

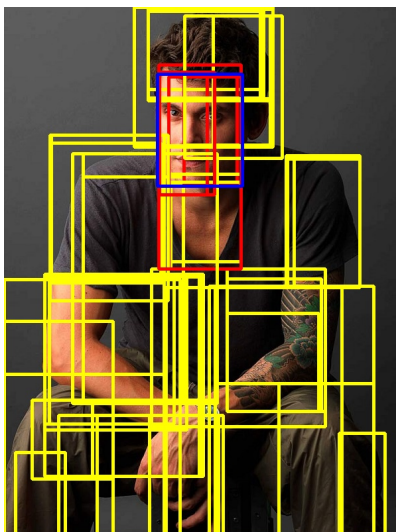
人脸识别系统作业报告

一、基本实现思路

人脸识别系统的实现主要由人脸检测和人脸识别两部分组成，这两部分可独立分别实现，最终合并

1. 人脸检测

- 使用**Selective Search**在训练集和测试集的原图中提取一系列包含可能目标的足够多的子窗口，并根据人脸标注计算子窗口与标注区域交并比 (IoU)，将大于阈值的子窗口标记为正样本，否则为负样本。其中测试集的正负样本标注只用于测试精度，不参与模型训练。提取效果如下图所示，其中蓝色框为人工标注，红色框为Selective Search提取出的正样本，黄色框为负样本（下图来自作业中的 `selectiveSearch.py`）：



- 对训练集中的所有子窗口缩放到同一尺寸 (20 x 20)，并提取**Haar特征**
- 使用**SVM**对Haar特征进行二分类训练，判断子窗口是否为人脸区域
- 测试集采用相同方法提取出一系列子窗口，缩放到同一尺寸并提取Haar特征，用训练完成的SVM模型对子窗口类别进行预测并测试模型精度

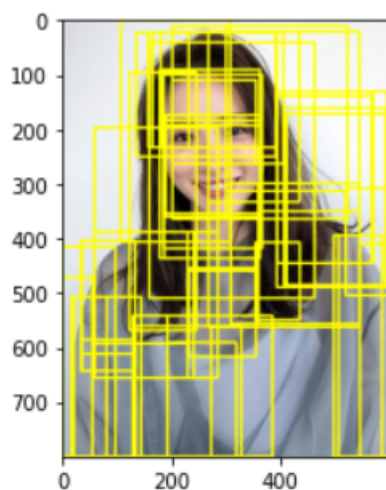
2. 人脸识别

- 截取所有的人脸区域并缩放至(100, 100)，再flatten至10000维的一维向量
- 每一类随机选5张人脸作为训练集，共10类，剩余作为测试集
- 将训练集人脸通过**PCA**方法提取**Eigenface**（特征脸），并将训练集中**每一类的所有人脸样本**用 *Eigenface* 进行表示后求均值，得到训练集中**每一类人脸的平均Eigenface表示**
- 将测试集样本同样使用 *Eigenface* 表示，并采用KNN（由于提前把每一类人脸样本用平均 *Eigenface* 表示，因此K=1，即最近邻算法）预测类别，即计算测试样本和每一类人脸在 *Eigenface* 空间的距离，取最小者作为预测结果。

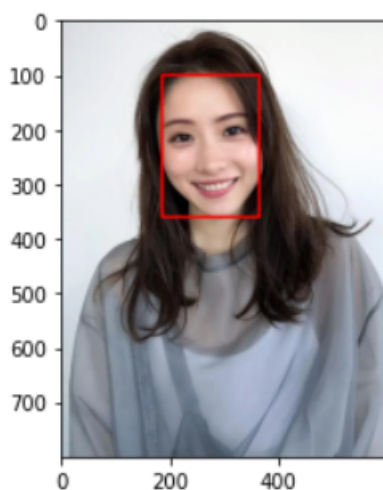
3. 模型合并

该部分详见 `demonstration.ipynb`

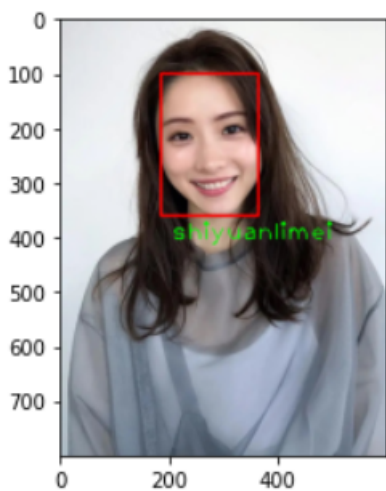
- 测试样本经过Selective提取一系列子窗口，并由训练后的**SVM**对子窗口进行分类，得到一系列的预测人脸子窗口。



- 在所有的预测人脸子窗口中，选择与其他子窗口交并比之和最大的窗口作为最终的检测结果



- 截取该窗口，将其映射到*Eigenface*表示空间，计算测试样本和每一类人脸在*Eigenface*空间的距离，取距离最小者（最近邻算法）类别作为识别结果



二、系统模型及算法分析

1. Selective Search

在人脸检测的训练中，首先需要对输入图片提取一系列可能含有目标的子窗口，进而对这些子窗口提取特征进行二分类。其中一种直接的子窗口提取算法是滑窗遍历法，通过一系列不同尺寸的滑窗在图片上滑动，遍历选取所有可能的子窗口，因此计算量庞大。

Selective Search是一种基于区域的区域提议算法，用以快速在输入图片中获取可能含有目标的区域。

Selective Search的算法流程：

1. 使用Felzenszwalb的快速方法得到初始分割区域集合 $R = \{r_1, r_2, \dots, r_n\}$;
2. 计算区域集R里两两相邻区域之间的相似度，并添加到相似度集合 S 中， $S = \{s_1, s_2, \dots\}$
3. 从相似度集合 S 中找出相似度最大的两个区域 r_i 和 r_j ，将其合并成为一个区域 r_t ，并删去原先与 r_i 和 r_j 相邻区域的相似度，计算 r_t 与其相邻区域的相似度，将其结果加入到相似度集合 S 中。同时将新区域 r_t 添加到区域集合 R 中；
4. 重复步骤3直到相似度集合为空集，此时区域集合 R 即所有可能含有目标的区域
5. 获取每个区域的Bounding Boxes

相似度计算方法：

在上述算法的步骤3中，比较两个区域的相似度在论文《Selective Search for Object Recognition》中主要由以下几方面构成：

1. 颜色相似度 (color similarity)

每一个区域用三通道的颜色直方图表示，每个通道下以 $bins = 25$ 计算直方图，每个区域的颜色直方图有 $25 * 3 = 75$ 个区间。对直方图除以区域尺寸做归一化后使用下式计算相似度：

$$s_{colour}(r_i, r_j) = \sum_{k=1}^n \min(c_i^k, c_j^k).$$

在区域合并后，颜色直方图计算如下：

$$C_t = \frac{\text{size}(r_i) \times C_i + \text{size}(r_j) \times C_j}{\text{size}(r_i) + \text{size}(r_j)}.$$

2. 纹理相似度 (texture similarity)

论文采用方差为1的高斯分布在8个方向做梯度统计，然后将统计结果（尺寸与区域大小一致）以 $bins=10$ 计算直方图。直方图区间数为 $8*10=240$ （使用RGB色彩空间）。相似度计算公式为：

$$s_{texture}(r_i, r_j) = \sum_{k=1}^n \min(t_i^k, t_j^k).$$

3. 尺度相似度 (size similarity)

为了尽量让较小的区域先合并，保证区域合并操作的尺度较为均匀，采用如下尺度相似度计算：

$$s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(r_j)}{\text{size}(im)}$$

4. 形状重合度 (shape compatibility measure)

合并后区域的Bounding Box越小，其重合度越高，形状重合度用以衡量两个区域重合程度：

$$fill(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)}$$

5. 最终相似度

两个区域最终的相似度是上述四部分相似度的和：

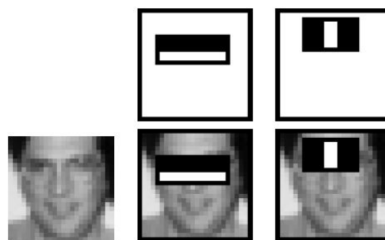
$$s(r_i, r_j) = a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j),$$

其中, $a_i \in \{0, 1\}$, 代表是否采用某一部分相似度。

2. Haar人脸特征提取

(1) Haar的概念

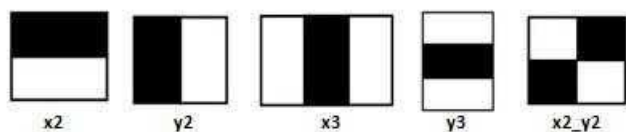
该方法通过矩形来描绘脸部特征，直观上如下图所示：



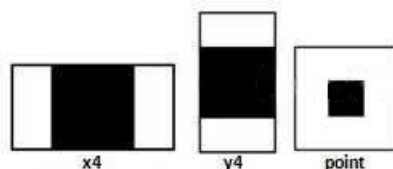
上图中两个矩形特征，表示出人脸的某些特征。中间一幅图表示眼睛区域的颜色比脸颊区域的颜色深，右边一幅表示鼻梁两侧比鼻梁的颜色要深。同样，其他目标，如眼睛等，也可以用一些矩形特征来表示。

常用的矩形特征如下图所示：

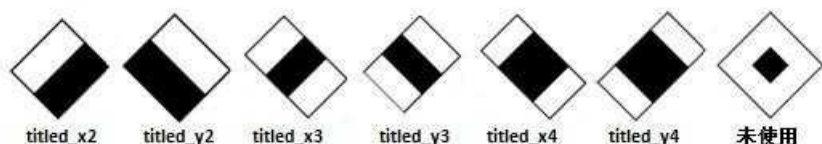
1. BASIC



2. CORE



3. ALL(Titled)



特征模板的特征值定义为： $Haar$ 特征值=整个 $Haar$ 区域内像素和 \times 权重 + 黑色区域内像素和 \times 权重

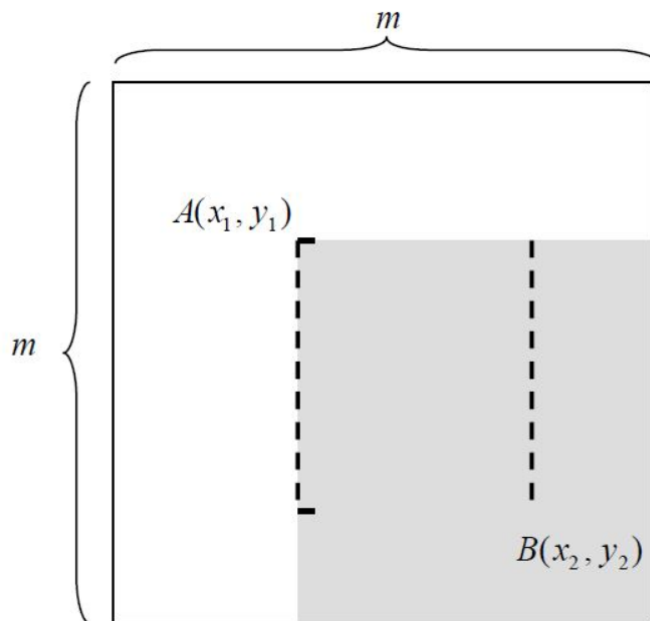
$$featureValue(x) = weight_{all} \times \sum_{Pixel \in all} Pixel + weight_{black} \times \sum_{Pixel \in black} Pixel$$

特征模板可以在子窗口内以任意尺寸任意位置放置，每一种形态都称为一个特征。因此人脸检测的一大关键点就是对每一个子窗口计算所有的Haar特征值，用以进一步的分类。

(2) Haar的相关计算

子窗口特征数量计算

一个 $m * m$ 大小的子窗口，需要计算在该尺寸下的子窗口内存在多少个Haar特征



对于 $m \times m$ 子窗口，只需要确定了矩形左上顶点 $A(x_1, y_1)$ 和右下顶点 $B(x_2, y_2)$ ，即可以确定一个矩形。如果这个矩形还必须满足以下两个条件（称为 (s, t) 条件，这个矩形即条件矩形）：

- 1) x 方向边长必须能被自然数 s 整除（能均等分成 s 段）；
- 2) y 方向边长必须能被自然数 t 整除（能均等分成 t 段）；

则，这个矩形的最小尺寸为 $s \times t$ 或 $t \times s$ ，最大尺寸为 $[m/s] \cdot s \times [m/t] \cdot t$ 或 $[m/t] \cdot t \times [m/s] \cdot s$ ；其中 $[]$ 为取整运算符。

通过以下两步计算即可定位一个满足条件的矩形：

1) 确定 $A(x_1, y_1)$: $x_1 \in \{1, 2, \dots, m-s, m-s+1\}$, $y_1 \in \{1, 2, \dots, m-t, m-t+1\}$;

2) 确定 A 点后, B 点只能在图 10 中阴影内 (包括边缘) 取值, 因此有:

$$x_2 \in X = \{x_1 + s - 1, x_1 + 2 \cdot s - 1, \dots, x_1 + (p-1) \cdot s - 1, x_1 + p \cdot s - 1\},$$

$$y_2 \in Y = \{y_1 + t - 1, y_1 + 2 \cdot t - 1, \dots, y_1 + (q-1) \cdot t - 1, y_1 + q \cdot t - 1\},$$

$$\text{其中 } p = \left\lceil \frac{m-x_1+1}{s} \right\rceil, \quad q = \left\lceil \frac{m-y_1+1}{t} \right\rceil. \text{ 并且知道: } |X| = p, \quad |Y| = q.$$






因此, 在 $m \times m$ 的子窗口中, 满足 (s, t) 条件的所有矩形的数量为:

$$\begin{aligned} \Omega_{(s,t)}^m &= \sum_{x_1=1}^{m-s+1} \sum_{y_1=1}^{m-t+1} p \cdot q \\ &= \sum_{x_1=1}^{m-s+1} \sum_{y_1=1}^{m-t+1} \left\lceil \frac{m-x_1+1}{s} \right\rceil \cdot \left\lceil \frac{m-y_1+1}{t} \right\rceil \\ &= \sum_{x_1=1}^{m-s+1} \left\lceil \frac{m-x_1+1}{s} \right\rceil \cdot \sum_{y_1=1}^{m-t+1} \left\lceil \frac{m-y_1+1}{t} \right\rceil \\ &= \left(\left\lceil \frac{m}{s} \right\rceil + \left\lceil \frac{m-1}{s} \right\rceil + \dots + \left\lceil \frac{s+1}{s} \right\rceil + 1 \right) \cdot \left(\left\lceil \frac{m}{t} \right\rceil + \left\lceil \frac{m-1}{t} \right\rceil + \dots + \left\lceil \frac{t+1}{t} \right\rceil + 1 \right) \end{aligned}$$

由于共有5中特征模板, 因此矩形特征的数量就是满足5种 (s, t) 条件的矩形特征数量和:

$$\Omega^m = \Omega_{(1,2)}^m + \Omega_{(2,1)}^m + \Omega_{(1,3)}^m + \Omega_{(3,1)}^m + \Omega_{(2,2)}^m$$

以 24×24 的子窗口为例计算特征总数量如下:

特征模板	数量
1、  2、 	$2 \times \Omega_{(1,2)}^{24} = 2 \times \left(\left\lfloor \frac{24}{1} \right\rfloor + \left\lfloor \frac{23}{1} \right\rfloor + \cdots + \left\lfloor \frac{2}{1} \right\rfloor + 1 \right) \times \left(\left\lfloor \frac{24}{2} \right\rfloor + \left\lfloor \frac{23}{2} \right\rfloor + \cdots + \left\lfloor \frac{3}{2} \right\rfloor + 1 \right)$ $= 2 \times (24 + 23 + \cdots + 2 + 1) \times (12 + 11 + 11 + \cdots + 2 + 1 + 1)$ $= 2 \times 300 \times 144 = 2 \times 43,200$ $= 86,400$
3、  4、 	$2 \times \Omega_{(1,3)}^{24} = 2 \times \left(\left\lfloor \frac{24}{1} \right\rfloor + \left\lfloor \frac{23}{1} \right\rfloor + \cdots + \left\lfloor \frac{2}{1} \right\rfloor + 1 \right) \times \left(\left\lfloor \frac{24}{3} \right\rfloor + \left\lfloor \frac{23}{3} \right\rfloor + \cdots + \left\lfloor \frac{4}{3} \right\rfloor + 1 \right)$ $= 2 \times (24 + 23 + \cdots + 2 + 1) \times (8 + 7 + 7 + \cdots + 1 + 1 + 1)$ $= 2 \times 300 \times 92 = 2 \times 27,600$ $= 55,200$
5、 	$\Omega_{(2,2)}^{24} = \left(\left\lfloor \frac{24}{2} \right\rfloor + \left\lfloor \frac{23}{2} \right\rfloor + \cdots + \left\lfloor \frac{3}{2} \right\rfloor + 1 \right) \times \left(\left\lfloor \frac{24}{2} \right\rfloor + \left\lfloor \frac{23}{2} \right\rfloor + \cdots + \left\lfloor \frac{3}{2} \right\rfloor + 1 \right)$ $= (12 + 11 + 11 + \cdots + 2 + 1 + 1) \times (12 + 11 + 11 + \cdots + 2 + 1 + 1)$ $= 144 \times 144$ $= 20,736$
总数	$\Omega^{24} = 162,336$

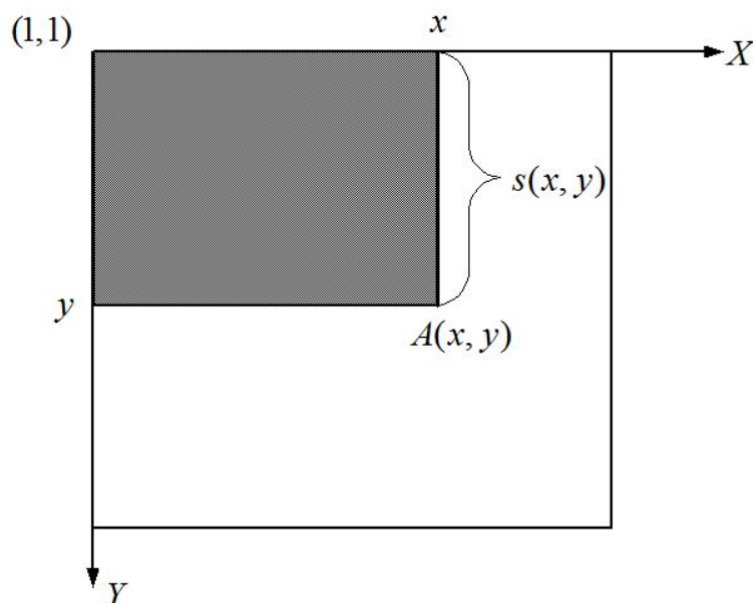
利用积分图求解特征值

定义： $ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$ 为图像内一点 (x, y) 的积分图，其中 $i(x', y')$ 为点 (x', y') 处的原始图，即颜色值。积分图也可以由以下两式迭代求出：

$$s(x, y) = s(x, y - 1) + i(x, y)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

直观上如下图所示，其中阴影部分为A点的积分图， $s(x, y)$ 为A点y方向上所有像素之和

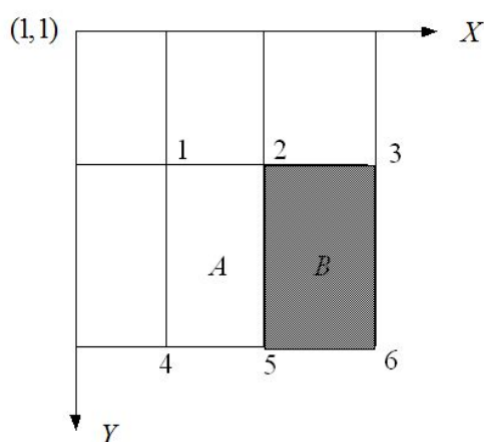


设图像大小为 $m \times n$ ，则积分图矩阵（图像上所有像素的积分图）为：

$$\begin{pmatrix} ii(1,1)=ii(0,0)+s(1,1) & ii(2,1)=ii(1,1)+s(2,1) & \dots & ii(m-1,1)=ii(m-2,1)+s(m-1,1) & ii(m,1)=ii(m-1,1)+s(m,1) \\ s(1,1)=s(1,0)+i(1,1) & s(2,1)=s(2,0)+i(2,1) & \dots & s(m-1,1)=s(m-1,0)+i(m-1,1) & s(m,1)=s(m,0)+i(m,1) \\ \hline ii(1,2)=ii(0,2)+s(1,2) & & & & ii(m,2)=ii(m-1,2)+s(m,2) \\ s(1,2)=s(1,1)+i(1,2) & & & & s(m,2)=s(m,1)+i(m,2) \\ \vdots & & & & \vdots \\ ii(1,n-1)=ii(0,n-1)+s(1,n-1) & & & & ii(m,n-1)=ii(m-1,n-1)+s(m,n-1) \\ s(1,n-1)=s(1,n-2)+i(1,n-1) & & & & s(m,n-1)=s(m,n-2)+i(m,n-1) \\ \hline ii(1,n)=ii(0,n)+s(1,n) & ii(1,n-1)=ii(0,n-1)+s(1,n-1) & \dots & ii(m-1,n)=ii(m-2,n)+s(m-1,n) & ii(m,n)=ii(m-1,n)+s(m,n) \\ s(1,n)=s(1,n-1)+i(1,n) & s(1,n-1)=s(1,n-2)+i(1,n-1) & \dots & s(m-1,n)=s(m-1,n-1)+i(m-1,n) & s(m,n)=s(m,n-1)+i(m,n) \end{pmatrix}$$

可见，只需要遍历图像一次，迭代 $m \times n \times 2$ 次，即可以得到整个积分图矩阵。

由以上分析，一个Haar特征的特征值就可以通过积分图来计算（以边缘模板举例）：



根据定义，改Haar特征的特征值为区域A的像素值减去区域B的像素值

区域 A 的像素值 $= ii(5) + ii(1) - ii(2) - ii(4)$ 。

区域 B 的像素值 $= ii(6) + ii(2) - ii(5) - ii(3)$ 。

所以：

该矩形特征的特征值 $=$

$$ii(5) + ii(1) - ii(2) - ii(4) - [ii(6) + ii(2) - ii(5) - ii(3)] =$$

$$[ii(5) - ii(4)] + [ii(3) - ii(2)] - [ii(2) - ii(1)] - [ii(6) - ii(5)]$$

所以Haar特征值只与端点积分图有关，特征计算速度很快，也提高了目标的检测速度。

3. SVM

在人脸检测部分，配合快速的Selective Search对原始图片提取出一系列子窗口，使用SVM算法进行分类预测，可以有效地从众多子窗口中对正负样本进行分类，得到正确的人脸区域。SVM的分析如下：

三要素	要素内容
模型	SVM是一种二分类模型，它的基本模型是定义在特征空间上的 间隔最大的线性分类器 ，又分为以下三种情况：线性可分支持向量机、线性支持向量机、非线性支持向量机，分别用于在线性可分的训练数据、近似线性可分的训练数据、线性不可分的训练数据上实现二分类任务。
策略	间隔最大化（1）对于 线性可分 的训练数据，使用 硬间隔最大化 学习，得到的SVM模型是一个线性的分类器，也称为硬间隔分类器；（2）对于 近似线性可分 的训练数据，使用 软间隔最大化 学习，同样得到一个线性分类器；（3）对于 线性不可分 的训练数据，使用 核函数 将输入空间映射到特征空间，得到近似线性可分的数据特征，并通过 软间隔最大化 学习，等价于隐式地在高维的特征空间中学习一个线性分类器。
算法	在本项目中，使用序列最小最优化算法（SMO），它用于快速求解SVM学习的凸二次规划问题，其基本思路是将原问题不断分解为对两个变量的子问题的求解，这两个变量通过启发式算法得到，一个为违反KKT条件最严重的，另一个由约束条件决定。通过解析方法不断求解子问题，直到所有变量都满足KKT条件即得解。

4. PCA

在人脸识别部分，可以使用PCA将人脸区域转换为由**Eigenface空间**表示的低维数据，便于进一步实现人脸特征的比较和识别。

PCA是一种**无监督的数据降维**算法，它通过**正交变换**把由**线性相关变量**表示的观测数据转换成为少数几个由**线性无关变量（主成分）**表示的数据，而转换后主成分的个数通常小于原始变量个数，可以实现有效的数据降维。

Eigenface的理论计算过程

1. 获取所有的人脸图片 (I_1, I_2, \dots, I_M) 并缩放至统一尺寸并 *flatten* 至一维，以相同长度一维向量表示 $(\Gamma_1, \Gamma_2, \dots, \Gamma_M)$

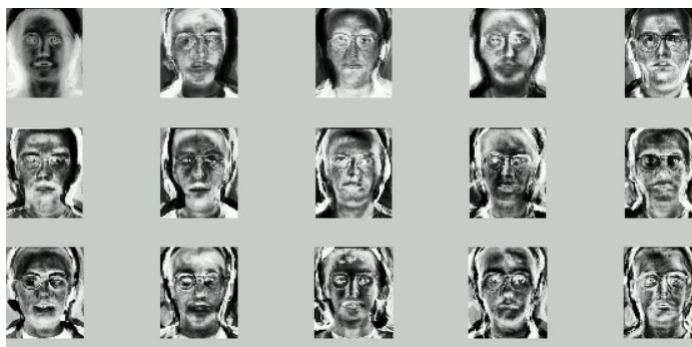
2. 计算步骤1中所有向量的平均向量 $\Psi = \frac{1}{M} \sum_{i=1}^M \Gamma_i$ ，并将每个样本向量减去平均向量

$$\Phi_i = \Gamma_i - \Psi, \text{ 得到0均值样本向量}$$

3. 将训练集的样本向量按列排放组成样本矩阵 $A = [\Phi_1, \Phi_2, \dots, \Phi_M]$

4. 计算A的协方差矩阵C: $C = \frac{1}{M} \sum_{n=1}^M \Phi_n \Phi_n^T = AA^T$

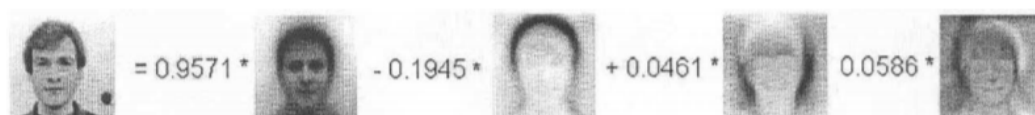
5. 使用截断奇异值分解 $C = U\Sigma V^T$ 计算C的特征值和对应的单位特征向量，此时得到的V的前k列构成k个样本主成分，这k个特征向量也称为 *Eigenface*，将其还原为二维矩阵，将直观上得到如下图所示的特征脸：



6. 每张脸可以近似视为 k 个样本主成分 (k 张 *Eigenface*) 的线性组合:

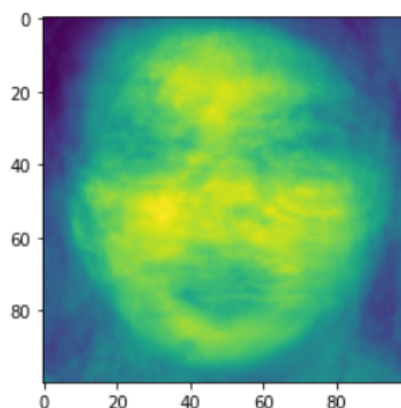
$\hat{\Phi}_i - \text{mean} = \sum_{j=1}^K w_j u_j$, ($w_j = u_j^T \Phi_i$), 此时每张脸可以由一个 K 维向量进行表示:

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ \dots \\ w_K^i \end{bmatrix}, \quad i = 1, 2, \dots, M, \quad \text{直观上如下图所示:}$$



由于协方差矩阵 $C = AA^T$ 维度极高, 导致计算量庞大, 因此实际实现中采用了更为简便快速的计算技巧, 详见报告第四部分[Eigenface的快速求法](#)

注: 步骤2中训练集中所有样本的平均向量可以还原为二维图像, 其意义为训练集的“平均脸”。在本作业训练集中可得到如下“平均脸” (下图来自 `demonstration.ipynb`) :



5. KNN

在人脸识别中, 将待检测人脸和训练集人脸分别映射到 *Eigenface* 表示空间中, 并计算待检测人脸到训练样本的距离, 使用 *KNN* 实现人脸类别的预测。在本项目的实现中, 由于预先将每一类人脸映射到 *Eigenface* 后取了平均值, 即每一类人脸都在 *Eigenface* 空间存在唯一表示, 因此 *KNN* 只需要取 $k = 1$, 即最近邻算法。

KNN 算法分析如下:

三要素	要素内容
距离度量	<p>欧式距离：</p> $d(x, y) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}.$ <p>反映两个样本点的相似程度，也可以更一般的L_p距离或$Minkowski$距离</p>
k 值的 选择	<p>选择较小的k值，只有与输入实例较近的训练样本对预测结果起作用，近似误差减小，但会造成预测结果对邻近点的敏感，估计误差增大，且造成模型复杂，易过拟合；选择较大的k值则相反，使得近似误差增大，估计误差减小，模型简单</p>
分类决策规则	<p>采用多数表决的分类决策规则，由输入样本的k个邻近训练样本中的多数类别决定预测类别</p>

三、数据准备及处理

1. 原始数据整理

将每张原始图片路径及对应的 label（即所属类别和人脸框）作为一条样本，并全部打乱，以字典的形式存放于 json 文件，用于进一步的数据读取。json 文件内容如下所示：

```
{
  "dataset/huangbo/huangbo_01.png":
    {"bbox": [115, 146, 355, 421], "class": "huangbo"},
  "dataset/shiyuanlimei/shiyuanlimei_18.png":
    {"bbox": [116, 33, 255, 245], "class": "shiyuanlimei"},
  .....
}
```

2. 人脸检测正负样本划分

- 先从[步骤1](#)整理好的原始数据中随机选取 80% 作为训练集，剩余 20% 作为测试集
- 采用**Selective Search**对每张原始图片提取一系列可能含有目标的矩形框
- 对**Selective Search**提取出的所有矩形框计算与 ground truth bbox 的交并比（IoU），选择 $IoU > 0.4$ 的矩形框作为正样本，其余为负样本。由于负样本数量远多于正样本，因此在所有负样本中随机选取和正样本数量相同的矩形框，参与训练。
- 将训练集和测试集的图片路径与对应正负样本，分别以字典形式存于两个 json 文件当中，文件内容如下所示：

```
{
  "dataset/jingtian/jingtian_09.png":{
    "p_sample": [[184,71,192,232],[177,173,133,138],[160,57,246,218]],
    "n_sample": [[0,0,168,182],[18,0,162,244],[0,0,305,642]]
  },
  .....
}
```

3. 人脸识别数据处理

在人脸识别部分需要将训练集每一类人脸映射到 *Eigenface* 表示空间中，因此从每一类人脸中随机抽样 M 个样本（实现中M取5），并将其保存于字典中。数据组织内容如下：

```
{
  'huangbo': ['dataset/huangbo/huangbo_20.png',
'dataset/huangbo/huangbo_12.png', 'dataset/huangbo/huangbo_15.png',
'dataset/huangbo/huangbo_09.png', 'dataset/huangbo/huangbo_05.png'],
  'jingtian': ['dataset/jingtian/jingtian_14.png',
'dataset/jingtian/jingtian_02.png', 'dataset/jingtian/jingtian_12.png',
'dataset/jingtian/jingtian_19.png', 'dataset/jingtian/jingtian_05.png'],
  .....
}
```

四、存在问题及解决方案

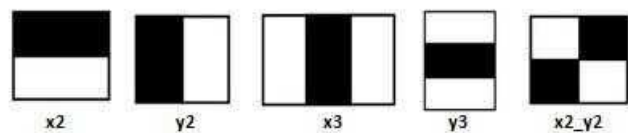
1. Haar特征值计算

问题：

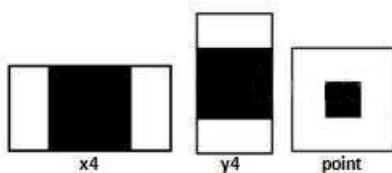
在部分网上参考资料中，Haar特征值定义为：特征值 = Haar区域内白色部分像素值之和 - 黑色部分像素值之和，而参考了opencv中的源码，特征值定义为：特征值=整个Haar区域内像素和×权重 + 黑色区域内像素和×权重，因此以下对Haar特征值的计算详细讨论。

分析：

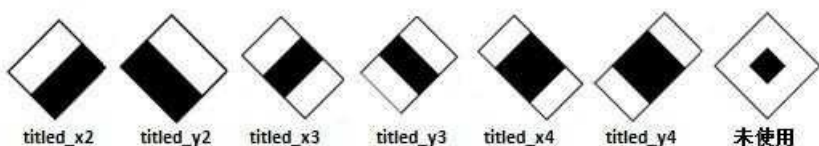
1. BASIC



2. CORE



3. ALL(Titled)



opencv源码中对Haar特征值的定义是更为严谨的，即：

$$featureValue(x) = weight_{all} \times \sum_{Pixel \in all} Pixel + weight_{black} \times \sum_{Pixel \in black} Pixel$$

- 对于图中的x3和y3特征， $weight_{all} = 1$ ， $weight_{black} = -3$ ；
- 对于point特征， $weight_{all} = 1$ ， $weight_{black} = -9$ ；
- 其余11种特征均为 $weight_{all} = 1$ ， $weight_{black} = -2$

因此，对于x2, y2这类黑白区域面积相等的特征值，计算过程如下：

$$(\text{黑} + \text{白}) * 1 + \text{黑} * (-2) = \text{白} - \text{黑}$$

此时特征值才等价于Haar区域内白色部分像素值之和 - 黑色部分像素值之和，而对于x3, y3这类黑白区域面积不相等的特征值，则不成立。

特征值采用加权和的原因：

设置权值是为了抵消面积不等带来的影响，保证所有Haar特征的特征值在灰度分布绝对均匀的图中为0。

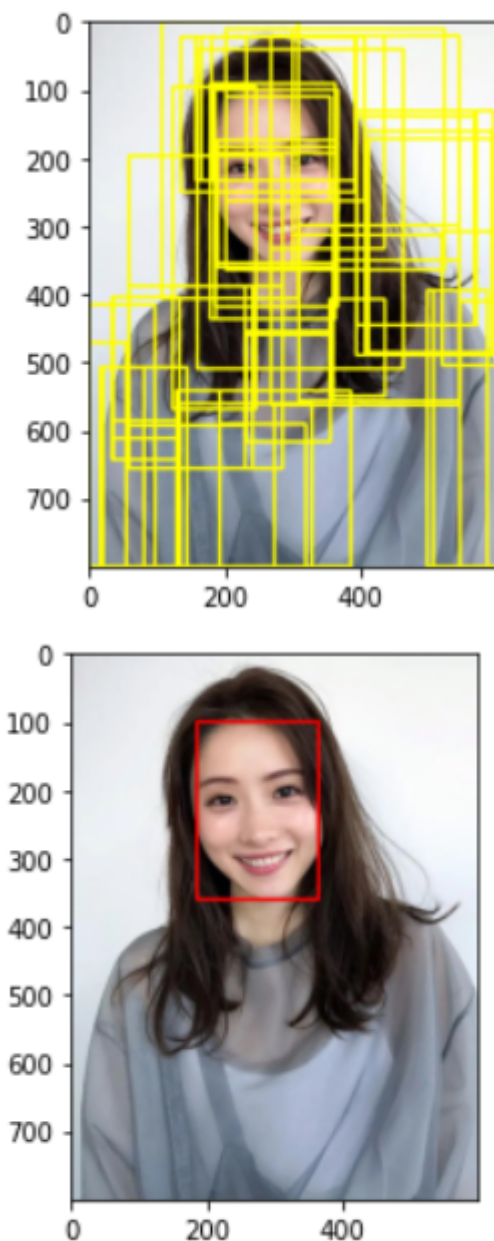
2. 选取最终检测结果

问题：

由于使用`SelectiveSearch`得到的一系列子窗口中，有很大一部分会被SVM分类为人脸区域，而SVM是非概率模型，较难直接使用非极大值抑制得到最终检测结果。

解决方案：

假设最优的子窗口与其他含人脸区域的子窗口的 IoU 之和最高，则在所有被分类为人脸区域的子窗口中，计算每个子窗口与其他子窗口的 IoU 之和，选择 IoU 之和最高的子窗口作为最终的检测结果。如下图所示：



3. KNN识别正确率低

问题:

在人脸识别部分中，最初方案为：将测试图片和训练集的人脸都映射到 *Eigenface* 表示空间中，计算测试样本与所有训练样本的欧式距离，并用KNN预测识别结果，最终只取得了10%左右的正确率。

可能原因:

由于收集的数据集样本少，且同类人脸也存在较大差异，最终导致测试图片在和所有训练样本计算距离时误差很大。

解决方案:

将训练集的人脸映射到 *Eigenface* 表示空间中后，每一类的特征脸再求平均值，即**每一类人脸求得唯一的 *Eigenface* 平均表示**，以此模糊同类人脸的差异。测试图片映射到特征空间后，和每一类平均特征脸求距离，利用最近邻算法 ($K=1$) 来预测类别，正确率能达到50%以上，有显著的提高

4. Eigenface快速求法

问题:

假设每个样本 Φ_i 为长度为 N^2 的一维向量（人脸区域为 $N \cdot N$ 的矩阵），共 M 个样本，样本矩阵 $A = [\Phi_1, \Phi_2, \dots, \Phi_M]$ ，其协方差矩阵为 $C = \frac{1}{M} \sum_{n=1}^M \Phi_n \Phi_n^T = AA^T$ ，此时协方差矩阵大小为 $N^2 \cdot N^2$ ，此时求解特征值与特征向量计算量庞大，实际不采用该方法。

解决方案：

- 考虑矩阵 $A^T A$ ，其大小为 $M \cdot M$
- 计算 $A^T A$ 的特征值与特征向量， $A^T A v_i = \mu_i v_i$
- 根据以下推导可知： $A^T A$ 与 AA^T 具有相同特征值，且特征向量关系为 $u_i = A v_i$

$$A^T A v_i = \mu_i v_i \Rightarrow AA^T A v_i = \mu_i A v_i \Rightarrow$$

$$C A v_i = \mu_i A v_i \text{ or } C u_i = \mu_i u_i \text{ where } u_i = A v_i$$

- 因此可以通过计算 $A^T A$ 的特征值与特征向量得到 AA^T 的 M 个样本主成分 $AA^T: u_i = A v_i$

通过上述方法，可以大大降低求解Eigenface的计算量

五、实验结果

经过多轮测试，最终选定以下预设参数（均存放于 `tools/config.py`）

参数意义	参数值
SVM最大迭代次数	5000
检测部分窗口缩放尺寸	(20, 20)
检测训练集数据占比	80%
检测正样本阈值	0.4
有效子窗口最小面积	5000
有效子窗口最大面积	240000
识别部分训练集每类样本数	5
识别部分人脸区域缩放尺寸	(100, 100)

最终的检测测试结果正确率 $\approx 50\%$ ，识别测试结果正确率在 45% - 58%之间，（两部分独立训练并测试）

由于两部分的训练集和测试集不同，因此没有模型合并部分没有进行正确率测试，模型合并的演示详见 `demonstration.ipynb`

[1] OpenCV AdaBoost + Haar目标检测技术内幕（上） <https://zhuanlan.zhihu.com/p/31427728>

[2] Uijlings, J. R R, Sande V D, et al. Selective Search for Object Recognition[J]. International Journal of Computer Vision, 2013, 104(2):154-171. http://www.vision.jhu.edu/teaching/vision08/Handouts/case_study_pca1.pdf

[3] Viola P A , Jones M J . Rapid Object Detection using a Boosted Cascade of Simple Features[C]// Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. IEEE, 2001. http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf