教育部产学合作协同育人项目资助
2022年北京交通大学《深度学习》课程

第七讲 卷积神经网络II

主讲教师：丛润民

目录

7.1 参数学习

7.2 其他卷积方式：转置、空洞

7.3 典型网络简介：

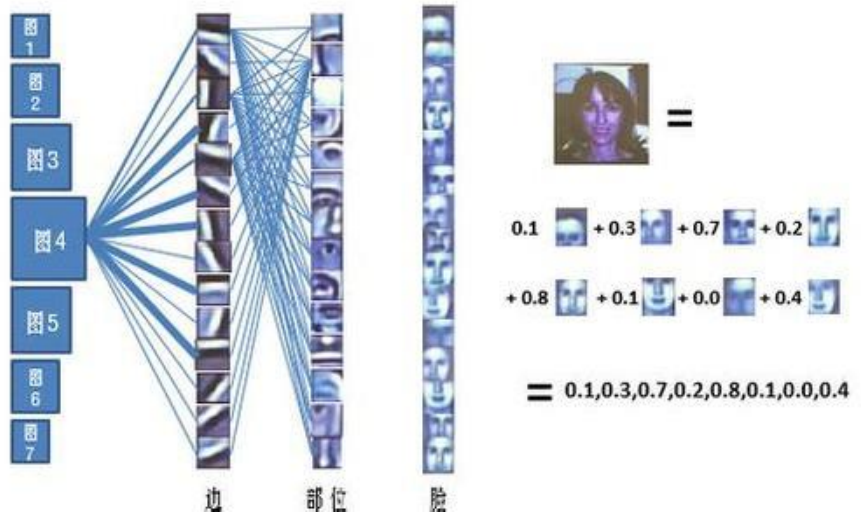
- LeNet
- AlexNet
- Inception网络
- 残差网络

7.1 参数学习

参数学习　　如何训练?

训练是一种有监督学习。

大量的样本数据:



前向传播　**损失函数**　**反向传播**　**更新权重**

训练什么?

训练输入的样本和分类对象是已定的，训练的深度（隐藏层的层数）和卷积核（神经元）的数量、大小都是训练前根据经验设定的。

如果训练参数设置不合理会导致过拟合或者欠拟合！！

在全连接前馈神经网络中，梯度主要通过每一层的误差项δ进行反向传播，并进一步计算每层参数的梯度。和全连接前馈网络类似，卷积网络也可以通过误差反向传播算法来进行参数学习，参数为卷积核中权重以及偏置。

在卷积神经网络中，主要有两种不同功能的神经层：卷积层和池化层。而参数为卷积核以及偏置，因此只需要计算卷积层中参数的梯度。

使用反向传播算法的前馈神经网络随机梯度下降训练过程

输入：训练集$D=\{(\boldsymbol{x}^{(n)},y^{(n)})\}_{n=1}^{N}$，验证集$V$，学习率$\alpha$，正则化系数$\lambda$，网络层数$L$，神经元数量$\{M_l\}_{l=1}^{L}$.

输出：$\boldsymbol{W}$, $\boldsymbol{b}$

随机初始化$\boldsymbol{W}$, $\boldsymbol{b}$;

repeat

  对训练集D中的样本随机重排序;

  For $n=1\ldots N$ **do**

    从训练集D中选取样本$(\boldsymbol{x}^{(n)},y^{(n)})$;

    前馈计算每一层的净输入$\boldsymbol{z}^{(l)}$和激活值$\boldsymbol{a}^{(l)}$，直到最后一层;

    反向传播计算每一层的误差$\boldsymbol{\delta}^{(l)}=f_l'(\boldsymbol{z}^{(l)})\odot\left((\boldsymbol{W}^{(l+1)})^{\mathrm{T}}\boldsymbol{\delta}^{(l+1)}\right)$; //最后一层的误差为$\boldsymbol{\delta}^{(L)}=f_L'(\boldsymbol{z}^{(L)})\odot\frac{\partial L(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial\hat{\boldsymbol{y}}}$

    //计算每一层的梯度

    $\frac{\partial L(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial \boldsymbol{W}^{(l)}}=\boldsymbol{\delta}^{(l)}(\boldsymbol{a}^{(l-1)})^{\mathrm{T}}$;

    $\frac{\partial L(\boldsymbol{y},\hat{\boldsymbol{y}})}{\partial \boldsymbol{b}^{(l)}}=\boldsymbol{\delta}^{(l)}$;

    //更新参数

    $\boldsymbol{W}^{(l)}\leftarrow=\boldsymbol{W}^{(l)}-\alpha\left(\boldsymbol{\delta}^{(l)}(\boldsymbol{a}^{(l-1)})^{\mathrm{T}}+\lambda\boldsymbol{W}^{(l)}\right)$;

    $\boldsymbol{b}^{(l)}=\boldsymbol{b}^{(l)}-\alpha\boldsymbol{\delta}^{(l)}$;

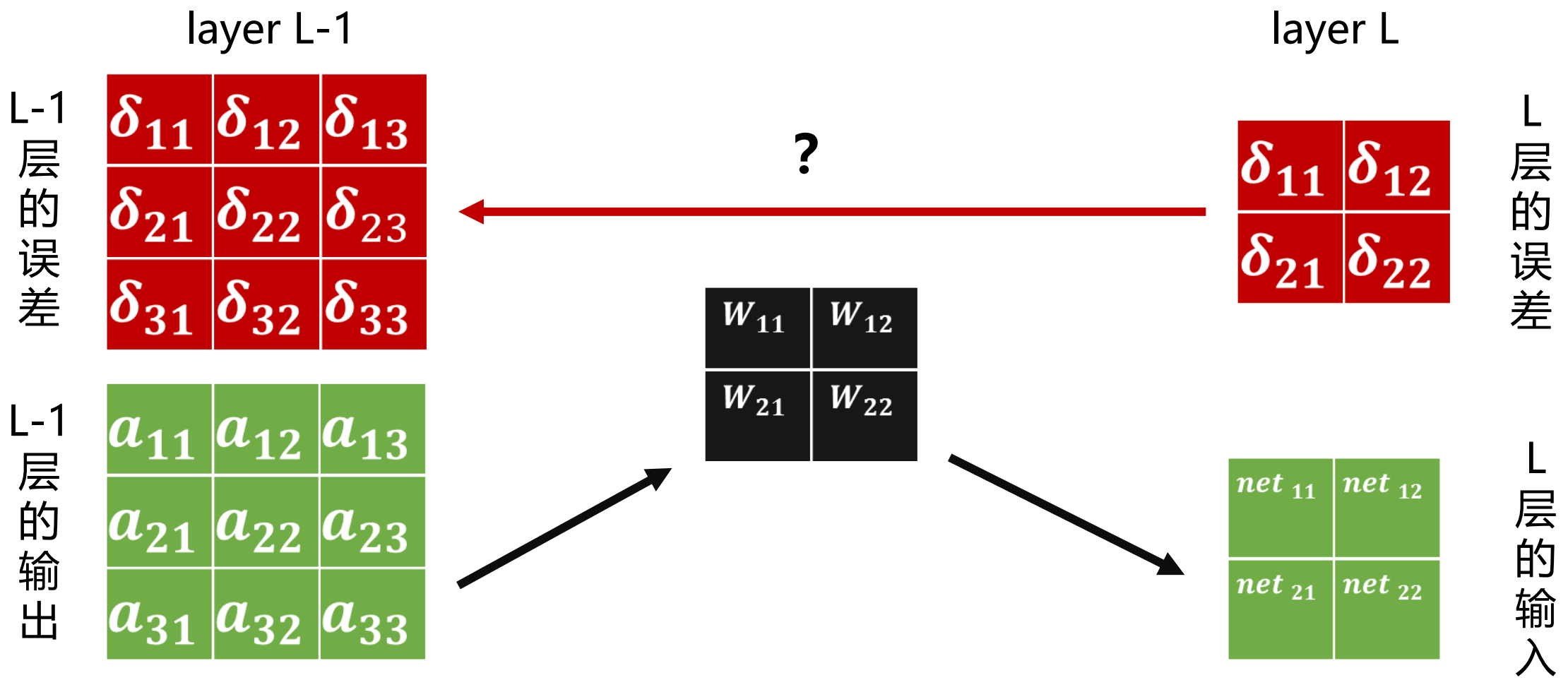
  end;

until 模型在验证集V上的错误率不再下降;

CNN反向传播计算过程

- 传统的神经网络是全连接形式的，如果进行反向传播，只需要由下一层对前一层不断的求偏导，即求链式偏导就可以求出每一层的误差敏感项，然后求出权重和偏置项的梯度，即可更新权重。
- 而卷积神经网络有两个特殊的层：卷积层和池化层。池化层输出时不需要经过激活函数，是一个滑动窗口的最大值，一个常数，那么它的偏导是1。池化层相当于对上层图片做了一个压缩，这个反向求误差敏感项时与传统的反向传播方式不同。从卷积后的feature_map反向传播到前一层时，由于前向传播时是通过卷积核做卷积运算得到的feature_map，所以反向传播与传统的也不一样，需要更新卷积核的参数。

参数学习


layer L-1
layer L
L-1层的误差
L-1层的输出
?
L层的误差
L层的输入
δ_{11} δ_{12} δ_{13}
δ_{21} δ_{22} δ_{23}
δ_{31} δ_{32} δ_{33}
a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}
W_{11} W_{12}
W_{21} W_{22}
δ_{11} δ_{12}
δ_{21} δ_{22}
net_{11} net_{12}
net_{21} net_{22}

以δ_{11}^{l-1}, δ_{22}^{l-1}为例

$$\delta_{11}^{l-1} = -\frac{\partial E_d}{\partial net_{11}^{l-1}} = -\frac{\partial E_d}{\partial a_{11}^{l-1}}\frac{\partial a_{11}^{l-1}}{\partial net_{11}^{l-1}} \quad , \quad a_{11}^{l-1}\text{仅能通过}net_{11}^{l}\text{来影响E}$$

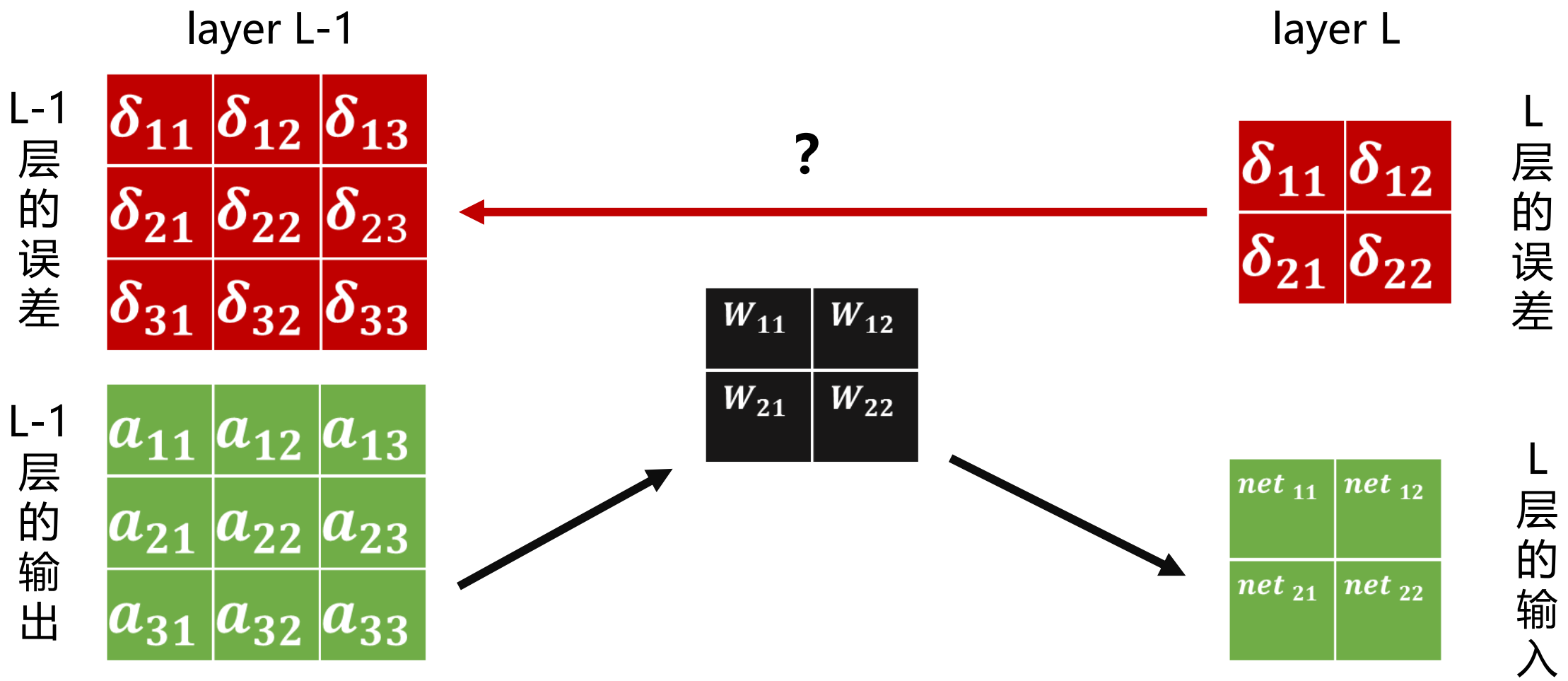
$$net_{11}^{l} = w_{11}\cdot a_{11}^{l-1} + w_{12}\cdot a_{12}^{l-1} + w_{21}\cdot a_{21}^{l-1} + w_{22}\cdot a_{22}^{l-1},$$

$$\Rightarrow \quad -\frac{\partial E_d}{\partial a_{11}^{l-1}} = -\frac{\partial E_d}{\partial net_{11}^{l}}\frac{\partial net_{11}^{l}}{\partial a_{11}^{l-1}} = \delta_{11}^{l}\cdot w_{11}$$

$$a_{11}^{l-1} = f(net_{11}^{l-1}), \quad \Rightarrow \quad \frac{\partial a_{11}^{l-1}}{\partial net_{11}^{l-1}} = f'\left(net_{11}^{l-1}\right)$$

$$\boxed{\delta_{11}^{l-1} = \delta_{11}^{l}\cdot w_{11}\cdot f'\left(net_{11}^{l-1}\right)}$$

参数学习


layer L-1
layer L
L-1层的误差
L-1层的输出
?
L层的误差
L层的输入
δ_{11} δ_{12} δ_{13}
δ_{21} δ_{22} δ_{23}
δ_{31} δ_{32} δ_{33}
a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}
W_{11} W_{12}
W_{21} W_{22}
δ_{11} δ_{12}
δ_{21} δ_{22}
net_{11} net_{12}
net_{21} net_{22}

$$\delta_{22}^{l-1} = -\frac{\partial E_d}{\partial net_{22}^{l-1}} = -\frac{\partial E_d}{\partial a_{22}^{l-1}} \frac{\partial a_{22}^{l-1}}{\partial net_{22}^{l-1}}$$

a_{22}^{l-1}可以通过net_{11}^{l}、 net_{12}^{l}、 net_{21}^{l}、 net_{22}^{l}来影响E

因 $net_{11}^{l} = w_{11} \cdot a_{11}^{l-1} + w_{12} \cdot a_{12}^{l-1} + w_{21} \cdot a_{21}^{l-1} + w_{22} \cdot a_{22}^{l-1}$,

$net_{12}^{l} = w_{11} \cdot a_{12}^{l-1} + w_{12} \cdot a_{13}^{l-1} + w_{21} \cdot a_{22}^{l-1} + w_{22} \cdot a_{23}^{l-1}$,

$net_{21}^{l} = w_{11} \cdot a_{21}^{l-1} + w_{12} \cdot a_{22}^{l-1} + w_{21} \cdot a_{31}^{l-1} + w_{22} \cdot a_{32}^{l-1}$,

$net_{22}^{l} = w_{11} \cdot a_{22}^{l-1} + w_{12} \cdot a_{23}^{l-1} + w_{21} \cdot a_{32}^{l-1} + w_{22} \cdot a_{33}^{l-1}$,

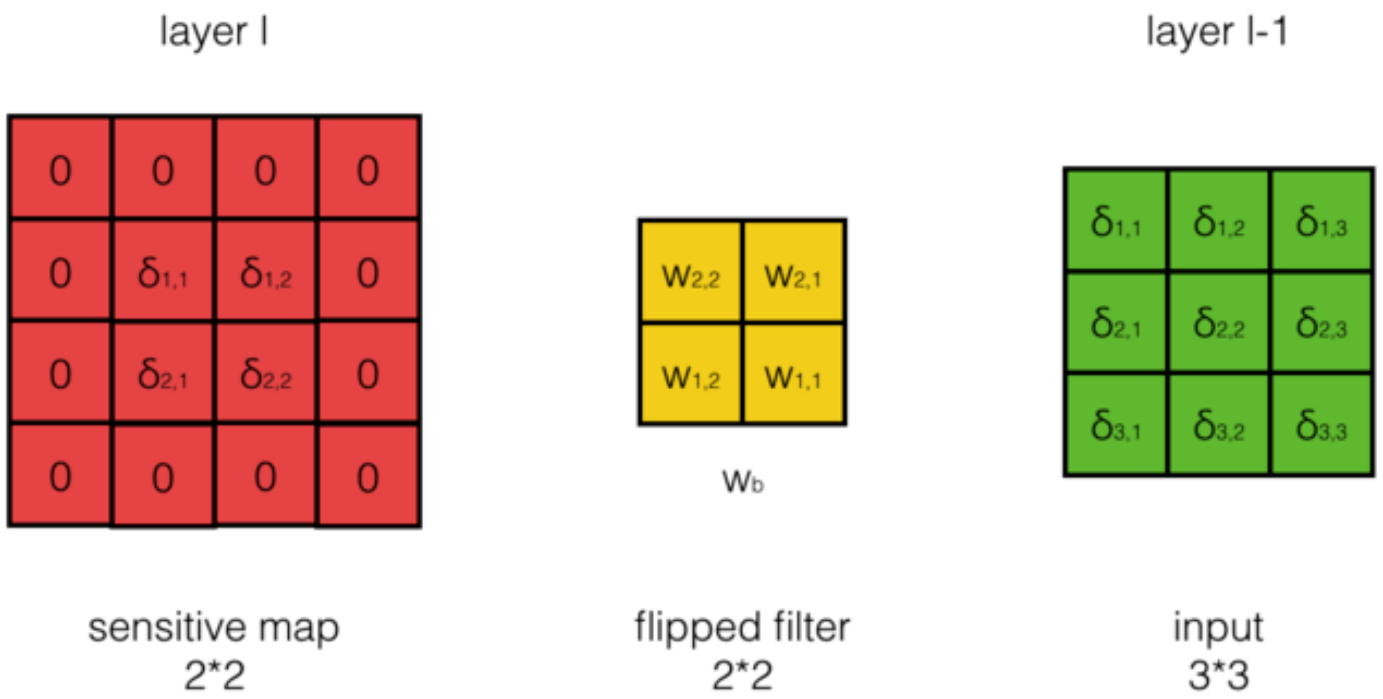
故 $-\frac{\partial E_d}{\partial a_{22}^{l-1}} = \delta_{11}^{l} \cdot w_{22} + \delta_{12}^{l} \cdot w_{21} + \delta_{21}^{l} \cdot w_{12} + \delta_{22}^{l} \cdot w_{11}$

又因为 $\frac{\partial a_{22}^{l-1}}{\partial net_{22}^{l-1}} = f'\left(net_{22}^{l-1}\right)$

$$\delta_{22}^{l-1} = \left(\delta_{11}^{l} \cdot w_{22} + \delta_{12}^{l} \cdot w_{21} + \delta_{21}^{l} \cdot w_{12} + \delta_{22}^{l} \cdot w_{11}\right) \cdot f'\left(net_{22}^{l-1}\right)$$

$$\delta_{11}^{l-1} = \delta_{11}^{l} \cdot w_{11} \cdot f'\left(net_{11}^{l-1}\right)$$

$$\delta_{22}^{l-1} = \left(\delta_{11}^{l} \cdot w_{22} + \delta_{12}^{l} \cdot w_{21} + \delta_{21}^{l} \cdot w_{12} + \delta_{22}^{l} \cdot w_{11}\right) \cdot f'\left(net_{22}^{l-1}\right)$$



卷积层filter权重梯度的计算

权重项w_{ij}可以通过net_{ij}^{l}影响的值进而影响E，以w_{11}为例

由$net_{11}^{l} = w_{11} \cdot a_{11}^{l-1} + w_{12} \cdot a_{12}^{l-1} + w_{21} \cdot a_{21}^{l-1} + w_{22} \cdot a_{22}^{l-1}$，可知

$$net_{12}^{l} = w_{11} \cdot a_{12}^{l-1} + w_{12} \cdot a_{13}^{l-1} + w_{21} \cdot a_{22}^{l-1} + w_{22} \cdot a_{23}^{l-1},$$

$$net_{21}^{l} = w_{11} \cdot a_{21}^{l-1} + w_{12} \cdot a_{22}^{l-1} + w_{21} \cdot a_{31}^{l-1} + w_{22} \cdot a_{32}^{l-1},$$

$$net_{22}^{l} = w_{11} \cdot a_{22}^{l-1} + w_{12} \cdot a_{23}^{l-1} + w_{21} \cdot a_{32}^{l-1} + w_{22} \cdot a_{33}^{l-1},$$

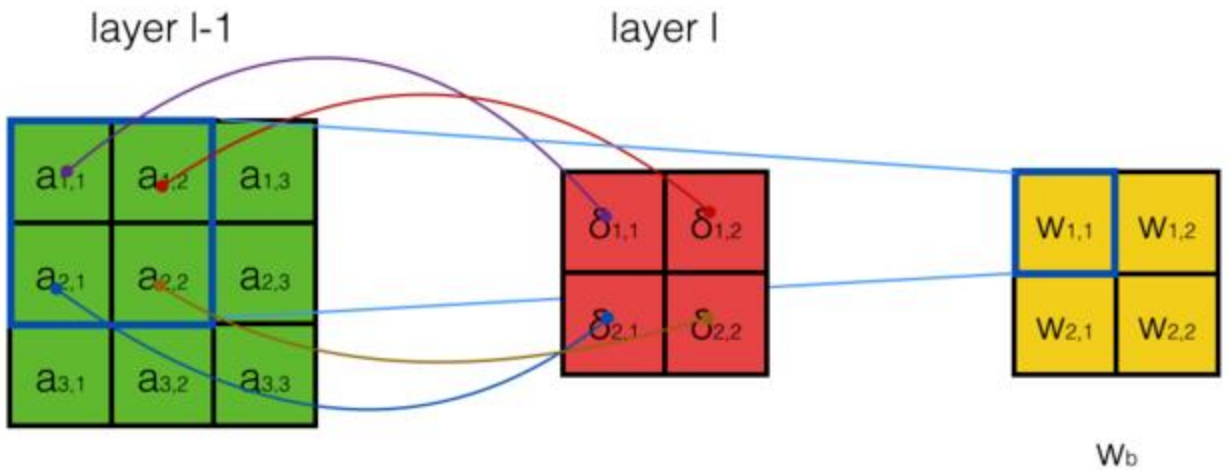
$$\frac{\partial E_d}{\partial w_{11}} = \frac{\partial E_d}{\partial net_{11}^{l}} \frac{\partial net_{11}^{l}}{\partial w_{11}} + \frac{\partial E_d}{\partial net_{12}^{l}} \frac{\partial net_{12}^{l}}{\partial w_{11}} + \frac{\partial E_d}{\partial net_{21}^{l}} \frac{\partial net_{21}^{l}}{\partial w_{11}} + \frac{\partial E_d}{\partial net_{22}^{l}} \frac{\partial net_{22}^{l}}{\partial w_{11}}$$

$$= -(\delta_{11}^{l} a_{11}^{l-1} + \delta_{12}^{l} a_{12}^{l-1} + \delta_{21}^{l} a_{21}^{l-1} + \delta_{22}^{l} a_{22}^{l-1})$$

$$\frac{\partial E_d}{\partial w_{12}} = -(\delta_{11}^{l} a_{12}^{l-1} + \delta_{12}^{l} a_{13}^{l-1} + \delta_{21}^{l} a_{22}^{l-1} + \delta_{22}^{l} a_{23}^{l-1})$$

卷积层filter权重梯度的计算

与误差传播类似，相当于l层的误差项（sensitivity map）与 l-1层的输出项做卷积操作，得到卷积核（filter）的梯度



$$\frac{\partial E_d}{\partial w} = \delta^l * a^{l-1}$$

大部分池化层没有需要训练的参数，只需要将误差传递。以Max Pooling为例

$$net\ _{11}^{l} = \max(net_{11}^{l-1}, net\ _{12}^{l-1}, net_{21}^{l-1}, net_{22}^{l-1})$$

也就是说，只有这块区域中最大的值net_{ij}^{l-1}才会对net_{ij}^{l}产生影响，假设最大值为net_{11}^{l-1}，上式可以写成

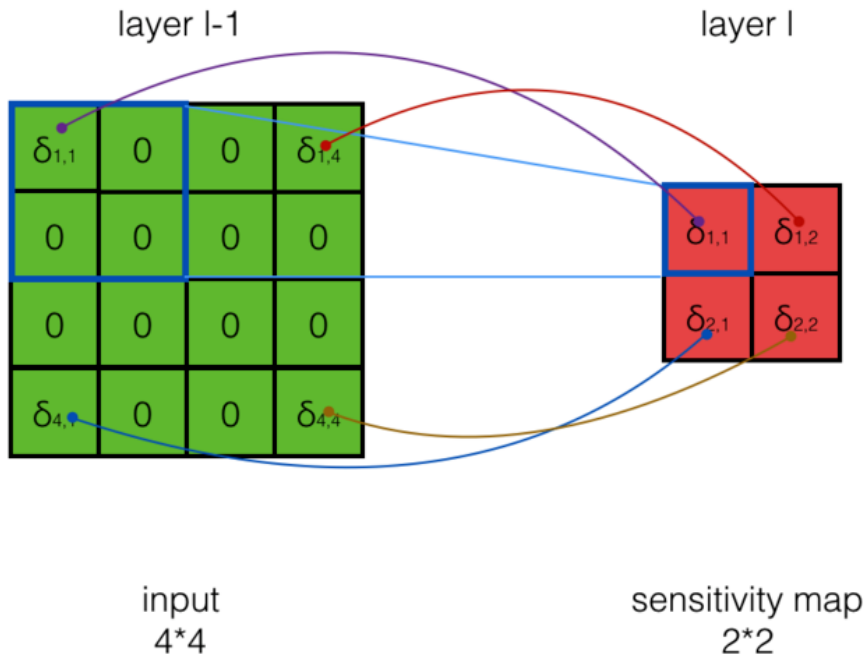
$$net\ _{11}^{l} = net_{11}^{l-1}$$

可得 $\frac{\partial net_{11}^{l}}{\partial net_{11}^{l-1}}=1,\frac{\partial net_{11}^{l}}{\partial net_{12}^{l-1}}=0,\frac{\partial net_{11}^{l}}{\partial net_{21}^{l-1}}=0,\frac{\partial net_{11}^{l}}{\partial net_{22}^{l-1}}=0,$

$$\delta_{11}^{l-1}=-\frac{\partial E_d}{\partial net_{11}^{l-1}}=-\frac{\partial E_d}{\partial net_{11}^{l}}\frac{\partial net_{11}^{l}}{\partial net_{11}^{l-1}}=\delta_{11}^{l}$$

$$\delta_{12}^{l-1}=0,\quad \delta_{21}^{l-1}=0,\quad \delta_{22}^{l-1}=0$$

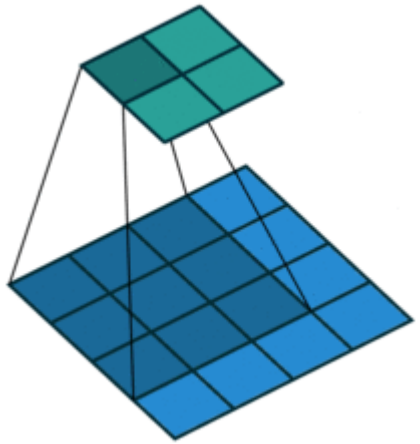
规律：对于max pooling，下一层的误差项的值会原封不动的传递到上一层对应区块中的最大值所对应的神经元，而其他神经元的误差项的值都是0。



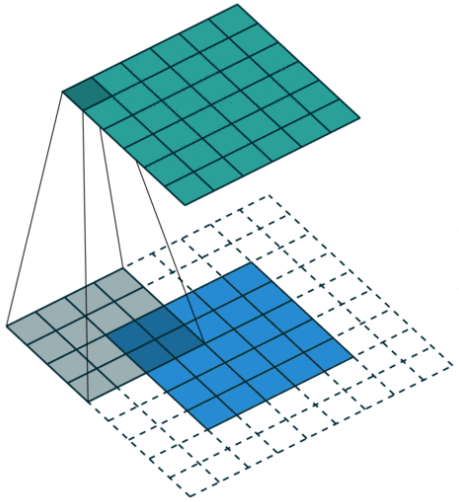
7.2 其它卷积方式

其他卷积方式

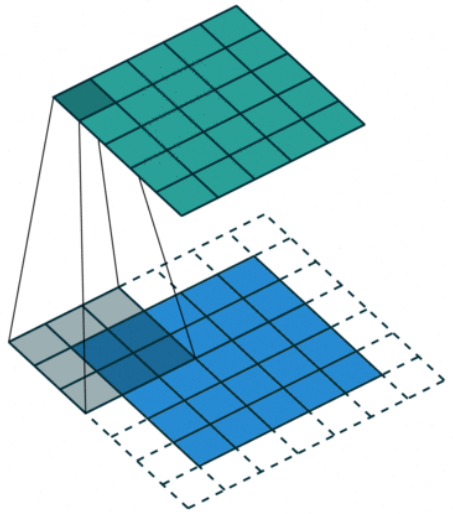
普通卷积



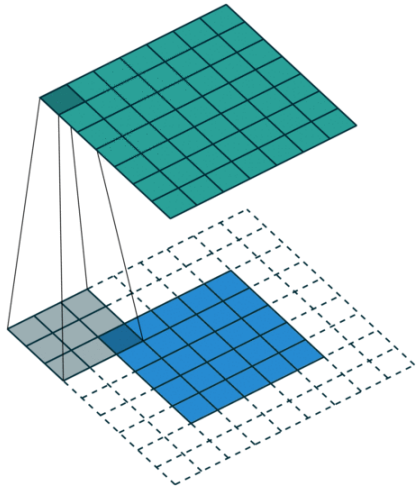
No padding, no strides



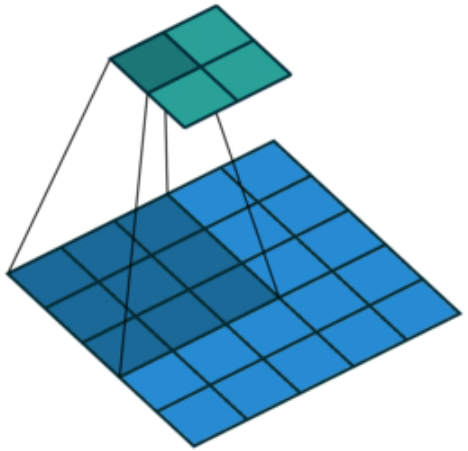
Arbitrary padding no strides



Half padding, no strides



Full padding, no strides



No padding, strides

通过卷积操作可以实现高维特征到低维特征的转换。比如在一维卷积中，一个5维的输入特征，经过一个大小为3的卷积核，其输出为3维特征。如果设置步长大于1，可以进一步降低输出特征的维数。但在一些任务中，需要将低维特征映射到高维特征，并且依然希望通过卷积操作来实现。

假设有一个高维向量为 $\boldsymbol{x} \in \mathbb{R}^d$和一个低维向量为 $\boldsymbol{z} \in \mathbb{R}^p, p < d$. 如果用仿射变换（Affine Transformation）来实现高维到低维的映射，

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}, \tag{1}$$

其中$\boldsymbol{W} \in \mathbb{R}^{p \times d}$为转换矩阵。通过转置$\boldsymbol{W}$可以实现低维到高维的反向映射，即

$$\boldsymbol{x} = \boldsymbol{W}^T\boldsymbol{z}. \tag{2}$$

需要说明的是，公式(1)和公式(2)并不是逆运算，两个映射只是形式上的转置关系。

卷积操作也可以写为仿射变换的形式。假设一个5维向量$\boldsymbol{x}$，经过大小为3的卷积核$\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T$进行卷积，得到3维向量$\boldsymbol{z}$。卷积操作可以写为

$$
\begin{aligned}
\boldsymbol{z} &= \boldsymbol{w} \otimes \boldsymbol{x} \\
&= \begin{bmatrix} w_1 & w_2 & w_3 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 \end{bmatrix} \boldsymbol{x} \\
&= \boldsymbol{C}\boldsymbol{x},
\end{aligned}
$$

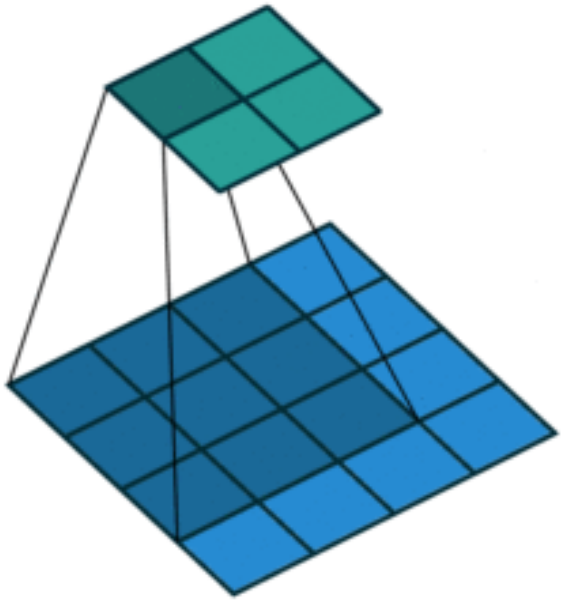
其中$\boldsymbol{C}$是一个稀疏矩阵，其非零元素来自于卷积核$\boldsymbol{\omega}$中的元素。

如果要实现3维向量z到5维向量的映射x，可以通过仿射矩阵的转置来实现，即

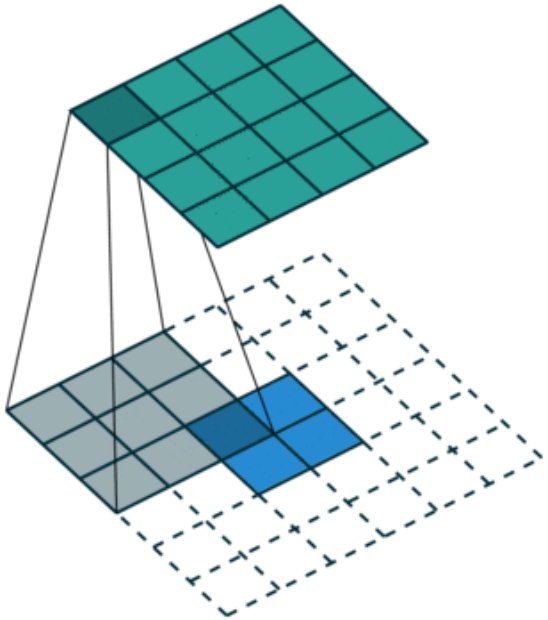
$$
\begin{aligned}
\boldsymbol{x} &= \boldsymbol{C}^{\mathrm{T}}\boldsymbol{z} \\
&= \begin{bmatrix} w_1 & 0 & 0 \\ w_2 & w_1 & 0 \\ w_3 & w_2 & w_1 \\ 0 & w_3 & w_2 \\ 0 & 0 & w_3 \end{bmatrix} \boldsymbol{z} \\
&= \mathrm{rot180}(\boldsymbol{w}) \tilde{\otimes} \boldsymbol{z},
\end{aligned}
$$

其中rot180(·)表示旋转180度。

从仿射变换的角度来看两个卷积操作$\boldsymbol{z} = \boldsymbol{w} \otimes \boldsymbol{x}$和$\boldsymbol{x} = \text{rot180}(\boldsymbol{w}) \tilde{\otimes} \boldsymbol{z}$也是形式上的转置关系。因此，将低维特征映射到高维特征的卷积操作称为转置卷积（Transposed Convolution），也称为反卷积（Deconvolution）。



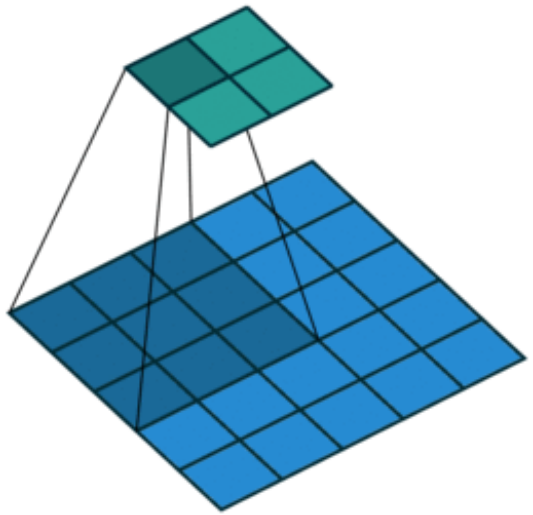
正常卷积



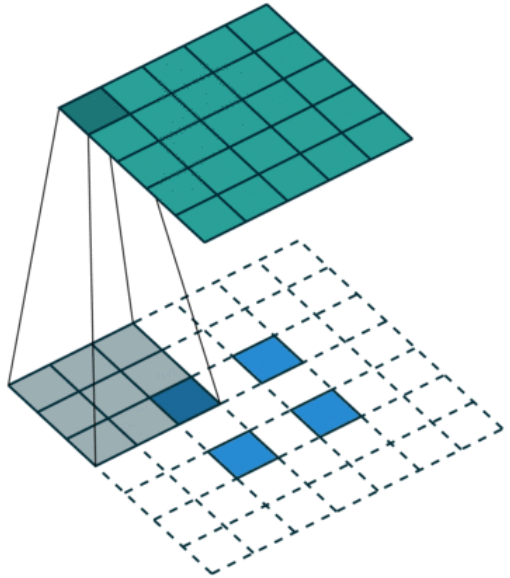
转置卷积

其他卷积方式　　转置卷积

通过增加卷积操作的步长 $S > 1$ 可以实现对输入特征的下采样操作，大幅降低特征维数。同样，也可以通过减少转置卷积的步长 $S < 1$ 来实现上采样操作，大幅提高特征维数。步长$S < 1$的转置卷积也称为微步卷积（Fractionally-Strided Convolution）。为了实现微步卷积，可以在输入特征之间插入D个0来间接地使得步长变小。



普通卷积：S=2,P=0



转置卷积：S=1,P=2,D=1

如何增加输出单元的感受野?

- 增加卷积核的大小;
- 增加层数，比如两层3 × 3 的卷积可以近似一层5 × 5 卷积的效果;
- 在卷积之前进行池化操作。

前两种方式会增加参数数量，而第三种方式会丢失一些信息。

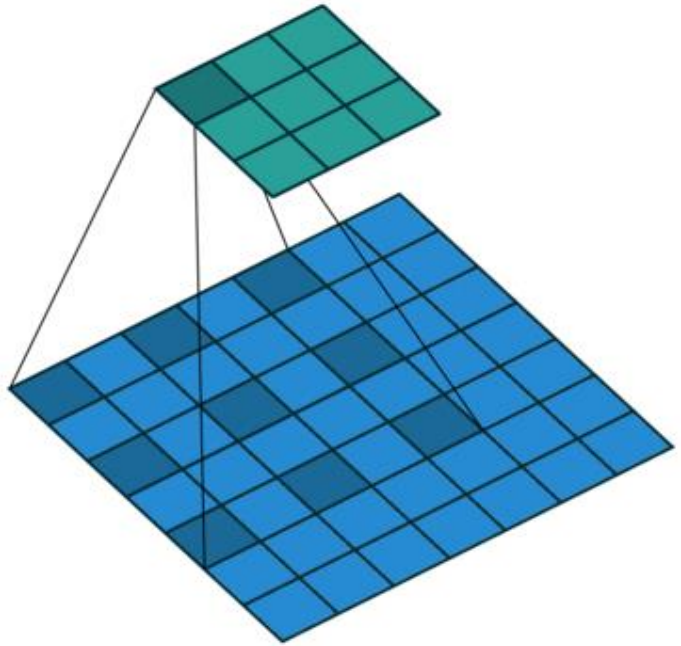
空洞卷积（Atrous Convolution）是一种不增加参数数量，同时增加输出单元感受野的方法，也称为膨胀卷积（Dilated Convolution）。空洞卷积通过给卷积核插入“空洞”来变相地增加其大小。

如果在卷积核的每两个元素之间插入$D-1$个空洞，卷积核的有效大小为：

$$K' = K + (K-1) \times (D-1),$$

其中D称为膨胀率（Dilation Rate）。

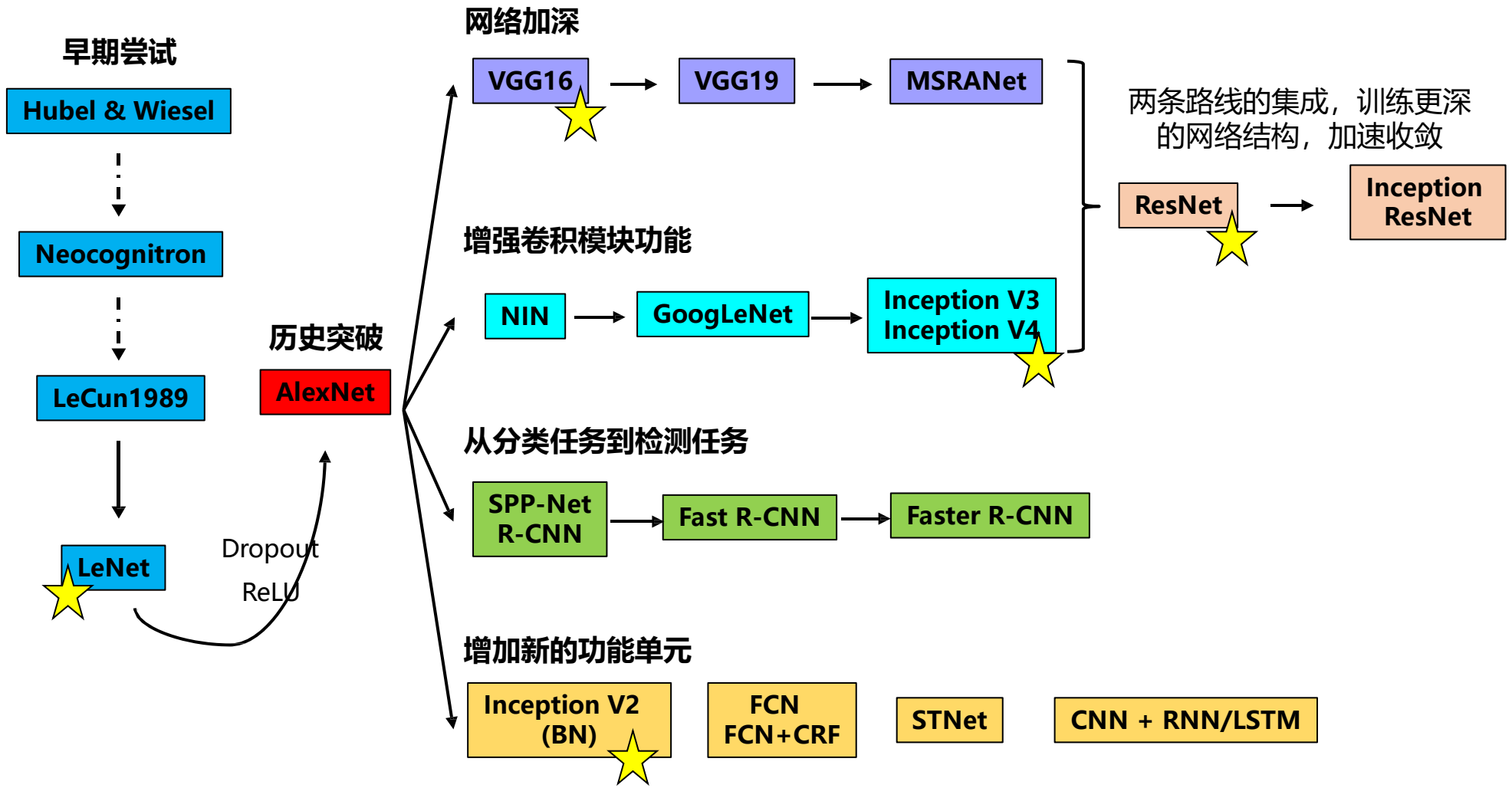
当$D=1$时卷积核为普通的卷积核。



7.3 典型网络简介

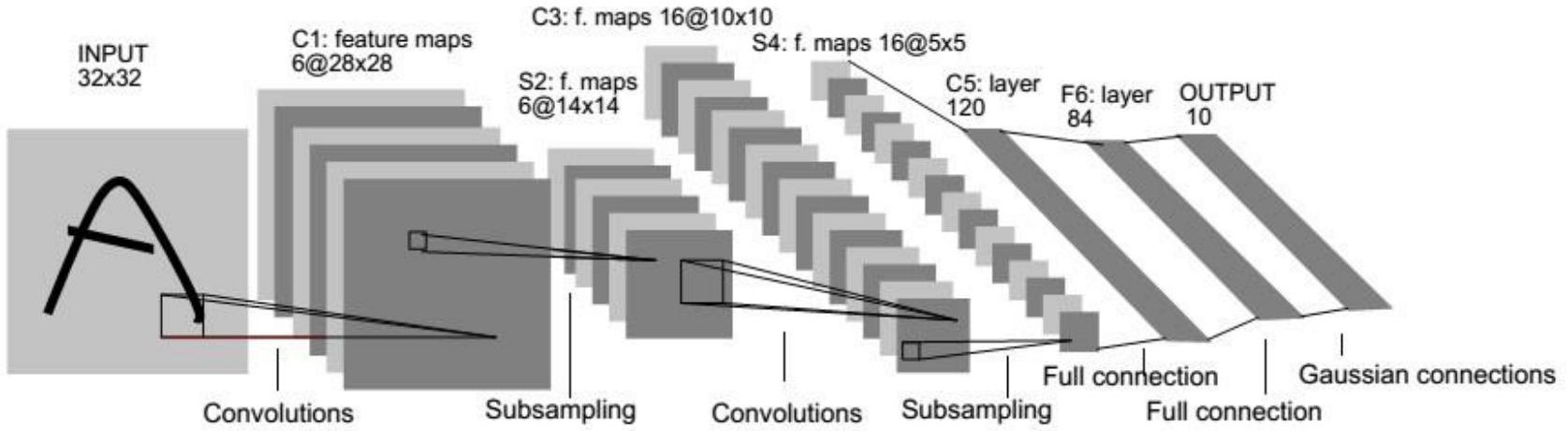
典型网络简介

演化脉络


早期尝试
Hubel & Wiesel
Neocognitron
LeCun1989
LeNet
Dropout
ReLU
历史突破
AlexNet
网络加深
VGG16
VGG19
MSRANet
两条路线的集成，训练更深的网络结构，加速收敛
ResNet
Inception ResNet
增强卷积模块功能
NIN
GoogLeNet
Inception V3
Inception V4
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 (BN)
FCN FCN+CRF
STNet
CNN + RNN/LSTM

LeNet 1998

LeNet-5[LeCun et al., 1998] 虽然提出的时间比较早，但它是一个非常成功的神经网络模型。基于LeNet-5的手写数字识别系统在20世纪90年代被美国很多银行使用，用来识别支票上面的手写数字。LeNet-5 的网络结构如图所示。



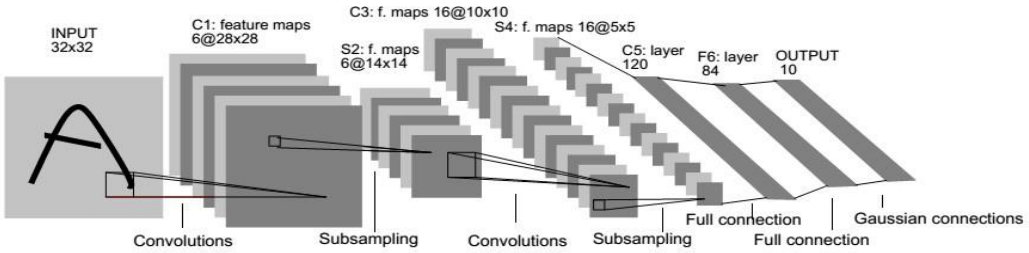
LeNet 1998

最早的深度卷积神经网络模型，用于字符识别。网络具有如下特点：

- 卷积神经网络使用三个层作为一个系列： 卷积、池化、非线性
- 使用卷积提取空间特征
- 使用映射到空间均值的下采样（subsample）
- 双曲线（tanh）或S型（sigmoid）形式的非线性
- 多层神经网络（MLP）作为最后的分类器

LeNet提供了利用卷积层堆叠进行特征提取的框架，开启了深度卷积神经网络的发展。

典型网络简介 LeNet 1998



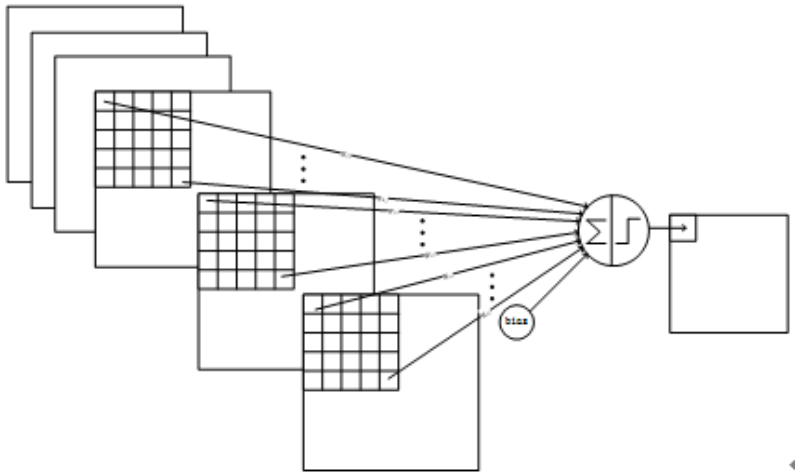
- 输入图像是32x32的大小，卷积核的大小是5x5的，由于不考虑对图像的边界进行拓展，则卷积核将有28x28个不同的位置，也就是C1层的大小是28x28。这里设定有6个不同的C1层，每一个C1层内的权值是相同的。
- S2层是一个下采样层，即池化层。在LeNet-5系统，下采样层比较复杂，由4个点下采样的加权平均为1个点，因为这4个加权系数也需要学习得到，这显然增加了模型的复杂度。
- 根据对前面C1层同样的理解，很容易得到C3层的大小为10x10。只不过，C3层的变成了16个10x10网络，有16个卷积核。 如果S2层只有1个平面，那么由S2层得到C3就和由输入层得到C1层是完全一样的。但是，S2层由多层，那么，只需要按照一定的顺序组合这些层就可以了。

具体的组合规则，在 LeNet-5 系统中给出了下面的表格：

S2层feature map

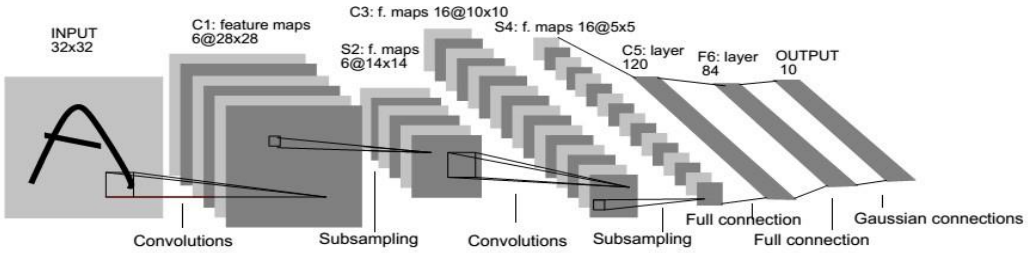
C3层feature map

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X



简单的说，例如对于C3层第0张特征图，其每一个节点与S2层的第0张特征图、第1张特征图、第2张特征图，总共3个5x5个节点相连接。后面依次类推，C3层每一张特征映射图的权值是相同的。

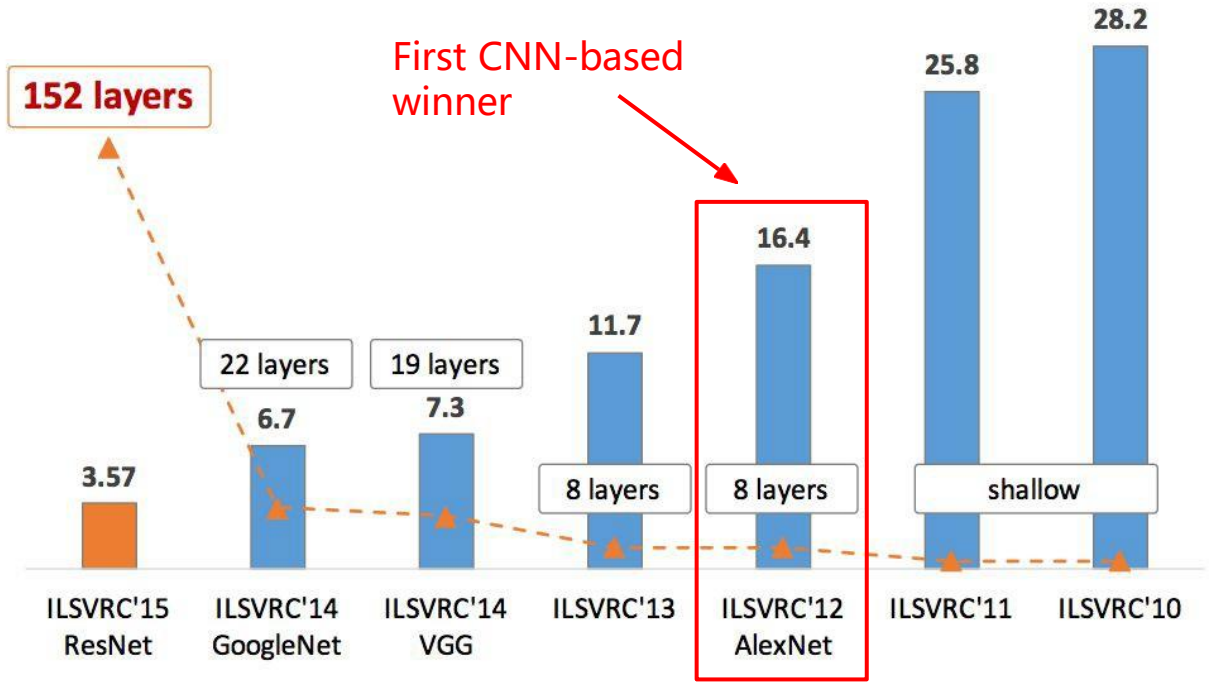
典型网络简介 LeNet 1998



4. S4 层是在C3层基础上下采样，前面已述。
5. C5层是一个卷积层，有120个特征图。每个单元与S4层的全部16个单元的5x5邻域相连，故C5特征图的大小为1x1：这构成了S4和C5之间的全连接。之所以仍将C5表示为卷积层而非全连接层，是因为如果LeNet-5的输入变大，而其他的保持不变，那么此时特征图的维数就会比1x1大。C5层有48120个可训练连接。
6. F6层有84个单元（之所以选这个数字的原因来自于输出层的设计），与C5层全相连。有10164个可训练参数。如同经典神经网络，F6层计算输入向量和权重向量之间的点积，再加上一个偏置。然后将其传递给sigmoid函数产生节点的输出。

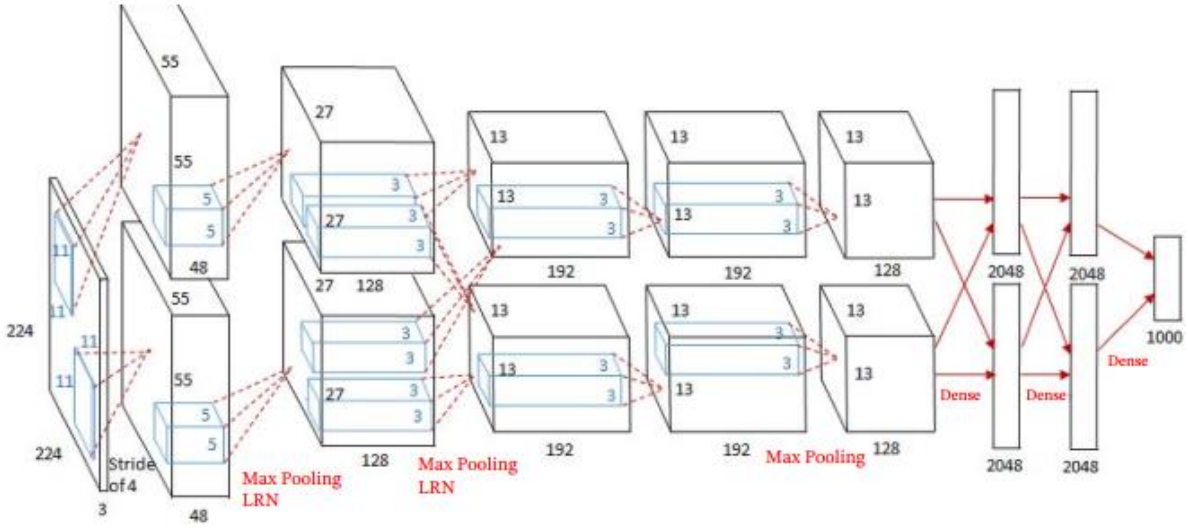
AlexNet 2012

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners! ！



AlexNet 2012

- **2012-吾家有女初长成：**
  - 非线性激活函数：ReLU
  - 防止过拟合：Dropout, 数据增广
  - 大数据训练：百万级ImageNet图像数据
  - 分Group实现双GPU并行，LRN归一化层
  - 5个卷积层、3个汇聚层和3个全连接层

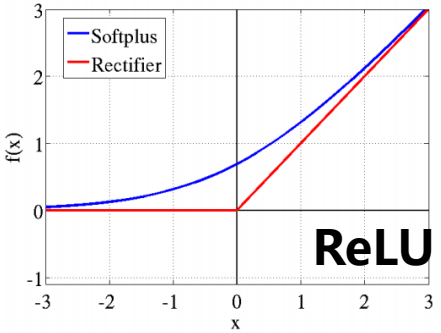


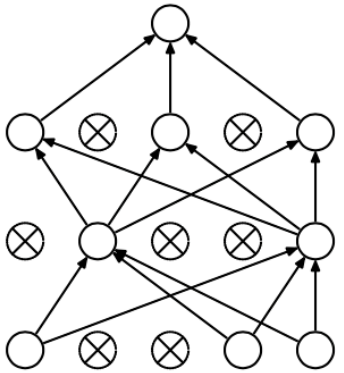
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "**Imagenet Classification with Deep Convolutional Neural Networks**." Advances in neural information processing systems. 2012.

AlexNet 2012

AlexNet在LeNet基础上进行了更宽更深的网络设计，首次在CNN中引入了ReLU、Dropout 和Local Response Norm (LRN)等技巧。网络的技术特点如下：

- 使用ReLU（Rectified Linear Units）作为CNN的激活函数，并验证其效果在较深的网络超过了Sigmoid，成功解决了Sigmoid在网络较深时的梯度弥散问题，提高了网络的训练速率。
- 为避免过拟合，训练时使用Dropout随机忽略一部分神经元。
- 使用重叠的最大池化(max pooling)。最大池化可以避免平均池化的模糊化效果，而采用重叠技巧可以提升特征的丰富性。





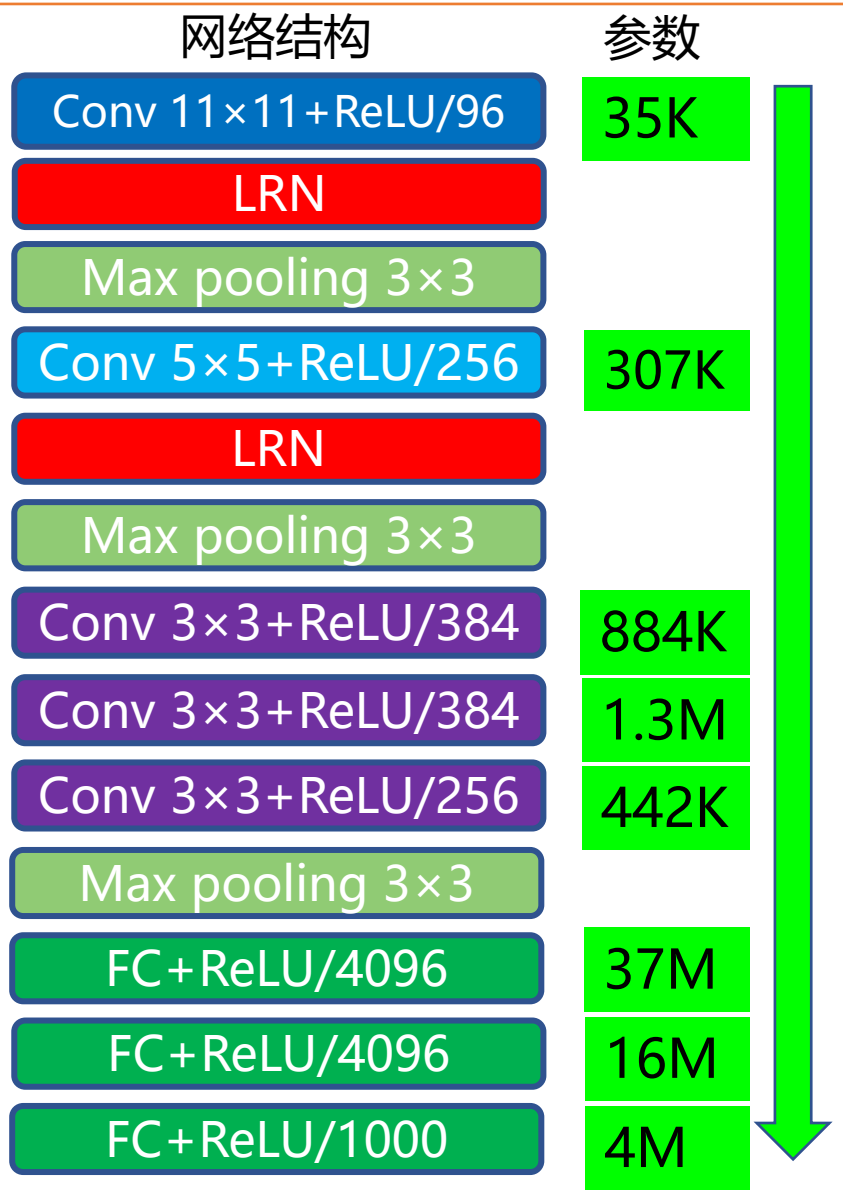
Dropout

AlexNet 2012

AlexNet在LeNet基础上进行了更宽更深的网络设计，首次在CNN中引入了ReLU、Dropout 和Local Response Norm (LRN)等技巧。网络的技术特点如下：

- 提出了LRN层（ReLU后进行归一化处理），对局部神经元的活动创建竞争机制，使得其中响应比较大的值变得相对更大，并抑制其他反馈较小的神经元，增强了模型的泛化能力。
- 利用GPU强大的并行计算能力加速网络训练过程，并采用GPU分块训练的方式解决显存对网络规模的限制。
- 数据增强。利用随机裁剪和翻转镜像操作增加训练数据量，降低过拟合。

典型网络简介　AlexNet 2012

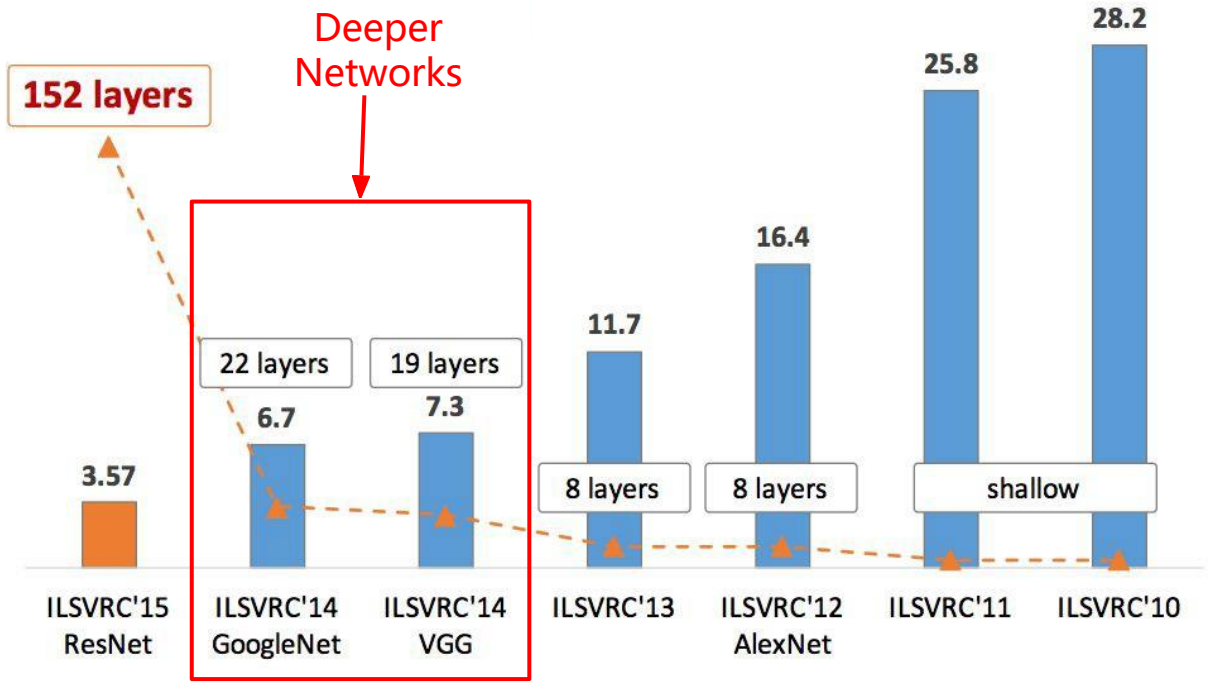


AlexNet网络配置和参数数量

- 卷积核大小递减，依次为11×11、5×5和3×3。第一层卷积步长为4，之后保持为1。
- 在前两层卷积之后使用了LRN层。
- 与全连接层相比，卷积层包含较少的参数。因此可通过减少全连接层降低网络参数，提高训练时间，在Network in Network中利用了这一点。

AlexNet在ILSVRC2012图像分类竞赛中将top-5 错误率降至16.4%，掀起了深度卷积神经网络在各个领域的研究热潮。

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners! ！



Inception网络

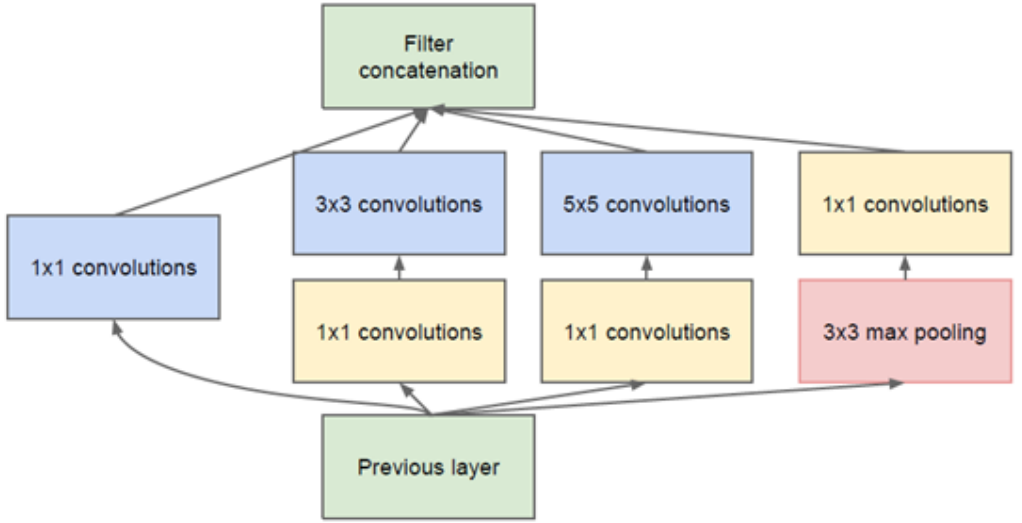
- 2014 ILSVRC winner （22层）
  - 参数：GoogLeNet (4M) vs AlexNet (60M)
  - 错误率：6.7%
  - Inception网络是由有多个inception模块和少量的汇聚层堆叠而成。
  - Inception 网络有多个版本，其中最早的Inception v1 版本就是非常著名的GoogLeNet [Szegedy et al., 2015]。GoogLeNet 不写为GoogleNet，是为了向LeNet 致敬。GoogLeNet 赢得了2014 年ImageNet 图像分类竞赛的冠军。

Inception网络

Inception module 包含四个分支：

- Shortcut连接：将前一层输入通过1×1卷积
- 多尺度滤波：输入通过1×1卷积之后分别连接卷积核大小为3和5的卷积
- 池化分支：相继连接3×3 pooling和1×1卷积

四个分支的输出经过串联恢复到输入通道大小

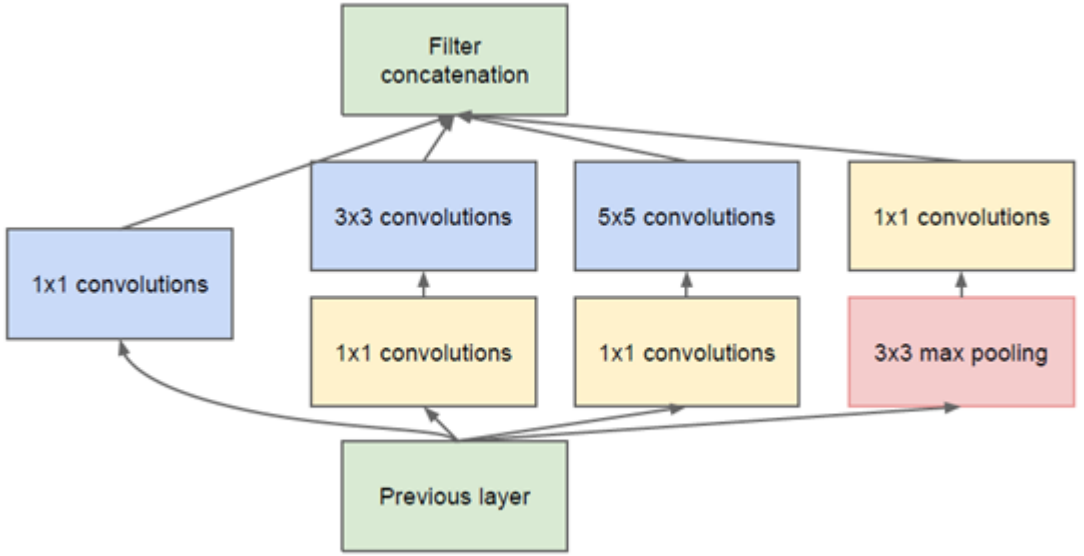


Inception Module

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. **Going Deeper with Convolutions**. CVPR, 2015.

Inception module 优点一：**减少网络参数，降低运算量**

- 上一层的输出为100x100x128，经过具有256个通道的5x5卷积层之后(stride=1，pad=2)，输出数据为100x100x256，其中，卷积层的参数为128x5x5x256= 819200。
- 而假如上一层输出先经过具有32个通道的1x1卷积层，再经过具有256个输出的5x5卷积层，那么输出数据仍为为100x100x256，但卷积参数量已经减少为128x1x1x32 + 32x5x5x256= 204800，大约减少了4倍。

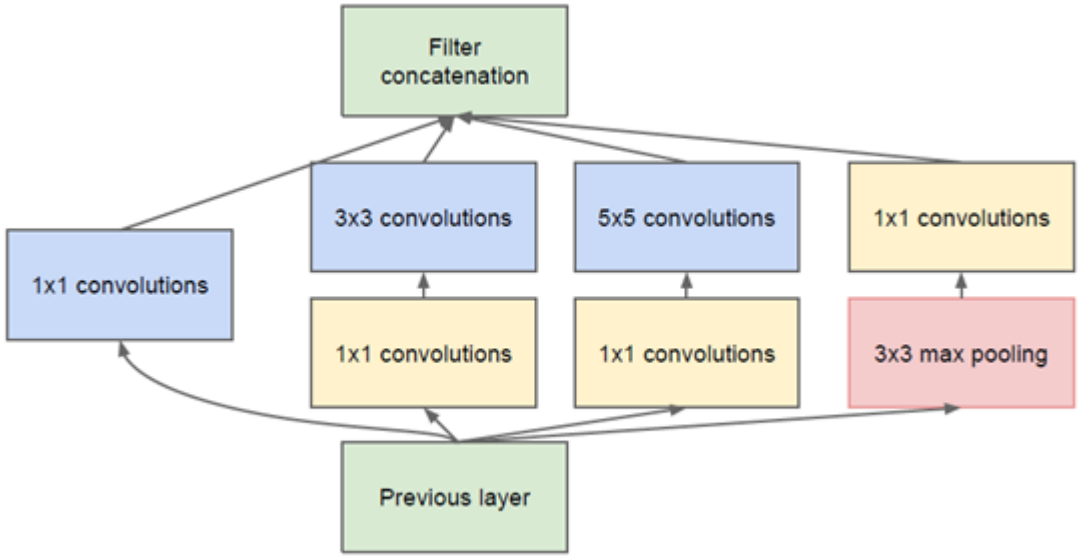


Inception module

因此，1×1卷积的作用之一是通过降维减少网络开销

Inception module 优点二： **多尺度、多层次滤波**

- 多尺度：对输入特征图像分别在3×3和5×5的卷积核上滤波，提高了所学特征的多样性，增强了网络对不同尺度的鲁棒性。
- 多层次：符合Hebbian原理，即通过1×1卷积把具有高度相关性的不同通道的滤波结果进行组合，构建出合理的稀疏结构。

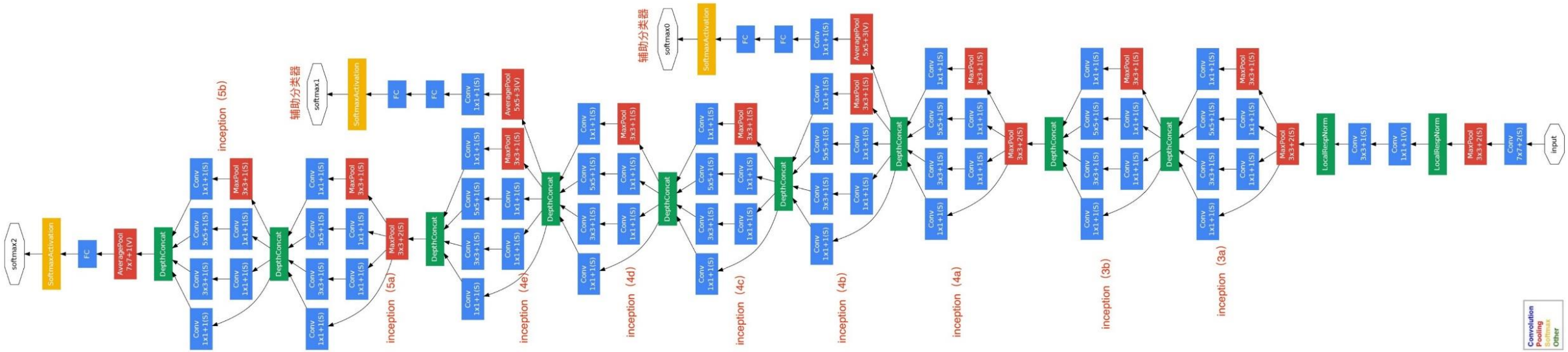


Inception module

因此，1×1卷积的另一作用是对低层滤波结果进行有效的组合

Inception模块 v1：GoogLeNet

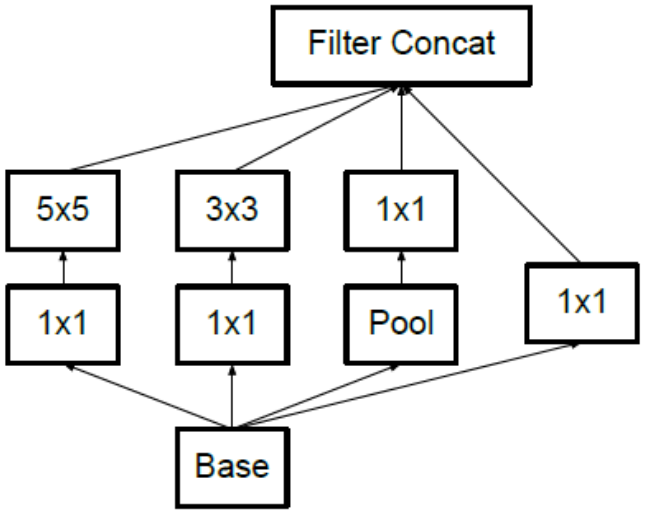
GoogLeNet的网络参数为AlexNet的1/12，由9个inception模块和5个池化层堆叠而成，在ILSVRC 2014 top-5错误率降至6.67%。



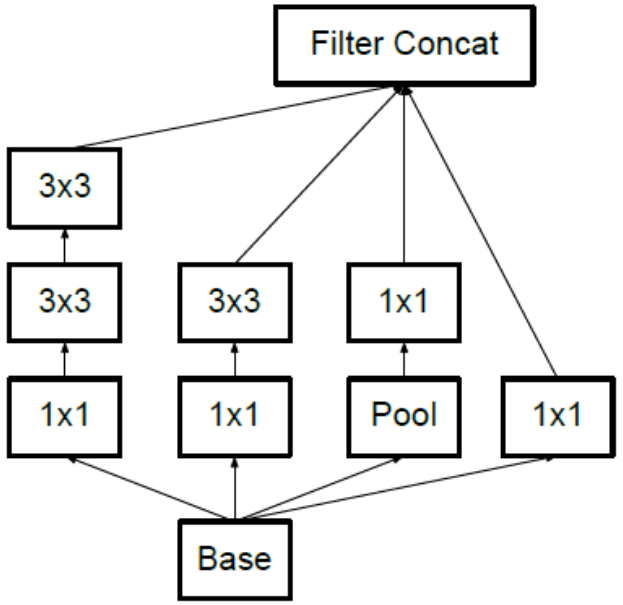
Inception网络

Inception模块 v3

- 用多层的小卷积核来替换大的卷积核，以减少计算量和参数量。
  - ✓使用两层3x3的卷积来替换v1中的5x5的卷积
  - ✓使用连续的nx1和1xn来替换nxn的卷积。

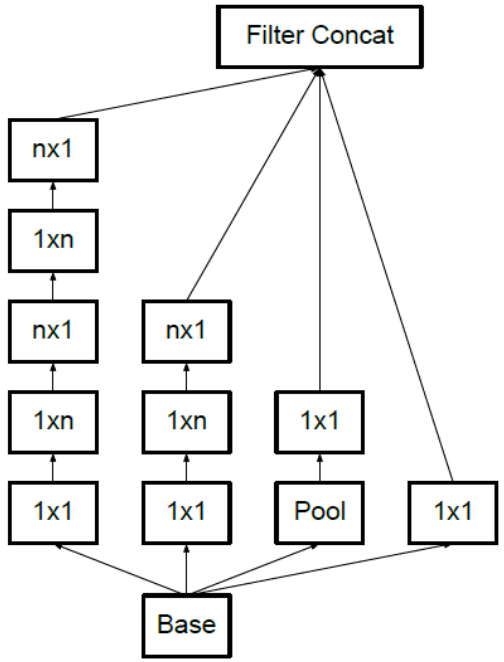


Inception V1 结构



用 3x3 卷积核代替 5x5 卷积核

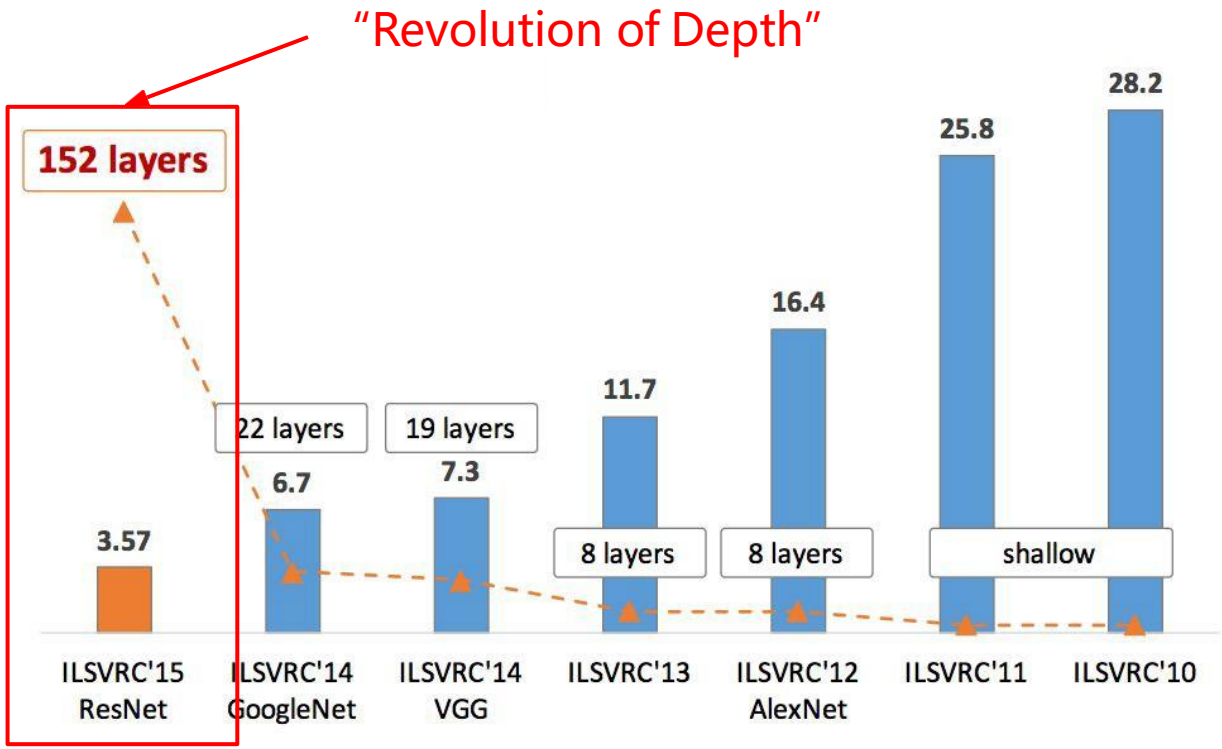
（小卷积核代替大卷积核）



用 1xn 和 nx1 卷积核代替 nxn 卷积核

Inception V2 结构

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners! !



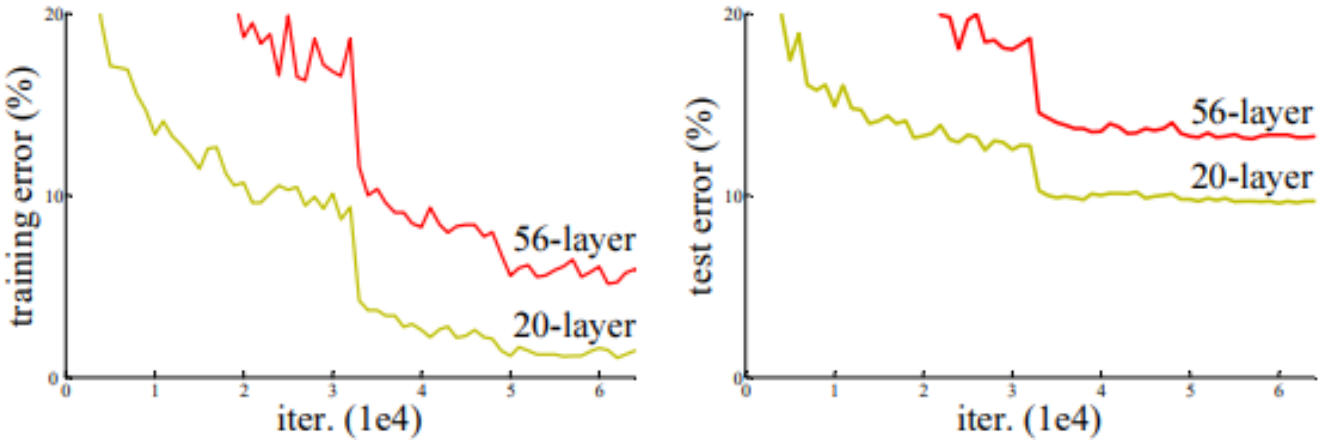
是否可以通过简单的层数堆叠学习更好的网络?

- 梯度消失和爆炸：随着网络的加深，在网络中反向传播的梯度会随着连乘变得不稳定，变得特别大或者特别小。

  => 通过Normalized initialization 和 Batch normalization得到解决。

- 网络退化（degradation）：随着网络的加深，准确率首先达到饱和，然后快速退化。

  => 在训练集上的错误率同样增加，因此并非受过拟合的影响。

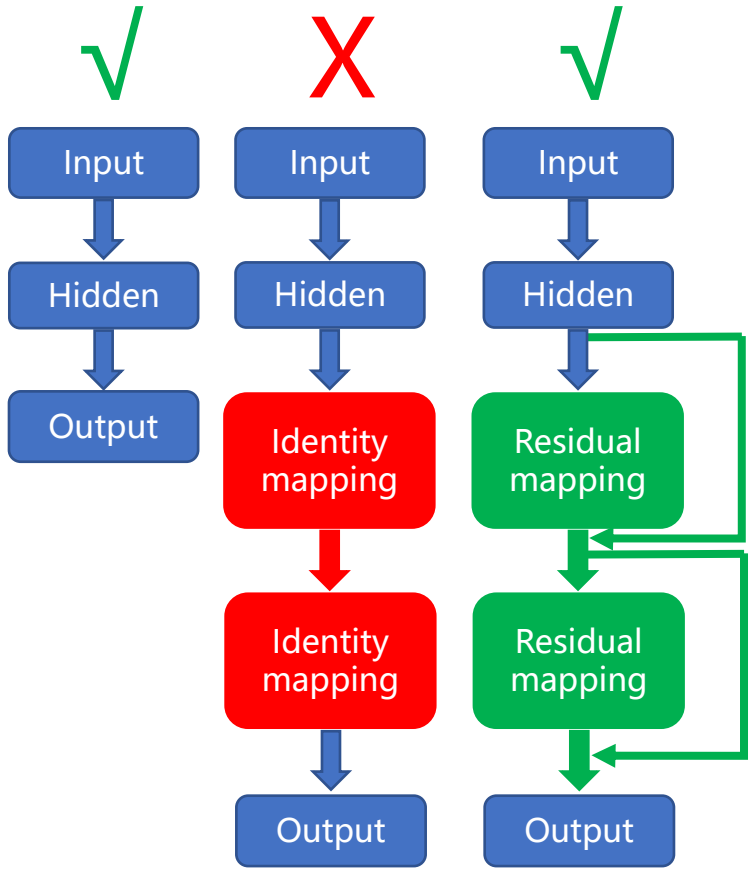


图：不同深度网络在CIFAR-10上的训练和测试错误率变化

简单的层数堆叠不能提升网络性能，如何利用网络加深带来的优势?

残差网络

- 实验表明，通过添加恒等映射不能提高网络准确率，因此，网络对恒等映射的逼近存在困难。
- 相比于恒等映射，网络对恒等映射附近扰动的学习更加简单。



残差学习

残差网络Residual Network, ResNet

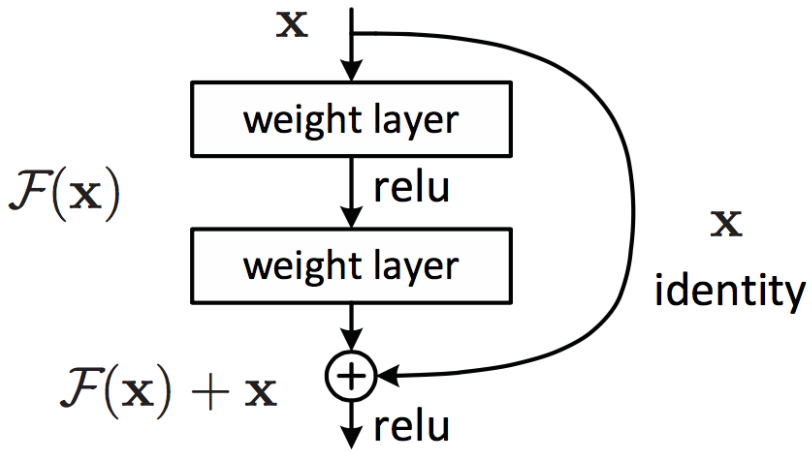
Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. **Deep Residual Learning for Image Recognition**. CVPR, 2016.

为什么是残差?

- ✓ 非常深的残差网络能够很容易的优化。
- ✓ 深度残差网络能够容易地从增加的深度中得到精度收益，同时比先前的网络产生了更好的效果。

残差映射

基本的残差映射(相同的维度)



$\mathcal{F}(\mathbf{x})$

$\mathcal{F}(\mathbf{x}) + \mathbf{x}$

$\mathbf{x}$

优点：

- 没有带来额外的参数和计算开销。
- 便于和具有相同结构的“平常”网络进行对比。

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

残差函数　　恒等函数

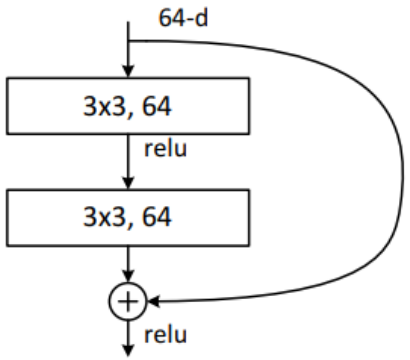
$$\mathcal{F} = W_2\sigma(W_1\mathbf{x})$$

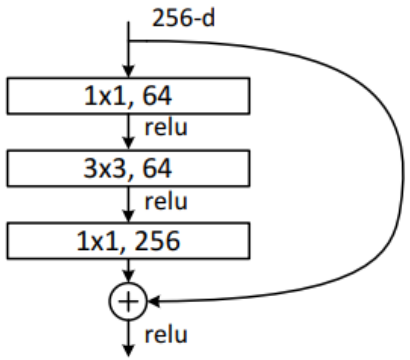
基本的残差映射(不同的维度)

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

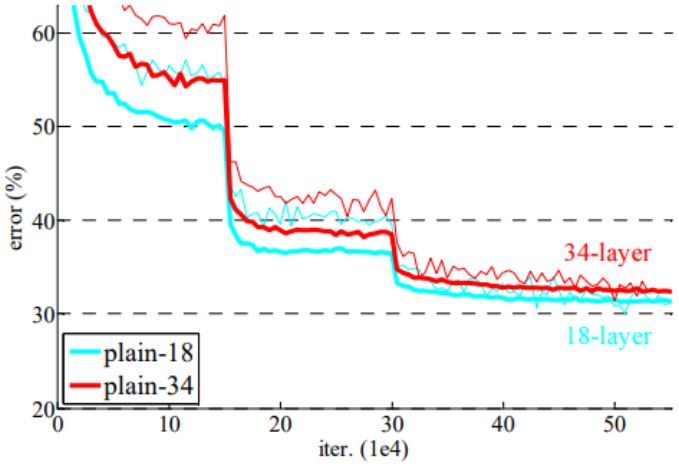
“但是我们将通过实验说明恒等映射能够经济高效的解决网络中的退化问题，因此W_s仅在匹配维度时使用。”

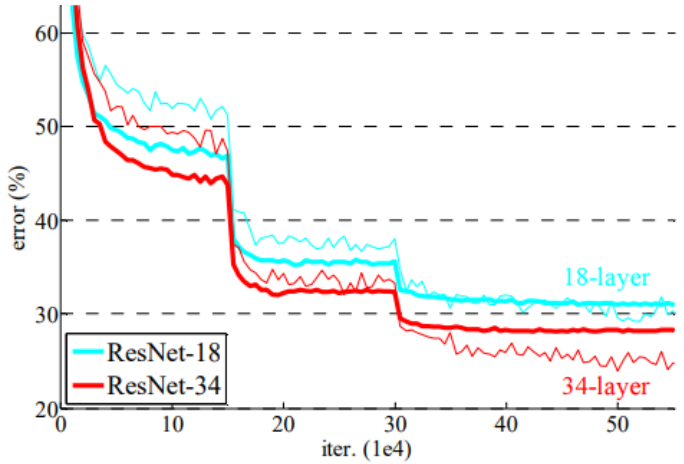
- 更深的残差结构
  - 用三层连接代替两层(1×1, 3×3, 1×1 卷积)
    - 1×1：降维和升维
    - 3×3：具有较小输入输出维度的卷积
- 利用深度增加带来的优势，同时减小了网络计算开销。结合了GoogLeNet和ResNet的优点。





更深的残差模块，用于更深的网络结构。左图的浅层模块用于构建ResNet-34，右图的深层模块用于构建ResNet-50/101/152。





18层和34层网络在ImageNet上训练结果

Case Study: ResNet

[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!