Caffe 入门心得+SRCNN 实现

一、caffe 环境的配置：

Ubuntu 下环境配置费了好大的工夫具体参考的教程太多了没法指定某个具体的教程，就把过程中的心得体会写出来，应该会有所帮助。

1. 首先，一定要开 Google 搜索，光靠百度你可能永远也解决不了问题，当然有谷歌只是配好环境的一个必要不充分条件。从一开始的教程，到后面遇到的各种报错，把错误放到 Google 里搜一下，大部分都有答案（英文／中文），当然个别没有对应的答案，要从类似错误中自己举一反三想办法解决。
2. 其次，建议以某个全套的教程为主线进行配置，不要装 cuda 看一个教程，装 opencv 再看另一个教程，装 caffe 再看另一个，这样很容易出问题。(同时这也体现了这个 main 教程挑选的重要性，建议从完整性、详细性、以及与自己的电脑配置类似（如显卡型号）程度等方面考虑。)这样做的目的在于，有些东西可能有不止一种方式安装（比如命令行安装、下载压缩包安装），不同方式可能略有区别，有些情况下需要设置某些参数，这就体现了整体考虑的重要性。比如我在开始安装的时候，发现 g++的版本太高，需要降低版本，这本来在某个完整教程中有提到的，但是我看的是另一个教程，没注意这个细节，所以当时我后来尝试修改这个版本的时候，本来已经配好的整个环境崩溃了，牵一发而动全身，最后没办法只好重装系统，从头开始。说多了都是泪。
3. 至于具体需要装什么 cuda、cudnn、opencv、caffe 等等教程会有，这里不多说，我强调的是，一定要注意版本问题，并不是版本越高越好，一定要适合自己的电脑，而且有时候教程里面给的云盘链接并不一定比官网下载的新版差，所以推荐找那种提供下载链接的教程。
4. 其实很多时候感觉中间没什么问题，一直到最后一步 caffe 的时候，make all 过不了，各种玄学问题，有可能是前面某一步没做好，比如我就是 opencv 出了问题，后来回去又重新换了另一种方式装 opencv，中间 make 各种报错到绝望，一个个解决，总遇到一个问题过不了，后来在 google 上找啊找，忘记了是加了一句 cuda_generation = kapler 还是什么，总之就好了。（玄学问题要用玄学解决-_-!）
5. 第二遍装的时候又遇到不同的问题[捂脸]，好像还是上次的地方出错，

尝试各种 Google，最后也忘了怎么做了一下就好了。

6. 其他的想不起来了，想起来再加吧

二、SRCNN paper 的实现

1. 文章来自香港中文大学

   http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html

   Learning a Deep Convolutional Network for Image Super-Resolution

   这个网站有 paper 和相关的模型文件和 Matlab 源码，我的任务是把这个当作入门，弄懂这篇文章，并用 python 自己实现。与原文的误差不能超过 0.05dB，甚至更好。

2. 对 solver 和 net 文件的理解。不是很难，搞清楚就好。（在早期花了很多时间在学习 conv、ReLu、pooling、FC、BP 等算法上，后来发现这些东西并不用掌握，只要在 caffe 中调用就行了[捂脸]，不过理解了这些底层实现的东西，也是会有潜移默化的帮助的。比如 coursera 上吴恩达（Andrew Ng）机器学习系列里面，讲到对卷积层的理解、以及各层参数的计算关系，在后面理解文章上有很大的帮助，再到后来按照自己的要求调整模型也有很大的帮助。）再推荐一个 Stanford 李飞飞等主讲的 CS231n: Convolutional Neural Networks for Visual Recognition 课程，我觉得也挺好的，推荐，给上链接：

   http://vision.stanford.edu/teaching/cs231n/index.html

3. 网站上提供的代码是 Matlab 生成 hdf5 文件，作为 caffe 的输入进行训练，训练好了之后用 matcaffe 生成 Matlab 可以用的.mat 文件用于 Matlab 提取 w、b 等参数进行模型搭建并计算出结果。由于我 matcaffe 没搞好，所以没法从 caffemodel 中提取出参数并计算，我曾用 python 尝试过把参数提取到变量中，可是发现用这些变量来恢复整个结构又是一个巨大的问题，后来发现并不用把参数提出来，直接在 caffe 中得到 net，然后 forward()就行，用一个 deploy 文件。

4. 就用它的模型进行训练，训练 70 万、80 万代基本上 bicubic 和 SRCNN 的差值就可以达到 paper 中的标准了，训练 500 万代结果更贴近 paper（个别张图片稍差 0.01 数量级），而且不论 SRCNN 还是 bicubic，计算结果与 paper 都相比偏小，这来源于计算 PSNR 时候的裁剪（或者 pad）。因为根据 paper 中的模型，最终出来的结果会缩小 12 个像素，所以你可以在我的 python 代码中看到，在计算 PSNR 之前又一个（6:-6，6:-6）的操作，就是为了统一。我试过，如果不裁掉这 12 像素，而是将输

出的结果外围补 12 像素（参数为'edge'），bicubic 的数值很容易达到 paper 的数值甚至更高（但是 SRCNN 的结果不太好），所以这个数值普遍比 paper 偏小的现象并不代表模型的优劣，而仅仅是计算方法的区别而已，完全可以通过裁剪、pad 等方式，寻找一个合适的算法。另我也试过，修改 net 模型，如 kernel、stride、pad 等，使得输入输出等大，这样最后一步就省去了裁剪或者 pad 的操作，而且显然，这样一来，最终的结果更好，bicubic 和 SRCNN 差值均高于第一种模型，且 bicubic 的值不再小于 paper，而是与 paper 相同，甚至略高。

5. 要说明的是当你修改了 net 模型之后，要同时修改，deploy 文件，同时修改 python 中最终的计算代码，以及要修改生成 hdf5 文件的 Matlab 代码。
6. 关于 caffe 输入文件的类型，这个 paper 提供的代码是用 hdf5（虽然一开始搞不懂这种文件类型，可能便携吧。直接读图片训练比较慢，也有人用 lmdb 文件，caffe 有内置的生成这种文件的代码）。一开始他的 hdf5 文件中 data 是 33x33 的，label 是 21x21 的，到后来才知道这个相差 12，正是 net 结构输入和输出的像素差。所以当你修改 net 结构之后，这里的 label 的 size 应该和 data 一致。
7. 另外，deploy 的 input 的 size（dim）一定要跟前面生成 hdf5 训练文件的 data 的 size 一致。
8. 我发现当我直接用图片生成图片原始尺寸大小的 data 和 label 的时候（data 和 label 的差值当然要跟之前保持一致），比如 256x256 之类的，直接作为 deploy 的输入 data，模型照样可以用。所以，我想了想，一开始原始材料中的 data 是 33x33 可能是为了分成小块提高训练的效率或者速度吧（？）。举个例子，比如尽管看上去只有几十张张训练图片，但是每张图片被分为好多块 33x33 的块（Matlab 生成 hdf5 的代码中是以 14 为 stride），所以从这个意义上将，其实真正的训练图片数量应该以小块的数量计算，尽管小块的尺寸很小。每一个小块在训练的时候，是以 net 文件的 kernel 和 stride 等参数训练的。所以在最终测试的时候，输入的这个 hdf5 的 data 的 size 其实是不是 33x33 没关系的，他都会用 net 模型，以 net 的 kernel、stride 等参数进行计算。只不过，这个最终测试的输入的 data 数据类型要和训练的时候一致。且 deploy 的 dim 依然和训练时候保持一致，如依旧为 33x33（这个我其实不太懂这个参数有什么用，因为最终测试的 input 的 data 的 size 可以随便取，

总之这个根 net 保持一致就行了）。

侯碧潭

2017/12/10 下午

于新图