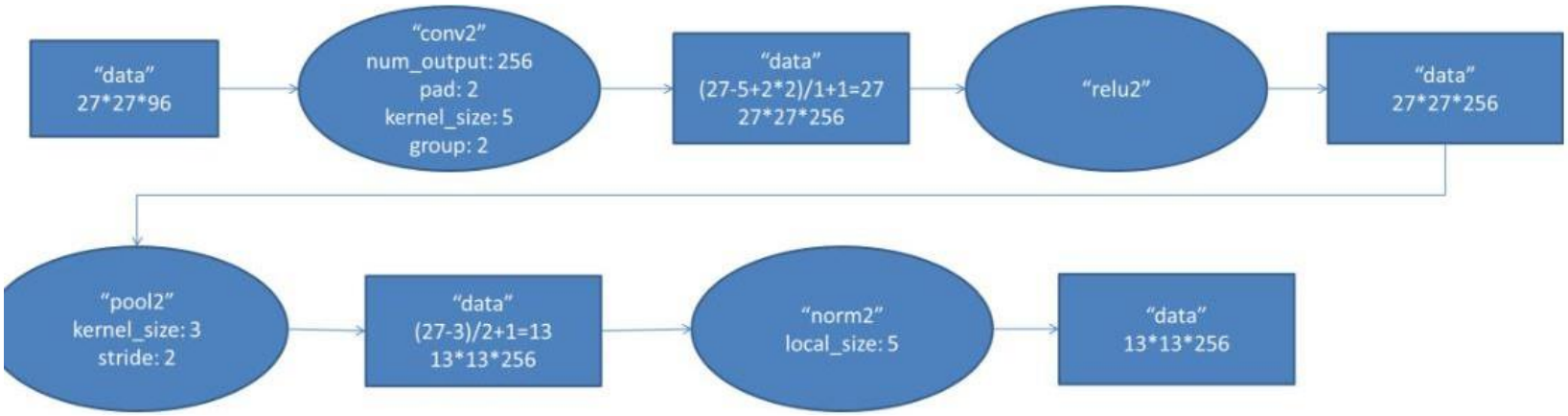
第二次卷积：卷积 – ReLU - 池化



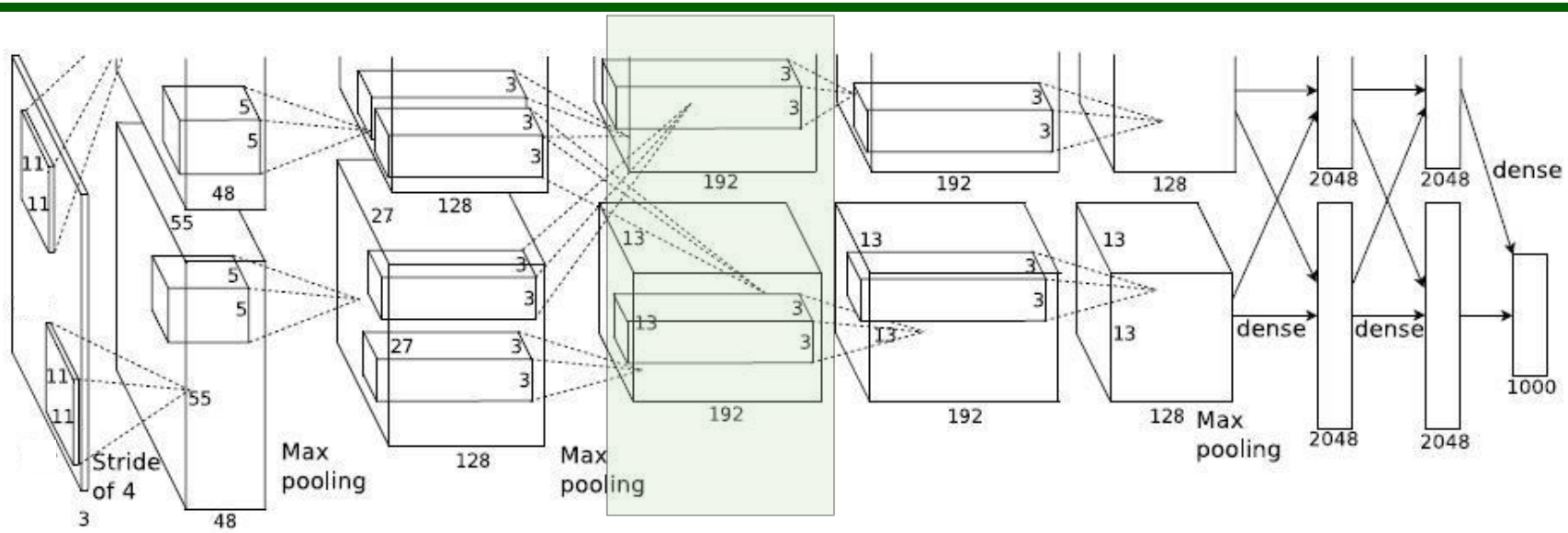
AlexNet分层解析



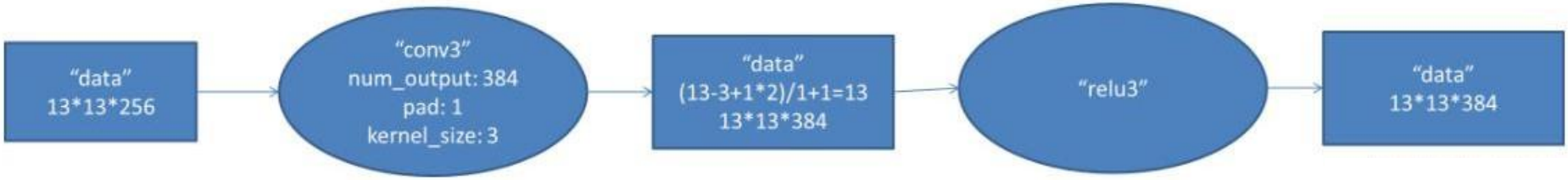
第二次卷积：卷积 – ReLU - 池化



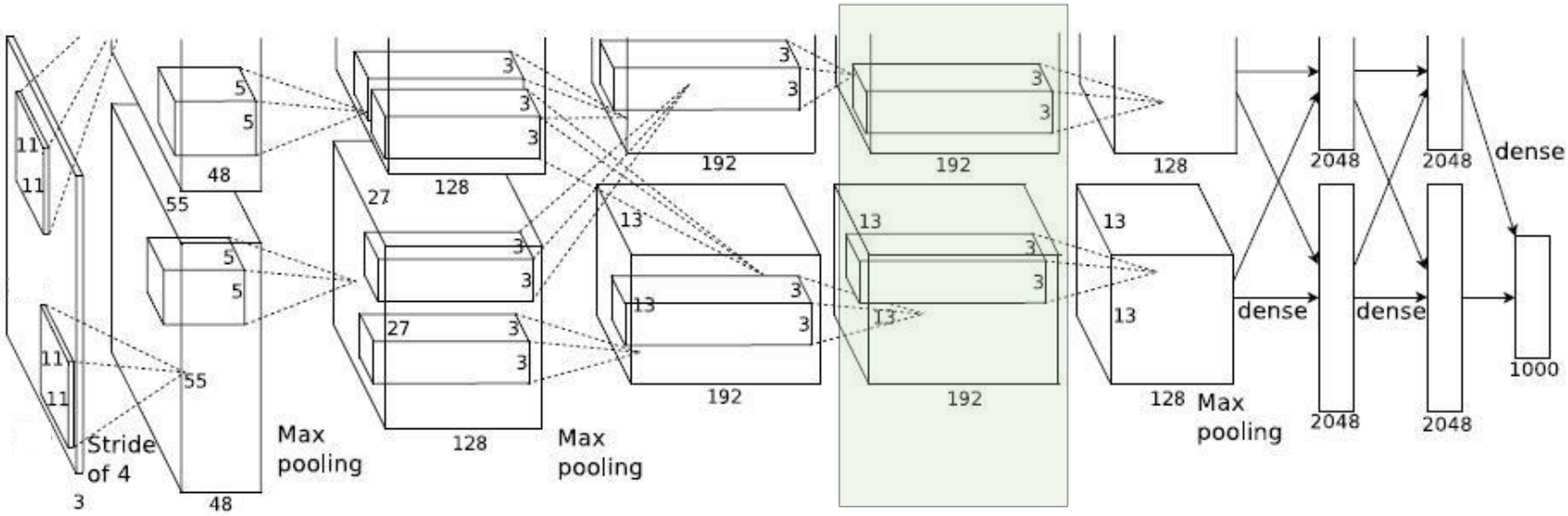
AlexNet分层解析



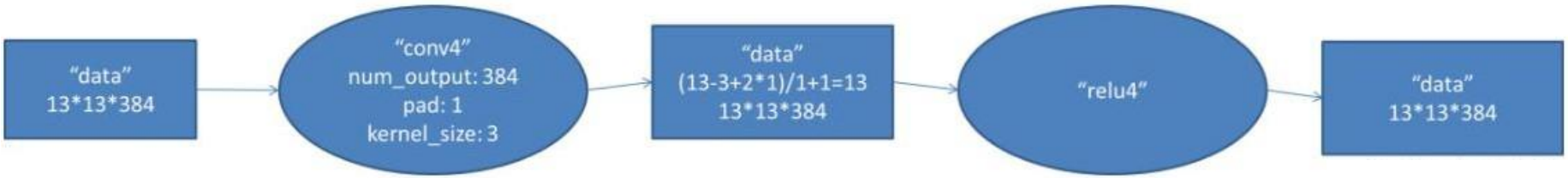
第三次卷积：卷积 - ReLU



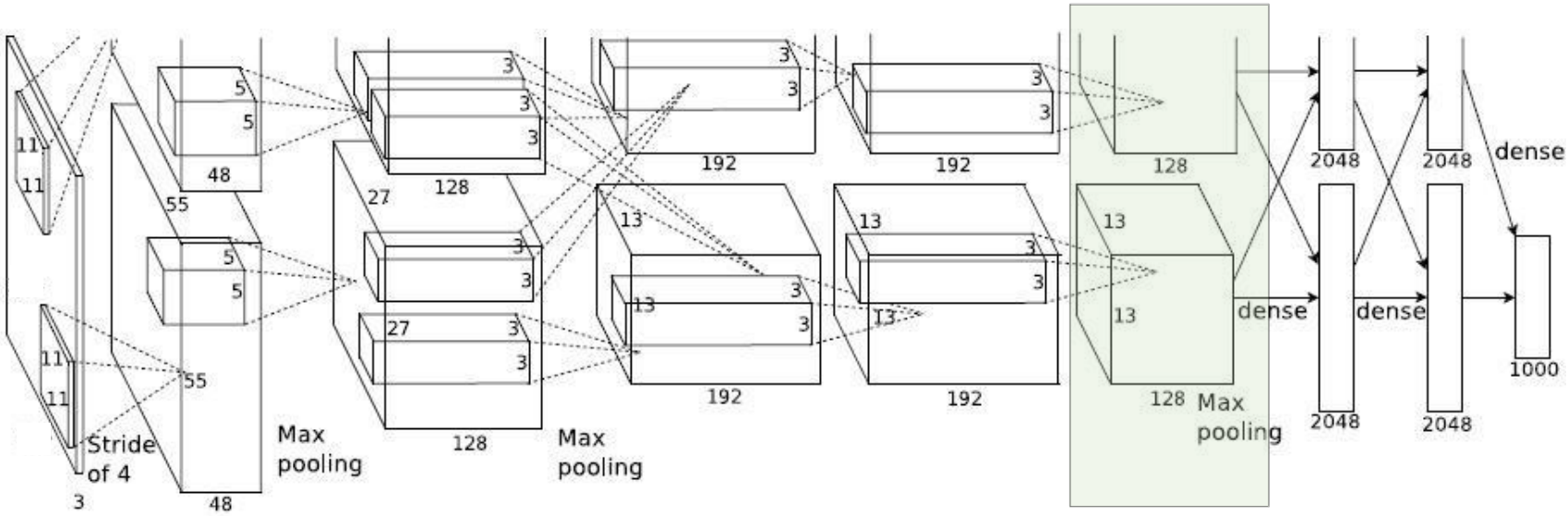
AlexNet分层解析



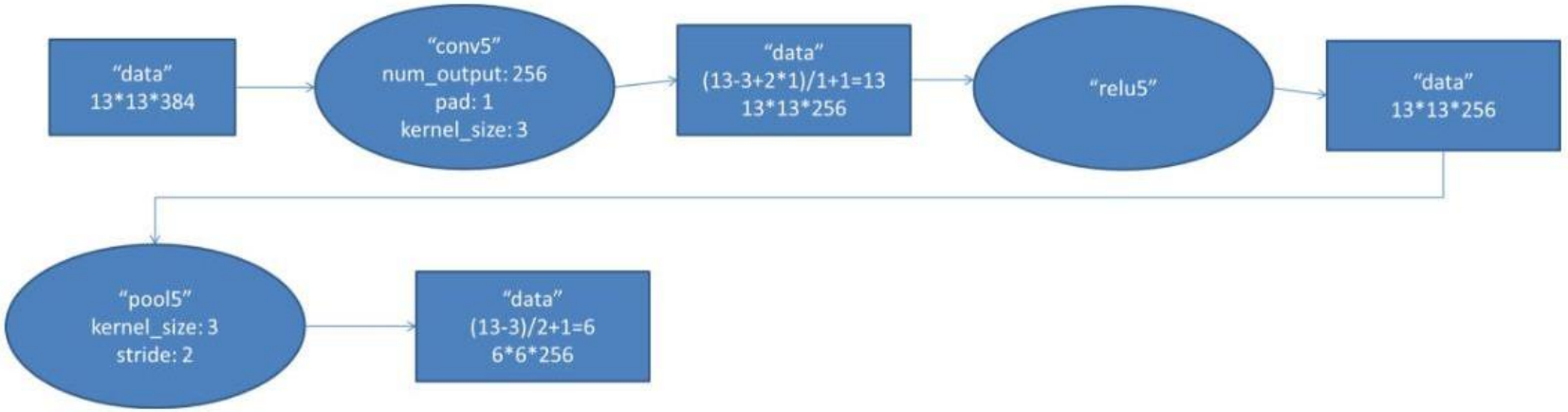
第四次卷积：卷积 - ReLU



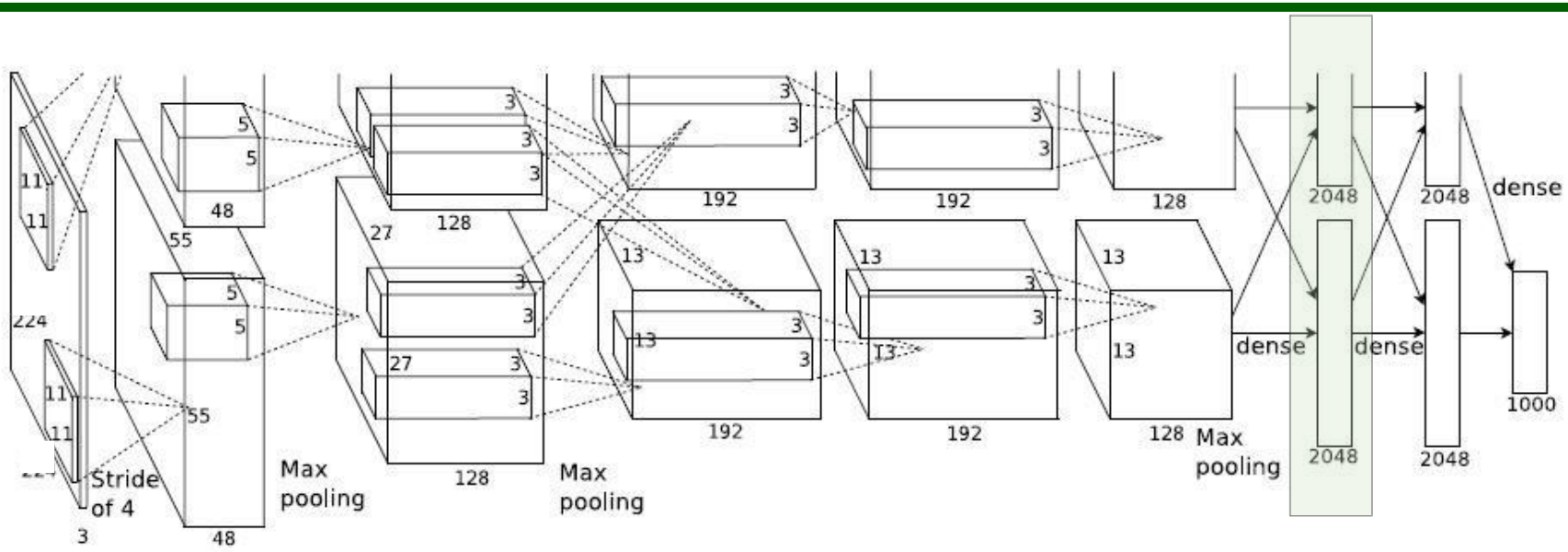
AlexNet分层解析



第五次卷积：卷积 – ReLU – 池化

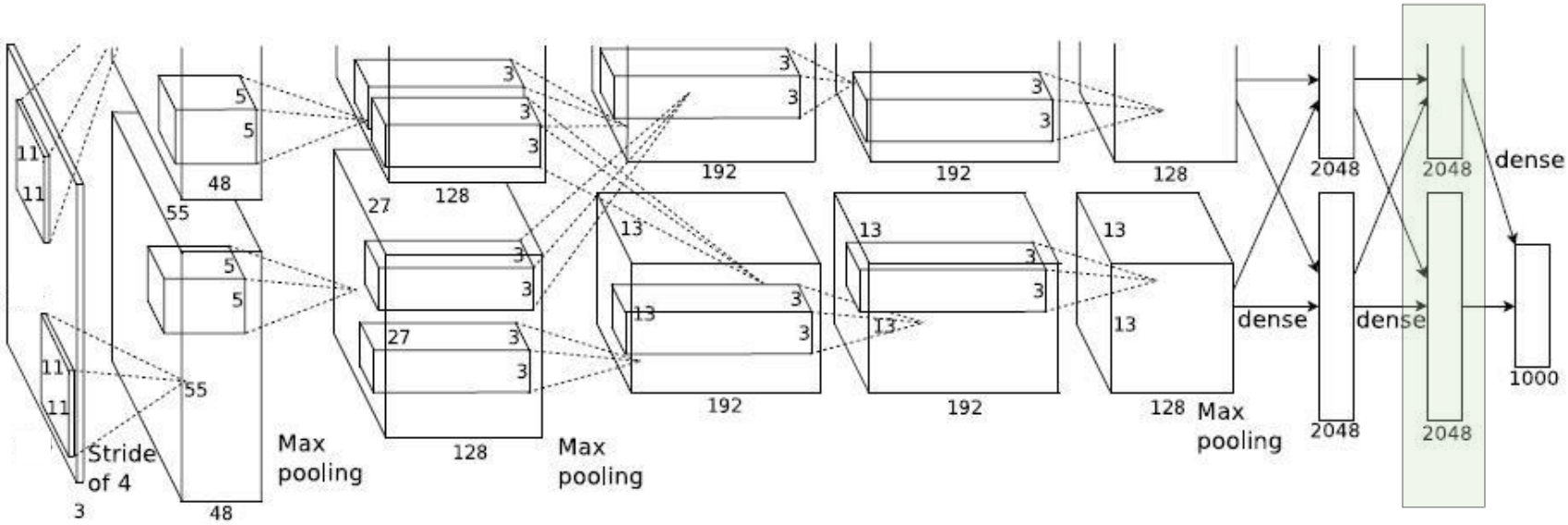


AlexNet分层解析

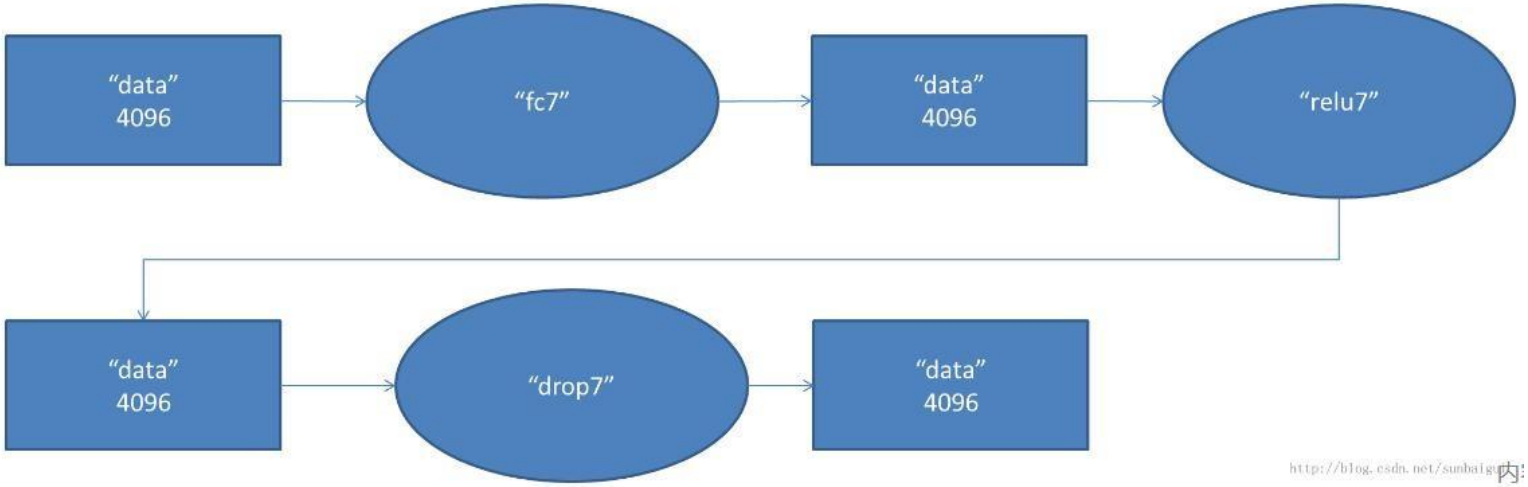


第六层：全连接 – ReLU – DropOut

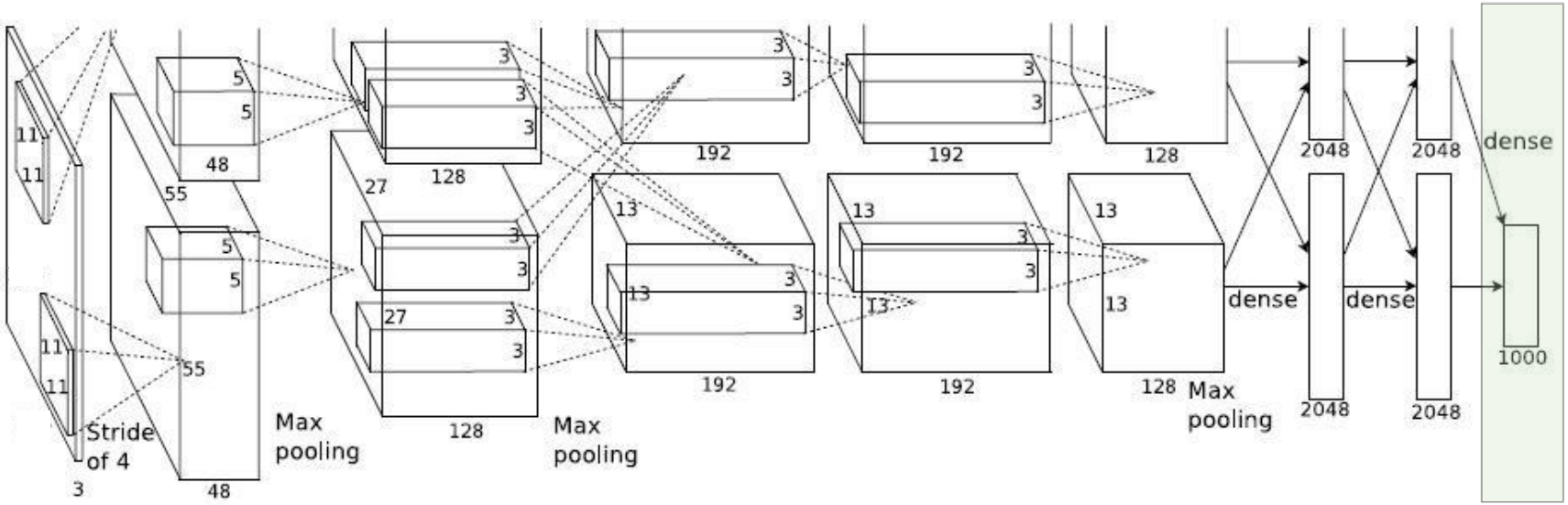
AlexNet分层解析



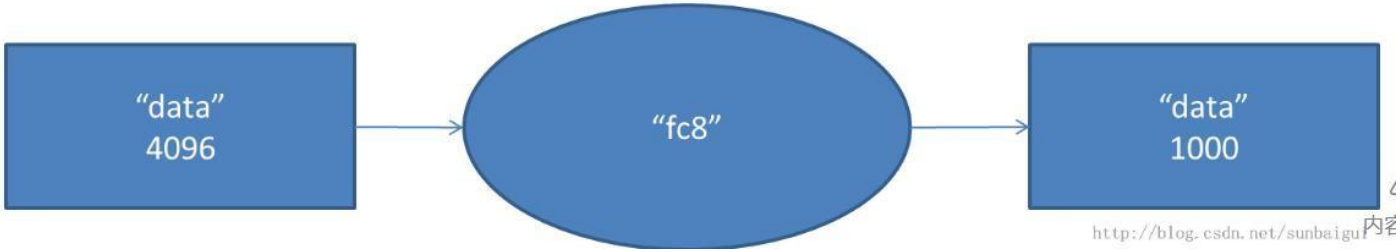
第七层：全连接 – ReLU – DropOut



AlexNet分层解析



第八层：全连接 – SoftMax



AlexNet分层解析

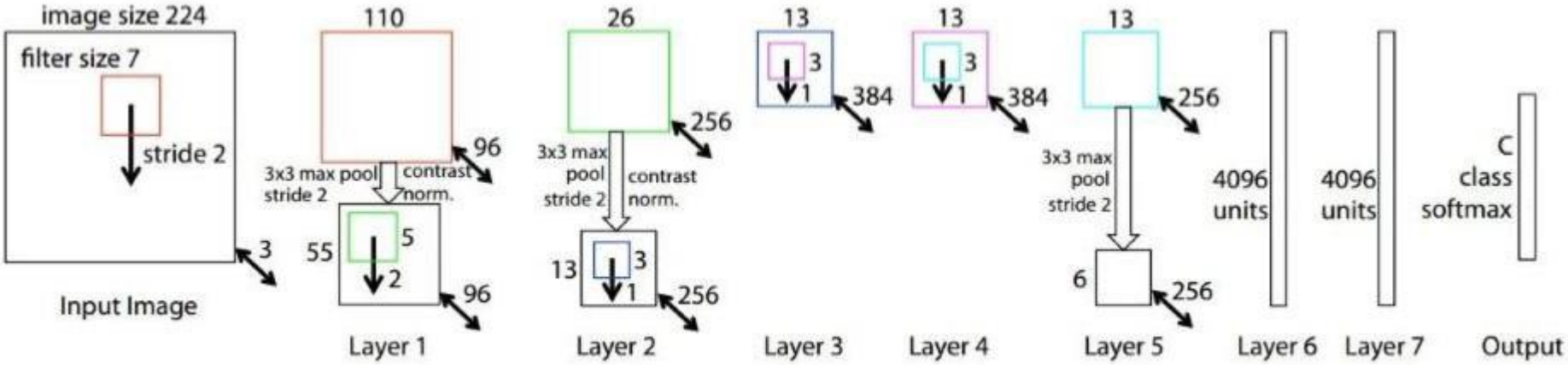
参数数量

层名称	说明	节点数量	参数数量(w和b)
Input	W*H*C = 224*224*3	224*224*3 = 150528	N/A
CNN1	96个11x11x3的卷积核	55*55*48*2 = 290400	11*11*3*96+96 = 34848
CNN2	256个5x5x48卷积核，只读当前GPU	27*27*128*2 = 186624	(5*5*48*128+128)*2 = 307456
CNN3	384个3x3x256卷积核	13*13*192*2 = 64896	3*3*256*384+384 = 885120
CNN4	384个3x3x192卷积核，只读当前GPU	13*13*192*2 = 64896	(3*3*192*192+192)*2 = 663936
CNN5	256个3x3x192卷积核，只读当前GPU	13*13*128*2 = 43264	(3*3*192*128+128)*2 = 442624
FC1	全连接层,上一层有maxpooling(s=2)	4096	(6*6*128*2)*4096+4096 = 37752832
FC2	全连接层	4096	4096*4096 + 4096= 16781312
FC3 (output)	全连接层	1000	4096*1000 + 1000 = 4097000
合计	N/A	809800	本列全部相加结果：60965128

提纲

- 绪论
  1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络
- 基本组成结构
  1. 卷积 2.池化 3.全连接
- 卷积神经网络典型结构
  1. AlexNet **2. ZFNet** 3. VGG 4. GoogleNet 5. ResNet

ZFNet



- 网络结构与AlexNet相同
- 将卷积层1中的感受野大小由 11X11　改为 7X7，步长由4改为2
- 卷积层3，4，5中的滤波器个数由384，384，256改为512，512，1024

2013年ImageNet图像分类竞赛的冠军
ImageNet top 5 error: 16.4% -> 14.8%

提纲

- 绪论
  - 1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络
- 基本组成结构
  - 1. 卷积 2.池化 3.全连接
- 卷积神经网络典型结构
  - 1. AlexNet 2. ZFNet **3. VGG** 4. GoogleNet 5. ResNet

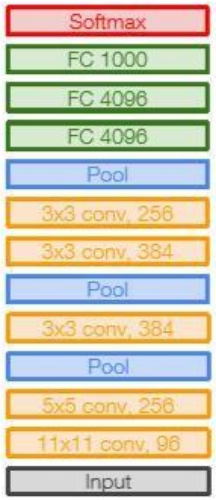
卷积神经网络典型结构 – VGG

- **VGG是一个更深网络**
  - 8 layers (AlexNet) -> 16 – 19 (VGG)
  - ILSVRC top 5 错误率从11.7% -> 7.3%

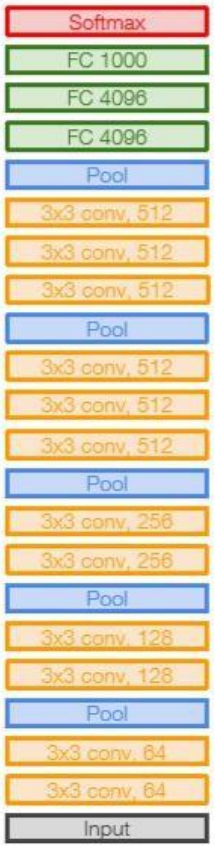
是2014年竞赛的第二名

训练这样一个网络是非常困难的，当时还没有BatchNorm技术

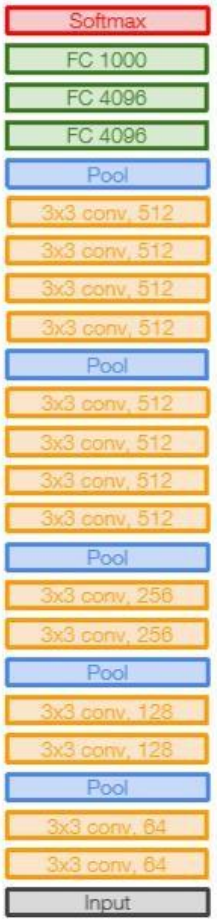
VGG网络影响巨大，一直到今天仍然很多人在使用



AlexNet

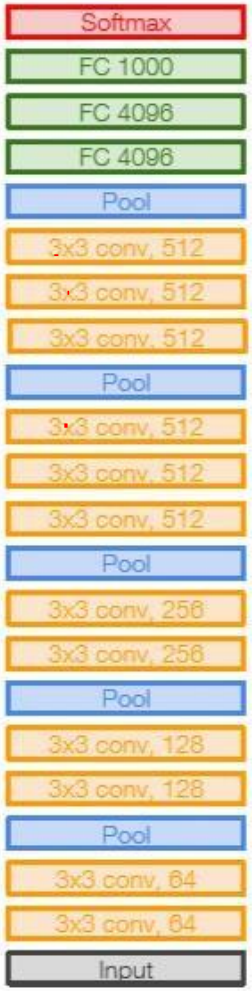


VGG16



VGG19

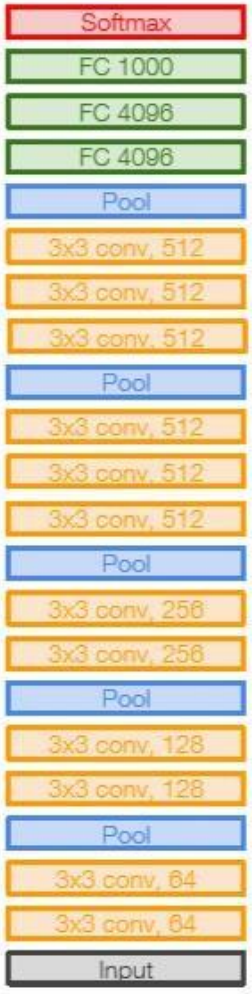
卷积神经网络典型结构 – VGG



VGG16

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,45
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

卷积神经网络典型结构 – VGG



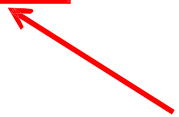
VGG16

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,45
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL params: 138M parameters，差不多是 AlexNet 的两倍

卷积神经网络典型结构 – VGG

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,45
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



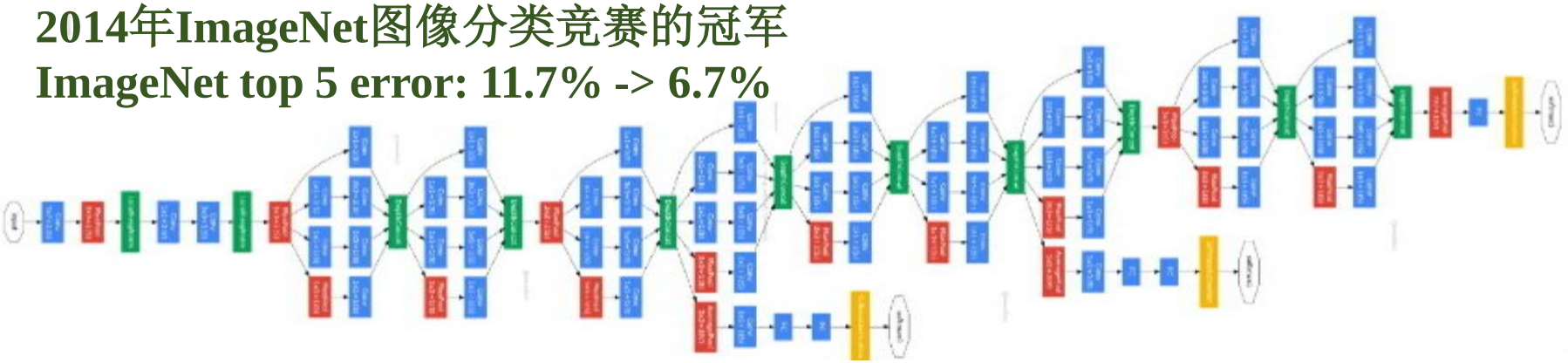
参数大多在全连接层中

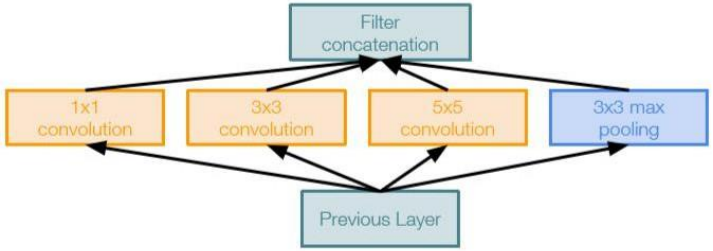
提 纲

- 绪论
  1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络
- 基本组成结构
  1. 卷积 2.池化 3.全连接
- 卷积神经网络典型结构
  1. AlexNet 2. ZFNet 3. VGG **4. GoogleNet** 5. ResNet
- 总结
  1. 参考文献 2. 代码 3. 作业

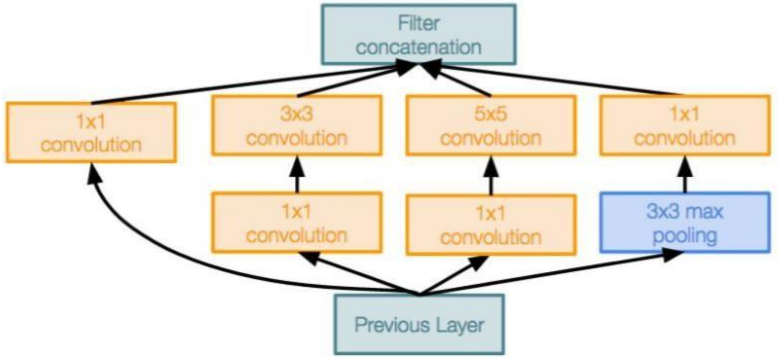
卷积神经网络典型结构 – GoogleNet

2014年ImageNet图像分类竞赛的冠军
ImageNet top 5 error: 11.7% -> 6.7%





原始inception模块

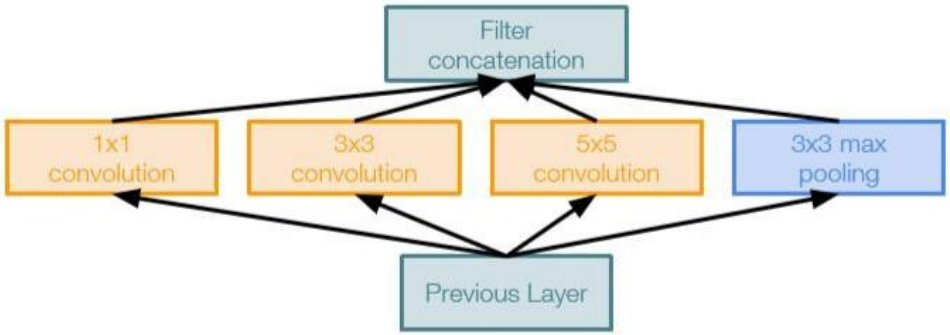


inception v2

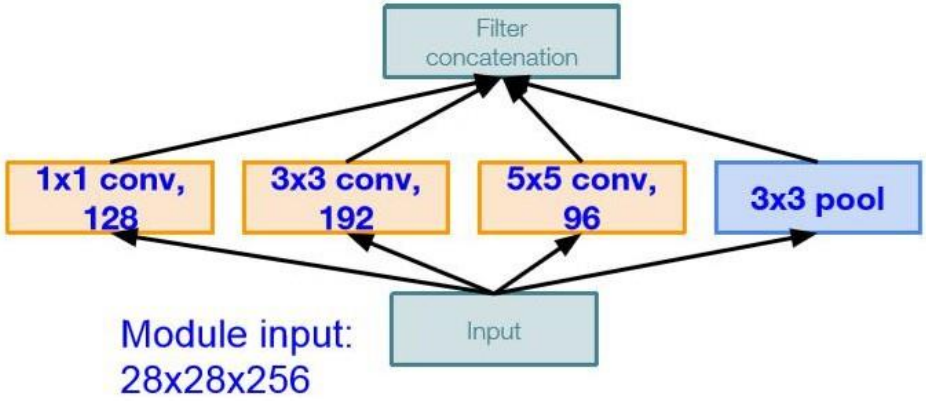
- 网络总体结构：
  - 网络包含22个带参数的层（如果考虑pooling层就是27层），独立成块的层总共约有100个；
  - 参数量大约是 AlexNet 的 1/12
  - 没有 FC 层

GoogleNet: Naive Inception

初衷：多卷积核增加特征多样性



原始inception模块



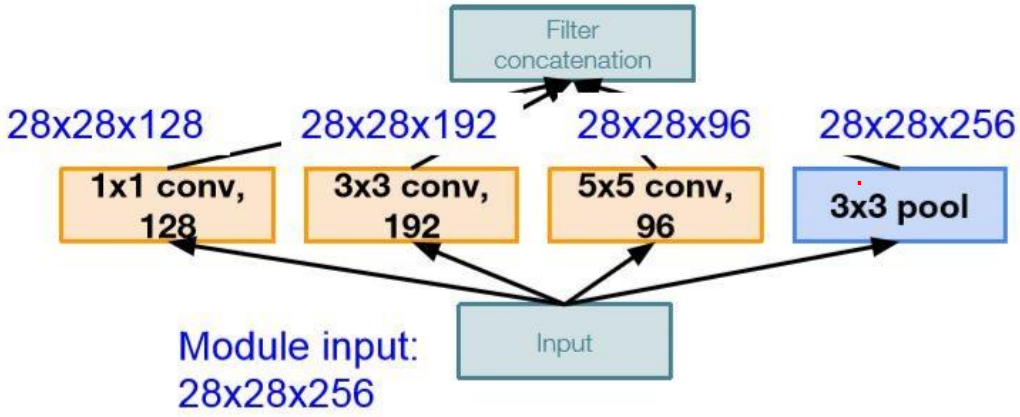
原始inception模块

输入：$28\times28\times256$

filter：128, 1×1

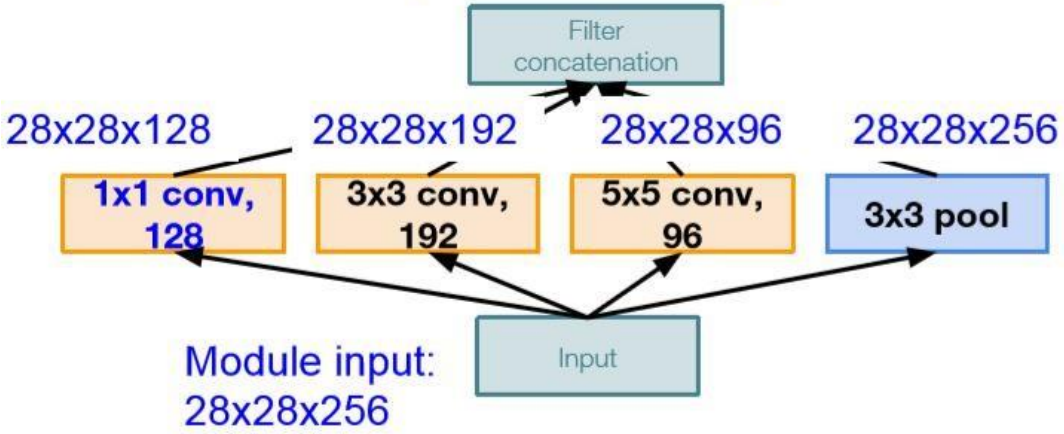
输出：$28\times28\times128$

GoogleNet: Naive Inception



Naive Inception module

28x28x(128+192+96+256) = 28x28x672



原始inception模块

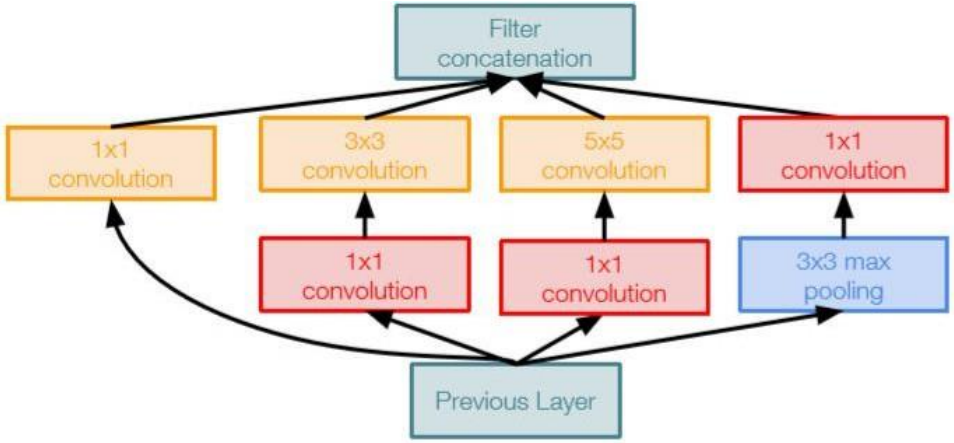
不同filter的输出?

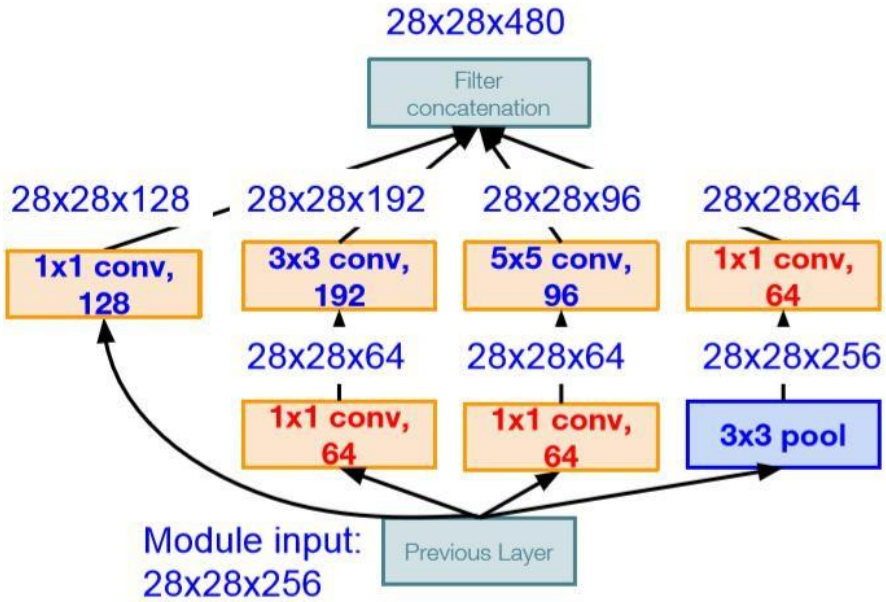
最终的输出?

问题?
计算复杂度过高

GoogleNet: Inception V2

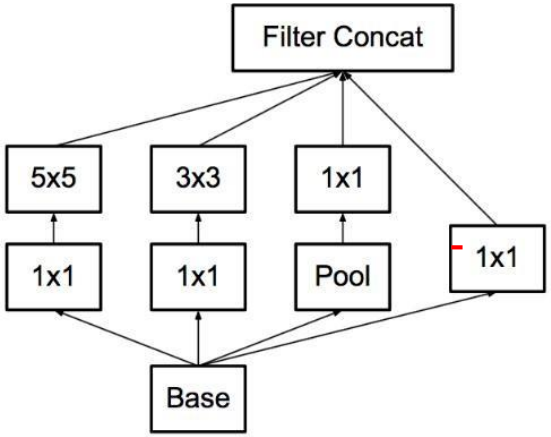
解决思路：插入1*1卷积核进行降维



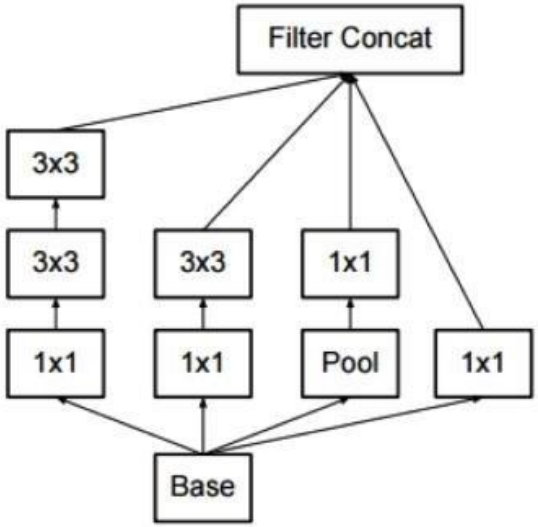


GoogleNet: Inception V3

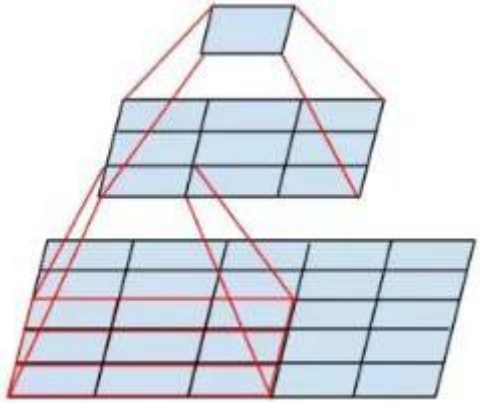
- **Inception V3** 进一步对 v2的参数数量进行降低



Inception V2

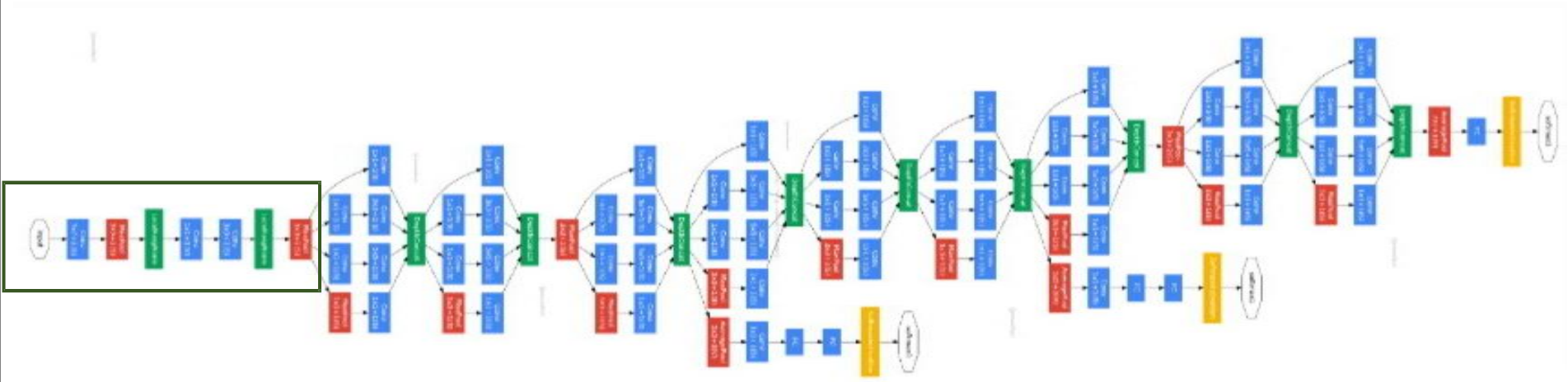


用小的卷积核替代大的卷积核



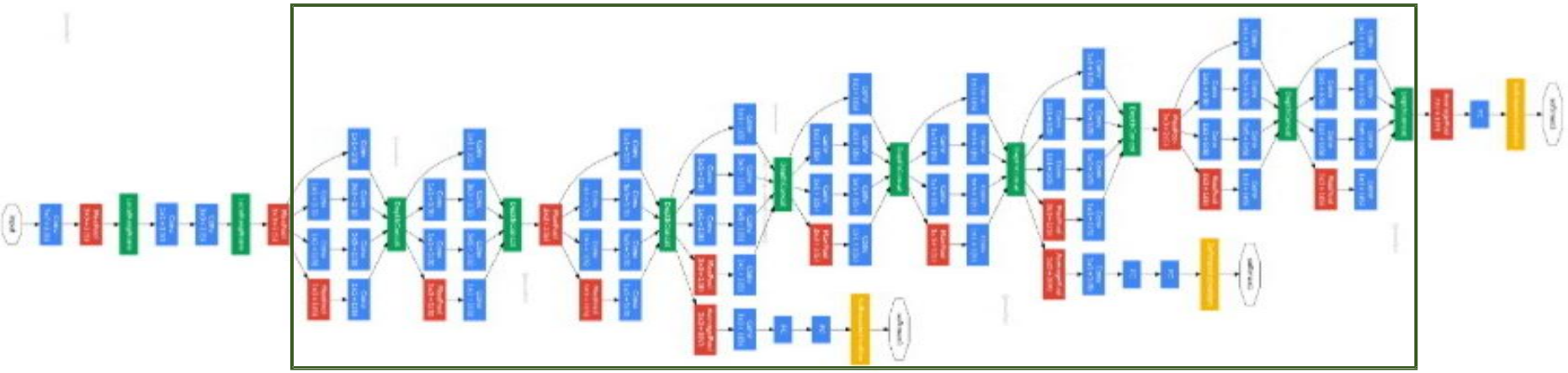
- 降低参数量
- 增加非线性激活函数：增加非线性激活函数使网络产生更多独立特(disentangled feature),表征能力更强，训练更快。

卷积神经网络典型结构 – GoogleNet



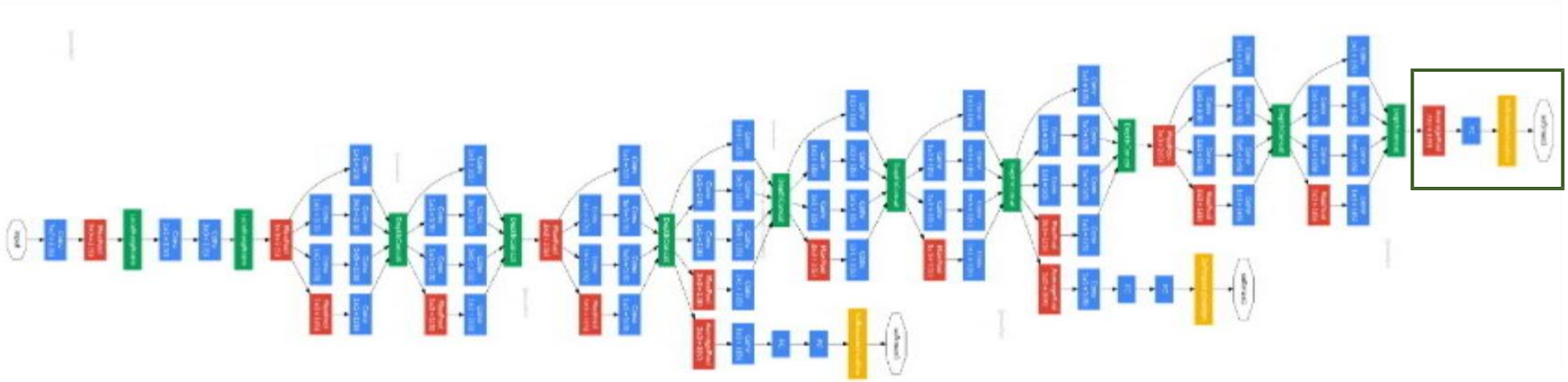
- Stem部分（stem network）：卷积 – 池化 – 卷积 –卷积 – 池化

卷积神经网络典型结构 – GoogleNet



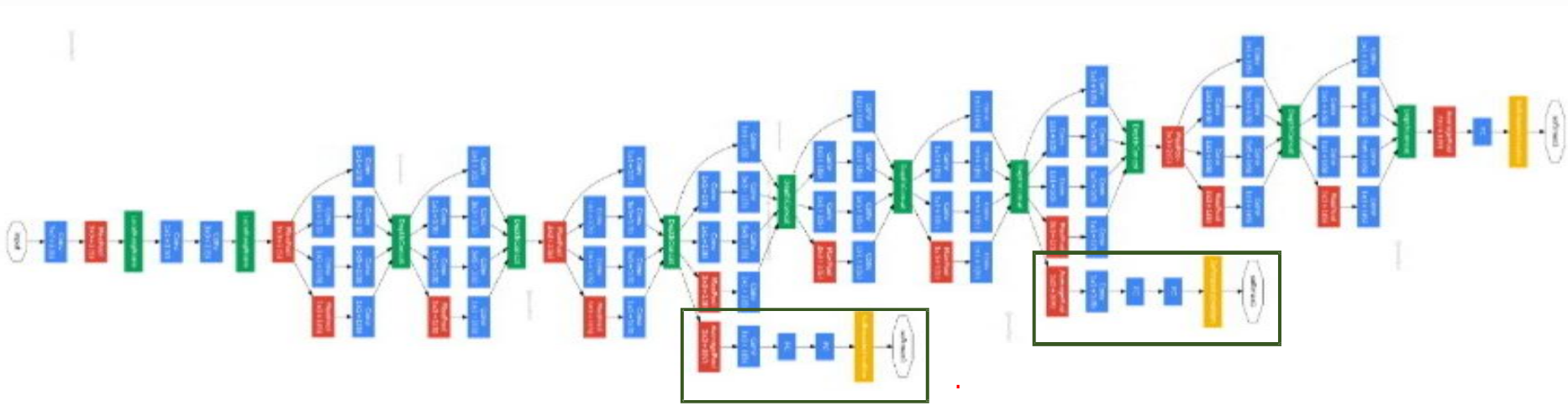
- 多个Inception结构堆叠

卷积神经网络典型结构 – GoogleNet



- 输出：没有额外的全连接层（除了最后的类别输出层）

卷积神经网络典型结构 – VGG



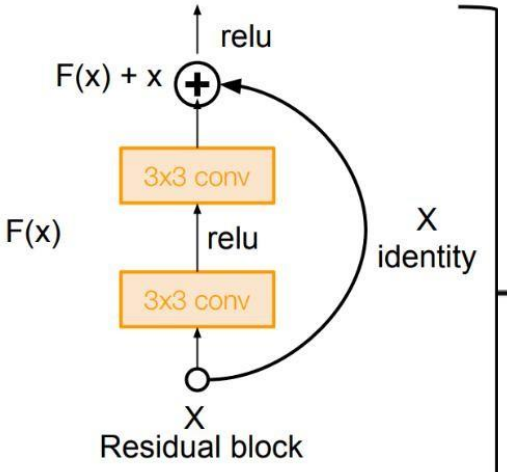
- 辅助分类器：解决由于模型深度过深导致的梯度消失的问题。
- 在V3以后去掉了

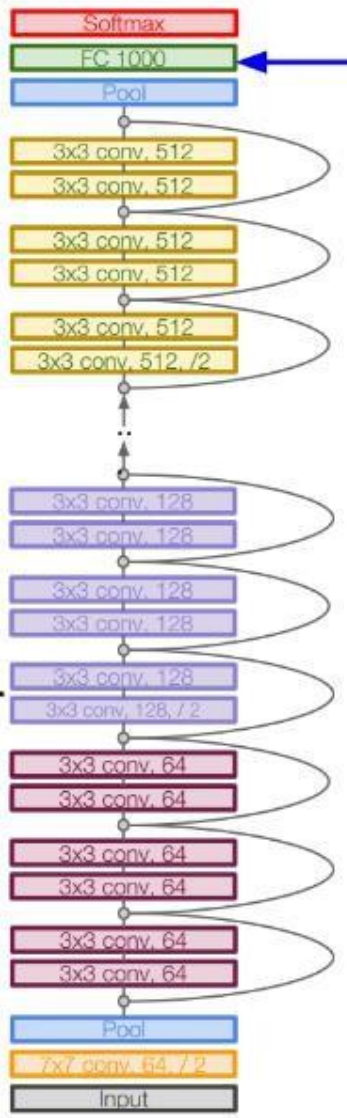
提纲

- 绪论
  1. 卷积神经网络的应用 2. 传统神经网络vs卷积神经网络
- 基本组成结构
  1. 卷积 2.池化 3.全连接
- 卷积神经网络典型结构
  1. AlexNet 2. ZFNet 3. VGG 4. GoogleNet **5. ResNet**

ResNet

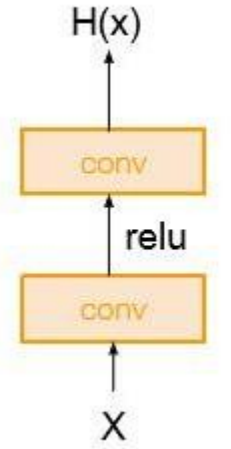
- 残差学习网络（deep residual learning network）
- 2015年ILSVRC竞赛冠军， top 5 错误率从6.7% -> 3.57%
- 深度有152层



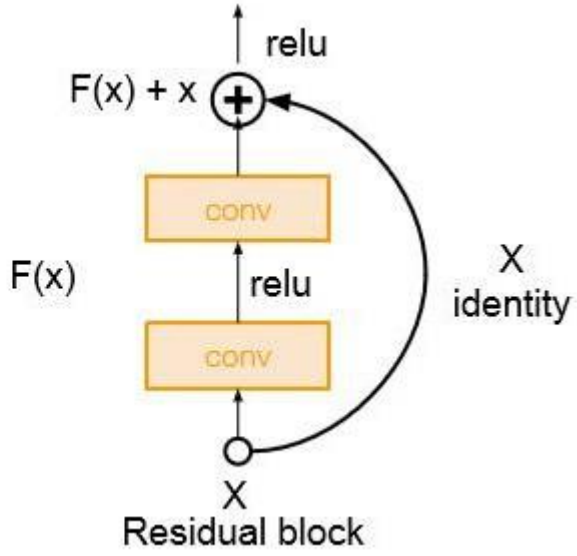


除了输出层之外没有其他全连接层。

ResNet



传统结构



- 残差的思想: 去掉相同的主体部分，从而突出微小的变化。
- 可以被用来训练非常深的网络

THANKS