

## 第八讲：卷积神经网络

Convolutional Neural Networks, LeNet, AlexNet, ZF-Net, VGGNet, GoogLeNet  
and ResNet

张盛平

s.zhang@hit.edu.cn

计算学部  
哈尔滨工业大学

2021 年秋季学期



# 卷积操作



$$s_t = \sum_{a=0}^{\infty} x_{t-a} w_{-a} = (x * w)_t$$

↑  
input      filter  
convolution

- 假定正在使用激光传感器在离散的时间间隔跟踪一架飞机的位置
- 激光传感器可能存在噪声
- 为了使得估计的位置包含尽量少的噪声，需要对多次的测量进行平均
- 最近的测量在平均时应该占更重要的作用，因此，需要计算赋权平均

$$s_t = \sum_{a=0}^6 x_{t-a} w_{-a}$$

- 实际中，只在一个小窗口中求和
- 权重数组 ( $w$ ) 也称为滤波器 (filter)
- 在输入上滑动滤波器，依据  $x_t$  上的滑动窗口来计算  $s_t$
- 这里，输入和滤波器均是 1 维的
- 卷积操作也能作用于 2 维

W

$w_{-6} \quad w_{-5} \quad w_{-4} \quad w_{-3} \quad w_{-2} \quad w_{-1} \quad w_0$

0.01	0.01	0.02	0.02	0.04	0.4	0.5
------	------	------	------	------	-----	-----

X

1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70
------	------	------	------	------	------	------	------	------	------	------	------

S

1.80	1.96	2.11	2.16	2.28	2.42
------	------	------	------	------	------

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$



- 一幅图像就是一个 2D 的输入
- 使用一个 2D 的滤波器 ( $m \times n$ )
- 2D 卷积的公式
- 这个公式在求和时，考虑的是 (i,j) 位置左上方的邻居
- 实际中，使用如下公式，求和时考虑 (i,j) 位置右下方的邻居

$$S_{ij} = (I * K)_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i+a, j+b} K_{a,b}$$

Input

a	b	c	d
e	f	g	h
i	j	k	$\ell$

Kernel

w	x
y	z

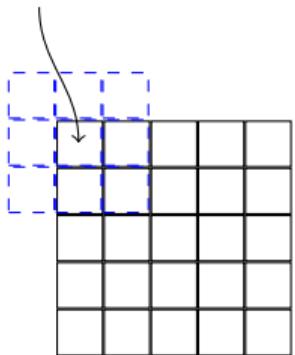
- 看左侧的例子

Output

$aw + bx + ey + fz$	$bw + cx + fy + gz$	$cw + dx + gy + hz$
$ew + fx + iy + jz$	$fw + gx + jy + kz$	$gw + hx + ky + \ell z$

$$S_{ij} = (I * K)_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a, j-b} K_{\frac{m}{2}+a, \frac{n}{2}+b}$$

pixel of interest



- 滑动窗口放置在当前位置的右下角仍然不方便
- 考虑将滑动窗口以当前位置为中心放置，如公式所示
- 因此，能看到当前位置四周的邻居



## 2D 卷积应用于图像的例子



$$\begin{matrix} & 1 & 1 & 1 \\ * & 1 & 1 & 1 \\ & 1 & 1 & 1 \end{matrix} =$$



blurs the image



$$\begin{matrix} & 0 & -1 & 0 \\ * & -1 & 5 & -1 \\ & 0 & -1 & 0 \end{matrix} =$$



sharpens the image



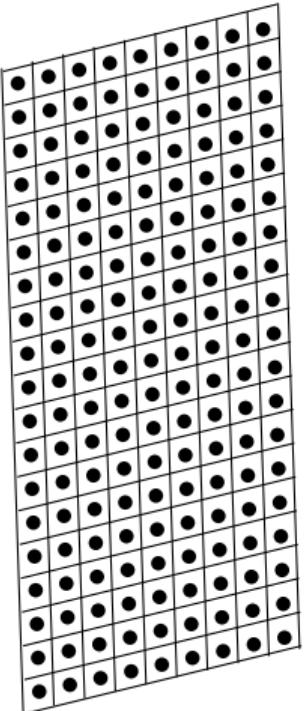
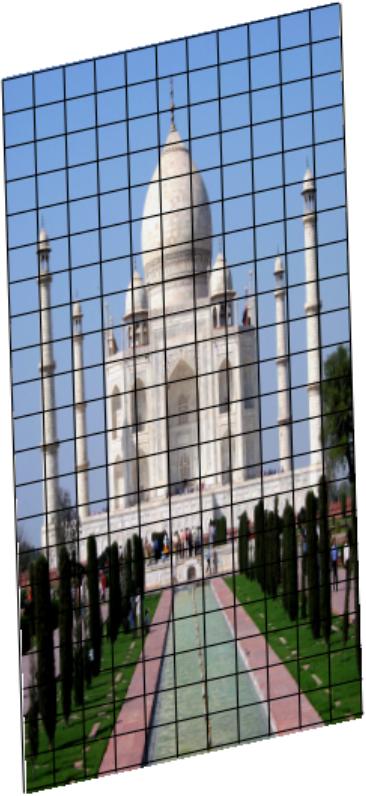
$$\begin{matrix} & 1 & 1 & 1 \\ * & 1 & -8 & 1 & = \\ & 1 & 1 & 1 \end{matrix}$$



detects the edges



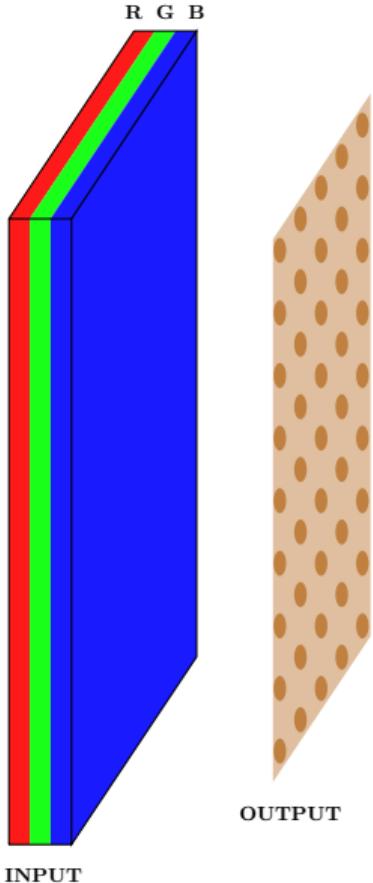
## 2D 卷积的具体实现



- 将滤波器沿着图像滑动
- 当滤波器停留在每一个位置时，能够计算一个加权平均值作为输出
- 所有得到的输出称为特征图（feature map）
- 当使用多个滤波器时，能得到多个特征图

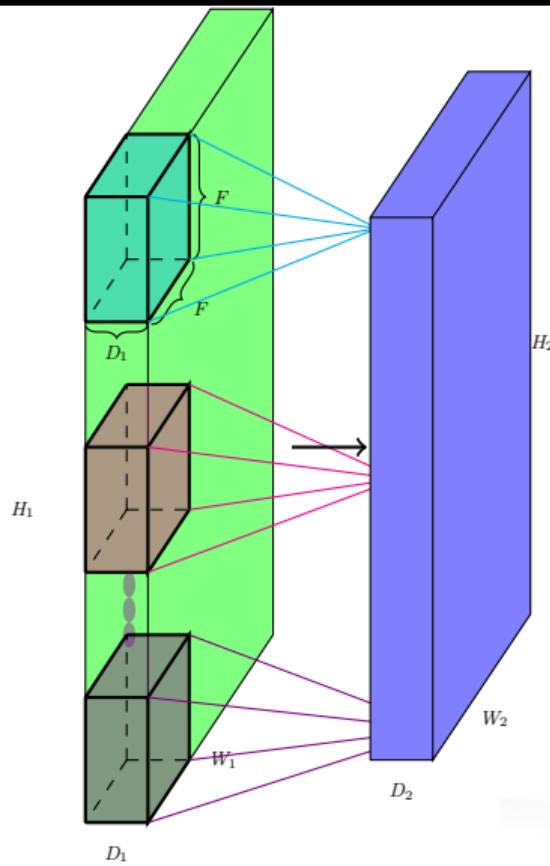
## Question

- 对于 1D 的输入，使用一个 1D 的滤波器在其上面滑动
- 对于 @D 的输入，使用一个 @D 的滤波器在其上面滑动
- 对于一个 3D 的输入呢？

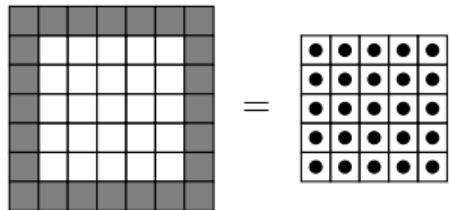


- 3D 的滤波器长什么样?
- 3D 的滤波器称为一个『体』(a volume)
- 在 3D 的输入里面滑动『体』来计算卷积结果
- 假设滤波器沿着图像深度方向的大小和图像深度一致
- 也就是说，当滤波器在 3D 输入中滑动时，实际上执行的是 2D 卷积
- 因此，输出的特征图也是 2D 的
- 当使用多个滤波器时，也能得到多个特征图

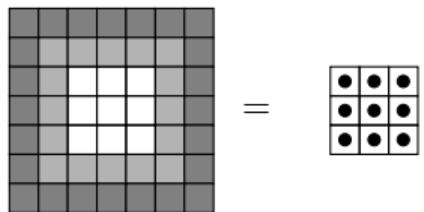
# 输入大小、输出大小和滤波器大小之间的关系



- 先定义一些变量
- 输入的宽度 ( $W_1$ ), 高度 ( $H_1$ ) and 深度 ( $D_1$ )
- 滑动步长  $S$
- 滤波器数量  $K$
- 滤波器的大小 ( $F$ ) (滤波器的深度和输入的深度一致)
- 输出的大小是  $W_2 \times H_2 \times D_2$



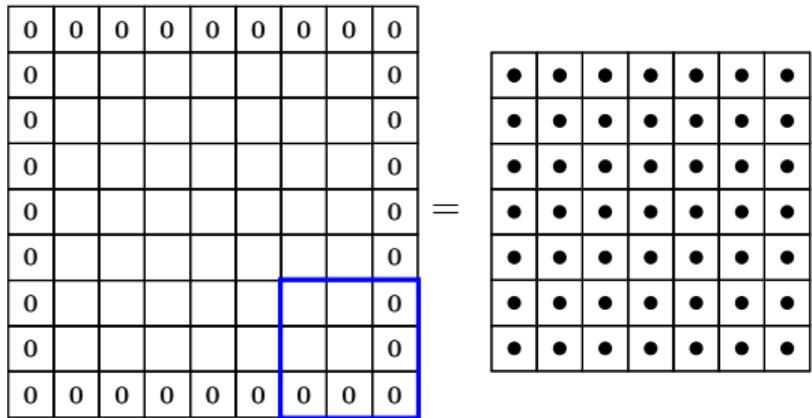
- 计算  $W_2, H_2$ )
- 不能把滤波器的中心放置于输入的边界上
- 如图中阴影的位置
- 这导致输出的维数比输入的要小



- 计算  $W_2, H_2$ )
- 不能把滤波器的中心放置于输入的边界上
- 如图中阴影的位置
- 这导致输出的维数比输入的要小
- 当滤波器的大小较大时, 输入上的更多位置不能进行卷积操作
- 例如, 考虑一个  $5 \times 5$  的滤波器
- 得到的输出更小

一般来说,  $W_2 = W_1 - F + 1$

$$H_2 = H_1 - F + 1$$

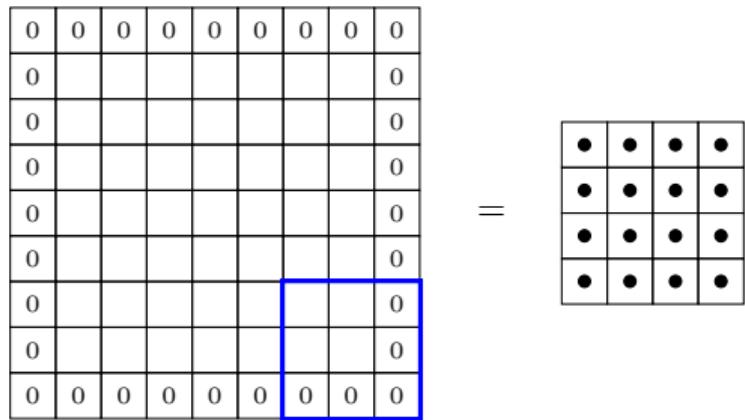


- 如何让输出与输入大小一样?
- 需要对输入进行填充 (padding)
- 通过对输入填充一些 0, 这样滤波器就能在输入的边界处进行卷积操作
- 对于  $3 \times 3$  的滤波器, 让  $P = 1$
- 在输入的最上行、最下行、最左列、最右列填充 0

因此,

$$W_2 = W_1 - F + 2P + 1$$

$$H_2 = H_1 - F + 2P + 1$$

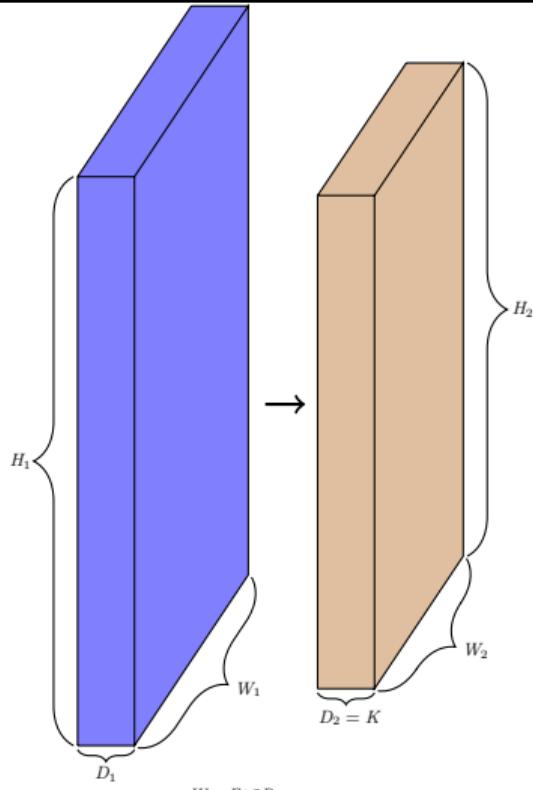


- 滑动步长 S 的作用?
- 它定义了滤波器在两次滑动之间的间隔 (here  $S = 2$ )
- 也会导致输出的维数比输入要小

最终的公式如下：

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$



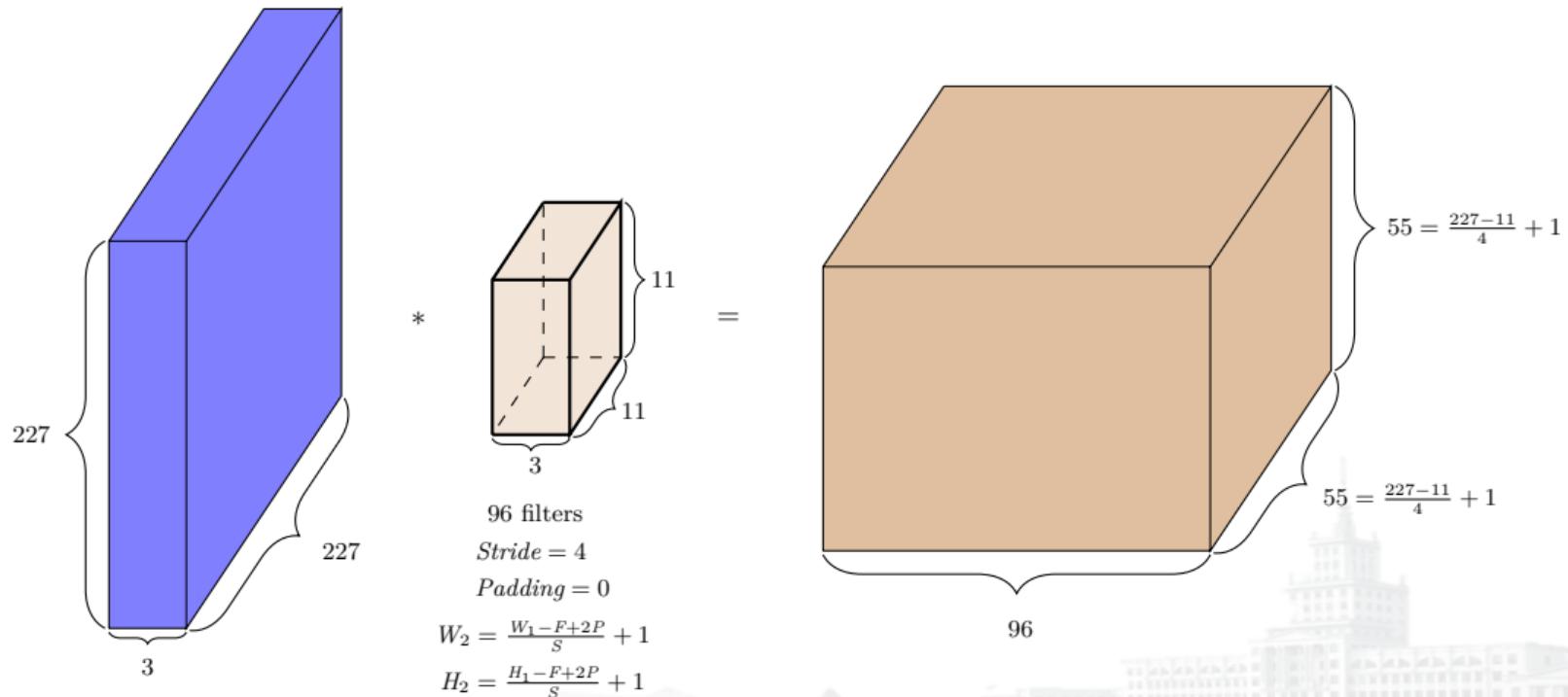
$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

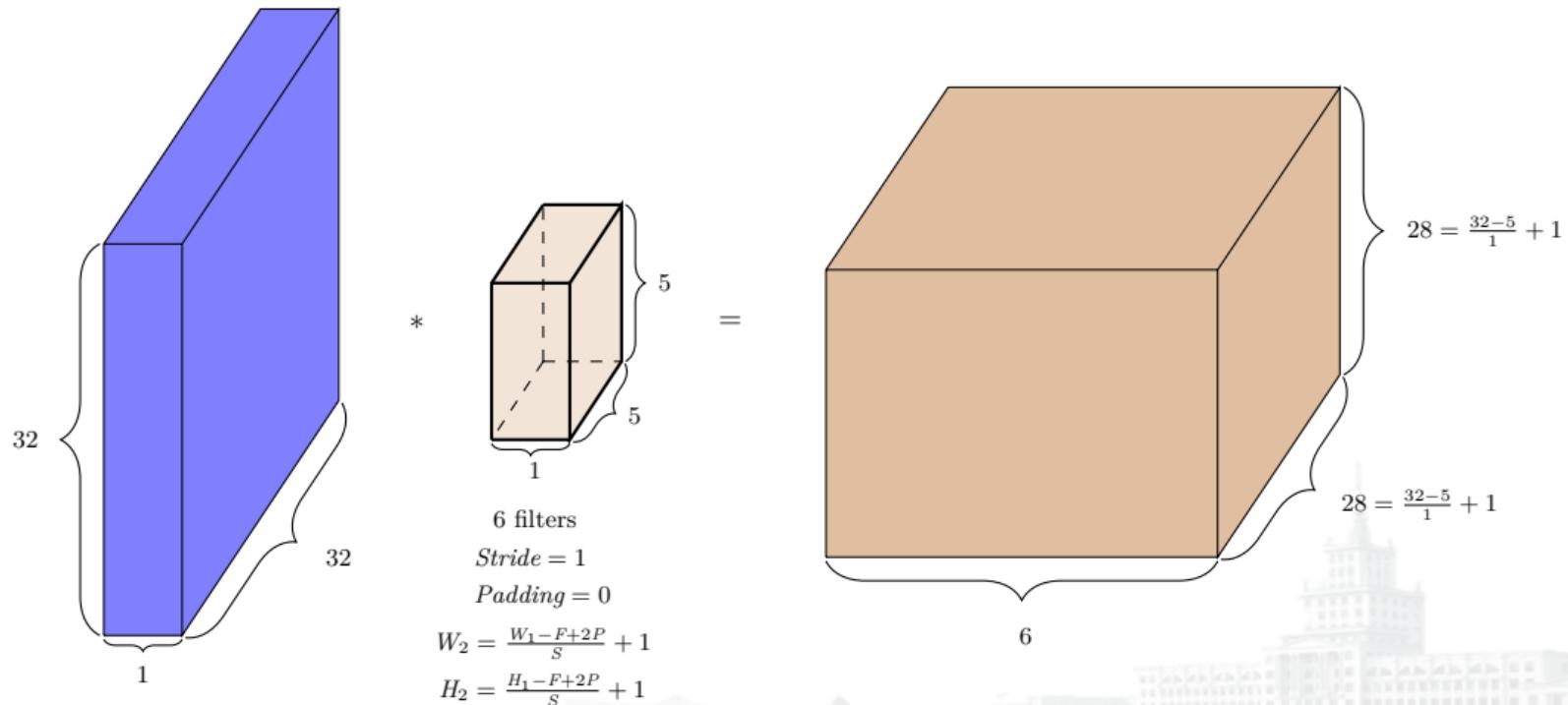
$$D_2 = K$$

- 每个滤波器产生一个 2D 的输出
- $K$  个滤波器能产生  $K$  个 2D 的输出
- 最终产生的输出是一个  $K \times W_2 \times H_2$  的 volume
- 因此  $D_2 = K$

一些例子



一些例子



# 卷积神经网络 (Convolutional Neural Networks)

## Putting things into perspective

- 卷积操作和神经网络有什么联系?
- 以图像分类任务为例



## Features



*Raw pixels*



→ car, bus, **monument**, flower



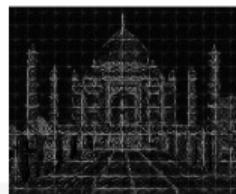
*Edge Detector*



→ car, bus, **monument**, flower



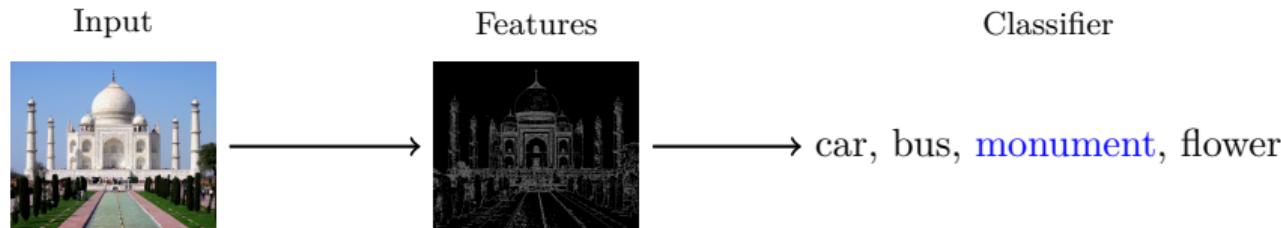
*SIFT/HOG*



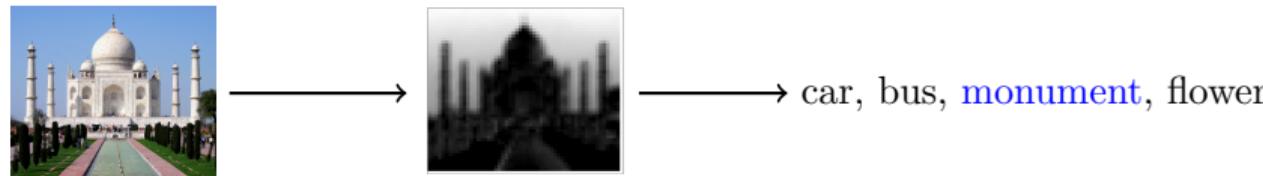
static feature extraction (no learning)

→ car, bus, **monument**, flower

learning weights of classifier

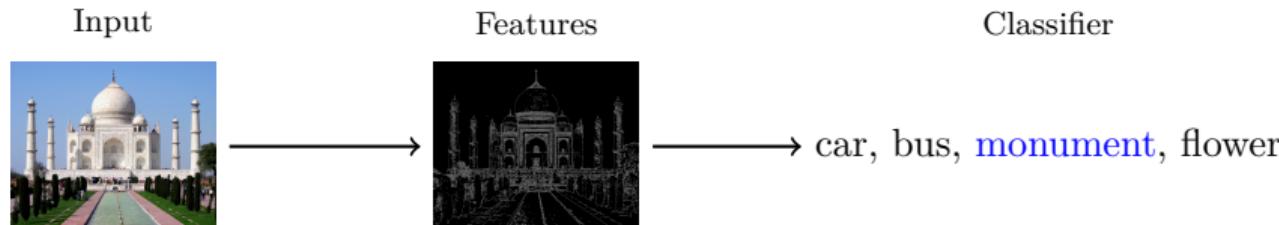


$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & -8 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$



$$\begin{matrix} 4.213568e-03 & 12.05358e-03 & \dots & \dots & -2.0601572e-02 \\ -4.277512e-03 & 2.013632e-03 & \dots & \dots & -1.092438e-02 \\ \vdots & \vdots & \vdots & \leftarrow \text{Learn these weights} & \vdots \\ 4.57090e-04 & -1.189207e-03 & \dots & \dots & 4.903527e-03 \end{matrix}$$

- 不使用 handcrafted 滤波器（例如边缘检测器），可以从数据中学习更有意义的滤波器

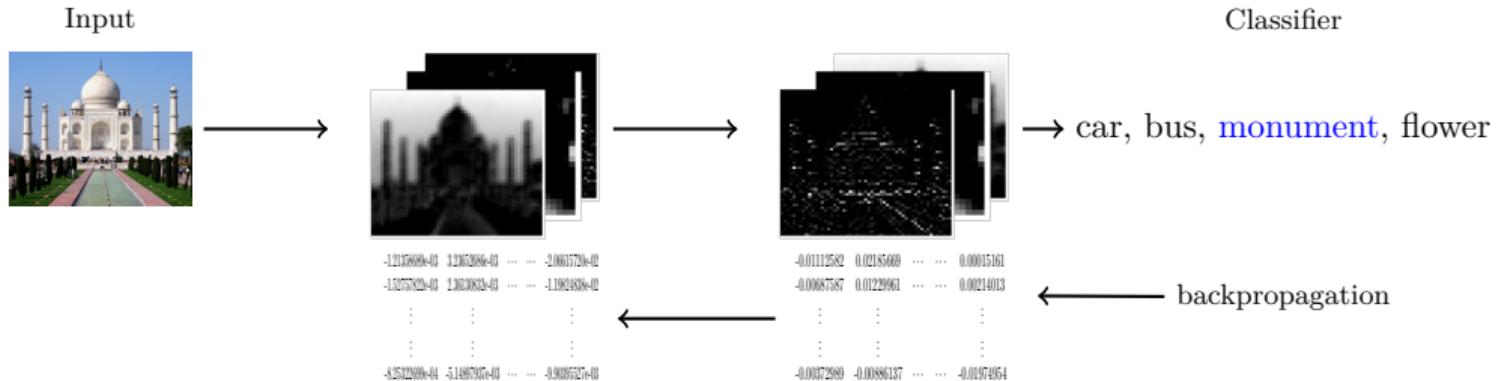


$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & -8 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$



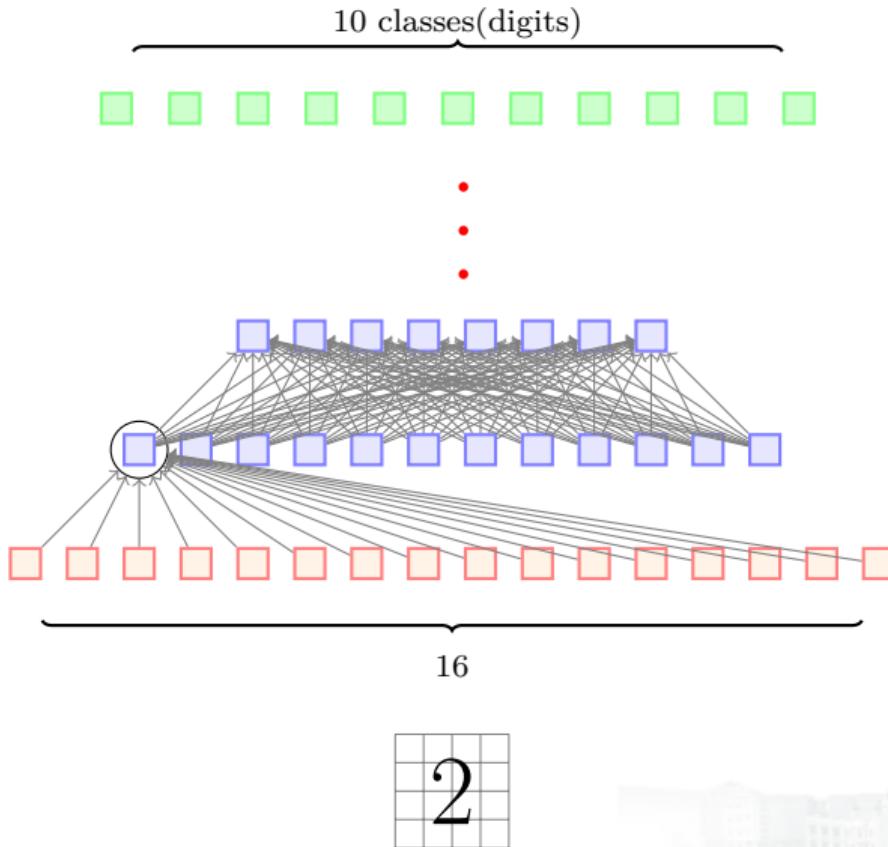
$$\begin{matrix} -0.01871333 & -0.01075948 & \dots & \dots & 0.04684572 \\ 0.00104325 & 0.01935937 & \dots & \dots & 0.01016542 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.03008777 & 0.00335217 & \dots & \dots & -0.02791128 \end{matrix}$$

- 不使用 handcrafted 滤波器（例如边缘检测器），可以从数据中学习 **多个** 有意义的滤波器

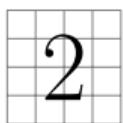


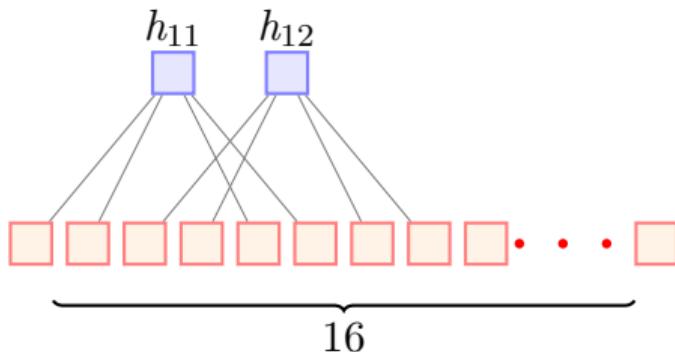
- 除了学习分类器权重外，能否学习多层 **layers** 有意义的滤波器？
- 将滤波器当作参数，和分类器权重一起学习
- 这样的神经网络称为卷积神经网络（Convolutional Neural Network）

卷积神经网络和前馈神经网络有什么区别?



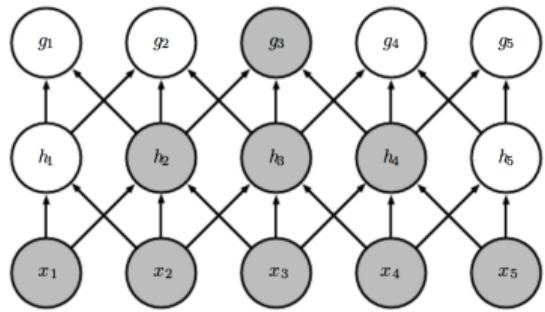
- 一个标准的前馈神经网络用于手写数字识别
- 稠密连接（全连接）
- 输入层的所有 16 个神经元与  $h_{11}$  相连，都参与  $h_{11}$  的计算
- 卷积神经网络有何不同？





$$\begin{matrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{matrix} \quad * \quad \begin{matrix} \bullet & \bullet \\ \bullet & \bullet \end{matrix} = h_{14}$$

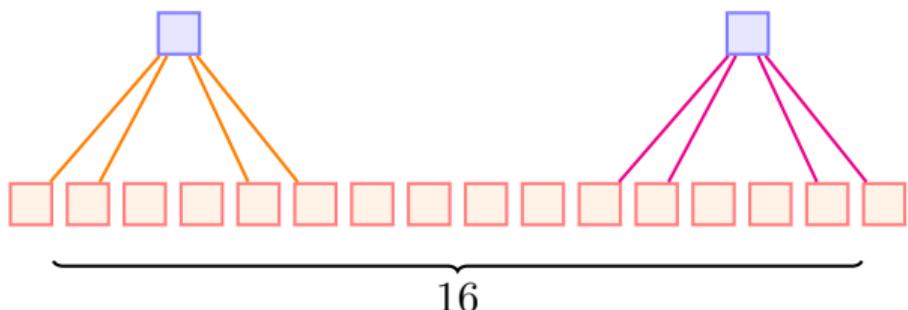
- 只有一小部分输入神经元参与  $h_{11}$  的计算
- 例如，只有像素 1, 2, 5, 6 参与  $h_{11}$  的计算
- 神经元之间的连接更稀疏
- 动机：以图像为例，在图像平面，空间相邻的像素才体现图像的结构信息
- 另外，这种 **稀疏连接**也大大减少了模型参数



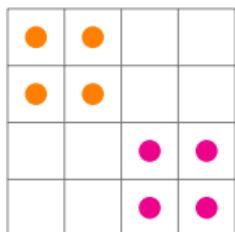
- 稀疏连接有时也有不足
- 会丢失输入中的一部分信息
- 两个神经元 ( $x_1$  &  $x_5$ )<sup>\*</sup> 在 *layer 1* 中没有交互
- 但它们在计算  $g_3$  时有间接交互
- 这也是为什么深度学习模型要深的原因之一

---

\* Goodfellow-et-al-2016

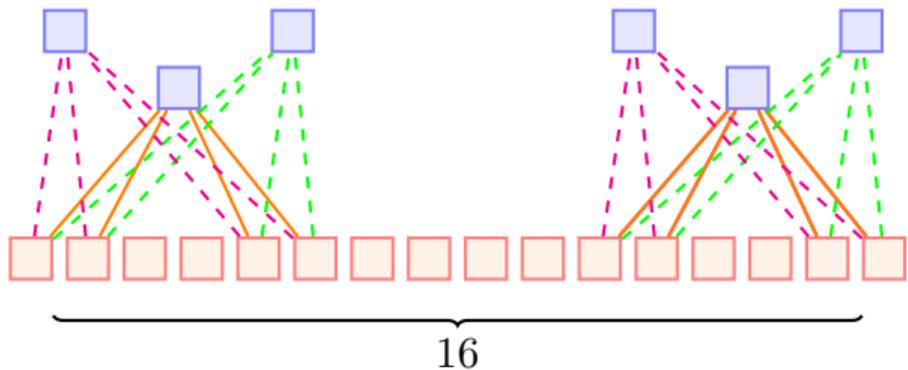


- Kernel 1
- Kernel 2



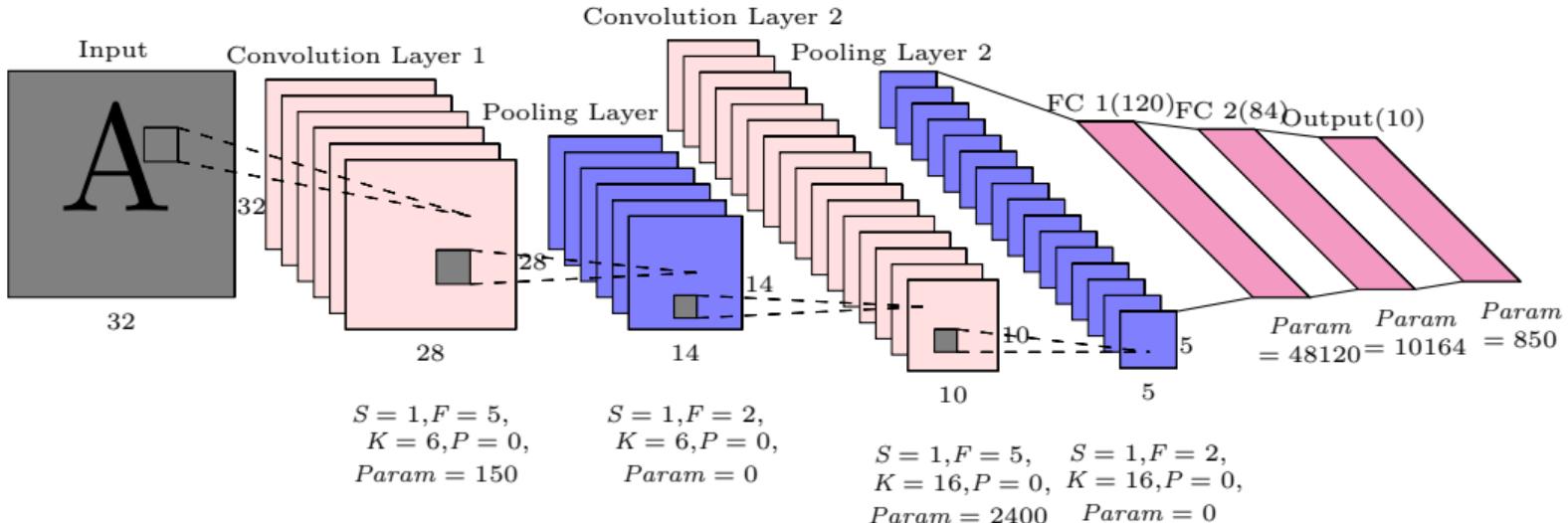
4x4 Image

- CNN 的另一个特性是权重共享 (weight sharing)
- 考虑左侧的网络
- 两个滤波器, 分别作用于图像的不同位置。两个滤波器的权重不一样
- 假设希望学习一个滤波器来检测图像中的边缘
- 因为图像中的每伸位置都有可能出现边缘, 这个滤波器应该作用于图像的所有位置

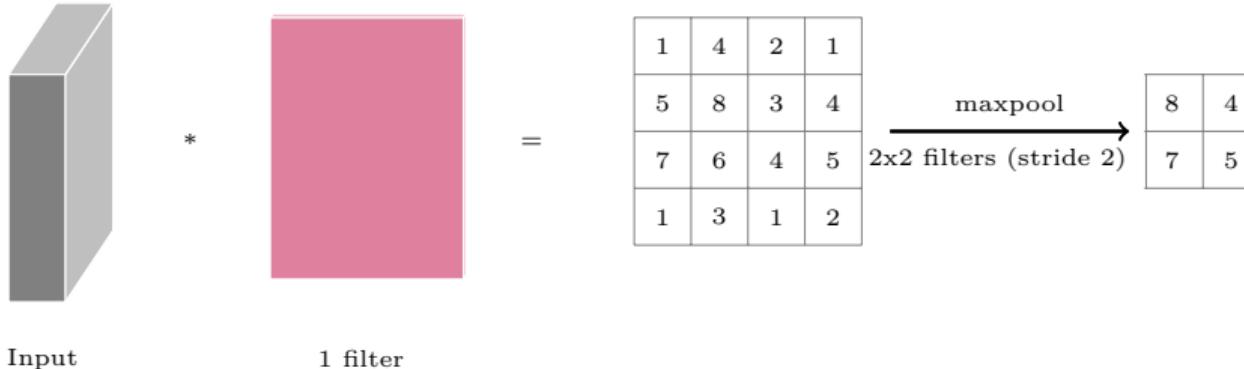


- 因此滤波器 *orange* 和 *pink* 应该是一样的
- 这样，一个滤波器可以在整张图像上进行操作
- 这也会使得学习滤波器变得更容易
- 这并不意味着只能使用一个滤波器
- 可以使用多个滤波器，每个滤波器的权重在图像的所有位置共享
- 这称为权重共享 “weight sharing”

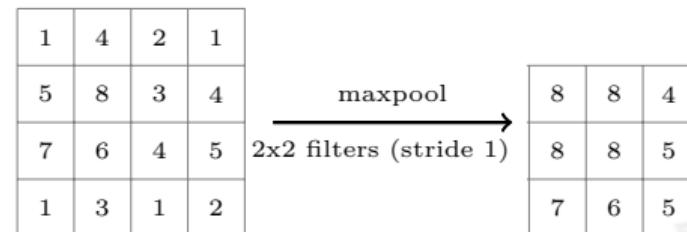
- 到目前为止，只是讨论了卷积操作
- 看一个完整的卷积神经网络



- 除了卷积层外，还有池化（pooling）层
- 池化层的作用是？



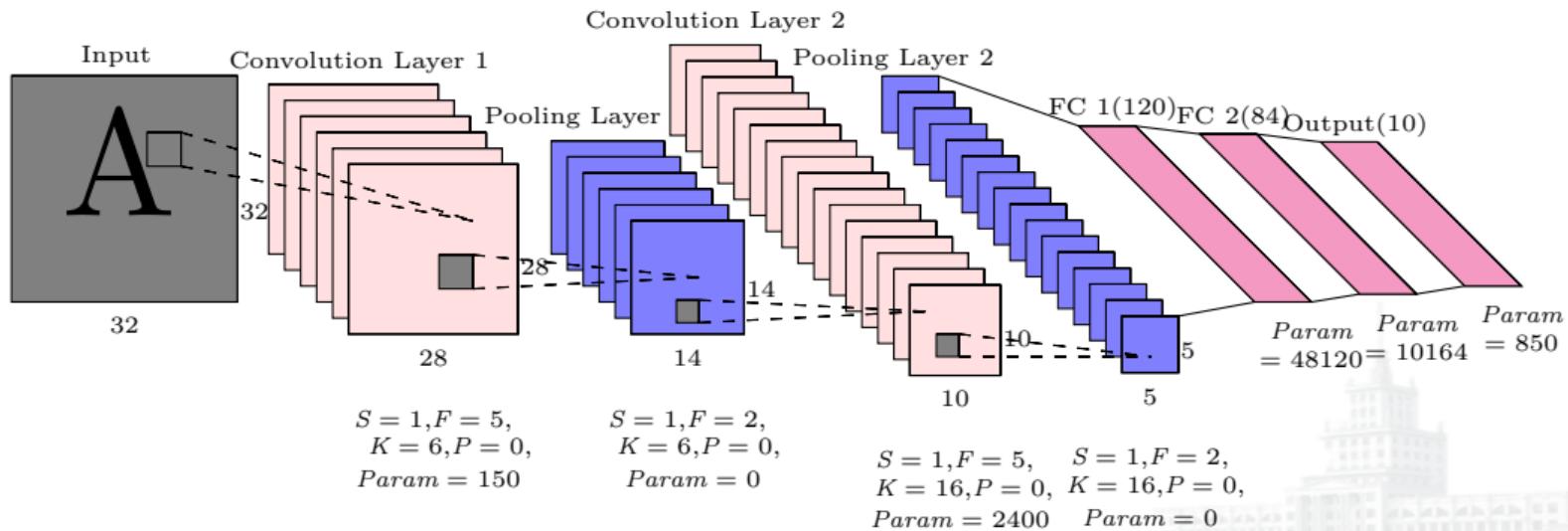
Input                  1 filter



- 除了最大池化，还有平均池化 (average pooling)

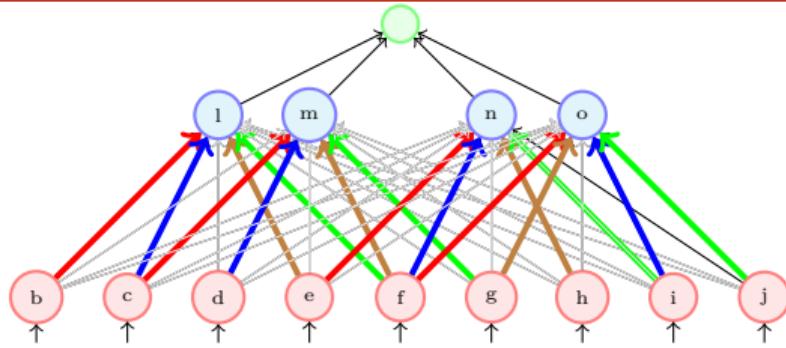
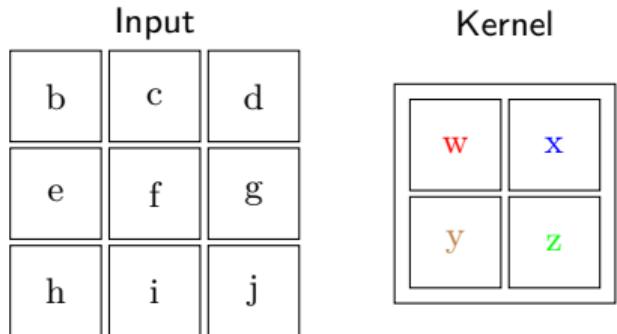
## 经典的卷积神经网络

## LeNet-5 for handwritten character recognition





- 如何训练 CNN?



- CNN 可以被实现成一个前馈神经网络
- 少部分权重（彩色）被激活
- 剩下的权重（灰色）是 0

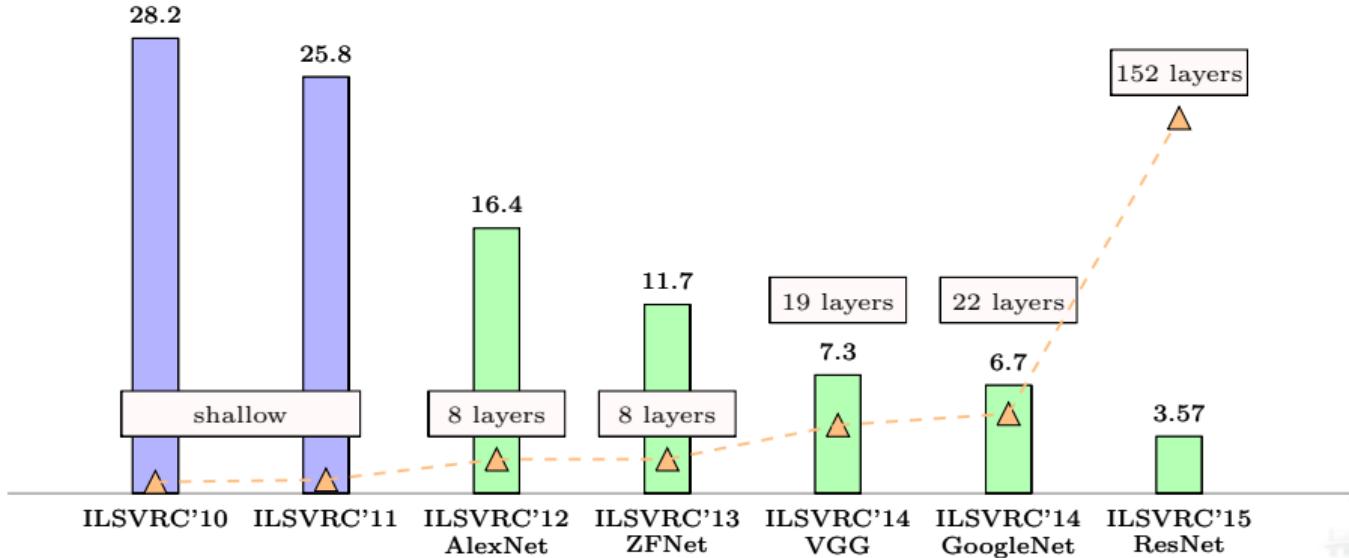
- 可以使用反向传播来训练 CNN

# CNNs (success stories on ImageNet)



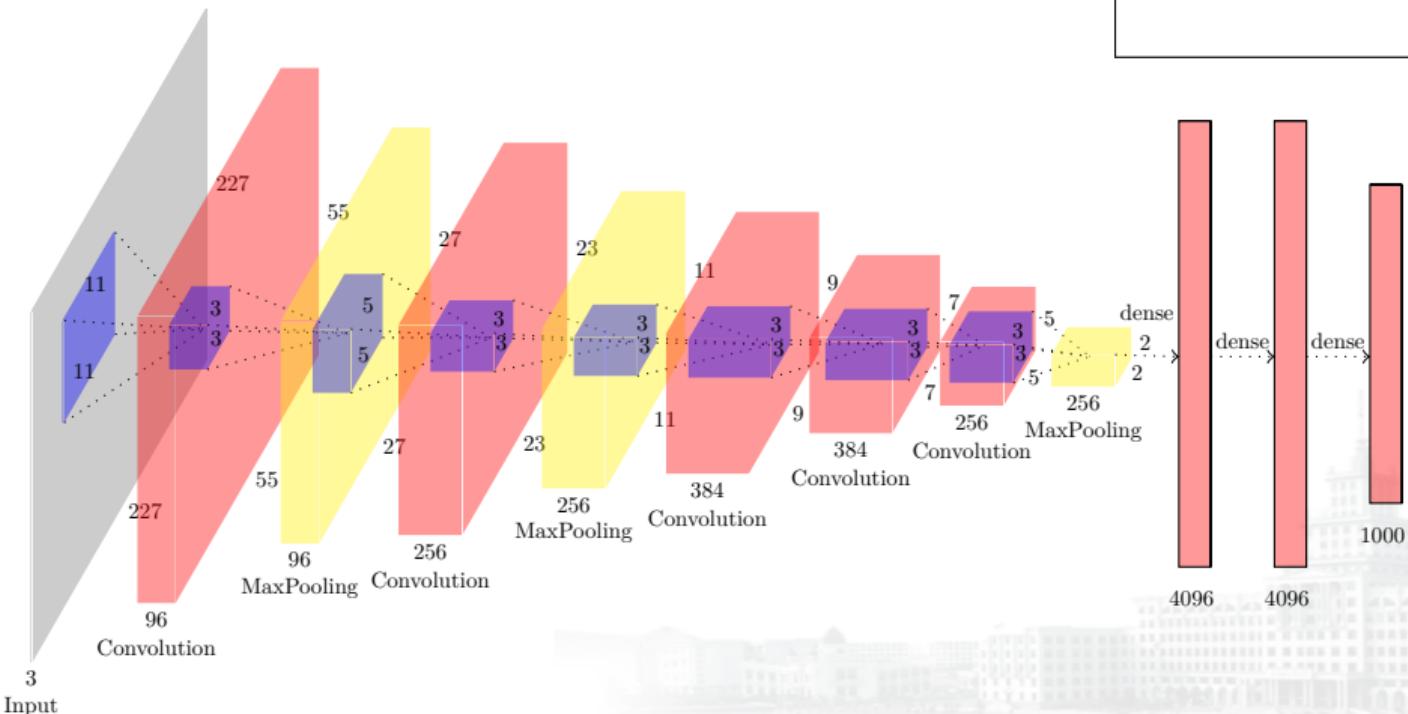
## ImageNet 比赛上的冠军模型

- AlexNet
- ZFNet
- VGGNet

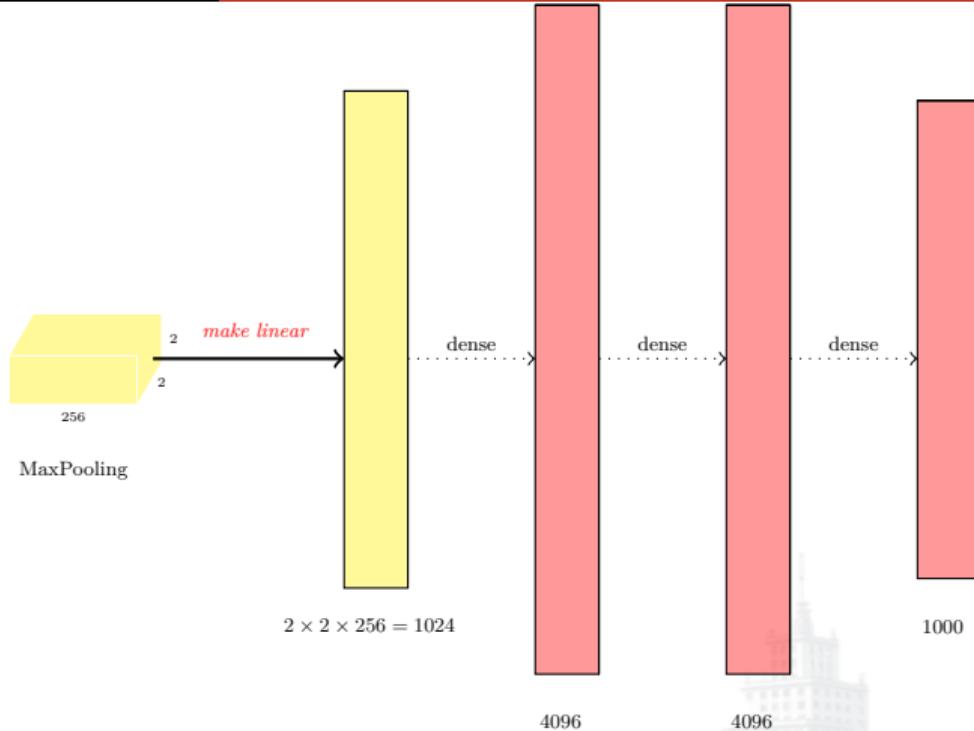


## ImageNet 比赛上的冠军模型

- AlexNet
- ZFNet
- VGGNet

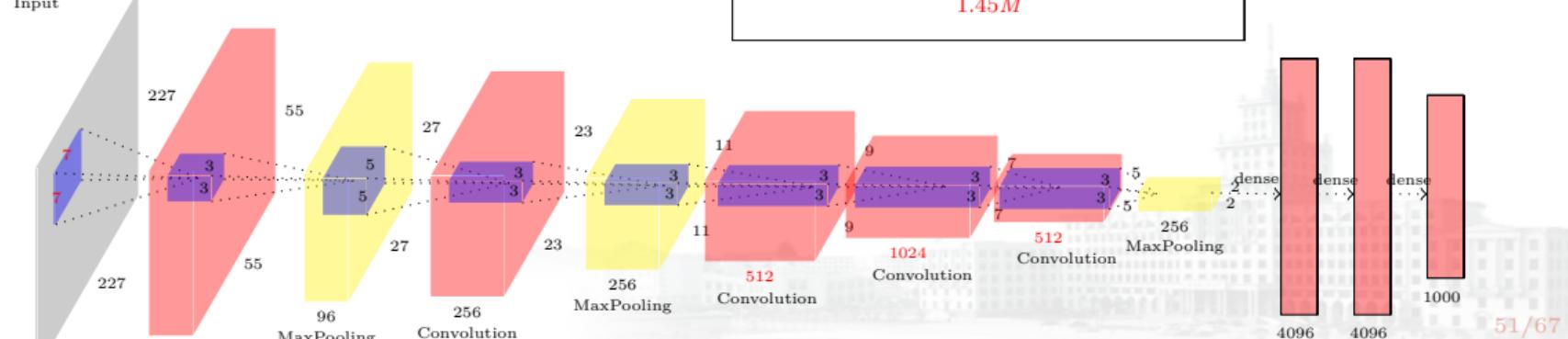
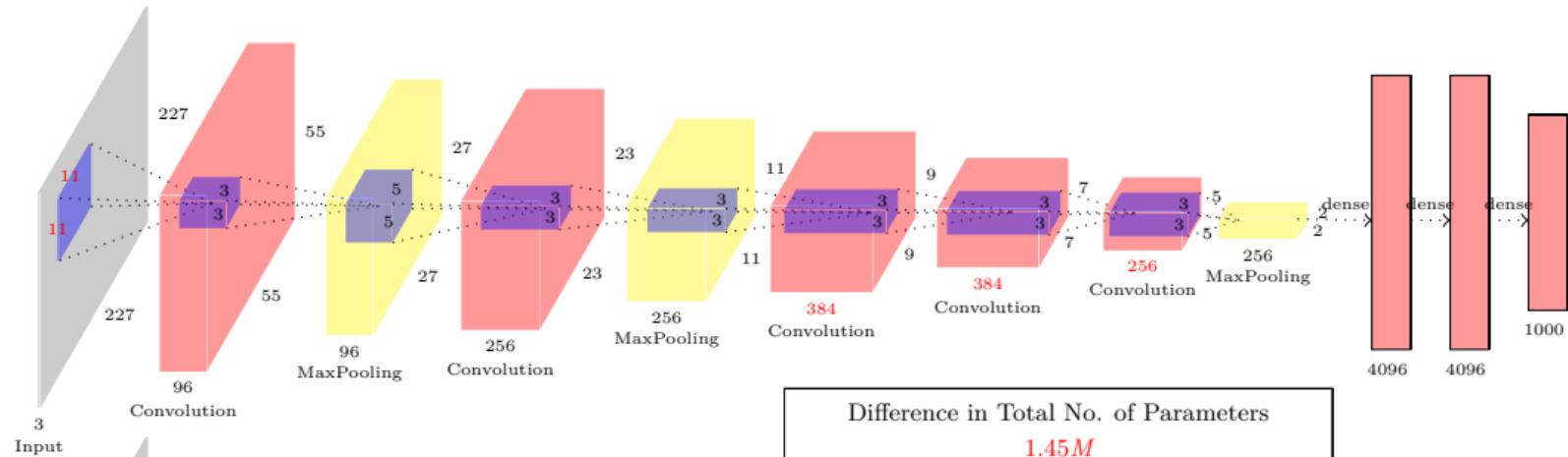


- 全连接层的具体实现
- 将上一卷积或池化层的输出展开成一个 1D 的向量
- 所得到的 1D 被全连接到下一层的神经元



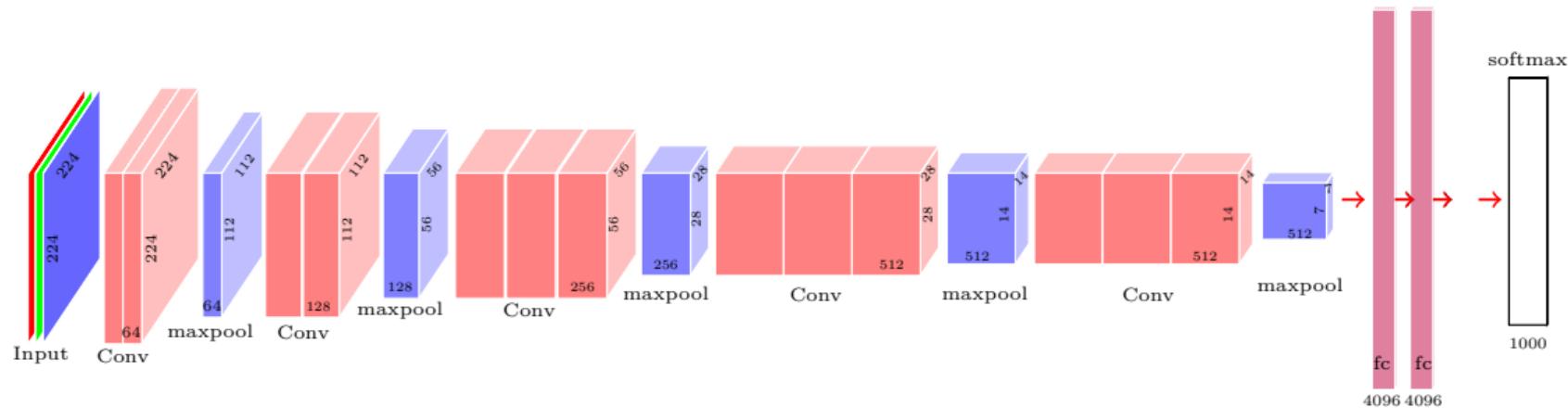
## ImageNet 比赛上的冠军模型

- AlexNet
- ZFNet
- VGGNet



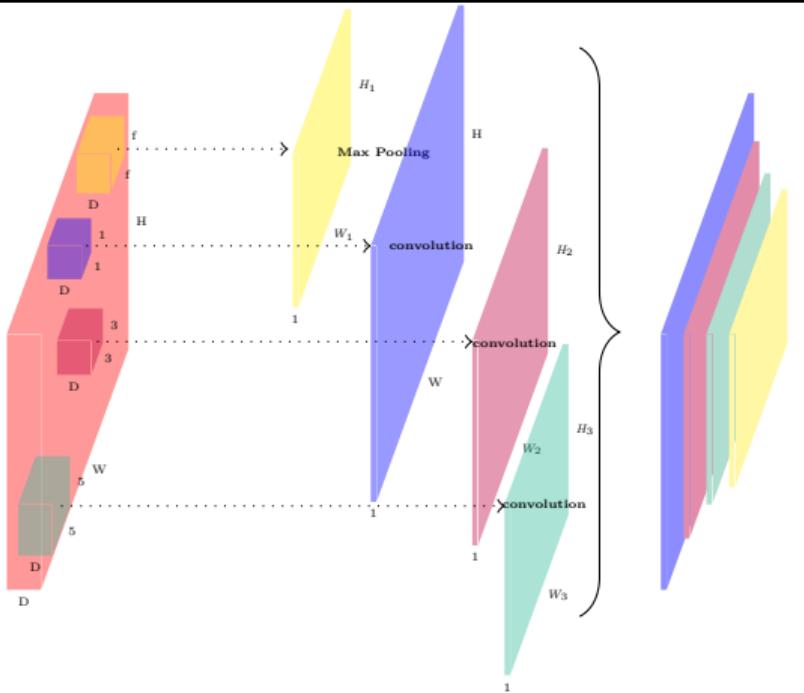
## ImageNet 比赛上的冠军模型

- AlexNet
- ZFNet
- VGGNet

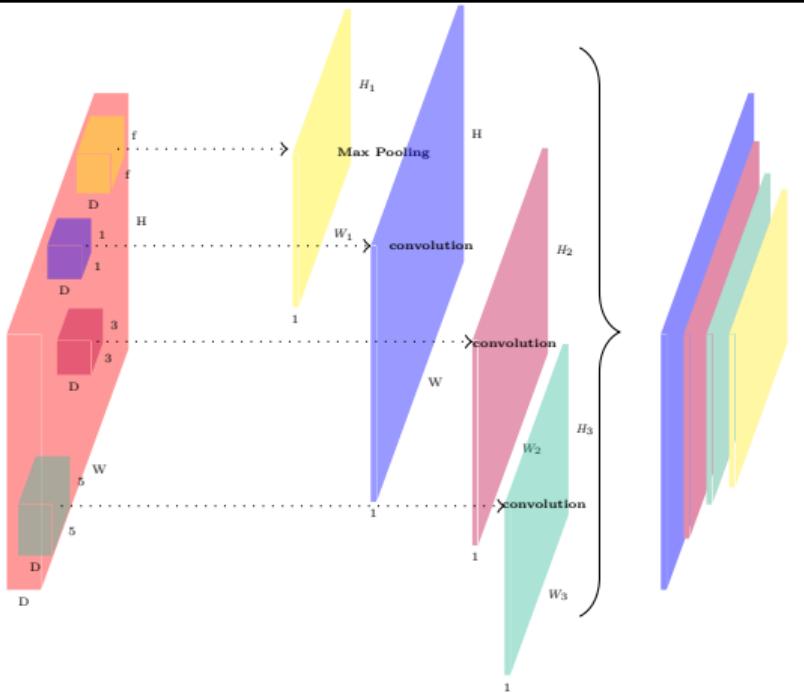


- 濾波器大小是  $3 \times 3$
- 非全连接层的参数量是  $\sim 16M$
- 全连接层的所有参数量 =  $(512 \times 7 \times 7 \times 4096) + (4096 \times 4096) + (4096 \times 1024) = \sim 122M$
- 大多数参数都在第一个全连接层 ( $\sim 102M$ )

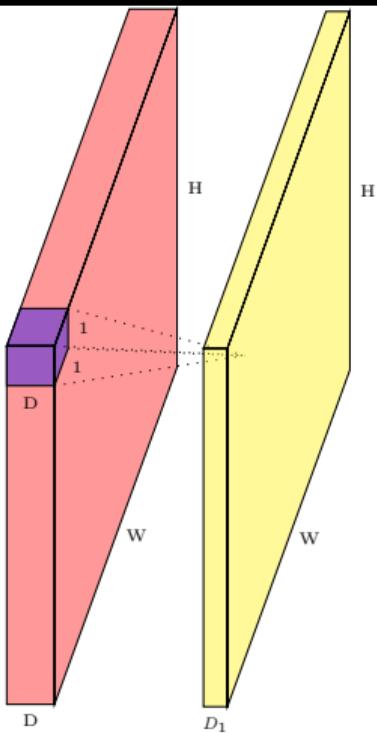
# Image Classification continued (GoogLeNet and ResNet)



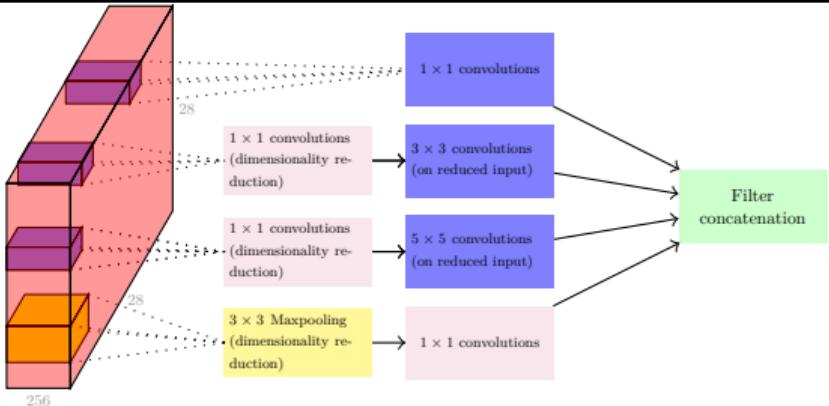
- 考虑 CNN 某一层的输出
- 在这一层的后面，可能使用一个最大池化层
- 或者是一个  $1 \times 1$  卷积
- 或者是一个  $3 \times 3$  卷积
- 或者是一个  $5 \times 5$  卷积
- **问题:** 为什么只是从这几个选项中选择一个? Why choose between these options (convolution, maxpooling, filter sizes)?
- **想法:** 为什么不同时使用上面的所有选项，再把它们的特征图串联起来



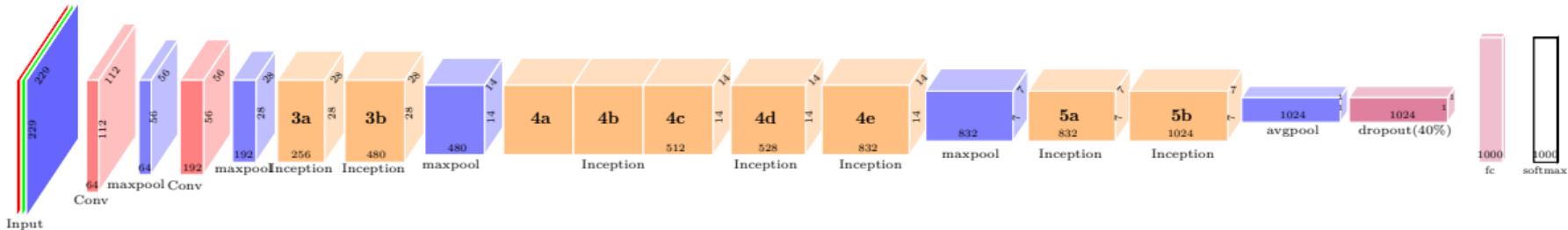
- 这个想法，会导致大量的计算
- 如果  $P = 0 \& S = 1$ , 使用一个  $F \times F \times D$  的滤波器来卷积一个  $W \times H \times D$  的输入, 得到一个  $(W - F + 1)(H - F + 1)$  的输出
- 输出中的每一个元素需要  $O(F \times F \times D)$  次计算
- 是否能降低计算量?

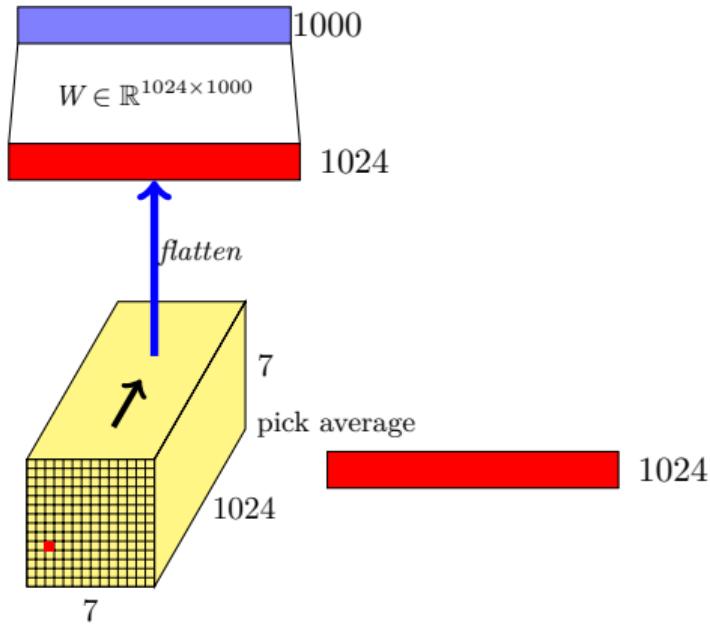


- 可以, 使用  $1 \times 1$  卷积
- $1 \times 1$  卷积能做什么?
- 沿着深度方向对特征进行聚集
- 使用  $D_1$  个  $1 \times 1$  ( $D_1 < D$ ) 的滤波器来卷积一个  $D \times W \times H$  的输入, 得到一个  $D_1 \times W \times H$  的输出 ( $S = 1, P = 0$ )
- 如果  $D_1 < D$ , 能够有效地降低输入的维数, 因此减少计算量
- 能将原来的计算量  $O(F \times F \times D)$  降低为  $O(F \times F \times D_1)$
- 接下来, 再使用  $3 \times 3$  和  $5 \times 5$  的滤波器



- 随后的  $3 \times 3$  和  $5 \times 5$  的滤波器都作用在同一个维数减少的输出可能不太好，最好是分开
- 因此，在  $3 \times 3$  和  $5 \times 5$  滤波器之前，分别使用  $D_1$  和  $D_2$  个  $1 \times 1$  的滤波器
- 在维数减少之前，增加一个最大池化层
- 新的  $1 \times 1$  卷积
- 最后，将所有这些层的输出串联
- 这称为 **Inception module**
- GoogLeNet** 包含很多这样的 inception modules

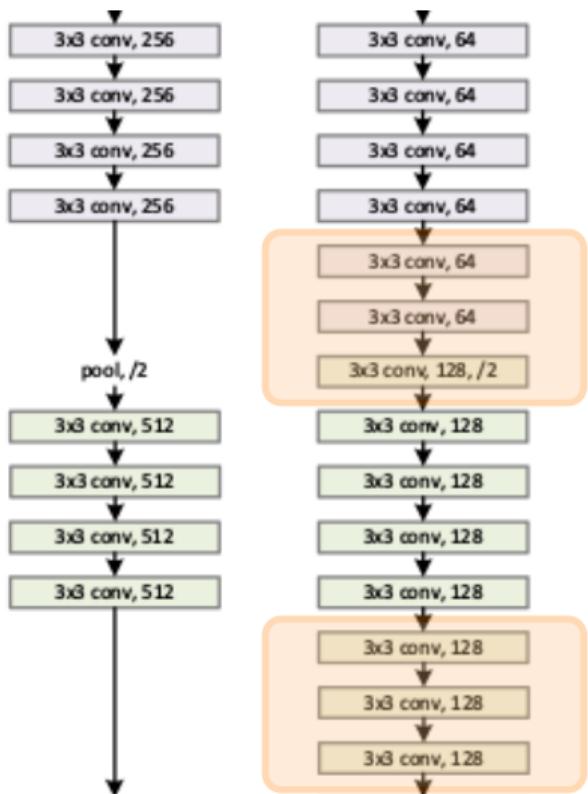




- 跟 AlexNet 相比, 参数个数减少为 12 分之一
- 计算量增加 2 倍



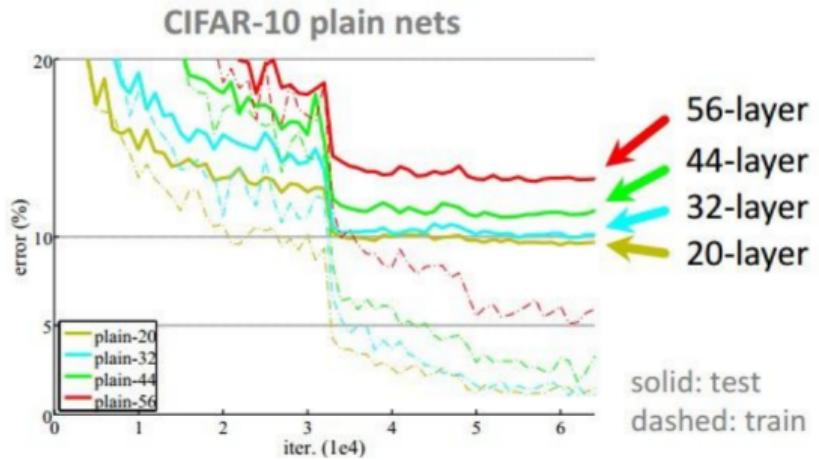
- GoogLeNet
- ResNet

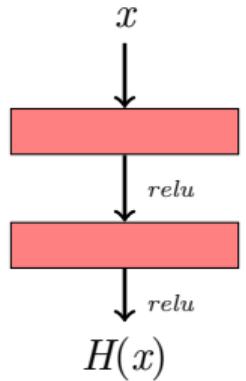


- 假定能够很好地学习一个浅层的神经网络
- 通过增加一些层 (in orange), 可以得到一个更深的网络
- 直观上来说, 如果浅层的网络表现好, 深层网络也会表现好 (哪怕简单地将新增加的层学习到的函数设为对等函数)
- 本质上说, 浅层神经网络的解空间是深层网络解空间的子集

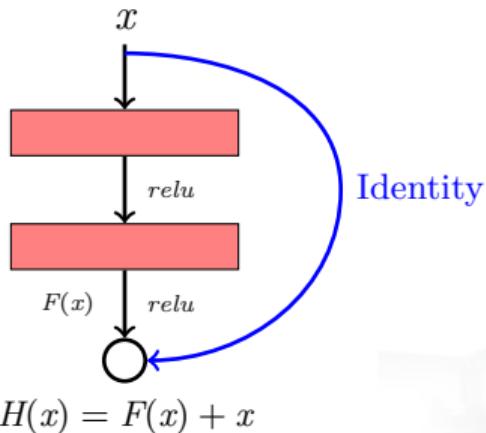


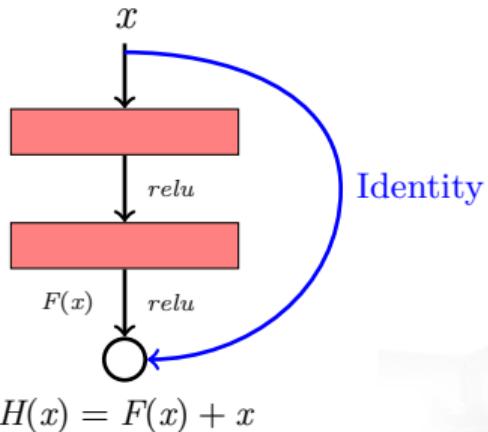
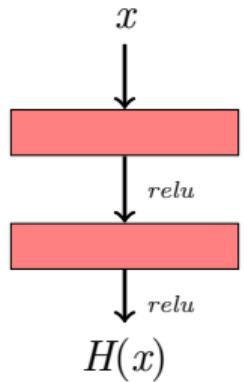
- 但是，在实际观察发现并不是这样
- 在测试集上，越深的层得到越高的错误率





- 考虑 CNN 的两个堆叠的层
- 它们本质上在学习输入的某个函数
- 学习输入的一个残差函数呢？





- 学习残差函数有什么用?
- 记住: 一个浅层网络, 通过增加一些层得到一个更深的网络, 更深的网络会表现的不比浅层网络差, 最坏情况也是在新增加的层上学习相等变换
- 这种相等变换允许 ResNet 得到输入的一份复制
- 使用这个地, 能训练非常深的网络



ResNet, 152 layers

1<sup>st</sup> place in all five main tracks

- **ImageNet Classification:** “Ultra-deep” 152-layer nets
- **ImageNet Detection:** 16% better than the 2nd best system
- **ImageNet Localization:** 27% better than the 2nd best system
- **COCO Detection:** 11% better than the 2nd best system
- **COCO Segmentation:** 12% better than the 2nd best system



## 训练技巧

- 每个卷积层后进行批归一化
- Xavier/2 initialization from [He et al]
- SGD + Momentum (0.9)
- Learning rate:0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

ResNet, 152 layers