# CNN based Image Super Resolution实现

[shateng@baidu.com](mailto:shateng@baidu.com)

2017/2/6

1. Reference

VDSR: Accurate Image Super-Resolution Using Very Deep Convolutional Networks, CVPR2016

SRCNN:Image Super-Resolution Using Deep Convolutional Networks, ECCV2015

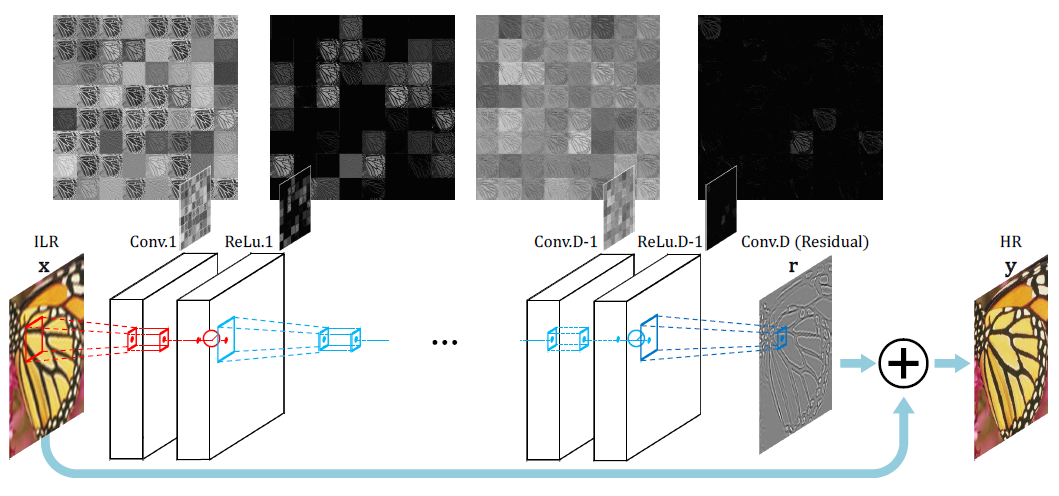
鉴于VDSR比SRCNN更好更快，只实现VDSR算法

1. Dependency

Paddle(local and paddle\_platform)：deeplearning.baidu.com

PIL、Numpy

1. VDSR原理



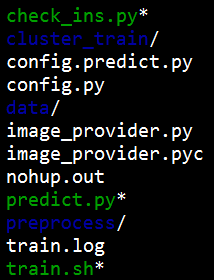
输入：hr图像缩小后又经bicubic放大的图像（hr图像作为label）；

输出：残差图像＋bicubic放大的图像

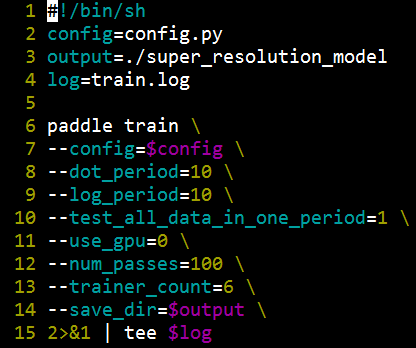
cost：输出图像与label图像的平方误差

实际上因为人眼对亮度信号最敏感所以只计算ycbcr格式的亮度通道数据即可

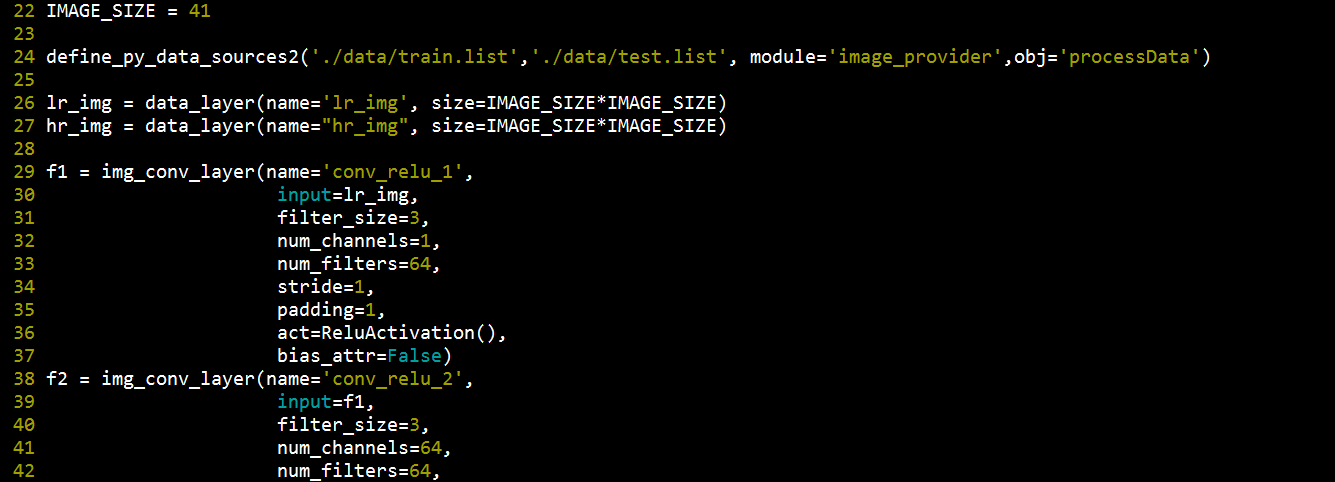
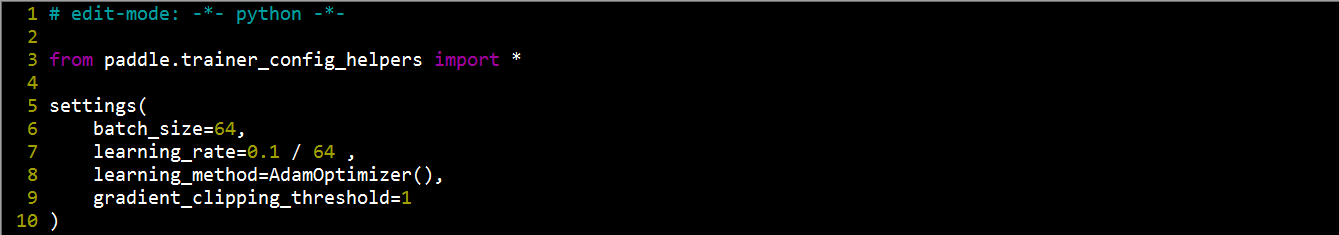
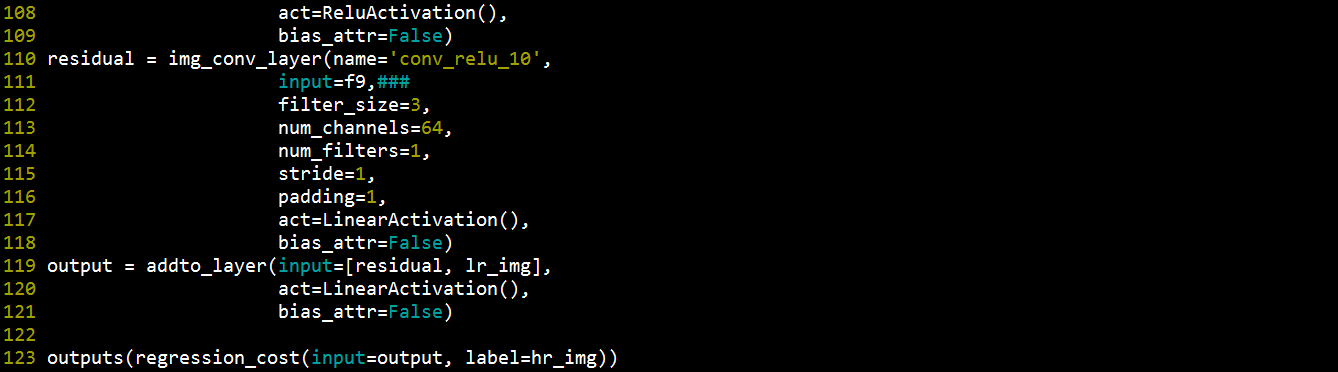
1. 基于paddle的实现



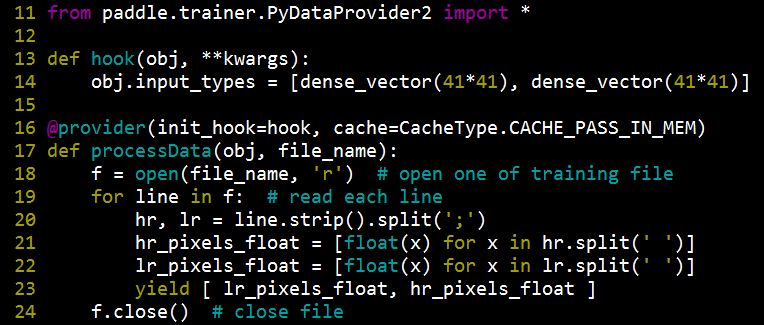
1. 训练数据准备（preprocess）
   1. 获取91张高分辨率图像集
   2. 通过旋转翻转等方式增强数据集：
   3. 将高分辨率的大图切成41\*41的高低分辨率的sub-image pair
   4. 生成train和test set的pair list, 并指定数据集内图片的数量
   5. 根据train和test的pair list生成对应的图像像素文本（提取Y通道并转成0~1之间的浮点数向量并以空格分隔，hr和lr的像素向量之间分号分隔）文件以及.list文件
2. 本地单机多线程调试（train.sh -> config.py -> image\_provider.py）
   1. Train.sh



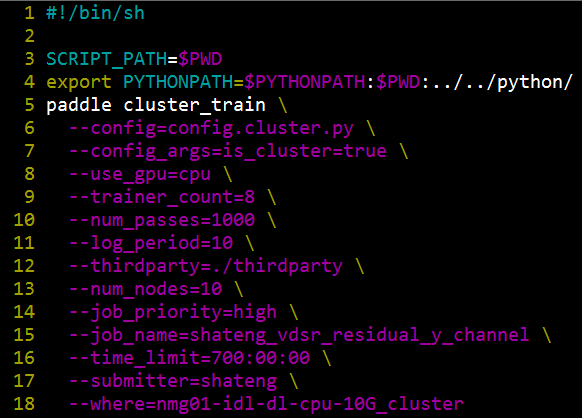
* 1. config.py

… …

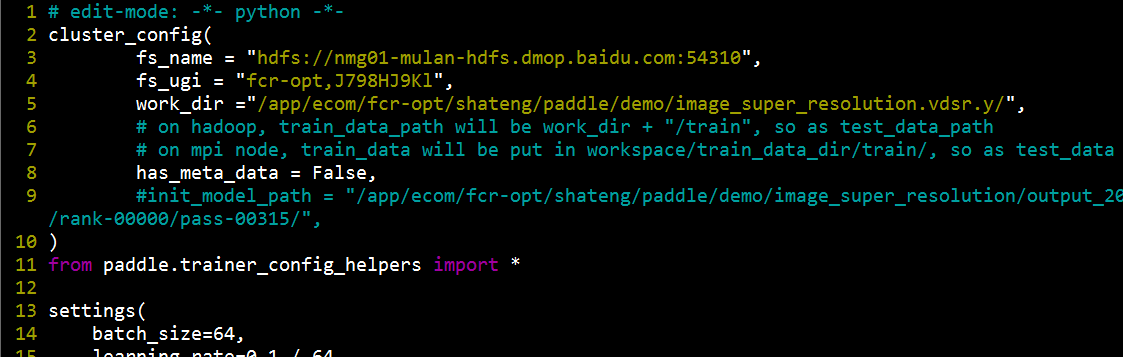
* 1. image\_provider.py



1. 集群训练（cluster\_train）
   1. 上传训练数据到集群的train、test目录（paddle\_platform脚本会自动下载到MPI节点的workspace/train\_data\_dir/train/和workspace/test\_data\_dir/test/目录），准备好train.list和test.list并随thirdparty分发到节点的workspace（provider根据这两个list读取数据）
   2. cluster.sh



* 1. config.cluster.py（在local config的基础上增加集群配置）



1. 参数及fine tunning

3\*3的kernel，10-20层img\_conv\_layer with relu，batch\_size=64，初始lr=0.1/batch\_size，每10轮lr=lr\*0.1，gradient\_clipping\_threshold=1，AdamOptimizer or momentum

1. 训练耗时：10000个sub\_image pair，10个节点（CPU machine）每个节点8个线程，10轮左右即可达到较好的效果，每轮耗时8分钟，即一两个小时即可，远胜SRCNN的长达几周的训练时间
2. 测试重建效果（predict.py）

swig\_paddle.initPaddle("--use\_gpu=0")

network = swig\_paddle.GradientMachine.createFromConfigProto(conf.model\_config)

network.loadParameters(model\_path)

converter = util.DataProviderWrapperConverter(False, [DenseSlot(image\_size\*image\_size), DenseSlot(image\_size\*image\_size)])

input = converter(ins)

print network.forwardTest(input)

output = network.getLayerOutputs('output')

output\_list = output['output'][0].tolist()

将output\_list（Y通道）与bicubic图像的色彩通道叠加并转为图像即可。

1. 一些经验和踩过的坑
   1. 验证你的数据的正确性：对输出的特征向量进行样本复原，比如对训练数据里的像素向量复原成图像，来检查你的数据预处理是否符合预期；检查数据的分布是否都在预期的区间内等等
   2. 提前计算你的网络的loss大概应该在多少，如果不符合预期就需要立即检查数据和网络的正确性，比如VDSR的loss是重建图像和原始图像的平方误差，那就能大概估计到loss最坏应该不大于41\*41\*0.1^2=16.81，如果经过几轮迭代还大于这个数，那就是哪里有问题了（分类问题同理）
   3. 先在较小规模的训练数据上（几十或几百条ins）进行单机训练，确保网络能够收敛的很好或loss能接近0
   4. 监控训练和测试集的loss或者准确率，当训练集的loss停止不动时很大概率是学习率太高，让更新算法在不收敛的地方来回震荡，尝试调低学习率
   5. 使用Y通道代替RGB通道来进行训练：特征数量降低3倍，使模型更容易收敛到较好的结果上，且对图像质量无影响
   6. numpy的bug导致提取的y通道错误

解决：numpy的array在加载ycbcr格式的Image时有bug，可以这样代替numpy.array(image)：

ycbcr = image.convert('YCbCr')

np\_array = numpy.ndarray((image.size[1], image.size[0], 3), 'u1', ycbcr.tostring())

np\_array.flags.writeable = True

np\_array\_y = np\_array[:,:,0]

* 1. paddle的script与版本兼容问题：paddle会在Python的site-packages下安装paddle对应的conf解析等脚本以及swig\_paddle等lib，更新较频繁，容易引起解析错误

解决：尽量将单机的paddle升级安装为最新版（有一键编译工具，可pull最新版本进行编译）；同时使用最新的platform receiver

1. 一些结果

（插值算法放大两倍）（VDSR放大两倍）

1. 使用VDSR模型

wget [work@cq01-pl25.cq01.baidu.com:~/shateng/paddle/image\_super\_resolution\_vdsr.y\_channel/try/](mailto:work@cq01-pl25.cq01.baidu.com:~/shateng/paddle/image_super_resolution_vdsr.y_channel/try/) .

./predict.py