浙江大学

课程名称：数字媒体资源管理

姓　　名：刘佳润

学　　院：计算机科学与技术学院

专　　业：数字媒体技术

学　　号：3180105640

指导教师：张宏鑫

2020　年　　11 月　　2　日

浙江大学实验报告

课程名称：数字媒体资源管理 实验类型：综合

实验项目名称：作业 5：图像相似度计算与图像检索方法探究

学生姓名：刘佳润 专业：数字媒体技术 学号：3180105640

同组学生姓名： 指导老师：张宏鑫

实验地点： 实验日期：2020 年 10 月 29 日

一、 实验目的和要求

基于一张样板图片，对九张其他图像进行相似度的计算，得到“最相似”的一张图片。尝试多种算法，并对图像检索方法进行探索。

要求：基于 PIL 库或者 OpenCV

二、 实验内容和原理

1. 颜色矩原理：

颜色矩是一种基于图像色彩通道数值的统计学概念。我们通常使用一张图的一阶中心距（均值）、二阶中心距（方差）、三阶中心距来描述。计算公式如下：

First order moment (mean) $$\mu_i = \frac{1}{n}\sum_{j=1}^{n} I_{ij}$$

Second order moment (variance) $$\sigma_i^2 = \frac{1}{n}\sum_{j=1}^{n} (I_{ij} - \mu_i)^2$$

Third order moment (skewness) $$s_i^3 = \frac{1}{n}\sum_{j=1}^{n} (I_{ij} - \mu_i)^3$$

我们计算两个图片这两个特征向量的余弦距离，值越大说明相似度越高。

2.结构相似性（SSIM）原理：

假设我们输入的两张图像分别是 x 和 y，那么

$$SSIM(x,y)=[l(x,y)]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{\gamma}---(1)$$

式 1 是 SSIM 的数学定义，其中：

$$l(x,y)=\frac{2\mu_x\mu_y+c_1}{\mu_x^2+\mu_y^2+c_1},$$

l(x,y)是亮度比较；

$$c(x,y)=\frac{2\sigma_{xy}+c_2}{\sigma_x^2+\sigma_y^2+c_2},$$

c(x,y)是对比度比较；

$$s(x,y)=\frac{\sigma_{xy}+c_3}{\sigma_x\sigma_y+c_3}$$

s(x,y)是结构比较。μx 和 μy 分别代表 x,y 的平均值，σ xσx 和 σ yσy 分别代表 x,y 的标准差。σxy 代表 x 和 y 的协方差。而 $c1$， $c2$， $c3$ 分别为常数，避免分母为 0 带来的系统错误。其中 $c_1=(k_1L)^2$， $c_2=(k_2L)^2$。L 是像素值的动态范围。在实际工程计算中，我们一般设定 $\alpha=\beta=\gamma=1$，$k_1=0.01, k_2=0.03$ 以及 $c3=c2/2$，可以将 SSIM 简化为下：

$$SSIM(x,y)=\frac{(2\mu_x\mu_y+c_1)(\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$$

SSIM 是一个 0 到 1 之间的数，越大表示输出图像和无失真图像的差距越小，即图像质量越好。当两幅图像一模一样时，SSIM=1。

三、 实验器材

开发工具：Python 3.7　Flask　opencv-python 库　numpy 库

四、 实验步骤

实验依托于强大的 numpy 数学库。颜色通道的提取通过 opencv 的函数 cvtColor 完成：

```
# Convert BGR to HSV colorspace
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
# Split the channels - h,s,v
h, s, v = cv2.split(hsv)
```

均值、方差与三阶中心距的计算均可以利用 numpy 库的函数实现：

```
color_feature = []
# N = h.shape[0] * h.shape[1]
# The first central moment - average
h_mean = np.mean(h)  # np.sum(h)/float(N)
s_mean = np.mean(s)  # np.sum(s)/float(N)
v_mean = np.mean(v)  # np.sum(v)/float(N)
color_feature.extend([h_mean, s_mean, v_mean])
# The second central moment - standard deviation
h_var = np.var(h)
s_var = np.var(s)
v_var = np.var(v)
color_feature.extend([h_var, s_var, v_var])
# The third central moment - the third root of the skewness
h_skewness = np.mean(abs(h - h.mean())**3)
s_skewness = np.mean(abs(s - s.mean())**3)
v_skewness = np.mean(abs(v - v.mean())**3)
color_feature.extend([h_skewness, s_skewness, v_skewness])
color_feature = np.asarray(color_feature)
```

经过这一步，我们得到了 color_feature 这个九元向量。我们用它来作为一张图片的特征值。

余弦距离的计算也是依靠 numpy 函数：

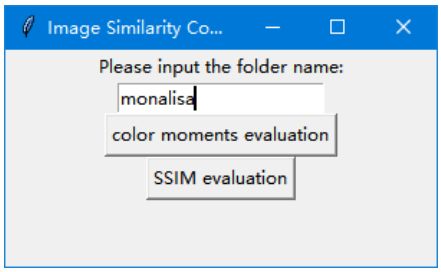
```
def cosine_coef(x, y):
    x_norm = np.linalg.norm(x)
    y_norm = np.linalg.norm(y)

    coef = np.dot(x, y) / (x_norm * y_norm)
    coef = round(coef, 5)
    return coef
```

下面是 SSIM 分析的实现。ssim 函数的两个参数是两个图像 numpy 数组，通过断言来确定两个图像的通道数都为二维。所以在传参之前，我们会先把图像的

通道转到 YCbCr，然后只取 Y 通道来分析。下面都是按照公式复现。

```python
def ssim(img1 , img2):
    '''
    Structural Similarity Index, range[0,1]
    :param img1:
    :param img2:
    :return:
    '''
    assert len(img1.shape) == 2 and len(img2.shape) == 2
    mu1 = img1.mean()
    mu2 = img2.mean()
    sigma1 = np.sqrt(((img1 - mu1) ** 2).mean())
    sigma2 = np.sqrt(((img2 - mu2) ** 2).mean())
    sigma12 = ((img1 - mu1) * (img2 - mu2)).mean()
    k1, k2, L = 0.01, 0.03, 255
    C1 = (k1 * L) ** 2
    C2 = (k2 * L) ** 2
    C3 = C2 / 2
    l12 = (2 * mu1 * mu2 + C1) / (mu1 ** 2 + mu2 ** 2 + C1)
    c12 = (2 * sigma1 * sigma2 + C2) / (sigma1 ** 2 + sigma2 ** 2 + C2)
    s12 = (sigma12 + C3) / (sigma1 * sigma2 + C3)
    ssim = l12 * c12 * s12
    return ssim
```

最后通过 tkinter 做了一个简单的 UI：



通过 matplotlib 来显示图片和计算结果。

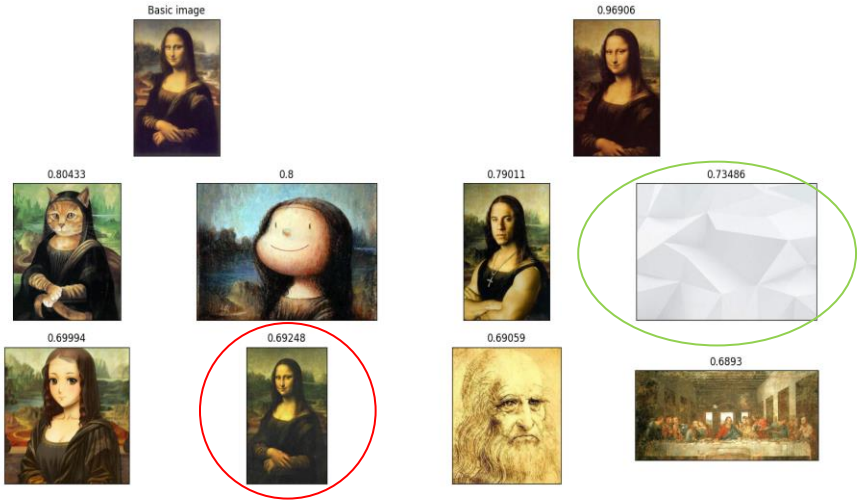
五、 实验结果分析

我设计了两套测试图集：

测试集 1：蒙娜丽莎

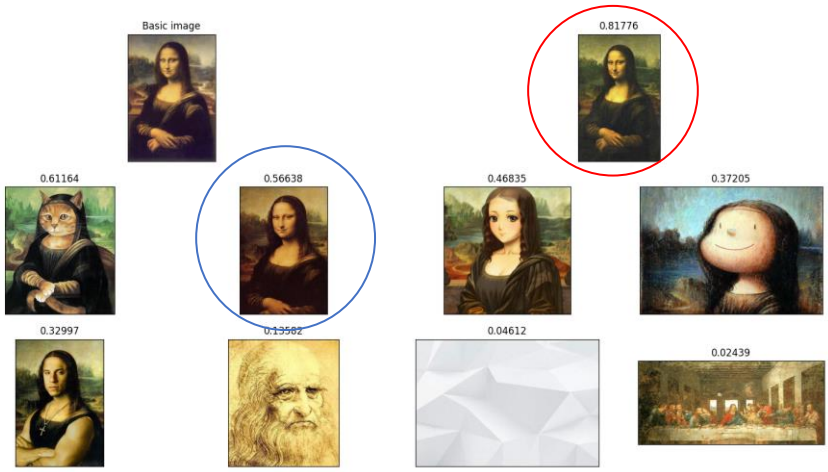
测试集 2：马

下面是测试集 1 在颜色矩估计下的相似度检测结果：



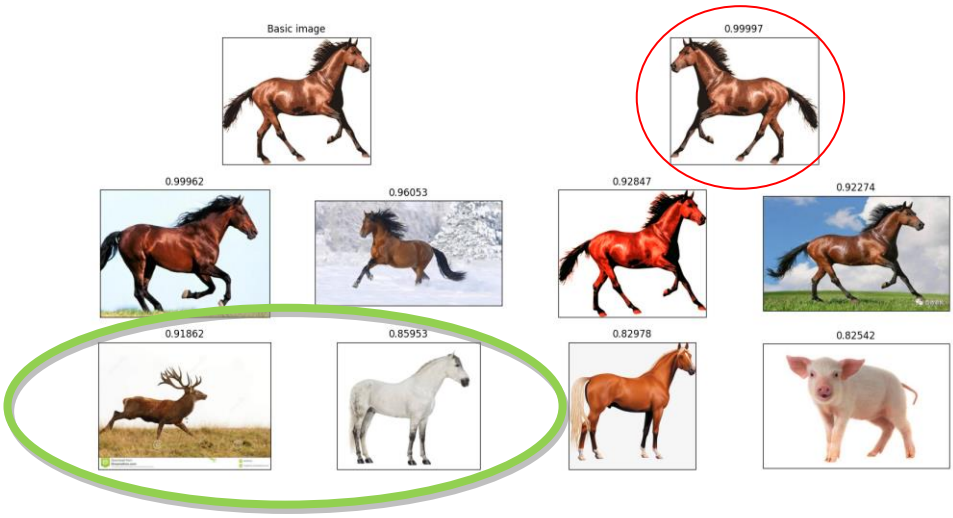
可以看到很遗憾的是。红色标记的图片因为与原图的色差，计算得出的相似度竟然还不如绿色标记的白色图片。然而实际上红色圈中的图片与原图在结构上几乎完全一致，只是有略微的色差。从中可以看出，颜色矩估计对于结构的检测能力为 0，仅仅是通过某些通道上像素值的统计结果来判断。

下面是测试集 1 在 SSIM 估计下的相似度检测结果：



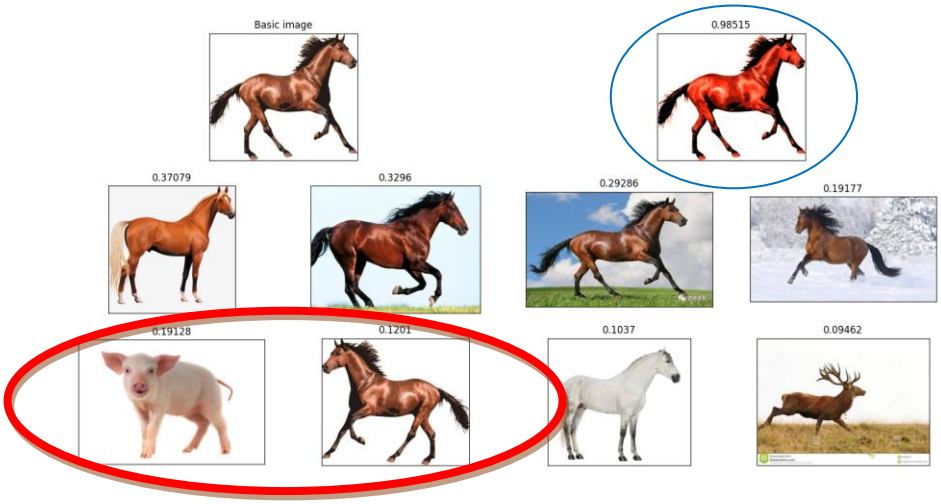
在 SSIM 估计下的结果要比颜色矩估计结果好很多。我们看到了之前那张红色图片被确定为相似度最高，结构相似度高达 80%。而蓝色框的图像（颜色矩中认为最像的）其实只有原图的部分，所以在 SSIM 估计中被排在较后。

下面是测试集 2 在颜色矩估计下的相似度检测结果：



在这里又发生了很有趣的事情：颜色矩估计对于图像的结构变化鲁棒性又显得很强。我们看到这张红色的图其实就是原图的对称版本。由于颜色的统计数据是一样的，因此相似度接近 100%。可是又有非常有意思的事情：绿色框里，由于颜色统计的问题，颜色矩估计认为这张鹿的图片比起一匹白马更像原图。这说明颜色矩估计没有语义判断的能力，不能用于图像检索。

下面是测试集 2 在 SSIM 估计下的相似度检测结果：



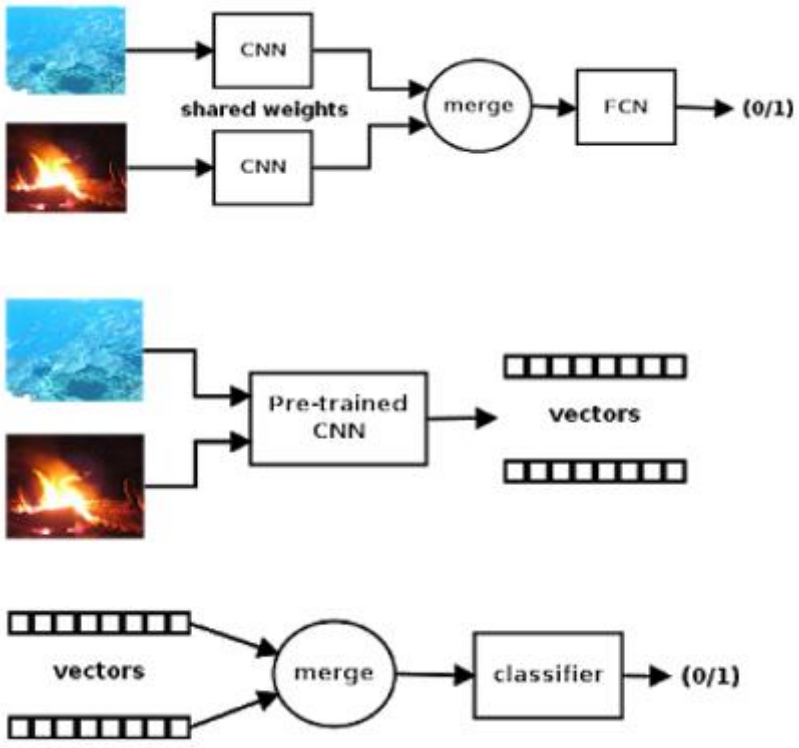
我们看到最相似的一张图是与原图有所色差的形状一模一样的图片。这一点 SSIM 的表现势必要比颜色矩统计更好。但是红框中的比较结果就很让人震惊：之前被颜色矩估计认为最佳的图片在 SSIM 里因为与原图结构完全不同因此计算结果不如一张猪的图片。可见 SSIM 的估计结果也让人感觉到不甚满意。但是我们可以通过增加图像变换后的数据来对 SSIM 检测加以改良，提高容错率。

六、 感想与反思

最后，结合实验内容以及课上所学对于基于内容的图像检索做一些分析和探索。

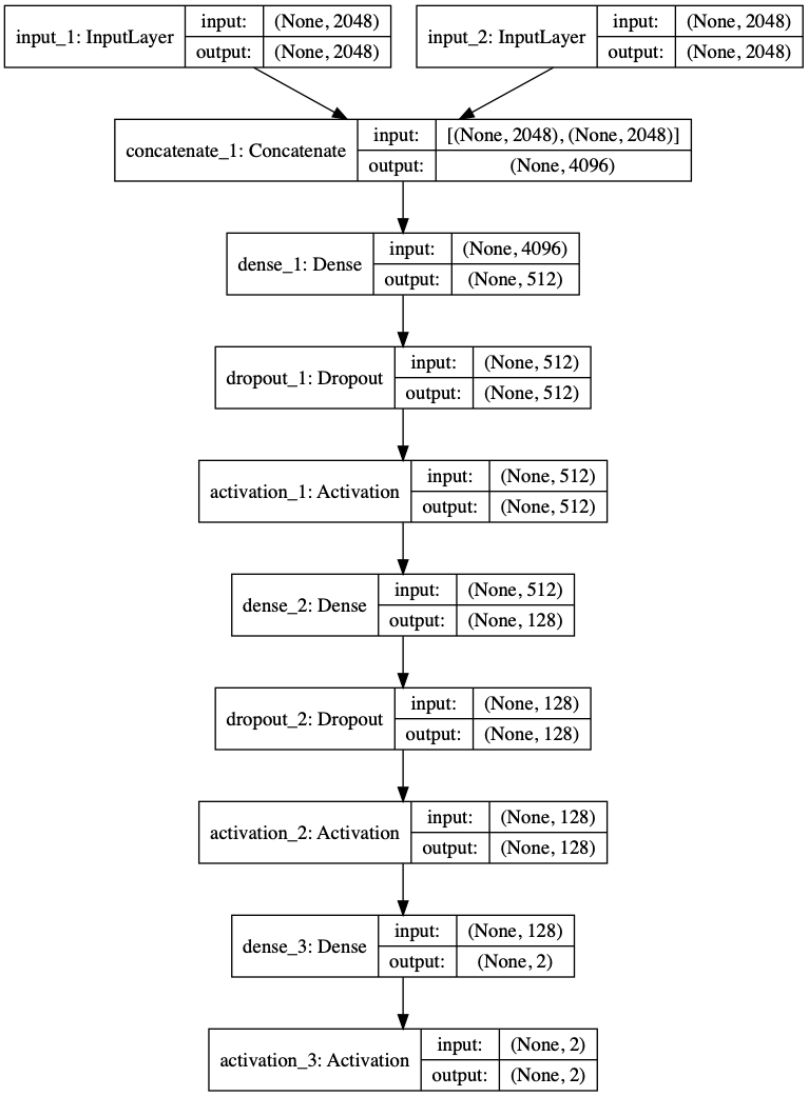
要进行 CBIR 系统的构建，首先对算法提出了如下要求：1.可以用很少的时间处理大量的数据；2.可以对给出的图片的语义进行分析与判断。单从第二个要求来看，实验中实现的简单的颜色矩估计与 SSIM 估计都是不靠谱的——尽管两种方法各有千秋，但他们对于同一类物体的辨识能力不强。其实仔细分析，这两个要求就限定了——一定要用深度学习的方法来训练这个搜索网络。

在之前的阅读中，我认为有一种基于孪生神经网络（Siamese Network）的图像相似度检测技术是可以学习的。其基本原理如下：用 CNN 计算特征向量——使用预训练的 inception v3 模型。然后用 FCN merge 两幅图像的特征向量。我们定义的交叉熵损失函数可以反向传播更新 FCN。最终得出的结果是“相似”或者“不相似”的结果。



预训练 CNN 的模型可以使用 vgg、inception 等

在 merge 部分，可以采用简单的合并方式。最后神经网络的结构示意图如下：



采用这种网络训练的，可以做到简单的语义上筛查，不会出现把颜色相近的“猪”当成是“马”。

但是构建 CBIR 系统的另一个要求就是——无论对于什么样的神经网络，合适的训练集是检索正确性与语义相关性的前提条件。因此数据集的选择也是很必要的，也就是 MMDB 的构建是 CBIR 的大前提。好的多媒体数据库才能够为基于内容的检索提供平台。现在我能够拿到的诸如 MNIST 之类的简单数据集显然只是学习用，根本不够 CBIR 的构建。不过练练手也是好的！