实验报告

2

2.1 基本任务

1.

2. SSIM 实现思路：

1）先对两个输入图像进行判断，是否为 RGB 彩色图像，如果为彩色图像，则转为 YUV 通道，并取 Y 通道进行对比。

```
if (ndims(input_imgx) > 2)
    imgx = rgb2ycbcr(input_imgx);
    imgx = double(imgx(:,:,1));
    imgy = rgb2ycbcr(input_imgy);
    imgy = double(imgy(:,:,1));
else
    imgx = double(input_imgx);
    imgy = double(input_imgy);
end
```

2）求出两个图像的均值矩阵：先生成一个 size 为 11*11、sigma 为 1.5 高斯核，并用原图与该高斯核做卷积得出。

3）根据方差公式：

$$s^2 = \frac{\sum (X_i - \mu)^2}{n} = \frac{\sum X_i^2}{n} - \mu^2$$

可知，先对图像各像素点进行平方运算，然后与步骤相同，对其做高斯均值滤波，最后减去均值矩阵的平方得到方差矩阵。

```
Sigmax = filter2d(G, imgx.*imgx) - (Ux.*Ux);
Sigmay = filter2d(G, imgy.*imgy) - Uy.*Uy;
Sigmaxy = filter2d(G, imgx.*imgy)-Ux.*Uy;
```

4）最后根据公式：

$$SSIM(X,Y)=\frac{(2\mu_X\mu_Y+c_1)(2\sigma_{XY}+c_2)}{(\mu_X^2+\mu_Y^2+c_1)(\sigma_X^2+\sigma_Y^2+c_2)}$$

对上述的矩阵进行运算并得到 SSIM 矩阵，最后对其进行做均值，即得出 SSIM 值。

3. 双三次插值算法

双三次插值算法与之前的双线性插值相似，但不同是双三次插值利用了原图像像素点附近 16 个点来估计目标像素点的灰度值，而非双线性插值的 4 个点。

(I-1,J-1)			(I+2,J+2)
	(I,J)		
(I+2,J+2)			(I+2,J+2)

1）对于目标图像的每个像素(x,y)，横纵坐标分别乘以对应的放缩比 row_rate, col_rate 得到对应的原图像位置(i,j)以及余数 m,n

2）取对应图像像素点附近 16 个坐标点，并根据公式：

$$p(x,y)=\sum_{i=0}^{3}\sum_{j=0}^{3}p(x_i,y_j)W(x-x_i)W(y-y_j)\,,$$

其中权重函数为：

$$W(x)=\begin{cases}1.5|x|^3-2.5|x|^2+1 & 0\le|x|\le 1\\ -0.5|x|^3+2.5|x|^2-4|x|+2 & 1<|x|\le 2\\ 0 & otherwise\end{cases}$$

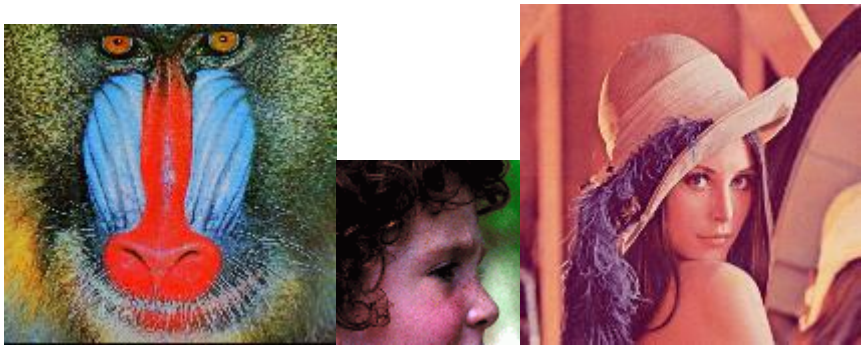
代入后计算 16 个点的总和。（对于超过图像边界的点，我取边界值代替）

```
%越界处理
if (tempI < 1)
    tempI = 1;
end
if (tempJ < 1)
    tempJ = 1;
end
if (tempI > src_col)
    tempI = src_col;
end
if (tempJ > src_row)
    tempJ = src_row;
end
%disp(weight(x-tempI));
sum = sum + img(tempJ,tempI)*weight(u-m)*weight(v-n);
```

3）最后将乘积和赋给目标像素点即可。

4. 双三插值进行超分辨率

A）以下为部分图像：



B）

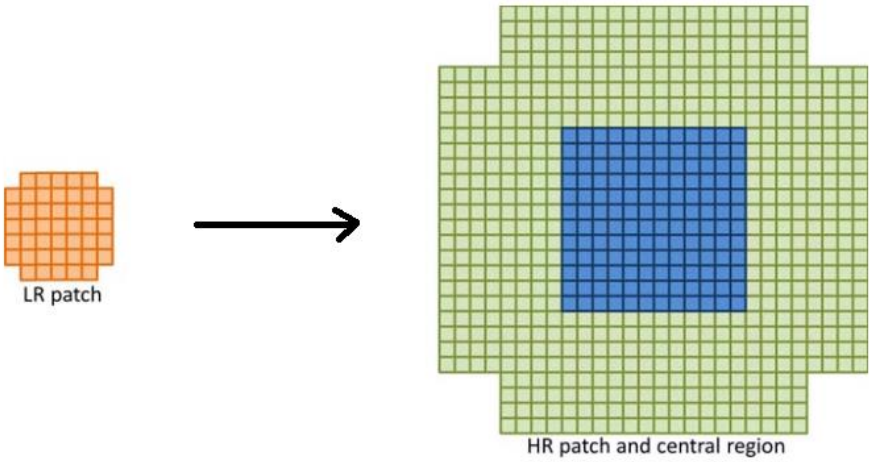
C）表格在内容 6 中

5. 基本任务超分辨率算法

根据论文内容，我大致将这个算法分为三阶段：一、聚类阶段；二、训练系数函数阶段；三、测试阶段

一、聚类阶段

1）根据论文提示裁剪出 lr_patch 跟对应的 hr_patch 的中心区域：



由于电脑配置原因，这里我选用的样本为 200000 个 lr_patch，方差为 1.2。

首先，将原图像进行高斯模糊，用的其中高斯核的生成由公式可得：

$$G(u,v)=\frac{1}{2\pi\sigma^2}e^{-(u^2+v^2)/(2\sigma^2)}$$

然后使用双三次插值算法下采样得出 lr 图像。

接着，对 lr 图像的每个像素点（除去了 3 个像素点的边界）进行裁剪范围为 7*7 的 lr_patch 以列向量的形式存储在 lr_patch(i)中

此时，得到的 lr_patch 为一个 49*200000 的矩阵.

2）用所得 lr_patch 和 hr_patch 生成特征值 feature，lr_feature 为 lr_patch 去掉 4 个角落求均值得到 lr_patchmean，然后用去掉角落的 lr_patch 减去均值 lr_patchmean 得出.

3）对上述得出的 lr_feature 使用聚类函数 kmeans(X, K, ……)其中 X 为 lr_feature 的转置 lr_feature’，而 K 为类的个数。（这里我选择了 512 个类），最后存储聚类结果 C

```
options = statset('UseParallel',1,'MaxIter',400);
[idx, C] = kmeans(lr_featureselect', num_cluster, 'Display','Iter', 'Options', options);
```

二、训练系数矩阵阶段

1）首先裁剪 lr_patch 和 hr_patch，lr_patch 的裁剪与一阶段相同，而这里为了使得系数矩阵更为准确我选用了 2000000 个样本进行训练，hr-patch 的裁剪则是根据其对应 lr_patch 的中心点位置给出：

```
hr_phbegin = (r-2)*scale_rate+1;
hr_phend = (r+1)*scale_rate;
hr_pwbegin = (c-2)*scale_rate+1;
hr_pwend = (c+1)*scale_rate;
temp1 = hr_img(hr_phbegin:hr_phend,hr_pwbegin:hr_pwend);
hr_patch(:,i) = temp1(:);
```

2）对上述得到的 lr_patch 和 hr_patch 求出特征值 feature，lr 的特征值同一阶段，而 hr_feature 则是由 hr_patch 减去 lr_patch 的均值 lr_patchmean 得出。

3）对于每个 lr_feature(i)向量，求出与他最近的聚类中心点，用 idx(i)存储其最近聚类中心点的编号，然后用同一个类 i 的 hr_feature(idx == i)/lr_feature(idx == i)得出这个类的系数矩阵 Cmatrix(i)；最终得出的 Cmatrix 为 512*81*45 矩阵。

三、测试阶段

1）对每个测试图像先三次插值下采样为原来的 1/3，如果为彩色图像，则对 Y 通道进行操作 UV 通道进行三次插值下采样后再三次插值上采样。

2）将原图像得到 lr_img 扩展边界两个单位，使得 lr_patch3*3 的中心区域刚好贴到扩展前的边界，然后裁剪得出 lr_patch（裁剪方法与一二阶段相同）并得出 lr_feature（为了使得 lr_patch 的中心区域对应于 hr_patch 的中心区域）

3）对于每个 lr_feature 求出与其最近的类中心点，然后用这个类的系数矩阵 Cmatrix(i)乘以 lr_feature 向量，得出 hr_feature，进一步得到 hr_patch。

4）将得到的 hr_patch 覆盖到目标图像上，其位置同样为：

```
hr_phbegin = (r-2)*scale_rate+1;
hr_phend = (r+1)*scale_rate;
hr_pwbegin = (c-2)*scale_rate+1;
hr_pwend = (c+1)*scale_rate;
temp1 = hr_img(hr_phbegin:hr_phend,hr_pwbegin:hr_pwend);
hr_patch(:,i) = temp1(:);
```

(r,c)为 lr_patch 中心点分别往上往左偏移 2 个单位。在覆盖的同时用一个矩阵 patchnum_inimg 存储每个像素点覆盖的 hr_patch 的个数，每当某个像素点（i，j）被一个新的 hr_patch 覆盖时，patchnum_inimg（i，j）++。

5）最后用每个像素点的覆盖的总值除以覆盖的 patch 的个数，即得出目标图像的像素值。这时候如果是彩色图像，则将其余两通道合并，即得输出图像。

四、部分结果

五、结果分析

可以看到，在使用了 SF 的方法之后，所有图像的 PSNR 和 SSIM 均超过了双三次插值的结果，然而提高的幅度没有想象中高，个人认为有以下原因：

1）由于电脑配置限制，在选取训练聚类样本的时候，只选取了其中的 20W 样本，而且只是顺序选取没有做到随机选取，因此在训练集靠后位置的图像并没有被使用，导致某些例如白色黑色区域较多的图像的处理效果相对其他图像比较差。

2）在训练系数矩阵的时候，matlab 多次报出秩不足的警告，询问同学得知，这是由于即使我选取了 200w 样本进行系数矩阵的训练，但是仍有某些类的样本数目不足，从而出现该情况。对于这个问题，论文作者是通过每个均选取 1000 个样本来处理。

3）需要测试的图像尺寸不是 3 的倍数时，得出的结果图像右边和下边界部分会出现一条黑边，这是由于在下采样时，选用了想下取整的方法。这样导致 PSNR 和 SSIM 相对较低，而在后面我的代码中我对此进行修改，改为向上取整，这时会出现结果图像比原图像尺寸大，而此时在通过三次插值的方法，将结果图像缩小到与原图像相同的尺寸，这样的 PSNR 和 SSIM 相比直接裁掉黑色边界要高。

6. 不同算法在 Set14 测试集上的 PSNR，SSIM 以及运行时间

1.表格：

Set14 images	双三次插值算法		基本任务超分辨率算法		
	PSNR	SSIM	PSNR	SSIM	Time
baboon	21.584838	0.536932	23.184016	0.581106	13.61s
barbara	24.720607	0.754927	26.354871	0.766884	22.16s
bridge	23.355589	0.653365	24.699114	0.689932	10.77s
coastguard	24.928411	0.613697	26.09735	0.640036	5.66s
comic	22.584005	0.721810	23.943741	0.756060	5.35s
face	31.606577	0.776282	32.522949	0.803030	3.92s
flowers	26.740503	0.808405	27.927914	0.825512	10.87s
foreman	27.481564	0.900604	28.509095	0.907294	5.64s
lenna	31.369353	0.855804	32.079116	0.864628	15.83s
man	26.219385	0.749199	27.611360	0.775763	15.92s
monarch	29.115749	0.921523	30.132445	0.922149	22.41s
pepper	29.451132	0.861385	30.406476	0.869915	15.86s
ppt3	22.996068	0.890382	25.139693	0.910449	20.96s
zebra	26.587258	0.809800	28.127084	0.829955	13.76s
average	23.338646	0.775294	27.653401	0.795908	13.05s

2.程序运行结果：

1）任务四：

```
>> main_Task4
      name          PSNR             SSIM
baboon.bmp          21.584838          0.536932
barbara.bmp          24.720607          0.754927
bridge.bmp          23.355589          0.653365
coastguard.bmp          24.928411          0.613697
comic.bmp          22.584005          0.721810
face.bmp          31.606577          0.776282
flowers.bmp          26.740503          0.808405
foreman.bmp          27.481564          0.900604
lenna.bmp          31.369353          0.855804
man.bmp          26.219385          0.749199
monarch.bmp          29.115749          0.921523
pepper.bmp          29.451132          0.861385
ppt3.bmp          22.996068          0.890382
zebra.bmp          26.587258          0.809800
mean          26.338646          0.775294
```

2）任务五：

```
>> main_Task5
      name          PSNR             SSIM
baboon.bmp          23.184016             0.581106
时间已过 15.977741 秒。
barbara.bmp          26.354871             0.766884
时间已过 25.082667 秒。
bridge.bmp          24.699114             0.689932
时间已过 12.298336 秒。
coastguard.bmp          26.509735             0.640036
时间已过 6.415444 秒。
comic.bmp          23.943741             0.756060
时间已过 6.119114 秒。
face.bmp          32.522949             0.803030
时间已过 4.578812 秒。
flowers.bmp          27.927914             0.825512
时间已过 12.151284 秒。
foreman.bmp          28.509095             0.907294
时间已过 6.484518 秒。
lenna.bmp          32.079116             0.864628
时间已过 17.816964 秒。
man.bmp          27.611360             0.775763
时间已过 17.922412 秒。
monarch.bmp          30.132445             0.922149
时间已过 25.343105 秒。
pepper.bmp          30.406476             0.869915
时间已过 17.914538 秒。
ppt3.bmp          25.139693             0.910449
时间已过 24.007703 秒。
zebra.bmp          28.127084             0.829955
时间已过 15.683564 秒。
mean              27.653401               0.795908
```