PSNR结果偏差的一些原因

1. 来自预处理的偏差

Crop方面，python中处理中采用的是PIL.Image.CenterCrop，matlab里采用的是modcrop.m，即左上角剪裁，给bicubic\_psnr的计算带来了偏差；其次，我怀疑torchvision.transforms里的插值方法实现和matlab中并不相同（也有可能是数据格式的原因？见下文）我在多次尝试手动实现modcrop之后仍然只得到了相近的结果（Set5 bicubicPSNR结果相差大约0.3），但也不相同，这一点我到现在也没有解决，只有这种解释了。

Downsample方面，由于LR的生成中涉及到了下采样方法的问题。由于先前没有得到bicubic的benchmark值，paper中也没有提到如何blur和下采样，便以为是采样方法不同，于是尝试了PIL.Image中所有内置的方法。一个有趣的结论是，Image.LANCZOS相对来说保存了最多的高频信息。

另外，由于本文的算法中采用了sub\_image的概念，即crop成小image然后分别丢进去训练，便产生了把LR也裁成小image分别bicubic然后平均的想法，得到的平均值略高于直接bicubic的结果（这一组实验本身并没有什么意义，只是当时复现不出bicubic结果有些魔怔了。。）于是便思考对testimage是不是也可以先crop，SR，再拼起来组合成prediction，是否会得到更好的结果。。这一组实验没有进行完，待续。

1. PSNR的计算偏差

首先是数据格式上。Image处理中包括Resize在内的步骤涉及到了对像素值的计算，这就需要考虑到像素点的数据存储类型（int8、float32、double）。在PIL.Image中似乎没有给出改变存储类型的接口，意味着严格来说，resize的工作需要在torch.ToTensor之后再进行才能计算出准确结果。这一点在matlab中体现的较为清晰，数据格式在存储的时候都清楚的给出了，也有调整的接口。我在操作之前需要进行int8转double的操作这耽误了不少时间，曾经误判了问题来源。

PSNR的计算方法参考了众多代码，得到正确复现结果的一般都引入了shave\_border的操作，即对复原图片进行[scale:height-scale, scale:width-scale]的剪裁，相当于仅在中间部分进行复原的value计算。我并没有想清楚原因，我觉得并不必要。猜测剪裁的像素与是与外围padding计算相关，予舍去，还待考量。这一点也是自己手写的时候没有写进去、读了很多人代码之后才添上的部分，不添加这部分shave\_border就不能得到benchmark psnr的结果，还要再思考。如果不考虑shave\_border的话，psnr的计算直接可以用Tensor的nn.MSELoss()计算，公式改成10\*log10(1/mse.item())，经验证结果是正确的。

总结：

在计算psnr的debug过程中，最大的问题来源于对数据本身的理解。两个方面，一是关于数据维度的含义。由于代码经验较少，对基本函数和模块不够熟悉，导致自己并不能很清楚的理解数据在函数调用下的维度变化，以及Image通过不同方式load或者转化后，各个存储格式的channel到底如何对应，这一点上耽误了不少时间。其次是框架，从mat(int8, double)到numpy.ndarray(float32)，到torch.tensor、torch.cuda.tensor的相互转化，都带来过一些麻烦。但是经过这此学习和处理之后就熟悉很多了，再解决的时候应该就会比较熟练的面对了。