# 深度学习测试题

**1. 请简要介绍下tensorflow的计算图**

答：Tensorflow是一个通过计算图的形式来表述计算的编程系统，计算图也叫数据流图，可以把计算图看做是一种有向图，Tensorflow中的每一个节点都是计算图上的一个Tensor, 也就是张量，而节点之间的边描述了计算之间的依赖关系(定义时)和数学操作(运算时)。

**2. Sigmoid、Tanh、ReLu这三个激活函数有什么缺点或不足，有没改进的激活函数？**

答：

sigmoid缺点：若激活值很大的时候或者很小，激活函数在其区域梯度很小使得训练速度很慢。

Tanh缺点：同上。

Tanh优点：数据类似于集中于零左右。

ReLU缺点：在零的时候不可导，若激活值小于零，梯度为零，使得训练速度很慢，不过这种情况很少发生，有足够多的神经元使得其z值大于零。

改进：采用含虚弱的ReLU激活函数，即若小于零时，也让其有点梯度。比如小于零时，激活函数为：0.01Z。

**3. 为什么引入非线性激励函数？**

答： **激励函数可以给神经网络引入非线性因素，可以把当前特征空间映射到其他空间，使其几乎能拟合现实中任何问题，增加网络的能力。**

第一，对于神经网络来说，网络的每一层相当于f(wx+b)=f(w'x)，对于线性函数，其实相当于f(x)=x，那么在线性激活函数下，每一层相当于用一个矩阵去乘以x，那么多层就是反复的用矩阵去乘以输入。根据矩阵的乘法法则，多个矩阵相乘得到一个大矩阵。所以线性激励函数下，多层网络与一层网络相当。比如，两层的网络f(W1\*f(W2x))=W1W2x=Wx。

第二，非线性变换是深度学习有效的原因之一。原因在于非线性相当于对空间进行变换，变换完成后相当于对问题空间进行简化，原来线性不可解的问题现在变得可以解了。

下图可以很形象的解释这个问题，左图用一根线是无法划分的，经过一系列变换后，就变成线性可解的问题了，很像SVM中的核函数（如高斯核函数）。

图表

描述已自动生成

如果不用激励函数（其实相当于激励函数是f(x) = x），在这种情况下你每一层输出都是上层输入的线性函数，很容易验证。无论你神经网络有多少层，输出都是输入的线性组合，与没有隐藏层效果相当，这种情况就是最原始的感知机（Perceptron）了。

正因为上面的原因，我们决定引入非线性函数作为激励函数，这样深层神经网络就有意义了（不再是输入的线性组合，可以逼近任意函数）。最早的想法是sigmoid函数或者tanh函数，输出有界，很容易充当下一层输入（以及一些人的生物解释）。

**4. 什麽样的资料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。

**5. 如何解决梯度消失和梯度膨胀？**

（1）梯度消失：

根据链式法则，如果每一层神经元对上一层的输出的偏导乘上权重结果都小于1的话，那么即使这个结果是0.99，在经过足够多层传播之后，误差对输入层的偏导会趋于0。可以采用ReLU激活函数有效的解决梯度消失的情况，也可以用Batch Normalization解决这个问题。

（2）梯度膨胀

根据链式法则，如果每一层神经元对上一层的输出的偏导乘上权重结果都大于1的话，在经过足够多层传播之后，误差对输入层的偏导会趋于无穷大

可以通过激活函数来解决，或用Batch Normalization解决这个问题。

**6. 神经网络中激活函数的真正意义？一个激活函数需要具有哪些必要的属性？还有哪些属性是好的属性但不必要的？**

(1). 非线性：即导数不是0**。**这个条件是多层神经网络的基础，保证多层网络不退化成单层线性网络。这也是激活函数的意义所在。

(2). 几乎处处可微（可导）：可微性保证了在优化中梯度的可计算性。传统的激活函数如sigmoid等满足处处可微。对于分段线性函数比如ReLU，只满足几乎处处可微（即仅在有限个点处不可微）。对于SGD算法来说，由于几乎不可能收敛到梯度接近零的位置，有限的不可微点对于优化结果不会有很大影响[1]。

(3). 计算简单：非线性函数有很多。极端的说，一个多层神经网络也可以作为一个非线性函数，类似于Network In Network[2]中把它当做卷积操作的做法。但激活函数在神经网络前向的计算次数与神经元的个数成正比，因此简单的非线性函数自然更适合用作激活函数。这也是ReLU之流比其它使用Exp等操作的激活函数更受欢迎的其中一个原因。

(4). 非饱和性（saturation）：饱和指的是在某些区间梯度接近于零（即梯度消失），使得参数无法继续更新的问题。最经典的例子是Sigmoid，它的导数在x为比较大的正值和比较小的负值时都会接近于0。更极端的例子是阶跃函数，由于它在几乎所有位置的梯度都为0，因此处处饱和，无法作为激活函数。ReLU在x>0时导数恒为1，因此对于再大的正值也不会饱和。但同时对于x<0，其梯度恒为0，这时候它也会出现饱和的现象（在这种情况下通常称为dying ReLU）。Leaky ReLU[3]和PReLU[4]的提出正是为了解决这一问题。

(5). 单调性（monotonic）：即导数符号不变。这个性质大部分激活函数都有，除了诸如sin、cos等。个人理解，单调性使得在激活函数处的梯度方向不会经常改变，从而让训练更容易收敛。

(6). 输出范围有限：有限的输出范围使得网络对于一些比较大的输入也会比较稳定，这也是为什么早期的激活函数都以此类函数为主，如Sigmoid、TanH。但这导致了前面提到的梯度消失问题，而且强行让每一层的输出限制到固定范围会限制其表达能力。因此现在这类函数仅用于某些需要特定输出范围的场合，比如概率输出（此时loss函数中的log操作能够抵消其梯度消失的影响[1]）、LSTM里的gate函数。

(7). 接近恒等变换（identity）：即约等于x。这样的好处是使得输出的幅值不会随着深度的增加而发生显著的增加，从而使网络更为稳定，同时梯度也能够更容易地回传。这个与非线性是有点矛盾的，因此激活函数基本只是部分满足这个条件，比如TanH只在原点附近有线性区（在原点为0且在原点的导数为1），而ReLU只在x>0时为线性。这个性质也让初始化参数范围的推导更为简单[5][4]。额外提一句，这种恒等变换的性质也被其他一些网络结构设计所借鉴，比如CNN中的ResNet[6]和RNN中的LSTM。

(8). 参数少：大部分激活函数都是没有参数的。像PReLU带单个参数会略微增加网络的大小。还有一个例外是Maxout[7]，尽管本身没有参数，但在同样输出通道数下k路Maxout需要的输入通道数是其它函数的k倍，这意味着神经元数目也需要变为k倍；但如果不考虑维持输出通道数的情况下，该激活函数又能将参数个数减少为原来的k倍。

(9). 归一化（normalization）：这个是最近才出来的概念，对应的激活函数是SELU[8]，主要思想是使样本分布自动归一化到零均值、单位方差的分布，从而稳定训练。在这之前，这种归一化的思想也被用于网络结构的设计，比如Batch Normalization[9]。

**7. 什么是梯度爆炸？来的的问题是？**

w = w - lr\*梯度  
反向传播中，求导的链式法则，将梯度不断相乘相乘再相乘，如果都是>1的话，最后得到的梯度会很大，损失函数将会发散而不会收敛。

**梯度的定义：**误差梯度是神经网络训练过程中**计算的方向和数值**，用于以正确的方向和合适的量更新网络权重。

**梯度爆炸产生的原因：**在深层网络或循环神经网络中，误差梯度可在更新中累积，变成非常大的梯度，然后导致网络权重的大幅更新，并因此使网络变得不稳定。  
在极端情况下，权重的值变得非常大，以至于溢出，导致 NaN 值。网络层之间的梯度（值大于 1.0）重复相乘导致的指数级增长会产生梯度爆炸。

**梯度爆炸带来的问题：**在深度多层感知机网络中，梯度爆炸会引起网络不稳定，最好的结果是无法从训练数据中学习，而最坏的结果是出现无法再更新的 NaN 权重值。

梯度爆炸导致学习过程不稳定。  
  
在循环神经网络中，梯度爆炸会导致网络不稳定，无法利用训练数据学习，最好的结果是网络无法学习长的输入序列数据。

**8. 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：

模型无法从训练数据中获得更新（如低损失）。

模型不稳定，导致更新过程中的损失出现显著变化。

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。

训练过程中模型梯度快速变大。

训练过程中模型权重变成 NaN 值。

训练过程中，每个节点和层的误差梯度值持续超过 1.0。

**9. 如何修复梯度爆炸问题？**

1. 重新设计网络模型

在深度神经网络中，梯度爆炸可以通过重新设计层数更少的网络来解决。

使用更小的批尺寸对网络训练也有好处。

在循环神经网络中，训练过程中在更少的先前时间步上进行更新（沿时间的截断反向传播，truncated Backpropagation through time）可以缓解梯度爆炸问题。

2. 使用 ReLU 激活函数

在深度多层感知机神经网络中，梯度爆炸的发生可能是因为激活函数，如之前很流行的 Sigmoid 和 Tanh 函数。

使用 ReLU 激活函数可以减少梯度爆炸。采用 ReLU 激活函数是最适合隐藏层的新实践。

3. 使用长短期记忆网络

在循环神经网络中，梯度爆炸的发生可能是因为某种网络的训练本身就存在不稳定性，如随时间的反向传播本质上将循环网络转换成深度多层感知机神经网络。

使用长短期记忆（LSTM）单元和相关的门类型神经元结构可以减少梯度爆炸问题。

采用 LSTM 单元是适合循环神经网络的序列预测的最新最好实践。

4. 使用梯度截断（Gradient Clipping）

在非常深且批尺寸较大的多层感知机网络和输入序列较长的 LSTM 中，仍然有可能出现梯度爆炸。如果梯度爆炸仍然出现，你可以在训练过程中检查和限制梯度的大小。这就是梯度截断。

处理梯度爆炸有一个简单有效的解决方案：如果梯度超过阈值，就截断它们。

——《Neural Network Methods in Natural Language Processing》，2017.

具体来说，检查误差梯度的值是否超过阈值，如果超过，则截断梯度，将梯度设置为阈值。

梯度截断可以一定程度上缓解梯度爆炸问题（梯度截断，即在执行梯度下降步骤之前将梯度设置为阈值）。

——《深度学习》，2016.

在 Keras 深度学习库中，你可以在训练之前设置优化器上的 clipnorm 或 clipvalue 参数，来使用梯度截断。

5. 使用权重正则化（Weight Regularization）

如果梯度爆炸仍然存在，可以尝试另一种方法，即检查网络权重的大小，并惩罚产生较大权重值的损失函数。该过程被称为权重正则化，通常使用的是 L1 惩罚项（权重绝对值）或 L2 惩罚项（权重平方）。

对循环权重使用 L1 或 L2 惩罚项有助于缓解梯度爆炸。

——On the difficulty of training recurrent neural networks，2013.

在 Keras 深度学习库中，你可以通过在层上设置 kernel\_regularizer 参数和使用 L1 或 L2 正则化项进行权重正则化。

**10. LSTM结构推导，为什么比RNN好？**  
  
答：  
  
推导forget gate，input gate，cell state， hidden information等的变化；因为LSTM有进有出且当前的cell informaton是通过input gate控制之后叠加的，RNN是叠乘，因此LSTM可以防止梯度消失或者爆炸。

**11. 为什么LSTM模型中既存在sigmoid又存在tanh两种激活函数，而不是选择统一一种sigmoid或者tanh？这样做的目的是什么？**

答：

图示

描述已自动生成

sigmoid 用在了各种gate上，产生0~1之间的值，这个一般只有sigmoid最直接了。

tanh 用在了状态和输出上，是对数据的处理，这个用其他激活函数或许也可以。

**12. 深度学习常用的方法有哪些？**  
  
解析：  
  
全连接DNN（相邻层相互连接、层内无连接）：   
  
AutoEncoder(尽可能还原输入)、Sparse Coding（在AE上加入L1规范）、RBM（解决概率问题）—–>特征探测器——>栈式叠加 贪心训练   
  
RBM—->DBN   
  
解决全连接DNN的全连接问题—–>CNN   
  
解决全连接DNN的无法对时间序列上变化进行建模的问题—–>RNN—解决时间轴上的梯度消失问题——->LSTM  
  
DNN是传统的全连接网络，可以用于广告点击率预估，推荐等。其使用embedding的方式将很多离散的特征编码到神经网络中，可以很大的提升结果。  
  
CNN主要用于计算机视觉(Computer Vision)领域，CNN的出现主要解决了DNN在图像领域中参数过多的问题。同时，CNN特有的卷积、池化、batch normalization、Inception、ResNet、DeepNet等一系列的发展也使得在分类、物体检测、人脸识别、图像分割等众多领域有了长足的进步。同时，CNN不仅在图像上应用很多，在自然语言处理上也颇有进展，现在已经有基于CNN的语言模型能够达到比LSTM更好的效果。在最新的AlphaZero中，CNN中的ResNet也是两种基本算法之一。  
  
GAN是一种应用在生成模型的训练方法，现在有很多在CV方面的应用，例如图像翻译，图像超清化、图像修复等等。  
  
RNN主要用于自然语言处理(Natural Language Processing)领域，用于处理序列到序列的问题。普通RNN会遇到梯度爆炸和梯度消失的问题。所以现在在NLP领域，一般会使用LSTM模型。在最近的机器翻译领域，Attention作为一种新的手段，也被引入进来。  
  
除了DNN、RNN和CNN外， 自动编码器(AutoEncoder)、稀疏编码(Sparse Coding)、深度信念网络(DBM)、限制玻尔兹曼机(RBM)也都有相应的研究。

**13**. **梯度下降法的神经网络容易收敛到局部最优，为什么应用广泛？**

解析：

深度神经网络“容易收敛到局部最优”，很可能是一种想象，实际情况是，我们可能从来没有找到过“局部最优”，更别说全局最优了。

很多人都有一种看法，就是“局部最优是神经网络优化的主要难点”。这来源于一维优化问题的直观想象。在单变量的情形下，优化问题最直观的困难就是有很多局部极值，如

图示

描述已自动生成

人们直观的想象，高维的时候这样的局部极值会更多，指数级的增加，于是优化到全局最优就更难了。然而单变量到多变量一个重要差异是，单变量的时候，Hessian矩阵只有一个特征值，于是无论这个特征值的符号正负，一个临界点都是局部极值。但是在多变量的时候，Hessian有多个不同的特征值，这时候各个特征值就可能会有更复杂的分布，如有正有负的不定型和有多个退化特征值（零特征值）的半定型

图表, 表面图

描述已自动生成

在后两种情况下，是很难找到局部极值的，更别说全局最优了。

　　现在看来，神经网络的训练的困难主要是鞍点的问题。在实际中，我们很可能也从来没有真的遇到过局部极值。Bengio组这篇文章Eigenvalues of the Hessian in Deep Learning（https://arxiv.org/abs/1611.07476）里面的实验研究给出以下的结论：

• Training stops at a point that has a small gradient. The norm of the gradient is not zero, therefore it does not, technically speaking, converge to a critical point.

• There are still negative eigenvalues even when they are small in magnitude.

　　另一方面，一个好消息是，即使有局部极值，具有较差的loss的局部极值的吸引域也是很小的Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes。（https://arxiv.org/abs/1706.10239）

For the landscape of loss function for deep networks, the volume of basin of attraction of good minima dominates over that of poor minima, which guarantees optimization methods with random initialization to converge to good minima.

　　所以，很可能我们实际上是在“什么也没找到”的情况下就停止了训练，然后拿到测试集上试试，“咦，效果还不错”。

　　补充说明，这些都是实验研究结果。理论方面，各种假设下，深度神经网络的Landscape 的鞍点数目指数增加，而具有较差loss的局部极值非常少。

**14. 什么是RNN？**

解析：

RNNs的目的使用来处理序列数据。在传统的神经网络模型中，是从输入层到隐含层再到输出层，层与层之间是全连接的，每层之间的节点是无连接的。但是这种普通的神经网络对于很多问题却无能无力。例如，你要预测句子的下一个单词是什么，一般需要用到前面的单词，因为一个句子中前后单词并不是独立的。

RNNs之所以称为循环神经网路，即一个序列当前的输出与前面的输出也有关。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐藏层之间的节点不再无连接而是有连接的，并且隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出。

理论上，RNNs能够对任何长度的序列数据进行处理。但是在实践中，为了降低复杂性往往假设当前的状态只与前面的几个状态相关，下图便是一个典型的RNNs：

图示

描述已自动生成

RNNs包含输入单元(Input units)，输入集标记为{x0,x1,...,xt,xt+1,...}，而输出单元(Output units)的输出集则被标记为{y0,y1,...,yt,yt+1.,..}。RNNs还包含隐藏单元(Hidden units)，我们将其输出集标记为{s0,s1,...,st,st+1,...}，这些隐藏单元完成了最为主要的工作。你会发现，在图中：有一条单向流动的信息流是从输入单元到达隐藏单元的，与此同时另一条单向流动的信息流从隐藏单元到达输出单元。在某些情况下，RNNs会打破后者的限制，引导信息从输出单元返回隐藏单元，这些被称为“Back Projections”，并且隐藏层的输入还包括上一隐藏层的状态，即隐藏层内的节点可以自连也可以互连。

图示, 示意图

描述已自动生成

上图将循环神经网络进行展开成一个全神经网络。例如，对一个包含5个单词的语句，那么展开的网络便是一个五层的神经网络，每一层代表一个单词。对于该网络的计算过程如下：

（1）xt表示第t,t=1,2,3...步(step)的输入。比如，x1为第二个词的one-hot向量(根据上图，x0为第一个词)；

（2） st为隐藏层的第t步的状态，它是网络的记忆单元。 st根据当前输入层的输出与上一步隐藏层的状态进行计算。st=f(Uxt+Wst−1)，其中f一般是非线性的激活函数，如tanh或ReLU，在计算s0时，即第一个单词的隐藏层状态，需要用到s−1，但是其并不存在，在实现中一般置为0向量；

（3）ot是第t步的输出，如下个单词的向量表示，ot=softmax(Vst).

**15. rcnn、fast-rcnn和faster-rcnn三者的区别是什么？**

解析：

首先膜拜RBG（Ross B. Girshick）大神，不仅学术牛，工程也牛，代码健壮，文档详细，clone下来就能跑。断断续续接触detection几个月，将自己所知做个大致梳理，业余级新手，理解不对的地方还请指正。

传统的detection主流方法是DPM(Deformable parts models)， 在VOC2007上能到43%的mAP，虽然DPM和CNN看起来差别很大，但RBG大神说“Deformable Part Models are Convolutional Neural Networks”（http://arxiv.org/abs/1409.5403）。

CNN流行之后，Szegedy做过将detection问题作为回归问题的尝试（Deep Neural Networks for Object Detection），但是效果差强人意，在VOC2007上mAP只有30.5%。既然回归方法效果不好，而CNN在分类问题上效果很好，那么为什么不把detection问题转化为分类问题呢？

RBG的RCNN使用region proposal（具体用的是Selective Search Koen van de Sande: Segmentation as Selective Search for Object Recognition）来得到有可能得到是object的若干（大概10^3量级）图像局部区域，然后把这些区域分别输入到CNN中，得到区域的feature，再在feature上加上分类器，判断feature对应的区域是属于具体某类object还是背景。当然，RBG还用了区域对应的feature做了针对boundingbox的回归，用来修正预测的boundingbox的位置。

RCNN在VOC2007上的mAP是58%左右。RCNN存在着重复计算的问题（proposal的region有几千个，多数都是互相重叠，重叠部分会被多次重复提取feature），于是RBG借鉴Kaiming He的SPP-net的思路单枪匹马搞出了Fast-RCNN，跟RCNN最大区别就是Fast-RCNN将proposal的region映射到CNN的最后一层conv layer的feature map上，这样一张图片只需要提取一次feature，大大提高了速度，也由于流程的整合以及其他原因，在VOC2007上的mAP也提高到了68%。

探索是无止境的。Fast-RCNN的速度瓶颈在Region proposal上，于是RBG和Kaiming He一帮人将Region proposal也交给CNN来做，提出了Faster-RCNN。Fater-RCNN中的region proposal netwrok实质是一个Fast-RCNN，这个Fast-RCNN输入的region proposal的是固定的（把一张图片划分成n\*n个区域，每个区域给出9个不同ratio和scale的proposal），输出的是对输入的固定proposal是属于背景还是前景的判断和对齐位置的修正（regression）。Region proposal network的输出再输入第二个Fast-RCNN做更精细的分类和Boundingbox的位置修正。

Fater-RCNN速度更快了，而且用VGG net作为feature extractor时在VOC2007上mAP能到73%。个人觉得制约RCNN框架内的方法精度提升的瓶颈是将dectection问题转化成了对图片局部区域的分类问题后，不能充分利用图片局部object在整个图片中的context信息。

可能RBG也意识到了这一点，所以他最新的一篇文章YOLO（http://arxiv.org/abs/1506.02640）又回到了regression的方法下，这个方法效果很好，在VOC2007上mAP能到63.4%，而且速度非常快，能达到对视频的实时处理（油管视频：https://www.youtube.com/channel/UC7ev3hNVkx4DzZ3LO19oebg），虽然不如Fast-RCNN，但是比传统的实时方法精度提升了太多，而且我觉得还有提升空间。

**16. 在神经网络中，有哪些办法防止过拟合？**

解析：

缓解过拟合：

① Dropout

② 加L1/L2正则化

③ BatchNormalization

④ 网络bagging

**17. CNN是什么？CNN关键的层有哪些？**

解析：

CNN是卷积神经网络，具体详见此文：https://blog.csdn.net/v\_july\_v/article/details/51812459。

其关键层有：

① 输入层，对数据去均值，做data augmentation等工作

② 卷积层，局部关联抽取feature

③ 激活层，非线性变化

④ 池化层，下采样

⑤ 全连接层，增加模型非线性

⑥ 高速通道，快速连接

⑦ BN层，缓解梯度弥散

**18. GRU是什么？GRU对LSTM做了哪些改动？**

解析：

GRU是Gated Recurrent Units，是循环神经网络的一种。

GRU只有两个门（update和reset），LSTM有三个门（forget，input，output），GRU直接将hidden state 传给下一个单元，而LSTM用memory cell 把hidden state 包装起来。

**19. 请简述应当从哪些方向上思考和解决深度学习中出现的的over fitting问题？**

解析：

如果模型的训练效果不好，可先考察以下几个方面是否有可以优化的地方。

(1)选择合适的损失函数（choosing proper loss ）

神经网络的损失函数是非凸的，有多个局部最低点，目标是找到一个可用的最低点。非凸函数是凹凸不平的，但是不同的损失函数凹凸起伏的程度不同，例如下述的平方损失和交叉熵损失，后者起伏更大，且后者更容易找到一个可用的最低点，从而达到优化的目的。

- Square Error（平方损失）

- Cross Entropy（交叉熵损失）

(2)选择合适的Mini-batch size

采用合适的Mini-batch进行学习，使用Mini-batch的方法进行学习，一方面可以减少计算量，一方面有助于跳出局部最优点。因此要使用Mini-batch。更进一步，batch的选择非常重要，batch取太大会陷入局部最小值，batch取太小会抖动厉害，因此要选择一个合适的batch size。

(3)选择合适的激活函数（New activation function）

使用激活函数把卷积层输出结果做非线性映射，但是要选择合适的激活函数。

- Sigmoid函数是一个平滑函数，且具有连续性和可微性，它的最大优点就是非线性。但该函数的两端很缓，会带来猪队友的问题，易发生学不动的情况，产生梯度弥散。

- ReLU函数是如今设计神经网络时使用最广泛的激活函数，该函数为非线性映射，且简单，可缓解梯度弥散。

(4)选择合适的自适应学习率（apdative learning rate）

- 学习率过大，会抖动厉害，导致没有优化提升

- 学习率太小，下降太慢，训练会很慢

(5)使用动量（Momentum）

在梯度的基础上使用动量，有助于冲出局部最低点。

如果以上五部分都选对了，效果还不好，那就是产生过拟合了，可使如下方法来防止过拟合，分别是

- 1.早停法（earyly stoping）。早停法将数据分成训练集和验证集，训练集用来计算梯度、更新权重和阈值，验证集用来估计误差，若训练集误差降低但验证集误差升高，则停止训练，同时返回具有最小验证集误差的连接权和阈值。

- 2.权重衰减（Weight Decay）。到训练的后期，通过衰减因子使权重的梯度下降地越来越缓。

- 3.Dropout。Dropout是正则化的一种处理，以一定的概率关闭神经元的通路，阻止信息的传递。由于每次关闭的神经元不同，从而得到不同的网路模型，最终对这些模型进行融合。

- 4.调整网络结构（Network Structure）。

**20. 训练过程中模型不收敛，是否说明这个模型无效，致模型不收敛的原因有哪些?**

不一定。导致模型不收敛的原因有很多种可能，常见的有以下几种：

* 没有对数据做归一化。
* 没有检查过你的结果。这里的结果包括预处理结果和最终的训练测试结果。
* 忘了做数据预处理。
* 忘了使用正则化。
* Batch Size设的太大。
* 学习率设的不对。
* 最后一层的激活函数用的不对。
* 网络存在坏梯度。比如Relu对负值的梯度为0，反向传播时，0梯度就是不传播。
* 参数初始化错误。
* 网络太深。隐藏层神经元数量错误。

**21. 神经网络中权值共享的理解？**

权值(权重)共享这个词是由LeNet5模型提出来的。以CNN为例，在对一张图偏进行卷积的过程中，使用的是同一个卷积核的参数。 比如一个3×3×1的卷积核，这个卷积核内9个的参数被整张图共享，而不会因为图像内位置的不同而改变卷积核内的权系数。说的再直白一些，就是用一个卷积核不改变其内权系数的情况下卷积处理整张图片（当然CNN中每一层不会只有一个卷积核的，这样说只是为了方便解释而已）。

**22. 图像处理中平滑和锐化操作是什么？**

**平滑处理（smoothing）也称模糊处理（bluring）**，主要用于消除图像中的噪声部分，平滑处理常用的用途是用来减少图像上的噪点或失真，平滑主要使用图像滤波。在这里，我个人认为可以把图像平滑和图像滤波联系起来，因为图像平滑常用的方法就是图像滤波器。 在OpenCV3中常用的图像滤波器有以下几种：

* 方框滤波——BoxBlur函数
* 均值滤波（邻域平均滤波）——Blur函数
* 高斯滤波——GaussianBlur函数
* 中值滤波——medianBlur函数
* 双边滤波——bilateralFilter函数 图像锐化操作是为了突出显示图像的边界和其他细节，而图像锐化实现的方法是通过各种算子和滤波器实现的——Canny算子、Sobel算子、Laplacian算子以及Scharr滤波器。

**23. VGG使用2个3\*3卷积的优势在哪里？**

(1). **减少网络层参数**。用两个3*3卷积比用1个5*5卷积拥有更少的参数量，只有后者的2∗3∗35∗5=0.72。但是起到的效果是一样的，两个3*3的卷积层串联相当于一个5*5的卷积层，感受野的大小都是5×5，即1个像素会跟周围5\*5的像素产生关联。把下图当成动态图看，很容易看到两个3×3卷积层堆叠（没有空间池化）有5×5的有效感受野。

形状

描述已自动生成

(2). **更多的非线性变换**。2个3*3卷积层拥有比1个5*5卷积层更多的非线性变换（前者可以使用两次ReLU激活函数，而后者只有一次），使得卷积神经网络对特征的学习能力更强。

**24. Relu比Sigmoid效果好在哪里？**

ReLU激活函数公式如下：

图示

中度可信度描述已自动生成

relu函数方程 ReLU 的输出要么是 0, 要么是输入本身。虽然方程简单，但实际上效果更好。在网上看了很多版本的解释，有从程序实例分析也有从数学上分析，我找了个相对比较直白的回答，如下： (1). ReLU函数计算简单，可以减少很多计算量。反向传播求误差梯度时，涉及除法，计算量相对较大，采用ReLU激活函数，可以节省很多计算量； (2). **避免梯度消失问题**。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问题（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），从而无法完成深层网络的训练。 (3). 可以缓解过拟合问题的发生。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。

**25. 对fine-tuning(微调模型的理解)，为什么要修改最后几层神经网络权值？**

使用预训练模型的好处，在于利用训练好的SOTA模型权重去做特征提取，可以节省我们训练模型和调参的时间。

至于为什么只微调最后几层神经网络权重，是因为： (1). CNN中更靠近底部的层（定义模型时先添加到模型中的层）编码的是更加通用的可复用特征，而更靠近顶部的层（最后添加到模型中的层）编码的是更专业业化的特征。微调这些更专业化的特征更加有用，它更代表了新数据集上的有用特征。 (2). 训练的参数越多，过拟合的风险越大。很多SOTA模型拥有超过千万的参数，在一个不大的数据集上训练这么多参数是有过拟合风险的，除非你的数据集像Imagenet那样大。

**26. 卷积层和池化层有什么区别？**

* 卷积层有参数，池化层没有参数
* **经过卷积层节点矩阵深度会改变**，池化层不会改变节点矩阵的深度，但是它可以缩小节点矩阵的大小

**27. dropout具体工作流程是怎样的？**

以 标准神经网络为例，正常的流程是：我们首先把输入数据x通过网络前向传播，然后把误差反向传播一决定如何更新参数让网络进行学习。使用dropout之后，过程变成如下：

(1). 首先随机（临时）删掉网络中一半的隐藏神经元，输入输出神经元保持不变（图3中虚线为部分临时被删除的神经元）； (2). 然后把输入x通过修改后的网络进行前向传播计算，然后把得到的损失结果通过修改的网络反向传播。一小批训练样本执行完这个过程后，在没有被删除的神经元上按照随机梯度下降法更新对应的参数（w，b）； (3). 然后重复这一过程：

* 恢复被删掉的神经元（此时被删除的神经元保持原样没有更新w参数，而没有被删除的神经元已经有所更新）
* 从隐藏层神经元中随机选择一个一半大小的子集临时删除掉（同时备份被删除神经元的参数）。
* 对一小批训练样本，先前向传播然后反向传播损失并根据随机梯度下降法更新参数（w，b） （没有被删除的那一部分参数得到更新，删除的神经元参数保持被删除前的结果）。

**28. 如何选择激活函数？**

1. 如果输出是 0、1 值（二分类问题），则输出层选择 sigmoid 函数，然后其它的所有单元都选择 Relu 函数。
2. 如果在隐藏层上不确定使用哪个激活函数，那么通常会使用 Relu 激活函数。有时，也会使用 tanh 激活函数，但 Relu 的一个优点是：当是负值的时候，导数等于 0。
3. sigmoid 激活函数：除了输出层是一个二分类问题基本不会用它。
4. tanh 激活函数：tanh 是非常优秀的，几乎适合所有场合。
5. 如果遇到了一些死的神经元，我们可以使用 Leaky ReLU 函数。

**29. 解释反卷积的原理和用途**

反卷积即转置卷积，正向传播时乘以卷积核的转置矩阵，反向传播时乘以卷积核矩阵

由卷积输出结果近似重构输入数据，上采样

**30. 卷积神经网络为什么会具有平移不变性？**

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。

**31. 1x1卷积有什么用途？**

通道降维，保证卷积神经网络可以接受任何尺寸的输入数据

**32. 为什么现在倾向于用小尺寸的卷积核？**

用多个小卷积核串联可以有大卷积核同样的能力，而且参数更少，另外有更多次的激活函数作用，增强非线性

**33. 解释GoogLeNet的Inception模块的原理**

对输入图像用多个不同尺寸的卷积核、池化操作进行同时处理，然后将输出结果按照通道拼接起来。

**34. 什么是鞍点问题？**

梯度为0，Hessian矩阵不定的点，不是极值点。

**35. 有哪些聚类算法？**

层次聚类，k均值算法，DBSCAN算法，OPTICS算法，谱聚类。

**36. 说明在图像分类任务中可视化CNN特征的两种方法？**

答：

* 输入遮挡：遮挡输入图像的一部分，看看哪部分对分类的影响最大。 例如，针对某个训练好的图像分类模型，将下列图像作为输入。如果我们看到第三幅图像被分类为狗狗的概率为98%，而第二幅图像的准确率仅为65%，则说明眼睛对于对分类的影响更大。
* 激活最大化：创建一个人造的输入图像，以最大化目标响应（梯度上升）。

**37. 如何理解Adam优化器的概念？**

答：Adam结合了两个想法来改善收敛性：每个参数更新可加快收敛速度；动量可避免卡在鞍点上。

**38. 什么是数据扩充并举例说明？**

答：数据扩充是一种技术，通过操作原始数据来增加输入数据。例如，对于图像，我们可以执行以下操作：旋转图像、翻转图像、添加高斯模糊等。

**39. 如何理解GAN的概念？**

答：GAN（Generative Adversarial Network）即生成对抗网络，通常由两个神经网络D和G组成，其中D指的是判别器（Discriminator），而G指生成网络（Generative Network）。这种模型的目标是创建数据，例如创建与真实图像并无二样的图像。假设我们想要创建一只猫的对抗示例。神经网络G负责生成图像，而神经网络D则负责判断图像是否是猫。G的目标是“愚弄”D——将G的输出始终分类为猫。

**40. 使用Batchnorm有什么优势？**

答：Batchnorm能够加快训练过程，而且（一些噪音的副产品）还具有调节作用。