如何利用这篇 guide

深度学习的程序可能出错的地方有很多，这其中有一些错误发生频率比其他更高。通常我会从如下一些方面先行入手：

◆从简单并且得到广泛应用的网络开始，如 VGG，如果可以的话使用标准的损失函数。

◆暂时去掉所有的 trick，如数据增强(Data Augmentation)和正则化(regularization)。

◆如果是微调(finetuning)模型，再次检查数据的预处理，保证其和原始网络训练时一致。

◆检查输入数据是正确的。

◆从很少量的数据开始(2-20 样本)，使其过拟合，然后逐渐增加样本。

◆逐渐增加 trick，数据增强，正则化，新的损失函数，更复杂的网络等等。

如果上面的仍然不 work 的话，请按照下面的逐条实验。

1. 数据集问题

1）检查输入数据

检查输入给网络的数据是不是有意义的。比如，我不止一次将高度和宽度混淆（输入的图像是转置的）。有时因为数据预处理的错误输入的是全 0 图像。或者将同一 batch 的图像反复送入网络。所以，先打印出几个 batch 的输入保证正确性。

2）尝试随机输入

尝试输入随机数量的样本检查是否出现相同的错误。如果仍然出现同样的错误，说明网络本身有问题。这时尝试逐层调试。

3）检查 Data loader

不同的框架有不同的 data loader，如 tensorflow 的 tfrecords，keras 的 imagegenerator 等。数据本身可能没什么问题，但 data loader 可能有出错地方。先打印第一层的输入进行检查。

4）确保网络输入和输出是一致的

检查输入和预定的输出(label)是不是一致的，比如是否同时进行相同的 shuffle。

5）输入和输出的关系是否太过随机？

可能输入和输出的关系随机的成分更大，关系成分更小，如股票预测。这种情况下输入和输出不存在有效的关系。没有通用的方法检查这种情况，因为这决定于数据。

6）数据集标签噪音是否过大？

这种情况多发于自己从网上爬数据或者某些小型比赛。比如我从食物网站爬数据时经常有很多标签是不正确的。先手动检查一部分数据的标签是不是有很大噪声。

7）打乱(shuffle)数据

如果你的数据没有打乱而是有一定顺序的，这可能会给网络学习带来很负面的影响。确保你的数据经过了 shuffle。

8）减小数据不平衡

数据的不平衡指的是不同类别样本数量相差很大，如医疗数据就经常出现这种情况。数据的不平衡会导致损失函数的不平衡。由于篇幅限制，本文不再讨论如何平衡数据，可以参考 https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/。（可能需要科学上网哦）

9）训练样本是否足够

如果网络是从头开始训练而不是 fine-tuning，那么可能需要大量的数据。对于分类任务来说，通常一个类需要 1000 样本甚至更多。

10）确保一个 batch 中样本不是都属于同一类

这种情况可能发生于样本是有序的情况下，不过通过 shuffle 可以很容易解决这个问题。

11）减小 batch size

论文《On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima》指出非常大的 batch 可能会减弱网络的泛化性能。

Additional：使用标准数据集

当测试新的网络时首先使用标准的数据集如 mnist 而不是自己的数据，因为标准数据集标签噪声很小。如果网络在标准数据集上能够很好工作那么网络本身可能没有问题。

2. 数据增强和归一化

12）归一化数据

检查是否将输入归一化为 0 均值标准方差。

13）数据增强是否太过？

数据增强有正则化的效果 ,但是太多的数据增强和其他形式的正则化如 dorpout, l2 regularization 一起可能会导致欠拟合。

14）查看预训练网络的预处理方式

如果你使用的是预训练的网络，确保你用了相同的预处理方式。如图像像素范围在[0,1],[-1,1]还是[0,255]。

15）检查对于训练集、验证集、测试集的预处理

斯坦福 CS231 课程对于数据预处理有详细论述，参考 http://cs231n.github.io/neural-networks-2/#datapre

数据预处理必须只在训练集上计算，然后将参数用于验证集和测试集的处理。

3. 实现过程问题

16）尝试解决问题的简化版本

这将帮助找到问题的来源。如当需要解决的问题是找到物体的具体位置和类别，可以先尝试暂时忽略定位任务，只完成分类任务

17）寻找正确的损失“at chance”

参考自 CS231n，

http://cs231n.github.io/neural-networks-3/#sanitycheck。

"at chance"的意思是如果有 10 类样本，"at chance"会得到 10%的正确率，这时 Softmax 的损失应该为-ln(0.1)=2.302。可以对比自己网络的损失检查。

18）检查损失函数

如果损失函数是自己实现的不是标准损失函数，检查是否有错，可以进行单独的输入测试。

19）检查损失函数的输入

如果损失函数来自于深度学习框架如 tensorflow 自带，确保你的输入和规定的一致。如在 pytorch 中 NLLLoss 和 CrossEntropyLoss 的输入容易被混淆，前者需要输入 softmax 形式的输入，而后者不需要。

20）调整损失权重

如果你的损失函数是有几个小的损失函数组成，检查他们的权重是否合理。不同组合的权重可能会导致结果差异很大。

21）监控其他度量(metrics)

有时损失并不是最合适的预测，如果可以的话，监测其他度量如准确率。

22）检查自己写的 layer(custom layer)

如果自己写了框架中没有的层，一定要再三确保所写的 layer 正常工作。

23）检查 layer 或变量是否“forzen”

检查是否不小心将需要学习的 layer 或者变量“ frozen”了。

24）增加网络深度/宽度

可能你的网络出现了欠拟合的问题，尝试增大网络。如增加卷积层或增加全连接层神经元的数目。

25）检查隐层数据维度是否有误

如果输入是(64,64,64)这样的维度形式，很容易出现维度错误，如通道维度和宽度互换。这种情况下使用特殊的数字进行检查，如不同的维度使用不同的数字。

26）检查梯度

如果有些层是自己手动写的梯度传播公式，检查是否实现有误。（有些框架中自己实现的 layer 是要自己写梯度）。参考 CS231n 的梯度检查说明 http://cs231n.github.io/neural-networks-3/#gradcheck

4. 训练问题

27）在非常小的数据集上训练

当数据集比较小时非常容易过拟合。这种情况下需要数据增强，过采样，或者使用外部数据等手段增加数据量。

28）检查权重初始化方式

如果不确定哪一种初始化方式对自己网络最有效时，推荐使用 Xavier 和 He 初始化。不合适的初始化可能会导致网络进入坏的局部最小值点，所以尝试不同的初始化方式。

29）改变超参数

超参数对网络影响很大，如果不确定哪种设置更好，可以采用 grid search。参考 http://scikit-learn.org/stable/modules/grid_search.html

30）减弱正则化

太多的正则可能导致网络能力不足，在适当情况下减弱如 dropout, batch norm, weight l2 regularzation 等。参考 Jeremy 的课程 http://course.fast.ai/

31）Give it time

可能你的网络需要更多的时间去训练才能产出有意义的预测。如果 loss 在稳定的下降，请给网络更多的时间训练再做决定。

32）从训练模式转换为测试模式

Batch Norm，Dropout 等层在测试和训练时操作不同，注意在网络设置时这些层参数是否根据训练和测试的不同有所改变。如 tensorflow 中 tf.contrib.layers 中的 batch norm 需要手动传入是否在训练的参数。

33）训练可视化

利用 tensorboard 和 crayon 等工具对 weights/bias/activations 等进行可视化，确保他们的值正常。比如 weights 更新的强度应该在大约 1e-3 左右。参考 https://deeplearning4j.org/visualization#usingui

34）尝试不同的优化器

即使不同的优化器也不应该使网络不work除非选用了特别差的参数。即使如此，不同的优化器也会带来不同的效果，选择合适的优化器不仅能带来好的输出，还能节约大量时间。在不清楚用何种优化器时推荐使用 Adam 或 SGD with momentum。

参考 Sebastian 的文章

http://ruder.io/optimizing-gradient-descent/

35）梯度消失/爆炸

检查梯度是否出现了很大或很小的情况。检查 layer 的激活值，Deeplearning4j 上有一个很好的指导：*a good standard deviation for the activations is on the order of 0.5 to 2.0. Significantly outside of this range may indicate vanishing or exploding activations*

36）增加/减少学习率

小的学习率会让网络收敛很慢。大的学习率可能会在训练末尾难以找到一个好的解。在你当前的学习率进行 10/0.1 倍调整。

37）克服 NaNs

出现 Nan 是训练网络尤其是 RNN 时一个非常大的问题。一些可能的帮助手段如下：

◆减小学习率，尤其是在前 100 iteration 就出现 Nan

◆Nan 可能发生在除 0 或者 log0 的情况下，检查是否有这些问题

◆检查网络每一层，看 Nan 在哪里先出现的

5. 总结

现有的经验总结大抵如此，调试网络时候经常像是在搞玄学，但这些经验也能给一些指导，希望对你的工作有所帮助。

延伸阅读：

[1] http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html
[2]