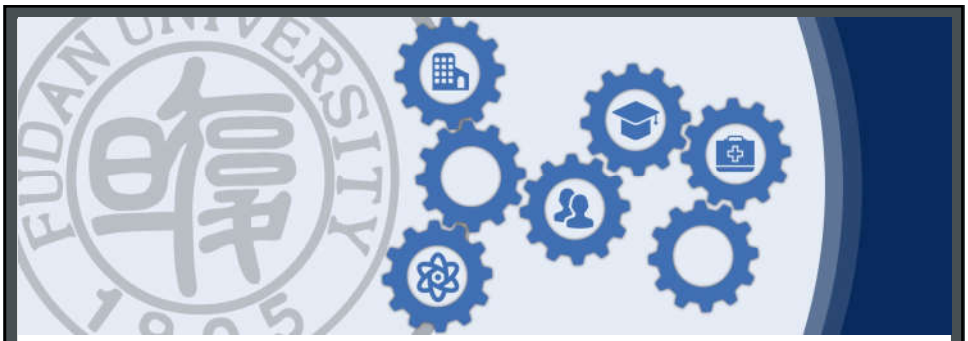


智能系统原理与开发
第12章 CNN与目标检测

李斌

复旦大学 计算机科学技术学院

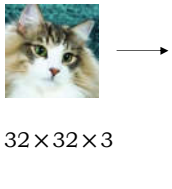
总览

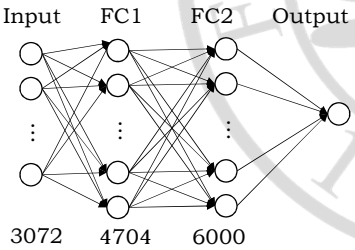
卷积神经网络简介

- 为什么要用卷积神经网络
- 卷积神经网络基本结构

为什么要用卷积神经网络

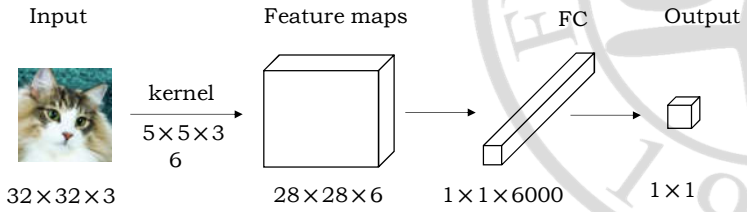
- 全连接网络有什么问题?
  - 两层全连接网络的图像分类
    - 首先将32×32×3的图像展开成长度为3072列向量，然后输入到网络中
    - Input映射到FC1网络参数为3072×4704≈14M
    - 全连接网络参数数量太大，容易出现过拟合





为什么要用卷积神经网络

- 为什么要用卷积神经网络?
  - 一个卷积层的卷积神经网络图像分类
    - Feature maps中神经元总数为28×28×6=4704
    - Input经kernel映射到Feature maps的参数为5×5×3×6=450



为什么要用卷积神经网络

- 为什么要用卷积神经网络?
  - 对比两个网络中第一层连接的参数数量
    - 输出神经元大小相同，CNN的参数数量是DNN约三万分之一
    - 网络结构固定，输入图片尺寸越大DNN网络的参数越多
    - 网络结构固定，输入图片尺寸不管增大多少，CNN网络参数不变
    - CNN网络结构极大地减少了网络参数数量

$$N^{DNN} = 32 \times 32 \times 3 \times 4074 = 14450688$$

$$N^{CNN} = 5 \times 5 \times 3 \times 6 = 450$$

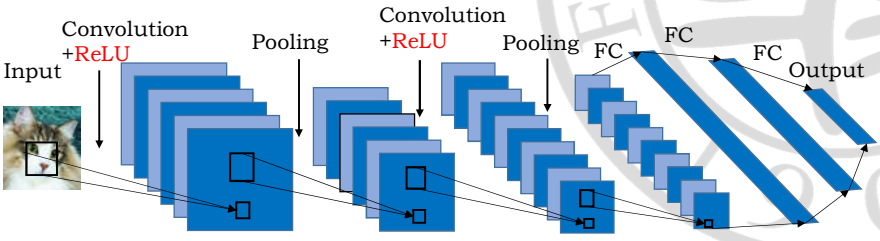
$$\frac{N^{DNN}}{N^{CNN}} = 32112.64 \approx 3万$$

卷积神经网络简介

- 为什么要用卷积神经网络
- 卷积神经网络基本结构

卷积神经网络基本结构

- 卷积神经网络基本结构
  - 一个完整的经典卷积神经网络由输入层、卷积层、池化层、全连接层和输出层组成
    - 卷积层由多个卷积核组成，对输入层进行特征提取
    - 池化层的作用是清除冗余特征，压缩特征
    - 全连接层将卷积和池化层提取到的所有特征进行汇总，输入到分类器中

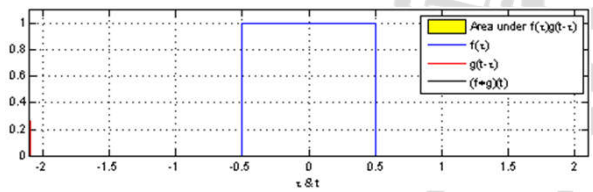


卷积运算与池化操作

- 卷积函数
- 卷积运算
- 卷积核特性
- 池化操作

卷积函数

- 什么是卷积?
  - 连续情况：$s(t) = \int f(\tau)g(t-\tau)d\tau$
  - 写成卷积形式：$s(t) = (f * g)(t)$，*表示卷积运算



  - 离散情况：$s(t) = \sum x(a)w(t-a)$
  - 卷积操作就是对输入函数的每一个位置进行加权累加

卷积函数

- 卷积滤波
  - 卷积是图像处理中一种常用的线性滤波方法
  - 使用卷积可以达到图像的降噪、锐化等多种滤波效果
  - 二维图片I和二维卷积滤波矩阵K的卷积操作如下

$$s(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

  - 深度学习中的卷积运算为

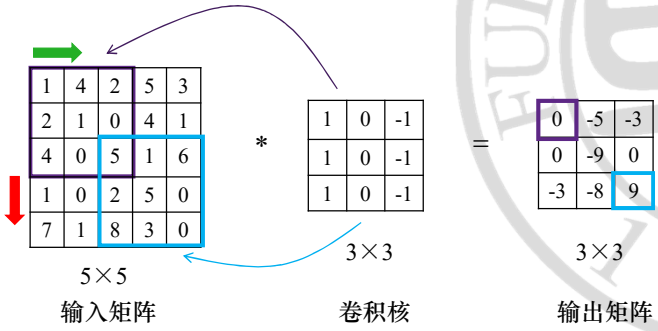
$$s(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

卷积运算与池化操作

- 卷积函数
- 卷积运算
- 卷积核特性
- 池化操作

卷积运算

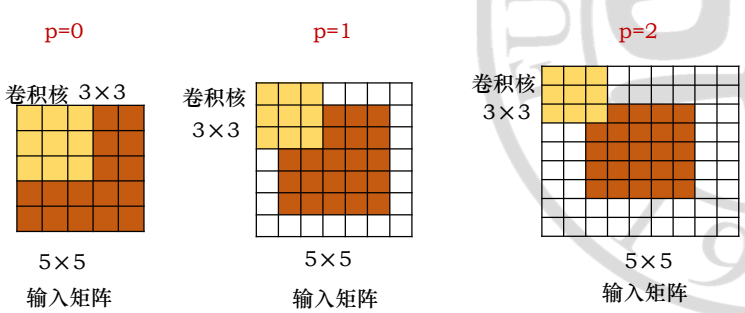
- 卷积运算过程
  - 卷积运算规则：卷积核与输入矩阵中对应的位置相乘再累加
  - 对应位置移动方式为自左向右，自上而下



5×5 输入矩阵　　3×3 卷积核　　3×3 输出矩阵

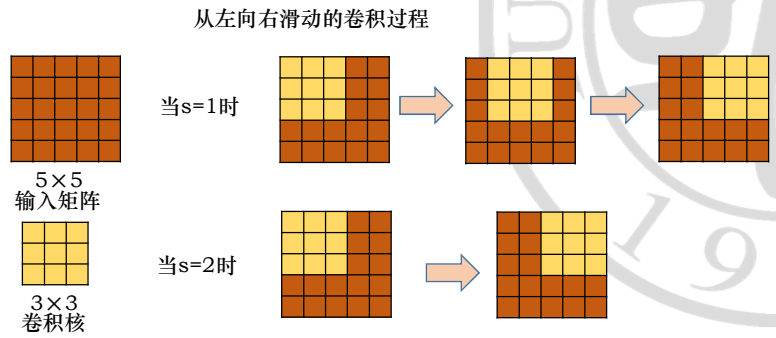
卷积过程中的重要超参数

- Padding
  - 表示对输入矩阵外围填充0
  - Padding的值表示填充行/列的数目，通常记做p
  - 当p=0时表示不进行填充
  - Padding可取值的范围是[0,f-1], f为卷积核大小



卷积过程中的重要超参数

- Stride
  - 表示卷积核滑动的步长，即移动时跳过的像素数目
  - Stride通常用s表示
  - Stride的取值范围一般为$[1, f]$,其中f为卷积核大小



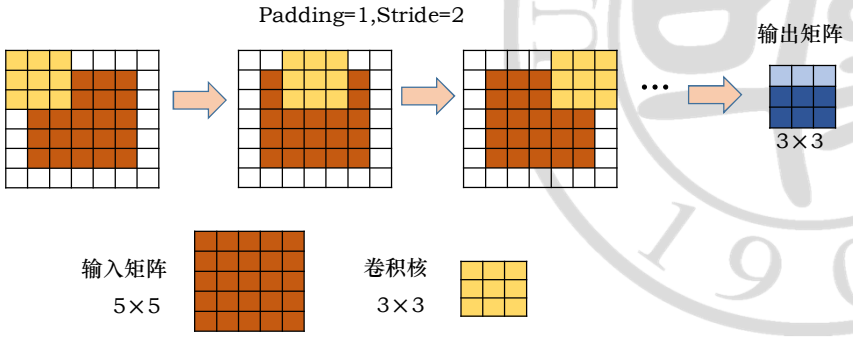
卷积过程中的重要超参数

- 输出矩阵大小计算
  - 输入矩阵大小为：$n \times n$
  - 卷积核大小为：$f \times f$
  - 卷积核滑动步长stride为：s
  - 卷积核填充数padding为：p
  - 输出矩阵大小为：$m \times m$,其计算方式为：

$$m = \begin{cases} \left(\frac{(n-f+2p)}{s} + 1\right), \text{当}(n-f+2p)\text{可以被}s\text{整除} \\ \left\lfloor \frac{(n-f+2p)}{s} + 1 \right\rfloor, \text{当}(n-f+2p)\text{不可以被}s\text{整除} \end{cases}$$

卷积过程中的重要超参数

- 输出矩阵大小计算
  - 卷积核向左移动时，卷积运算得到的向量长度为输出矩阵的宽度
  - 输出矩阵宽度=(5-3+2×1)/2+1=3
  - 同理，可知输出矩阵高度=宽度=3

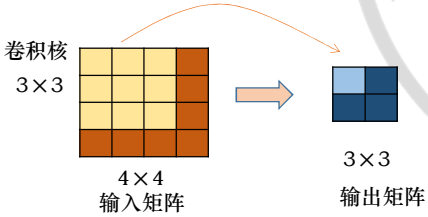


卷积操作常用的几种类型

- Valid操作
  - 输入矩阵大小为$n \times n$,卷积核大小为$f \times f$, Padding为0，Stride为s,输出矩阵大小为$m \times m$,则:

$$m = \left(\frac{(n-f)}{s} + 1\right)$$

  - 输出矩阵尺寸小于输入矩阵的尺寸

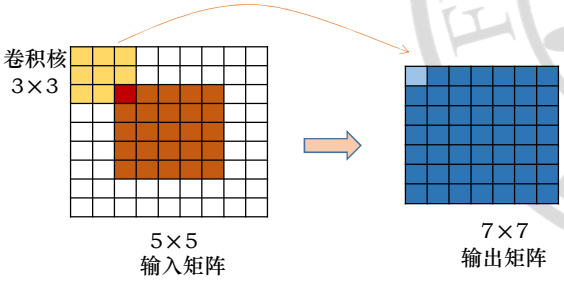


卷积操作常用的几种类型

- Full操作
  - 输入矩阵大小为$n \times n$,卷积核大小为$f \times f$, Padding为$f-1$，Stride为1,输出矩阵大小为$m \times m$，则：

$$m = (n + f - 1)$$

  - 输出矩阵尺寸大于输入尺寸

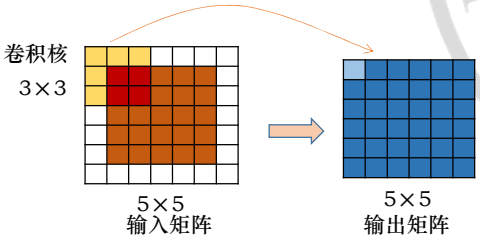


卷积操作常用的几种类型

- Same操作
  - 输入矩阵大小为$n \times n$,卷积核大小为$f \times f$, Padding为$(f-1)/2$，Stride为1,输出特征大小为$m \times m$,则：

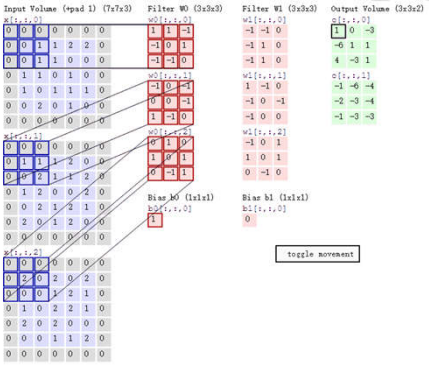
$$m = \left(\frac{(n - f + 2p)}{s} + 1\right) = n$$

  - 输出矩阵尺寸等于输入矩阵尺寸



卷积运算

- 输入矩阵有多个通道的卷积运算
  - 输出矩阵的通道数，等于卷积核的个数



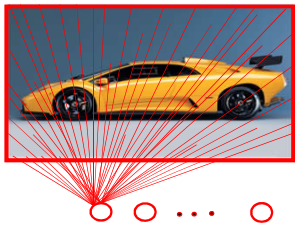
https://blog.csdn.net/v_july_v/article/details/51812459

卷积运算与池化操作

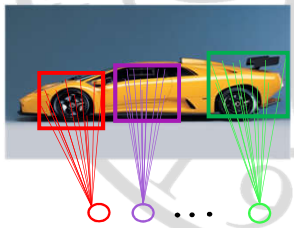
- 卷积函数
- 卷积运算
- 卷积核特性
- 池化操作

卷积核特性

- 局部感知
  - 感受野：计算输出矩阵中每一个元素时所对应的输入矩阵中区域的大小
  - 全连接网络的感受野是整个输入矩阵，CNN的感受野是卷积核的大小区域
  - CNN中，每个神经元只需要对局部信息进行感知，然后在更高层将局部的信息综合起来得到更大区域信息，随着卷积层数的增加，获得的特征信息越来越全面



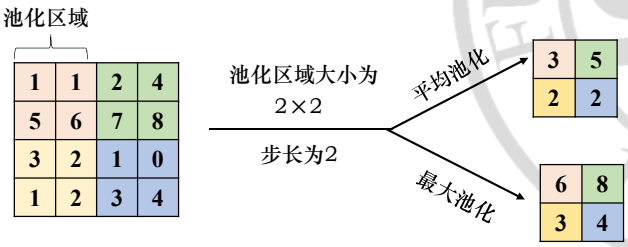
全连接网络



局部连接网络(卷积神经网络)

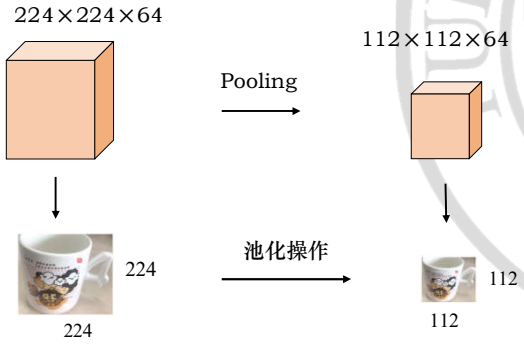
池化操作

- 池化操作
  - 在一定区域内（池化区域）进行采样
    - 通常是取最大值（最大池化）或平均值（平均池化）
    - 池化区域大小通常为2×2
    - 输入矩阵的每个通道单独进行池化操作
    - 池化操作的输出矩阵通道数和输入矩阵通道数相同



池化的作用

- 池化对图片主要有以下三个作用
  - 特征压缩，把图像中的冗余信息去除，提取出最重要的特征
  - 使特征图变小，简化网络计算，在一定程度上防止过拟合
  - 具有平移不变性



卷积神经网络正反向传播

- 向前传播
- 反向传播

向前传播

■ 第l层为卷积层

$$\boldsymbol{z}^l = \boldsymbol{h}^{l-1} * \boldsymbol{k}^l + \boldsymbol{b}^l$$

$$\boldsymbol{h}^l = \sigma(\boldsymbol{z}^l)$$

□ 其中,σ为激活函数，$\mathbf{z}^l$为第l层的中间变量，$\boldsymbol{h}^{l-1}$为第l层卷积层的输入。 $\boldsymbol{h}^l$为第l 层的输出,$\boldsymbol{k}^l$ 为第l层卷积核, $\boldsymbol{b}^l$为偏置, *为卷积运算符号

■ 第l层为池化层

$$\boldsymbol{h}^l = MaxPooling(\boldsymbol{h}^{l-1})$$

□ 以最大池化为例

向前传播

■ 第l层为全连接层

$$\boldsymbol{z}^l = (\boldsymbol{W}^l)^T \boldsymbol{h}^{l-1} + \boldsymbol{b}^l$$

$$\boldsymbol{h}^l = \sigma(\boldsymbol{z}^l)$$

□ $\boldsymbol{h}^{l-1}$表示第l层的输入矩阵，$\mathbf{z}^l$表示第l层中间变量， $\boldsymbol{h}^l$表示第l层的输出矩阵，$\mathbf{W}^l$ 和 $\mathbf{b}^l$表示第l层的权重矩阵和偏置向量

■ 第l层为输出层

$$\mathbf{y} = softmax(\boldsymbol{h}^{l-1})$$

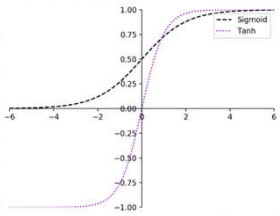
$$C = -\sum_i \hat{y}_i \, ln y_i$$

□ C表示交叉熵损失函数，$\hat{y}_i$表示标签， y_i表示网络的输出结果

向前传播

- 激活函数
  - sigmoid：可以用来拟合概率，导数计算简单，但容易出现“梯度消失”问题，常用于全连接层

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

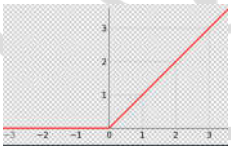


  - tanh：收敛速度比较快，常用于全连接层

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

  - ReLU：分类效果好，收敛速度快，计算速度快，避免了“梯度消失”问题，常用于卷积层

$$ReLU(x) = max\,(0, x)$$



图片来源：百度百科

卷积神经网络正反向传播

- 向前传播
- 反向传播

反向传播

■ 全连接层梯度计算

□ CNN中全连接梯度计算与DNN相同，由上一章结论直接得第l层误差为

$$\boldsymbol{\delta}^l = \frac{\partial C}{\partial \boldsymbol{z}^l} = \begin{cases} \left(\left(\boldsymbol{w}^{l+1}\right)^T \boldsymbol{\delta}^{l+1}\right) \odot \ \sigma'\left(\boldsymbol{z}^l\right), & l < L \\ \nabla_y C \odot \sigma'\left(\boldsymbol{z}^L\right), & l = L \end{cases}$$

其中，$\boldsymbol{\delta}^l$是一个长度等于该层神经元数量的向量，其中L层为输出层的前一层，$\nabla_y C$表输出层的误差

□ 第l层权重以及偏置的偏导数为

$$\frac{\partial C}{\partial \boldsymbol{w}^l} = \boldsymbol{\delta}^l \times \left(\boldsymbol{h}^{l-1}\right)^T$$

$$\frac{\partial C}{\partial \boldsymbol{b}^l} = \boldsymbol{\delta}^l$$

反向传播

■ 卷积层梯度计算

□ 以一个通道为例，设第l层卷积层输入矩阵和卷积核如下

$$\boldsymbol{h}^{l-1} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix} \qquad \boldsymbol{k}^l = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}$$

□ 第l层卷积层向前传播时，输出矩阵为

$$\boldsymbol{z}^l = \boldsymbol{h}^{l-1} * \boldsymbol{k}^l + \boldsymbol{b}^l = \begin{bmatrix} \boldsymbol{z}_{11} & \boldsymbol{z}_{12} \\ \boldsymbol{z}_{21} & \boldsymbol{z}_{22} \end{bmatrix}$$

$$\boldsymbol{h}^l = \sigma\left(\boldsymbol{z}^l\right)$$

反向传播

■ 卷积层梯度计算

□ 和前馈神经网络一样，我们首先定义第l层的误差函数为：

$$\boldsymbol{\delta}^l = \frac{\partial C}{\partial \boldsymbol{z}^l}$$

□ 第l层的第i行第j列神经元误差为：

$$\delta_{ij}^l = \frac{\partial C}{\partial h_{ij}^l}\frac{\partial h_{ij}^l}{\partial z_{ij}^l} = \frac{\partial C}{\partial h_{ij}^l}\sigma'(z_{ij}^l)$$

$$= \left(\sum_m \sum_n \frac{\partial C}{\partial z_{mn}^{l+1}}\frac{\partial z_{mn}^{l+1}}{\partial h_{ij}^l}\right)\sigma'(z_{ij}^l)$$

$$= \left(\sum_m \sum_n \delta_{mn}^{l+1}\frac{\partial z_{mn}^{l+1}}{\partial h_{ij}^l}\right)\sigma'(z_{ij}^l)$$

$rot180(\boldsymbol{k}^{l+1})$

为什么等于这个将在后面解释

$$\boldsymbol{\delta}^l = \left(\boldsymbol{\delta}^{l+1} * rot180(\boldsymbol{k}^{l+1})\right) \odot \sigma'(\boldsymbol{z}^l)$$

反向传播

■ 卷积层梯度计算

□ 得到了神经元的误差，我们可以进一步计算卷积核参数的梯度：

$$\frac{\partial C}{\partial k_{ij}^l} = \sum_m \sum_n \left(\frac{\partial C}{\partial z_{mn}^l}\frac{\partial z_{mn}^l}{\partial k_{ij}^l}\right)$$

$$= \sum_m \sum_n \left(\delta_{mn}^l \frac{\partial\left(\sum_{i'=1}^f \sum_{j'=1}^f h_{m+i'-1,n+j'-1}^{l-1} \times k_{i'j'}^l + b^l\right)}{\partial k_{ij}^l}\right)$$

$$= \sum_m \sum_n \left(\delta_{mn}^l h_{m+i-1,n+j-1}^{l-1}\right)$$

□ 所以，整体的第l层卷积核的梯度为：$\frac{\partial C}{\partial \boldsymbol{k}^l} = \boldsymbol{h}^{l-1} * \boldsymbol{\delta}^l$

□ 完全一样的公式可以算出对$\boldsymbol{b}$的梯度为：$\frac{\partial C}{\partial \boldsymbol{b}^l} = \boldsymbol{\delta}^l$

反向传播

- 池化层
  - 由于池化层没有参数，在反向传播时只对误差δ的计算有影响：
  - 回顾前面卷积层的误差公式为：

$$\boldsymbol{\delta}^{l}=\left(\boldsymbol{\delta}^{l+1}*rot180\left(\boldsymbol{k}^{l+1}\right)\right)\odot\sigma'\left(\boldsymbol{z}^{l}\right)$$

  - 池化层的误差没有经过参数$\boldsymbol{k}$的作用，仅仅是池化操作本身对误差传播的路径产生影响。因此，需要一个操作让误差沿着正确路径传播，称为upsample操作：

$$\boldsymbol{\delta}^{l}=\left(upsample\left(\boldsymbol{\delta}^{l+1}\right)\right)\odot\sigma'\left(\boldsymbol{z}^{l}\right)$$

  由于池化层没有激活函数，为了形式上的统一，可以令$\sigma(z)=z$，则$\sigma'\left(\boldsymbol{z}^{l}\right)$=1

  - 下面简介一下这一操作

反向传播

- 池化层
  - 假设已知第$l+1$层的误差$\boldsymbol{\delta}^{l+1}$，需要求第$l$层的误差$\boldsymbol{\delta}^{l}$

$$\boldsymbol{\delta}^{l+1}=\begin{bmatrix}2 & 8\\ 4 & 6\end{bmatrix}$$

  - 设池化区域为2×2，且是最大值池化，那么：

$$\boldsymbol{\delta}^{l}=\begin{bmatrix}0 & 2 & 0 & 0\\ 0 & 0 & 8 & 0\\ 0 & 4 & 0 & 6\\ 0 & 0 & 0 & 0\end{bmatrix}$$

其中，2846出现的位置为池化操作时最大值的位置

  - 若以上为平均池化，那么：

$$\boldsymbol{\delta}^{l}=\begin{bmatrix}0.5 & 0.5 & 2 & 2\\ 0.5 & 0.5 & 2 & 2\\ 1 & 1 & 1.5 & 1.5\\ 1 & 1 & 1.5 & 1.5\end{bmatrix}$$

将δ的值平均分摊到平均池化设计的神经元

反向传播

■ 补充

□ 最后解释下前面为何是$rot180(\boldsymbol{k}^{l+1})$，已知：

$$\boldsymbol{h}^l=\begin{bmatrix}h_{11}&h_{12}&h_{13}\\h_{21}&h_{22}&h_{23}\\h_{31}&h_{32}&h_{33}\end{bmatrix}\qquad \boldsymbol{k}^{l+1}=\begin{bmatrix}k_{11}&k_{12}\\k_{21}&k_{22}\end{bmatrix}$$

$$\boldsymbol{z}^{l+1}=\boldsymbol{h}^l*\boldsymbol{k}^{l+1}=\begin{bmatrix}z_{11}^{l+1}&z_{12}^{l+1}\\z_{21}^{l+1}&z_{22}^{l+1}\end{bmatrix}$$ （暂且忽略偏置项）

□ 由公式可计算得到：$\delta_{ij}^l=\left(\sum_m\sum_n\delta_{mn}^{l+1}\frac{\partial z_{mn}^{l+1}}{\partial h_{ij}^l}\right)\sigma'(z_{ij}^l)$

$$\delta_{11}^l=\delta_{11}^{l+1}k_{11}\sigma'(z_{11}^l)$$
$$\delta_{12}^l=(\delta_{11}^{l+1}k_{12}+\delta_{12}^{l+1}k_{11})\sigma'(z_{12}^l)$$
$$\delta_{13}^l=\delta_{12}^{l+1}k_{12}\sigma'(z_{13}^l)$$

反向传播

■ 补充

$$\delta_{21}^l=(\delta_{11}^{l+1}k_{21}+\delta_{21}^{l+1}k_{11})\sigma'(z_{21}^l)$$
$$\delta_{22}^l=(\delta_{11}^{l+1}k_{22}+\delta_{12}^{l+1}k_{21}+\delta_{21}^{l+1}k_{12}+\delta_{22}^{l+1}k_{11})\sigma'(z_{22}^l)$$
$$\delta_{23}^l=(\delta_{12}^{l+1}k_{22}+\delta_{22}^{l+1}k_{12})\sigma'(z_{23}^l)$$
$$\delta_{31}^l=\delta_{21}^{l+1}k_{21}\sigma'(z_{31}^l)$$
$$\delta_{32}^l=(\delta_{21}^{l+1}k_{22}+\delta_{22}^{l+1}k_{21})\sigma'(z_{32}^l)$$
$$\delta_{33}^l=\delta_{22}^{l+1}k_{22}\sigma'(z_{33}^l)$$

□ 观察这九项不难得到矩阵形式的表示：

$$\boldsymbol{\delta}^l=\begin{bmatrix}0&0&0&0\\0&\delta_{11}^{l+1}&\delta_{12}^{l+1}&0\\0&\delta_{21}^{l+1}&\delta_{22}^{l+1}&0\\0&0&0&0\end{bmatrix}*\begin{bmatrix}k_{22}&k_{21}\\k_{12}&k_{11}\end{bmatrix}\odot\sigma'(\boldsymbol{z}^l)$$

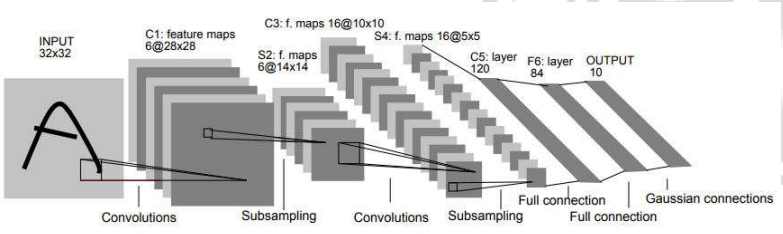
由此得到卷积核矩阵需要旋转180°的原因

经典卷积神经网络结构

- LeNet-5
- AlexNet
- VGG
- GoogLeNet
- ResNet
- 经典卷积神经网路对比

LeNet-5

- 最简单也是最早出现的卷积神经网络模型
  - 由两个卷积层，两个池化层，两个全连接层，一个高斯连接层组成
  - 所有卷积核大小都是5×5， padding=0，stride=1
  - 池化层使用Maxpooling
  - 激活函数为sigmoid和tanh函数



LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

LeNet-5

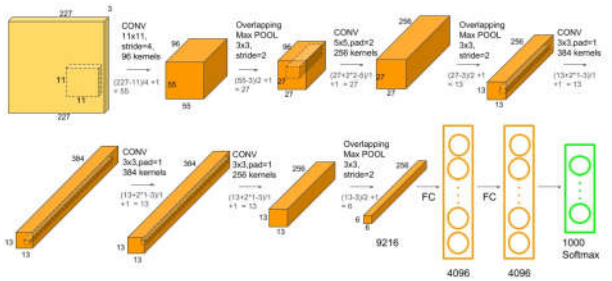
- 对于输入的32×32×1的图片：
  - 经过第一层由6个5×5×1的卷积核组成的卷积层，得到28×28×6的feature maps；
  - 经过第二层池化区域为2×2，步长为2的池化层，经过sigmoid激活函数后得到14×14×6的feature maps；
  - 经过第三层由16个5×5×6的卷积核组成的卷积层，得到10×10×16的feature maps；
  - 经过第四层池化区域为2×2，步长为2的池化层，经过sigmoid激活函数后得到5×5×16的feature maps；
  - 经过第五层由120个5×5×16的卷积核组成的卷积层，得到1×1×120的feature maps；
  - 经过第六层全连接层得到84维向量；
  - 经过第七层的RBF(Euclidean Radial Basis Function)单元得到10维向量作为输出的分类预测值。

经典卷积神经网络结构

- LeNet-5
- AlexNet
- VGG
- GoogLeNet
- ResNet
- 经典卷积神经网路对比

AlexNet

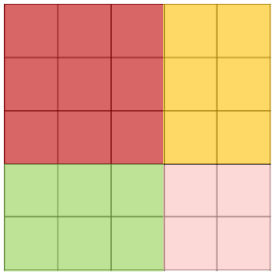
- 网络结构：5层卷积层，3层全连接层，最后softmax做分类输出
- 主要特点：
  - 首次使用ReLU激活函数和双GPU运算：大大提高训练速度
  - 采用Overlapping Pooling：提高精度模型，不容易产生过拟合
  - 使用数据扩充，Dropout等技巧：减少过拟合



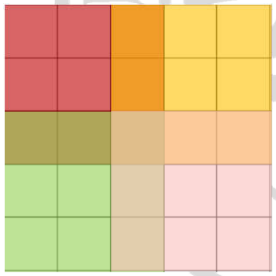
https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png

AlexNet

- 重叠池化（Overlapping Pooling）
  - 池化区域（窗口）大小大于步长的×池化方式；
  - 如正常池化时，步长s=3，池化窗口=3×3；overlappling中，步长s=2，池化窗口=3×3，有一部分区域会进行重复池化。



正常池化



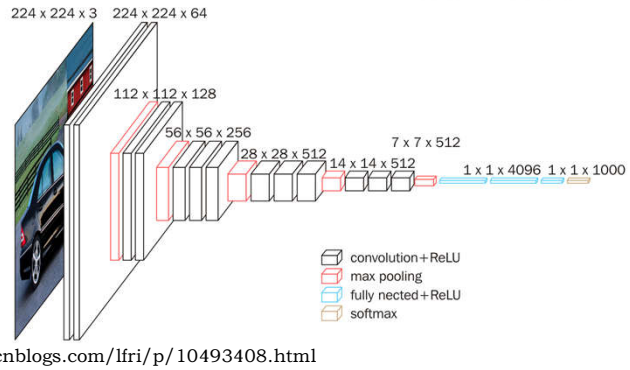
重叠池化

经典卷积神经网络结构

- LeNet-5
- AlexNet
- VGG
- GoogLeNet
- ResNet
- 经典卷积神经网路对比

VGG网络

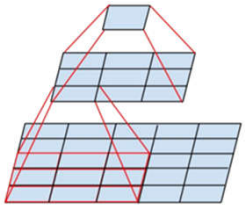
- 网络结构：13层卷积层、3层全连接层以及一个softmax层
  - 创新：所有卷积层都使用3x3卷积核，并且卷积的步长为1
  - 优势：卷积核尺寸小，参数更少，网络计算更简单，网络深度更深
  - 特点：特征图的空间分辨率单调递减，特征图的通道数单调递增



https://www.cnblogs.com/lfri/p/10493408.html

VGG网络

- 使用2个3×3卷积核来替代1个5×5的卷积核：
  - 对于5×5的卷积核，可将特征图中5×5的区域直接映射到1×1大小；
  - 而这个过程可以用两个步长为1的3×3卷积核堆叠来替代；
  - 两个3×3卷积核总参数为3×3+3×3=18，而一个5×5卷积核参数为5×5=25，可见这样在增加网络深度的同时反而减少了网络参数。



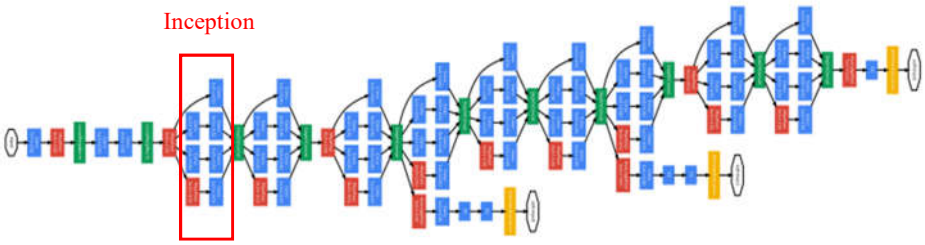
Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.

经典卷积神经网络结构

- LeNet-5
- AlexNet
- VGG
- GoogLeNet
- ResNet
- 经典卷积神经网路对比

GoogLeNet

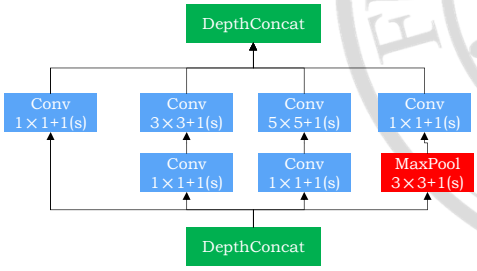
- GoogLeNet
  - 出发点：增加神经网络的宽度和深度，得到更好的模型
  - 出现的问题：计算量增加，并且容易出现过拟合
  - 设计角度：减少参数，提高训练速度



Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

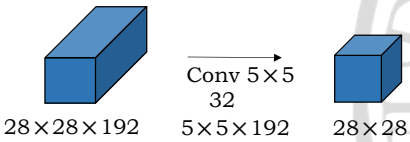
GoogLeNet

- Inception模块
  - 包含1×1、3×3和5×5三种大小的卷积核以及一个池化操作
  - 模块中的MaxPool操作不改变输入矩阵大小的特殊池化
  - 模块中三种卷积核卷积运算均不改变输入矩阵大小
  - 使用1×1卷积降低网络参数，大大减少模型计算量
  - 能够获得更丰富的特征，使得模型效果更好



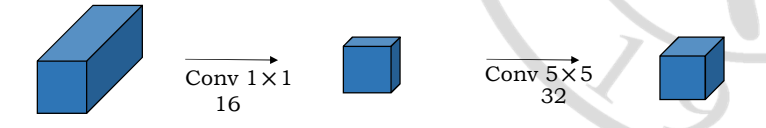
GoogLeNet

- 1×1卷积降维
  - 在3×3和5×5卷积前先进行1×1卷积可以大大减少计算量和网络参数
    - 未加1×1卷积的情况



运算次数：28×28×32×5×5×192=120422400≈120M

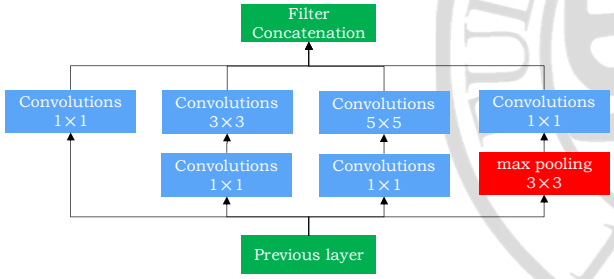
- 加上1×1卷积的情况



运算次数：1×1×192+28×28×32×5×5×16=10035392≈10M

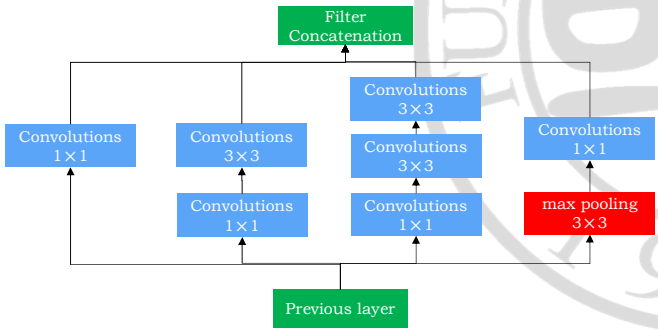
Inception类型

- Inception v1
  - 最早的Inception结构



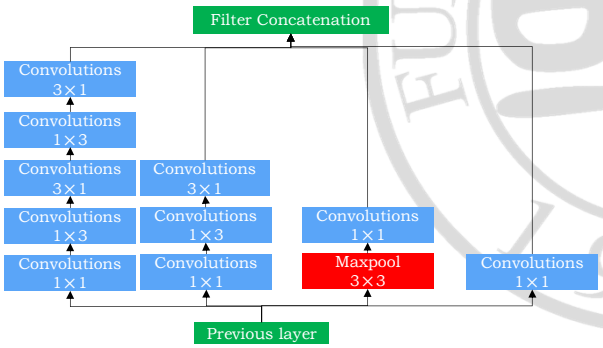
Inception类型

- Inception v2
  - 两个3×3的卷积层和一个5×5的卷积层具有相似的效果
  - 不仅减少了参数，还产生了更多的非线性转换，增加模型表达能力
  - 用Batch Normalization代替Dropout和LRN，大幅提升训练速度和精度



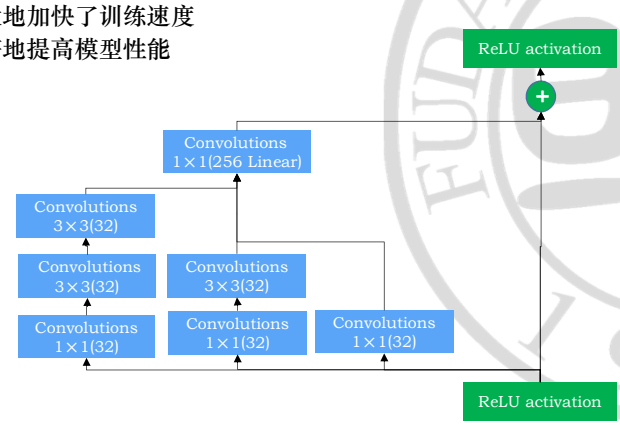
Inception结构

- Inception v3
  - 因式分解成更小的卷积：3×3分解成1×3和3×1
  - 再次减少网络参数，加速训练，降低过拟合的可能性
  - 增加了一层非线性操作，扩展了模型学习能力
  - 非对称卷积结构更有效地处理丰富的空间特征，特征的多样性被大大提高



Inception结构

- Inception v4
  - 和Residual Connection结合
  - 极大地加快了训练速度
  - 显著地提高模型性能

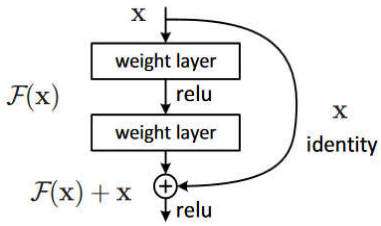


经典卷积神经网络结构

- LeNet-5
- AlexNet
- VGG
- GoogLeNet
- ResNet
- 经典卷积神经网路对比

ResNet

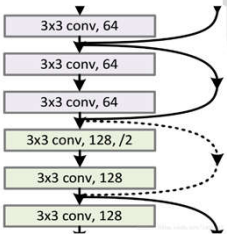
- 提出背景
  - 随着神经网络层数的增加，会出现“梯度消失”问题，网络难以训练
  - 解决办法：将l层的输入与l+1层的输出相加，当作l+1层的输出,反向传播时，因为$\frac{d(F(x)+x)}{dx}=\frac{d(F(x))}{dx}+1$，因此很大程度上避免“梯度消失”的情况
- Residual block



He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

ResNet

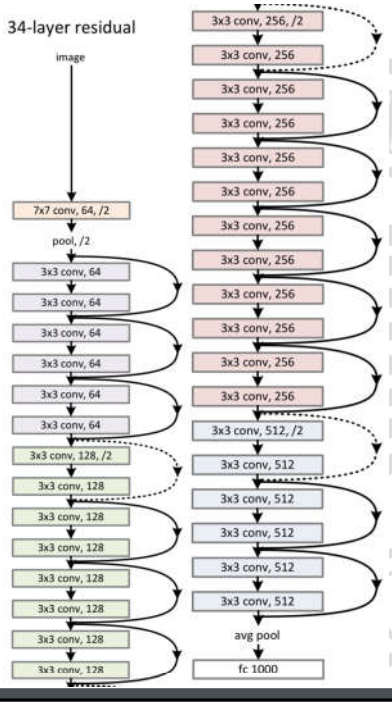
- F(x)和x按通道维度相加，但有时会出现维度不一致的情况
  - 空间上维度（特征图尺寸）不一致
    - 使用线性映射：y=F(x)+w*x
  - 深度上维度（通道数）不一致
    - 将x用0填充使其通道数与F(x)相同（一般F(x)的维度大于x的维度）
    - 使用1×1的卷积核调整x的通道数



He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

ResNet

■ ResNet网络整体结构图



He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

经典卷积神经网络结构

■ LeNet-5

■ AlexNet

■ VGG

■ GoogLeNet

■ ResNet

■ 经典卷积神经网络对比

经典卷积神经网络对比

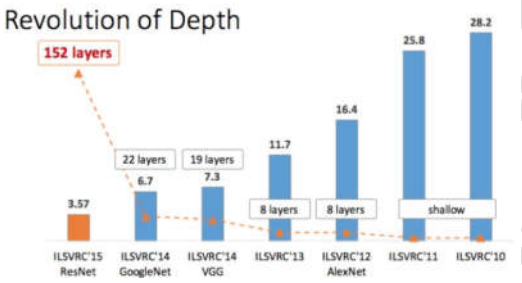
■ 几种经典卷积神经网络结构对比

模型	AlexNet	VGG	GoogLeNet	ResNet
出现时间	2012	2014	2014	2015
总的层数	8	19	22	152
Top-5	16.4%	7.3%	6.7%	3.57%
卷积层数	5	16	21	151
卷积核大小	11，5，3	3	7，1，3，5	7，1，3，5
全连接层数	3	3	1	1
全连接大小	4096，1000	4096，1000	1000	1000

经典卷积神经网络对比

■ 经典卷积神经网络的图像分类效果对比

- 2012年，卷积神经网络首次在ILSVRC（ImageNet Large Scale Visual Recognition Challenge）大赛中出现，并取得第一名
- 与以往的浅层网络相比，AlexNet的错误率提高了近10个百分点
- 几种卷积神经网络模型相比，ResNet分类效果最好



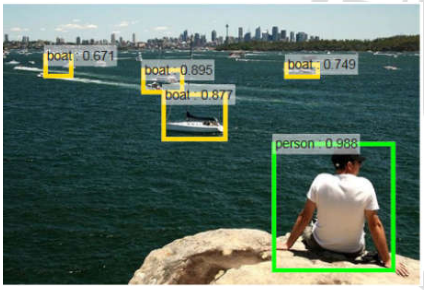
https://www.jianshu.com/p/628d7099e019

卷积神经网络应用-目标检测

- 目标检测简介
- Two-stage方法
- One-stage方法

什么是目标检测?

- “在哪里”+“是什么”
  - 通过回归方法得到检测框的位置信息，对目标进行定位
  - 通过分类方法预测目标类别，给出相应的类别概率



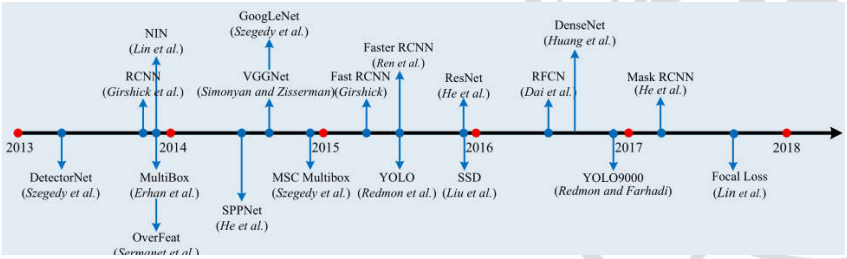
Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.

目标检测算法框架

- 在目标检测的深度学习算法近年来的发展中，主要分为两个派别：
  - 以一定的准确率为代价，追求更快速度的One-stage方法
    - YOLO
    - SSD
    - ……
  - 以一定的速度为代价，追求更高准确率的Two-Stage方法
    - Fast R-CNN
    - Faster R-CNN
    - ……
- 在本章中我们主要通过SSD和Faster R-CNN两个算法来举例讲述目标检测算法。

目标检测

- 基于深度学习的目标检测算法发展历程



图中向上箭头表示的是Two-Stage算法，向下箭头表示的是One-Stage算法

Liu L, Ouyang W, Wang X, et al. Deep learning for generic object detection: A survey[J]. arXiv preprint arXiv:1809.02165, 2018.

卷积神经网络应用-目标检测

- 目标检测简介
- One-stage方法
- Two-stage方法

One-stage方法

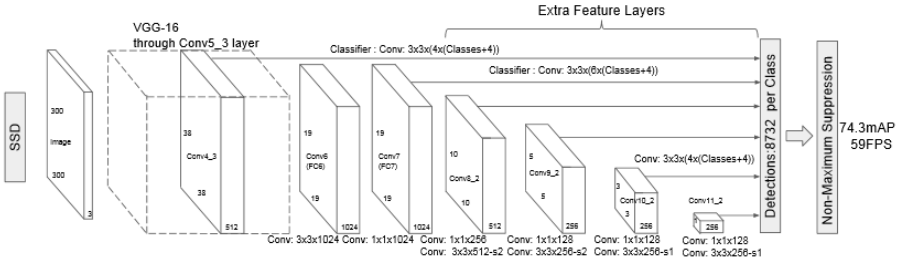
- One-stage方法
  - 首先根据预测网络预测得到每个anchor的分类置信度和偏移量
  - 根据的分类置信度，筛选出一定数量较好框住物体的anchor；
  - 再对筛选出的anchor用偏移量进行修正得到更为精确的predicted box。
- 特点
  - 直接从bounding box只经过一次偏移量的回归便得到predicted box，并且直接对所框物体进行精细的多类别分类，故被称为One-stage；
  - 网络结构简单，训练速度快；
  - 一般来说，One-stage 方法精度略低于Two-stage方法。

SSD

- SSD（Single Shot MultiBox Detector）的算法流程
  - 对于输入的图像，首先通过卷积操作得到不同尺度的多个feature map；
  - 对于不同尺度的每个feature map，以其每个像素作为框的中心，设置K个不同大小和长宽比的先验框；
  - 对于每个框中的内容，通过卷积操作最终得到对于c+1个类别的分类置信度和偏移量；
  - 根据的分类置信度，筛选出一定数量较好框住物体的anchor；
  - 根据偏移量调整anchor，得到的predicted box；
  - SSD最大的特点是其在多个不同尺度的feature map上都进行了predicted box的预测，将所有尺度上得到的predicted box进行NMS（非极大值抑制）方法得到最终的结果。

SSD

- 网络结构图

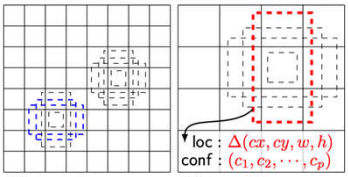


- Conv: 3×3×512-s1表示卷积核的大小3×3，有512个卷积核，卷积运算时滑动步长s=1
- Classifier: Conv:3×3×(4×(Classes+4))表示Classifier的输入是一个经过4×(Classes+4))个大小为3×3卷积核卷积后的特征图
- 最终可以得到38×38×4+19×19×6+10×10×6+5×5×6+3×3×4+1×1×4=8732个先验框，也即最终得到8732×(*Classes*+4)大小的预测结果

Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.

SSD

- 在多尺度feature map上设置先验框
  - 先验框是一系列人为事先设定好位置和大小的框，常被称为anchor box；
  - SSD中先验框的配置为面积一致，长宽比为$\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$的五个框，再加上长宽比为1但面积大小稍小的一个框，共6个框；
  - 多尺度的作用是较大的特征图检测小目标，较小的特征图检测大目标。

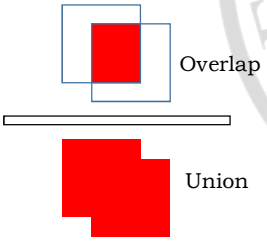


(b) 8 × 8 feature map (c) 4 × 4 feature map

Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.

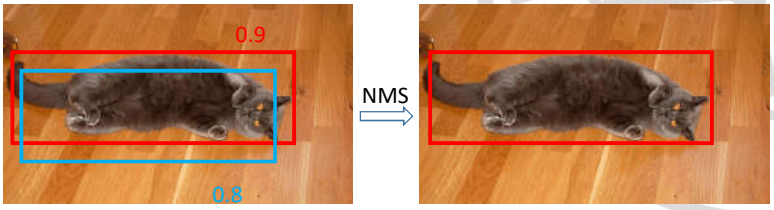
SSD

- 交并比（IoU）
  - $IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$
  - 衡量两个区域重叠程度
  - 当IoU大于某个阈值（IOU的阈值一般为0.7）时，则两个区域认为重叠程度较高



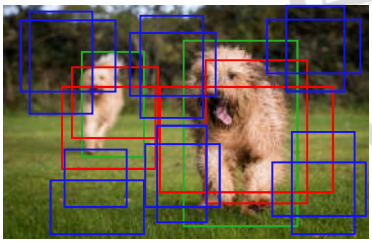
SSD

- Non-Maximum Suppression（NMS）
  - NMS主要为了消除冗余边界框，找到最佳的目标边界框
  - 假设有ABCDEF六个边界框
    - 先将所有边界框的scores排序，得到A<B<C<D<E<F
    - 保留边界框F，依次计算A~E与F的IoU，并判断是否大于设定的阈值
    - 如果B、D满足条件，则删除边界框B和D
    - 再从A、C、E中选出得分最高的框保留下来，然后继续上述过程



SSD

- 对于One-stage算法而言，其面临最大的问题为训练时的类别不平衡问题：
  - 由于SSD穷举了大量的先验框，而这些先验框中绝大多数都是背景框，导致训练时对于框分类的学习中，负样本要远远大于正样本；
- 为此我们常用Hard negative mining策略来改善：
  - 计算损失时，不使用所有的负样本，而是对负样本进行抽样
    - 按照置信度降序，选择置信度误差高的背景作为负样本
    - 按照1:3的正负比例，选取负样本数



SSD

- 损失函数
  - 损失函数由分类和回归两部分组成，并采用超参数α控制两部分的权重

$$L(x,c,l,g)=\frac{1}{N}(L_{conf}(x,c)+\alpha L_{loc}(x,l,g))$$

  - 分类损失计算

$$C_{conf}(x,c)=-\sum_{i\in Pos}^{N}x_{ij}^{p}\log(\hat{c}_i^p)-\sum_{i\in Neg}\log(\hat{c}_i^0),where\hat{c}_i^p=\frac{exp(c_i^p)}{\sum_p exp(c_i^p)}$$

  - 回归损失计算

$$C_{loc}(x,l,g)=\sum_{i\in Pos}^{N}\sum_{m\in\{cx,cy,w,h\}}x_{ij}^{k}\,smooth_{L1}(l_i^m-\hat{g}_i^m)$$

$$smooth_{L1}(l_i^m-\hat{g}_i^m)=\begin{cases}0.5|l_i^m-\hat{g}_i^m|^2, & if|l_i^m-\hat{g}_i^m|<1\\ |l_i^m-\hat{g}_i^m|-0.5, & otherwise\end{cases}$$

卷积神经网络应用-目标检测

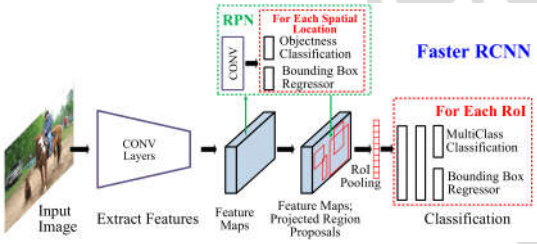
- 目标检测简介
- One-stage方法
- Two-stage方法

Two-stage方法

- 主要是基于region proposal的方法
  - 首先通过分类和回归网络得到anchor的前背景分类置信度和坐标偏移量；
  - 根据对前景置信度高的anchor，使用回归得到的偏移量进行调整，得到region proposal；
  - 然后再对得到的region proposal进一步修正，对其进行精细的多分类和偏移量回归得到最终的结果predicted box；
  - 相当于对初始的anchor box进行两次调整，因此检测结果精度较高。
- 特点
  - 先产生region proposal，再对其进行第二次精修，所以被称为Two-stage方法；
  - 一般来说，Two-stage方法速度略慢于One-stage方法。

Faster R-CNN

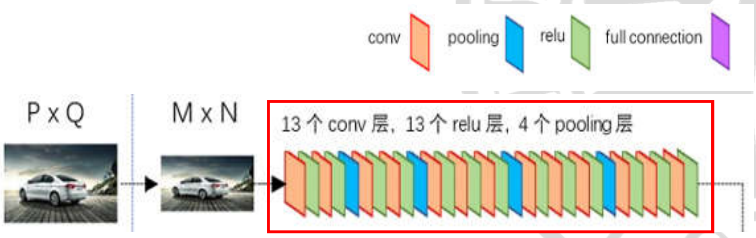
- Faster R-CNN网络整体框架主要分为4个部分
  - CONV Layers：特征提取
  - RPN：生成候选框region proposal
  - ROI pooling：将不同大小的框映射为相同维度的特征向量
  - Classification：对预测框进行偏移量的回归和类别的精细分类



Liu L, Ouyang W, Wang X, et al. Deep learning for generic object detection: A survey[J]. arXiv preprint arXiv:1809.02165, 2018.

CONV Layers

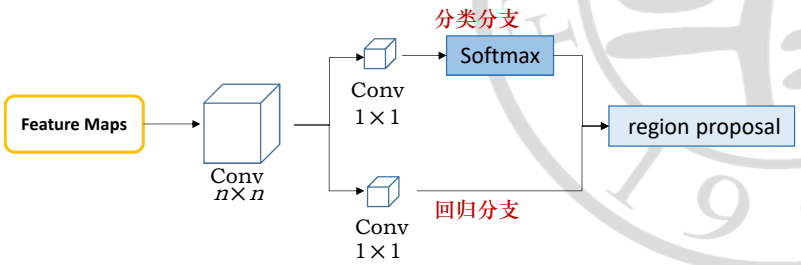
- CONV Layers 网络结构图如下：
  - 先将输入图片resize成M×N大小，为了保证输入网络的图片大小一致
  - CONV Layers的主要作用是进行特征提取
  - 红色框中为卷积神经网络结构，可根据需求使用不同的结构



https://zhuanlan.zhihu.com/p/31426458

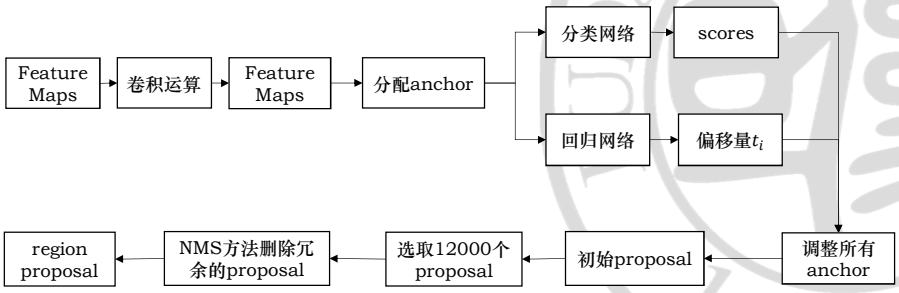
RPN(Region Proposal Networks)

- RPN结构图
  - RPN网路中有分类和回归两条分支
    - 分类网络用于预测anchor属于背景和前景的概率，
    - 回归网络用于预测anchor往Ground-Truth(GT)方向调整需要的偏移量
  - 根据分类和回归结果，调整anchor boxes，生成region proposal



RPN(Region Proposal Networks)

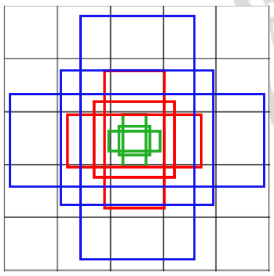
■ RPN网络生成region proposal具体操作过程



RPN(Region Proposal Networks)

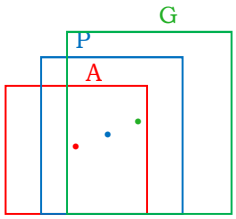
■ 先验框anchor box的设置

- Faster R-CNN中先验框被设置为具有3种不同的大小$\{128^2, 256^2, 512^2\}$，3种不同的长宽比$\{1, 2, \frac{1}{2}\}$，共9种框；
- Faster R-CNN的先验框仅在CONV Layers最终输出的feature map上设置，并不像SSD那样利用上多尺度的信息。



RPN(Region Proposal Networks)

- Bounding box regression
  - 将边界框的预测视为回归任务，回归网络预测出一个偏移量t_i
    - 回归的预测值为$t_i = [t_x, t_y, t_w, t_h]$，真值为$t_i^* = [t_x^*, t_y^*, t_w^*, t_h^*]$
    - 理想情况下，根据偏移量t_i^*调整anchor的位置和大小，使其能与GT 重合
    - 实际情况，根据偏移量t_i调整anchor的位置和大小，只能得到与GT boxes近似的P boxes（即预测框proposal）
  - Bounding box regression过程就是根据回归预测出的偏移量t_i，调整anchor位置和大小，最后得到proposal的过程



RPN(Region Proposal Networks)

- 偏移量计算
  - 令先验anchor的定位信息为$[x_a, y_a, w_a, h_a]$，GT 的定位信息为$[x^*, y^*, w^*, h^*]$，Proposal的定位信息为$[x, y, w, h]$
  - anchor与GT的偏移量t_i^*的计算方式为

$$t_x^* = \frac{(x^* - x_a)}{w_a} \qquad t_y^* = (y^* - y_a)/h_a$$

$$t_w^* = log(w^*/w_a) \qquad t_h^* = log(h^*/h_a)$$

  - anchor与proposal的偏移量t_i的计算方式为

$$t_x = \frac{(x - x_a)}{w_a} \qquad t_y = (y - y_a)/h_a$$

$$t_w = log(w/w_a) \qquad t_h = log(h/h_a)$$

RPN(Region Proposal Networks)

- Loss Function
  - 损失函数由分类损失和回归损失两部分组成，通过超参数λ控制分类和回归的权重比例

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

其中

$$L_{cls}(p_i, p_i^*) = -p_i^* log p_i$$

$$L_{reg}(t_i, t_i^*) = smooth_{L1}(t_i, t_i^*)$$

$$smooth_{L1}(t_i, t_i^*) = \begin{cases} 0.5(t_i - t_i^*)^2, & if|t_i - t_i^*| < 1 \\ |t_i - t_i^*| - 0.5, & otherwise \end{cases}$$

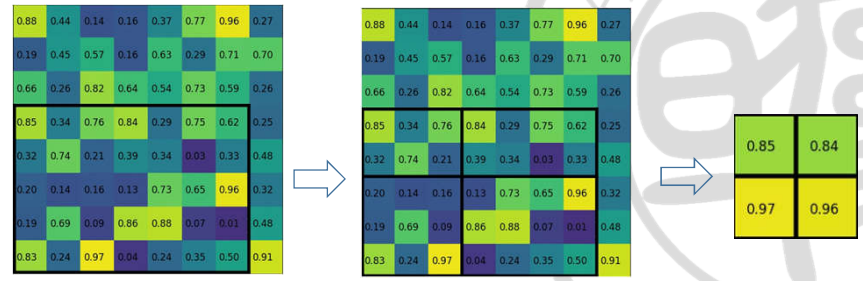
使用$smooth_{L1}$ 损失函数对离群点更加鲁棒

ROI pooling

- ROI pooling：将不同大小的输入转化成相同大小进行输出
- ROI pooling层有两个输入：
  - 从CONV Layers层得到的固定大小的feature map
  - RPN生成的不同大小的region proposal
- ROI pooling具体操作
  - 将proposal映射到feature maps对应位置
  - 将映射后的区域平均划分为个数相同的块
  - 对每个块进行max pooling操作，于是得到相同大小的输出

ROI pooling

- ROI pooling具体操作过程



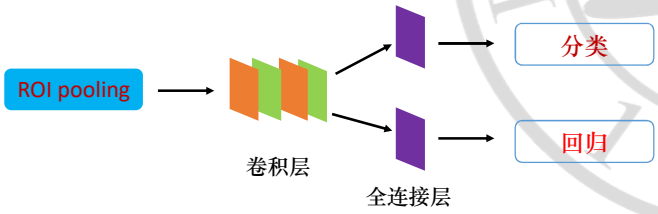
Feature map 上的proposal

将proposal划分为2×2的区域

池化后
feature maps

Classification

- 网络有两条分支，进行分类和回归
  - RPN中的分类是一个二分类问题，判断anchor是背景框还是目标框
  - 而此处是一个多分类问题，预测anchor框中是属于哪类目标或者背景
  - 回归方式与RPN中相同
- 输出的结果可与看做是对RPN结果的进一步微调
- Faster R-CNN网络对边界框和目标类别进行了两次分类与回归，因此被称为Two-stage方法



Thanks

Email: libin@fudan.edu.cn