卷积神经网络基础

计算机视觉

卷积的源起

- 1968 Hubel & Wiesel
  Receptive fields and functional architecture of monkey striate cortex
- 1980 福岛邦彦(Fukusima Kunihiko)
  Neocognitron A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position
- 1989 Yann LeCun
  Backpropagation Applied to Handwritten Zip Code Recognition
- 1998 Yann LeCun
  Gradient-Based Learning Applied to Document Recognition
- 2006 Jake Bouvrie
  Notes on Convolutional Neural Networks
- 2012 Alex
  Imagenet classification with deep convolutional neural networks

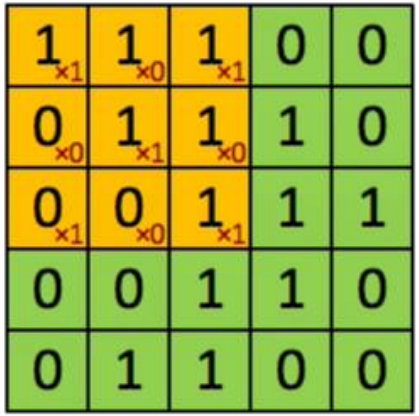
卷积神经网络基础

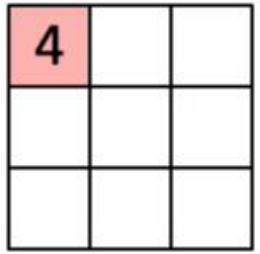
- 卷积
  - 步长，补齐和感受野
- 池化
  - 最大池化，平均池化和全局平均池化
- 新的技巧
  - Batch Normalization
  - Dropout

卷积（convolution）

直观理解：

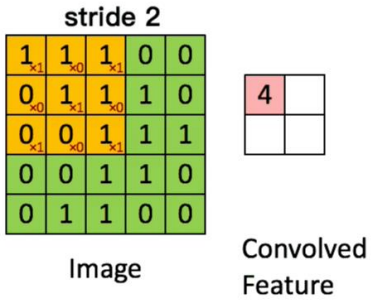
- 用一个小的权重矩阵去覆盖输入数据，对应位置元素加权相乘，其和作为结果的一个像素点。
- 这个权重在输入数据上滑动，形成一张新的矩阵
- 这个权重矩阵称为卷积核（convolution kernel）
- 其覆盖的位置称为感受野（receptive field）
- 生成的新的矩阵叫做特征图（feature map）





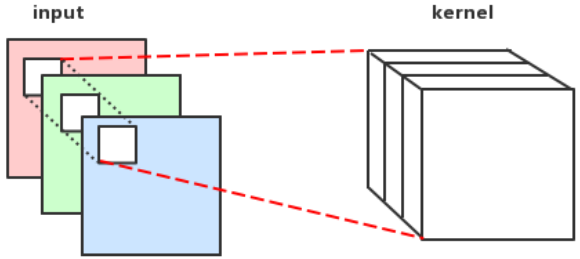
关于卷积，你应该知道……

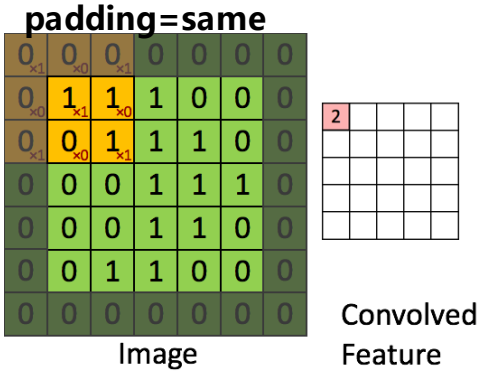
- 滑动的像素数量叫做步长（stride）



- 以卷积核的边还是中心点作为开始/结束的依据，决定了卷积的补齐（padding）方式。
此前我们看到的都是valid方式，而same方式则会在图像边缘用0补齐。

- 如果输入通道不只一个，那么卷积核是三阶的。所有通道的结果做累加。



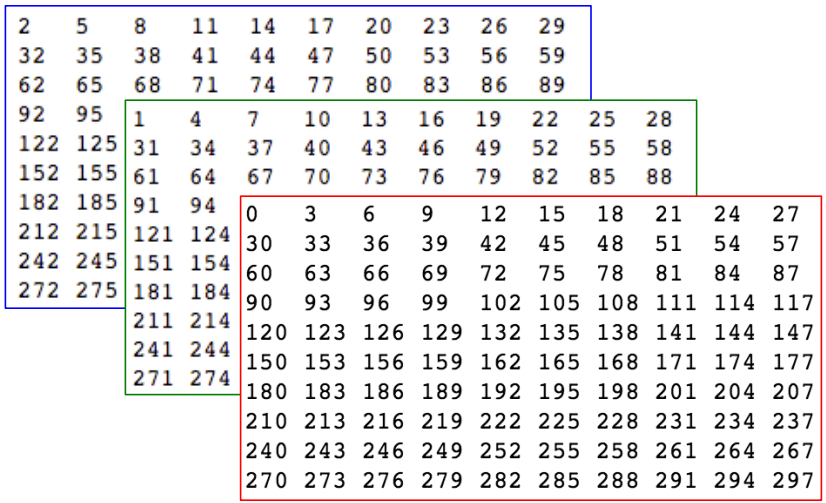


卷积的数学和代码表达

卷积核将某个局部感受野生成一个数值（新的feature map中的一个像素）

$$output_{ij} = \sigma(b + \Sigma_m \Sigma_n \Sigma_d w_{m,n,d} x_{i+m,j+n,d})$$

$$output = X * W + b$$



*

0	0	0
1	1	1
0	0	0

0	1	0
0	1	0
0	1	0

1	0	0
0	1	0
0	0	1

bias=0

=

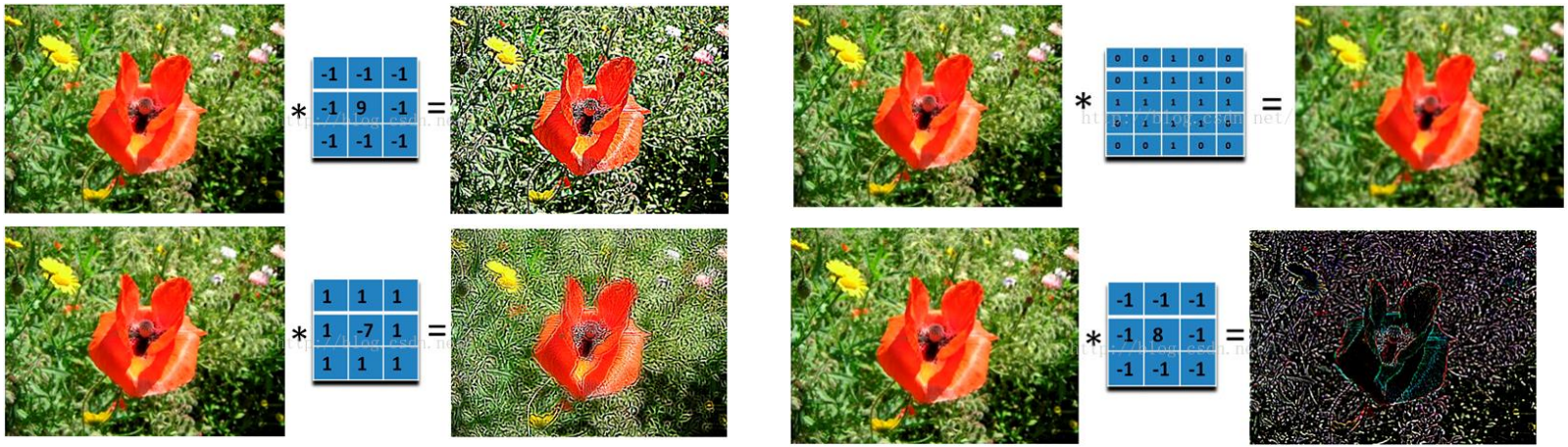
306	333	360	387	414	441	468	495
576	603	630	657	684	711	738	765
846	873	900	927	954	981	1008	1035
1116	1143	1170	1197	1224	1251	1278	1305
1386	1413	1440	1467	1494	1521	1548	1575
1656	1683	1710	1737	1764	1791	1818	1845
1926	1953	1980	2007	2034	2061	2088	2115
2196	2223	2250	2277	2304	2331	2358	2385

卷积带来的特性

- 局部相关性
  - 相邻的像素结合起来表达一个特征，距离较远的像素影响较小
  - 随着层数的增加，feature map里面的点映射到原图的感受野越来越大
  - 不仅仅可以适用于图像，其他具有局部相关性的数据均可以尝试卷积网络处理
  - 带来一定的遮挡不变性
- 平移不变性
  - 在任何位置都可以激活神经元
- 缩放不变性
  - 一定程度内的图像缩放（通常认为50%以内）不会产生太大干扰
- 少许降维
  - $(H,W)_{output} = ((H,W)_{input} - (H,W)_{kernel} + padding * 2)/stride + 1$
  - 例如：10x10的图像，用5x5的卷积核，采用valid方式（padding=0），stride=2
    输出feature map为： floor((10 – 5 + 0*2)/2) + 1 = 3 x 3

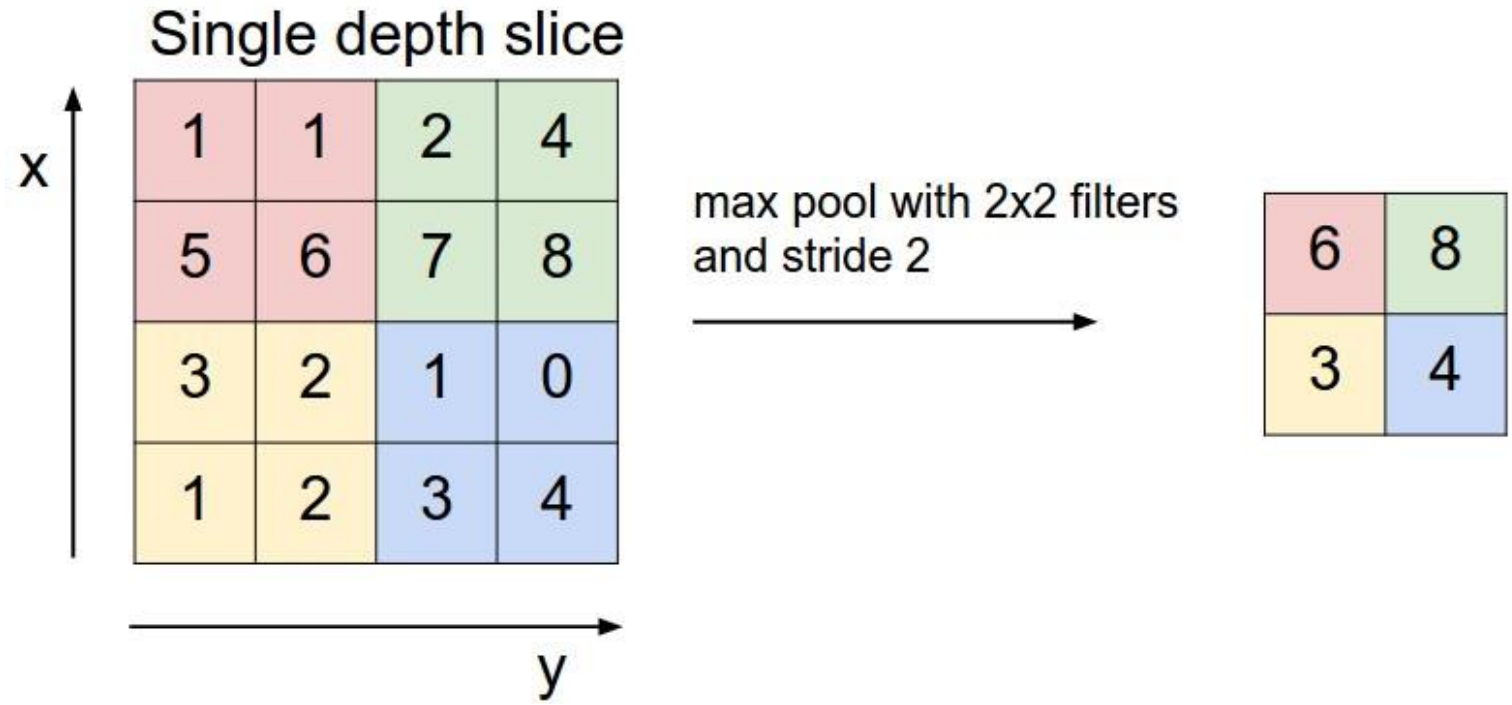
卷积与图像滤波

- 以前人们通过精心设计的卷积核来做图像运算
- 通过不同的卷积核可以起到边缘提取、锐化、模糊等效果
- 现在的卷积神经网络的卷积核很多时候也是起到同样的作用



池化（Pooling）

池化不会改变channel数



池化（pooling）

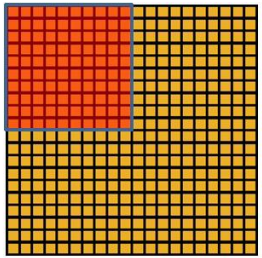
直观理解：

- 一个选择框，将输入数据某个范围（矩阵）的所有数值进行相应计算，得到一个新的值，作为结果的一个像素点
- 池化也有步长和补齐的概念，但是很少使用，通常选择框以不重叠的方式，在padding=0的输入数据上滑动，生成一张新的特征图

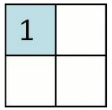
又称为降采样（down_sampling）

池化的类型：

- 最大池化（max pooling）
  在感受野内取最大值输出
- 平均池化（average pooling）
  将感受野内去平均值进行输出
- 其他如L2池化等



Convolved feature



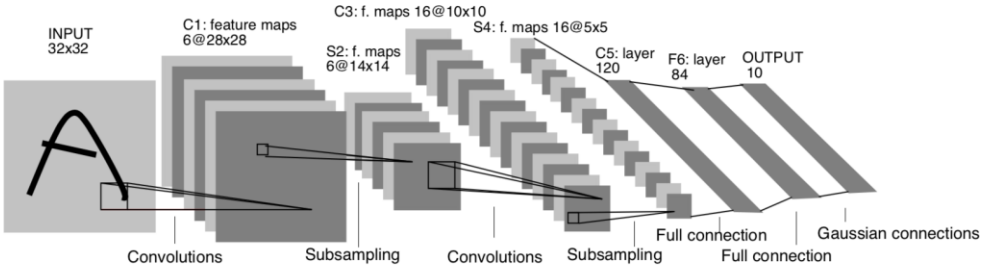
Pooled feature

池化带来的特性

- 平移不变性
- 旋转不变性
- 缩放不变性
- 降维数据

卷积神经网络

- LeNet-5
  gradient-based learning applied to document recognition(1998)
- 卷积神经网络的开端
- 使用了复杂的局部连接
- 当时并没有使用softmax和交叉熵，而是使用了欧式径向基函数和均方误差



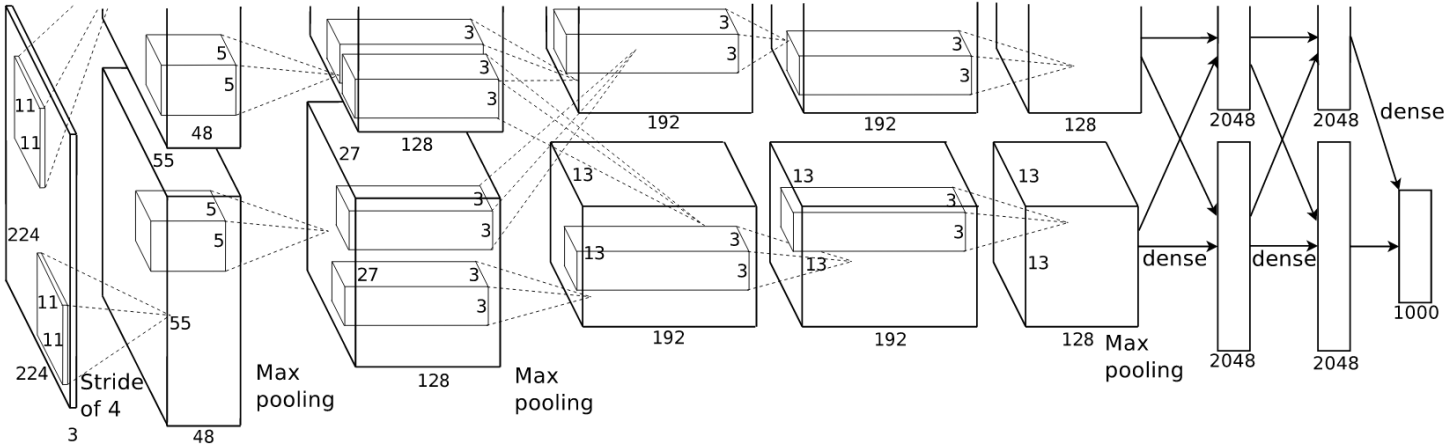
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

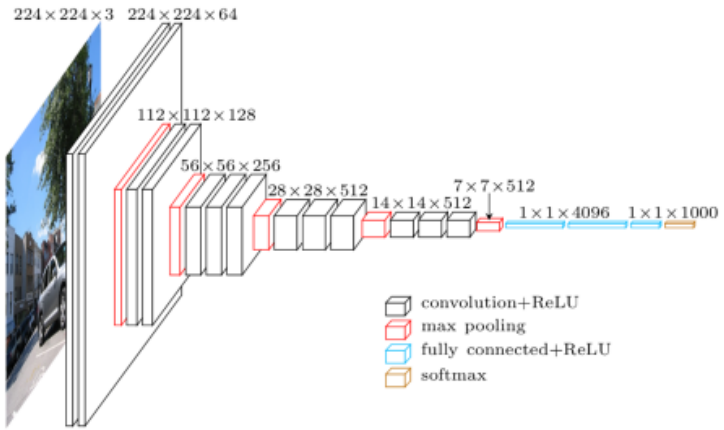
卷积神经网络

- Alexnet
  imagenet classification with deep convolutional neural networks（2012）
- 采用了较大的深度
- 将网络从depth维度拆成了两份进行训练
- dropout



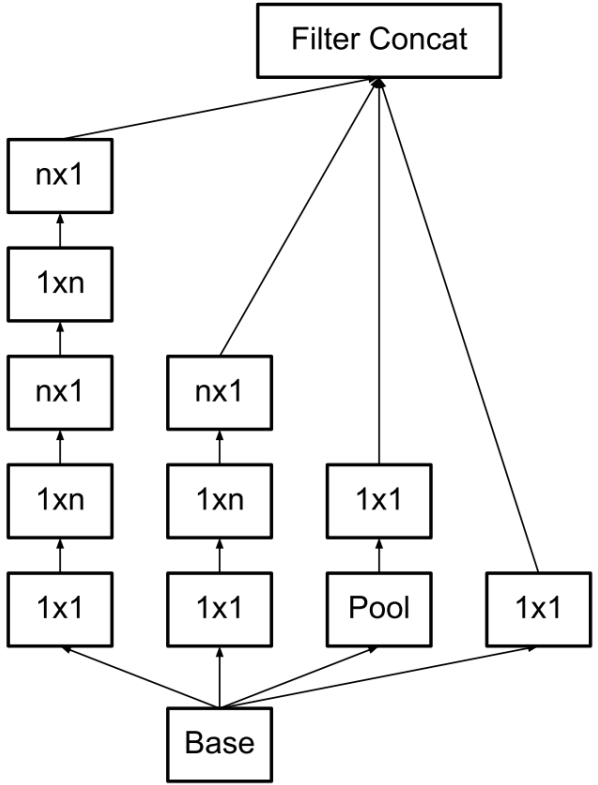
卷积神经网络

- VGG
  Very Deep Convolutional Networks for Large-Scale Image Recognition(2014), arXiv:1409.1556
- 结构简洁
- 进一步提高了深度
- 简单的结构使其权重用途广泛



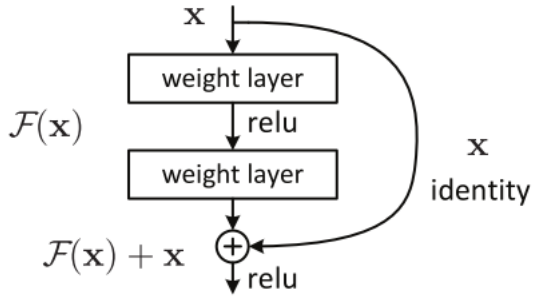
卷积神经网络

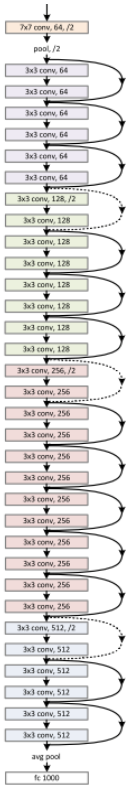
- Inception
  arXiv:1512.00567
- branch结构
- 1x1卷积进行降维
- 非对称卷积结构
- 以较少的权重实现了更好的效果



卷积神经网络

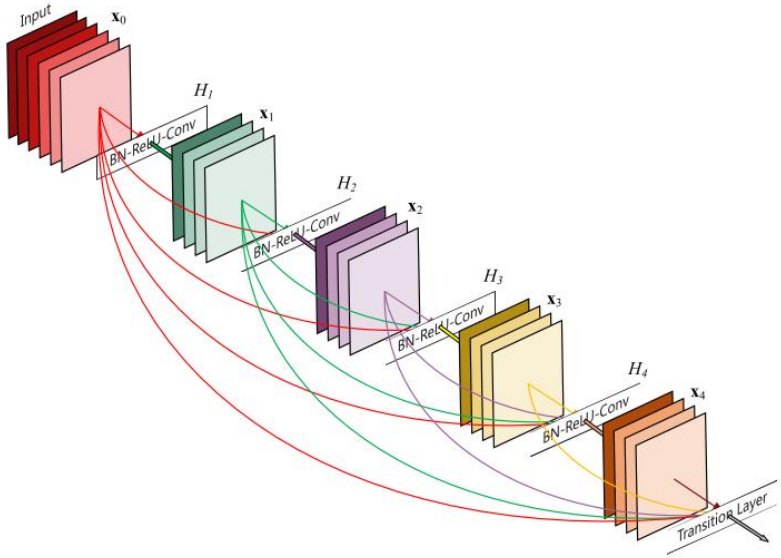
- Deep residual network（ResNet）
  arXiv：1512.03385
- 残差block
- 堆叠式结构
- 深度的突破和终结





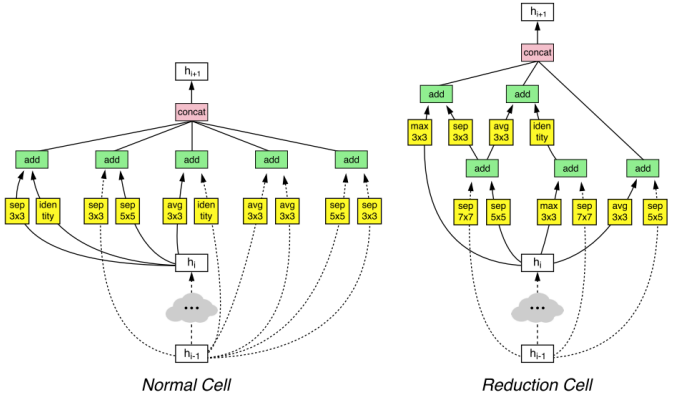
DenseNet

- DenseNet
  arXiv:1608.06993
- 博采众长
- 1x1 bottleneck
- Block+Branch
- 可扩展的设计
- 可能还不够优秀，但是展示了一个新的方向



卷积神经网络

- NasNet
  arXiv:1707.07012
- 神经网络设计的神经网络
- 具备较少权重的同时提供更好的性能



卷积神经网络的反向传播

回顾一下全连接网络的反向传播：

$$\delta_i^{(L)} = \nabla_y Cost \times \sigma'(logit_i^{(L)}) \qquad (BP1)$$

$$\delta^{(l)} = \delta^{(l+1)} w^{(l+1)} \sigma'(logit^{(l)}) \qquad (BP2)$$

$$\frac{\partial Cost}{\partial bias_i^{(l)}} = \delta_i^{(l)} \qquad (BP3)$$

$$\frac{\partial Cost}{\partial w_{ij}^{(l)}} = h_j^{(l-1)} \delta_i^{(l)} \qquad (BP4)$$

推导过程中使用了

对于卷积神经网络，

$$output^{(l+1)} = \sigma(W \cdot output^{(l)} + b)$$

$$output^{(l+1)} = \sigma(output^{(l)} * W + b) \qquad \#convoluting$$

$$output^{(l+1)} = subsampling(output^{(l)} + b) \qquad \#pooling$$

卷积神经网络的BP

$$\delta_i^{(L)} = \nabla_y Cost \times \sigma'(logit_i^{(L)}) \quad (BP1)$$
$$\delta^{(l)} = \delta^{(l+1)} w^{(l+1)} \sigma'(logit^{(l)}) \quad (BP2)$$
$$\frac{\partial Cost}{\partial bias_i^{(l)}} = \delta_i^{(l)} \quad (BP3)$$
$$\frac{\partial Cost}{\partial w_{ij}^{(l)}} = h_j^{(l-1)} \delta_i^{(l)} \quad (BP4)$$

BP1仅和损失函数、激活函数有关，不受卷积操作的形态影响

BP2的推导过程中涉及到了 $output^{(l+1)} = \sigma(W \cdot output^{(l)} + b)$

现在我们使用卷积和池化的公式重新进行推导

$$output^{(l+1)} = \sigma(output^{(l)} * W + b) \qquad \#convoluting$$
$$output^{(l+1)} = subsampling(output^{(l)} + b) \qquad \#pooling$$

首先回忆一下定义： $\delta^{(l)} = \frac{\partial Cost}{\partial logits^{(l)}}$

$$
\begin{aligned}
output^{(l+1)} &= \sigma(output^{(l)} * W + b) \\
\delta^{(l)} &= \frac{\partial Cost}{\partial logit^{(l)}} \\
&= \frac{\partial Cost}{\partial logits^{(l+1)}} \cdot \frac{\partial logits^{(l+1)}}{\partial logits^{(l)}} \\
&= \delta^{(l+1)} \cdot \frac{\partial (output^{(l)} * W + b)}{\partial logits^{(l)}} \\
&= \delta^{(l+1)} \cdot \frac{\partial (output^{(l)} * W + b)}{\partial output^{(l)}} \cdot \frac{\partial output^{(l)}}{\partial logits^{(l)}} \\
&= \delta^{(l+1)} \cdot \frac{\partial (output^{(l)} * W + b)}{\partial output^{(l)}} \cdot \sigma'(logits^{(l)})
\end{aligned}
$$

接下来就是计算 $\frac{\partial (output^{(l)} * W + b)}{\partial output^{(l)}}$

$$\frac{\partial(output^{(l)} * W + b)}{\partial output^{(l)}}$$

$$\begin{pmatrix} o_{11} & o_{12} & o_{13} \\ o_{21} & o_{22} & o_{23} \\ o_{31} & o_{32} & o_{33} \end{pmatrix} * \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} = \begin{pmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{pmatrix}$$

$$l_{11} = o_{11}w_{11} + o_{12}w_{12} + o_{21}w_{21} + o_{22}w_{22}$$
$$l_{12} = o_{12}w_{11} + o_{13}w_{12} + o_{22}w_{21} + o_{23}w_{22}$$
$$l_{21} = o_{21}w_{11} + o_{22}w_{12} + o_{31}w_{21} + o_{32}w_{22}$$
$$l_{22} = o_{22}w_{11} + o_{23}w_{12} + o_{32}w_{21} + o_{33}w_{22}$$

$$\frac{\partial(output^{(l)} * W + b)}{\partial output^{(l)}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{bmatrix}$$

$$\frac{\partial l}{\partial o_{11}} = w_{11}$$
$$\frac{\partial l}{\partial o_{12}} = w_{11} + w_{12}$$
$$\dots$$
$$\frac{\partial l}{\partial o_{22}} = w_{11} + w_{12} + w_{21} + w_{22}$$
$$\dots$$
$$\frac{\partial l}{\partial o_{33}} = w_{22}$$

$$\delta^{(l)} = \delta^{(l+1)} * rot180(W) \odot \sigma'(logits^{(l)})$$

→新的BP2

$$\frac{\partial Cost}{\partial bias_i^{(l)}} = \delta_i^{(l)} \qquad (BP3)$$

对于全连接网络 $b, \delta^{(l)} \in \mathbb{R}(n_{out}^{(l)})$

对于卷积网络

$$b \in \mathbb{R}(n_{out}^{(l)})$$

$$\delta^{(l)} \in \mathbb{R}(H_{out}^{(l)}, W_{out}^{(l)}, n_{out}^{(l)})$$

处理方式为将 $\delta^{(l)}$ 的H和W维度求和

$$\frac{\partial Cost}{\partial bias_i^{(l)}} = \sum_H \sum_W (\delta_{(H,W,i)}^{(l)}) \qquad (BP3)$$

$$\frac{\partial Cost}{\partial w^{(l)}} = h^{(l-1)}\delta^{(l)} \qquad (BP4)$$

这里的计算方法和BP2的推导过程一样，也是通过链式法则，将其中涉及 $W \cdot X + b$ 的部分改为用 $X * W + b$ 做同样的推导。

得到的结果也是类似的。

$$\frac{\partial Cost}{\partial w^{(l)}} = rot180(\delta^{(l)} * h^{(l-1)}) \qquad (BP4)$$

池化层的反向传播

对于max pooling，需要记录下最大池化时的最大值坐标位置，并反向填充

例如：

$$input^{(l)} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 2 & 1 \\ 3 & 1 & 1 & 2 \\ 4 & 2 & 1 & 3 \end{bmatrix}, max_pooling(input^{(l)}) = \begin{bmatrix} 3_{(2,2)}, 4_{(1,2)} \\ 4_{(2,1)}, 3_{(2,2)} \end{bmatrix}$$

若：$\delta^{(l)} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ $upsampling(\delta^{(l)}) = \begin{bmatrix} 0 & 0 & 0 & 6 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 8 \end{bmatrix}$

对于average pooling，需要将 $\delta^{(l)}$ 平均

例如：

$$\delta^{(l)} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \quad upsampling(\delta^{(l)}) = \begin{bmatrix} 1.25 & 1.25 & 1.5 & 1.5 \\ 1.25 & 1.25 & 1.5 & 1.5 \\ 1.75 & 1.75 & 2 & 2 \\ 1.75 & 1.75 & 2 & 2 \end{bmatrix}$$

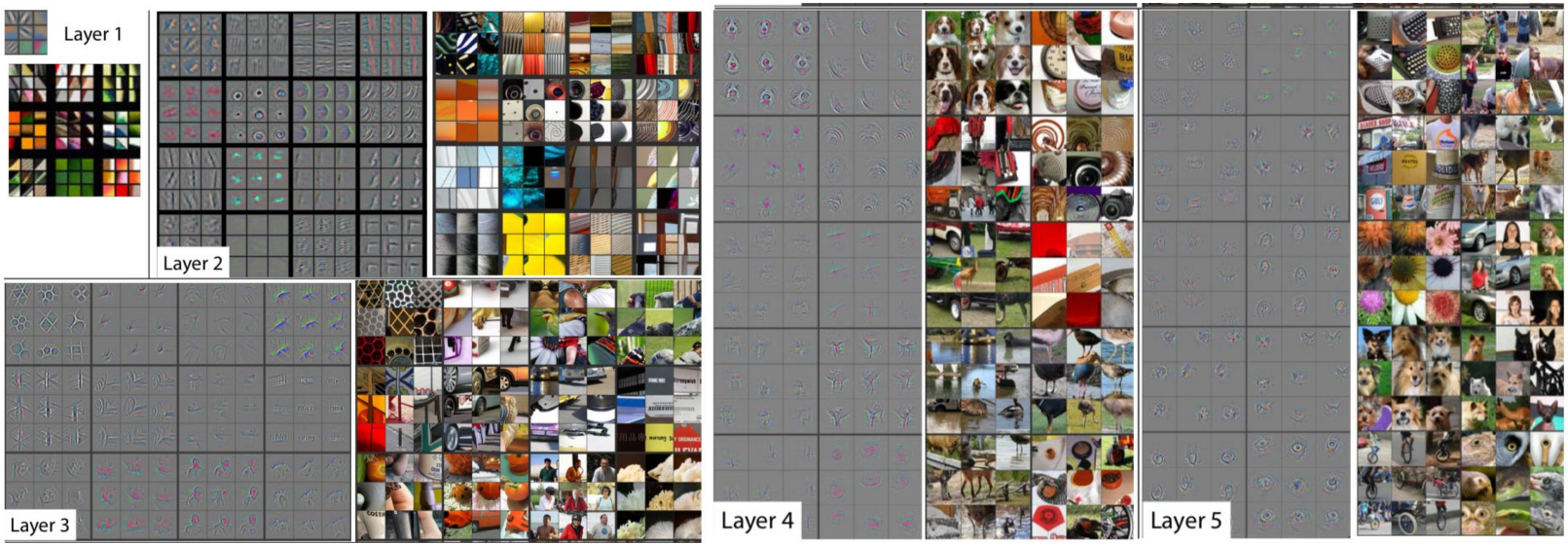
而 $\delta^{(l-1)} = upsample(\delta_k^{(l)}) \odot \sigma'(logit^{(l-1)})$

注意*l*为池化层，池化层自身无激活函数

*(l-1)*层可能不是池化层

卷积神经网络到底学到了什么

- Visualizing and Understanding Convolutional Networks arXiv:1311.2901



新技术的出现

Batch Normalization
arXiv：1502.03167

加快网络收敛速度

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}\right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

In $N_{\mathrm{BN}}^{\mathrm{inf}}$, replace the transform $y = \mathrm{BN}_{\gamma,\beta}(x)$ with

$$y = \frac{\gamma}{\sqrt{\mathrm{Var}[x] + \epsilon}} \cdot x + \left(\beta - \frac{\gamma\, \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}}\right)$$

$$\mathrm{E}[x] \leftarrow \mathrm{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$

$$\mathrm{Var}[x] \leftarrow \frac{m}{m-1}\mathrm{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

新技术的出现

- Dropout
  imagenet classification with deep convolutional neural networks
  Improving neural networks by preventing co-adaptation of feature detectors(arXiv:1207.0580

- 训练时以几率p将神经元的输出置为0
- 改善过拟合情况
- 可能会减缓收敛速度
- 不会对训练和预测速度有太大影响

Dropout的实现要点：

- 一般是实施在分类器之前
  - 论文是放在最后一层分类器之后
- Dropout以概率p置零神经元，这种情况下，保留的神经元的输出要除以(1-p)
  - 论文是在inference时把所有权重乘以p
- 通常p初始取值0.5

Augmentation

- 数据的预处理
  - Normalization
    - Vgg style
    - Inception style
  - Augmentation
    - Flip
    - Crop
    - Brightness/Contrast/Saturation/Hue
    - Rotation

finetune

- 和迁移学习（transfer learning）的关系
- finetune的方式
  - 使用同一网络，用预训练的权重来初始化网络
  - 使用不同网络，但截取预训练的网络权重的一部分作为网络的前置
    - 冻结权重
    - 不冻结权重
- Model Zoo

Tensorflow训练一个卷积神经网络

- Tensorflow原生写法
- Tensorflow layer
- Tensorflow Keras
- Tensorflow Slim

Talk is cheap

Just show me the CODE！

CSDN
不 止 于 代 码

THANK YOU

AI100