PSNR 实现：对于彩色图像，首先将图像由 RGB 色彩空间转换到 YCbCr 色彩空间，然后只用 Y 通道来计算PSNR。直接根据所给公式计算即可。对于灰度图像，则不需要转换色彩空间。

2.

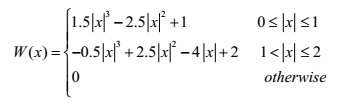
SSIM 实现：同PSNR一样，对于灰度图片，不需要转换色彩空间，而对于彩色图像，则必须先将图像由 RGB 色彩空间转换到 YCbCr 色彩空间，然后只用 Y 通道来计算SSIM。之后，将输入的两张图像作下采样处理，然后作高斯滤波。最后再计算SSIM公式的各个变量的值即可。（包括两张图像的均值，方差，协方差）。注意：这里的均值，方差，协方差都是通过对两张图像的各个局部区域计算来的，因而均值，方差，协方差都是矩阵来的。

3.

双三次插值算法：双三次插值是一种比双线性插值更加复杂的插值方式，它能创造出比[双线性插值](https://baike.baidu.com/item/%E5%8F%8C%E7%BA%BF%E6%80%A7%E6%8F%92%E5%80%BC" \t "https://baike.baidu.com/item/%E5%8F%8C%E4%B8%89%E6%AC%A1%E6%8F%92%E5%80%BC/_blank)更平滑的图像边缘。在这种方法中，插值图像在点 (x,y) 的值可以通过缩放比例确定原图相应位置最近的十六个采样点的[加权平均](https://baike.baidu.com/item/%E5%8A%A0%E6%9D%83%E5%B9%B3%E5%9D%87" \t "https://baike.baidu.com/item/%E5%8F%8C%E4%B8%89%E6%AC%A1%E6%8F%92%E5%80%BC/_blank)得到。

具体实现思路：首先，为了处理边界问题，将原图像各个方向拓展两行两列，总共拓展四行四列；接着，对插值图像的各个位置赋值即可。插值图像中某点（x，

y）通过比例关系对应到扩展后的图像某点（i，j），通过计算点（i，j）附近16个点的加权均值，最后赋值给点（x，y）。遍历插值图像所有点，逐一赋值即可。公式中的16个邻近点的权重由以下公式计算得出。



1. 与论文三中表格数据对比，从均值来看，发现PSNR值相差了1.2左右。

实验数据表格如下：

表格 1：不同算法在 Set14 测试集上的 PSNR, SSIM 以及运行时间

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Set14  images | 双三次插值算法 | | 基本任务超分辨率算法 | | | 高级任务超分辨率算法 | | |
| PSNR | SSIM | PSNR | SSIM | Time | PSNR | SSIM | Time |
| baboon | 21.4105 | 0.7904 | 22.6389 | 0.8432 | 13.995842 |  |  |  |
| barbara | 24.7206 | 0.8387 | 26.4085 | 0.8955 | 22.363543 |  |  |  |
| bridge | 23.3556 | 0.8296 | 24.4092 | 0.8944 | 10.590791 |  |  |  |
| coastguard | 24.7227 | 0.6106 | 24.5607 | 0.5676 | 6.293973 |  |  |  |
| comic | 22.6129 | 0.7221 | 20.7062 | 0.6119 | 5.649571 |  |  |  |
| face | 31.6066 | 0.7763 | 32.9226 | 0.8167 | 4.755864 |  |  |  |
| flowers | 26.7457 | 0.8090 | 27.9819 | 0.8302 | 10.689878 |  |  |  |
| foreman | 27.4872 | 0.9007 | 24.8293 | 0.8656 | 6.439004 |  |  |  |
| lenna | 31.3682 | 0.9406 | 32.3203 | 0.9641 | 15.481100 |  |  |  |
| man | 26.2202 | 0.8827 | 27.3371 | 0.9281 | 13.924974 |  |  |  |
| monarch | 29.1779 | 0.9669 | 28.9187 | 0.9774 | 22.390027 |  |  |  |
| pepper | 29.4521 | 0.9520 | 30.0326 | 0.9737 | 16.139904 |  |  |  |
| ppt3 | 22.9341 | 0.9301 | 21.8597 | 0.9215 | 20.144164 |  |  |  |
| zebra | 26.5305 | 0.9064 | 22.4101 | 0.8127 | 12.249681 |  |  |  |
| average | 26.3103 | 0.8469 | 26.2383 | 0.8502 | 12.936308 |  |  |  |

超分辨率算法内容：这是一种基于机器学习的超分辨率算法，其中应用到的主要算法就是K-means 聚类算法。K-means算法是输入[聚类](https://baike.baidu.com/item/%E8%81%9A%E7%B1%BB" \t "https://baike.baidu.com/item/K-MEANS%E7%AE%97%E6%B3%95/_blank)个数k，以及包含大量[数据对象](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AF%B9%E8%B1%A1" \t "https://baike.baidu.com/item/K-MEANS%E7%AE%97%E6%B3%95/_blank)的数据库，输出满足方差最小标准k个聚类的一种算法。k-means 算法接受输入量 聚类个数k ；然后将所有[数据对象](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AF%B9%E8%B1%A1" \t "https://baike.baidu.com/item/K-MEANS%E7%AE%97%E6%B3%95/_blank)划分为 k个[聚类](https://baike.baidu.com/item/%E8%81%9A%E7%B1%BB" \t "https://baike.baidu.com/item/K-MEANS%E7%AE%97%E6%B3%95/_blank)以使得所获得的聚类满足：同一聚类中的对象相似度较高；而不同聚类中的对象相似度较低。K-means 算法基本步骤：

（1）从 n个[数据对象](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AF%B9%E8%B1%A1" \t "https://baike.baidu.com/item/K-MEANS%E7%AE%97%E6%B3%95/_blank)任意选择 k 个对象作为初始聚类中心；

（2）根据每个[聚类](https://baike.baidu.com/item/%E8%81%9A%E7%B1%BB" \t "https://baike.baidu.com/item/K-MEANS%E7%AE%97%E6%B3%95/_blank)对象的[均值](https://baike.baidu.com/item/%E5%9D%87%E5%80%BC" \t "https://baike.baidu.com/item/K-MEANS%E7%AE%97%E6%B3%95/_blank)（中心对象），计算每个对象与这些中心对象的距离；并根据最小距离重新对相应对象进行划分；

（3）重新计算每个（有变化）聚类的均值（中心对象）；

（4）计算标准测度函数，当满足一定条件，如函数收敛时，则算法终止；如果条件不满足则回到步骤（2）。

实现思路：首先，对训练集HR图像做双三次插值处理，生成LR图像，并对LR图像进行切割，取7\*7 LR小块，维度为45\*1；（去掉了四个角）。接着让LR 小块减去各自小块的均值，得到LR 特征小块。然后，从大量的LR 特征块中选取200000的数目拿去作K-means聚类，这里，聚类数目我取的是512。聚类完成后可以得到一个描述各个类质心位置的矩阵，应用这个矩阵，另取大量LR特征小块，以及对应的HR特征小块，计算得到各个类的函数变换矩阵C。到这里，训练完毕。接着是测试部分。对测试图像作双三次插值处理后得到LR图像，接着对LR图像切割，取特征，得到一幅测试图像的所有LR特征块。然后，对于每个LR 小块，通过计算它与每个类别的距离，得到它所从属的类别 j；根据公式 HR 块 =  \* LR 块；计算出它对应的高分辨率patch ，即HR 块。遍历一幅图像所有LR 小块，得到对应的所有HR 小块。即最终获得了高分辨率图像。

实验结果分析：从表格可以看出，我的基本超分辨率算法得到的PSNR值和SSIM值与双三次插值结果相比，从均值来看结果数据差不多，相差0.1左右。有的图像PSNR值和SSIM值高过双三次插值结果，也有部分图像低于双三次插值结果。

我的看法是应该是由于自己在聚类时样本数目选取200000，太少，以及类数512，也偏小。

1. means算法主要缺点：首先，在 K-means 算法中 K 是事先给定的，这个 K 值的选定是非常难以估计的。很多时候，事先并不知道给定的数据集应该分成多少个类别才最合适；其次，在 K-means 算法中，首先需要根据初始聚类中心来确定一个初始划分，然后对初始划分进行优化。这个初始聚类中心的选择对聚类结果有较大的影响，一旦初始值选择的不好，可能无法得到有效的聚类结果；最后，该算法需要不断地进行样本分类调整，不断地计算调整后的新的聚类中心，因此当数据量非常大时，算法的时间开销是非常大的。

针对K-means算法的改进：

1.k-modes 算法：实现对离散数据的快速聚类，保留了k-means算法的效率同时将k-means的应用范围扩大到离散数据。

2.k-Prototype算法：可以对离散与数值属性两种混合的数据进行聚类，在k-prototype中定义了一个对数值与离散属性都计算的相异性度量标准。