前言：

博主参加了很多面试，应聘岗位是计算机视觉（深度学习方向），现将问到我的一些问题，总结如下，回答的有哪些不对，麻烦指正，大家也可以自己去查答案。特别说一下：面试时一定要把自己项目用到的东西涉及到的东西，都要搞懂，并且有目的性引导面试官问你准备的问题。

问题：

1、深度学习要这么深?

答：1、一个直观的解释，从模型复杂度角度。如果我们能够增强一个学习模型的复杂度，那么它的学习能力能够提升。如何增加神经网络的复杂度呢？要么变宽，即增加隐层网络神经元的个数；要么变深，即增加隐层的层数。当变宽的时候，只不过是增加了一些计算单元，增加了函数的个数，在变深的时候不仅增加了个数，还增加了函数间的嵌入的程度。

2、深度学习可以通过多个layer的转换学习更高维度的特征来解决更加复杂的任务。

3、那现在我们为什么可以用这样的模型？有很多因素，第一我们有了更大的数据；第二我们有强力的计算设备；第三我们有很多有效的训练技巧

4、像在ZFNet网络中已经体现，特征间存在层次性，层次更深，特征不变性强，类别分类能力越强，要学习复杂的任务需要更深的网络

2、如何解决数据不平衡问题?

答：1、利用重采样中的下采样和上采样，对小数据类别采用上采用，通过复制来增加数据，不过这种情况容易出现过拟合，建议用数据扩增的方法，对原有数据集进行翻转，旋转，平移，尺度拉伸，对比度，亮度，色彩变化来增加数据。对大数据类别剔除一些样本量。数据增强具体代码：常用数据增强方法代码

2、组合不同的重采样数据集：假设建立十个模型，选取小数据类1000个数据样本，然后将大数据类别10000个数据样本分为十份，每份为1000个，并训练十个不同的模型。

3、更改分类器评价指标： 在传统的分类方法中，准确率是常用的指标。 然而在不平衡数据分类中，准确率不再是恰当的指标，采用精准率即查准率P：真正例除以真正例与假正例之和。召回率即查全率F。真正例除以真正例与假反例之和。或者F1分数查全率和查准率加权平衡=2*P*R/(P+R)。

3、对于训练集与验证集测试集分布不同的处理办法

1、若训练集与验证集来自不同分布，比如一个网络爬虫获取的高清图像，一个是手机

不清晰图像，人工合成图像，比如不清晰图像，亮度高的图像。

2、两种来源的数据一个来源数据大比如20万张，一个来源数据小，如五千张小数据集是我们优化目标，一种情况是将两组数据合并在一起，然后随机分配到训练验证测试集中好处是，三个数据集来自同一分布。缺点：瞄准目标都是大数据那一类的数据，而不是我们的目标小数据集。另外一种情况是训练集全部用大数据集，开发与测试集都是小数据集数据，优点：瞄准目标，坏处是不同分布。

3、分析偏差和方差方法和同一分布的方法不一样，加一个训练开发集（从训练集留出一部分数据）。总共四个数据集，训练集、训练开发集、开发集、测试集。看训练开发集的准确率与训练集验证集的区别来判别式方差还是数据分布不匹配的造成的误差。具体看如下链接：https://blog.csdn.net/koala_tree/article/details/78319908

4、如何改善训练模型的效果呢?

答：1、通过提升数据，获取良好的数据。对数据预处理；零均值1方差化，数据扩充或者增强，

2、诊断网络是否过拟合欠拟合。通过偏差方差。正则化解决过拟合，早停法遏制过拟合。

3、通过学习率，激活函数的选择，改善网络全连接层个数啊层数啊，优化算法，随机梯度，RMSprop,动量，adam，使用batchnormlization.

3、权值初始化Xavier初始化，保持输入与输出端方差一致，避免了所有输出都趋向于0；

5、如何解决梯度爆炸与消失。

-答：1、预训练加微调 - 梯度剪切、权重正则（针对梯度爆炸） -

2、使用不同的激活函数 -

3、使用batchnorm -

4、使用残差结构 -

5、使用LSTM网络

6、你做过其他与职位申请相关项目吗，解释现在的硕士研究内容，有什么效果吗?

答：解答建议。自己做的事情和学的任何技能能够与申请的岗位建立联系。

7、为什么要使用许多小卷积核(如3x 3)而不是几个大卷积核?

这在VGGNet的原始论文中得到了很好的解释。原因有二：首先，您可以使用几个较小的核而不是几个较大的核来获得相同的感受野并捕获更多的空间上下文，但是使用较小的内核时，您使用的参数和计算量较少。其次，因为使用更小的核，您将使用更多的滤波器，您将能够使用更多的激活函数，从而使您的CNN学习到更具区分性的映射函数。

8、为什么在图像分割中CNNs通常具有编码器-解码器结构?

编码器CNN基本上可以被认为是特征提取网络，而解码器使用该信息通过“解码”特征并放大到原始图像大小来预测图像分割区域。

9、为什么我们对图像使用卷积而不仅仅是FC层?

这个很有趣，因为公司通常不会问这个问题。正如你所料，我是从一家专注于计算机视觉的公司那里得到这个问题的。这个答案有两部分。首先，卷积保存、编码和实际使用来自图像的空间信息。如果我们只使用FC层，我们将没有相对的空间信息。其次，卷积神经网络(CNNs)具有部分内建的平移不变性，因为每个卷积核充当其自身的滤波器/特征检测器。，而且这样减少大量的参数，减轻过拟合。

10、什么是数据正则化/归一化(normalization)？为什么我们需要它?

我觉得这一点很重要。数据归一化是非常重要的预处理步骤，用于重新缩放输入的数值以适应特定的范围，从而确保在反向传播期间更好地收敛。一般来说采取的方法都是减去每个数据点的平均值并除以其标准偏差。如果我们不这样做，那么一些特征(那些具有高幅值的特征)将在cost函数中得到更大的加权(如果较高幅值的特征改变1 %，则该改变相当大，但是对于较小的特征，该改变相当小)。数据归一化使所有特征的权重相等。

11、解释降维(dimensionality reduction)，降维在哪里使用，降维的好处是什么?

降维是通过获得一组基本上是重要特征的主变量来减少所考虑的特征变量的过程。特征的重要性取决于特征变量对数据信息表示的贡献程度

数据集降维的好处可以是:

(1)减少所需的存储空间。

(2)加快计算速度(例如在机器学习算法中)，更少的维数意味着更少的计算，并且更少的维数可以允许使用不适合大量维数的算法。

(3)将数据的维数降低到2D或3D可以允许我们绘制和可视化它，可能观察模式，给我们提供直观感受。

(4)太多的特征或太复杂的模型可以导致过拟合。

1、1*1卷积作用。

答：1. 实现跨通道的交互和信息整合

2. 进行卷积核通道数的降维和升维

3、实现多个feature map的线性组合，实现通道个数的变换。

4、对特征图进行一个比例缩放。

2、CNN池化层有什么作用?

答：1、减小图像尺寸，数据降维。

2、缓解过拟合。

3、保持一定程度的旋转和平移不变性，MaxPooling能保证卷积神经网络在一定范围内平移特征能得到同样的激励，具有平移不变形。

3、卷积神经网络中空洞卷积的作用是什么?

答、空洞卷积也叫扩张卷积，在保持参数个数不变的情况下增大了卷积核的感受野，同时它可以保证输出的特征映射（feature map）的大小保持不变。一个扩张率为2的3×3卷积核，感受野与5×5的卷积核相同，但参数数量仅为9个。

4、深度学习中常用的损失函数?

答：交叉熵损失，平方差损失，绝对值损失，Hinge Loss。具体介绍：损失函数具体介绍

5、 Sigmoid激活函数为什么会出现梯度消失？Sigmoid函数导数的最大值出现在哪个值?

答：为什么会出现梯度消失，从两方面来看，首先先看本身函数，若输入值X过大，sigmoid函数导数为零，第二方面：sigmoid函数求导，导数最大是等于1/4，小于1，经过深的网络传递就会出现梯度消失的问题。

在x=0时导数最大。

6.faster rcnn是怎么样一个框架?

答：有一篇博客讲的挺清楚的。rcnn系列详解

7、faster rcnn,roi pooling具体是如何工作的？（如何把不同大小的框，pooling到同样的大小）

答、RoIPool首先将浮点数值的RoI量化成离散颗粒的特征图，然后将量化的RoI分成几个空间的小块（spatial bins），最后对每个小块进行max pooling操作生成最后的结果。

8、评价指标有哪些?

答、机器学习中评价指标： Accuracy（准确率）、 Precision（查准率或者精准率）、Recall（查全率或者召回率）。

目标检测的指标：识别精度，识别速度，定位精度。

a、目标检测中衡量识别精度的指标是mAP（mean average precision）。多个类别物体检测中，每一个类别都可以根据recall和precision绘制一条曲线，AP就是该曲线下的面积，mAP是多个类别AP的平均值。

b、 目标检测评价体系中衡量定位精度的指标是IoU,IoU就是算法预测的目标窗口和真实的目标窗口的交叠（两个窗口面积上的交集和并集比值）

9、深度学习训练时网络不收敛的原因有哪些？如何解决?

答：不收敛一般都是数据不干净，学习率设置不合理，网络等问题，详细知识根据博主的经验和看的的别人一些经验整理了下，

不收敛的原因与解决办法

10、如何应对图像光照变化大?

答：1、直方图均衡化

2、对比度拉伸，或者调节

3、若受光源影响，使得图片整体色彩往一方面移动，用白平衡算法进行修正，使其发黄、发蓝、发红的照片更加趋于自然光下的图像

4、若是过爆或者过暗，可是设计阈值函数，不用全局阈值，对特定区域进行特定阈值分割。

5、若是太暗，可以采用对数变化，对数图像增强是图像增强的一种常见方法，其公式为： S = c log(r+1)，对数使亮度比较低的像素转换成亮度比较高的，而亮度较高的像素则几乎没有变化，这样就使图片整体变亮。

6、采用拉普拉斯算子增强， filter2D(src,dst)

11、常用的分割方法有哪些?

答：1、基于阈值的分割方法：比较常用的阂值法有大律法、最小误差法

2、基于边缘的分割方法：常见的微分算子包括Robert算子、Prewitt算子、Sobel算子、Laplaeian算子、Canny算子等

3、基于区域的分割方法：主要包括种子区域生长法、区域分裂合并法和分水岭法等几种类型。

4、基于图论的分割方法：Graph Cut方法

5、深度学习：语义分割等

深度学习训练时网络不收敛的原因分析总结

很多同学会发现，为什么我训练网络的时候loss一直居高不下或者准确度时高时低，震荡趋势，一会到11，一会又0.1，不收敛。 又不知如何解决，博主总结了自己训练经验和看到的一些方法。

首先你要保证训练的次数够多，不要以为一百两百次就会一直loss下降或者准确率一直提高，会有一点震荡的。只要总体收敛就行。若训练次数够多（一般上千次，上万次，或者几十个epoch）没收敛，则试试下面方法：

1. 数据和标签

数据分类标注是否准确？数据是否干净？数据库太小一般不会带来不收敛的问题，只要你一直在train总会收敛（rp问题跑飞了不算）。反而不收敛一般是由于样本的信息量太大导致网络不足以fit住整个样本空间。样本少只可能带来过拟合的问题

2. 学习率设定不合理

在自己训练新网络时，可以从0.1开始尝试，如果loss不下降的意思，那就降低，除以10，用0.01尝试，一般来说0.01会收敛，不行的话就用0.001. 学习率设置过大，很容易震荡。不过刚刚开始不建议把学习率设置过小，尤其是在训练的开始阶段。在开始阶段我们不能把学习率设置的太低否则loss不会收敛。我的做法是逐渐尝试，从0.1,0.08,0.06,0.05 .....逐渐减小直到正常为止，

有的时候候学习率太低走不出低估，把冲量提高也是一种方法，适当提高mini-batch值，使其波动不大。，

3.网络设定不合理

如果做很复杂的分类任务，却只用了很浅的网络，可能会导致训练难以收敛，换网络换网络换网络，重要的事情说三遍，或者也可以尝试加深当前网络。

4.数据集label的设置

检查lable是否有错，有的时候图像类别的label设置成1，2，3正确设置应该为0,1,2。

5、改变图片大小

博主看到一篇文章，说改变图片大小可以解决收敛问题，具体博主没试过，只看到有这个方法，具体文章链接：https://blog.csdn.net/Fighting_Dreamer/article/details/71498256

感兴趣的可以去看看。

6、数据归一化

神经网络中对数据进行归一化是不可忽略的步骤，网络能不能正常工作，还得看你有没有做归一化，一般来讲，归一化就是减去数据平均值除以标准差，通常是针对每个输入和输出特征进行归一化

NN最成功的应用是在CV，那为什么NLP和Speech的很多问题也可以用CNN解出来？为什么AlphaGo里也用了CNN？这几个不相关的问题的相似性在哪里？CNN通

过什么手段抓住了这个共性？
CNN是用于挖掘数据之间的空间相关性（相对于RNN用于挖掘时序相关性），所以不管是NLP，Speech还是AlphaGo，只要数据之间存在空间相关性，就可以使用CNN。而CNN最成功的应用在CV是因为在图像数据中的空间相关性最明显，像素之间存在两个维度上的空间相关性。
回答CNN通过什么手段抓住此共性这个问题，实质上是回答卷积和池化的特点。主要手段有局部连接(local connectivity)、权值共享(parameter sharing)和池化(pooling)（跟原文回答有点出入，觉得多层次是深度学习的特点而不是CNN的特点）。局部连接和权值共享有效地减少了网络的权值参数，抑制过拟合，同时保持了原始数据之间的空间联系。池化是下采样的一种方式，只取数值最大的特征，减少参数并且在空间上对原图物体保持平移不变性。

###什么样的资料集不适合用深度学习?

数据集太小。数据集太小时，用深度学习提取出来的特征非常容易过拟合，没有泛化能力，因而不及传统的机器学习方法。其本质是样本空间太稀疏，不能反映总体空间的分布，因而用深度学习得到的模型会过拟合。
数据之间没有局部相关性。深度学习对于结构化的数据提取特征有非常好的效果，但对于没有结构化的数据，即数据之间不存在局部相关性，如可能存在统计关系等，此时用深度学习就不是很合适。

R-CNN --> FAST-RCNN --> FASTER-RCNN

R-CNN:
(1)输入测试图像；
(2)利用selective search 算法在图像中从上到下提取2000个左右的Region Proposal；
(3)将每个Region Proposal缩放(warp)成227*227的大小并输入到CNN，将CNN的fc7层的输出作为特征；
(4)将每个Region Proposal提取的CNN特征输入到SVM进行分类；
(5)对于SVM分好类的Region Proposal做边框回归，用Bounding box回归值校正原来的建议窗口，生成预测窗口坐标.
缺陷:
(1) 训练分为多个阶段，步骤繁琐：微调网络+训练SVM+训练边框回归器；
(2) 训练耗时，占用磁盘空间大；5000张图像产生几百G的特征文件；
(3) 速度慢：使用GPU，VGG16模型处理一张图像需要47s；

(4)　测试速度慢：每个候选区域需要运行整个前向CNN计算；
(5)　SVM和回归是事后操作，在SVM和回归过程中CNN特征没有被学习更新.

FAST-RCNN:
(1)输入测试图像；
(2)利用selective search 算法在图像中从上到下提取2000个左右的建议窗口(Region Proposal)；
(3)将整张图片输入CNN，进行特征提取；
(4)把建议窗口映射到CNN的最后一层卷积feature map上；
(5)通过RoI pooling层使每个建议窗口生成固定尺寸的feature map；
(6)利用Softmax Loss(探测分类概率) 和Smooth L1 Loss(探测边框回归)对分类概率和边框回归(Bounding box regression)联合训练.

相比R-CNN，主要两处不同:
(1)最后一层卷积层后加了一个ROI pooling layer；
(2)损失函数使用了多任务损失函数(multi-task loss)，将边框回归直接加入到CNN网络中训练
改进:
(1)　测试时速度慢：R-CNN把一张图像分解成大量的建议框，每个建议框拉伸形成的图像都会单独通过CNN提取特征.实际上这些建议框之间大量重叠，特征值之间完全可以共享，造成了运算能力的浪费.
FAST-RCNN将整张图像归一化后直接送入CNN，在最后的卷积层输出的feature map上，加入建议框信息，使得在此之前的CNN运算得以共享.
(2)　训练时速度慢：R-CNN在训练时，是在采用SVM分类之前，把通过CNN提取的特征存储在硬盘上.这种方法造成了训练性能低下，因为在硬盘上大量的读写数据会造成训练速度缓慢.
FAST-RCNN在训练时，只需要将一张图像送入网络，每张图像一次性地提取CNN特征和建议区域，训练数据在GPU内存里直接进Loss层，这样候选区域的前几层特征不需要再重复计算且不再需要把大量数据存储在硬盘上.
(3)　训练所需空间大：R-CNN中独立的SVM分类器和回归器需要大量特征作为训练样本，需要大量的硬盘空间.FAST-RCNN把类别判断和位置回归统一用深度网络实现，不再需要额外存储.

FASTER -RCNN:

(1)输入测试图像；
(2)将整张图片输入CNN，进行特征提取；
(3)用RPN生成建议窗口(proposals)，每张图片生成300个建议窗口；
(4)把建议窗口映射到CNN的最后一层卷积feature map上；
(5)通过RoI pooling层使每个RoI生成固定尺寸的feature map；
(6)利用Softmax Loss(探测分类概率) 和Smooth L1 Loss(探测边框回归)对分类概率和边框回归(Bounding box regression)联合训练.

相比FASTER-RCNN，主要两处不同:
(1)使用RPN(Region Proposal Network)代替原来的Selective Search方法产生建议窗口；
(2)产生建议窗口的CNN和目标检测的CNN共享

改进:
(1)　如何高效快速产生建议框?
FASTER-RCNN创造性地采用卷积网络自行产生建议框，并且和目标检测网络共享卷积网络，使得建议框数目从原有的约2000个减少为300个，且建议框的质量也有本质的提高.

- Linear SVM和LR都是线性分类器
- Linear SVM不直接依赖数据分布，分类平面不受一类点影响；LR则受所有数据点的影响，如果数据不同类别strongly unbalance一般需要先对数据做balancing。
- Linear SVM依赖数据表达的距离测度，所以需要对数据先做normalization；LR不受其影响
- Linear SVM依赖penalty的系数，实验中需要做validation
- Linear SVM和LR的performance都会收到outlier的影响，其敏感程度而言，谁更好很难下明确结论。

其实这两个分类器还是很类似的，都是在最大化两类点之间的距离，但是 LR 把所有点都纳入模型考量的范围了，而 SVM 则只看 support vectors, 也就是离分类平面最近的点。所以说 SVM 的优点就在于，通

过忽略已经分类正确的点，最后训练出来的模型更加稳健，对 outlier 不敏感。

具体到 loss function 上来看， LR 用的是 log-loss, SVM 用的是 hinge-loss, 两者的相似之处在于 loss 在错误分类的时候都很大，但是对于正确分类的点，hinge-loss 就不管了，而 log-loss 还要考虑进去。此外因为 log-loss 在 mis-classified 的点上是指数级增长的，而 hinge-loss 是线性增长，所以 LR 在偶尔出现 mis-label 的情况下的表现会比较糟糕。

此外还有一点就是用 SVM 来预测概率意义不大，人家模型本身就不是基于概率的。LR 则是基于 log-likelihood ratio，方便给出概率，并且向 multi-class 的扩展更加直接。（SVM 做 multi-class 也不是不可以，但是 objective function 很乱，实践中一般直接用 one-vs-all）

另外 regularization 在这里没有区别，L1/L2 两个都能用，效果也差不多。Class imbalance 的话 SVM 一般用 weight 解决，LR 因为可以预测概率，所以也可以直接对最后的结果进行调整，取不同的阈值来达到理想的效果。

实践中 LR 的速度明显更快，维度小的时候 bias 小也不容易 overfit. 相反 Kernel SVM 在大规模数据集的情况下基本不实用，但是如果数据集本身比较小而且维度高的的话一般 SVM 表现更好。

CNN不适用的场景

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。

神经网络效果不好时，可以思考的角度

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）
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）
是否选择了合适的激活函数？（各个激活函数的来源和差异）
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）
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）
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