调参技巧（炼丹）

可复现性和一致性

我们会遇到需要类似的问题，但是假如从头开始写代码训练模型会比较耗时。建议构建一个自己熟知的baseline（先用成熟的开源项目以及它的默认配置），确保模型的可复现性。谨记在训练新的模型前先把该更改的代码进行更改，不要等它报错再改，将自己过去效果较好的代码进行记录，经常阅读。此时将该模型应用到相似的业务上时便可以避免因超参的微小改变而导致结果出现显著差异，假如出现便是数据太少。

资源利用

在数据集很大的情况下，建议先用 1/100、1/10 的数据跑一跑，对模型性能和训练时间有个底，一下全量数据到底需要跑多久。在没有足够的信心前不做大规模实验。

因为没有大量的卡供使用，在实验过程中最好单卡半天得一次结论进行对比，切记小数据用小模型，防止大半时间在等数据，减少数据处理的事件，前期把精力放在实验的效率上。

模型不 work

去除一些优化模型的组件，比如数据增广，各种类型的学习率和超参，损失函数，模型。可以先造一些尽量工作的模型，验证代码正确性。然后进行逐步准确率的加强。

有些指标是有滞后性的，需要等训练一段时间才开始动。很多人训练几步看没什么效果就把程序停掉开始 debug 了，但其实代码毫无问题。如此反复好几天甚至一两周都在原地踏步，其实需要做的仅仅是让程序自个儿安安静静地跑上几个小时或者一天...

模型不能得到预期的结果，有可能是网络结构搭错了。pytorch可能好一些，tensorflow里面的函数式网络定义很容易出现有些层并没有实际进入back propogation。建议先生成tflite或者pb文件，然后用可视化工具比如tensorboard来看一下网络结构和梯度传播。

数据增广

感性认识就是，数据是否需要增加或增广。

图像裁剪，对称变换，旋转平移，对比度，曝光度，色调都可以让模型在验证集上的表现更好。

模型大小

模型是大了还是小了，再根据速度和精度期望开始寻找合适的模型。

能用全卷积的任务，少用全连接层，参数量小。

组件的添加

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在大部分实验中，效果提升都非常明显.

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.

当你的模型有 Batch Normalization，初始化通常不需要操心，激活函数默认 Relu 即可。

一般顺序是 Conv - BN - Relu。有BN就别加dropout

如果没有 BN（很多任务上，BN降低训练难度，但是可能影响最终性能 ）

防止过拟合，常用的防止过拟合方法有使用L1正则项、L2正则项、dropout、提前终止、数据集扩充。

全卷积网络（FCN）生成任务中bn可能没啥用，甚至导致负效果。

数据预处理

试着要做一些数据归一化，要做梯度归一化,即算出来的梯度除以minibatch size

zero-center ,这个挺常用的. X -= np.mean(X, axis = 0) # zero-center X /= np.std(X, axis = 0) # normalize

PCA whitening,这个用的比较少.

如果输入的state是图片，那么可以进行resize和变成灰度图。进行resize的时候，尽可能保持原来的尺寸，这样进行输出action的时候能够保持尺度一致性。至于要不要变成灰度图，需要看看输入的state是否有较好的边缘，颜色是否携带较多信息等等。

将图片rescale到(-1, 1), 即除以127.5然后减去1，或者rescale到(0, 1)，即除以255，或者rescale之后减去均值除以方差。如果用全卷积网络做输出是image的任务，我一般用前两种；第三种常见于各种分类检测以及其他输出是label的任务。

激活函数

虽然有至少十种激活函数，但在 Relu 外只推荐试一下 Swish。

另外，构建序列神经网络（RNN）时要优先选用tanh激活函数。

优化器

优化器只推荐 Momentum 和 Adam。在这些方面做尝试意义不大，如果性能提升反倒可能说明模型不成熟。

adam,adadelta等,在小数据上,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.

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.

网络结构

基本模型上 ResNet, Unet 结构还是主流。

不推荐自己进行网络设计，比如把某层卷积改大一点，或者微调一下通道数。除非有特别要求，不要自己乱设计新组件。

超参

调参请在验证集上！

首先我们假设我们手上有一个正确的，没有bug，可以训练的模型，以及预先设立的误差目标。

那么分三种情况：

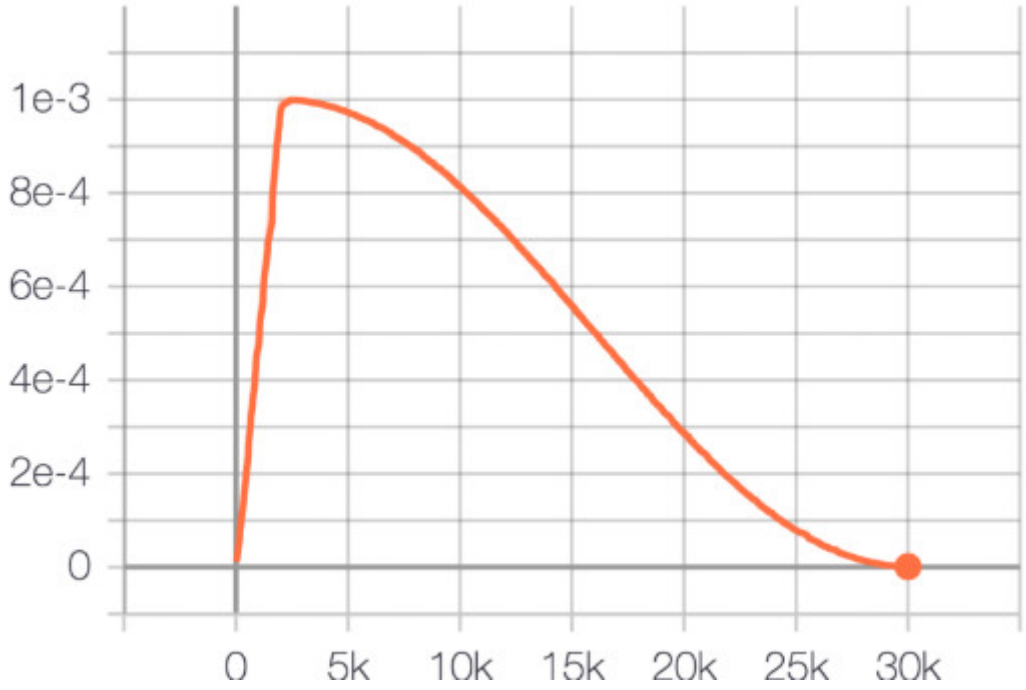
1、模型表现非常好，在训练集和验证集上都满足我们的目标。那不用调了。

2、模型在训练集上的误差很小，且各种softmax等loss 能减小到一个很小的值（对于我自己而言小于0.01），但是验证集的 loss 相对较大（对我的问题而言，一般会在0.3~0.6）。那就是过拟合了。

3、在训练集和验证集上的loss都比较大，都大于0.3，那可能就是欠拟合了。

learning rate 最重要，推荐了解 cosine learning rate 和 cyclic learning rate，其次是 batchsize 和 weight decay。记住loss变成NaN大多数是因为学习率选择不当，步子大了容易扯到蛋。

cosine learning rate如下图,开始为warmup热身阶段学习率较小防止NaN,后期开始变小更好到最低谷.



之前在8卡服务器跑imagenet的时候看的论文印象中这两个参数可以按线性调节，即batch size增大一倍，learning rate也可以相应增大一倍。大模型，大batch size，有batch norm可以用较大的学习率（1e-3这样子，imagenet上面有batch size256直接用0.1初始学习率的），目前我做gan based image2image translation一般是batch size16，learning rate 2e-4或者3e-4，训练非常稳定。

一般学习率从0.1或0.01开始尝试。学习率设置太大会导致训练十分不稳定，甚至出现Nan，设置太小会导致损失下降太慢。学习率一般要随着训练进行衰减。衰减系数设0.1，0.3，0.5均可，衰减时机，可以是验证集准确率不再上升时，或固定训练多少个周期以后自动进行衰减。

权重，损失可视化

权重可视化更多的还是帮助人类以自己熟悉的方式来观察网络.

因为, 你是不可能边观察网络, 还边调参的. 你只是训练完成后(或者准确率到达一个阶段后), 才能可视化. 假如看到中间结果乱七八糟, 全黑全白, 这时候你直接看最后准确率就可以知道这网络没救了.

看到一个不满足平滑结果的图像,去考虑为什么模型训练不好呢? 是数据不好? 没有预处理? 网络结构问题? Learning Rate太大或者太小? 或者就是差了一个LRN层(之前我就遇到, 加个LRN就能出平滑的weights, 当然这其实和预处理有关)?

参数初始化

uniform均匀分布初始化： w = np.random.uniform(low=-scale, high=scale, size=[n_in,n_out])

Xavier初始法：

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

He kaiming初始化：

适用于ReLU：scale = np.sqrt(6/n)

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

svd初始化：对RNN有比较好的效果。

训是用normal初始化cnn的参数，最后acc只能到70%多，仅仅改成xavier，acc可以到98%。

最好的初始化是用已经训练好的模型进行初始化，这个也叫做迁移学习。

看paper（不止于最新的）

多看顶会的paper，但不能只喜欢漂亮的模型结构，瞧不起调参数的论文/实验报告，看论文时经常不看超参数设置等细节。

在自己没有太多资源实验的情况下，实验报告类文章就是业界良心！！！

最终武器

Ensemble

Ensemble是论文刷结果的终极核武器,深度学习中一般有以下几种方式

- 同样的参数,不同的初始化方式
- 不同的参数,通过cross-validation,选取最好的几组
- 同样的参数,模型训练的不同阶段，即不同迭代次数的模型。
- 不同的模型,进行线性融合. 例如RNN和传统模型.

奇招

一轮加正则，一轮不加正则，反复进行。

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。

理想情况下表现优越的模型一般长的高高瘦瘦（深，每层卷积核不多），但是为了便于我们训练，有时候不得不让他更矮壮一些。模型的调参除了学习速率和正则化这些比较耗时的工作外，还需要一些灵感，来确定模型的身材。

google的inception系列按它论文里说的永远无法复现。

rnn中在不考虑时间成本的情况下，batch size=1是一个很不错的。

优先调 learning rate!加 Dropout， 加 BN, 加Data Argument，调模型的层数和卷积核数量。