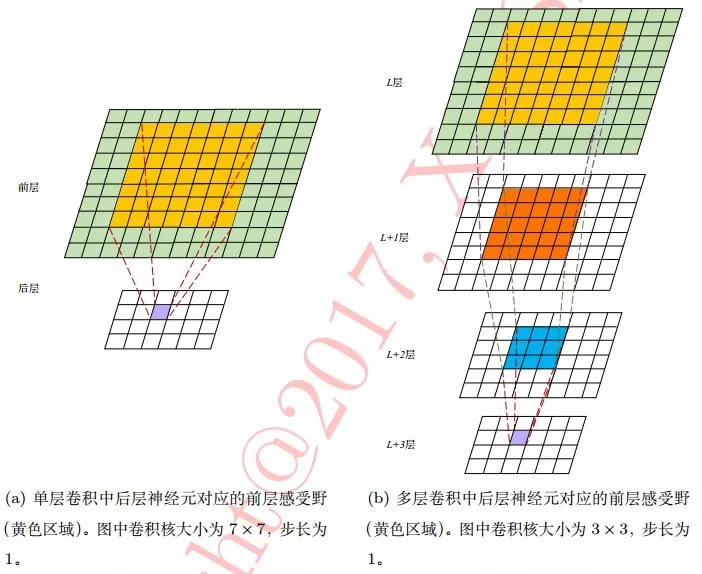
1. **什么是感受视野？**

后一层神经元在前一层神经元的感受空间，如下图所示：



注意：小卷积核（如33）通过多层叠加可取得与大卷积核（如77）同等规模的感受野，此外采用小卷积核有两个优势：

1、小卷积核需多层叠加，加深了网络深度进而增强了网络容量(model capacity)和复杂度（model complexity）

2、增强了网络容量的同时减少了参数个数。

1. **介绍下卷积操作作用**

卷积网络中的卷积核参数是通过网络训练出来的

通过卷积核的组合以及随着网络后续操作的进行，卷积操作可获取图像区域不同类型特征；基本而一般的模式会逐渐被抽象为具有高层语义的“概念”表示，也就是自动学习到图像的高层特征。

1. **CNN权值共享问题**

首先权值共享就是滤波器共享，滤波器的参数是固定的，即是用相同的滤波器去扫一遍图像，提取一次特征特征，得到feature map。在卷积网络中，学好了一个滤波器，就相当于掌握了一种特征，这个滤波器在图像中滑动，进行特征提取，然后所有进行这样操作的区域都会被采集到这种特征，就好比上面的水平线。

1. **CNN结构特点**

局部连接，权值共享，池化操作，多层次结构。

1、局部连接使网络可以提取数据的局部特征

2、权值共享大大降低了网络的训练难度，一个Filter只提取一个特征，在整个图片（或者语音／文本） 中进行卷积

3、池化操作与多层次结构一起，实现了数据的降维，将低层次的局部特征组合成为较高层次的特征，从而对整个图片进行表示。

1. **pooling层作用**

增加特征平移不变性。汇合可以提高网络对微小位移的容忍能力。

减小特征图大小。汇合层对空间局部区域进行下采样，使下一层需要的参数量和计算量减少，并降低过拟合风险。

最大汇合可以带来非线性。这是目前最大汇合更常用的原因之一。

1. **深度特征的层次性**

卷积操作可获取图像区域不同类型特征，而汇合等操作可对这些特征进行融合和抽象，随着若干卷积、汇合等操作的堆叠，各层得到的深度特征逐渐从泛化特征（如边缘、纹理等）过渡到高层语义表示（躯干、头部等模式）。

1. **什么样的数据集不适合深度学习**

数据集太小，数据样本不足时，深度学习相对其它机器学习算法，没有明显优势。

数据集没有局部相关特性，目前深度学习表现比较好的领域主要是图像／语音／自然语言处理等领域，这些领域的一个共性是局部相关性。图像中像素组成物体，语音信号中音位组合成单词，文本数据中单词组合成句子，这些特征元素的组合一旦被打乱，表示的含义同时也被改变。对于没有这样的局部相关性的数据集，不适于使用深度学习算法进行处理。举个例子：预测一个人的健康状况，相关的参数会有年龄、职业、收入、家庭状况等各种元素，将这些元素打乱，并不会影响相关的结果。

1. **什么造成梯度消失问题**

神经网络的训练中，通过改变神经元的权重，使网络的输出值尽可能逼近标签以降低误差值，训练普遍使用BP算法，核心思想是，计算出输出与标签间的损失函数值，然后计算其相对于每个神经元的梯度，进行权值的迭代。

梯度消失会造成权值更新缓慢，模型训练难度增加。造成梯度消失的一个原因是，许多激活函数将输出值挤压在很小的区间内，在激活函数两端较大范围的定义域内梯度为0，造成学习停止。

1. **Overfitting怎么解决**

首先所谓过拟合，指的是一个模型过于复杂之后，它可以很好地“记忆”每一个训练数据中随机噪音的部分而忘记了去“训练”数据中的通用趋势。过拟合具体表现在：模型在训练数据上损失函数较小，预测准确率较高；但是在测试数据上损失函数比较大，预测准确率较低。

Parameter Norm Penalties(参数范数惩罚)；Dataset Augmentation (数据集增强)；Early Stopping(提前终止)；Parameter Tying and Parameter Sharing (参数绑定与参数共享)；Bagging and Other Ensemble Methods(Bagging 和其他集成方法)；dropout；regularization；batch normalizatin。是解决Overfitting的常用手段。

1. **L1和L2区别**

L1 范数（L1 norm）是指向量中各个元素绝对值之和，也有个美称叫“稀疏规则算子”（Lasso regularization）。比如 向量 A=[1，-1，3]， 那么 A 的 L1 范数为 |1|+|-1|+|3|。简单总结一下就是：

1、L1 范数: 为 x 向量各个元素绝对值之和。

2、L2 范数: 为 x 向量各个元素平方和的 1/2 次方，L2 范数又称 Euclidean 范数或 Frobenius 范数

3、Lp 范数: 为 x 向量各个元素绝对值 p 次方和的 1/p 次方. 在支持向量机学习过程中，L1 范数实际是一种对于成本函数求解最优的过程，因此，L1 范数正则化通过向成本函数中添加 L1 范数，使得学习得到的结果满足稀疏化，从而方便人类提取特征。

L1 范数可以使权值参数稀疏，方便特征提取。L2 范数可以防止过拟合，提升模型的泛化能力。

1. **TensorFlow计算图**

Tensorflow 是一个通过计算图的形式来表述计算的编程系统，计算图也叫数据流图，可以把计算图看做是一种有向图，Tensorflow 中的每一个计算都是计算图上的一个节点，而节点之间的边描述了计算之间的依赖关系。

1. **BN（批归一化）的作用**

(1). 可以使用更高的学习率。如果每层的scale不一致，实际上每层需要的学习率是不一样的，同一层不同维度的scale往往也需要不同大小的学习率，通常需要使用最小的那个学习率才能保证损失函数有效下降，Batch Normalization将每层、每维的scale保持一致，那么我们就可以直接使用较高的学习率进行优化。

(2). 移除或使用较低的dropout。 dropout是常用的防止overfitting的方法，而导致overfit的位置往往在数据边界处，如果初始化权重就已经落在数据内部，overfit现象就可以得到一定的缓解。论文中最后的模型分别使用10%、5%和0%的dropout训练模型，与之前的40%-50%相比，可以大大提高训练速度。

(3). 降低L2权重衰减系数。还是一样的问题，边界处的局部最优往往有几维的权重（斜率）较大，使用L2衰减可以缓解这一问题，现在用了Batch Normalization，就可以把这个值降低了，论文中降低为原来的5倍。

(4). 取消Local Response Normalization层。由于使用了一种Normalization，再使用LRN就显得没那么必要了。而且LRN实际上也没那么work。

(5). Batch Normalization调整了数据的分布，不考虑激活函数，它让每一层的输出归一化到了均值为0方差为1的分布，这保证了梯度的有效性，可以解决反向传播过程中的梯度问题。目前大部分资料都这样解释，比如BN的原始论文认为的缓解了Internal Covariate Shift(ICS)问题。

1. **什么是梯度消失和爆炸，怎么解决？**

当训练较多层数的模型时，一般会出现梯度消失问题（gradient vanishing problem）和梯度爆炸问题（gradient exploding problem）。注意在反向传播中，当网络模型层数较多时，梯度消失和梯度爆炸是不可避免的。

深度神经网络中的梯度不稳定性，根本原因在于前面层上的梯度是来自于后面层上梯度的乘积。当存在过多的层次时，就出现了内在本质上的不稳定场景。前面的层比后面的层梯度变化更小，故变化更慢，故引起了梯度消失问题。前面层比后面层梯度变化更快，故引起梯度爆炸问题。

解决梯度消失和梯度爆炸问题，常用的有以下几个方案：

预训练模型 + 微调

梯度剪切 + 正则化

relu、leakrelu、elu等激活函数

BN批归一化

CNN中的残差结构

LSTM结构

1. **RNN循环神经网络理解**

循环神经网络（recurrent neural network, RNN）, 主要应用在语音识别、语言模型、机器翻译以及时序分析等问题上。在经典应用中，卷积神经网络在不同的空间位置共享参数，循环神经网络是在不同的时间位置共享参数，从而能够使用有限的参数处理任意长度的序列。 RNN可以看做作是同一神经网络结构在时间序列上被复制多次的结果，这个被复制多次的结构称为循环体，如何设计循环体的网络结构是RNN解决实际问题的关键。RNN的输入有两个部分，一部分为上一时刻的状态，另一部分为当前时刻的输入样本。

1. **图像处理中平滑和锐化操作是什么？**

平滑处理（smoothing）也称模糊处理（bluring），主要用于消除图像中的噪声部分，平滑处理常用的用途是用来减少图像上的噪点或失真，平滑主要使用图像滤波。在这里，我个人认为可以把图像平滑和图像滤波联系起来，因为图像平滑常用的方法就是图像滤波器。在OpenCV3中常用的图像滤波器有以下几种：

方框滤波——BoxBlur函数

均值滤波（邻域平均滤波）——Blur函数

高斯滤波——GaussianBlur函数（高斯低通滤波是模糊，高斯高通滤波是锐化）

中值滤波——medianBlur函数

双边滤波——bilateralFilter函数 图像锐化操作是为了突出显示图像的边界和其他细节，而图像锐化实现的方法是通过各种算子和滤波器实现的——Canny算子、Sobel算子、Laplacian算子以及Scharr滤波器。

1. **训练过程中模型不收敛，是否说明这个模型无效，致模型不收敛的原因有哪些?**

不一定。导致模型不收敛的原因有很多种可能，常见的有以下几种：

没有对数据做归一化。

没有检查过你的结果。这里的结果包括预处理结果和最终的训练测试结果。

忘了做数据预处理。

忘了使用正则化。

Batch Size设的太大。

学习率设的不对。

最后一层的激活函数用的不对。

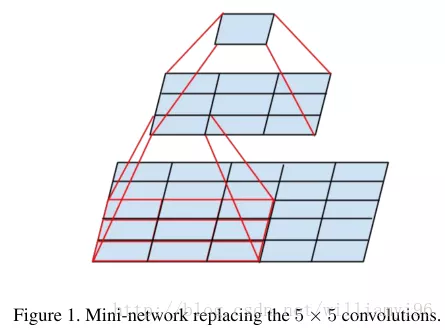
网络存在坏梯度。比如Relu对负值的梯度为0，反向传播时，0梯度就是不传播。

参数初始化错误。

网络太深。隐藏层神经元数量错误。

1. **VGG使用2个3\*3卷积的优势在哪里？**

(1). 减少网络层参数。用两个33卷积比用1个55卷积拥有更少的参数量，只有后者的2∗3∗35∗5=0.72。但是起到的效果是一样的，两个33的卷积层串联相当于一个55的卷积层，感受野的大小都是5×5，即1个像素会跟周围5\*5的像素产生关联。把下图当成动态图看，很容易看到两个3×3卷积层堆叠（没有空间池化）有5×5的有效感受野。



2个３\*3卷积层

(2). 更多的非线性变换。2个33卷积层拥有比1个55卷积层更多的非线性变换（前者可以使用两次ReLU激活函数，而后者只有一次），使得卷积神经网络对特征的学习能力更强。

paper中给出的相关解释：三个这样的层具有7×7的有效感受野。那么我们获得了什么？例如通过使用三个3×3卷积层的堆叠来替换单个7×7层。首先，我们结合了三个非线性修正层，而不是单一的，这使得决策函数更具判别性。其次，我们减少参数的数量：假设三层3×3卷积堆叠的输入和输出有C个通道，堆叠卷积层的参数为3(32C2)=27C2个权重；同时，单个7×7卷积层将需要72C2=49C2个参数，即参数多81％。这可以看作是对7×7卷积滤波器进行正则化，迫使它们通过3×3滤波器（在它们之间注入非线性）进行分解。

1. **Relu比Sigmoid效果好在哪里？**

Sigmoid函数公式如下：



ReLU激活函数公式如下：



relu激活函数

relu函数方程 ReLU 的输出要么是 0, 要么是输入本身。虽然方程简单，但实际上效果更好。在网上看了很多版本的解释，有从程序实例分析也有从数学上分析，我找了个相对比较直白的回答，如下：

1、ReLU函数计算简单，可以减少很多计算量。反向传播求误差梯度时，涉及除法，计算量相对较大，采用ReLU激活函数，可以节省很多计算量；

2、避免梯度消失问题。对于深层网络，sigmoid函数反向传播时，很容易就会出现梯度消失问题（在sigmoid接近饱和区时，变换太缓慢，导数趋于0，这种情况会造成信息丢失），从而无法完成深层网络的训练。

3、可以缓解过拟合问题的发生。Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生。

4、相比sigmoid型函数，ReLU函数有助于随机梯度下降方法收敛。

1. **神经网络中权值共享的理解？**

权值(权重)共享这个词是由LeNet5模型提出来的。以CNN为例，在对一张图偏进行卷积的过程中，使用的是同一个卷积核的参数。比如一个3×3×1的卷积核，这个卷积核内9个的参数被整张图共享，而不会因为图像内位置的不同而改变卷积核内的权系数。说的再直白一些，就是用一个卷积核不改变其内权系数的情况下卷积处理整张图片（当然CNN中每一层不会只有一个卷积核的，这样说只是为了方便解释而已）。

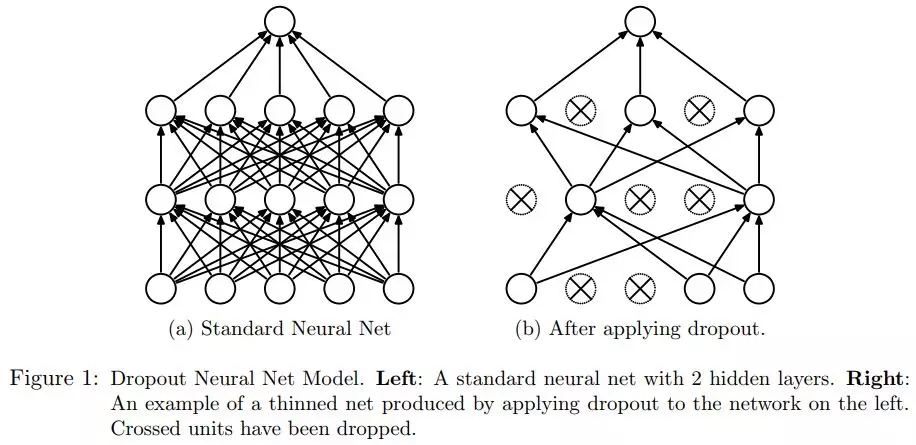
1. **对fine-tuning(微调模型的理解)，为什么要修改最后几层神经网络权值？**

使用预训练模型的好处，在于利用训练好的SOTA模型权重去做特征提取，可以节省我们训练模型和调参的时间。

至于为什么只微调最后几层神经网络权重，是因为：(1). CNN中更靠近底部的层（定义模型时先添加到模型中的层）编码的是更加通用的可复用特征，而更靠近顶部的层（最后添加到模型中的层）编码的是更专业业化的特征。微调这些更专业化的特征更加有用，它更代表了新数据集上的有用特征。(2). 训练的参数越多，过拟合的风险越大。很多SOTA模型拥有超过千万的参数，在一个不大的数据集上训练这么多参数是有过拟合风险的，除非你的数据集像Imagenet那样大。

1. **什么是dropout?**

dropout可以防止过拟合，dropout简单来说就是：我们在前向传播的时候，让某个神经元的激活值以一定的概率p停止工作，这样可以使模型的泛化性更强，因为它不会依赖某些局部的特征。



dropou直观展示

1. **dropout具体工作流程？**

以 标准神经网络为例，正常的流程是：我们首先把输入数据x通过网络前向传播，然后把误差反向传播一决定如何更新参数让网络进行学习。使用dropout之后，过程变成如下：

(1). 首先随机（临时）删掉网络中一半的隐藏神经元，输入输出神经元保持不变（图3中虚线为部分临时被删除的神经元）；

(2). 然后把输入x通过修改后的网络进行前向传播计算，然后把得到的损失结果通过修改的网络反向传播。一小批训练样本执行完这个过程后，在没有被删除的神经元上按照随机梯度下降法更新对应的参数（w，b）；

(3). 然后重复这一过程：

1、恢复被删掉的神经元（此时被删除的神经元保持原样没有更新w参数，而没有被删除的神经元已经有所更新）

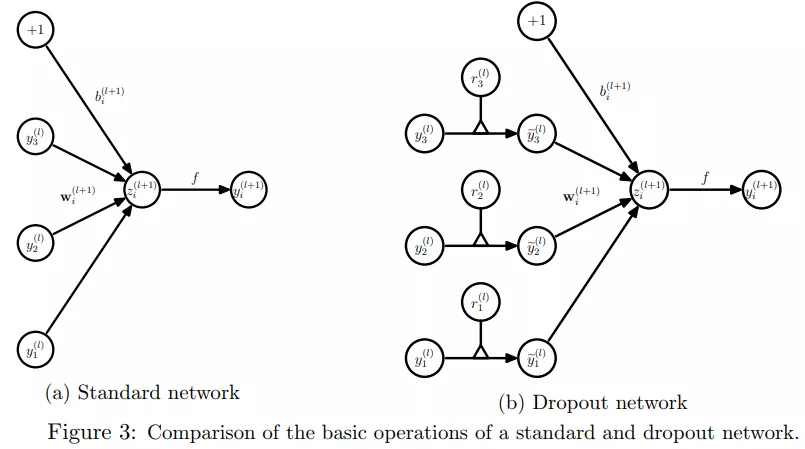
2、 从隐藏层神经元中随机选择一个一半大小的子集临时删除掉（同时备份被删除神经元的参数）。

3、对一小批训练样本，先前向传播然后反向传播损失并根据随机梯度下降法更新参数（w，b） （没有被删除的那一部分参数得到更新，删除的神经元参数保持被删除前的结果）。

1. **dropout在神经网络中的应用？**

(1). 在训练模型阶段

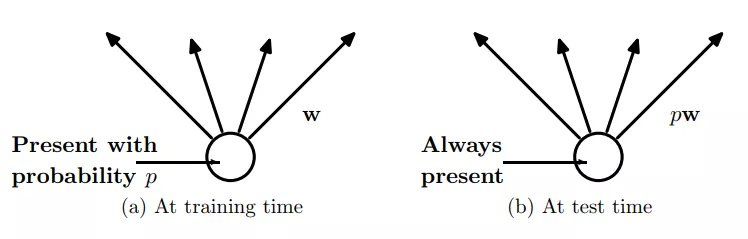
不可避免的，在训练网络中的每个单元都要添加一道概率流程，标准网络和带有dropout网络的比较图如下所示：



dropout在训练阶段

(2). 在测试模型阶段

预测模型的时候，输入是当前输入，每个神经单元的权重参数要乘以概率p。



dropout在测试模型时

1. **如何选择dropout 的概率？**

input 的dropout概率推荐是0.8， hidden layer 推荐是0.5， 但是也可以在一定的区间上取值。（All dropout nets use p = 0.5 for hidden units and p = 0.8 for input units.）

1. **HOG算法原理描述**

方向梯度直方图（Histogram of Oriented Gradient, HOG）特征是一种在计算机视觉和图像处理中用来进行物体检测的特征描述子。它通过计算和统计图像局部区域的梯度方向直方图来构成特征。在深度学习取得成功之前，Hog特征结合SVM分类器被广泛应用于图像识别中，在行人检测中获得了较大的成功。

HOG特征原理

HOG的核心思想是所检测的局部物体外形能够被光强梯度或边缘方向的分布所描述。通过将整幅图像分割成小的连接区域（称为cells），每个cell生成一个方向梯度直方图或者cell中pixel的边缘方向，这些直方图的组合可表示出（所检测目标的目标）描述子。为改善准确率，局部直方图可以通过计算图像中一个较大区域(称为block)的光强作为measure被对比标准化，然后用这个值(measure)归一化这个block中的所有cells。这个归一化过程完成了更好的照射/阴影不变性。与其他描述子相比，HOG得到的描述子保持了几何和光学转化不变性（除非物体方向改变）。因此HOG描述子尤其适合人的检测。

HOG特征提取方法就是将一个image：

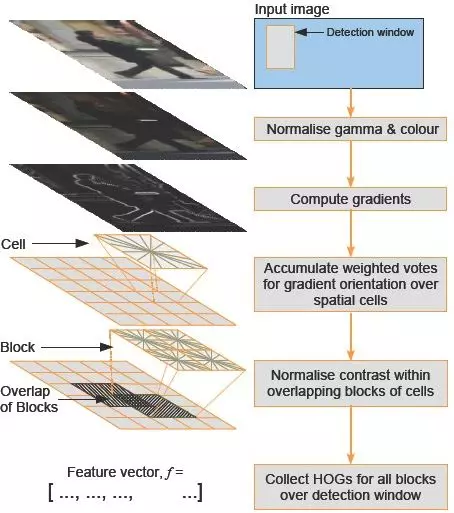
1、灰度化（将图像看做一个x,y,z（灰度）的三维图像）

2、划分成小cells（2\*2）

3、计算每个cell中每个pixel的gradient（即orientation）

4、统计每个cell的梯度直方图（不同梯度的个数），即可形成每个cell的descriptor。

HOG特征检测步骤



颜色空间归一化——–>梯度计算————->梯度方向直方图———->重叠块直方图归一化———–>HOG特征

1. **移动端深度学习框架知道哪些，用过哪些？**

知名的有TensorFlow Lite、小米MACE、腾讯的ncnn等

1. **如何提升网络的泛化能力**

和防止模型过拟合的方法类似，另外还有模型融合方法。

1. **BN算法，为什么要在后面加加伽马和贝塔，不加可以吗？**

最后的“scale and shift”操作则是为了让因训练所需而“刻意”加入的BN能够有可能还原最初的输入。不加也可以。

1. **激活函数的作用**

激活函数实现去线性化。神经元的结构的输出为所有输入的加权和，这导致神经网络是一个线性模型。如果将每一个神经元（也就是神经网络的节点）的输出通过一个非线性函数，那么整个神经网络的模型也就不再是线性的了，这个非线性函数就是激活函数。常见的激活函数有：ReLU函数、sigmoid函数、tanh函数。

1. **卷积层和池化层有什么区别**

1、卷积层有参数，池化层没有参数

2、经过卷积层节点矩阵深度会改变，池化层不会改变节点矩阵的深度，但是它可以缩小节点矩阵的大小。

1. **卷积层参数数量计算方法**

假设输入层矩阵维度是96963，第一层卷积层使用尺寸为55、深度为16的过滤器（卷积核尺寸为55、卷积核数量为16），那么这层卷积层的参数个数为５53\*16+16=1216个

1. **卷积层输出大小计算**

卷积中的特征图大小计算方式有两种，分别是‘VALID’和‘SAME’，卷积和池化都适用，除不尽的结果都向下取整。公式：O = (W-F+2P)/S+1，输入图片（Input）大小为I=WW，卷积核（Filter）大小为FF，步长（stride）为S，填充（Padding）的像素数为P。

1、SAME填充方式：填充像素。conv2d函数常用。

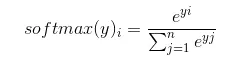
2、VALID填充方式：不填充像素，Maxpooling2D函数常用。"SAME"卷积方式，对于输入55图像，图像的每一个点都作为卷积核的中心。最后得到55的结果，通俗的来说：首先在原图外层补一圈0，将原图的第一点作为卷积核中心，若一圈0不够，继续补一圈0。如下图所示：



1. **神经网络为什么用交叉熵损失函数**

判断一个输出向量和期望的向量有多接近，交叉熵（cross entroy）是常用的评判方法之一。交叉熵刻画了两个概率分布之间的距离，是分类问题中使用比较广泛的一种损失函数。给定两个概率分布p和q，通过q来表示p的交叉熵公式为：H(p,q)=−∑p(x)logq(x)

softmax公式写一下：



1. **1\*1卷积的主要作用有哪些？**

1、降维（ dimension reductionality ）。比如，一张500 \* 500且厚度depth为100 的图片在20个filter上做11的卷积，那么结果的大小为500500\*20。

2、加入非线性。卷积层之后经过激励层，1\*1的卷积在前一层的学习表示上添加了非线性激励（ non-linear activation ），提升网络的表达能力；

1. **目标检测基本概念**

准确率、召回率、F1

混淆矩阵：

True Positive(真正例, TP)：将正类预测为正类数.

True Negative(真负例, TN)：将负类预测为负类数.

False Positive(假正例, FP)：将负类预测为正类数 → 误报 (Type I error).

False Negative(假负例子, FN)：将正类预测为负类数 → 漏报 (Type II error).

查准率（准确率）P = TP/(TP+FP) 查全率（召回率）R = TP/(TP+FN) 准确率描述了模型有多准，即在预测为正例的结果中，有多少是真正例；召回率则描述了模型有多全，即在为真的样本中，有多少被我们的模型预测为正例。以查准率P为纵轴、查全率R为横轴作图，就得到了查准率－查全率曲线，简称\*\*”P-R“曲线，显示改该曲线的图称为”P-R“图。查准率、查全率性能的性能度量，除了”平衡点“（BEP），更为常用的是F1度量\*\*：$$F1 = \frac{2PR}{P+R} = \frac{2\*TP}{样例总数+TP-TN}$$

F1度量的一般形式：$F\_{\beta}$，能让我们表达出对查准率/查全率的偏见，公式如下：$$F\_{\beta} = \frac{1+\beta ^{2}PR}{(\beta ^{2}\*P)+R}$$ $\beta >1$对查全率有更大影响，$\beta < 1$对查准率有更大影响。

不同的计算机视觉问题，对两类错误有不同的偏好，常常在某一类错误不多于一定阈值的情况下，努力减少另一类错误。在目标检测中，mAP（mean Average Precision）作为一个统一的指标将这两种错误兼顾考虑。

map指标解释

具体来说就是，在目标检测中，对于每张图片检测模型会输出多个预测框（远超真实框的个数），我们使用IoU(Intersection Over Union，交并比)来标记预测框是否预测准确。标记完成后，随着预测框的增多，查全率R总会上升，在不同查全率R水平下对准确率P做平均，即得到AP，最后再对所有类别按其所占比例做平均，即得到mAP指标。

交并比IOU

交并比（Intersection-over-Union，IoU），目标检测中使用的一个概念，是产生的候选框（candidate bound）与原标记框（ground truth bound）的交叠率，即它们的交集与并集的比值。最理想情况是完全重叠，即比值为1。计算公式如下：

代码实现如下：

# candidateBound = [x1, y1, x2, y2]

def calculateIoU(candidateBound, groundTruthBound):

cx1 = candidateBound[0]

cy1 = candidateBound[1]

cx2 = candidateBound[2]

cy2 = candidateBound[3]

gx1 = groundTruthBound[0]

gy1 = groundTruthBound[1]

gx2 = groundTruthBound[2]

gy2 = groundTruthBound[3]

carea = (cx2 - cx1) \* (cy2 - cy1) #C的面积

garea = (gx2 - gx1) \* (gy2 - gy1) #G的面积

x1 = max(cx1, gx1)

y1 = min(cy1, gy1) # 原点为(0, 0)，所以这里是min不是max

x2 = min(cx2, gx2)

y2 = max(cy2, gy2)

w = max(0, (x2 - x1))

h = max(0, (y2 - y1))

area = w \* h #C∩G的面积

iou = area / (carea + garea - area)

return iou

1. **数据增强方法，离线数据增强和在线数据增强有什么区别？**

常用数据增强方法：

翻转：Fliplr,Flipud。不同于旋转180度，这是类似镜面的翻折，跟人在镜子中的映射类似，常用水平、上下镜面翻转。

旋转：rotate。顺时针/逆时针旋转，最好旋转90-180度，否则会出现边缘缺失或者超出问题，如旋转45度。

缩放：zoom。图像可以被放大或缩小，imgaug库可用Scal函数实现。

裁剪：crop。一般叫随机裁剪，操作步骤是：随机从图像中选择一部分，然后降这部分图像裁剪出来，然后调整为原图像的大小。

平移：translation。平移是将图像沿着x或者y方向（或者两个方向）移动。我们在平移的时候需对背景进行假设，比如说假设为黑色等等，因为平移的时候有一部分图像是空的，由于图片中的物体可能出现在任意的位置，所以说平移增强方法十分有用。

放射变换：Affine。包含：平移(Translation)、旋转(Rotation)、放缩(zoom)、错切(shear)。

添加噪声：过拟合通常发生在神经网络学习高频特征的时候，为消除高频特征的过拟合，可以随机加入噪声数据来消除这些高频特征。imgaug库使用GaussianBlur函数。

亮度、对比度增强：这是图像色彩进行增强的操作

锐化：Sharpen。imgaug库使用Sharpen函数。

数据增强分两类，一类是离线增强，一类是在线增强：

离线增强 ： 直接对数据集进行处理，数据的数目会变成增强因子 x 原数据集的数目 ，这种方法常常用于数据集很小的时候

在线增强 ： 这种增强的方法用于，获得 batch 数据之后，然后对这个batch的数据进行增强，如旋转、平移、翻折等相应的变化，由于有些数据集不能接受线性级别的增长，这种方法长用于大的数据集，很多机器学习框架已经支持了这种数据增强方式，并且可以使用GPU优化计算。

1. **ROI Pooling替换为ROI Align，及各自原理**

faster rcnn将roi pooling替换为roi align效果有所提升

ROI Pooling原理

ROI Align原理

Reference

1.深度学习中的数据增强

Reference

1、《Batch Normalization Accelerating Deep Network Training by Reducing Internal Covariate Shift》阅读笔记与实现

2、深度学习中 Batch Normalization为什么效果好

3、详解机器学习中的梯度消失、爆炸原因及其解决方法

1. **Python装饰器解释**

装饰器本质上是一个 Python 函数或类，它可以让其他函数或类在不需要做任何代码修改的前提下增加额外功能，装饰器的返回值也是一个函数/类对象。它经常用于有切面需求的场景，比如：插入日志、性能测试、事务处理、缓存、权限校验等场景，装饰器是解决这类问题的绝佳设计。有了装饰器，我们就可以抽离出大量与函数功能本身无关的雷同代码到装饰器中并继续重用。概括的讲，装饰器的作用就是为已经存在的对象添加额外的功能。

1. **多进程与多线程区别**

线程是进程的一部分，一个进程至少有一个线程；

对于操作系统来说，一个任务就是一个进程，进程内的“子任务”称为线程；

多线程和多进程最大的不同在于，多进程中，同一个变量，各自有一份拷贝存在于每个进程中，互不影响，而多线程中，所有变量都由所有线程共享，所以，任何一个变量都可以被任何一个线程修改，因此，线程之间共享数据最大的危险在于多个线程同时改一个变量，把内容给改乱了。

进程间调用、通讯和切换开销均比多线程大，单个线程的崩溃会导致整个应用的退出。

存在大量IO，网络耗时或者需要和用户交互等操作时，使用多线程有利于提高系统的并发性和用户界面快速响应从而提高友好性。

1. **map与reduce函数用法解释下**

1、 map函数接收两个参数，一个是函数，一个是Iterable，map将传入的函数依次作用到序列的每个元素，并将结果作为新的Iterator返回，简单示例代码如下：

# 示例１

def square(x):

return x \*\* 2

r = map(square, [1, 2, 3, 4, 5, 6, 7])

squareed\_list = list(r)

print(squareed\_list) # [1, 4, 9, 16, 25, 36, 49]

# 使用lambda匿名函数简化为一行代码

list(map(lambda x: x \* x, [1, 2, 3, 4, 5, 6, 7, 8, 9]))

# 示例２

list(map(str, [1, 2, 3, 4, 5, 6, 7, 8, 9]))　＃　['1', '2', '3', '4', '5', '6', '7', '8', '9']

注意map函数返回的是一个Iterator（惰性序列），要通过list函数转化为常用列表结构。map()作为高阶函数，事实上它是把运算规则抽象了。

2、reduce()函数也接受两个参数，一个是函数（两个参数），一个是序列，与map不同的是reduce把结果继续和序列的下一个元素做累积计算,效果如下：reduce(f, [x1, x2, x3, x4]) = f(f(f(x1, x2), x3), x4) 示例代码如下：

from functools import reduce

CHAR\_TO\_INT = {

'0': 0,

'1': 1,

'2': 2,

'3': 3,

'4': 4,

'5': 5,

'6': 6,

'7': 7,

'8': 8,

'9': 9

}

def str2int(str):

ints = map(lambda x:CHAR\_TO\_INT[x], str) # str对象是Iterable对象

return reduce(lambda x,y:10\*x + y, ints)

print(str2int('0'))

print(str2int('12300'))

print(str2int('0012345')) # 0012345

1. **Python深拷贝、浅拷贝区别**

1、直接赋值：其实就是对象的引用（别名）。

2、浅拷贝(copy)：拷贝父对象，不会拷贝对象的内部的子对象。copy浅拷贝，没有拷贝子对象，所以原始数据改变，子对象会改变。

3、深拷贝(deepcopy)：copy 模块的 deepcopy 方法，完全拷贝了父对象及其子对象，两者是完全独立的。深拷贝，包含对象里面的自对象的拷贝，所以原始对象的改变不会造成深拷贝里任何子元素的改变。看一个示例程序，就能明白浅拷贝与深拷贝的区别了：

#!/usr/bin/Python3

# -\*-coding:utf-8 -\*-

import copy

a = [1, 2, 3, ['a', 'b', 'c']]

b = a # 赋值，传对象的引用

c = copy.copy(a) # 浅拷贝

d = copy.deepcopy(a) # 深拷贝

a.append(4)

a[3].append('d')

print('a = ', a)

print('b = ', b)

print('c = ', c)

print('d = ', d) # [1, 2, 3, ['a', 'b', 'c']]

程序输出如下：

a = [1, 2, 3, ['a', 'b', 'c', 'd'], 4] b = [1, 2, 3, ['a', 'b', 'c', 'd'], 4] c = [1, 2, 3, ['a', 'b', 'c', 'd']] d = [1, 2, 3, ['a', 'b', 'c']]

1. **图像锐化方法**

锐化主要影响图像中的低频分量，不影响图像中的高频分量像锐化的主要目的有两个：

1、增强图像边缘，使模糊的图像变得更加清晰，颜色变得鲜明突出，图像的质量有所改善，产生更适合人眼观察和识别的图像；

2、过锐化处理后，目标物体的边缘鲜明，以便于提取目标的边缘、对图像进行分割、目标区域识别、区域形状提取等，进一步的图像理解与分析奠定基础。

图像锐化一般有两种方法：

1、微分法

2、高通滤波法

一般来说，图像的能量主要集中在其低频部分，噪声所在的频段主要在高频段，同时图像边缘信息也主要集中在其高频部分。这将导致原始图像在平滑处理之后，图像边缘和图像轮廓模糊的情况出现。为了减少这类不利效果的影响，就需要利用图像锐化技术，使图像的边缘变得清晰。图像锐化处理的目的是为了使图像的边缘、轮廓线以及图像的细节变得清晰，经过平滑的图像变得模糊的根本原因是因为图像受到了平均或积分运算，因此可以对其进行逆运算(如微分运算)就可以使图像变得清晰。微分运算是求信号的变化率，由傅立叶变换的微分性质可知，微分运算具有较强高频分量作用。从频率域来考虑，图像模糊的实质是因为其高频分量被衰减，因此可以用高通滤波器来使图像清晰。但要注意能够进行锐化处理的图像必须有较高的性噪比，否则锐化后图像性噪比反而更低，从而使得噪声增加的比信号还要多，因此一般是先去除或减轻噪声后再进行锐化处理。