深度学习知识：

1. 反向传播（Backpropagation）的概念：输入数据经过神经网络的输入层、隐含层和输出层并得到输出结果，前向传播过程。计算估计值与真实值之间的误差，通过链式求导法则反向回传到隐含层，调整隐含层的权值，直到迭代收敛的过程。
2. 衡量分类器好坏指标：TP、FP、TN、FN 准确率P=TP/(TP+FP) 召回率R=TP/(TP+FN) 2/F1=1/P+1/R ROC曲线 AUC值 AP值（以IOU为分割阈值）
3. 正负样本不平衡的办法：均衡不同类别对于Loss的贡献。

过采样（容易导致过拟合）和下采样（容易忽略重要信息）

数据合成与数据增强（随即裁剪，改变对比度，随机翻转）

在损失函数上进行改进 对于不同的样本赋予不同的权重Focal loss

OHEM（online hard examples mine）不适用于 end-to-end 需要迭代交替进行

1. 样本不均衡如何影响网络的分类能力：分类器根据先验知识只预测多数类。
2. 激活函数的分类

Sigmoid：容易造成梯度消失、不是以0为中心，更新速度慢、含有指数运算。

Tanh：仅仅解决了第二点

Relu:max(x,0)，存在问题1和问题2

Leaky Relu：max(x,0.01x),不存在包饱和机制，更新速度快

pRelu：利用超参数代替0.01

FRelu：利用神经网络训练这个参数 结合了上下文的语义信息，或许能达到更好的情况

Mish：Mish = x\*tanh(ln(1+e^x))

1. ResNet与DenseNet对比 解决了梯度弥散和精度下降的问题。

ResNet 基本组件分为 恒等mapping和残差mapping

如果网络已经到达最优，继续加深网络，residual mapping将被push为0，只剩下identity mapping，这样理论上网络一直处于最优状态了，网络的性能也就不会随着深度增加而降低了。 利用+对 恒等mapping和残差mapping 进行连接。

DenseNet 每一层的输入都是前面所有层的输出直接关联 利用concat对特征图链接。

1. BatchNormalization思想及作用

机器学习的基本假设 IID独立同分布假设

BatchNorm就是在深度神经网络训练过程中使得每一层神经网络的输入保持相同分布的

BN的基本思想其实相当直观：因为深层神经网络在做非线性变换前的激活输入值（就是那个x=WU+B，U是输入）随着网络深度加深或者在训练过程中，其分布逐渐发生偏移或者变动，之所以训练收敛慢，一般是整体分布逐渐往非线性函数的取值区间的上下限两端靠近（对于Sigmoid函数来说，意味着激活输入值WU+B是大的负值或正值），所以这导致反向传播时低层神经网络的梯度消失，这是训练深层神经网络收敛越来越慢的本质原因，而BN就是通过一定的规范化手段，把每层神经网络任意神经元这个输入值的分布强行拉回到均值为0方差为1的标准正态分布，其实就是把越来越偏的分布强制拉回比较标准的分布，这样使得激活输入值落在非线性函数对输入比较敏感的区域，这样输入的小变化就会导致损失函数较大的变化，意思是这样让梯度变大，避免梯度消失问题产生（也可以理解为退化），而且梯度变大意味着学习收敛速度快，能大大加快训练速度。

1. 卷积层和池化层有什么区别

卷积层用于局部感知：人的大脑识别图片的过程中，并不是一下子整张图同时识别，而是对于图片中的每一个特征首先局部感知，然后更高层次对局部进行综合操作，从而得到全局信息

池化（Pooling）也称为欠采样或下采样。主要用于特征降维，压缩数据和参数的数量，减小过拟合，同时提高模型的容错性。主要有最大池化和平均池化

1. 梯度消失（网络层数深，损失函数不合适）、梯度爆炸（权值初始值太大）原因及解决方案

越靠近输出层 权值更新越快

多个激活函数偏导数的连乘，和多个权重参数的连乘

如果激活函数求导后与权重相乘的积大于1，那么随着层数增多，求出的梯度更新信息将以指数形式增加，即发生梯度爆炸；如果此部分小于1，那么随着层数增多，求出的梯度更新信息将会以指数形式衰减，即发生梯度消失。

前面的层较后面的层

解决方案

(1).逐层预训练+微调 先寻找局部最优 后寻找全局最优

(2).梯度剪切、权重正则化 避免权值过大

(3).选择合适的激活函数

(4).BN

(5).残差结构

(6).LSTM长短期记忆网络

1. AP值计算

mpre=np.concatenate(([0],pre,[0]))

mrec=np.concatenate(([0],rec,[1]))

for i in range(mpre.size-1,0,-1):

mpre[i-1]=np.maximum(mpre[i],mpre[i-1])

i=np.where(mrec[1:]!=mrec[:-1])[0]

ap=np.sum((mrec[i+1]-mrec[i])\*mpre[i])

print(ap)

1. 提高网络泛化能力

扩充数据、大batch\_size、调整数据分布、数据增强、权值正则化、dropout、早停。

1. 模型压缩方法

前端压缩：知识蒸馏、滤波器剪枝、紧凑模型设计

后端压缩：量化、低秩近似、未加限制剪枝

卷积核 低秩近似、量化、剪枝、知识蒸馏

1. 知识蒸馏是什么

知识蒸馏分为 目标蒸馏 和 特征蒸馏 掌握该方法的损失函数

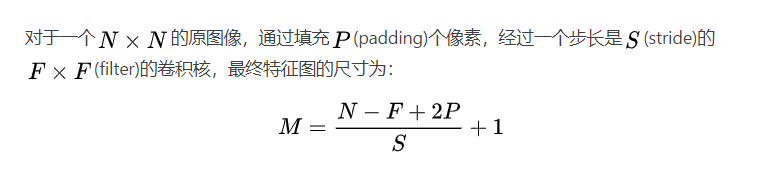
1. 训练teacher网络
2. 修改softmax函数，引入 T 获得soft-target
3. 以soft-target,th和hard-target,1，作为输入训练T
4. 数据扩充、数据预处理、网络参数初始化代码
5. 数据增强 Mixup、Cutout、CutMix、Mosaica
6. 神经网络参数计算类型
7. 卷积层 nn.Conv2d(3,20,(3,3),bias=true)

3\*3\*3\*20+20

1. 线性层 nn.Linear(3,20,bias=True)

3\*20+20

1. Maxpool和bn层 参数为0
2. 卷积层输出图片的尺寸



1. FOCAL LOSS

减小简单样本的所占权重，防止简单样本的loss在梯度下降中占主导地位

（1-p）\*\*γ \*logp p=e \*\* （-logp）

1. 神经网络中的优化器 汇总

SGD 随机梯度下降：

选择任意的一个阶段，计算误差，回传梯度，进行梯度更行

Mountain 动量梯度下降：

对∆J 运用 指数加权平均更新，综合考虑了前面若干回合的梯度值

加速 收敛速度

NAG 反向动量梯度下降：

对∆J 运用 指数加权平均更新，但是最后一项取 来模拟下次的梯度值，可以使梯度下降收敛速度变快 同时停的也快

Adagrad 梯度下降（从这开始 开始考虑优化学习率 α）

学习率 梯度值 随机梯度

Adadelta 优化梯度和变为 梯度的指数加权平方和在开根号

Adam 两个部分全部利用指数加权平均来进行训练