1. **CNN的卷积核是单层的还是多层的？**

一般而言，深度卷积网络是一层又一层的。层的本质是特征图, 存贮输入数据或其中间表示值。一组卷积核则是联系前后两层的网络参数表达体, 训练的目标就是每个卷积核的权重参数组。

描述网络模型中某层的厚度，通常用名词通道channel数或者特征图feature map数。不过人们更习惯把作为数据输入的前层的厚度称之为通道数（比如RGB三色图层称为输入通道数为3），把作为卷积输出的后层的厚度称之为特征图数。

卷积核(filter)一般是3D多层的，除了面积参数, 比如3x3之外, 还有厚度参数H（2D的视为厚度1). 还有一个属性是卷积核的个数N。

卷积核的厚度H, 一般等于前层厚度M(输入通道数或feature map数). 特殊情况M > H。

卷积核的个数N, 一般等于后层厚度(后层feature maps数，因为相等所以也用N表示)。

卷积核通常从属于后层，为后层提供了各种查看前层特征的视角，这个视角是自动形成的。

卷积核厚度等于1时为2D卷积，对应平面点相乘然后把结果加起来，相当于点积运算；

卷积核厚度大于1时为3D卷积，每片分别平面点求卷积，然后把每片结果加起来，作为3D卷积结果；1x1卷积属于3D卷积的一个特例，有厚度无面积, 直接把每片单个点乘以权重再相加。

归纳之，卷积的意思就是把一个区域，不管是一维线段，二维方阵，还是三维长方块，全部按照卷积核的维度形状，对应逐点相乘再求和，浓缩成一个标量值也就是降到零维度，作为下一层的一个feature map的一个点的值

可以比喻一群渔夫坐一个渔船撒网打鱼，鱼塘是多层水域，每层鱼儿不同。

船每次移位一个stride到一个地方，每个渔夫撒一网，得到收获，然后换一个距离stride再撒，如此重复直到遍历鱼塘。

A渔夫盯着鱼的品种，遍历鱼塘后该渔夫描绘了鱼塘的鱼品种分布；

B渔夫盯着鱼的重量，遍历鱼塘后该渔夫描绘了鱼塘的鱼重量分布；

还有N-2个渔夫，各自兴趣各干各的；

最后得到N个特征图，描述了鱼塘的一切

2D卷积表示渔夫的网就是带一圈浮标的渔网，只打上面一层水体的鱼；

3D卷积表示渔夫的网是多层嵌套的渔网，上中下层水体的鱼儿都跑不掉；

1x1卷积可以视为每次移位stride，甩钩钓鱼代替了撒网；

下面解释一下特殊情况的 M > H：

实际上，除了输入数据的通道数比较少之外，中间层的feature map数很多，这样中间层算卷积会累死计算机（鱼塘太深，每层鱼都打，需要的鱼网太重了）。所以很多深度卷积网络把全部通道/特征图划分一下，每个卷积核只看其中一部分（渔夫A的渔网只打捞深水段，渔夫B的渔网只打捞浅水段）。这样整个深度网络架构是横向开始分道扬镳了，到最后才又融合。这样看来，很多网络模型的架构不完全是突发奇想，而是是被参数计算量逼得。特别是现在需要在移动设备上进行AI应用计算(也叫推断), 模型参数规模必须更小, 所以出现很多减少握手规模的卷积形式, 现在主流网络架构大都如此。

1. **什么是卷积？**

对图像（不同的数据窗口数据）和滤波矩阵（一组固定的权重：因为每个神经元的多个权重固定，所以又可以看做一个恒定的滤波器filter）做内积（逐个元素相乘再求和）的操作就是所谓的『卷积』操作，也是卷积神经网络的名字来源。

1. **什么是CNN的池化pool层？**

池化，简言之，即取区域平均或最大

1. **deep learning（rnn、cnn）调参的经验**

 参数初始化

1) uniform均匀分布初始化：

传统的初始化权重问题是用标准正态分布（均值为0，方差为1）随机初始化

w = np.random.uniform(low=-scale, high=scale, size=[n\_in,n\_out])

把权值与偏置进行 均匀分布的初始化。用min 与 max 来控制它们的的上下 限，默认为（0，1）

2) Xavier初始法，适用于普通激活函数(tanh,sigmoid)：scale = np.sqrt(3/n)

适用于普通激活函数 (tanh,sigmoid)：stdev = np.sqrt(n)

3) He初始化，适用于ReLU：scale = np.sqrt(6/n)

4) normal高斯分布初始化：w = np.random.randn(n\_in,n\_out) \* stdev

# stdev为高斯分布的标准差，均值设为0

5) svd初始化：对RNN有比较好的效果。

 数据预处理方式

1） 0均值（zero-center） 是最常用的预处理方法，就是把数据的每一维-每一维的均值，这样数据就变成0均值的了。在numpy中，这个操作可以写成：X -= np.mean(X, axis = 0)。对于图片来讲，我们可以更简单地对所有pixel减去同一个均值（如 X -= np.mean(X)），当然也可以对RGB三个通道分别减均值。

2）归一化(Normalization) 是指将数据归一化到相同的尺度。通常有两种归一化的方法。第一种是0均值以后的数据的每一维/每一维的标准差（X /= np.std(X, axis = 0)）；另一种是将数据归一化到每一维的最大最小值为1和-1。这种归一化只适用于当你认为数据的不同维度应该具有相同的重要性时。对于图片来说，不同像素的尺度是基本一致的（0-255），因此我们并不需要对它进行归一化操作。

3） PCA和白化（PCA whitening） 是另一种形式的预处理方法。首先我们将数据变成0均值的，然后计算数据的协方差矩阵来得到数据不同维度之间的相关性：

 训练技巧

1） 要做梯度归一化,即算出来的梯度除以minibatch size

2）clip c(梯度裁剪): 限制最大梯度,其实是value = sqrt(w1^2+w2^2….),如果value超过了阈值,就算一个衰减系系数,让value的值等于阈值: 5,10,15

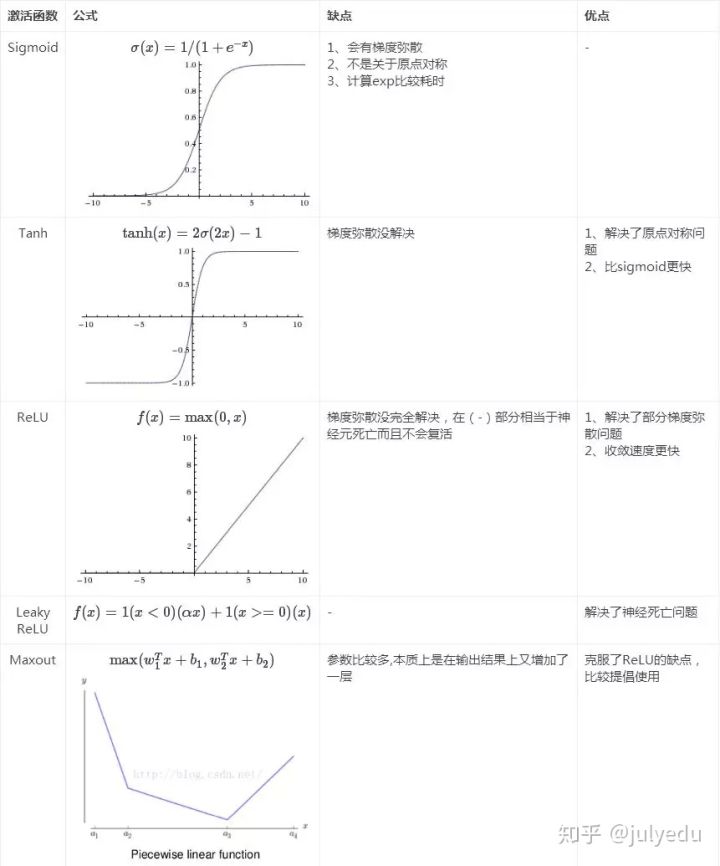
3）dropout对小数据防止过拟合有很好的效果,值一般设为0.5,小数据上dropout+sgd在我的大部分实验中，效果提升都非常明显.因此可能的话，建议一定要尝试一下。 dropout的位置比较有讲究, 对于RNN,建议放到输入->RNN与RNN->输出的位置.

4） adam,adadelta等,在小数据上,我这里实验的效果不如sgd, sgd收敛速度会慢一些，但是最终收敛后的结果，一般都比较好。如果使用sgd的话,可以选择从1.0或者0.1的学习率开始,隔一段时间,在验证集上检查一下,如果cost没有下降,就对学习率减半. 我看过很多论文都这么搞,我自己实验的结果也很好. 当然,也可以先用ada系列先跑,最后快收敛的时候,更换成sgd继续训练.同样也会有提升.据说adadelta一般在分类问题上效果比较好，adam在生成问题上效果比较好。

5） 除了gate之类的地方,需要把输出限制成0-1之外,尽量不要用sigmoid,可以用tanh或者relu之类的激活函数.1. sigmoid函数在-4到4的区间里，才有较大的梯度。之外的区间，梯度接近0，很容易造成梯度消失问题。2. 输入0均值，sigmoid函数的输出不是0均值的。

6) rnn的dim和embdding size,一般从128上下开始调整. batch size,一般从128左右开始调整.batch size合适最重要,并不是越大越好

1. **激活函数**



1. **请问人工神经网络中为什么ReLu要好过于tanh和sigmoid function？**

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。

1. **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？**

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

1. **如何解决RNN梯度爆炸和弥散的问题？**

就是当梯度大于一定阈值的的时候，将它截断为一个较小的数。

解决梯度弥散的问题，我们介绍了两种方法。第一种方法是将随机初始化，改为一个有关联的矩阵初始化。第二种方法是使用ReLU（Rectified Linear Units）代替sigmoid函数。ReLU的导数不是0就是1.因此，神经元的梯度将始终为1，而不会当梯度传播了一定时间之后变小。

1. **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。

1. **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。

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。—《深度学习》，2016。

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。

如何确定是否出现梯度爆炸？

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，模型损失变成 NaN。

训练过程中，每个节点和层的误差梯度值持续超过 1.0。

1. **请简述神经网络的发展史**

sigmoid会饱和，造成梯度消失。于是有了ReLU

ReLU负半轴是死区，造成梯度变0。于是有了LeakyReLU，PReLU

强调梯度和权值分布的稳定性，由此有了ELU，以及较新的SELU

太深了，梯度传不下去，于是有了highway

干脆连highway的参数都不要，直接变残差，于是有了ResNet

强行稳定参数的均值和方差，于是有了BatchNorm

在梯度流中增加噪声，于是有了 Dropout

RNN梯度不稳定，于是加几个通路和门控，于是有了LSTM

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

LSTM简化一下，有了GRU

GAN的JS散度有问题，会导致梯度消失或无效，于是有了WGAN

WGAN对梯度的clip有问题，于是有了WGAN-GP

1. **如何修复梯度爆炸问题？**

有很多方法可以解决梯度爆炸问题，本节列举了一些最佳实验方法。

（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。

默认值为 clipnorm=1.0 、clipvalue=0.5。详见：https://keras.io/optimizers/。

（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。

1. **在神经网络中，有哪些办法防止过拟合？**
2. Dropout
3. 加L1/L2正则化
4. BatchNormalization
5. 网络bagging
6. **CNN是什么，CNN关键的层有哪些？**

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
8. **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？**

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。

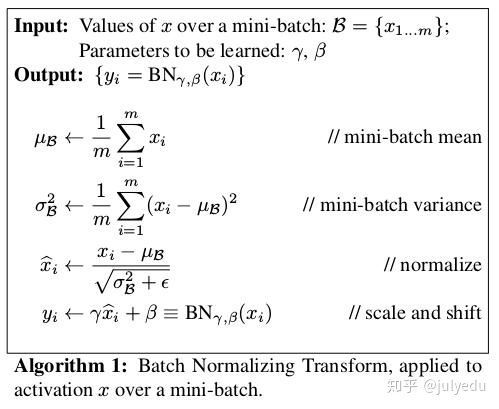
- 4.调整网络结构（Network Structure）。

1. **对于分类任务，通常使用交叉熵损失函数,回归任务中那样使用均方差**
2. **请写出Batch Normalization的计算方法及其应用**

批量归一化（Batch Normalization简称BN，其中，Normalization是数据标准化或归一化、规范化，Batch可以理解为批量，加起来就是批量标准化。解决在训练过程中中间层数据分布发生改变的问题，以防止梯度消失或爆炸、加快训练速度

与激活函数层、卷积层、全连接层、池化层一样，BN(Batch Normalization)也属于网络的一层。

BN的本质原理：在网络的每一层输入的时候，又插入了一个归一化层，也就是先做一个归一化处理（归一化至：均值0、方差为1），然后再进入网络的下一层。不过归一化层可不像我们想象的那么简单，它是一个可学习、有参数（γ、β）的网络层。



输入：输入数据x1..xm（这些数据是准备进入激活函数的数据）

计算过程中可以看到,

1.求数据均值；

2.求数据方差；

3.数据进行标准化（个人认为称作正态化也可以）

4.训练参数γ，β

5.输出y通过γ与β的线性变换得到新的值

因为BN保证每一层的输入分布稳定，这一点本身可以使得训练加速，而且另一方面它也可以帮助减少梯度消失和梯度爆炸的现象。