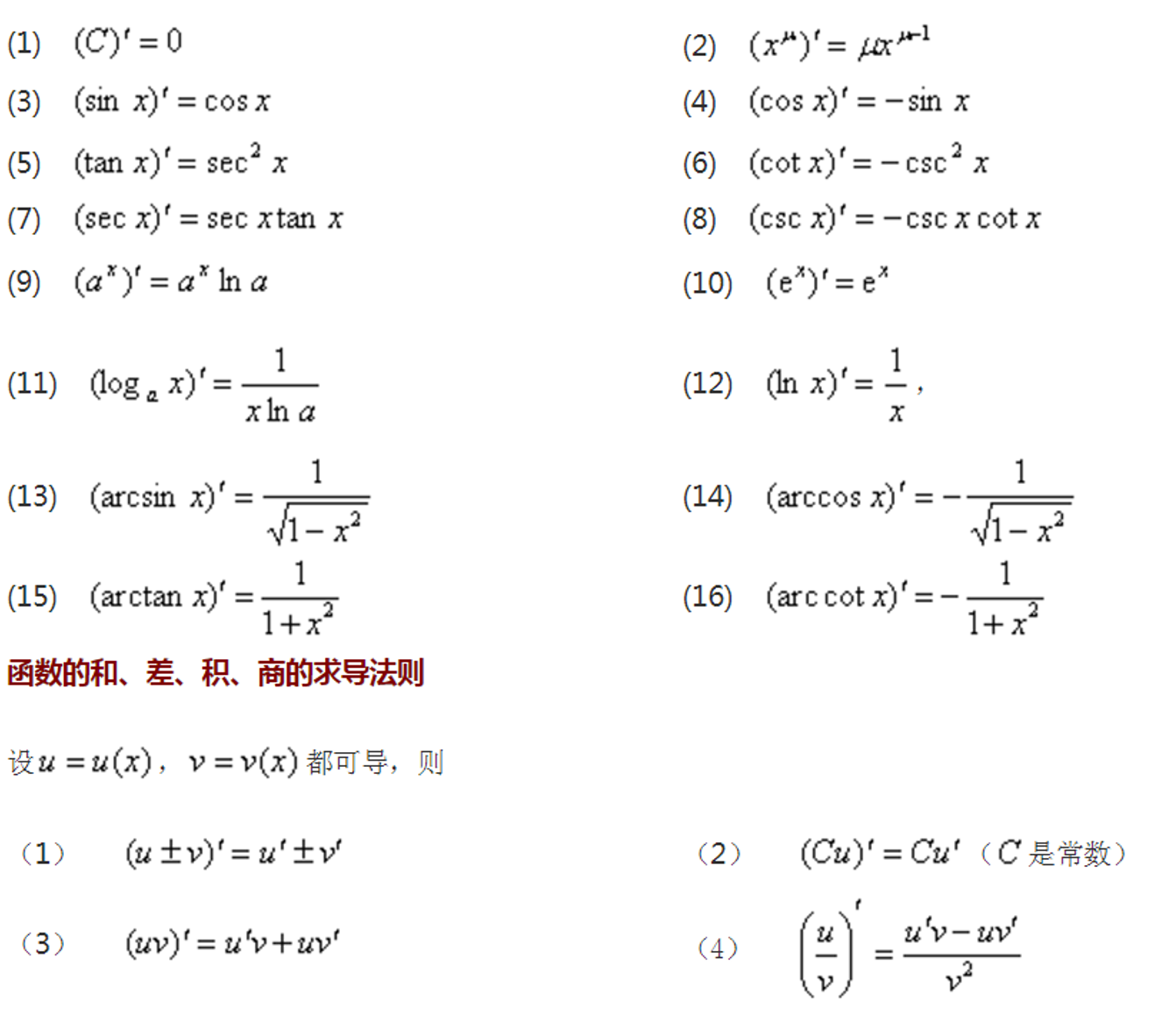
深度学习

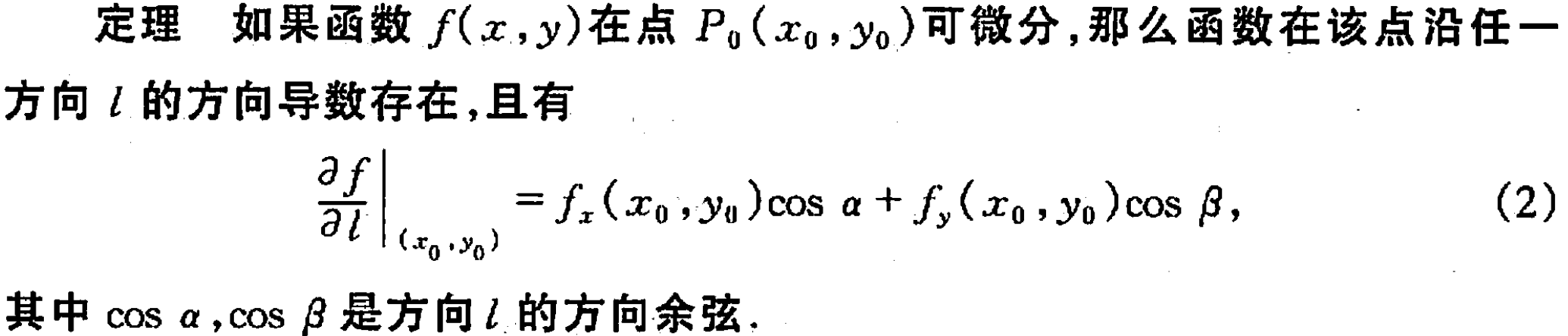
1. 数学相关
   1. 求导公式

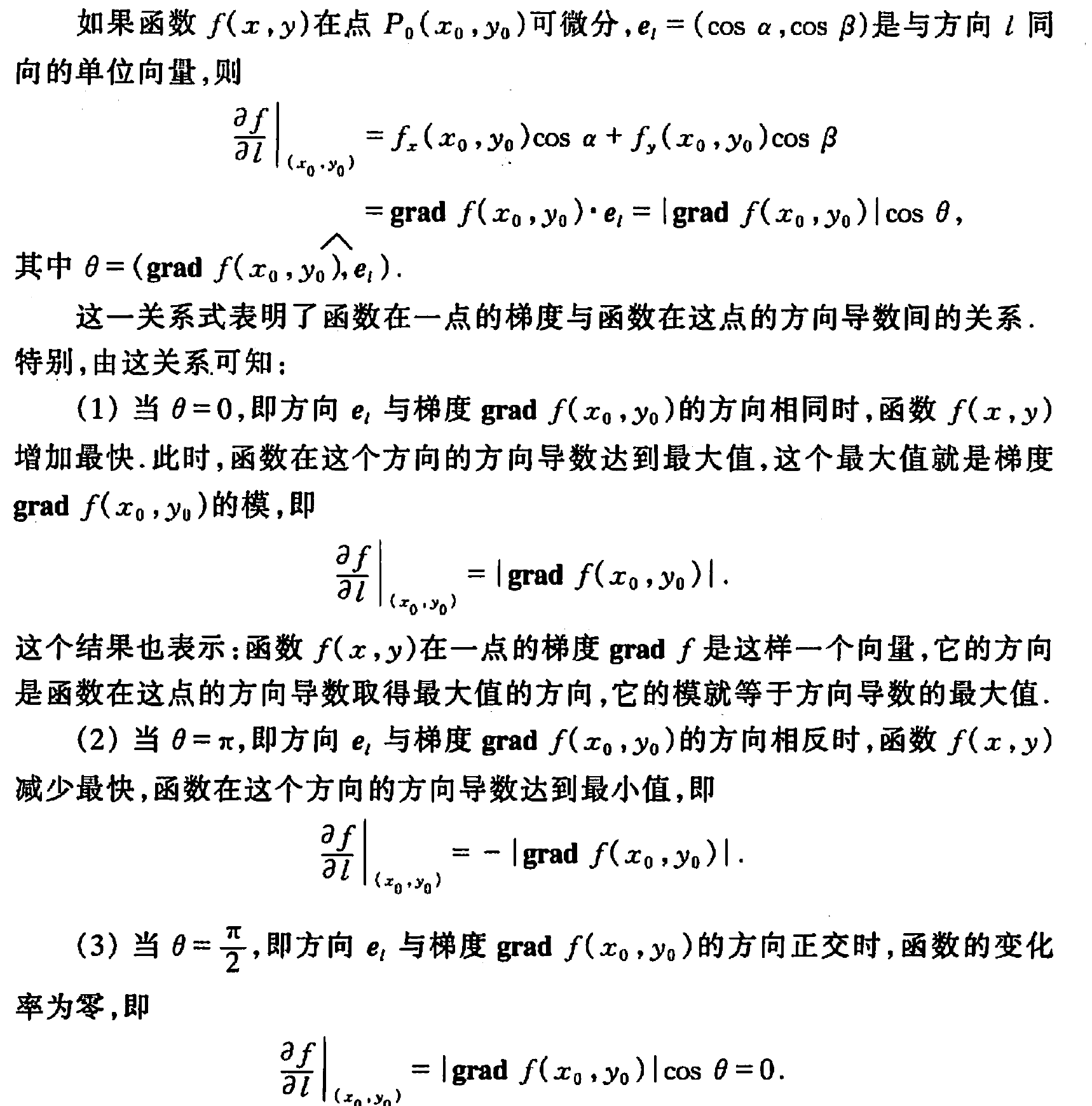


* 1. **Softmax求导**
  2. **雅可比矩阵**

一阶偏导数组成的矩阵

* 1. **什么是导数、偏导数、方向导数和梯度**？
* 导数：函数在某点的导数是函数在该点的瞬时变化率，几何上即该点切线的斜率
* 偏导数：函数在某点沿坐标轴方向的导数
* 方向导数：函数在某点沿某一方向的导数（偏导数是方向导数沿坐标轴方向的特例）
* 梯度：梯度是一个由函数对各坐标轴的偏导数组成的向量，方向导数可以写成梯度grad和单位方向向量l的内积,故当grad与l同方向时，方向导数取得最大值，故梯度的方向是方向导数取最大值时的方向。
  1. **数学上的梯度推导**





1. 正则化
   1. **L1、L2正则化**

看那篇文章就够了

* 1. **Dropout为什么能防止过拟合？**

一般情况下，对于同一组训练数据，利用不同的神经网络训练之后，求其输出的平均值可以减少overfitting。Dropout就是利用这个原理，每次丢掉一半的隐藏层神经元，相当于在不同的神经网络上进行训练，这样就减少了神经元之间的依赖性，即每个神经元不能依赖于某几个其他的神经元（指层与层之间相连接的神经元），使神经网络更加能学习到与其他神经元之间的更加健壮robust的特征。

1. CNN相关
   1. **如果一个CNN网络的输入channel数目和卷积核数目都减半，总的计算量变为原来的多少？**

1/4。参考CNN的时间复杂度。

* 1. **data augmentation有哪些技巧和方法？**

水平翻转、旋转、缩放、裁剪、平移、加高斯噪声、加椒盐噪声

* 1. **inception是如何缓解梯度消失的？**

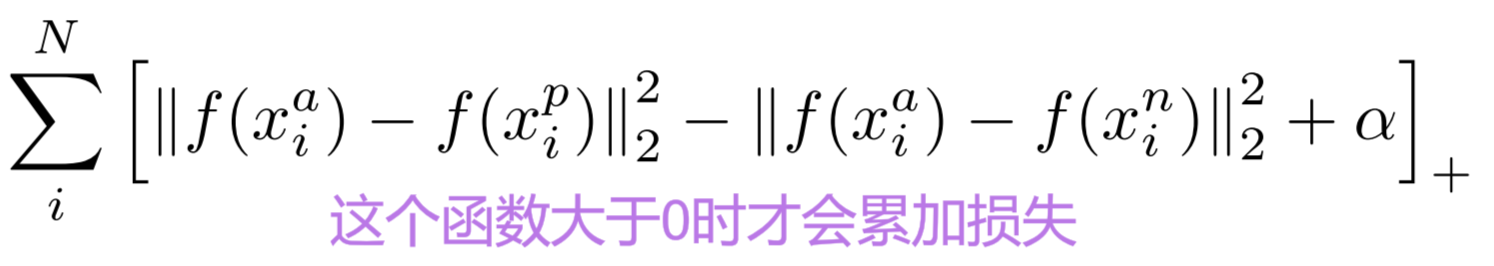
train的时候，额外增加了两个softmax通道

* 1. **卷积核如何进行权值共享的？大的卷积核和小的卷积核相比有什么优缺点，为什么现在基本不用大卷积核？**

所谓的权值共享就是说，给定一张图片，用一个filter去扫这张图片，filter里面的数值就叫做权值，这张图片的每个位置都是被同样的filter扫过的，所以权值是一样的，这就是权值共享。

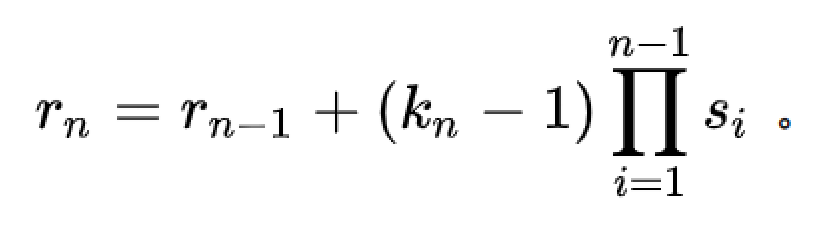
大的卷积核感受野更大，但是当感受野相同时，大的卷积核参数量更大，因此现在普遍使用3\*3卷积，至于为什么不使用更小的卷积核，3\*3卷积核是能捕捉到上下左右概念的最小尺寸。

* 1. **triplet loss，写出数学形式**



* 1. **什么是感受野，感受野的计算公式？**

CNN中的感受野是指网络中某一层的NN在输入图像中对应的感受区域大小



* 1. **手撕代码：卷积实现**

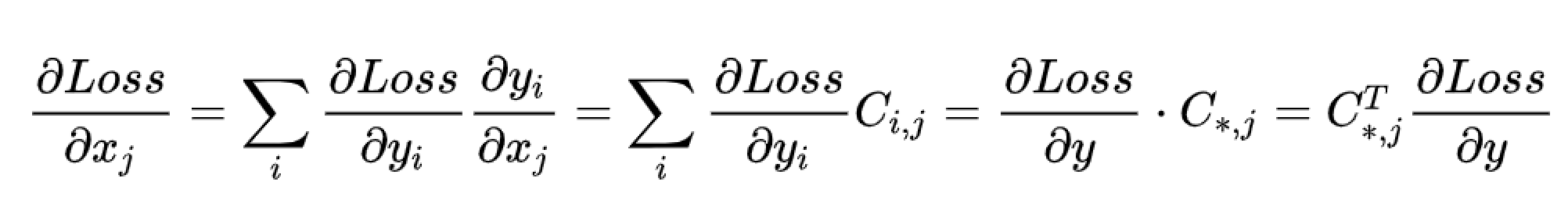
这里有两种实现方法：普通实现和im2col的实现，普通实现在作矩阵乘法时，由于访问的一小块矩阵不是连续内存，速度受到限制，因此在作卷积之前，先把输入图像的每个小块转成一个向量，具体可以参考[这篇文章](https://blog.csdn.net/dwyane12138/article/details/78449898)。

Naïve版本的实现是直接用4个for循环

下面是cs231n中的im2col实现：

1. **def** conv\_forward\_im2col(x, w, b, conv\_param):
2. """
3. A fast implementation of the forward pass for a convolutional layer
4. based on im2col and col2im.
5. Input:
6. - x: Input data of shape (N, C, H, W)
7. - w: Filter weights of shape (F, C, HH, WW)
8. - b: Biases, of shape (F,)
9. - conv\_param: A dictionary with the following keys:
10. - 'stride'
11. - 'pad'
12. """
13. # x: (N, C, H, W)
14. # w: (F, C, HH, WW)
15. # out: (N, F, Hout, Wout)
16. # 使用im2col，w->wcol, x->xcol,
17. #   wcol:(F, C\*HH\*WW)    xcol:(C\*HH\*WW, Hout\*Wout\*N)
18. #   res=wcol.dot(xcol): (F, Hout\*Wout\*N)
19. #   out=res.reshape(F, Hout, Wout, N).transpose(3, 0, 1, 2)
20. N, C, H, W = x.shape
21. num\_filters, \_, filter\_height, filter\_width = w.shape
22. stride, pad = conv\_param['stride'], conv\_param['pad']
24. # Check dimensions
25. **assert** (W + 2 \* pad - filter\_width) % stride == 0, 'width does not work'
26. **assert** (H + 2 \* pad - filter\_height) % stride == 0, 'height does not work'
28. # Create output
29. out\_height = (H + 2 \* pad - filter\_height) // stride + 1
30. out\_width = (W + 2 \* pad - filter\_width) // stride + 1
31. out = np.zeros((N, num\_filters, out\_height, out\_width), dtype=x.dtype)
33. x\_cols = im2col\_cython(x, w.shape[2], w.shape[3], pad, stride)
34. res = w.reshape((w.shape[0], -1)).dot(x\_cols) + b.reshape(-1, 1)
36. out = res.reshape(w.shape[0], out.shape[2], out.shape[3], x.shape[0])
37. out = out.transpose(3, 0, 1, 2)
39. cache = (x, w, b, conv\_param, x\_cols)
40. **return** out, cache
    1. **反卷积原理**

在正向卷积时，把输入x和卷积核C都拉成向量的形式，可以得到输出y=Cx，反向计算时，对x求导可以得到下面的结果：



也就是说正向时左乘C，反向时左乘C的转置，把这个顺序反过来就是反卷积。

1. 检测模型
   1. 为啥用Smooth L1 loss

对输入x，输出f(x)，标签Y：

L2 loss = |f(x) -Y|^2，其导数为 2(f(x) -Y)f'(x)

L1 loss = |f(x) -Y|，其导数为 ±f'(x)

因此L1 loss对噪声（outliers）更鲁棒。

* 1. SSD
* **难例挖掘、按置信度排序**

难例挖掘指的是负例。生成所有default boxes后，执行两种匹配规则，匹配之后少数boxes被分为了正例，但是大多数boxes被分为了负例，正负样本严重不均衡，这时候对负例按置信度损失降序排列（就是说这些负例的分类结果应该是0,0代表背景，如果这个概率很小，那它的损失应该很大），选出top-k个负例，使正负样本个数保持在1:3。

* **L2 Normalization**

<https://blog.csdn.net/weixin_35653315/article/details/72715367>

That was discovered in my other paper (ParseNet) that conv4\_3 has different scale from other layers. That is why I add L2 normalization for conv4\_3 only.

这里的 scale指的应该是数值的大小。

* **数据增广**

第一步是像素级别的变化，包括亮度、对比度、颜色通道交换等变化

第二步是图像维度的变化，包括RandomExpand、RandomCrop等

第三步就是随机镜像

上面三种变化都是以0.5的概率发生的，具体过程参考那篇博客

* **损失函数**
* **NMS**

<https://blog.csdn.net/shuzfan/article/details/52711706>

代码在paper文件夹下

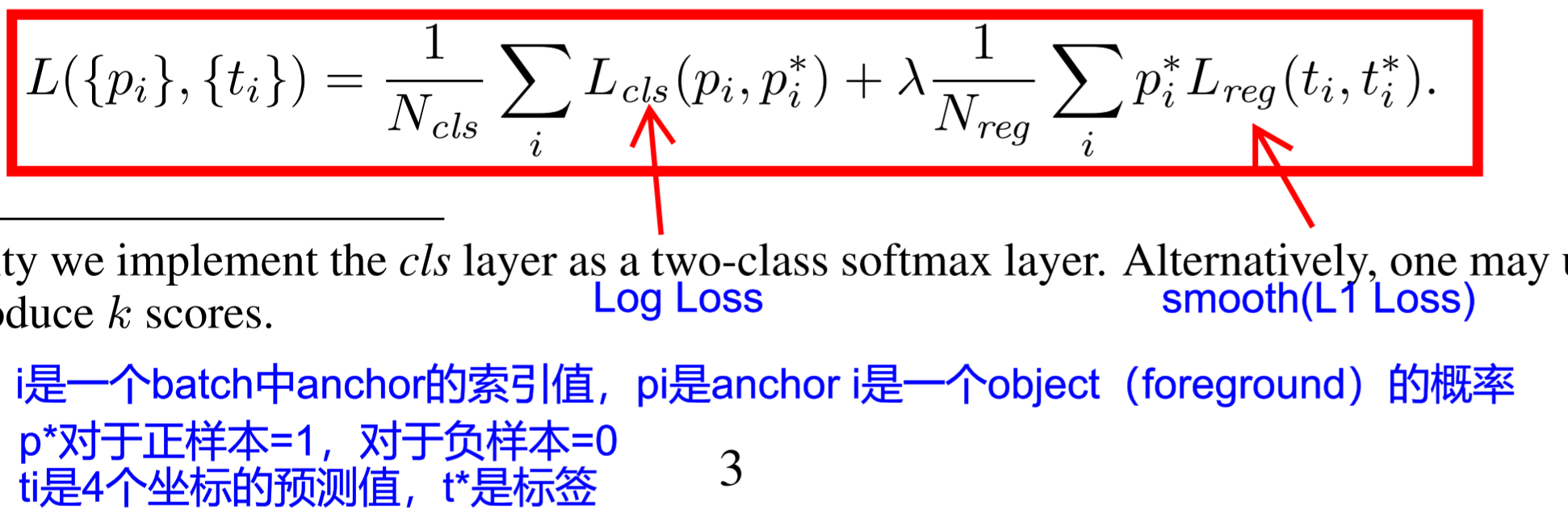
* 1. Faster RCNN
* **RoI Pooling是怎么做的？有什么缺陷？**

RoI Pooling层有两个输入：basenet卷积得到的特征图、RPN输出的多个RoI（是一个向量[N,5,1,1]， N个RoI，第二维是1个RoI索引+左上角坐标+右下角坐标，坐标对应的是原图的坐标）。

由于缩放尺度是16，先根据原图的坐标去计算对应的特征图中的两个坐标，之后把这两个坐标划出的矩形特征图区域进行RoI Pooling，得到统一大小的特征图输出。

对于一个region proposal，首先从原图经过全卷积网络到特征图，得到的候选框位置可能存在浮点数，进行取整操作从而出现第一次量化；其次，在ROI Pooling求取每个小网格的位置时也同样存在浮点数取整的情况。这两次量化的结果都使得候选框的位置会出现偏差。

* **RPN的损失函数(多任务损失:二分类损失+SmoothL1损失)**



* **Faster R-CNN的训练过程**
  1. 用ImageNet分类网络初始化RPN，进行RPN的训练
  2. 用ImageNet分类网络初始化Fast RCNN中的detection部分，并用第一步中RPN生成的RoI训练这个网络
  3. 用detection网络初始化RPN，但是固定共享的卷积层，只训练RPN独有的部分
  4. 固定3中训练的模型中的卷积部分，微调Fast RCNN中的全连接层
* **Faster R-CNN是如何解决正负样本不平衡的问题？**

正样本选取：先从GT出发，匹配IoU最大的anchor，再从所有的anchor出发去匹配任一个GT，只要IoU大于0.7，就把它当成正样本。

负样本选取：如果某个anchor和所有GT的IoU都小于0.3，就把它当成负样本。

* **mAP**

1. 深度学习基本方法
   1. **BN如何处理梯度问题**

看知乎那篇文章

* 1. **初始化方法**

w=np.random.randn(n) \* sqrt(2.0/n)

推导看cs231n

* 1. **pooling的作用，为什么max pooling要更常用？哪些情况下，average pooling比max pooling更合适？max pooling反向传播怎么做？**
  2. **relu的负半轴导数都是0，这部分产生的梯度消失怎么办？**

Leaky relu

* 1. **batch size对收敛速度的影响**

会影响训练的稳定性，Batch size过小会使Loss曲线振荡的比较大，大小一般按照2的次幂规律选择，至于为什么？是为了硬件计算效率考虑的，说GPU训练的时候开的线程是2的次幂个。

* 1. **LR与线性回归的区别**

LR通常用于二分类，使用的是交叉熵损失函数；线性回归用于回归，使用的是均方误差损失函数。

* 1. **分类中为什么交叉熵损失函数比均方误差损失函数更常用？**

<https://www.jianshu.com/p/d20e293a0d34>

* 1. **解释AUC**

参考ROC PR Curve Notes，就是ROC曲线下方的面积

* 1. **混淆矩阵是什么，ROC、PR曲线是什么**

参考ROC PR Curve Notes

1. 待读论文

* Mask RCNN
* FPN
* RetinaNet
* MobileNet
* DenseNet
* SqueezeNet
* Focal Loss