链滴

计算机视觉面试 31 题：CV 面试考点，精准详尽解析（1-10）

作者：julyedu

原文链接：https://ld246.com/article/1622022808322

来源网站：链滴

<h2 id="添加微信-julyedufu77-回复---1-1--领取最新升级版-名企-AI-面试-100-题-电子书--">添微信：julyedufu77，回复 “ 1 1” ，领取最新升级版《名企 AI 面试 100 题》电子书！！ </h2>
<h2 id="1-基于深度学习的目标检测技术演进-R-CNN-Fast-R-CNN-Faster-R-CNN-YOLO-SSD">、基于深度学习的目标检测技术演进：R-CNN、Fast R-CNN、Faster R-CNN、YOLO、SSD</h2>
一、目标检测常见算法
object detection，就是在给定的图片中精确找到物体所在位置，并标注出物体的类别。所以，oject detection 要解决的问题就是物体在哪里以及是什么的整个流程问题。
然而，这个问题可不是那么容易解决的，物体的尺寸变化范围很大，摆放物体的角度，姿态不定而且可以出现在图片的任何地方，更何况物体还可以是多个类别。
目前学术和工业界出现的目标检测算法分成 3 类：

传统的目标检测算法：Cascade + HOG/DPM + Haar/SVM 以及上述方法的诸多改进、优化；
候选区域/框 + 深度学习分类：通过提取候选区域，并对相应区域进行以深度学习方法为主的分的方案，如：

R-CNN（Selective Search + CNN + SVM）
SPP-net（ROI Pooling）
Fast R-CNN（Selective Search + CNN + ROI）
Faster R-CNN（RPN + CNN + ROI）
R-FCN
等系列方法；
<ol start="3">
基于深度学习的回归方法：YOLO/SSD/DenseBox 等方法；以及最近出现的结合 RNN 算法的 RC detection；结合 DPM 的 Deformable CNN 等

<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/ba496380a452431a50f7d1f05588cb3.jpeg?imageView2/2/interlace/1/format/jpg">
传统目标检测流程：
1）区域选择（穷举策略：采用滑动窗口，且设置不同的大小，不同的长宽比对图像进行遍历，间复杂度高）
2）特征提取（SIFT、HOG 等；形态多样性、光照变化多样性、背景多样性使得特征鲁棒性差）
3）分类器分类（主要有 SVM、Adaboost 等）
<h2 id="2-请简单解释下目标检测中的这个IOU评价函数-intersection-over-union-">2、请简单解下目标检测中的这个 IOU 评价函数（intersection-over-union）</h2>
解析一
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口和原来标窗口的交叠率。具体我们可以简单的理解为： 即检测结果(DetectionResult)与 Ground Truth 的交比上它们的并集，即为检测的准确率 IoU :
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/33f2d547ac3d4feb45169e793d63d7f.png?imageView2/2/interlace/1/format/jpg">
如下图所示：GT = GroundTruth; DR = DetectionResult;
黄色边框框起来的是：
GT□DR
绿色框框起来的是：
GT□DR
应该够详细了，上幅图直观些。当然最理想的情况就是 DR 与 GT 完全重合，即
IoU=1
下面附上图例说明
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/722e149166d44

c88b45ceeaca51f4a4.jpeg?imageView2/2/interlace/1/format/jpg">
原图则如下
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/0181c25ff01c427815dcb985f5bbaea.jpeg?imageView2/2/interlace/1/format/jpg">
<h2 id="3-KNN与K-means区别-">3、KNN 与 K-means 区别？</h2>
Wikipedia 上的 KNN 词条 中有一个比较经典的图如下：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/1a819536d9534cb981be24508fde82a.jpeg?imageView2/2/interlace/1/format/jpg">
KNN 算法流程：
从上图中我们可以看到，图中的数据集是良好的数据，即都打好了 label，一类是蓝色的正方形一类是红色的三角形，那个绿色的圆形是我们待分类的数据。
如果 K=3，那么离绿色点最近的有 2 个红色三角形和 1 个蓝色的正方形，这 3 个点投票，于是色的这个待分类点属于红色的三角形。
如果 K=5，那么离绿色点最近的有 2 个红色三角形和 3 个蓝色的正方形，这 5 个点投票，于是色的这个待分类点属于蓝色的正方形。
我们可以看到，KNN 本质是基于一种数据统计的方法！其实很多机器学习算法也是基于数据统的。
KNN 是一种 memory-based learning，也叫 instance-based learning，属于 lazy learning 即它没有明显的前期训练过程，而是程序开始运行时，把数据集加载到内存后，不需要进行训练，就以开始分类了。具体是每次来一个未知的样本点，就在附近找 K 个最近的点进行投票。
K-Means 介绍：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/68e39109e824448925dee59e7246573.jpeg?imageView2/2/interlace/1/format/jpg">
如图所示，数据样本用圆点表示，每个簇的中心点用叉叉表示：
(a)刚开始时是原始数据，杂乱无章，没有 label，看起来都一样，都是绿色的。
(b)假设数据集可以分为两类，令 K=2，随机在坐标上选两个点，作为两个类的中心点。
(c-f)演示了聚类的两种迭代。先划分，把每个数据样本划分到最近的中心点那一簇；划分完后，新每个簇的中心，即把该簇的所有数据点的坐标加起来去平均值。这样不断进行”划分—更新—划分更新”，直到每个簇的中心不在移动为止。(图文来自 Andrew ng 的机器学习公开课)。
现在，汇总一下 KNN 和 K-Means 的区别
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/ebee250126424268485a063ec9c3433.jpeg?imageView2/2/interlace/1/format/jpg">
最后，推荐关于 K-Means 的一篇博文：漫谈 Clustering (1)_ k-means pluskid -漫谈 Clusterin (1): k-means
<h2 id="4-K-means选择初始点的方法有哪些-优缺点是什么--列出两种以上-">4、K-means 选择始点的方法有哪些,优缺点是什么?(列出两种以上)</h2>
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/68e39109e824448925dee59e7246573.jpeg?imageView2/2/interlace/1/format/jpg">
KMeans 是数据挖掘十大算法之一，在数据挖掘实践中，我们也常常将 KMeans 运用于各种场，因为它原理简单、易于实现、适合多种数据挖掘情景。
如上图所示，数据样本用圆点表示，每个簇的中心点用叉叉表示：
(a)刚开始时是原始数据，杂乱无章，没有 label，看起来都一样，都是绿色的。
(b)假设数据集可以分为两类，令 K=2，随机在坐标上选两个点，作为两个类的中心点。
(c-f)演示了聚类的两种迭代。先划分，把每个数据样本划分到最近的中心点那一簇；划分完后，新每个簇的中心，即把该簇的所有数据点的坐标加起来去平均值。这样不断进行”划分—更新—划分更新”，直到每个簇的中心不在移动为止。(图文来自 Andrew ng 的机器学习公开课)。
初始中心点的选择：
初始中心点的选择最简单的做法是随机从样本中选 K 个作为中心点，但由于中心点的选择会影响 Means 的聚类效果，因此我们可以采取以下三种方式优化中心点的选取：

1.多次选取中心点进行多次试验，并用损失函数来评估效果，选择最优的一组；
2.选取距离尽量远的 K 个样本点作为中心点：随机选取第一个样本 C1 作为第一个中心点，遍历有样本选取离 C1 最远的样本 C2 为第二个中心点，以此类推，选出 K 个初始中心点
3.特别地，对于像文本这样的高维稀疏向量，我们可以选取 K 个两两正交的特征向量作为初始化心点。
<h2 id="5-简述线性分类器的原理-要求对权重矩阵进行剖析-">5、简述线性分类器的原理(要求对权矩阵进行剖析)</h2>
假设有一个包含很多图像的训练集，每个图像都有一个对应的分类标签。这里并且。这就是说，们有 N 个图像样例，每个图像的维度是 D，共有 K 种不同的分类。 比如在 CIFAR-10 中，我们有一个 N=50000 的训练集，每个图像有 D=32x32x3=3072 个像素，而 K=10，这是因为图片被分为 10 不同的类别（狗，猫，汽车等）。我们现在定义评分函数为：，该函数是原始图像像素到分类分值的射。
线性分类器
定义线性映射：

上式中，假设每个图像数据都被拉长为一个长度为 D 的列向量，大小为[D x 1]。其中矩阵 W 大为[K x D]，列向量 b 大小为[K x 1]，二者均称为该函数的参数（parameters）。以 CIFAR-10 为例就包含了第 i 个图像的所有像素信息，这些信息被拉成为一个[3072 x 1]的列向量，矩阵 W 大小为[10 3072]，列向量 b 的大小为[10x1]。因此将 3072 个数字(原始像素数值)输入函数，函数输出 10 个数(10 个类别得分), 参数 W 被称为权重（weights），b 被称为偏置向量（bias vector），因为它影响出数值，但是并不和原始数据产生关联。在实际情况中，人们常常混用权重和参数这两个术语。
其中权重 W 的每一行都代表一个分类器，10 行分类表示 10 个类别的分类器，最终输出 10 个别的得分情况。理解线性分类器 线性分类器计算图像中 3 个颜色通道中所有像素的值与权重的矩阵，从而得到分类分值。根据我们对权重设置的值，对于图像中的某些位置的某些颜色，函数表现出喜或者厌恶（根据每个权重的符号而定）。比如，可以想象“ship”类别就是被大量的蓝色所包围（对的就是水）。那么“ship”分类器在蓝色通道上的权重就有很多的正权重（它们的出现提高了“ship类别的分值），而在绿色和红色通道上的权重为负的就比较多（它们的出现降低了“ship”类别的分）。

上图是 一个将图像映射到分类分值的例子；为了便于可视化，假设图像只有 4 个像素（都是黑像素，这里不考虑 RGB 通道），有 3 个分类（红色代表猫，绿色代表狗，蓝色代表船，注意，这里红、绿和蓝 3 种颜色仅代表分类，和 RGB 通道没有关系）。首先将图像像素拉伸为一个列向量，与 进行矩阵乘，然后得到各个分类的分值。需要注意的是，这个 W 一点也不好：猫分类的分值非常低从上图来看，算法倒是觉得这个图像是一只狗，因为上图中“dog”的类别分值最高。
将线性分类器看做模板匹配：
关于权重 W 的另一个解释是它的每一行对应着一个分类的模板。一张图像对应不同分类的得分是通过使用内积来比较图像和模板，然后找到和哪个模板最相似。从这个角度来看，线性分类器就是利用学习到的模板，针对图像做模板匹配。
<h2 id="6-请简述下log对数-Hinge-Loss-折页--Cross-Entropy-Loss-交叉熵-这三个损失函数">6 请简述下 log 对数、Hinge Loss(折页)、Cross-Entropy Loss(交叉熵)这三个损失函数</h2>
log 对数损失函数
log 对数损失函数的标准形式如下：

特点：
(1) log 对数损失函数能非常好的表征概率分布，在很多场景尤其是多分类，如果需要知道结果于每个类别的置信度，那它非常适合。
(2)健壮性不强，相比于 hinge loss 对噪声更敏感。
(3)逻辑回归的损失函数就是 log 对数损失函数。

Hinge 损失函数
Hinge 损失函数标准形式如下：

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/62da3830ff6b43b930ade0e092a868d.png?imageView2/2/interlace/1/format/jpg">
特点：
(1)hinge 损失函数表示如果被分类正确，损失为 0，否则损失就为

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/dce759e025924fab3a1f3b0b3d66160.png?imageView2/2/interlace/1/format/jpg">
。SVM 就是使用这个损失函数。
(2)一般的

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/8482b8ef9b134209504207ffc8ba626.png?imageView2/2/interlace/1/format/jpg">
是预测值，在-1 到 1 之间，

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/260a4b093e144618a9f864a7c637c8b.png?imageView2/2/interlace/1/format/jpg">
是目标值(-1 或 1)。其含义是，

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/8482b8ef9b134209504207ffc8ba626.png?imageView2/2/interlace/1/format/jpg">
的值在-1 和 +1 之间就可以了，并不鼓励

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/836eead12f2841492959af98817f092.png?imageView2/2/interlace/1/format/jpg">
，即并不鼓励分类器过度自信，让某个正确分类的样本距离分割线超过 1 并不会有任何奖励，从使分类器可以更专注于整体的误差。
(3) 健壮性相对较高，对异常点、噪声不敏感，但它没太好的概率解释。
交叉熵损失函数 (Cross-entropy loss function)
交叉熵损失函数的标准形式如下:

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/2ab9cb1953a4495bccb70590e6ded97.png?imageView2/2/interlace/1/format/jpg">
注意公式中

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/a1bb6941690c45eae0c3c979822e953.png?imageView2/2/interlace/1/format/jpg">
表示样本，

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/260a4b093e144618a9f864a7c637c8b.png?imageView2/2/interlace/1/format/jpg">
表示实际的标签，

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/960df645229b490b71c17a4e1faf23e.png?imageView2/2/interlace/1/format/jpg">
表示预测的输出，

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/4f66dde7369a4edb1aebc074c62171c.png?imageView2/2/interlace/1/format/jpg">
表示样本总数量。
特点：
(1)本质上也是一种对数似然函数，可用于二分类和多分类任务中。二分类问题中的 loss 函数（输入

据是 softmax 或者 sigmoid 函数的输出）：多分类问题中的 loss 函数（输入数据是 softmax 或者 sigmoid 函数的输出）：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/46e3917ee797477b266ef41a0754737.png?imageView2/2/interlace/1/format/jpg">
(2)当使用 sigmoid 作为激活函数的时候，常用交叉熵损失函数而不用均方误差损失函数，因为可以完美解决平方损失函数权重更新过慢的问题，具有“误差大的时候，权重更新快；误差小的时候权重更新慢”的良好性质。
相关高频问题
1 交叉熵函数与最大似然函数的联系和区别？
区别：交叉熵函数使用来描述模型预测值和真实值的差距大小，越大代表越不相近；似然函数的本质是衡量在某个参数下，整体的估计和真实的情况一样的概率，越大代表越相近。
联系：交叉熵函数可以由最大似然函数在伯努利分布的条件下推导出来，或者说最小化交叉熵函数的质就是对数似然函数的最大化。怎么推导的呢？我们具体来看一下。
设一个随机变量
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/8e16a0656c1c4ac886a7864c21f6751.png?imageView2/2/interlace/1/format/jpg">
满足伯努利分布，
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/0d92b8cd2ea247586751646a058de7f.png?imageView2/2/interlace/1/format/jpg">
则
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/6a940ea30ff6407ab522144cf78d80c.png?imageView2/2/interlace/1/format/jpg">
的概率密度函数为：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/132cf03d95894f7b5dd3f11a3b7762e.png?imageView2/2/interlace/1/format/jpg">
因为我们只有一组采样数据
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/02dd91c940264d89e4f4c5df19f6f26.png?imageView2/2/interlace/1/format/jpg">
，我们可以统计得到
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/8e16a0656c1c4ac886a7864c21f6751.png?imageView2/2/interlace/1/format/jpg">
和
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/aa251b0958d545e9aa1b468d9574b81.png?imageView2/2/interlace/1/format/jpg">
的值，
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/7935352dcd564d988f61aca2a63fbfa.png?imageView2/2/interlace/1/format/jpg">
值。
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/db6f07ba2e4141ca1229690b1deead1.jpeg?imageView2/2/interlace/1/format/jpg">
可以看到上式和交叉熵函数的形式几乎相同，极大似然估计就是要求这个式子的最大值。而由于面函数的值总是小于 0，一般像神经网络等对于损失函数会用最小化的方法进行优化，所以一般会在面加一个负号，得到交叉熵函数（或交叉熵损失函数）：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV

试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/84217ff6da18441bda38ea61a948bbc.png?imageView2/2/interlace/1/format/jpg">
这个式子揭示了交叉熵函数与极大似然估计的联系，最小化交叉熵函数的本质就是对数似然函数最大化。
现在我们可以用求导得到极大值点的方法来求其极大似然估计，首先将对数似然函数对
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/1746c6dac8b445cb653e144bfee996b.png?imageView2/2/interlace/1/format/jpg">
进行求导，并令导数为 0，得到
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/140f5e2daa874b694d2f5d851f40403.jpeg?imageView2/2/interlace/1/format/jpg">
<ol start="2">
在用 sigmoid 作为激活函数的时候，为什么要用交叉熵损失函数，而不用均方误差损失函数？<b>
其实这个问题求个导，分析一下两个误差函数的参数更新过程就会发现原因了。
对于均方误差损失函数，常常定义为：

<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/b2e7954d476446fb3ef08b291cdffd2.jpeg?imageView2/2/interlace/1/format/jpg">
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/02c873f454fb46897f5dd649816f384.jpeg?imageView2/2/interlace/1/format/jpg">
那么为什么交叉熵损失函数就会比较好了呢？同样的对于交叉熵损失函数，计算一下参数更新的度公式就会发现原因。交叉熵损失函数一般定义为：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/2e75724543fc421809737c1448cb42b.jpeg?imageView2/2/interlace/1/format/jpg">
所以有：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/88ee2da404734c8b49ac69d85d96549.jpeg?imageView2/2/interlace/1/format/jpg">
候，权重更新慢。这是一个很好的性质。所以当使用 sigmoid 作为激活函数的时候，常用交叉损失函数而不用均方误差损失函数。
<h2 id="7-简述正则化与奥卡姆剃刀原则">7、简述正则化与奥卡姆剃刀原则</h2>
正则化 Regularization
正则化最主要的功能是防止网络过拟合，主要有 L1 正则和 L2 正则：
L2 正则化（岭回归）可能是最常用的正则化方法了，以通过惩罚目标函数中所有参数的平方来止过拟合。即对于网络中的每个权重，在目标函数中增加一个项，其中是正则化惩罚系数。加上后该子关于梯度就是而不是了。L2 正则化可以直观理解为它对于大数值的权重向量进行严厉惩罚，倾向更加分散的权重向量。由于输入和权重之间的乘法操作，使网络更倾向于使用所有输入特征，而不是重依赖输入特征中某些小部分特征。最后需要注意在梯度下降和参数更新的时候，使用 L2 正则化意着所有的权重都以 w += -lambda * W 向着 0 线性下降。
L1 正则化（套索回归）是另一个相对常用的正则化方法。对于每个我们都向目标函数增加一个。L1 和 L2 正则化也可以进行组合：，这也被称作弹性网络回归。L1 正则化有一个有趣的性质，它让权重向量在最优化的过程中变得稀疏（即非常接近 0）。也就是说，使用 L1 正则化的神经元最后用的是它们最重要的输入数据的稀疏子集，同时对于噪音输入则几乎是不变的了。相较 L1 正则化，L2 正则化中的权重向量大多是分散的小数字。在实践中，如果不是特别关注某些明确的特征选择，一般来 L2 正则化都会比 L1 正则化效果好。
最大范式约束（Max norm constraints）是另一种形式的正则化，给每个神经元中权重向量的级设定上限，并使用投影梯度下降来确保这一约束。在实践中，与之对应的是参数更新方式不变，然要求神经元中的权重向量必须满足这一条件，一般值为 3 或者 4。有研究者发文称在使用这种正则化法时效果更好。这种正则化还有一个良好的性质，即使在学习率设置过高的时候，网络中也不会出现

值“爆炸”，这是因为它的参数更新始终是被限制着的。
随机失活（Dropout）是一个简单又极其有效的正则化方法。该方法由 Srivastava 在论文 Dropout: A Simple Way to Prevent Neural Networks from Overfitting 中提出的，与 L1 正则化，L2 正化和最大范式约束等方法互为补充。在训练的时候，随机失活的实现方法是让神经元以超参数的概率激活或者被设置为 0。

图片展示了其核心思路：在训练过程中，随机失活可以被认为是对完整的神经网络抽样出一些子，每次基于输入数据只更新子网络的参数（然而，数量巨大的子网络们并不是相互独立的，网络之间数共享）。在推理阶段不使用随机失活，可以理解为是对数量巨大的子网络们做了模型集成（model nsemble），以此来计算出一个平均的预测。
奥卡姆剃刀原理：
这个原理称为"如无必要，勿增实体", 即"简单有效原理"。
奥卡姆剃刀原理是指，在科学研究任务中，应该优先使用较为简单的公式或者原理，而不是复杂。应用到机器学习任务中，可以通过减小模型的复杂度来降低过拟合的风险，即模型在能够较好拟合练集(经验风险)的前提下，尽量减小模型的复杂度(结构风险)。
<h2 id="8-图像尺寸为-77--卷积窗口大小为33--步长为3--是否能输出图像-如果能-输出图像大小为少-如果不能-说明原因-">8、图像尺寸为 77, 卷积窗口大小为 33, 步长为 3, 是否能输图像?如果能,输出图像大小为多少?如果不能,说明原因?</h2>
本题参考来源： Michael Yuan：CNN 中卷积层的计算细节
CS231n Convolutional Neural Networks for Visual Recognition
答案：根据计算公式（W-F+2P）/S + 1 P=2 时输出为 3X3

卷积层尺寸的计算原理
输入矩阵格式：四个维度，依次为：样本数(batch size)、图像高度、图像宽度、图像通道数</p

输出矩阵格式：与输入矩阵的维度顺序和含义相同，但是后三个维度（图像高度、图像宽度、图通道数）的尺寸发生变化。
权重矩阵（卷积核）格式：同样是四个维度，但维度的含义与上面两者都不同，为：卷积核高度卷积核宽度、深度、卷积核个数(输出通道数); 高度和宽度即 kernel size, 深度无需指定 默认为上一层征图的通道数，卷积核个数决定卷积后输出特征图的通道个数。
输入矩阵、权重矩阵、输出矩阵这三者之间的相互决定关系 :
卷积核的输入通道数（in depth）由输入矩阵的通道数所决定。
输出矩阵的通道数（out depth）由卷积核的输出通道数所决定 。
输出矩阵的高度和宽度（height, width）这两个维度的尺寸由输入矩阵、卷积核、stride 大小同决定, 计算公式如下。

标准卷积计算举例

以 AlexNet 模型的第一个卷积层为例 计算如上图所示，- 输入图片的尺寸统一为 227 x 227 x 3（高度 x 宽度 x 颜色通道数），- 共具有 96 个卷积核，- 每个卷积核的尺寸都是 11 x 11 x 3，3 为一层的通道数 可以不指定- 已知 stride = 4， padding = 0，- 假设 batch_size = 256，- 则输出矩的高度/宽度为 (227 - 11) / 4 + 1 = 55 ； 如果遇到无法整除的情况，则向上取整- 因为有 96 个卷积 所以输出通道数为 96
参数计算： 上述过程的参数量计算方式：11113961111 是卷积核大小，3 是输入通道个数，96 是卷积核个数。卷积采用参数共享机制，每个通道上卷积参数完全一样不同卷积核以及不同通道之间参数不同；一个卷积核对应输出特征图的一个通道，具体计算过程可以

考下图：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/8e87bf683d7845397a1f486c68510b1.jpeg?imageView2/2/interlace/1/format/jpg">
该图来源于 cs231n 课程，参数大小和上图有差别，但计算方式一样。蓝色表示原图共 3 个通道红色表示卷积核共 2 个， 绿色表示输出；卷积核的深度 d 和输入维度一致且 d[0],d[1],d[2]的参数是一样的，每个卷积核对应一个输出特征图且之间的参数也不一样的。
因此参数量为：卷积核宽度 * 卷积核高度 * 输入通道数 * 卷积核个数。
<h2 id="9-如果最后一个卷积层和第一个全连接层参数量太大怎么办-">9、如果最后一个卷积层和一个全连接层参数量太大怎么办?</h2>
本题解析参考来源：为什么为什么全局平均池化层有用，为什么可以替代全连接层？
答案：用全局池化层替换全连接层
在 CNN 卷积神经网络发展的初期，卷积层通过池化层(最大池化/平均池化)后总是要一个或多个连接层，最后经过 SoftMax 层进行分类。其中 FC 全连接层的参数超多，使得模型本身变得非常臃肿在 Network in Network 论文中提到了使用全局平局池化层代替全连接层的思路，以下是摘录的一部资料：
GAP(global average poolilng), 既然全连接网络可以使 feature map 的维度减少，进而输入到oftmax，但是又会造成过拟合，是不是可以用 pooling 来代替全连接。
答案是肯定的，Network in Network 工作使用 GAP 来取代了最后的全连接层，直接实现了降，更重要的是极大地减少了网络的参数(CNN 网络中占比最大的参数其实后面的全连接层)。
全连接层先将卷积层展开成列向量，之后再针对每个 feature map 进行分类，而 GAP 的思路就将上述两个过程合二为一，如下图说是
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/e20d1755341a4d7afa7fb072bd8a854.jpeg?imageView2/2/interlace/1/format/jpg">
通过两者合二为一的过程我们可以探索到 GAP 的真正意义是:对整个网路在结构上做正则化防止拟合。其直接剔除了全连接层中黑箱的特征，直接赋予了每个 channel 实际的类别意义。
事实上并不是单纯的全局平均池化层替代的全连接层，而是这个全局平均池化层以及之前的若干卷积层共同替代了全连接层。举个例子，从一个 n 维的全连接层降维到 m 维的全连接层，在计算时操作是，右乘以一个 m×n 的矩阵 W。而换一个角度来看，是否可以理解为一层 n×1×1 的特征图降到 m×1×1 的特征图，进行了一次 m×n×1×1 的卷积呢：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/a30880ce80b2451b22a968da201140b.jpeg?imageView2/2/interlace/1/format/jpg">
全连接层等价于 1×1 的卷积
而使用全卷积网络，则是使得特征图的边长知道最后一层全局池化层才被降到 1。即，该放全连层的位置对特征图降维，从 n×a×a 的特征图降维到 m×b×b（b != 1）的特征图，使用的是一次 m××c×c 的卷积：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/1409f6f5d6074c1b90702c4f6122c80.jpeg?imageView2/2/interlace/1/format/jpg">
全卷积网络中，替代全连接层降维功能的卷积
而全卷积网络，只需要保证最后一层的特征图，维度是输出向量的维度即可。比如手写数字识别务，我们可以让最后一层的维度是 10。尽管其长和宽还不确定，但只要经过一次全局平均池化，即转化为 10-D 的向量输出：
<img src="https://ld246.com/images/img-loading.svg" alt="计算机视觉面试 31 题：CV 试考点，精准详尽解析（1-10）" data-src="https://b3logfile.com/file/2021/05/dcdb9b530c714f488942502f7af932e.jpeg?imageView2/2/interlace/1/format/jpg">
全局平均池化
全连接层进行了信息的转化，卷积层进行信息的提取，而池化更多的是信息的压缩。但单靠全局均池化确实不能替代卷积层，还需要配合若干层卷积层，对信息进行提取和转化，最终整形到期望输的维度，再通过全局平均池化完成输出。
<h2 id="10-为什么说神经网络是端到端的网络-">10、为什么说神经网络是端到端的网络?</h2>
本题参考来源：什么是 end-to-end 神经网络？_深藏功与名-CSDN 博客

端到端指的是输入是原始数据，输出是最终目标; 神经网络输入的原始数据, 输出的是我们希望的终结果，所以是端到端网络。
以前的模型输入端不是直接的原始数据，而是在原始数据中提取的特征，这一点在图像问题上尤突出，因为图像像素数太多，数据维度高，会产生维度灾难，所以原来一个思路是手工提取图像的一关键特征，这实际就是就一个降维的过程。
那么问题来了，特征怎么提？
特征提取的好坏异常关键，甚至比学习算法还重要，举个例子，对一系列人的数据分类，分类结是性别，如果你提取的特征是头发的颜色，无论分类算法如何，分类效果都不会好，如果你提取的特是头发的长短，这个特征就会好很多，但是还是会有错误，如果你提取了一个超强特征，比如染色体数据，那你的分类基本就不会错了。
这就意味着，特征需要足够的经验去设计，这在数据量越来越大的情况下也越来越困难。于是就现了端到端网络，特征可以自己去学习，所以特征提取这一步也就融入到算法当中，不需要人来干预。发展过程如下图所示：

经典机器学习方式是以人类的先验知识将 RAW 原始数据预处理成 Feature 特征，然后对 Featur 进行分类。分类结果好坏十分取决于 Feature 的好坏。所以过去的机器学习专家将大部分时间花费设计 Feature 上。那时的机器学习有个更合适的名字叫特征工程(Feature Engineering)。
后来人们发现，利用神经网络，让网络自己学习如何提取 Feature 效果更佳。于是兴起了表征学(Representation Learning), 这种方式对数据的拟合更加灵活。
网络进一步加深，多层次概念的 Representation Learning 将识别率达到了另一个新高度。于提出了深度学习(Deep Learning)的概念，指多层次的特征提取器与识别器统一训练和预测的网络。
End to End 的好处：通过缩减人工预处理和后续处理，尽可能使模型从原始输入到最终输出，模型更多可以根据数据自动调节的空间，增加模型的整体契合度。
从目标检测角度对 end-to-end 的理解：

非 end-to-end 方法：

代表性的算法是 RCNN 系列目标检测方法，这种方法需要先在图像中提取可能含有目标的候选框Region Proposal)， 然后将这些候选框输入到 CNN 模型，让 CNN 判断候选框中是否真的有目标，及目标的类别是什么。在我们看到的结果中，往往是类似与下图这种，在整幅图中用矩形框标记目标位置和大小，并且告诉我们框中的物体是什么。
这种标记的过程，其实是有两部分组成，一是目标所在位置及大小，二是目标的类别。在整个算中，目标位置和大小其实是包含在 Region Proposal 的过程里，而类别的判定则是在 CNN 中来判定。
<ol start="2">
end-to-end 方法：

end-to-end 方法的典型代表就是有名的 YOLO 系列算法。上面面的方法中，CNN 本质的作用是用来分类，定位的功能其并没有做到。而 YOLO 算法就是只通过 CNN 网络，就能够实现目标的定和识别。也就是原始图像输入到 CNN 网络中，直接输出图像中所有目标的位置和目标的类别。这种法就是 end-to-end(端对端)的方法，一端输入我的原始图像，一端输出我想得到的结果。只关心输和输出，中间的步骤全部都不管。