CV 方向面试题

目录

第一章-图像传统算法

1. 常用的边缘检测算子

Roberts Cross 算子，Prewitt 算子，Sobel 算子，Canny 算子， Krisch 算子等等

2. hough 直线检测和 lsd 检测的区别？lsd 的基本原理

霍夫变换（Hough Transform）是图像处理中从图像中检测几何形状的基本方法之一。

优点：Hough 直线检测的优点是抗干扰能力强，对图像中直线的殘缺部分、噪声以及其它共存的非直线结构不敏感。

缺点：Hough 变换算法的特点导致其时间复杂度和空间复杂度都很高，并且在检测过程中只能确定直线方向，丢失了线段的长度信息。

LSD 是一种局部提取直线的算法，速度比 Hough 要快。

但是有局部算法的缺点：

(1)对于直线相交情况，因为设置了每个点是否 USED，因此每个点只能属于一条直线，若有相交必有至少一条直线被割裂为两条。又因为其基于梯度，直线交点梯度值往往又较小（不被检测为边缘点），因此很有可能相交的两条直线在交点处被割裂为四条线段。

(2)由于局部检测算法自增长的特点，对于长线段被遮挡、局部模糊等原因经常割裂为多条直线。这些缺点在 Hough 变换中不存在。

lsd 的基本原理：

LSD 快速直线检测算法通过对图像局部分析，得出直线的像素点集，再通过假设参数进行验证求解，将像素点集合与误差控制集合合并，进而自适应控制误检的数量 。一般来说，要检测图像中的直线，最基本的思想是检测图像中梯度变化较大的像素点集，LSD 算法也正是利用梯度信息和行列线（level-line）来进行直线检测的。

3. ransac 算法的原理

RANSAC 为 RANdom SAmple Consensus（随机抽样一致）的缩写，它是根据一组包含异常数据的样本数据集，通过迭代方式估计数学模型的参数 计算出数据的数学模型参数，得到有效样本数据的算法。

对于 RANSAC 算法有一个基本的假设：样本中包含正确数据(inliers，符合模型的数据)和异常数据(Outliers，不符合模型的数据)，即数据集中含有噪声。这些异常数据可能是由于错误的测量、错误的假设、错误的计算等产生的。同时 RANSAC 也假设， 给定一组正确的数据，存在可以计算出符合这些数据的模型参数的方法。

4. 介绍一下 SIFT

SIFT 的全称是 Scale Invariant Feature Transform，尺度不变特征变换。SIFT 特征对旋转、尺度缩放、亮度变化等保持不变性，是一种非常稳定的局部特征。

SIFT 算法利用 DoG(差分高斯)来提取关键点(或者说成特征点)，DoG 的思想是用不同的尺度空间因子(高斯正态分布的标准差 σ)对图像进行平滑，然后比较平滑后图像的区别，差别大的像素就是特征明显的点，即可能是特征点。对得到的所有特征点，我们剔除一些不好的，SIFT 算子会把剩下的每个特征点用一个 128 维的特征向量进行描述。

算法流程：

(1)尺度空间的极值检测：搜索所有尺度空间上的图像，通过高斯微分函数来识别潜在的对尺度和选择不变的兴趣点。

(2)特征点定位：在每个候选的位置上，通过一个拟合精细模型来确定位置尺度，关键点的选取依据他们的稳定程度。

(3)特征方向赋值：基于图像局部的梯度方向，分配给每个关键点位置一个或多个方向，后续的所有操作都是对于关键点的方向、尺度和位置进行变换，从而提供这些特征的不变性。

(4)特征点描述：在每个特征点周围的邻域内，在选定的尺度上测量图像的局部梯度，这些梯度被变换成一种表示，这种表示允许比较大的局部形状的变形和光照变换。

5. 介绍一下 HOG

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。具体实现方法是：

首先将图像分成小的连通区域，我们把它叫细胞单元。然后采集细胞单元中各像素点的梯度的或边缘的方向直方图。最后把这些直方图组合起来就可以构成特征描述器。

第二章-数据处理 or 特征工程

1. 常用的数据清洗方法有哪些？

数据清洗是将重复、多余的数据筛选清除，将缺失的数据补充完整，将错误的数据纠正或者删除，最后整理成为我们可以进一步加工、使用的数据。

A. 对于数据值缺失的处理，通常使用的方法有下面几种：

(1) 删除缺失值

当样本数很多的时候，并且出现缺失值的样本在整个的样本的比例相对较小，这种情况下，我们可以使用最简单有效的方法处理缺失值的情况。那就是将出现有缺失值的样本直接丢弃。这是一种很常用的策略。

(2) 均值填补法

根据缺失值的属性相关系数最大的那个属性把数据分成几个组，然后分别计算每个组的均值，把这些均值放入到缺失的数值里面就可以了。

(3) 热卡填补法

对于一个包含缺失值的变量，热卡填充法的做法是：在数据库中找到一个与它最相似的对象，然后用这个相似对象的值来进行填充。不同的问题可能会选用不同的标准来对相似进行判定。最常见的是使用相关系数矩阵来确定哪个变量（如变量 Y）与缺失值所在变量（如变量 X）最相关。然后把所有变量按 Y 的取值大小进行排序。那么变量 X 的缺失值就可以用排在缺失值前的那个个案的数据来代替了。

B. 异常值通常被称为“离群点”，对于异常值的处理，通常使用的方法有下

面几种：

(1) 简单的统计分析

拿到数据后可以对数据进行一个简单的描述性统计分析，譬如最大最小值可以用来判断这个变量的取值是否超过了合理的范围，如客户的年龄为-20 岁或200岁，显然是不合常理的，为异常值。

(2) 3∂原则

如果数据服从正态分布，在 3∂原则下，异常值为一组测定值中与平均值的偏差超过 3 倍标准差的值。如果数据服从正态分布，距离平均值 3∂之外的值出现的概率为 P(|x-u| > 3∂) <= 0.003，属于极个别的小概率事件。如果数据不服从正态分布，也可以用远离平均值的多少倍标准差来描述。

(3) 箱型图分析

箱型图提供了识别异常值的一个标准：如果一个值小于 QL-1.5IQR 或大于 QU-1.5IQR 的值，则被称为异常值。QL 为下四分位数，表示全部观察值中有四分之一的数据取值比它小；QU 为上四分位数，表示全部观察值中有四分之一的数据取值比它大；IQR 为四分位数间距，是上四分位数 QU 与下四分位数 QL 的差值，包含了全部观察值的一半。箱型图判断异常值的方法以四分位数和四分位距为基础，四分位数具有鲁棒性：25%的数据可以变得任意远并且不会干扰四分位数，所以异常值不能对这个标准施加影响。因此箱型图识别异常值比较客观，在识别异常值时有一定的优越性。

(4) 基于模型检测

首先建立一个数据模型，异常是那些同模型不能完美拟合的对象；如果模型是簇的集合，则异常是不显著属于任何簇的对象；在使用回归模型时，异常是相对远离预测值的对象

优缺点：1.有坚实的统计学理论基础，当存在充分的数据和所用的检验类型的知识时，这些检验可能非常有效；2.对于多元数据，可用的选择少一些，并且对于高维数据，这些检测可能性很差。

(5)基于距离

通常可以在对象之间定义邻近性度量，异常对象是那些远离其他对象的对象

优缺点：a.简单；b.缺点：基于邻近度的方法需要 O(m2)时间，大数据集不适用；c.该方法对参数的选择也是敏感的；d.不能处理具有不同密度区域的数据

集，因为它使用全局阈值，不能考虑这种密度的变化。

(6) 基于密度

当一个点的局部密度显著低于它的大部分近邻时才将其分类为离群点。适合非均匀分布的数据。

优缺点：1.给出了对象是离群点的定量度量，并且即使数据具有不同的区域也能够很好的处理；2.与基于距离的方法一样，这些方法必然具有 O(m2)的时间复杂度。对于低维数据使用特定的数据结构可以达到 O(mlogm)；3.参数选择困难。虽然算法通过观察不同的 k 值，取得最大离群点得分来处理该问题，但是，仍然需要选择这些值的上下界。

(7) 基于聚类：

基于聚类的离群点：一个对象是基于聚类的离群点，如果该对象不强属于任何簇。离群点对初始聚类的影响：如果通过聚类检测离群点，则由于离群点影响聚类，存在一个问题：结构是否有效。为了处理该问题，可以使用如下方法：对象聚类，删除离群点，对象再次聚类（这个不能保证产生最优结果）。

2. 做目标检测的时候，有哪些数据增强方式？

图像的目标增强普遍的包含旋转，平移，镜像等等，目标检测的图像增强跟普通的图像增强不同的地方是还得考虑 bounding box 如何进行相应的变化。

(1)裁剪(会改变 boundingbox): 普通的图像裁剪方式，得到新的坐标原点后，将 boundingbox 相应平移

(2)平移(会改变 boundingbox): 跟裁剪类似，也是改变了坐标原点

(3)旋转(会改变 boundingbox): 几何变化

(4)镜像(会改变 boundingbox): boundingbox 的镜像实现方式就是对称位置像素点交换值

(5)改变亮度

(6)加噪声

3. 为什么要对特征做归一化，常用的归一化方法有哪些？

归一化后有什么好处呢？原因在于神经网络学习过程本质就是为了学习数

据分布，一旦训练数据与测试数据的分布不同，那么网络的泛化能力也大大降低；另外一方面，一旦每批训练数据的分布各不相同(batch 梯度下降)，那么网络就要在每次迭代都去学习适应不同的分布，这样将会大大降低网络的训练速度，这也正是为什么我们需要对数据都要做一个归一化预处理的原因。

对于深度网络的训练是一个复杂的过程，只要网络的前面几层发生微小的改变，那么后面几层就会被累积放大下去。一旦网络某一层的输入数据的分布发生改变，那么这一层网络就需要去适应学习这个新的数据分布，所以如果训练过程中，训练数据的分布一直在发生变化，那么将会影响网络的训练速度。

在基于梯度下降的算法中，使用特征归一化方法将特征统一量纲，能够提高模型收敛速度和最终的模型精度。

(1)min-max 标准化（Min-Max Normalization）

也称为离差标准化，是对原始数据的线性变换，使结果值映射到[0 - 1]之间。

(2)Z-score 标准化方法

Z-score 标准化得到的结果是所有数据都聚集在 0 附近，方差为 1。

4. 如何解决数据不平衡问题

欠采样、过采样和生成合成数据。

这三种方法通常在训练分类器之前使用以平衡数据集。简单来说：欠采样：从样本较多的类中再抽取，仅保留这些样本点的一部分；过采样：复制少数类中的一些点，以增加其基数；生成合成数据：从少数类创建新的合成点，以增加其基数。当使用重采样方法（例如从 C0 获得的数据多于从 C1 获得的数据）时，我们在训练过程向分类器显示了两个类的错误比例。以这种方式学得的分类器在未来实际测试数据上得到的准确率甚至比在未改变数据集上训练的分类器准确率还低。实际上，类的真实比例对于分类新的点非常重要，而这一信息在重新采样数据集时被丢失了。因此也应当谨慎使用它们。

除了重采样外，我们还可以在数据集中添加一个或多个其他特征，使数据集更加丰富，这样我们可能获得更好的准确率结果。

第三章-深度学习基础理论知识

1. 深度学习参数初始化的方法有哪些，举例说明

(1) 全零初始化

(2) 随机初始化

(3) Xavier 初始化

(4) He initialization（MSRA）

2. 腾讯优图--常见损失函数

损失函数：

(1)Zero-one Loss（0-1 损失）0-1 loss 是最原始的 loss，它是一种较为简单的损失函数，如果预测值与目标值不相等，那么为 1，否则为 0

(2)Hinge loss 主要用于支持向量机（SVM）中

(3)softmax-loss（多类别）

(4)Logistic-loss（二分类的交叉熵损失函数）

(5)交叉熵，cross entropy（多分类）

(6)triplet loss

(7)均方误差（mean squared error，MSE），也叫平方损失或 L2 损失，常用在最小二乘法中

(8)平均绝对误差（Mean Absolute Error，MAE）是所有单个观测值与算术平均值的绝对值的平均，也被称为 L1 loss，常用于回归问题中

3. 介绍一下你了解的典型神经网络

前馈神经网络：

前馈神经网络采用一种单向多层结构。其中每一层包含若干个神经元，同一层的神经元之间没有互相连接，层间信息的传送只沿一个方向进行。其中第一层称为输入层。最后一层为输出层．中间为隐含层，简称隐层。隐层可以是一层。也可以是多层。

卷积神经网络：

卷积神经网络由一个或多个卷积层和顶端的全连接层（对应经典的神经网络）组成，同时也包括关联权重和池化层（pooling layer）。这一结构使得卷积神经网络能够利用输入数据的二维结构。与其他深度学习结构相比，卷积神经网络在图像和语音识别方面能够给出更好的结果。

循环神经网络：

RNN 是用来处理序列数据的神经网络，其引入了具有“记忆”性质的结构单元，计算除了本次的输入，还包括上一次的计算结果。

4.神经网络的优化算法有哪些(类似 sgd、adam、adadelta...)

sgd：随机梯度下降，对每个训练样本进行参数更新

mini-batch sgd：利用每个批次中的 n 个训练样本进行更新

adagrad: 学习率自动更新

adam:带有动量概念的学习率自动更新

5.CNN 是一种什么结构的网络

CNN 是常用于计算机视觉的一种前馈神经网络，包含卷积层这种利用局部感知野特性和权值共享保持学习能力降低参数的层次，和池化层这种具备采样操作的层次。常见层次还包括激励层和全连接层。

6.介绍一下 CNN 中的典型层次(layer)对应的操作和作用

卷积计算层：卷积核在上一级输入层上通过逐一滑动窗口计算，卷积核中的每一个参数都相当于传统神经网络中的权值参数。通过卷积计算可以从原始数据中提取特征。

激励层：利用 sigmoid、Relu 等非线性函数进行非线性变换。

池化层：缩减数据维度，降低计算量，减少过拟合。常用的池化方式包括：Max-Pooling 和 Mean-Pooling 等。

7. 腾讯优图--选择传统机器学习还是深度学习的标准是什么

于数据挖掘和处理类的问题，使用一般的机器学习方法，需要提前做大量的特征工程工作，而且特征工程的好坏会在很大程度上决定最后效果的优劣（也就是常说的一句话：数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已）。

使用深度学习的话，特征工程就没那么重要了，特征只需要做些预处理就可以了，因为它可以自动完成传统机器学习算法中需要特征工程才能实现的任务，特别是在图像和声音数据的处理中更是如此，但模型结构会比较复杂，训练较为麻烦。另一个方面，虽然深度学习让我们可以省去特征工程这一较为繁琐的过程，但也让我们失去了对特征的认识，如特征的重要性等

如何选择或衡量这两种方法：

第一看数据量，比如训练数据量达到百万以上，深度学习的方法会比较有优势。如果样本集不是大样本，那么特征工程加传统的机器学习方法使用起来泛化能力会更好。

第二看是否需要对结果有较强的解释性和可调节性，解释性是说我们能够了解到产生该输出结果的原因，这样我们能够知道特征的重要程度，并在出错时能够对错误原因进行分析。可调节性是指在出错或有特征的增删时，能够方便的对原模型进行修正以满足新的要求。在这一方面，一般的机器学习方法有一定的优势。

各自的优势领域：

深度学习：图像处理，自然语言处理等，因为图像、语言、文本都较难进行特征工程，交给深度学习是一个很好的选择。

机器学习：金融风控，量化分析，推荐系统，广告预测等，因为需要较好的可解释性，会更多的采用传统机器学习方法。

以上的领域，机器学习和深度学习都可以做，但因为各自的特点和要求，因此会有相对优势的偏向。

8.BN(Batch Normalization)层的作用

Batch Normalization 在实际工程中被证明了能够缓解神经网络难以训练的问题，BN 具有的有事可以总结为以下三点：

（1）BN 使得网络中每层输入数据的分布相对稳定，加速模型学习速度

BN 通过规范化与线性变换使得每一层网络的输入数据的均值与方差都在一定范围内，使得后一层网络不必不断去适应底层网络中输入的变化，从而实现了网络中层与层之间的解耦，允许每一层进行独立学习，有利于提高整个神经网络的学习速度。

（2）BN 使得模型对网络中的参数不那么敏感，简化调参过程，使得网络学习更加稳定

（3）BN 允许网络使用饱和性激活函数（例如 sigmoid，tanh 等），缓解梯度消失问题

在不使用 BN 层的时候，由于网络的深度与复杂性，很容易使得底层网络变化累积到上层网络中，导致模型的训练很容易进入到激活函数的梯度饱和区；通过 normalize 操作可以让激活函数的输入数据落在梯度非饱和区，缓解梯度消失的问题；另外通过自适应学习 与 又让数据保留更多的原始信息。

（4）BN 具有一定的正则化效果

在 Batch Normalization 中，由于我们使用 mini-batch 的均值与方差作为对整体训练样本均值与方差的估计，尽管每一个 batch 中的数据都是从总体样本中抽样得到，但不同 mini-batch 的均值与方差会有所不同，这就为网络的学习过程中增加了随机噪音，与 Dropout 通过关闭神经元给网络训练带来噪音类似，在一定程度上对模型起到了正则化的效果。

9.神经网络训练过程中，我们一般会调整哪些超参数

学习率、正则化参数、神经网络的层数、每层神经元个数、学习回合数 Epoch、minibatch、输出神经元的编码方式、损失函数、权重初始化方法、激活函数、训练数据规模。

10.训练过程中，batch size对loss变化的影响(batch size大和小，loss随时间有什么变化)？

一般来说，在合理的范围之内，越大的 batch size 使下降方向越准确，震荡越小；但 batch size 如果过大，则可能会出现局部最优的情况。loss 不会再下降。

小的 bath size 引入的随机性更大，如果过小会难以达到收敛，loss 不会下降。(小 batch size 在 loss 曲线看，loss 的抖动会非常大)

11.Dropout的为什么可以缓解过拟合

(1).Dropout 可以理解成构建多个神经网络取平均，不同的神经网络互相作用，可以减少各自的过拟合。

(2).Dropout 使神经元间的连接并不固定，使特征可以被更多神经元学习到，增强了网络的鲁棒性。

12.什么是感受野

某一层特征图中的一个 cell，对应到原始输入的响应的大小区域。

13.在优化算法的策略上，为什么先用Adam算法，后用SGD算法？

Adam 的收敛速度比 SGD 要快，但最终收敛的结果并没有 SGD 好。主要是后期 Adam 的学习率太低，影响了有效的收敛。所以我们可以前期用 Adam，享受 Adam 快速收敛的优势；后期切换到 SGD，慢慢寻找最优解。

14.什么是周期学习率？

学习率是深度神经网络中很重要的一个超参数，指代的是我们在梯度下降时参数更新的步长，选取合适的学习率十分重要，太小了会减慢收敛，太大会在最优点旁边左右摆动。而 Cyclical Learning Rates（CLR）是一种自适应学习率的方法，它可以和优化器结合，让学习率在一个范围内周期性地变化，从而加速收敛并提升模型性能。有时候会利用周期学习率让模型收敛到多个 loss 局部最低点，以便后续进行模型集成。

15. 怎么判断神经网络是否过拟合，缓解神经网络过拟合的方法有哪些?

(1).根据学习曲线判断是否过拟合。

(2).观察训练数据和验证数据的效果差异

缓解过拟合的方法：

(1).增加训练数据数

a.发生过拟合最常见的现象就是数据量太少而模型太复杂

b.过拟合是由于模型学习到了数据的一些噪声特征导致，增加训练数据的量能够减少噪声的影响，让模型更多地学习数据的一般特征

c.增加数据量有时可能不是那么容易，需要花费一定的时间和精力去搜集处理数据

d.利用现有数据进行扩充或许也是一个好办法。例如在图像识别中，如果没有足够的图片训练，可以把已有的图片进行旋转，拉伸，镜像，对称等，这样就可以把数据量扩大好几倍而不需要额外补充数据

e.注意保证训练数据的分布和测试数据的分布要保持一致，二者要是分布完全不同，那模型预测真可谓是对牛弹琴了

(2).使用正则化约束

在代价函数后面添加正则化项，可以避免训练出来的参数过大从而使模型过拟合。使用正则化缓解过拟合的手段广泛应用，不论是在线性回归还是在神经网络的梯度下降计算过程中，都应用到了正则化的方法。常用的正则化有 L1 正则和 L2 正则，具体使用哪个视具体情况而定，一般 L2 正则应用比较多

(3).减少特征数

欠拟合需要增加特征数，那么过拟合自然就要减少特征数。去除那些非共性特征，可以提高模型的泛化能力

(4).调整参数和超参数

不论什么情况，调参是必须的

(5).降低模型的复杂度

欠拟合要增加模型的复杂度，那么过拟合正好反过来

(6).使用 Dropout

这一方法只适用于神经网络中，即按一定的比例去除隐藏层的神经单元，使神经网络的结构简单化

(7).提前结束训练

即 early stopping，在模型迭代训练时候记录训练精度(或损失)和验证精度(或损失)，倘若模型训练的效果不再提高，比如训练误差一直在降低但是验证误差却不再降低甚至上升，这时候便可以结束模型训练了

16. 全卷积的神经网络有什么优点

全卷积的神经网络能够端到端得到每个像素的预测结果，同时保留了原始输入图像中的空间信息, 最后在上采样的特征图上进行逐像素分类。

17. 解释一下 1×1 卷积的原理，它主要用来干什么？

卷积核（convolutional kernel）：可以看作对某个局部的加权求和；它是对应局部感知，它的原理是在观察某个物体时我们既不能观察每个像素也不能一次观察整体，而是先从局部开始认识，这就对应了卷积。卷积核的大小一般有 1x1,3x3 和 5x5 的尺寸（一般是奇数 x 奇数）。

1*1 卷积作用是：(1)升/降特征的维度，这里的维度指的是通道数（厚度），而不改变图片的宽和高。(2)增加网络的深度。1x1 的卷积核虽小，但也是卷积核，加 1 层卷积，网络深度自然会增加。

18. 梯度下降法如果陷入局部最优怎么办？

(1)可尝试随机地增大学习速率/学习步长

(2)在每个 epoch/iteration 后需要把数据洗牌一遍，让每个 batch 的数据都不一样。

(3)可以尝试设置不同的初始值

(4)尝试使用其它优化方法如 Momentum 梯度下降法等

19.腾讯优图--神经网络为什么会出现梯度弥散(gradient vanish)问题，梯度爆炸呢

梯度消失：梯度趋近于零，网络权重无法更新或更新的很微小，网络训练再久也不会有效果；

梯度爆炸：梯度呈指数级增长，变的非常大，然后导致网络权重的大幅更新，使网络变得不稳定。

Sigmoid 导数的取值范围在 0~0.25 之间，而我们初始化的网络权值 w 通常都小于 1，因此，当层数增多时，小于 0 的值不断相乘，最后就导致梯度消失的情况出现。同理，梯度爆炸的问题也就很明显了，就是当权值 w 过大时，导致 $|\sigma'(z)w|>1$，最后大于 1 的值不断相乘，就会产生梯度爆炸。

梯度消失和梯度爆炸本质上是一样的，都是因为网络层数太深而引发的梯度反向传播中的连乘效应。

20.卷积神经网络中的卷积层，为什么需要 padding

a.保持边界信息，如果不加 padding，边界信息只会被卷积核扫描一次

b.可以通过 padding 对尺寸差异图片进行补齐

如果不加 padding，每次经过卷积层，feature map 都会变小，这样若干层卷积层之后，feature map 就很小了，通过 padding 可以维持 feature map 的 size

21.为什么使用 squeezenet 网络？它是一个怎样的网络

SqueezeNet 设计目标不是为了得到最佳的 CNN 识别精度，而是希望简化网络复杂度，同时达到 public 网络的识别精度。所以 SqueezeNet 主要是为了降低 CNN 模型参数数量而设计的。

SqueezeNet 小结：

1.Fire module 与 GoogLeNet 思想类似，采用 1*1 卷积对 feature map 的维数进行「压缩」，从而达到减少权值参数的目的；

2.采用与 VGG 类似的思想——堆叠的使用卷积，这里堆叠的使用 Fire module

22.阿里技术开发部-CNN 特性？

(1)平移不变性

简单来说，平移不变性（translation invariant）指的是 CNN 对于同一张图及其平移后的版本，都能输出同样的结果。这对于图像分类（image classification）问题来说肯定是最理想的，因为对于一个物体的平移并不应该改变它的类别。

(2)平移等价性/等变性 Equivalen

CNN 中 conv 层对应的是“等变性”（Equivariance），由于 conv 层的卷积核对于特定的特征才会有较大激活值，所以不论上一层特征图谱（feature map）中的某一特征平移到何处，卷积核都会找到该特征并在此处呈现较大的激活值。这应该就是“等变性”。

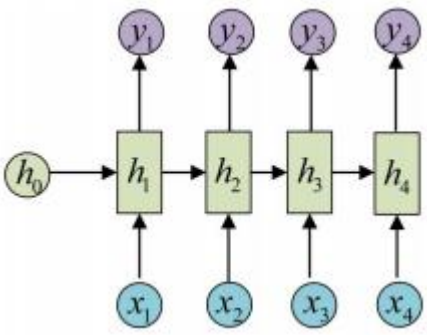
(3)尺度不变性

对于尺度不变性，是没有或者说具有一定的不变性（尺度变化不大）。

(4)旋转不变性

23.RNN 是一种什么样的神经网络

RNN 是用来处理序列数据的神经网络，其引入了具有“记忆”性质的结构单元，计算除了本次的输入，还包括上一次的计算结果。经典 RNN 结构示意图：

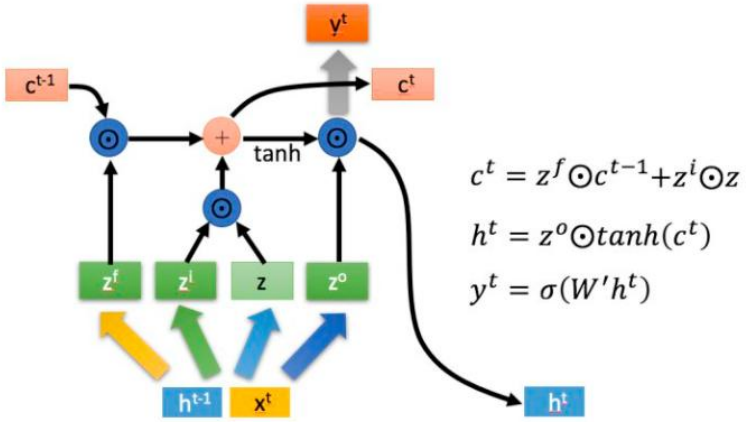


它的输入是 x1, x2, ...xn，输出为 y1, y2, ...yn，也就是说，输入和输出序列必须要是等长的。由于这个限制的存在，经典 RNN 的适用范围比较小，但也有一些问题适合用经典的 RNN 结构建模，如：

1.计算视频中每一帧的分类标签。因为要对每一帧进行计算，因此输入和输出序列等长。

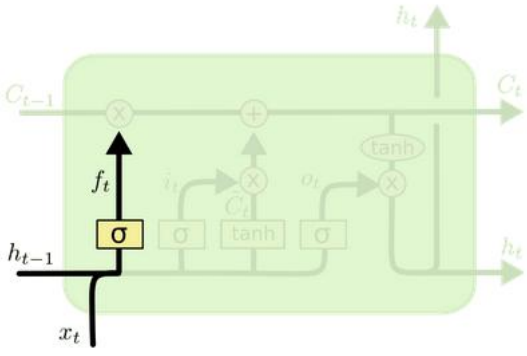
2.输入为字符，输出为下一个字符的概率。这就是著名的 Char RNN（详细介绍请参考：The Unreasonable Effectiveness of Recurrent Neural Networks，Char RNN 可以用来生成文章、诗歌，甚至是代码。)

24. 介绍一下 LSTM 的结构，为什么 LSTM 能缓解 BPTT 中的 gradient vanish



逐步理解 LSTM：

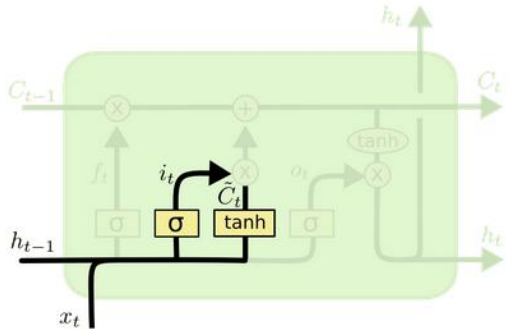
在我们 LSTM 中的第一步是决定我们会从细胞状态中丢弃什么信息。这个决定通过一个称为“忘记门”的结构完成。该忘记门会读取上一个输出 h_{t-1} 和当前输入 x_t，做一个 Sigmoid 的非线性映射，然后输出一个向量（该向量每一个维度的值都在 0 到 1 之间，1 表示完全保留，0 表示完全舍弃，相当于记住了重要的，忘记了无关紧要的），最后与细胞状态 C_{t-1} 相乘。



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

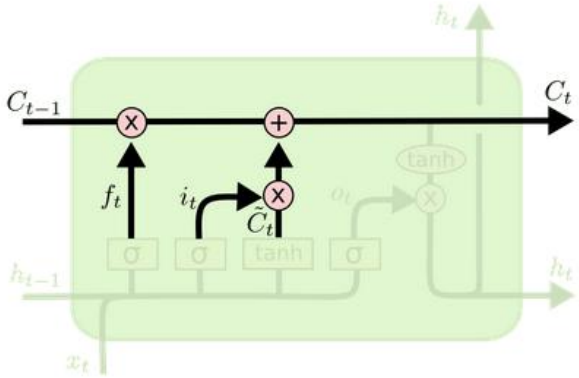
下一步是确定什么样的新信息被存放在细胞状态中。这里包含两个部分。第一，sigmoid 层称“输入门层”决定什么值我们将要更新。然后，一个 tanh 层创建一个新的候选值向量，$\tilde{C}_t$，会被加入到状态中。下一步，我们会讲这两个信息

来产生对状态的更新。



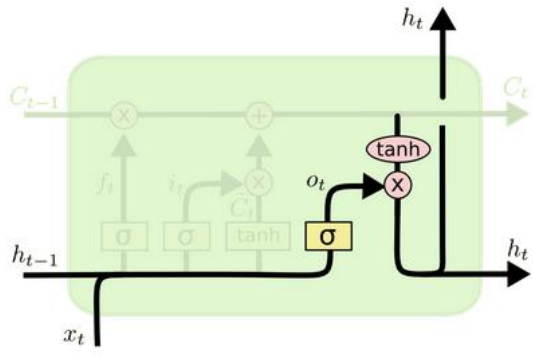
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

现在是更新旧细胞状态的时间了，C_{t-1} 更新为 C_t 。前面的步骤已经决定了将会做什么，我们现在就是实际去完成。我们把旧状态与 f_t 相乘，丢弃掉我们确定需要丢弃的信息。接着加上 $i_t * \widetilde{C}_t$。这就是新的候选值，根据我们决定更新每个状态的程度进行变化。



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

最终，我们需要确定输出什么值。这个输出将会基于我们的细胞状态，但是也是一个过滤后的版本。首先，我们运行一个 sigmoid 层来确定细胞状态的哪个部分将输出出去。接着，我们把细胞状态通过 tanh 进行处理（得到一个在-1 到 1 之间的值）并将它和 sigmoid 门的输出相乘，最终我们仅仅会输出我们确定输出的那部分。



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

以上就是 LSTM 的内部结构。通过门控状态来控制传输状态，记住需要长时间记忆的，忘记不重要的信息；而不像普通的 RNN 那样只能够“呆萌”地仅有一种记忆叠加方式。对很多需要“长期记忆”的任务来说，尤其好用。但也因为引入了很多内容，导致参数变多，也使得训练难度加大了很多。因此很多时候我们往往会使用效果和 LSTM 相当但参数更少的 GRU 来构建大训练量的模型。

25. GRU 的结构和 LSTM 有什么区别

(1) GRU 有两个门（重置门与更新门），而 LSTM 有三个门（输入门、遗忘门和输出门）。

(2) GRU 并不会控制并保留内部记忆（c_t），且没有 LSTM 中的输出门。

(3) LSTM 中的输入与遗忘门对应于 GRU 的更新门，重置门直接作用于前面的隐藏状态。

(4) GRU 参数更少因此更容易收敛。

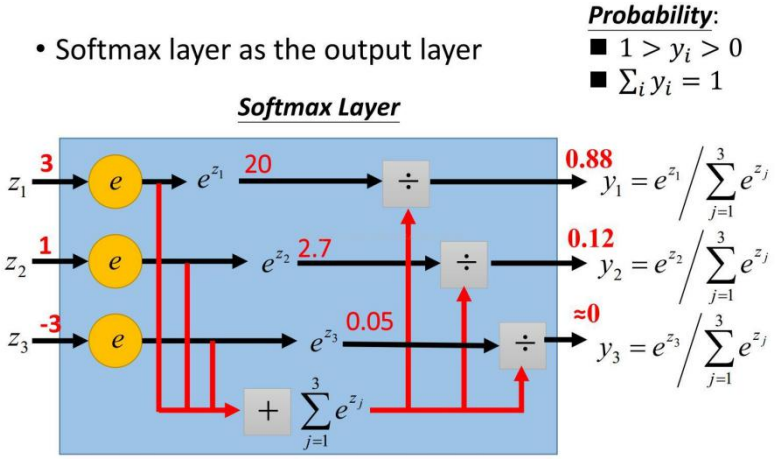
26. 介绍一下 softmax

softmax 用于多分类过程中，它将多个神经元的输出，映射到（0,1）区间内，可以看成概率来理解，从而来进行多分类。

假设我们有一个数组，V，Vi 表示 V 中的第 i 个元素，那么这个元素的 softmax 值就是：

$$S_i = \frac{e^i}{\sum_j e^j}$$

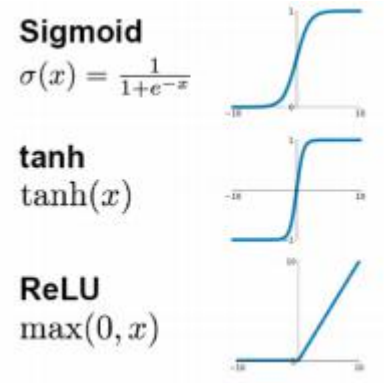
更形象的如下图表示：



从上图可知，softMax 的结果相当于输入图像被分到每个标签的概率分布，该函数是单调增函数，即输入值越大，输出也就越大，输入图像属于该标签的概率就越大。对 softmax 的结果计算交叉熵分类损失函数为：$L_i = -\log(\frac{e^{f_{y_i}}}{\sum_j e_j})$。取 log 里面的值就是这组数据正确分类的 Softmax 值，它占的比重越大，这个样本的 Loss 也就越小，这种定义符合我们的要求.

27.请问人工神经网络中为什么 ReLU 要好过于 tanh 和 Sigmoid function?

先看 Sigmoid、tanh 和 ReLU 的函数图：



第一，采用Sigmoid 等函数，算激活函数时（指数运算），计算量大，反向传播求误差梯度时，求导涉及除法和指数运算，计算量相对大，而采用ReLU 激活函数，整个过程的计算量节省很多。

第二，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的情况（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），这种现象称为饱和，从而无法完成深层网络的训练。而ReLU 就不会有饱和倾向，不会有特别小的梯度出现。

第三，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（以及一些人的生物解释 balabala）。当然现在也有一些对 ReLU 的改进，比如 PReLU，random ReLU 等，在不同的数据集上会有一些训练速度上或者准确率上的改进，具体的大家可以找相关的 paper 看。

多加一句，现在主流的做法，会多做一步 batch normalization，尽可能保证每一层网络的输入具有相同的分布 1。而最新的 paper2，他们在加入bypass

connection 之后，发现改变 batch normalization 的位置会有更好的效果。

28. 什么样的数据集不适合用深度学习？

1、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。

2、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。

29. 如何确定是否出现梯度爆炸

训练过程中出现梯度爆炸会伴随一些细微的信号，如：

（1）模型无法从训练数据中获得更新（如低损失）；

（2）模型不稳定，导致更新过程中的损失出现显著变化；

（3）训练过程中，模型损失变成 NaN。

如果你发现这些问题，那么你需要仔细查看是否出现梯度爆炸问题。以下是一些稍微明显一点的信号，有助于确认是否出现梯度爆炸问题。

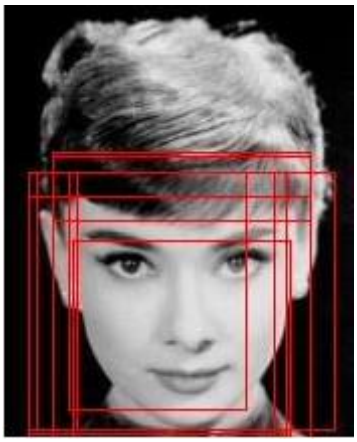
（1）训练过程中模型梯度快速变大；

（2）训练过程中模型权重变成 NaN 值；

（3）训练过程中，每个节点和层的误差梯度值持续超过 1.0。

30. 什么是非极大值抑制（NMS）？

R-CNN 会从一张图片中找出 n 个可能是物体的矩形框，然后为每个矩形框为做类别分类概率：



就像上面的图片一样，定位人脸，最后算法就找出一堆的方框，我们需要判别哪些矩形框是没用的。非极大值抑制的方法是：先假设有 6 个矩形框，根据分类器的类别分类概率做排序，假设从小到大属于车辆的概率分别为 A、B、C、D、E、F。

(1)从最大概率矩形框 F 开始，分别判断 A~E 与 F 的重叠度 IOU 是否大于某个设定的阈值;

(2)假设 B、D 与 F 的重叠度超过阈值，那么就扔掉 B、D；并标记第一个矩形框 F，是我们保留下来的。

(3)从剩下的矩形框 A、C、E 中，选择概率最大的 E，然后判断 E 与 A、C 的重叠度，重叠度大于一定的阈值，那么就扔掉；并标记 E 是我们保留下来的第二个矩形框。

就这样一直重复，找到所有被保留下来的矩形框。

非极大值抑制（NMS）顾名思义就是抑制不是极大值的元素，搜索局部的极大值。这个局部代表的是一个邻域，邻域有两个参数可变，一是邻域的维数，二是邻域的大小。这里不讨论通用的 NMS 算法，而是讨论用于在目标检测中提取分数最高的窗口的。例如在行人检测中，滑动窗又经提取特征，经分类器分类识别后，每个窗又都会得到一个分数。但是滑动窗又会导致很多窗又与其他窗又存在包含或者大部分交叉的情况。这时就需要用到 NMS 来选取那些邻域里分数最高（是行人的概率最大），并且抑制那些分数低的窗又。

31. 当神经网络的调参效果不好时，从哪些角度思考?

1）是否找到合适的损失函数？（不同问题适合不同的损失函数）（理解不同损失函数的适用场景）。

2）batch size 是否合适？batch size 太大 -> loss 很快平稳，batch size 太小 -> loss 会震荡（理解 mini-batch）。

3）是否选择了合适的激活函数？（各个激活函数的来源和差异）

4）学习率，学习率小收敛慢，学习率大loss 震荡（怎么选取合适的学习率）。

5）是否选择了合适的优化算法？（比如 adam）（理解不同优化算法的适用场景）。

6）是否过拟合？(深度学习拟合能力强，容易过拟合)（理解过拟合的各个解决方案）。

a. Early Stopping

b. Regularization（正则化）

c. Weight Decay（收缩权重）

d. Dropout（随机失活）

e. 调整网络结构

32. Adam 算法的原理?

Adam 是一种可以替代传统随机梯度下降过程的一阶优化算法，它能基于训练数据迭代地更新神经网络权重。

Adam 算法和传统的随机梯度下降不同。随机梯度下降保持单一的学习率（即 alpha）更新所有的权重，学习率在训练过程中并不会改变。而 Adam 通过计算梯度的一阶矩估计和二阶矩估计而为不同的参数设计独立的自适应性学习率。

Adam 算法的提出者描述其为两种随机梯度下降扩展式的优点集合，即：

r.适应性梯度算法（AdaGrad）为每一个参数保留一个学习率以提升在稀疏梯度（即自然语言和计算机视觉问题）上的性能。

s.均方根传播（RMSProp）基于权重梯度最近量级的均值为每一个参数适应性地保留学习率。这意味着算法在非稳态和在线问题上有很有优秀的性能。

Adam 算法同时获得了 AdaGrad 和 RMSProp 算法的优点。Adam 不仅如 RMSProp 算法那样基于一阶矩均值计算适应性参数学习率，它同时还充分利用了梯度的二阶矩均值（即有偏方差/uncentered variance）。

具体来说，算法计算了梯度的指数移动均值(exponential moving average),超参数 beta1 和 beta2 控制了这些移动均值的衰减率。移动均值的初始值和 beta1、beta2 值接近于 1（推荐值），因此矩估计的偏差接近于 0。该偏差通过首先计算带偏差的估计而后计算偏差修正后的估计而得到提升。如果对具体的实现细节和推导过程感兴趣，可以继续阅读该第二部分和原论文。

Adam 在深度学习领域内是十分流行的算法，因为它能很快地实现优良的结果。经验性结果证明 Adam 算法在实践中性能优异，相对于其他种类的随机优化算法具有很大的优势。

33. 请说下常见优化方法各自的优缺点？

比如：（BGD、SGD、MBGD、Momentum、NAG、Adagrad、Adadelta、RMSprop、Adam）

梯度下降法深入理解

对于优化算法，优化的目标是网络模型中的参数θ（是一个集合，θ1、θ2、θ3.....）目标函数为损失函数 L = 1/N ∑ Li （每个样本损失函数的叠加求均值）。这个损失函数 L 变量就是θ，其中 L 中的参数是整个训练集，换句话说，目标函数（损失函数）是通过整个训练集来确定的，训练集全集不同，则损失函数的图像也不同。

那么为何在 mini-batch 中如果遇到鞍点/局部最小值点就无法进行优化了呢？因为在这些点上，L 对于θ的梯度为零，换句话说，对θ每个分量求偏导数，带入训练集全集，导数为零。

对于 SGD/MBGD而言，每次使用的损失函数只是通过这一个小批量的数据确定的，其函数图像与真实全集损失函数有所不同，所以其求解的梯度也含有一定的随机性，在鞍点或者局部最小值点的时候，震荡跳动，因为在此点处，如果是训练集全集带入即 BGD，则优化会停止不动，如果是 minibatch 或者 SGD，每次找到的梯度都是不同的，就会发生震荡，来回跳动。

优化器算法简述

首先来看一下梯度下降最常见的三种变形 BGD，SGD，MBGD，这三种形式的区别就是取决于我们用多少数据来计算目标函数的梯度，这样的话自然就涉及到一个 trade－off，即参数更新的准确率和运行时间。

(1).Batch Gradient Descent（BGD）

梯度更新规则:

BGD 采用整个训练集的数据来计算 cost function 对参数的梯度：

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

缺点：

由于这种方法是在一次更新中，就对整个数据集计算梯度，所以计算起来非常慢，遇到很大量的数据集也会非常棘手，而且不能投入新数据实时更新模型。

```
for i in range(nb_epochs):
    params_grad = evaluate_gradient(loss_function, data, params)
    params = params - learning_rate*params_grad
```

我们会事先定义一个迭代次数 epoch, 首先计算梯度向量 params_grad，然后沿着梯度的方向更新参数 params，learning rate 决定了我们每一步迈多大。

Batch gradient descent 对于凸函数可以收敛到全局极小值，对于非凸函数可以收敛到局部极小值。

(2).Stochastic Gradient Descent (SGD)

梯度更新规则: 和 BGD 的一次用所有数据计算梯度相比, SGD 每次更新时对每个样本进行梯度更新，对于很大的数据集来说，可能会有相似的样本，这样 BGD 在计算梯度时会出现冗余，而 SGD 一次只进行一次更新，就没有冗余，而且比较快，并且可以新增样本。

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for example in data:
        params_grad = evaluate_gradient(loss_function, example, params)
        params = params - learning_rate * params_grad
```

看代码，可以看到区别，就是整体数据集是个循环，其中对每个样本进行一次参数更新。

随机梯度下降是通过每个样本来迭代更新一次，如果样本量很大的情况，那么可能只用其中部分的样本，就已经将 theta 迭代到最优解了，对比上面的批量梯度下降，迭代一次需要用到十几万训练样本，一次迭代不可能最优，如果迭代 10 次的话就需要遍历训练样本 10 次。

缺点是 SGD 的噪音较 BGD 要多，使得 SGD 并不是每次迭代都向着整体最优化方向。所以虽然训练速度快，但是准确度下降，并不是全局最优。虽然包含一定的随机性，但是从期望上来看，它是等于正确的导数的。

缺点：

SGD 因为更新比较频繁，会造成 Cost Function 有严重的震荡。

BGD 可以收敛到局部极小值，当然 SGD 的震荡可能会跳到更好的局部极小值处。

当我们稍微减小 learning rate，SGD 和 BGD 的收敛性是一样的。

(3).Mini-Batch Gradient Descent (MBGD)

梯度更新规则： MBGD 每一次利用一小批样本，即 n 个样本进行计算，这样它可以降低参数更新时的方差，收敛更稳定，另一方面可以充分地利用深度学习库中高度优化的矩阵操作来进行更有效的梯度计算。

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

和 SGD 的区别是每一次循环不是作用于每个样本，而是具有 n 个样本的批次。

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for batch in get_batches(data, batch_size=50):
        params_grad = evaluate_gradient(loss_function, batch, params)
        params = params - learning_rate * params_grad
```

超参数设定值: n 一般取值在 50～256

缺点：（两大缺点）

1)不过 Mini-batch gradient descent 不能保证很好的收敛性，learning rate 如果选择的太小，收敛速度会很慢，如果太大，loss function 就会在极小值处不停

地震荡甚至偏离。（有一种措施是先设定大一点的学习率，当两次迭代之间的变化低于某个阈值后，就减小 learning rate，不过这个阈值的设定需要提前写好，这样的话就不能够适应数据集的特点）。

对于非凸函数，还要避免陷于局部极小值处，或者鞍点处，因为鞍点周围的 error 是一样的，所有维度的梯度都接近于 0，SGD 很容易被困在这里。（会在鞍点或者局部最小点震荡跳动，因为在此点处，如果是训练集全集带入即 BGD，则优化会停止不动，如果是 mini-batch 或者 SGD，每次找到的梯度都是不同的，就会发生震荡，来回跳动）

2)SGD 对所有参数更新时应用同样的 learning rate，如果我们的数据是稀疏的，我们更希望对出现频率低的特征进行大一点的更新。LR 会随着更新的次数逐渐变小。

(4).Momentum

SGD 在 ravines 的情况下容易被困住，ravines 就是曲面的一个方向比另一个方向更陡，这时 SGD 会发生震荡而迟迟不能接近极小值：

Image 2: SGD without momentum

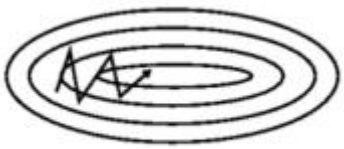


Image 3: SGD with momentum

梯度更新规则:

Momentum 通过加入 γv_t−1，可以加速 SGD，并且抑制震荡

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

$$\theta = \theta - v_t$$

当我们将一个小球从山上滚下来时，没有阻力的话，它的动量会越来越大，但是如果遇到了阻力，速度就会变小。

加入的这一项，可以使得梯度方向不变的维度上速度变快，梯度方向有所改变的维度上的更新速度变慢，这样就可以加快收敛并减小震荡。

超参数设定值: 一般 γ 取值 0.9 左右。

缺点：

这种情况相当于小球从山上滚下来时是在盲目地沿着坡滚，如果它能具备一

些先知，例如快要上坡时，就知道需要减速了的话，适应性会更好。

(5).Nesterov Accelerated Gradient

梯度更新规则:

用$\theta - \gamma v_{t-1}$来近似当做参数下一步会变成的值，则在计算梯度时，不是在当前位置，而是未来的位置上

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$
$$\theta = \theta - v_t$$

超参数设定值: 一般 γ 仍取值 0.9 左右。

效果比较：

蓝色是 Momentum 的过程，会先计算当前的梯度，然后在更新后的累积梯度后会有一个大的跳跃。

而 NAG 会先在前一步的累积梯度上(brown vector)有一个大的跳跃，然后衡量一下梯度做一下修 正(red vector)，这种语气的更新可以避免我们走的太快。

NAG 可以使 RNN 在很多任务上有更好的表现。

目前为止，我们可以做到，在更新梯度时顺应 loss function 的梯度来调整速度，并且对 SGD 进行加速。

我们还希望可以根据参数的重要性而对不同的参数进行不同程度的更新。

(6).Adagrad （Adaptive gradient algorithm）

这个算法就可以对低频的参数做较大的更新，对高频的做较小的更新，也因此，对于稀疏的数据它的表现很好，很好地提高了 SGD 的鲁棒性，例如识别 Youtube 视频里面的猫，训练 GloVe word embeddings，因为它们都是需要在低频的特征上有更大的更新。梯度更新规则:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i}$$

其中 g 为：t 时刻参数θ_i的梯度$g_{t,i} = \nabla_\theta J(\theta_i)$

如果是普通 SGD，那么θ_i在每一时刻的梯度更新公式为：

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$$

但这里的 learning rate η 也随 t 和 i 而变：

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i}$$

其中G_t是个对角矩阵，(i,i)元素就是 t 时刻参数θ_i的梯度平方和。

Adagrad 的优点是减少了学习率的手动调节

超参数设定值：一般 η 选取 0.01

(7).Adadelta

这个算法是对 Adagrad 的改进。

和 Adagrad 相比，就是分母的 G 换成了过去的梯度平方的衰减平均值，指数衰减平均值

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t$$

这个分母相当于梯度的均方根 root mean squared (RMS)，在数据统计分析中，将所有值平方求和，求其均值，再开平方，就得到均方根值 ，所以可以用 RMS 简写：

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t} g_t$$

其中 E 的计算公式如下，t 时刻的依赖于前一时刻的平均和当前的梯度：

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

梯度更新规则:

此外，还将学习率 η 换成了 RMS[Δθ]，这样的话，我们甚至都不需要提前设定学习率了：

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

超参数设定值: γ 一般设定为 0.9

(8).RMSprop

RMSprop 是 Geoff Hinton 提出的一种自适应学习率方法。

RMSprop 和 Adadelta 都是为了解决 Adagrad 学习率急剧下降问题的，

梯度更新规则:

RMSprop 与 Adadelta 的第一种形式相同：（使用的是指数加权平均，旨在消除梯度下降中的摆动，与 Momentum 的效果一样，某一维度的导数比较大，则指数加权平均就大，某一维度的导数比较小，则其指数加权平均就小，这样就保证了各维度导数都在一个量级，进而减少了摆动。允许使用一个更大的学习率 η）

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t$$

超参数设定值:

Hinton 建议设定 γ 为 0.9, 学习率 η 为 0.001。

(9).Adam：Adaptive Moment Estimation

这个算法是另一种计算每个参数的自适应学习率的方法。相当于 RMSprop + Momentum 除了像 Adadelta 和 RMSprop 一样存储了过去梯度的平方 vt 的指数衰减平均值 ，也像 momentum 一样保持了过去梯度 mt 的指数衰减平均值：

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

如果 mt 和 vt 被初始化为 0 向量，那它们就会向 0 偏置，所以做了偏差校正，通过计算偏差校正后的 mt 和 vt 来抵消这些偏差：

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v} = \frac{v_t}{1-\beta_2^t}$$

梯度更新规则:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \varepsilon}} \hat{m}_t$$

超参数设定值:

建议 β1 ＝ 0.9，β2 ＝ 0.999，ϵ ＝ 10e−8

实践表明，Adam 比其他适应性学习方法效果要好。

33.如何选择优化算法

如果数据是稀疏的，就用自适用方法，即 Adagrad, Adadelta, RMSprop, Adam。RMSprop, Adadelta, Adam 在很多情况下的效果是相似的。Adam 就是在 RMSprop 的基础上加了 biascorrection 和 momentum，随着梯度变的稀疏，Adam 比 RMSprop 效果会好。整体来讲，Adam 是最好的选择。很多论文里都会用 SGD，没有 momentum 等。SGD 虽然能达到极小值，但是比其它算法用的时间长，而且可能会被困在鞍点。如果需要更快的收敛，或者是训练更深更复杂的神经网络，需要用一种自适应的算法。

第四章-常用评价指标

1.简述一下 ROC 曲线，如何通过 ROC 曲线计算 AUC？

roc 曲线：接收者操作特征(receiveroperating characteristic),roc 曲线上每个点反映着对同一信号刺激的感受性。

横轴：负正类率(false postive rate, FPR)，特异度，划分实例中所有负例占所有负例的比例；(1-Specificity)

纵轴：真正类率(true postive rate, TPR)，灵敏度，Sensitivity(正类覆盖率)

计算出 ROC 曲线下面的面积，就是 AUC 的值。

2.目标检测中 mAP 怎么计算？

常规的 mAP 计算为（这是一个 N 类检测任务）：

(1)计算单张图片中 class1 的精度 P

(2)循环所有测试集图片，重复 1 过程求所有图片 P 的均值即为 class1 的 AP

(3)对剩余 N-1 类重复 1 和 2 过程，对每个类的 AP 求均值，即为 mAP

3.海康威视 CV 算法--什么是查准率，什么是查全率，请分别写出它们的公式？

我们以图片分类来举例，当然换成文本、语音等也是一样的。

Positive:

正样本。比如你要识别一组图片是不是猫，那么你预测某张图片是猫，这张图片就被预测成了正样本。

Negative:

负样本。比如你要识别一组图片是不是猫，那么你预测某张图片不是猫，这张图片就被预测成了负样本。

TP 一组预测为正样本的图片中，真的是正样本的图片数。

TN 一组预测为负样本的图片中，真的是负样本的图片数。

FP 一组预测为正样本的图片中，其实是负样本的图片数。又称“误检”

FN 一组预测为负样本的图片中，其实是正样本的图片数。又称“漏检”。

查准率/准确率 precision：一组预测为正样本的图片中，真的是正样本的图片所占的比例。计算公式为,

$$P = \frac{TP}{TP + FP}$$

查全率/召回率 recall：所有真的是正样本的图片中，被成功预测出来的图片所占的比例。

$$R = \frac{TP}{TP + FN}$$

4.海康威视 CV 算法--解释一下 AUC

在试图弄懂 AUC 和 ROC 曲线之前，要先理解混淆矩阵的定义。

混淆矩阵中有着 Positive、Negative、True、False 的概念，其意义如下：

(1)称预测类别为 1 的为 Positive（阳性），预测类别为 0 的为 Negative（阴性）。

(2)预测正确的为 True（真），预测错误的为 False（伪）。

对上述概念进行组合，就产生了如下的混淆矩阵：

		真实类别	
		1	0
预测类别	1 Positive （阳）	True Positive 真阳	False Positive 伪阳
	0 Negative （阴）	False Negative 伪阴	True Negative 真阴

然后，由此引出 True Positive Rate（真阳率）、False Positive（伪阳率）两个概念：

- $TPRate = \frac{TP}{TP + FN}$
- $FPRate = \frac{FP}{FP + TN}$

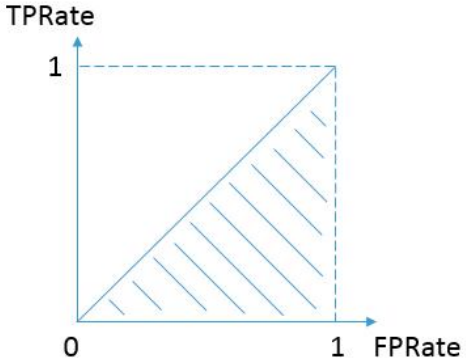
仔细看这两个公式，发现其实 TPRate 就是 TP 除以 TP 所在的列，FPRate 就是 FP 除以 FP 所在的列，二者意义如下：

(1)TPRate 的意义是所有真实类别为 1 的样本中，预测类别为 1 的比例。

(2)FPRate 的意义是所有真实类别为 0 的样本中，预测类别为 1 的比例。

理解了上述概念之后，就来说 AUC 和 ROC 的概念

按照定义，AUC 即 ROC 曲线下的面积，而 ROC 曲线的横轴是 FPRate，纵轴是 TPRate，当二者相等时，即 y=x，如下图:

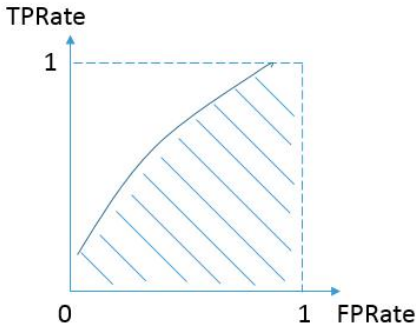


上图表示的意义是：对于不论真实类别是 1 还是 0 的样本，分类器预测为 1 的概率是相等的。

换句话说，分类器对于正例和负例毫无区分能力，和抛硬币没什么区别，一个抛硬币的分类器是我们能想象的最差的情况，因此一般来说我们认为 AUC 的

最小值为 0.5（当然也存在预测相反这种极端的情况，AUC 小于 0.5，这种情况相当于分类器总是把对的说成错的，错的认为是对的，那么只要把预测类别取反，便得到了一个 AUC 大于 0.5 的分类器）。

而我们希望分类器达到的效果是：对于真实类别为 1 的样本，分类器预测为 1 的概率（即 TPRate），要大于真实类别为 0 而预测类别为 1 的概率（即 FPRate），即 y＞x，因此大部分的 ROC 曲线长成下面这个样子：



最理想的情况下，既没有真实类别为 1 而错分为 0 的样本——TPRate 一直为 1，也没有真实类别为 0 而错分为 1 的样本——FP rate 一直为 0，AUC 为 1，这便是 AUC 的极大值。

AUC 就是衡量学习器优劣的一种性能指标。AUC 可通过对 ROC 曲线下各部分的面积求和而得。

AUC 的优势，AUC 的计算方法同时考虑了分类器对于正例和负例的分类能力，在样本不平衡的情况下，依然能够对分类器作出合理的评价。

5. 海康威视 CV 算法--为什么不用召回率精准率而用 AUC

召回率：把实际为真值的判断为真值的概率。

精确率：判断为真值，判断正确的概率（实际也为真值）。

recall = TP / (TP + FN)

precision = TP / (TP + FP)

一般说来，如果想要召回的多，精确率就要下降；想要精确率提升，召回的就少。因此，召回率与精确率是鱼与熊掌不可兼得。

AUC 它是一个用来评估分类模型性能的常见指标，优点是：

(1)适用于正负样本分布不一致的场景；

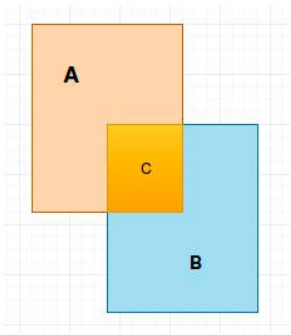
(2)对于分类器性能的评价，不限定单一的分类阈值；

6. IoU 是什么，怎么计算？mIoU 又怎么计算

在目标检测的评价体系中，有一个参数叫做 IoU ，简单来讲就是模型产生的目标窗又和原来标记窗又的交叠率。

可以简单的理解为：即检测结果 DetectionResult 与真实值 Ground Truth 的交集比上 它们的并集，即为检测的准确率 IoU ：

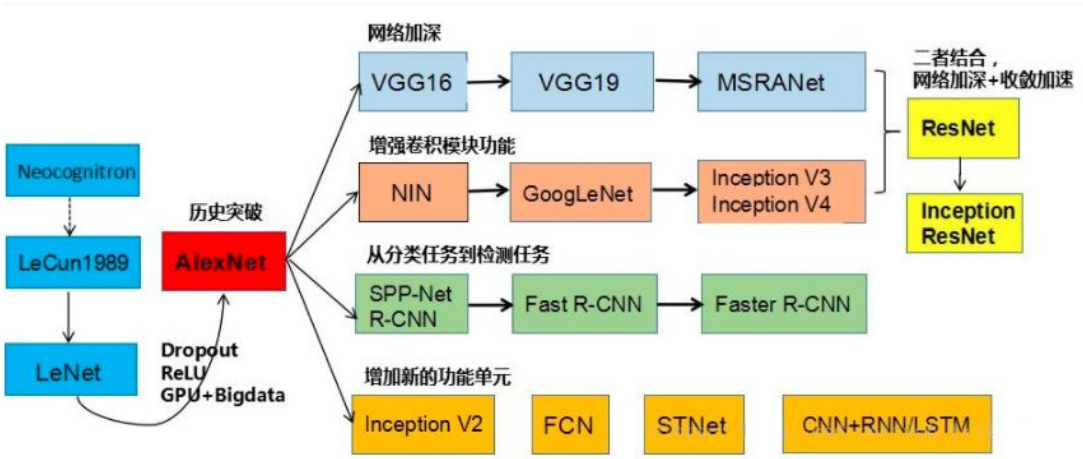
$$IOU = \frac{A \cap B}{A \cup B}$$



Mean Intersection over Union (MIoU)：计算每一类的 IoU 然后求平均。

第五章-神经网络模型

1. 腾讯优图--cnn 结构演化史



2.介绍一下 VGG/inception/ResNet

VGG 的最主要的思想就是增加网络深度，减小卷积核尺寸（3*3）。

inception 的主要思想是如何在不增加计算成本的前提下扩展神经网络。引入了 Inception module，同时为了降低计算量，使用 1×1 卷积来执行降维。

ResNet 的主要思想是通过学习残差，来构建深度更深的神经网络，达到更好的学习效果。

3.描述下 googLeNet 发展的几个过程

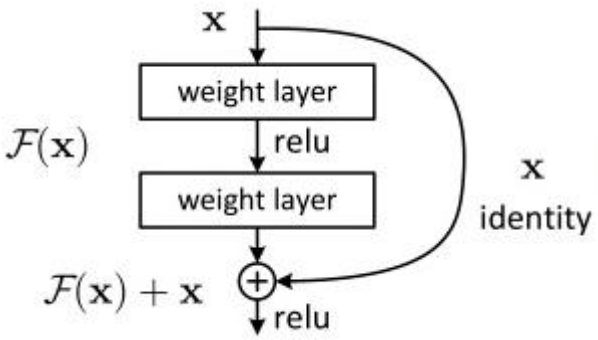
(1)Inception v1 的网络，将 1x1，3x3，5x5 的 conv 和 3x3 的 pooling，stack 在一起，一方面增加了网络的 width，另一方面增加了网络对尺度的适应性；

(2)v2 的网络在 v1 的基础上，进行了改进，一方面了加入了 BN 层，减少了 Internal Covariate Shift（内部 neuron 的数据分布发生变化），使每一层的输出都规范化到一个 N(0, 1)的高斯，另外一方面学习 VGG 用 2 个 3x3 的 conv 替代 inception 模块中的 5x5，既降低了参数数量，也加速计算；

(3)v3 一个最重要的改进是分解（Factorization），将 7x7 分解成两个一维的卷积（1x7,7x1），3x3 也是一样（1x3,3x1），这样的好处，既可以加速计算（多余的计算能力可以用来加深网络），又可以将 1 个 conv 拆成 2 个 conv，使得网络深度进一步增加，增加了网络的非线性，还有值得注意的地方是网络输入从 224x224 变为了 299x299，更加精细设计了 35x35/17x17/8x8 的模块；

(4)v4 研究了 Inception 模块结合 Residual Connection 能不能有改进？发现 ResNet 的结构可以极大地加速训练，同时性能也有提升，得到一个 Inception-ResNet v2 网络，同时还设计了一个更深更优化的 Inception v4 模型，能达到与 Inception-ResNet v2 相媲美的性能。

4. 画一下残差网络中残差块的结构，它相比之前的神经网络有什么优势，解决了什么问题？



深度残差网络有很多旁路的支线将输入直接连到后面的层，使得后面的层可以直接学习残差，这些支路就叫做 shortcut。传统的卷积层或全连接层在信息传递时，或多或少会存在信息丢失、损耗等问题。ResNet 在某种程度上解决了这个问题，通过直接将输入信息绕道传到输出，保护信息的完整性，整个网络则只需要学习输入、输出差别的那一部分，简化学习目标和难度。

5. ResNet 层次那么深，为什么不那么容易出现梯度弥散

传统的卷积神经网络模型，都以层叠卷积层的方式提高网络深度，从而提高识别精度。但层叠过多的卷积层会出现一个问题，就是梯度弥散(Vanishing)，backprop 无法把有效地把梯度更新到前面的网络层，导致前面的层参数无法更新。

ResNet 通过 skip connection 使梯度直接回传，减少了深层网络的梯度弥散现象。我们可以用公式表示一下

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i, w_i)$$

说明：输入是 x；F(x)相当于 residual，它只是普通神经网络的正向传播；输出是这两部分的加和 H(x) = F(x)（就是 residual）+ x（就是 shortcut，此代码 shortcut 部分做了一次卷积，也可以不做）。

之所以可以避免梯度消失问题，是因为反向传播时，ε 代表的是 loss 方程，由链式求导法得：

$$\frac{\partial \varepsilon}{\partial x_l}=\frac{\partial \varepsilon}{\partial x_L}\frac{\partial x_L}{\partial x_l}=\frac{\partial \varepsilon}{\partial x_L}(1+\frac{\partial}{\partial x_l}\sum_{i=1}^{L-1}F(x_i,w_i))$$

可以看出，反向传播的梯度由 2 项组成的：

(1)对 x 的直接映射，梯度为 1；

(2)通过多层普通神经网络映射结果为：$\frac{\partial}{\partial x_l}\sum_{i=1}^{L-1}F(x_i,w_i)$

即使新增的多层神经网络的梯度$\frac{\partial}{\partial x_l}\sum_{i=l}^{L-1}F(x_i,w_i)$为 0，那么输出结果也不会比传播前的 x 更差。同时也避免了梯度消失问题。

6.简述一下 Bilinear CNN 的工作流程？

Bilinear CNN，该模型中会有两个并列的网络 A 和 B(均由卷积层、池化层等构成)，两个网络在后边会通过外积合并到一起得到 bilinear vector，最后再通过 softmax 层。工作流程是：给模型喂进去输入图像后，网络 A 的作用是对物体 / 部件进行定位，即完成物体与局部区域检测工作，而网络 B 则是用来对网络 A 检测到的物体位置进行特征提取。两个网络相互协调作用，完成了细粒度图像分类过程中两个最重要的任务:物体、局部区域的检测与特征提取。

7.Faster-RCNN 相比于 Fast-RCNN 做了哪些改进？

在 Fast RCNN 的基础上，Faster RCNN 在性能上又有了进步。Faster RCNN 将特征抽取(feature extraction)，proposal 提取，bounding box regression，classification 都整合在了一个网络中，使得综合性能有较大提高，在检测速度方面尤为明显。对比 Fast-RCNN， 其实最重要的一点就是使用 RPN 来代替原来使用分割算法生成候选框的方式，极大的提升了检测框生成速度。总地来说，Faster RCNN 对 Fast RCNN 的改进点在于获得 region proposals 的速度要快很多。

8. 解释一下 RPN 网络的工作原理？

RPN 网络主要用于生成 region proposals，主要做两件事：一是把 feature map 分割成多个小区域，识别出哪些小区域是前景，哪些是背景，简称 RPN Classification；二是获取前景区域的大致坐标，简称 RPN bounding box regression；

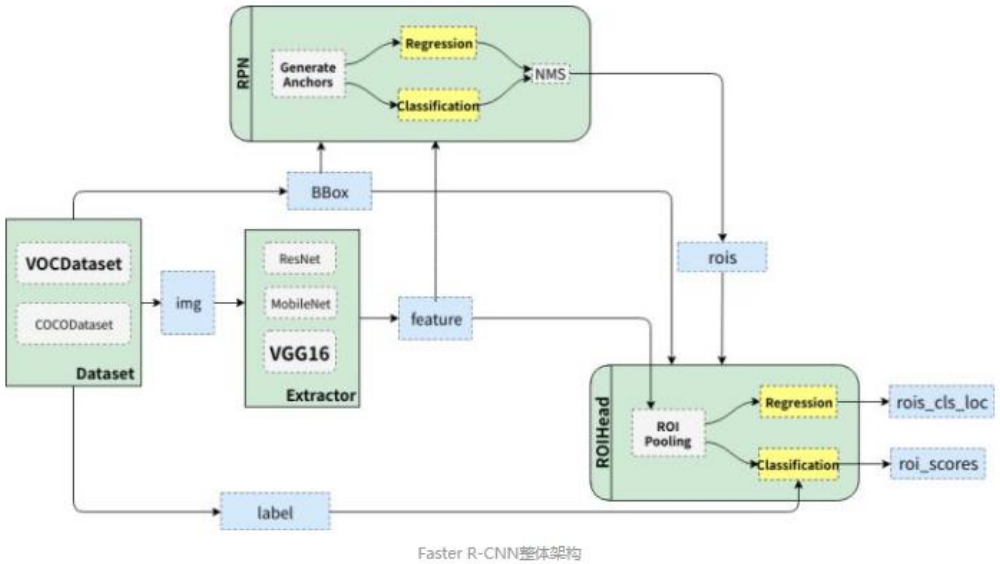
(1)RPN Classification 的过程就是个二分类的过程。先要在 feature map 上均匀的划分出 KxHxW 个区域（称为 anchor，K=9，H 是 feature map 的高度，W 是宽度），通过比较这些 anchor 和 ground truth 间的重叠情况来决定哪些 anchor 是前景，哪些是背景，也就是给每一个 anchor 都打上前景或背景的 label。有了 labels，你就可以对 RPN 进行训练使它对任意输入都具备识别前景、背景的能力。

(2)RPN bounding box regression 用于得出前景的大致位置。

RPN 训练时要把 RPN classification 和 RPN bounding box regression 的 loss 加到一起来实现联合训练。

9. 描述下 faster rcnn 流程，它的 loss 公式

Faster-rcnn 的流程:



Faster R-CNN整体架构

Faster R-CNN 主要分为四部分（图中四个绿色框）：

(1)Dataset：数据，提供符合要求的数据格式（目前常用数据集是 VOC 和 COCO）(2)Extractor： 利用 CNN 提取图片特征 features（原始论文用的是 ZF 和 VGG16，后来人们又用 ResNet101）

(3)RPN(*Region Proposal Network):* 负责提供候选区域 rois（每张图给出大概 2000 个候选框）

(4)RoIHead： 负责对 rois 分类和微调。对 RPN 找出的 rois，判断它是否包含目标，并修正框的位置和座标

Faster R-CNN 整体的流程可以分为三步：

(1)提特征： 图片（img）经过预训练的网络（Extractor），提取到了图片的特征（feature）

(2)Region Proposal： 利用提取的特征（feature），经过 RPN 网络，找出一定数量的 rois（region of interests）。

(3)分类与回归：将 rois 和图像特征 features，输入到 RoIHead，对这些 rois 进行分类，判断都属于什么类别，同时对这些 rois 的位置进行微调。

loss 函数：

Faster RCNN 的的损失主要分为 RPN 的损失和 Fast RCNN 的损失，计算公式如下，并且两部分损失都包括分类损失（cls loss）和回归损失（bbox regression loss）。

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

10. 阿里技术部开发部--YOLO 为什么对小物体不敏感，为什么 YOLO 一个格子可以检测到大物体

(1)对小物体检测不敏感。因为虽然每个 cell 都可以预测出 B 个 bounding box，但是在最终只选择 IOU 最高的 bounding box 作为物体检测输出，即：每个 cell 只能预测出一个物体。当物体较小时，所占画面比例较小，比如图像中包含牲畜群的时候，每个格子 kennel 包含多个物体，但是最后只能检测出其中的一个。

(2)YOLO 并不是将图片分割成 **单独的独立的** 网格 **分别** 输入模型进行预测，实际上，YOLO 网络最后的卷积层在原图上的感受野是远大于网格大小的，网格只是用于物体中心位置归属哪个网格的划分。

11. 阿里技术开发部--物体检测领域主流算法有哪些

R-CNN,Fast R-CNN, Faster R-CNN, SPP-Net, OverFeat, SSD, YOLO 系列等

12. 讲下 maskrcnn 结构

Mask R-CNN 是继承于 Faster R-CNN 的，Mask R-CNN 的方法通过添加一个与现有目标检测框回归并行的，用于预测目标掩码的分支来扩展 Faster R-CNN，通过添加一个用于在每个感兴趣区域（RoI）上预测分割掩码的分支来扩展 Faster R-CNN，就是在每个感兴趣区域（RoI）进行一个二分类的语义分割，在这个感兴趣区域同时做目标检测和分割，这个分支与用于分类和目标检测框回归的分支并行执行。

13. YOLOv1, v2, v3 的区别

从网络结构上说，YOLOv1 是卷积层、迟化层和全连接层的组合；

YOLOv2 在 v1 的基础上，去掉了全连接层，在每一个卷积层后边都添加了 BN（batch normalization)层，并且对每一批数据都做了归一化的预处理，这两个改变就可以提升了算法的速度；

YOLOv3 在网络结构上采用的是 Darknet-53，在 v2 的基础上每隔两层增加了一个 residual networks(残差网络),即 short cut 层，使用这种办法对于训练很深层的网络呢，可以解决梯度消失或者梯度爆炸的问题。

对于 yolov3 来说，坐标误差是和 YOLOv1 一样的，但是对于后三项，即 IOU 误差和分类误差来说，yolov3 使用的是二元交叉熵误差来替代。