## 1.基本概念

1.1 为什么神经网络中深度网络的表现比广度网络表现好？

浅层结构算法：其局限性在于有限样本和计算单元情况下对复杂函数的表示能力有限，针对复杂分类问题其泛化能力受到一定制约。深度学习可通过学习一种深层非线性网络结构，实现复杂函数逼近，表征输入数据分布式表示，并展现了强大的从少数样本集中学习数据集本质特征的能力。（多层的好处是可以用较少的参数表示复杂的函数）

深度学习的实质，是通过构建具有很多隐层的机器学习模型和海量的训练数据，来学习更有用的特征，从而最终提升分类或预测的准确性。

浅层神经网络可以模拟任何函数，但数据量的代价是无法接受的。深层解决了这个问题。相比浅层神经网络，深层神经网络可以用更少的数据量来学到更好的拟合。深层的前提是：空间中的元素可以由迭代发展而来的。

### 1.2 推导BP算法

误差反向传播算法简称反向传播算法（即BP算法）。使用反向传播算法的多层感知器又称为BP神经网络。BP算法是一个迭代算法，它的基本思想为：（1）先计算每一层的状态和激活值，直到最后一层（即信号是前向传播的）；（2）计算每一层的误差，误差的计算过程是从最后一层向前推进的（这就是反向传播算法名字的由来）；（3）更新参数（目标是误差变小）。迭代前面两个步骤，直到满足停止准则（比如相邻两次迭代的误差的差别很小）。

### 1.4 神经网络是生成模型还是判别模型？

判别模型，直接输出类别标签，或者输出类后验概率p(y|x)

### 1.3 什么是梯度消失和梯度爆炸？

梯度消失：这本质上是由于激活函数的选择导致的， 最简单的sigmoid函数为例，在函数的两端梯度求导结果非常小（饱和区），导致后向传播过程中由于多次用到激活函数的导数值使得整体的乘积梯度结果变得越来越小，也就出现了梯度消失的现象。

梯度爆炸：同理，出现在激活函数处在激活区，而且权重W过大的情况下。但是梯度爆炸不如梯度消失出现的机会多

### 1.4 常用的激活函数有哪些？特点

激活函数分为两类，饱和激活函数和不饱和激活函数。

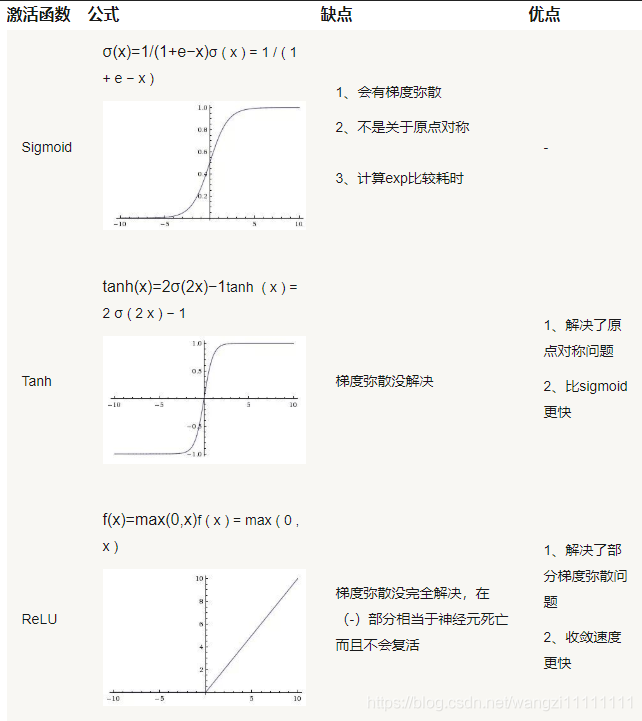
1. 饱和激活函数的代表是sigmoid，tanh。

* 特点是：收敛慢，容易梯度消失。

1. 非饱和激活函数的特点是：

* 收敛快，抑制梯度消失，抑制过拟合。

1. sigmoid：计算量大。梯度消失，会改变原始数据分布。
2. tanh：计算量大，梯度消失比sigmoid好点
3. relu：计算简单，有效防止了梯度消失和梯度爆炸，会出现神经元死亡。
4. leak relu：解决了神经元死亡的问题，但是多了一个参数a
5. ELU：避免dying神经元，并且处处连续，从而加速SGD，但是计算比较复杂



#### Leaky ReLU函数

为了解决 ReLU 激活函数中的梯度消失问题，当 x < 0 时，我们使用 Leaky ReLU——该函数试图修复 dead ReLU 问题。下面我们就来详细了解 Leaky ReLU。

函数表达式如公式5.1，相当于是一个比较简单的Maxout单元：

（5.1）

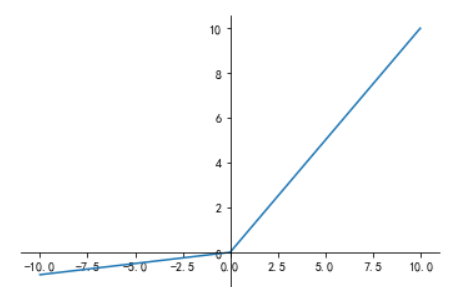


图5.1 Leaky ReLU函数图像（）

导函数图像如图5.2所示：

（5.2）

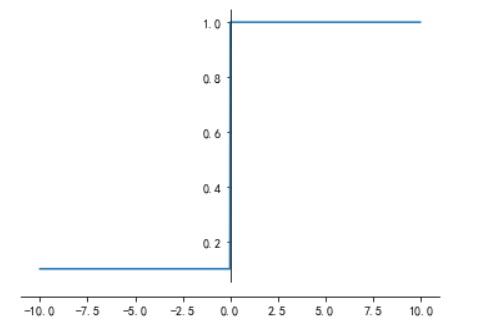


图5.2 Leaky ReLU导函数图像

* Leaky ReLU 通过把 x 的非常小的线性分量给予负输入（0.01x）来调整负值的零梯度（zero gradients）问题，当 x < 0 时，它得到 0.1 的正梯度。该函数一定程度上缓解了 dead ReLU 问题，
* Leak 有助于扩大 ReLU 函数的范围，通常的值为 0.01 左右；
* Leaky ReLU 的函数范围是（负无穷到正无穷），不会有Dead ReLU问题

尽管Leaky ReLU具备 ReLU 激活函数的所有特征（如计算高效、快速收敛、在正区域内不会饱和），但并不能完全证明在实际操作中Leaky ReLU 总是比 ReLU 更好。

### 1.5 常用的参数更新方法有哪些？

参考博客





### 1.6 解决过拟合的方法？

#### 1.6.1数据层面

1. 采集更多的数据
2. 数据增强：多用于图像方面，翻转，裁切，旋转，平移甚至高斯模糊等各种手段，主要分为离线增强和在线增强的方法。

* 离线增强是指数据集在本地进行处理。
* 在线增强：翻转（水平，垂直），旋转，缩放，裁剪，平移，添加噪声等

#### 1.6.2模型层面

* regularization：L1正则化和L2正则化非常类似，所以就放在一起说。这两个正则化的方法，就是在损失函数中添加一项，来防止过拟合。
* 添加上正则化项以后，模型的参数的数值就会变小，通常来讲，模型中的参数的数值越小，模型越趋向于简单，所以，可以通过这种方式防止过拟合。
* dropout只在训练的时候用，让一部分神经元随机失活，这样每一次训练网络都是不同的，最后多个训练的网络叠加，就会减少过拟合。
* Batch normalization是为了让输出都是单位高斯激活，方法是在全连接和激活函数之间加入BatchNorm层，计算每个特征的均值和方差进行规则化。
* Early stoping：



在训练的过程中，可以将训练集的一部分分出来，作为验证集，然后在训练过程中，同时观察error在训集上和验证集上的大小，如上图所示。当发现训练集上error在持续减小，但是验证集上的error不再减小反而增大的时候，就表示该模型可能对训练集发生了过拟合，这个时候，就可以停止训练了。

在验证集error最小的时候停止训练，从而可以得到一个最好的效果。

### 1.7过拟合和欠拟合

过拟合就是训练集上表现好但是验证集上表现不好。

欠拟合就是训练集上loss不降低，学习能力弱。

产生原因：

过拟合产生的原因：模型太复杂，样本噪声多，决策树容易过拟合，iou阈值过高，训练集迭代次数太多。

欠拟合：模型复杂度低；数据不纯净。

解决方案：

过拟合：

1. 正则化，l1，l2
2. dropout
3. 早停，earlystoping
4. 数据增强

欠拟合：

1. 增加模型复杂度
2. 数据预处理

### 1.7 对卡在极小值的处理方法？

1.调节步伐：调节学习速率，使每一次的更新“步伐”不同；

2.优化起点：合理初始化权重（weights initialization）、预训练网络（pre-train），使网络获得一个较好的“起始点”，如最右侧的起始点就比最左侧的起始点要好。常用方法有：高斯分布初始权重（Gaussian distribution）、均匀分布初始权重（Uniform distribution）、Glorot 初始权重、He初始权、稀疏矩阵初始权重（sparse matrix）。

### 1.8 为何使用Batch Normalization？

减少梯度消失，加快了收敛过程。

起到类似dropout一样的正则化能力，一定程度上防止过拟合。

随着网络的深度增加，每层特征值分布会逐渐的向激活函数的输出区间的上下两端（激活函数饱和区间）靠近，这样继续下去就会导致梯度消失。BN就是通过方法将该层特征值分布重新拉回标准正态分布，特征值将落在激活函数对于输入较为敏感的区间，输入的小变化可导致损失函数较大的变化，使得梯度变大，避免梯度消失，同时也可加快收敛。

### 1.9 学习的本质是什么？

将变体拆分成因素和知识（Disentangle Factors of Variation）

## 2.卷积神经网络

### 2.1 1\*1卷积核的作用有哪些？

1. 通过控制卷积核个数实现升维或者降维，从而减少模型参数
2. 对不同特征进行归一化操作（BN）,增加非线性（relu）
3. 用于不同channel上特征的融合

### 2.2 CNN的思想是什么？

改变全连接为局部连接，这是由于图片的特殊性造成的（图像的一部分的统计特性与其他部分是一样的），通过局部连接和参数共享大范围的减少参数值。可以通过使用多个filter来提取图片的不同特征（多卷积核）。

### 2.3 filter尺寸的选择

通常尺寸多为奇数（1，3，5，7）

通常使用多个小的卷积核效果比使用少量的大的卷积核要好。

### 2. 4 输出尺寸计算公式

参考博客

输出尺寸=(N - F +padding\*2)/stride + 1

步长可以自由选择通过补零的方式来实现连接。

Valid卷积意味着不填充，same是填充方式。

总结：padding = “Valid”，N = [(W-F+2P)/S]+1，这里表示的是向下取整再加1

padding = “same”，N = [W/S]，这里表示向上取整

### 2. 5 pooling池化的作用

虽然通过.卷积的方式可以大范围的减少输出尺寸（特征数），但是依然很难计算而且很容易过拟合，所以依然利用图片的静态特性通过池化的方式进一步减少尺寸。

可以认为卷积层是特征的生成，pooling层是特征的选择。

### 2.6反卷积

反卷积又叫做转置卷积，在计算机中计算的时候，转置卷积先将卷积核转为稀疏矩阵C的形式，然后计算的时候正向传播的时候左乘这个稀疏矩阵C的转置，反向传播的时候左乘这个稀疏矩阵C。

一般的卷积运算可以看成是一个其中非零元素为权重的稀疏矩阵C与输入的图像进行矩阵相乘，反向传播时的运算实质为C的转置与loss对输出y的导数矩阵的矩阵相乘

反卷积的运算过程与卷积正好相反，是正向传播时左乘C的转置，反向传播时左乘C

反卷积的用途

* 实现上采样；
* 近似重构输入图像，卷积层可视化

### 2.7如何理解卷积、池化等、全连接层等操作

1. 卷积的作用：相当于滤波器，提取图片不同的特征，生成feature\_map
2. 激活函数的作用：引入非线性因素
3. 池化的作用：1、减少特征维度大小，使特征更加可控，减少参数个数，从而控制过拟合程度，增加网络对略微变换后的图像的鲁棒性；2、达到一种不变性，包括translation，rotation，scale等。3、会造成梯度稀疏，丢失信息，GAN网络中通常使用带步长的卷积，进行下采样来替代pooling。
4. 全连接的作用：对提取到的特征进行分类和回归。

## 3.模型训练、优化和部署

### 3.1你知道哪些边缘端部署的方案？

目前大多数深度学习算法模型要落地对算力要求还是比较高的，如果在服务器上，可以使用GPU进行加速，但是在边缘端或者算力匮乏的开发板子上，不得不对模型进一步的压缩或者改进，也可以针对特定的场景使用市面上现有的推理优化加速框架进行推理。目前来说比较常见的几种部署方案为：

* nvidia GPU：pytorch->onnx->TensorRT
* intel CPU： pytorch->onnx->openvino
* 移动端（手机、开发板等）：pytorch->onnx->MNN、NCNN、TNN、TF-lite、Paddle-lite、RKNN等

### 3.2模型压缩的主要方法有哪些？

1. 从模型结构上来说分为：模型剪枝，模型蒸馏，NAS自动学习模型结构等。

* 模型剪枝的例子很多出现在轻量化网络上面，比如mobilenet v3里面出现的group conv，更改网络末端计算量大的层。深度分离卷积等。
* 模型蒸馏就是迁移学习。

1. 模型参数量化上包括数值精度量化到FP16等。

* 参数量也在mobilenet v3里面有体现，减少网络头部的卷积核的数量。

### 3.3深度机器学习中的mini-batch的大小对学习效果有何影响？

mini-batch太小会导致收敛变慢，太大内存利用率提高了，但是内存容量可能撑不住了，精度降低。泛化性不好。

### 3.4训练过程中,若一个模型不收敛,那么是否说明这个模型无效?导致模型不收敛的原因有哪些?

并不能说明这个模型无效,导致模型不收敛的原因可能有：

1. 数据分类的标注不准确；
2. 样本的信息量太大导致模型不足以fit整个样本空间；
3. 学习率设置的太大容易产生震荡,太小会导致不收敛；
4. 可能复杂的分类任务用了简单的模型；
5. 数据没有进行归一化的操作。

### 3.5 python读取图像可以用什么库，pytorch和tensorflow怎么增加图像的纬度？

Python可以用opencv，pillow和SKimage库来读取图片并处理。

pytorch可以用unsequeeze(0)，tensorflow可以用expand dim来增加维度。

### 3.6在机器学习中为什么经常使用梯度下降而不使用牛顿法？

1. 牛顿法的目标函数是二阶导数，在高维的情况下这个矩阵非常大，计算和储存都是问题。
2. 在小批量的情况下，牛顿法对噪声的估计太大。
3. 在目标函数非凸的情况下，牛顿法易受到鞍点和极大值点的吸引。

### 3.7 finetune

finetune就是通过修改预训练网络模型结构（如修改模型类别输出个数等）选择性的载入预训练网络模型的权重（载入除最后的全连接层之前的所有层）再用自己的数据集重新训练模型。

finetune的实践建议：

1. 预训练模型的限制，不能随意修改模型的网络架构，但是可以输入任意的图片。
2. 学习率：与重新训练相比，finetune需要使用更小的学习率。

### 3.8多尺度问题？

1. 降低网络下采样率与空洞卷积，主要是针对小物体的检测；
2. 根据任务，设计更好的anchor来匹配
3. 多尺度MST的训练，虽然耗时会严重一些
4. 特征融合，类似于FPN、DetNet、RefineDet等都采取了特征融合的方式
5. TridentNet：利用空洞卷积实现了多个尺度分离的检测
6. SNIP（Scale Normalization for Image Pyramids）主要思路：

* 在训练和反向传播更新参数时，只考虑那些在指定的尺度范围内的目标，由此提出了一种特别的多尺度训练方法。

### 3.9优化函数：SGD、Momentum、Adagard、Adam

首先由于病态曲率的存在会导致训练减慢以及局部极小值的问题。所以有了接下来的优化函数。

SGD算法在训练过程中很有可能选择被标记错误的标记数据，或者与正常数据差异很大的数据进行训练，那么使用此数据求得梯度就会有很大的偏差，因此SGD在训练过程中会出现很强的随机现象。SGD一次只进行一次更新，没有冗余，可以新增样本。但是SGD更新会造成严重的震荡。

所以出现了mini-batch梯度下降。

在随机梯度的学习算法中，每一步的步幅都是固定的，而在动量学习算法中，每一步走多远不仅依赖于本次的梯度的大小还取决于过去的速度。速度v是累积各轮训练参的梯度。动量法可以加速SGD，并且抑制震荡，使得梯度方向不变的维度更新更快，梯度方向改变的维度更新变慢，可以加快收敛，减少震荡，但是需要一定的先验知识。

前面的随机梯度和动量随机梯度算法都是使用全局的学习率，所有的参数都是统一步伐的进行更新的。AdaGrad其实很简单，就是将每一维各自的历史梯度的平方叠加起来，然后更新的时候除以该历史梯度值即可。所以Adagrad对低频的参数有较大的更新，对高频的参数有较小的更新，因此，对于稀疏的数据他的表现很好。缺点是Adagrad的学习率会不断收缩，最终变得非常小。

虽然AdaGrad在理论上有些较好的性质，但是在实践中表现的并不是很好，其根本原因就是随着训练周期的增长，学习率降低的很快。而RMSProp算法就在AdaGrad基础上引入了衰减因子

虽然动量加速了我们对最小值方向的搜索，但RMSProp阻碍了我们在振荡方向上的搜索.Adam通过名字我们就可以看出他是基于动量和RMSProp的微调版本，该方法是目前深度学习中最流行的优化方法。

如果数据是稀疏的就用自适应的方法，比如说：Adagrad，Adam，RMSprop等，大多数情况下Adam都可以取得比较好的效果，但是SGD在有比较好的初始化条件下，可以更快的收敛，也可以找到最小点。

### 3.10训练过程中的问题汇总。

#### 3.10.1 loss的问题：

1. 训练过程中loss为负数：

* 【原因】输入的训练数据没有归一化造成
* 【解决方法】把输入数值通过下面的函数过滤一遍，进行归一化

inputdata=(inputdata-inputdata.min())/(inputdata.max()-inputdata.min())

1. 怎么看loss和acc的变化

* train loss 不断下降，test loss不断下降，说明网络仍在学习;
* train loss 不断下降，test loss趋于不变，说明网络过拟合;
* train loss 趋于不变，test loss不断下降，说明数据集100%有问题;
* train loss 趋于不变，test loss趋于不变，说明学习遇到瓶颈，需要减小学习率或批量数目;
* train loss 不断上升，test loss不断上升，说明网络结构设计不当，训练超参数设置不当，数据集经过清洗等问题。

1. 初始学习率设置的问题：

* 可以从0.0001到0.1每次扩大10倍试验，直到验证集上的loss增大。
* 一般0.01-0.1左右
* mmdetection上面有自己的学习率设置方式，一般为0.125batch\_sizegpus。

1. 样本不平衡问题。（具体的我会另外再开一篇博文）

样本不平衡呈现长尾分布，解决方案：

* 上采样增强
* 下采样增强
* 训练动态加权采样
* 稀少数据专家模型

## 4 常用的网络知识点

### 4.1 常用的网络

你最常用的几种目标检测算法是什么？为什么选择这些算法，你选择它们的场景分别是什么？

在工作中，我通常会根据不同的任务选取不同的算法模型：

* 目标检测：yolov5、yolov3、CenterNet、SSD、Faster RCNN、EfficientDet；
* 图像分类：mobileNetv2、mobileNetv3、ghostNet、ResNet系列、ShuffleNetV2、EfficientNet；
* 实例分割：mask-rcnn、yolact、solo；
* 语义分割：deeplabv3、deeplabv3+、UNet；
* 文本检测：CTPN、PSENet、DBNet、YOLOV5；
* 文本识别：CRNN+CTC、CRNN+Attention；

通常，我比较喜欢性能好的模型，性能的指标由两部分，一个是精度，一个是速度。比如在目标检测中，用的比较多的是yolo系列，特别是v4、v5出来后。通常在图像分类的任务上，分类并不困难的情况下会选择一些轻量型的网络，能够一定程度上节省算力资源。其他领域的任务算法抉择也大同小异。

### 4.2 SSD网络

#### 4.2.1为什么SSD对小目标检测效果不好：

1. 小目标对应的anchor（4-6）比较少，其对应的feature map上的pixel难以得到训练，这也是为什么SSD在augmentation之后精确度上涨（因为crop之后小目标就变为大目标）
2. 要检测小目标需要足够大的feature map来提供精确特征，同时也需要足够的语义信息来与背景作区分

### 4.2 yolov3

#### 4.2.1 原理

1. 主干网络：yolov3采用了作者自己设计的darknet53作为主干网络，darknet53借鉴了残差网络的思想，与resnet101、resnet152相比，在精度上差不多的同时，有着更快的速度，网络里使用了大量的残差跳层连接，并且抛弃了pooling池化操作，直接使用步长为2的卷积来实现下采样。
2. 特征融合方面：为了加强小目标的检测，引入了类似与FPN的多尺度特征融合，特征图在经过上采样后与前面层的输出进行concat操作，浅层特征和深层特征的融合，使得yolov3在小目标的精度上有了很大的提升。
3. yolov3的输出分为三个部分：首先是置信度、然后是坐标信息，最后是分类信息。在推理的时候，特征图会等分成S x S的网格，通过设置置信度阈值对格子进行筛选，如果某个格子上存在目标，那么这个格子就负责预测该物体的置信度、坐标和类别信息。

#### 4.2.2 yolov3为什么会那么快

yolov3和SSD比网络更加深了，虽然anchors比SSD少了许多，但是加深的网络深度明显会增加更多的计算量，那么为什么yolov3会比SSD快3倍？

SSD用的很老的VGG16，V3用的其最新原创的Darknet，darknet-53与resnet的网络结构，darknet-53会先用1x1的卷积核对feature降维，随后再利用3x3的卷积核升维，这个过程中，就会大大降低参数的计算量以及模型的大小，有点类似于低秩分解。究其原因是做了很多优化，比如用卷积替代替代全连接，1X1卷积减小计算量等。

#### 4.2.3 yolov3 tiny网络结构

优缺点，前处理，后处理怎么操作

### 4.3 yolov5原理

1. yolov5和v4都是在v3基础上改进的，性能与v4基旗鼓相当，但是从用户的角度来说，易用性和工程性要优于v4，v5的原理可以分为四部分：输入端、backbone、Neck、输出端；
2. 输入端：针对小目标的检测，沿用了v4的mosaic增强，当然这个也是v5作者在他复现的v3上的原创，对不同的图片进行随机缩放、裁剪、排布后进行拼接；二是自适应锚框计算，在v3、v4中，初始化锚框是通过对coco数据集的进行聚类得到，v5中将锚框的计算加入了训练的代码中，每次训练时，自适应的计算不同训练集中的最佳锚框值；
3. backbone：沿用了V4的CSPDarkNet53结构，但是在图片输入前加入了Focus切片操作，CSP结构实际上就是基于Densnet的思想，复制基础层的特征映射图，通过dense block发送到下一个阶段，从而将基础层的特征映射图分离出来。这样可以有效缓解梯度消失问题，支持特征传播，鼓励网络重用特征，从而减少网络参数数量。在V5中，提供了四种不同大小的网络结构：s、m、l、x，通过depth（深度）和width（宽度）两个参数控制。
4. Neck：采用了SPP+PAN多尺度特征融合，PAN是一种自下而上的特征金字塔结构，是在FPN的基础上进行的改进，相对于FPN有着更好的特征融合效果。
5. 输出端：沿用了V3的head，使用GIOU损失进行边框回归，输出还是三个部分：置信度、边框信息、分类信息

### 4.4 yolov4和v5均引入了CSP结构，介绍一下它的原理和作用；

CSP结构是一种思想，它和ResNet、DenseNet类似，可以看作是DenseNet的升级版，它将feature map拆成两个部分，一部分进行卷积操作，另一部分和上一部分卷积操作的结果进行concate。主要解决了三个问题：

1. 增强CNN的学习能力，能够在轻量化的同时保持着准确性；
2. 降低计算成本；
3. 降低内存开销。CSPNet改进了密集块和过渡层的信息流，优化了梯度反向传播的路径，提升了网络的学习能力，同时在处理速度和内存方面提升了不少。

### 4.5 yolov3，v4，v5，每一个版本的网络结构，和版本内的版本之间的差异。

### 4.6 基于Anchor-base的目标检测算法相对于基于Anchor-free的目标检测算法有什么缺陷？

基于Anchor-base的目标检测算法需要在训练前通过聚类决定一系列的anchor，这些anchor具有很强的领域性，泛化差；

anchor机制增加了detection heads的复杂性，增加了预测数量，这在边缘AI系统中，是一个瓶颈。

anchor-free在yolox中的做法是：

* 在每个位置只预测一次，预测四个值：左上角xy坐标的偏移、宽、高。
* 将中心3X3=9的区域设置为正样本，称作：center sampling。

### 4.5 你还了解当下哪些比较流行的目标检测算法？

目前比较流行的目标检测算法有以下几种类型，不局限于这几种：

1. anchor-based：yolov3、yolov4、yolov5、pp-yolo、SSD、Faster-R-CNN、Cascade R-CNN、EfficientDet，RetinaNet、MTCNN；
2. anchor-free：CornerNet、CenterNet、CornerNet-lite、FCOS；
3. transform：DETR；
4. mobile-detector：mobileNet-yolo、mobileNet-SSD、tiny-yolo、nanodet、yolo-fastest、YOLObile、mobilenet-retinaNet、MTCNN；

还有很多很多。mmdetection里面就实现了几十种，可以去看一看，这里面最想总结的是移动端的det，很多都是一些大佬在原生算法基础上的改进，有时间出一篇文章专门记录这个类型的检测器。

### 4.6 了解哪些开源的移动端轻量型目标检测？

轻量型的目标检测其实有很多，大多数都是基于yolo、SSD的改进，当然也有基于其他算法改的；比较常用的改进方法是使用轻量型的backbone替换原始的主干网络，例如：

* mobilenet-ssd、
* mobilenet-yolov3
* yolo-fastest、yolobile
* yolo-nano
* nanodet
* tiny-yolo

等等，在减少了计算量的同时保持着不错的精度，经过移动部署框架推理后，无论是在服务器还是移动端都有着不错的精度和速度。

### 4.7 对于小目标检测，你有什么好的方案或者技巧？

1. 图像金字塔和多尺度滑动窗口检测（MTCNN）
2. 多尺度特征融合检测（FPN、PAN、ASFF等）
3. 增大训练、检测图像分辨率；
4. 超分策略放大后检测；

### 4.8 介绍一下NMS和IOU的原理；

NMS全称是非极大值抑制，顾名思义就是抑制不是极大值的元素。在目标检测任务中，通常在解析模型输出的预测框时，预测目标框会非常的多，其中有很多重复的框定位到了同一个目标，NMS的作用就是用来除去这些重复框，从而获得真正的目标框。而NMS的过程则用到了IOU，IOU是一种用于衡量真实和预测之间相关度的标准，相关度越高，该值就越高。IOU的计算是两个区域重叠的部分除以两个区域的集合部分，简单的来说就是交集除以并集。

在NMS中，首先对预测框的置信度进行排序，依次取置信度最大的预测框与后面的框进行IOU比较，当IOU大于某个阈值时，可以认为两个预测框框到了同一个目标，而置信度较低的那个将会被剔除，依次进行比较，最终得到所有的预测框。

### 4.9 目标检测中如何处理正负样本不平衡的问题？

在目标检测问题中，负样本更容易采集，所以我们能得到的负样本数量一般会比正样本数量多很多。但是负样本多了，就会引起训练数据类别不平衡问题，这会带来：

* 大量容易负样本（不提供有用的学习信息）会导致训练过程无效。
* 大量容易负样本产生的loss会压倒少量正样本的loss（即容易负样本的梯度占主导），导致模型性能衰退。

样本不平衡问题不是目标检测问题独有的困难，在反欺诈，灾害预测，垃圾邮件预测等等问题中也会有正样本过少导致训练集样本不平衡的问题。要解决这个问题，可以采用本文介绍的从数据层面和算法两个层面要思考解决方案。

#### 4.9.1数据层面

通过对训练集合数据的处理，让正样本的数量和负样本的数量比例趋于平衡（例如1:3）。常见的方式有数据重采样和数据增强。

1. 数据重采样：是指在训练之前或者训练时候，对样本多的类别采样频率减少，对样本少的类别采样频率增大，从而使在训练的时候各类类别样本数目比较平衡。
2. 多数样本下采样：拿二分类人脸检测来说，背景是数量样本较多的类别，而人脸是样本较少的类别。对于负样本来说，可以选择\*\*随机抛弃\*\*一部分样本来减少训练时背景样本的数量来达到平衡，但是这样做会降低训练数据的多样性而影响模型泛化能力一般不采用。正确的下采样方式是这样的，比如假设训练时负样本和正样本的比例为3:1（这个比例需要根据实际问题来做出合理的假设），那么\*\*在批训练时候，每批样本随机采集3个负样本（而不是更多）的时候就随机采集1个正样本，使每个批次的训练数据保持负样本比正样本比例大致为3:1；另外，根据训练模型在验证集上测试结果，\*\*观察假阳性的规律\*\*（比如手掌部位的假阳性较多），可以特地选取一些手掌的图片作为负样本进行训练（对负样本进行了约束，自然也就降低了负样本的数量）。
3. 少数样本的上采样：在训练时，对少数样本那一类进行\*\*有放回的抽样\*\*，以用来增加负样本在训练批次里面的数量比例。或者在训练之前，对少数样本那一类进行简单复制，也属于少数样本上采样的一种方式。需要注意的是，仅仅采用少数样本上采样，因为本质上没有真正增加少数样本的多样性，没有带来更多的信息，可能会引起模型\*\*过拟合\*\*问题。更保险和有效的方式是采取多数样本下采样和少数样本上采样\*\*相结合使用\*\*。
4. 数据增强：正样本不够，可以采取一些处理方式，增加正样本，这是一种简单易行的方式。对少数样本的图片添加噪音（例如高斯噪音），进行直方图均衡化操作，进行裁剪，小角度旋转，翻转等等，这些都可以在不改变样本种类的前提下增加少类样本的数量。

#### 4.9.1在算法层面

减轻类别不平衡的方法基本上是\*\*改造优化时的目标函数\*\*，使目标函数倾向于减轻多数样本的惩罚力度，加大少数样本的惩罚力度；或者加大难分样本的惩罚力度，减轻容易分样本的惩罚力度。这其中最具有典型性的是\*\*Focal Loss\*\*。关于Focal Loss的介绍，可以参见[《深度学习常用损失函数》](Captain1986/CaptainBlackboard)一文

4.10 网络模型结构yolov3结构，tiny版本结构，前处理后处理，

mobilenet，v1，v2，v3，

yolov3，v4，v5，每一个版本的网络结构，和版本内的版本之间的差异，

代码必须清楚知道怎么处理，改进

### 4.10为什么mobileNet在理论上速度很快，工程上并没有特别大的提升？

mobileNet虽然在计算量上减少了很多，但是由于深度可分离卷积的操作，使得网络的层数增加了很多，而我们的GPU计算是并行数据处理，假设GPU内存足够大的话所以GPU计算的速度核心是网络的层数。

## 5.传统机器学习

### 5.1简述一下SVM和函数的原理

核函数将数据映射到更高维的空间后处理，但是不用这种显式的映射，而是现将两个向量做内积，然后再用核函数做映射。这样做等价于先做映射，再做内积，而且避免了高维空间复杂的内积运算。

## 5.做项目遇到的问题，怎么解决的？