图像视频实验二 基于内容的初步图像检索 实验报告

计 71 庞博予 2017011315

一、实验设计

本次实验要求根据图像的颜色分布统计信息作为图像的特征向量，将图像嵌入到高维空间中并据此计算距离，算出在高维空间中相邻的图像并检测该相邻关系是否反应了图像的聚类。

二、算法步骤

1．预处理

由于图像是以 jpg 格式存储，如果每次进行查询都要读取图像进行计算得到向量进行比较效率会十分低下，所以提前将所有图片按照统一的格式处理成相同维度的向量存储起来。

以基本的 16 维向量空间举例，将红和蓝两个通道的数值右移 7 位（整除 128），其取值范围为[0,2)，将绿通道的数值右移 6 位（整除 64），其取值范围为[0,4)，将映射后的每一个像素的颜色类别写为 index=R*8+G*2+B，根据该 index 统计每个 index 出现的次数，因为 index 取值范围为[0,16)，故为十六维向量。

上述操作由于都是可以对全体像素统一操作的，在代码中用 python 的 DataFrame 相关操作对一张图片的所有像素进行批量处理，计算完 index 后将其序列化并用 count 直接统计。

2．查询

虽然查询有两种方式一种是对单张图片查询其最接近的 30 张中的命中率，一种是对全体图片计算平均命中率，不过两种的基本原理是一样的；都是基于给定的某张图片的向量为基向量，将所有图片对应的向量以关于该向量的距离从小到大排序并取出前 31 个（要排除排名第一且距离为 0 的原图）。图片的名称中自带类别信息，可用于判断是否命中。

由于有不同的距离衡量方法，故查询过程要支持通过参数控制程序执行，首先是距离的定义方式，基础的包括欧氏距离、直方图交集、Bhattacharyya 方法。使用 numpy 的相关功能实现三种距离的计算即可。其中欧氏距离直观易懂，距离越小说明两个向量距离越相近；直方图交集则越大越代表两个直方图分布较为一致，相近度较高；而 Bhattacharyya 方法经过简化其实是等价于计算两个向量对应位置几何平均数之和，在所有维度的数值总和相同时该和越大表示两个向量分布越接近。

注意由于 python 的 opencv 库会在读取图片时默认数据是 8-bit unsigned int 类型（在上一次实验中已经吃了亏），在计算距离时要将数据类型转化为实数类型或大整数类型（欧氏距离和直方图交集均可用整型计算）。

对于全体的查询命令，将对每一个图片查询最相近的图片并统计命中率，将每张图片的命中率和平均命中率输出到结果中；而对单一图片（其实也是一个图片集合）的查询，则将每张图片的最相近图片的名称和距离也要输出到其对应文件中。

三、性能分析

1．结果分析

平均准确率	16 维	128 维
L2	29.58%	37.62%
HI	32.53%	44.86%
Bh	37.86%	47.09%

2．错误分析

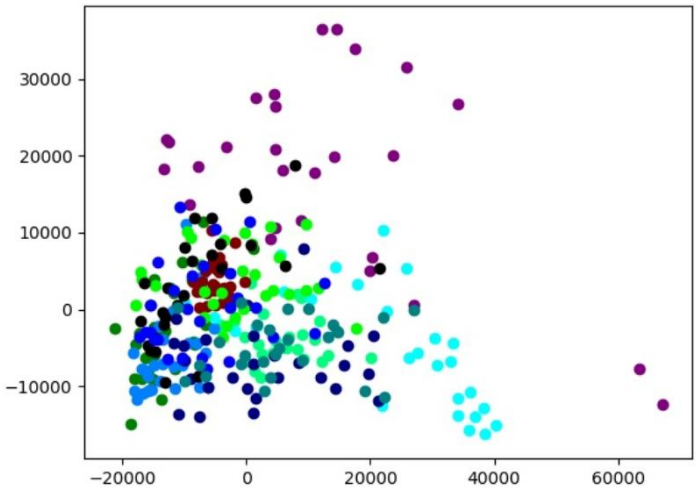
尽管尝试了多种手段查找最接近的图片，可以发现查找的正确率始终并不高，先手动观察一些错误率较高的查询样例并分析其错误原因：

例 1：beach/110.jpg 在 128 维欧氏距离条件下查询的前 30 接近图片中仅有一个是 beach 类别中的，而大量被判断为近似的图片则是 football 类别下的图片和 mountain 类别下的照片。


容易发现，这两个错误的近似图与原图在色调上接近，尤其是大片的蓝绿色区域是几张图共有的，这就使得这些图的距离更近。

例 2：与 elephant/511.jpg 最接近的图中大多数是 horses 类中的图，是因为这两类图均以草地丛林为背景，背景的匹配度极高，而图中的中体在算法中则被作为非重要信息几乎无视了。


为了进一步更直观观察产生错误的原因，尝试直接对描述图片的 128 维向量进行降维，将所有图片对应的向量降维成二维平面上的点并观察哪些本该属于同一类的点距离很远、本该属于不同类的点距离很近，并分析原因。



上图是根据 128 维向量生成的降维图，不同颜色的点代表不同类别。发现有些类别例如浅蓝色点和紫色点距离其他类别有一定距离，但是多数类别是相互穿插交织在一起，其区分度较低。

四、拓展功能

本次实验共从三个角度进行了拓展研究。

首先实验中尝试使用了在规定的三种距离计算方式之外的新的距离计算方式：交叉熵来衡量向量之间的距离，考虑到交叉熵是常用于判断两种概率分布的相似程度的方法，应用在这个场景中是有理论依据的。经过测试，有如下结果：

平均准确率	16 维	128 维
L2	29.58%	37.62%
HI	32.53%	44.86%
Bh	37.86%	47.09%
CE(交叉熵)	36.61%	45.11%

可以看到，交叉熵作为距离函数的性能是优于欧氏距离和直方图交集算法，稍逊于 Bhattacharyya 方法的。

而后是在 128 维向量空间的基础上考虑引入灰度维度，将像素的灰度值作为与红绿蓝三个通道同等地位的第四通道，将灰度值也整除 64 后映射到[0,4)的区间内，整体作为 512 维向量重复上述实验：

平均准确率	16 维	128 维	灰度 512 维
L2	29.58%	37.62%	36.65%
HI	32.53%	44.86%	44.93%
Bh	37.86%	47.09%	47.33%
CE(交叉熵)	36.61%	45.11%	45.55%

可以看到在增加了灰度维度后正确率有了小幅度上升，因为其实在原来的用 128 种取值表示一个像素的表示法中隐含了粗略的灰度区间，灰度维度只不过是将这个灰度区间精细化了，信息是有相当程度的冗余的，所以性能提升并不明显。

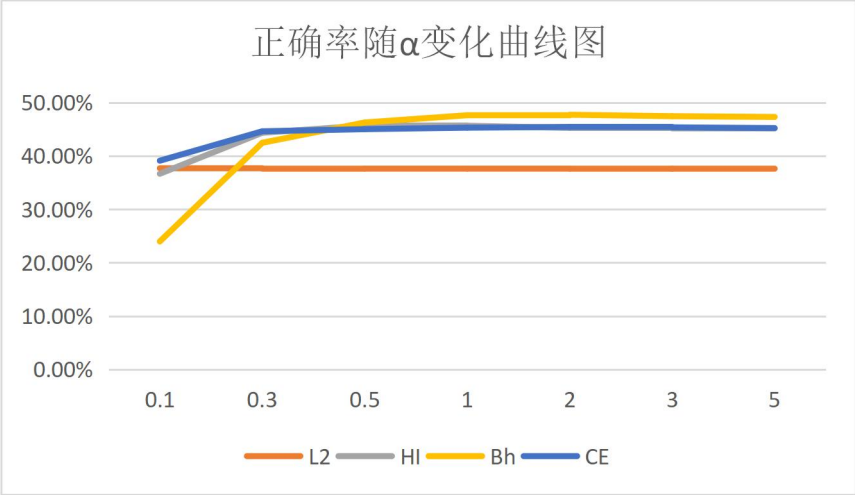
此外引入图像的区间信息，将图像按照区域划分成 8*8 的三通道块，对每个颜色通道的每个块计算该块内该颜色的平均值，得到 8*8*3=192 维附加信息，将其补充到原向量后得到 320 维向量，但要注意最好不要直接用该向量参与计算，因为该向量的前 128 维和后 192 维的取值范围是不同的，前 128 维是基于颜色空间的统计分布而后 192 维是区域颜色均值，如果直接使用距离函数进行测量距离会导致前后两部分的数据尺度不统一从而使结果不准确，故采用前后段各自求距离再按比例加和统一比较。

设前半段距离为 d1，后半段距离为 d2，则实际排序的距离为 d= α d1+d2。调整参数 α 使得正确率最高，并观察正确率随 α 的变化的变化规律。

下默认采取 128+192=320 维向量。

α	L2	HI	Bh	CE
0.1	37.70%	36.67%	23.98%	39.12%
0.3	37.62%	44.36%	42.48%	44.63%
0.5	37.63%	45.71%	46.28%	45.03%
1	37.63%	45.67%	47.64%	45.31%

2	37.62%	45.27%	47.73%	45.45%
3	37.62%	45.18%	47.46%	45.37%
5	37.62%	45.13%	47.32%	45.23%



显然对于不同的距离函数，α 会在不同的地方取到最值，但由于不修改 α 也并不能给正确率带来显著提升所以在此不做过多深入讨论。

除了上述三点之外，在 16、128 维度之外也曾尝试使用 1024 维（8:16:8）的颜色空间划分方法进行测试，但由于分的更细时本来可能处于同一组的相近颜色就被统计成了不同的组，少了很多模糊性和兼容性。同时运行速度大幅减低，正确率也出现大幅降低。

五、思考总结

本次实验中虽然从各个角度实现并尝试对这种基于颜色的图片检索算法进行改进但是算法准确率仍然堪忧，前 30 结果中平均只有不到 50%的结果是同类型图片（其中还有大多数高正确率检索是足球类中的检索，其他类检索正确率更低）。

究其原因，可能是图像的颜色本身的表达力就很有限，单纯从颜色联合分布角度入手更是忽视了大量的图片细节信息而只关注偏向于大局的信息。这样就很容易导致无法通过细节区分图片或者将氛围和底色相近但内容不同的图片归为一类。

想要解决这个问题，可以考虑捕捉图形的一些细节信息，最基本的比如边缘信息。例如对图像进行边缘提取后再通过卷积使得图像处在图片中心，再把这个居中边缘图作为向量检索接近的图片。当然，也可以直接用 CNN 训练自编码器将图片压缩成低维向量后观察这些图像向量的聚类情况，决定是否可以用自编码向量作为图片的检索标签。