脑认知与智能计算课程设计

Deepfake 面部特征提取算法

学　　院	电子与信息学院
专　　业	数据科学与大数据
学生姓名	苗成林
学生学号	1851804
指导教师	孙杳如
提交日期	2021 年 12 月 22 日

目　录

研究背景介绍

1.1 GAN 简介

生成对抗网络 (GAN, Generative Adversarial Network)[1]，这是一种较新的深度学习模型。该机器学习模型是由两个模块组成的：一是生成模型(Generative Model)，二是判别模型 (Discriminative Model)。前者是通过输入给定的一些信息，输出要观测的数据；而后者是输入要观测的数据，通过某种模型给出预测，做出判断。

下面举个训练生成假脸模型的例子说明 GAN 的基本原理。生成模型(G)：给定原始人脸图像和目标人脸图像，作为输入，通过算法生成合成人脸图像，该图像的人脸部分是目标脸的，表情口型等是原始脸的。判别模型(D)：给定一张人脸图像，判断其是原始人脸还是换脸合成图像。

G(x,y)是一个生成模型网络，x,y 为输入的原始脸和目标脸， G(x,y) 为输出的合成图像；D(z)是一个判别网络，z 表示输入的要进行识别的

ng图像（或者图像的某些特征信息），D(z)则是输出，表示 z 为原始脸的概率（或者说是判别结果人脸的真假）。如果 D(z)=1，则表示 z 是原始脸的概率为 100%；D(z)=0 则表示 z 一定是合成脸。

在训练过程中，生成网络 G 的目标就是尽量生成较为形象生动的合成人脸图，去欺骗它的“对手”对抗网络 D；而对抗网络 D 的目标则是提高识别能力，最大程度地将它的“对手”生成网络 G 生成的图像与真实的人脸区分开来。从而形成一个二者的动态博弈过程。在理想情况下，最终，生成网络 G 生成能够“以假乱真”的图像 G(x,y)。对于判别网络 D 而言，它已经很难区分其是真是假，所以其输出 D(G(x,y))=0.5。这样，我们就会得到一个很不错的生成模型 G，最终可用其生成合成人脸图像。

1.2 Deepfake 简介

Deepfake，是英文“Deep Learning”（深度学习）和“Fake”（伪造）的混成词，专门指代基于人工智能的，关于人体图像合成（Human Image Synthesis）技术的应用。此项技术能够将目标图像、影片叠加到原始的图像、影片上面。

照片处理从 19 世纪开始发展，并很快应用于影视作品，20 世纪飞速发展，数字视频的发展更为快速。从 20 世纪 90 年代开始，学术机构的研究人员开发了 Deepfake 技

术，后来网络社区的业余爱好者进一步发展了这种技术。最近，这种方法被广泛应用。

学术研究方面，与 Deepfake 相关的学术研究主要集中在计算机科学的一个分支领域——计算机视觉。早期的一个里程碑式的项目是 1997 年出版的视频重写程序，它修改了一个人讲话的现有视频片段，以描绘他在不同的音频轨道中说的话[2]。这是第一个完全自动化面部复活的系统，它使用机器学习技术将视频对象发出的声音和对象的面部形状联系起来。

当代学术项目的重点是创造更真实的视频和改进技术。2017 年出版的“Synthesizing Obama”计划[3]，修改美国前总统 Barack Obama 的视频片段，以描绘他在一个单独的音频轨道中说的话。该项目将其从音频合成口形的真实照片技术列为主要研究目的。2016 年出版的 Face2Face[4]程序修改了一个人面部的视频片段，以描绘他们实时模仿另一个人的面部表情。该项目不用使用能够捕捉深度的相机，即可在新脸上实时再现原脸的面部表情，这使得消费者使用普通相机实现该技术成为可能。

加州大学伯克利分校的研究人员，在 2018 年 8 月，发表了一篇与 Deepfake 技术有关的论文，其中介绍了一款假冒舞蹈的应用程序[5]，该应用程序可以使用人工智能创造出具有高超舞蹈技能的图像。之前的作品主要集中在头部或脸部的某个部位，而该项目则将 Deepfake 的应用扩展到全身。

算法原理介绍

2.1 图像人脸提取方法的比较

2.1.1 方法简介

人脸识别简单流程如下：首先将彩色图像转换为灰度图，三维变一维简化识别过程；然后将图像分成具有一定像素大小的小方块；在每块中，计算出各个主方向有多少个梯度，即有多少向上、向左上、向左、向左下等，用指向性最强的那个箭头代替原来的小方块；这样就将图像转为了 HOG (Histogram of Oriented Gradients)表达式，如图 2-1，我们可以从中轻松获得面部基本结构；接下来就是从 HOG 图像中找到面部区域，这时就要用到已经训练好的模型了，这些模型是经过大量面部数据的训练而得的，一般的训练思路是寻找面部标志点（例如，68 个标志点），如图 2-2。

图 2-1 HOG 图像

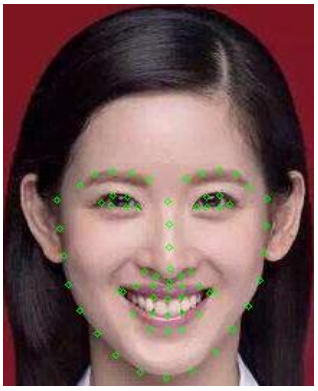


图 2-2 68 个 Landmark

接下来，我们对其中三种常用方法进行实验比较。分别是 OpenCV，Dlib 和 Face_recognition 库中的方法。

（1）在 OpenCV 中，我们运用的主要函数有两个：

```
cascade = cv2.CascadeClassifier(ClassifyModel)
rectangles = cascade.detectMultiScale(image, scaleFactor, minNeighbo
rs, minSize, maxSize, flags)
```

第一个函数是定义级联分类器，而其使用的模型是 OpenCV 自带的训练模型。第二个函数则是运用级联分类器，检测图像 image，在每个图像区域大小减少 scaleFactor，保留最小邻域 minNeighbors，最小尺寸 minSize，最大尺寸 maxSize，设定阈值函数 flags 的情况下，检测出人脸，返回其矩形框的坐标，左上点和右下点：[(x1,y1)(x2,y2)]。

（2）在 Dlib 中，我们用到了这两个函数：

```
detector = dlib.get_frontal_face_detector()
dets = detector(img,  1)
```

第一个函数是定义 dlib 自带的人脸检测器。第二个函数调用该检测器，其中，第一个参数是已通过函数读取的图像数据，第二个参数则是对图像上采样的次数。上采样则是为了让我们能够发现更多的人脸。返回值是 'dlib.dlib.rectangle' 类型，即识别出来的人脸区域的矩形框坐标，左上点和右下点：[(x1,y1)(x2,y2)]，可用如下代码获取具体值：

```
for i, d in enumerate(dets):
    x1, y1, x2, y2 = d.left(), d.top(), d.right(), d.bottom()
```

（3）在 Face_recognition 中，我们运用了这个函数

```
face_locations = face_recognition.face_locations(image)
```

该函数结构简单，输入参数是已读取的图像数据，可用 cv2.imread(imagepath)或者 face_recognition.load_image_file(imagepath)读取。返回值是识别的人脸矩形框坐标组，组数表示人脸个数，每组有四个值[top, right, bottom, left]表示矩形框的[y1,x2,y2,x1]。

2.1.2 实验对比

在做测试时，我们分别在三个数据集 Celeba, PGGAN[16], DFD 中抽样，下载地址：http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html ，https://github.com/zhangqianhui/progressive_growing_of_gans_tensorflow ，https://github.com/deepfakeinthewild/deepfake_in_the_wild，共计 20 张图像，如图 2-3 的 1，2 行所示。三种提取方法提取出来的人脸如图 2-4 所示，我们将未识别的图像保存为

黑色以便比较。三种提取人脸方法的运行时间、识别率、准确率统计如表 2-1 所示。

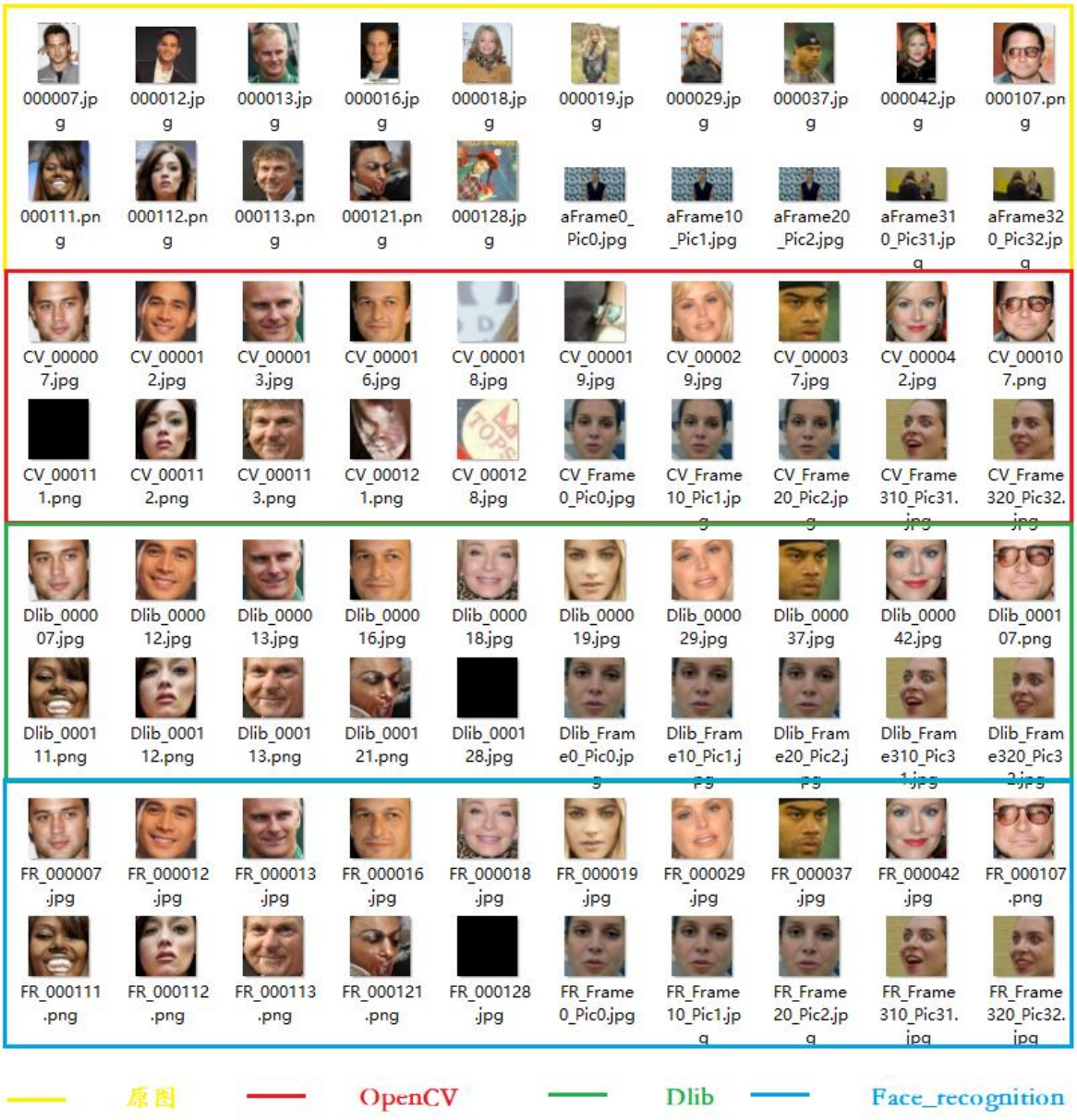


图 2-3 原图及三种方法检测出来的人脸图对比。第 1，2 行是抽样的 20 张测试图像；第 3，4 行是 OpenCV 提取的人脸；第 5，6 行是 Dlib 提取的人脸；第 7，8 行是 Face_recognition 提取的人脸。黑色图像表示对应的方法在原图像中未识别出人脸，不是人脸的图像则表示识别人脸错误。

表 2-1 三种方法运行时间、识别率统计表

方法	运行时间	识别率	准确率
OpenCV	0:00:02.572321	19/20=95.0%	15/19=78.9%
Dlib	0:00:14.168938	19/20=95.0%	19/19=100%
Face_recognition	0:00:14.273858	19/20=95.0%	19/19=100%

上表中，运行时间是从开始执行代码程序，到把图像中的人脸全部提取出来所用的时间。识别率是通过识别到人脸的图像数量除以总输入图像数量所得，即图 2-3 中各个部分减去黑色图像的数量除以总数量，皆为 19/20。准确率则是识别正确人脸图像的数量除以识别到人脸的图像数量，即图 2-3 中各个部分人脸图像数量除以非黑图像数量。

根据上表数据可以看出，OpenCV 的识别速度确实很快，但是识别率和准确率相比后两者有很大欠缺。后面两种方法在速度、识别率、准确率方面，从我们测试的这些数据来说不相上下，都表现出了不错的识别率与可靠的准确率。虽然我们测试的数据不算太多，但是大致能够看出不同方法不同方面的优劣好坏。第三种方法代码简单，我们最终采用的是第三种方法。

2.2 视频分帧存图的实现

视频是由一帧接一帧的图像组成的，帧速率则表示每秒钟播放的图像数量，一般有 24fps，25fps，30fps 等。我们要检测视频中人脸的真假，第一步要做的便是将视频按照一定的帧步长，截取图像。

读取视频的每一帧，我们采用 OpenCV 的以下几个函数：

```
capture = cv2.VideoCapture(video_input)
real, frame = capture.read()
cv2.waitKey(1)
capture.release()
```

其中，cv2.VideoCapture()方法是用来获取视频的。若参数是 0，则表示打开计算机的内置摄像头；若参数是文件路径，则表示打开该路径下的视频。

capture.read()方法表示按帧读取视频，real 和 frame 是 capture.read()方法的两个返回值。其中前一个参数是布尔值，如果读取的帧是正确的则返回 True，而如果读取位置到了文件结尾则返回 False。第二个参数就是每一帧的图像数据，是一个三维矩阵。

waitKey()方法本身表示等待键盘输入。参数为 0，即 cv2.waitKey(0)，则只显示当前

帧的画面，相当于将视频暂停；其参数是 1，对于视频而言，则表示等待 1ms 后切换到下一帧的图像；当这个参数过大时，例如 cv2.waitKey(2000)，视频渲染会因为等待时间过久而显得十分卡顿。

实验中采用 frame_step = 10，即每 10 帧存取一张图像，设置循环即可得到视频分帧存图的最终结果。

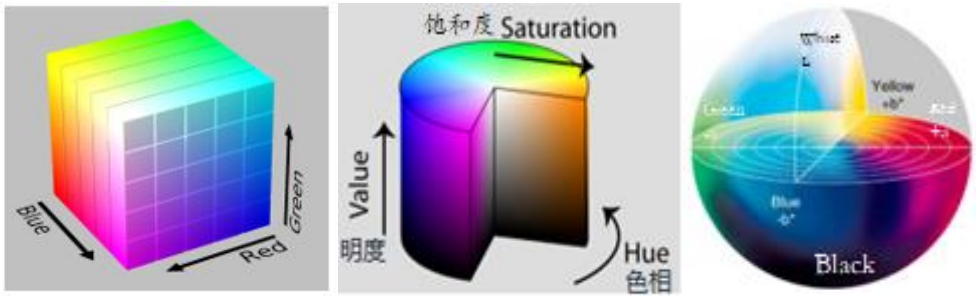
视频库是 Deepfake Detection，是由 Google 和 JigSaw 提供的 Deep-Fake检测数据集。下载地址上文已经给出。该数据集包含来自 28 个不同场景中的演员的 3000多个被操纵的视频。

2.3 颜色直方图简介

要讲颜色直方图[17]错误!未找到引用源.，我们首先要知道颜色模型有哪几种，而这便是了解我们做成统计图和统计表要获取的数据是什么的前提。

图像的颜色模型[19]常见的有以下三种：RGB(Red Green Blue)，HSV(Hue Saturation Value)，Lab(Lightness Channel-a Channel-b)。

1.RGB：该模型我们很容易理解，即图像三原色红(Red) 绿(Green) 蓝(Blue)，取值范围都是[0,255], [0,255], [0,255]。在连续变换颜色时不够直观。其颜色空间模型对应于三维直角坐标系，如图 2-4（a）



（a）RGB 类型颜色空间模型（b）HSV 类型颜色空间模型（c）Lab 类型颜色空间模型

图 2-4 三种不同类型的颜色空间模型

2.HSV：这是为了将图像数字化而提出来的概念，但不能很好地诠释人类眼睛解释图像的过程。H(Hue)色相，表示不同颜色相位，范围[0,360]；S(Saturation)饱和度，表示色彩纯净度，范围[0,1]，0 表示为无彩色纯白色，1 表示纯彩色；V(Value/Brightness)明度，表示亮度深浅，范围[0,1]， 0 表示为最暗纯黑色，1 表示最亮。在 OpenCV 中，HSV 三者范围变为[0,180], [0,255], [0,255]。

HSV 的颜色空间模型对应圆柱坐标系（如图 2-4（b））中的一个圆锥形子集（如图 2-5）。圆锥上底面对应于明度 V=1，包含 R，G，B=1，颜色最亮，顶点 V=0，表示黑色，由顶点沿圆锥母线 V 从 0 到 1 变化；而色相 H 由绕圆锥高线的旋转角给定，红色 R 对应角 0°，绿色 B 对应角 120°，蓝色 B 对应角 240°；饱和度 S 由底面圆心到圆周从 0 到 1 变化，底面圆心表示白色，越往外颜色饱和度越高。

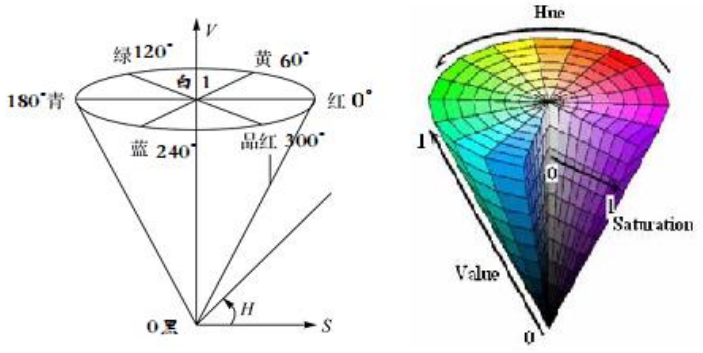


图 2-5 HSV 颜色空间定义：圆锥形子集

BGR 到 HSV 的转换运算如下：

max=max(R,G,B)

min=min(R,G,B)

if R = max, H = (G-B)/(max-min)

if G = max, H = 2 + (B-R)/(max-min)

if B = max, H = 4 + (R-G)/(max-min)

H = H * 60

if H < 0, H = H + 360;

V=max(R,G,B);

S=(V-min)/V

ifV=0，S=0

3.Lab：这是根据颜色之间的欧氏距离的具体含义来定义的（如图 2-4（c））。欧氏距离越大，人眼感觉两颜色差距越大。L 通道：表示像素的亮度(Lightness)从纯黑到纯白，范围是[0,100]，对应到球体中，下面黑色，中间灰色，上面白色；颜色通道 a：表示从绿色到红色，范围是[-128,127]，对应到球体中，左边绿色（负），右边红色（正）；颜色通道 b：表示从蓝色到黄色，范围是[-128,127]，对应到球体中，一端纯蓝（负），另一端

纯黄（正）。

经过对比，在实验中，我们采用最普遍最易于理解的 RGB 三原色颜色模型作为图像的特征，进行提取保存，用于训练判别等。

2.4 SURF 概念原理简介

SURF(Speeded Up Robust Features，加速稳健特征)[20][21][22]，是一种稳健的局部特征点检测与描述算法，其局部特征点的提取、描述和匹配流程有以下几个步骤：构建Hessian矩阵，用于生成所有兴趣点；构建尺度空间；定位特征点；确定特征点的主方向；生成特征点描述子；特征点匹配。SURF 有两大创举：integral images, box filters 在 Hessian 矩阵上的使用，降维特征 Descriptors 的使用，这使得其在执行效率方面有突出表现

1.构建 Hessian 矩阵。这个环节是为了生成图像稳定的边缘点。经过高斯滤波之后，在图像 I 中给定一点 **x** =（x，y），比例尺为 σ 的 **x** 点的 Hessian 矩阵表达式如下：

$$H(x,\sigma)=\begin{bmatrix}L_{xx}(x,\sigma) & L_{xy}(x,\sigma)\\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma)\end{bmatrix} \quad (2\text{-}1)$$

其中$L_{xx}(x,\sigma)$是图像 I 在点 **x** 处高斯二阶偏微分$\frac{\partial^2 g(\sigma)}{\partial x^2}$的卷积，$L_{xy}(x,\sigma)$ 和 $L_{yy}(x,\sigma)$ 含义类似。当 Hessian 矩阵的判别式 Det(H)取得局部最大值时，表示当前点是该区域中最亮或者最暗的点，由此定位关键点。同时，SURF 用 box filters 近似代替 Gaussian filters，可以提高运算速度，最终得到判别式表达式：

$$\mathrm{Det}(H)=D_{xx}\bullet D_{yy}-(0.9D_{xy})^2 \quad (2\text{-}2)$$

y 方向和 xy 方向的 Gaussian filters，box filters 示意图如下。灰色区域等于零。

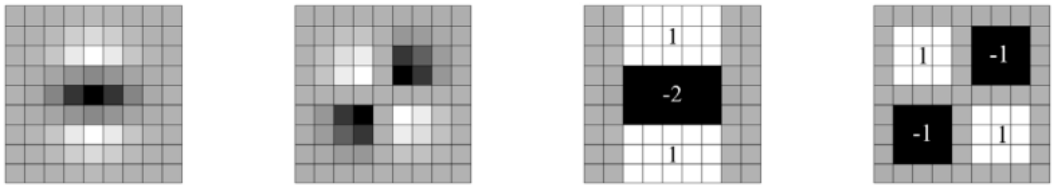


图 2-6 Gaussian filters，box filters 示意图。前两个是 Gaussian filters，后两个是 box filters。

2.构建尺度空间。尺度空间通常被表示成图像金字塔的形式。在 SURF 中，不同组间的图像尺寸一致，但是使用的盒状滤波器(box filters)模板尺寸逐渐增大，同一组不同层之间滤波器的尺寸相同而模糊系数逐渐增大。

3. 特征点定位。该环节将经过 Hessian 矩阵处理的各个像素点，与二维图像空间和尺度空间邻域内的 26 个点进行比较筛选，初步确定关键点位置。

4.主方向的确定。SURF 算法采用统计上面得出的特征点邻域内 Haar 小波特征的方法来确定主方向。即统计 π/3 的旋转扇形窗口内，水平方向和垂直方向 Haar 小波特征的总和。然后以 0.2 弧度角旋转扇形区域，再次进行统计特征总和工作。最后比较哪个扇形的方向最大，将其作为该特征点的主方向。该过程如图 2-7 所示。

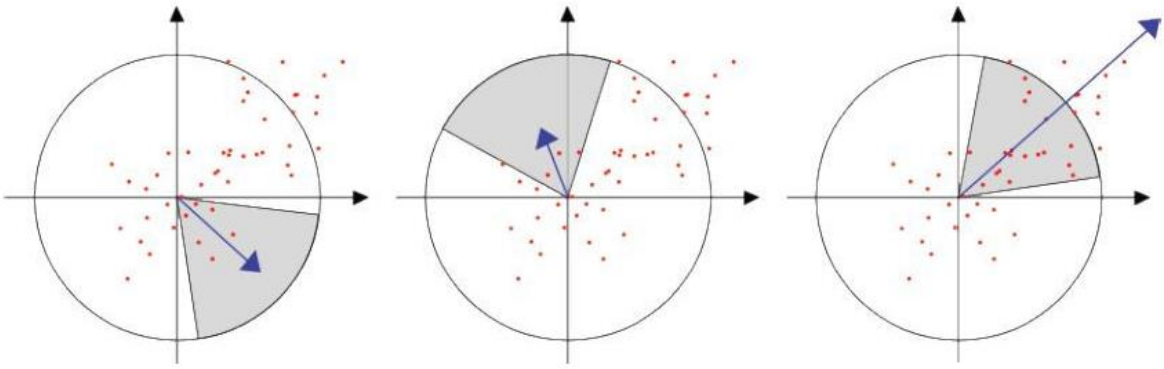


图 2-7 主方向的确定过程示意图

5.生成特征点描述向量。为了得到描述向量，第一步则是在兴趣点的周围，以其为中心建立一个矩形区域，而方向则沿着上文得到的关键点的主方向。然后把该矩形区域规则地划分为 4×4 个小的子区域，每个子区域统计 5×5 个像素的水平与垂直 Haar 小波的特征和，因此，在每个子区域内用一个四维的描述向量 **v** 来描述其强度变化模式，**v** =（∑dx，∑dy，∑|dx|，∑|dy|）。最终，所有的 4×4 个子区域内的描述向量组合在一起，就构成了 4×4×4=64 维描述向量。

6.特征点匹配。SURF 是通过计算两个关键点之间的欧氏距离(Euclidean distance)来确定匹配度的，距离越短表示匹配度越好。而且 SURF 还加入了 Hessian 矩阵迹的判断，这使得关键点的匹配更为靠谱。即若两关键点的 Hessian 矩阵迹的正负符号相同，则二者具有相同方向的对比度变化；若符号相反，则二者对比度变化方向相反，即使欧氏距离为 0 也予以排除。

经过以上的几个步骤，我们便会得到图像上的一些 SURF 特征点特征向量信息。

2.5 ELA 概念原理简介

ELA(Error Level Analysis 误差等级分析)[23]，适用于 JPEG 图像。JPEG 格式的图像

具有如下特点：保存图像时各个地方都具有相同的质量等级，倘若图像的某一部分有非常突出的质量等级，则它可能被数字化更改过，我们通常将其用亮度的明暗表示。这便是 ELA 的原理。

第 n1 行第 n2 列的像素点的误差等级 ELA(n1, n2)可以这么计算[24]：

$$ELA(n1, n2) = | X(n1, n2) - Xrc(n1, n2) | \quad (2\text{-}3)$$

其中，X 是原图像(n1, n2)点的灰度值，Xrc 是重压缩保存后(n1, n2)点的灰度值。对于彩色图，我们可以用以下公式计算：

$$ELA(n1, n2) = \frac{1}{3}\sum_{i=1}^{3} |X(n1, n2, i) - Xrc(n1, n2, i)| \quad (2\text{-}4)$$

对于 RGB 图像，i = 1 , 2 , 3 , 分别表示对应的颜色通道。

下面举个例子讲述 ELA 的具体实现过程：将一幅 JPEG 格式的质量等级为 80%的原图像 a（注：由于篡改部分只保存了 1 次，所以质量等级较高为 90，而未篡改区域被保存了 2 次，所以质量等级较低为 80），以 90%的质量等级重新保存为 JPEG 格式的临时图像 b，然后图像 a,b 逐像素做差，得到做差图像 c，并由差集得到增强因子高亮差异，得到 ELA 图像 d，如图 2-8 所示。原图 a 大片相同质量等级的地方，即未篡改的区域，与图像 b 的差基本相等，表现为基本近似的较暗颜色值（对于灰度图就是灰度值），如图 2-8 d) 大片的灰色区域所示；而修改过的地方则颜色异常，与众不同，如图 2-8 d) 少部分明亮的白色区域所示。

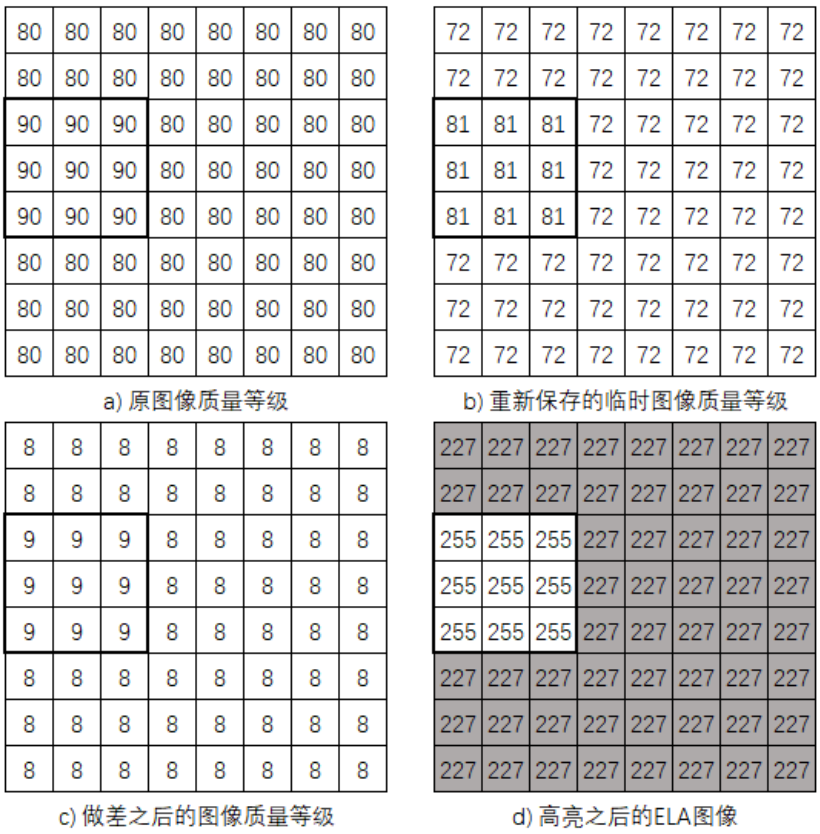


图 2-8 ELA 示例图

总结下来，ELA 有以下几步[25]：

1.重新保存原始图像，并压缩输入图像以根据指定的质量因子生成新图像。

2.逐像素计算两幅图像之间差的绝对值，并生成差集图像。

3.根据差集的最大像素值，得到增强因子。

4.根据增强因子调整差集图像的亮度，生成最终的增强 ELA 图像。

2.6 SVM 概念原理简介

支持向量机 (Support Vector Machines, SVM)[26][27]，是建立在统计学基础上的数据处理方法，能够很好地处理回归问题和分类问题。SVM 算法要解决的问题就是寻找一个可以直接用于分类的最优分界面（我们将其称为超平面）。关于该平面，要求有两点：一是该平面能够保证分类精度，即能够进行准确的分类判断；二是该平面两侧的空白域达到最大，即分界面分别到两种类别的临界面的距离空间达到最大。理论上 SVM 能够实现对线性可分数据的最优分类，而通过运用核函数变换运算，则可将其进行拓展使用，将高维非线性数据转化为线性可分类的问题。

2.6.1 线性可分

下面，我们以可线性分类数据的二分类为例，给定训练样本集$(\boldsymbol{x}_i, y_i)$, i = 1,2,⋯,l, $\boldsymbol{x} \in R^n, y \in \{\pm 1\}$，超平面可以记作$(\boldsymbol{w} \bullet \boldsymbol{x}_i)+b=0$，为满足上文提到的两点要求：分类精度高且两侧空白大，就必须满足以下约束 $y_i[(\boldsymbol{w} \bullet \boldsymbol{x}_i)+b] \geqslant 1$, i = 1,2,⋯,l, 可得分类间隔为 $2/||\boldsymbol{w}||$，它要最大，可认为要使$||\boldsymbol{w}||^2/2$ 最小，即：

$$\min \Phi(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 = \frac{1}{2}(\boldsymbol{w}' \bullet \boldsymbol{w}) \quad (2\text{-}5)$$

为解决该优化问题，在此，我们引入拉格朗日函数：

$$L(w,b,a) = ||w||/2 - a(y((w \bullet x)+b)-1) \quad (2\text{-}6)$$

式中 ai>0 为拉格朗日乘数，最终将凸二次规划问题转化为相应的对偶问题，得到最优解 $\boldsymbol{a}^* = (a_1^*, a_2^*, \cdots, a_l^*)^T$。由 $\boldsymbol{a}^*$得 $\boldsymbol{w}^*$，b^*：

$$w^* = \sum_{j=1}^{l} a_j^* y_j x_j \quad (2\text{-}7)$$

$$b^* = y_i - \sum_{j=1}^{l} y_j a_j^* (x_j \bullet x_i) \quad (2\text{-}8)$$

最终得到最优分类超平面函数：

$$f(x)=sgn\{(w^* \bullet x)+b^*\}=sgn\{(\sum_{j=1}^{l} a_j^* y_j (x_j \bullet x_i))+ b^*\}, \quad x \in Rn \quad (2\text{-}9)$$

2.6.2 线性不可分

对于线性不可分数据，SVM 的解决办法则将输入向量进行变换，映射到一个高维向量空间，在该空间我们可以进行线性运算，即在该特征空间我们可以构造最优超平面：

$$x \rightarrow \Phi(x)=(\ \Phi 1(x),\ \Phi 2(x),\ \cdots,\ \Phi l(x))T \quad (2\text{-}10)$$

$\boldsymbol{\Phi(x)}$代替 $\boldsymbol{x}$，得到最优分类超平面函数：

$$f(x) = sgn(w \bullet \Phi(x)+b)=sgn\{\sum_{i=1}^{l} a_i y_i \boldsymbol{\Phi}(\boldsymbol{x_i}) \bullet \boldsymbol{\Phi}(\boldsymbol{x})+ b\} \quad (2\text{-}11)$$

这样，训练样本之间仅仅是内积运算，高维空间会简化不少运算量。

2.6.3 其他相关概念

SVM 中还会涉及核函数、松弛变量、惩罚因子、多分类等相关概念[28][29]。

核函数的基本作用就是连通低维与高维的纽带，它接收两个低维空间里的向量，经

过某个变换后，计算出在高维空间里的向量内积。

松弛变量的作用就是为了拟合一些在两条分界线中间的点，处理那些极少数与众不同的离群点。

惩罚因子 C 则表示我们对离群点所带来损失的重视程度，当所有松弛变量的和一定时，给定的惩罚因子 C 越大，对目标函数的损失值也就越大，也就表示我们不愿意放弃这些离群点，我们要重视这些点的影响。

多分类指的是目标分类不止两个，而是多个，这就需要设计问题优化过程了。

在此，我们了解其含义即可，不再过多一一展开叙述。

2.7 总体实验流程图

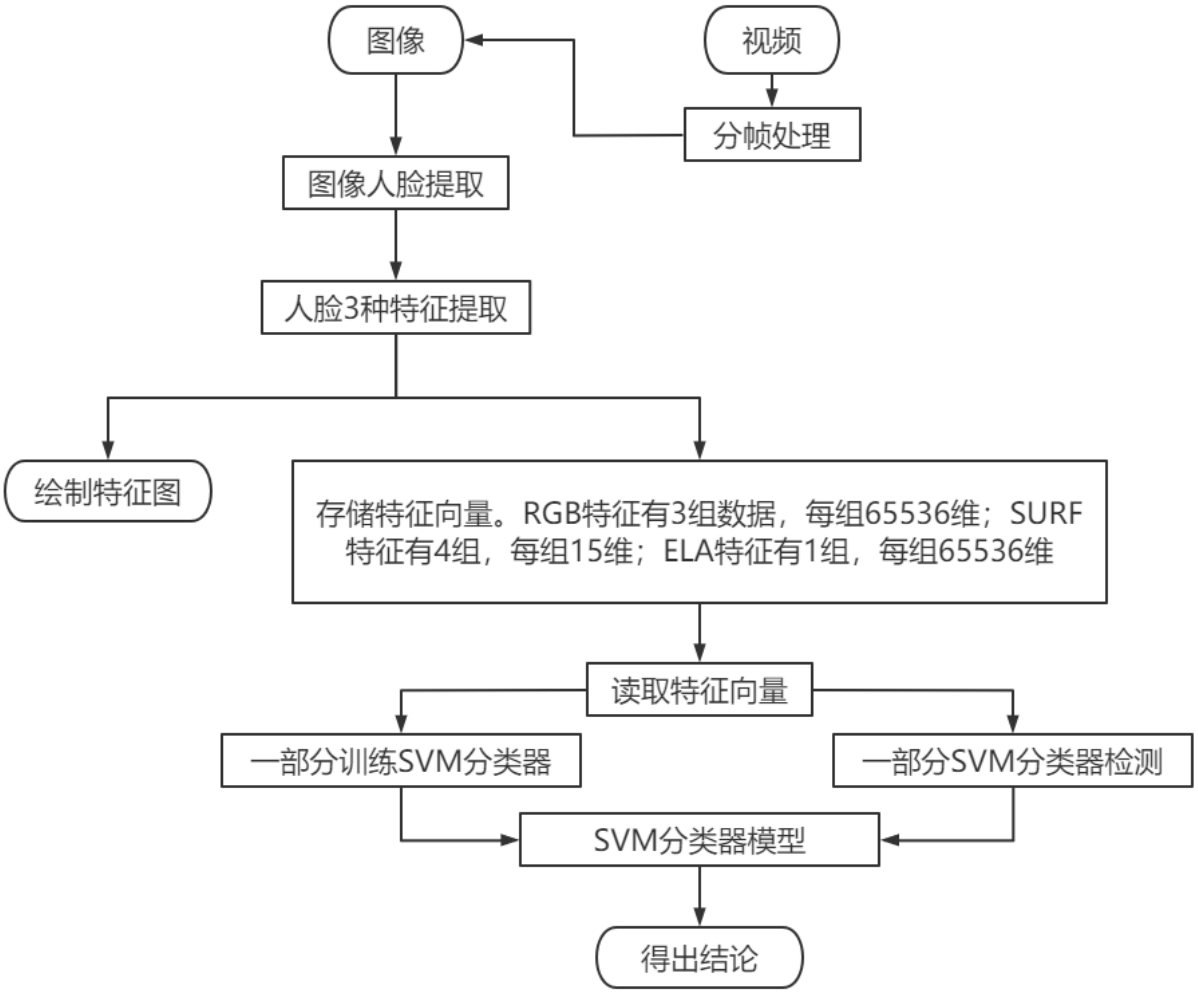


图 2-9 总体实验流程图

本实验具体步骤如下：

①视频分帧处理得到图像；

②对图像进行人脸检测得到人脸区域图像；

③3 种特征向量的提取；

④将特征数据绘制到图像中进行观察；

⑤将特征数据存储到 Excel 文件中。每幅图像像素大小 256×256，RGB 特征包含三个颜色通道数据，为 3 组数据，每组 256×256=65536 维；SURF 特征包含特征点横、纵坐标，角度，直径数据，共 4 组数据，每组数据取直径最大的 15 个点为 15 维；ELA 特征已经将图像转为灰度图，只有高亮后的灰度值 1 组数据，该组数据 256×256=65536 维；

⑥读取特征数据进行 SVM 分类器模型训练；

⑦读取特征数据输入已经训练好的 SVM 分类器模型进行检测，输出检测结果。

特征提取与分类

3.1 颜色直方图代码实现介绍

要描绘颜色直方图，首先就要提取图像各个像素点的三原色，即红绿蓝的值，接下来就是将其绘制成为折线图，横坐标用于表示某种颜色的亮度，取值范围[0,255]，纵坐标表示该亮度下的所有像素点的数量，将红绿蓝三种颜色绘制在同一张图表中，便构成了颜色直方图。

由于 OpenCV 提取三原色顺序是 BGR，即蓝绿红，所以绘图时要注意顺序。提取三原色并绘图的核心代码如下：

```
def draw_color_histogram(inputpath,outputpath):
    image = cv2.imread(inputpath)
    # 将图像颜色信息分为 3 个信道
    channels = cv2.split(image)
    colors = ('b', 'g', 'r')
    plt.figure()
    # 给图表起标题，x 坐标名称，y 坐标名称
    plt.title("RGB Histogram")
    plt.xlabel("Bins")
    plt.ylabel("# of Pixels")
    # 循环绘图
    for (channels, color) in zip(channels, colors):
        hist = cv2.calcHist([channels], [0], None, [256], [0, 255])
        plt.plot(hist, color = color)
        plt.xlim([0, 256])
    # 保存图表，展示图表
    plt.savefig(outputpath)
    plt.show()
    plt.close()
```

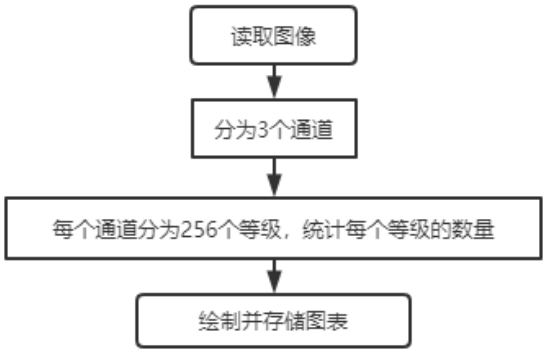


图 3-1 绘制颜色直方图流程框图

在下面的代码中，我们会将图像的三原色信息提取并保存至 Excel 文件之中，每 10 幅图像的特征信息组成 1 个 Excel表格，每 1 幅图像占 1 个 sheet，该sheet 名便取自图像名称，每个 sheet 有 3 列，即 BGR这 3 列。

```
def extract_color_data(inputpath, outputpath):
    image = cv2.imread(inputpath)
    B, G, R = cv2.split(image)
    # 数组展平
    b = B.ravel()
    g = G.ravel()
    r = R.ravel()
    # 矩阵转置
    channels = list(zip(b, g, r))
    colors = ['b', 'g', 'r']
    # 给列命名 bgr，给 sheet 命名图像文件名，将数据写入 Excel 文件
    dt = pd.DataFrame(channels, columns=colors)
    sheet_name = inputpath.split('/')[-1][:-4]
    dt.to_excel(outputpath, sheet_name=sheet_name, index=0)
```

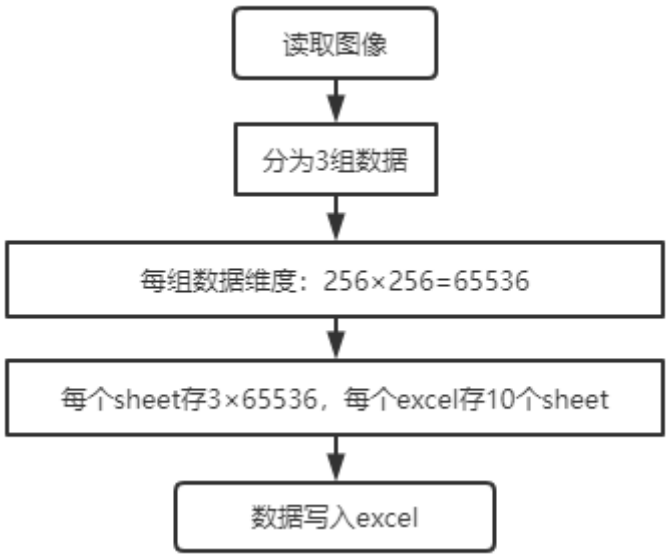


图 3-2 存储三原色特征数据流程框图

3.2 SURF 代码实现介绍

关于 SURF 特征，提取其特征向量，首先要创建一个 SURF 对象，创建时要设置好

一些参数，然后就要寻找特征点，最后便是绘制这些特征点。下面我们来介绍实现以上3个过程所需要的3个核心函数：这些函数包含于opencv-python 3.4.2.16版本中，由于新版本中创建 SURF 对象的函数已经商业化不能免费使用了，所以，我们在此介绍opencv-python 3.4.2.16版本中的函数。

1.创建一个SURF对象函数：

```
cv2.xfeatures2d.SURF_create(hessianThreshold, nOctaves, nOctaveLayer
s, extended, upright)
```

参数说明：

hessianThreshold：Hessian矩阵，默认为100；

nOctaves：表示金字塔的组数，默认为4；

nOctaveLayers：每组金字塔的层数，默认为3；

extended：扩展描述符标志，默认值为False，表示使用64个元素描述符，而True则表示使用128个元素描述符；

upright：垂直向上或旋转的特征标志，默认值为False，表示计算方向，而True则表示不计算方向。

该函数返回值为一个SURF对象。

2.寻找SURF特征点的函数：

```
surf.detect(img, mask)
```

参数说明：

image：输入的图像；

mask：寻找特征点的区域，mask为None则全图寻找。

返回值为特征点列表，包含特征点位置、角度、半径等信息。

3.绘制特征点函数：

```
cv2.drawKeypoint(image, keypoints, outImg, color, flags)
```

image：输入的将要绘制特征点的图像；

keypoints：上文中获得的特征点集kps；

outImage：输出图像文件，None表示无输出；

color：绘制特征点的颜色；

flags：绘制点的模式，有以下四种：DEFAULT＝0，对于每个关键点，只绘制中心点（即描绘的关键点只有位置信息而不含有尺寸方向等信息）；DRAW_OVER_OUTIMG

= 1，在输出图像的现有内容上绘制匹配项而不会创建输出图像矩阵；NOT_DRAW_SINGLE_POINTS = 2，不描绘单个特征点；DRAW_RICH_KEYPOINTS = 4，绘制含有特征点位置、大小、方向信息的圆。

我们选择 flags=4，这是最能显示特征所有属性的一种绘制方式。绘制 SURF 特征点图像的核心代码如下：

```
def draw_SURF(inputpath,outputpath):
    img = cv2.imread(inputpath)
    surf = cv2.xfeatures2d.SURF_create(4000)
    kp = surf.detect(img, None)
    img2 = cv2.drawKeypoints(img, kp, None, (0, 255, 0), 4)

    plt.figure()
    plt.imshow(img2[:,:,::-1])
    plt.title('SURF Threshold=4000')
    plt.axis('off')
    plt.tight_layout()
    plt.savefig(outputpath)
    plt.show()
    plt.close()
```

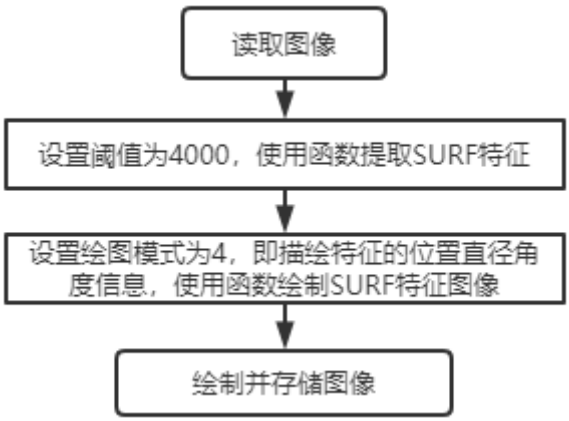


图 3-3 绘制 SURF 特征图像流程框图

上面代码也只是绘制了含有特征的图像，对于计算机而言，下面的提取特征数据更为有用，该代码将特征点集 kps 提取出来，然后循环获得每个特征点的坐标、角度、尺寸，分别写入 Excel 文件中，与上面的 color 特征一样，每 10 张图组成一个 Excel 文件，每张图像占一个 sheet，每个 sheet 分 4 列：x, y, angle, diameter。与上面一个特征的不同点是，每张图像的特征点数量并不一样，刚提取出来时，排列顺序也并非我们所想的按照直径大小来排。所以，一是为了后面的 SVM 训练方便不报错，二是为了尽可能将大

特征点保留，我们又做了两步数据再处理：统一特征点为 15 个并按照直径大小降序排列。

```
def extract_SURF_data(inputpath, outputpath):
    img = cv2.imread(inputpath)
    surf = cv2.xfeatures2d.SURF_create(4000)
    kps, features = surf.detectAndCompute(img, None)
    kps_data = []
    for kp in kps:
        # 关键点的 X、Y 坐标，从左到右 0~255，从上到下 0~255；关键点角度；关键点直径。
        kps_data.append([kp.pt[0], kp.pt[1], kp.angle, kp.size])
    # 统一特征点为 15 个
    kps_data_len = len(kps_data)
    if kps_data_len < 15:
        for i in range(15-kps_data_len):
            kps_data.append([0, 0, 0, 0])
    elif kps_data_len > 15:
        del kps_data[15:]
    # 按照直径排序
    kps_data.sort(key=lambda x: x[3], reverse=True)
    titles = ['x', 'y', 'angle', 'diameter']
    dt = pd.DataFrame(kps_data, columns=titles)
    sheet_name = inputpath.split('/')[-1][:-4]
    dt.to_excel(outputpath, sheet_name=sheet_name, index=0)
```

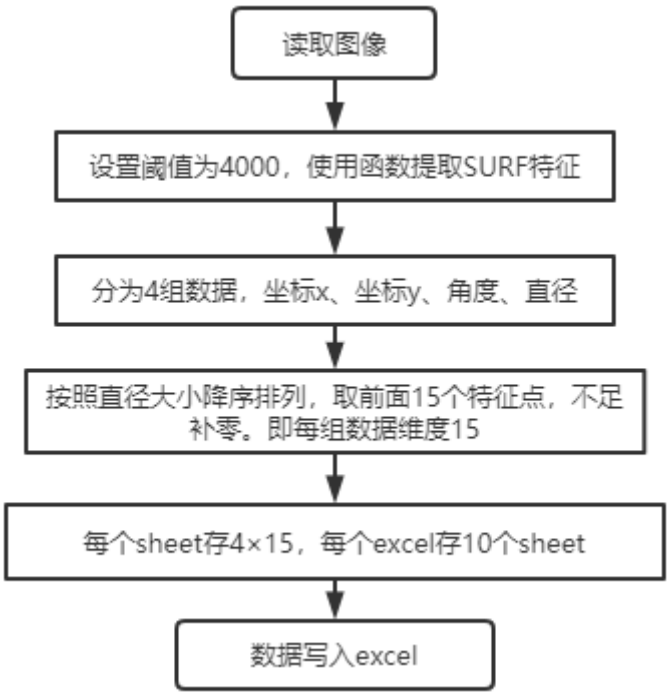


图 3-4 存储 SURF 特征数据流程框图

3.3 ELA 代码实现介绍

根据 ELA 原理，我们要将该特征提取过程分为如下几步：首先，以一定的压缩率将原 JPEG 图像重新存为临时的 JPEG 图像，然后，原图像与临时图像做差，最后，将差值高亮化，存为新的图像，即为含有 ELA 特征的图像。我们将要用到 PIL 库中的 Image, ImageChops, ImageEnhance。核心代码如下：

```
def draw_ELA(inputpath, outputpath):
    image = Image.open(inputpath)
    # 新压缩率存图
    image.save(tmp_path, 'JPEG', quality=quality_level)
    tmp_image = Image.open(tmp_path)
    # 做差
    ela_image = ImageChops.difference(image, tmp_image)
    # 计算增强因子
    scale = 10*255/max_diff
    # 高亮绘图
    ela_image = ImageEnhance.Brightness(ela_image).enhance(scale)
    ela_image.save(outputpath)
```

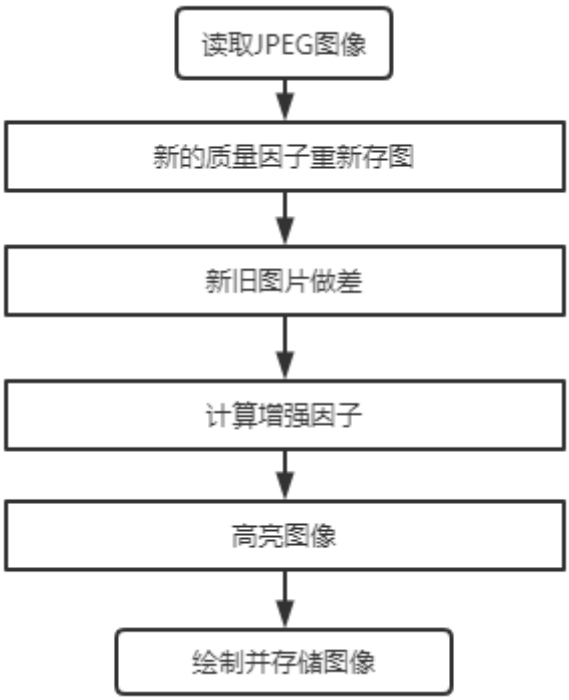


图 3-5 绘制 ELA 特征图像流程框图

类似前两种特征的处理，我们再将 ELA 数据存至 Excel 表中。而为了后续的训练

SVM 分类器方便，我们首先将图像灰度化处理，然后再进行 ELA 特征数据提取，最后将数据存入 Excel 的不同 sheet 中，每个 sheet 大小 255*255。核心代码如下所示：

```
def extract_ELA_data(inputpath, outputpath, outputfile):
    # 图像灰度化处理
    image_gray = image.convert('L')
    image_gray.save(tmp_path, 'JPEG', quality=quality_level)
    tmp_image = Image.open(tmp_path)
    # 做差
    ela_image = ImageChops.difference(image_gray, tmp_image)
    # 循环获取像素值
    width, height = ela_image.size[0], ela_image.size[1]
    pixel = []
    pixel_data = []
    for h in range(0, height):
        for w in range(0, width):
            pixel.append(ela_image.getpixel((w, h)))
        pixel_data.append(pixel)
        pixel = []
    # list 数据类型转为 DataFrame 数据类型，存入 Excel
    dt = pd.DataFrame(pixel_data)
    sheet_name = inputpath.split('/')[-1][:-4]
    dt.to_excel(outputfile, sheet_name=sheet_name, index=0)
```

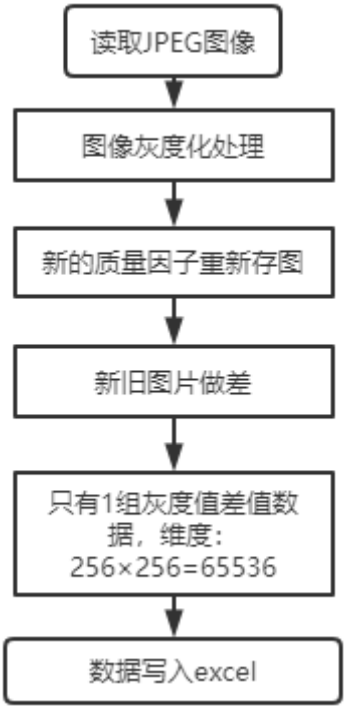


图 3-6 存储 ELA 特征数据流程框图

3.4 SVM 代码实现介绍

我们可以使用 sklearn 中的已有函数，进行 SVM 分类器的训练和测试。进行训练之前，我们首先要获取 Excel 表中的特征数据，每张图像为 1*n 数据集，而每 10 张图像的 Excel 表构成 X，之后每 1 个 Excel 又给 X 添加 10 个元素，直到 X 长度为 100 再进行训练，即每 100 张图像训练一次。在给 X 赋值的同时，我们也要给 Y（X 中每个元素的真假，也就是每张图像对应的真伪信息）赋值，1 表示真（原始图像，未篡改），0 表示假（该图像被修改过）。

sklearn 库中有关 SVM 的两个函数及其几个重要参数的简介如下：

```
svm.SVC([C, kernel, degree, gamma, coef0, …])
```

C：正则化参数，严格正数，表示惩罚因子；

kernel：指定要在算法中使用的核函数。上文中我们了解到核函数的作用，可将线性不可分问题转化为线性可分。它必须是以下几种：“rbf”、“linear”、“poly”、“precomputed”、“sigmoid”或可调用的函数。若未给出，则将使用默认的“rbf”。若给定了一个可调用的函数，它将用于从数据矩阵中预算出核矩阵用于拟合计算。

degree：表示多项式核（“poly”）的次数。而所有其他核函数忽略不计该参数。

gamma：“rbf”、“poly”和“sigmoid”的核系数。如果 gamma='scale'（默认值），则使用 1/(n_features*X.var())作为 gamma 的值；如果 gamma='auto'，则使用 1/ n_features。

……

```
linear_model.SGDClassifier([loss, penalty, …])
```

loss：要使用的损失函数。默认为’hinge’，它提供线性支持向量机。可选项有：‘hinge’, ‘log’, ‘modified_huber’, ‘squared_hinge’, ‘perceptron’, 或回归损失: ‘squared_loss’, ‘huber’, ‘epsilon_insensitive’, ‘squared_epsilon_insensitive’。

penalty：惩罚因子，表示要使用的惩罚（也称为规则化术语），可选项有：{‘l2’, ‘l1’, ‘elasticnet’}。默认值为'l2'，这是线性支持向量机模型的标准正则化器。而 ‘l1’ 和‘elasticnet’很可能会给训练的模型带来稀疏性，而'l2'则无法实现该功能。

……

在第二个分类器运用了随机梯度下降算法[错误!未找到引用源。][31]，其中，有两种训练算法：fit 和 partial_fit。

第一次训练时，两种方法训练模型的原理本质上其实是一样的。

但是对于后续数据的训练，二者差异明显。在每当我们有了新数据时，fit 方法则会直接覆盖之前的模型，生成匹配新数据的新的模型，达不到预期的多次训练的效果；而 partial_fit 方法可以在之前训练好的旧的模型基础上用新的数据再次训练，更新修正模型而非替换模型。因此，综合考虑之后我们采用 SGDClassifier 的 partial_fit 训练方法。

下面是训练 SVM 分类器核心代码：

```
    # 获取每个 sheet，即每张图的所有信息，展平为 1 维
    for sheet_name in sheet_names:
        sheets[i] = excel_file[sheet_name].values  # DataFrame->nump
y.ndarray
        sheets_1dim[i] = sheets[i].flatten()
        i += 1
    # 用特征数据训练 SVM 分类器
    X = true_data_train + fake_data_train
    clf = SGDClassifier()
    clf.partial_fit(X, Y, classes=np.array([0, 1]))
    joblib.dump(clf, savepath + '/' + 'clf.pkl')
```

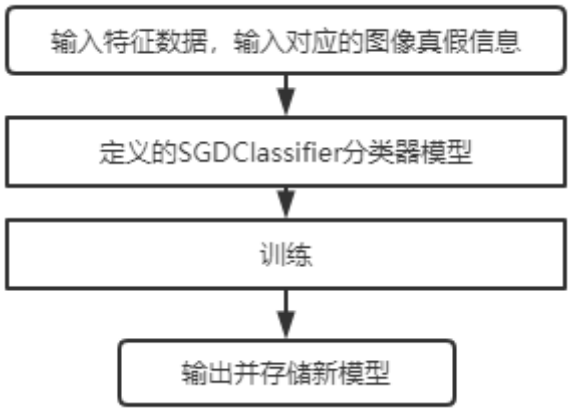
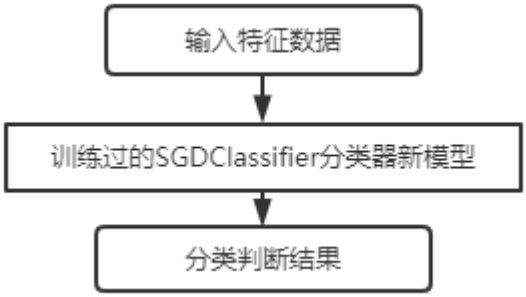


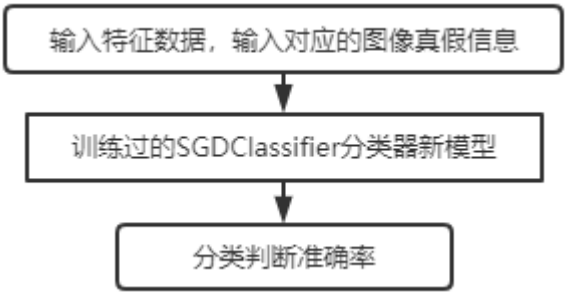
图 3-7 SVM 分类器训练流程框图

下面是 SVM 分类器预测数据核心代码：

```
    # 用已训练好的 SVM 分类器测试
    X = true_data_test + fake_data_test
    clf2 = joblib.load(savepath+'/'+'clf.pkl')
    Z = clf2.predict(X)
    accuracy = clf2.score(X, Y)
    print('测试数据实际真假：{}'.format(Y))
    print('测试数据预测真假：{}'.format(Z))
    print('clf 预测准确率：{}'.format(accuracy))
```



a) 分类判断结果



b) 分类判断准确率

图 3-8 SVM 分类器测试流程框图

实验结果与分析

4.1 视频分帧存图结果展示

将视频按照一定的帧间隔分开并存为图像，是提取人脸前的预处理工作，在论文 2.2 章节中我们已经对其原理和实现进行了详细的说明。下面是将 DFD 数据库中两个文件夹（original_c23 和 attack_c23，分别对应原视频和 Deepfake 之后的视频）中的视频以 10 为帧间隔，分帧存图后的成果，每个视频对应一个文件夹，文件夹以视频名称命名，其中的图像以帧序号图像序号命名：

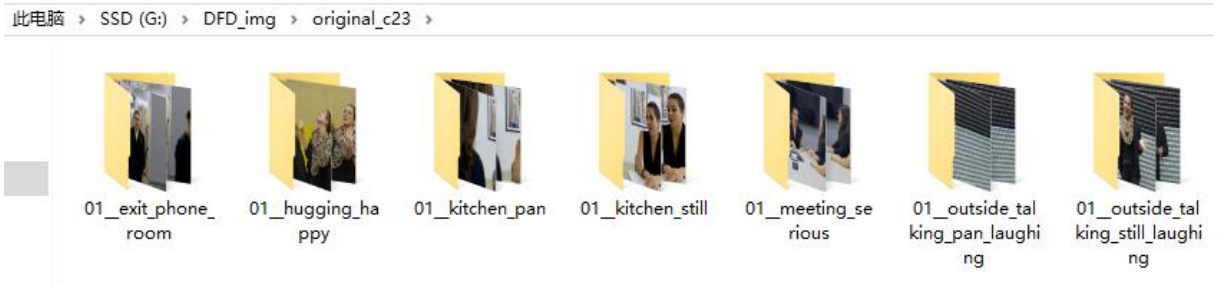


图 4-1 original_c23 文件夹视频分帧存图结果

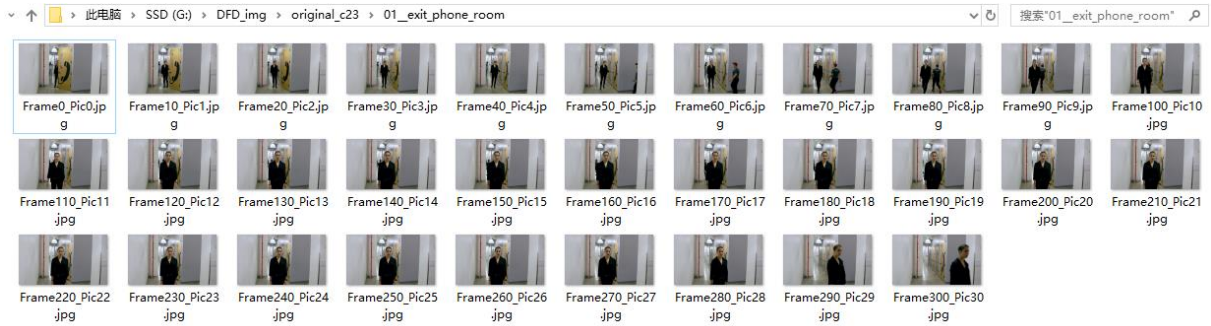


图 4-2 original_c23 其中一个视频分帧存图结果

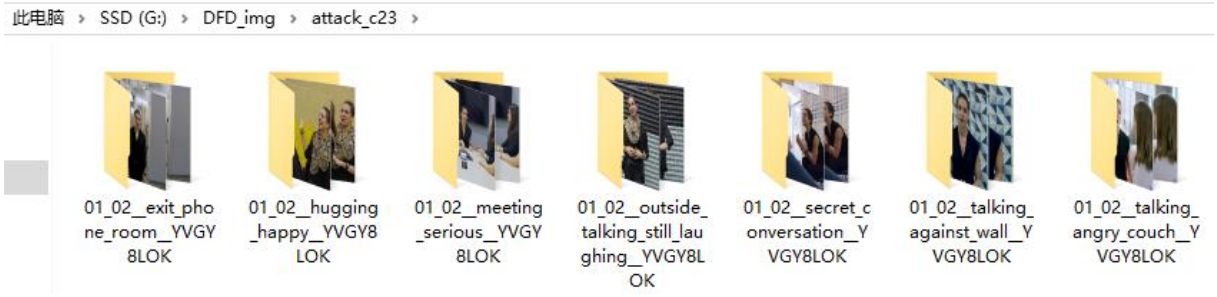


图 4-3 attack_c23 文件夹视频分帧存图结果

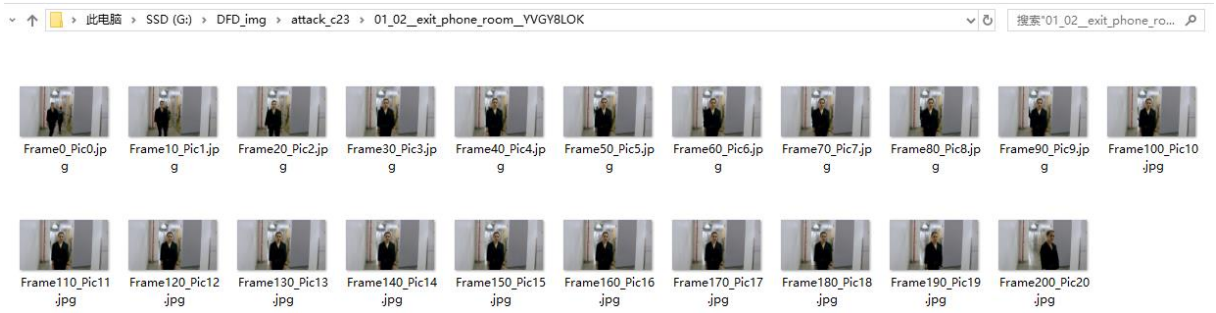


图 4-4 attack_c23 其中一个视频分帧存图结果

4.2 人脸提取结果展示

尽管 Face_recognition 的人脸识别率和准确率都比较高，但是还是有部分图像中的人脸未能识别出来，下图展示了原图像和识别出来的人脸，可见原图文件集中的 000089.jpg, 000124.jpg, 000128.jpg 这 3 张图像，未曾被识别出人脸。仔细观察不难发现，这是由于原图中的人脸有些是有遮挡物的，有些是头部有过度的倾斜。但是由于未识别的图像占据很少一部分，且识别出来人脸的图像人脸区域提取的准确率接近 100%，所以在后续实验中，我们直接使用 Face_recognition 提取出来的人脸。

图 4-5 人脸提取结果展示。绿色实线上面 1,2,3 行是原始图像，下面 4,5,6 行是

Face_recognition 提取的人脸；红色实框圈起来的 3 个图像为未识别到人脸的图像，下面三行无对应提取的人脸图像

4.3 颜色直方图特征提取结果展示

关于颜色特征，我们首先提取三原色数据，然后绘制直方图，最后将数据存入 Excel 表格。

Celeba 真脸图像及其颜色直方图如图 4-6，作为对比，PGGAN 假脸图像及其颜色直方图如图 4-7。DFD 原始图像及其颜色直方图如图 4-8，作为对比，生成的假脸图像及其颜色直方图如图 4-9。

通过观察 Celeba 和 PGGAN 的颜色直方图，我们不难发现：原始的、真正的人脸图像的颜色直方图三原色的峰值基本是错开的，而假脸的三原色峰值比较接近。通过观察 DFD 原始脸和变换脸的颜色直方图，我们发现上述的规律很难发现，原因可能是 Deepfake 技术比较强大，所以我们肉眼难以分辨真伪，三原色方面假脸会和真脸图像展现出比较接近的特点，即三原色峰值错开。

保存了三原色信息的 Excel 表格如图 4-10 所示，每幅图像存为一个 sheet，每个 sheet 包含 b,g,r 共 3 列，65536 行。

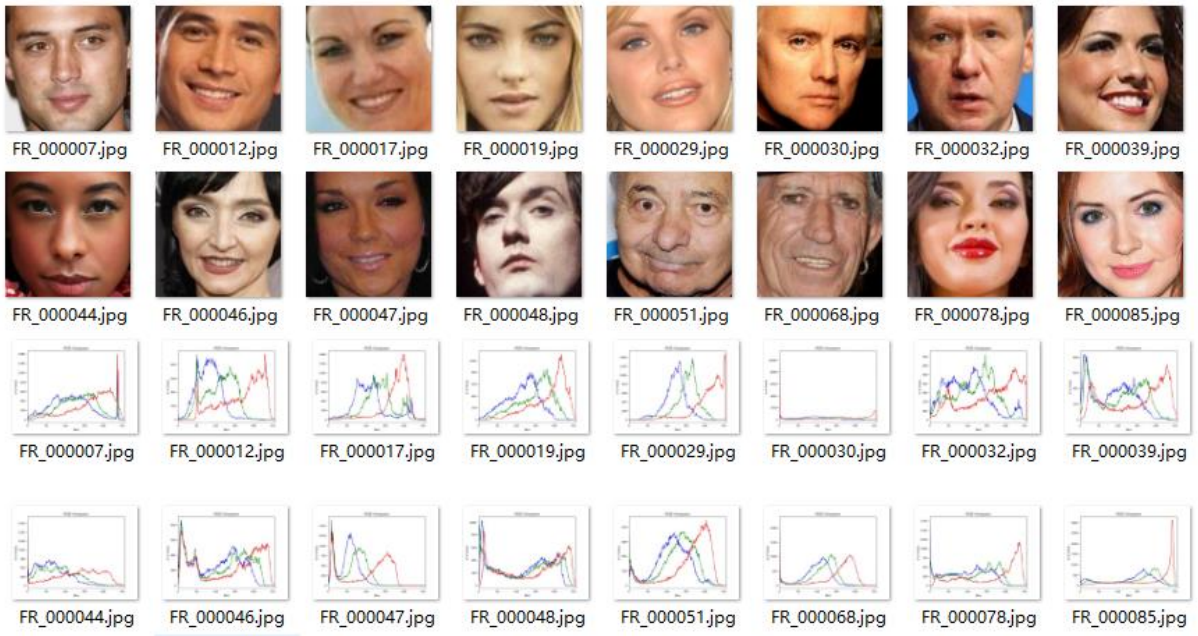


图 4-6 Celeba 真脸图像及其颜色直方图

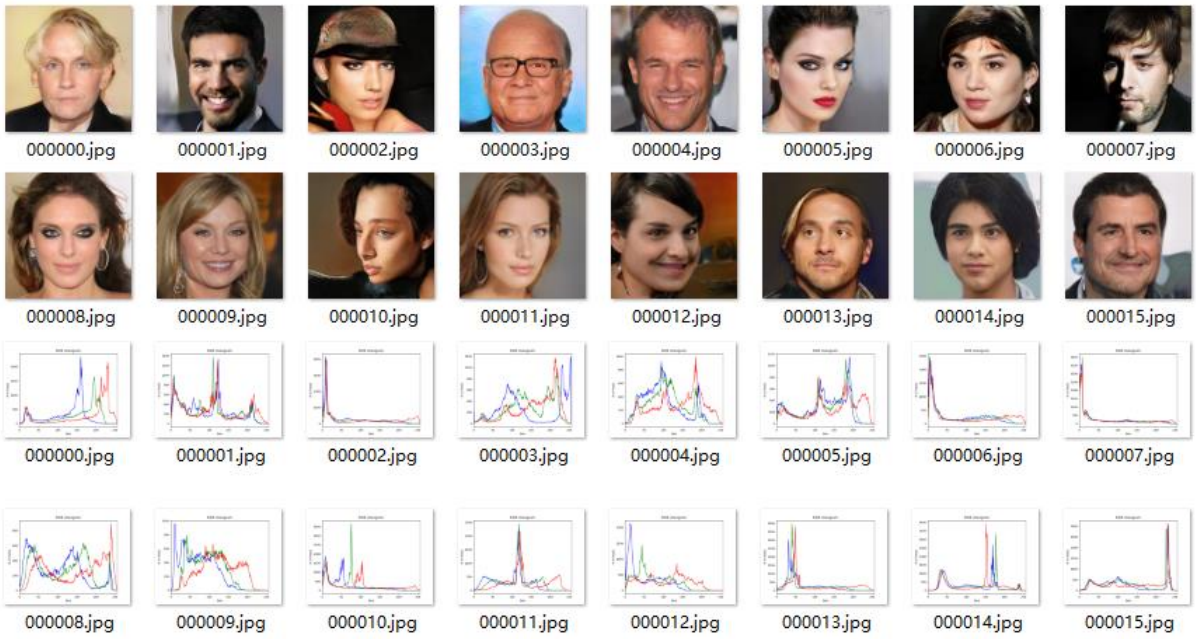


图 4-7 PGGAN 假脸图像及其颜色直方图

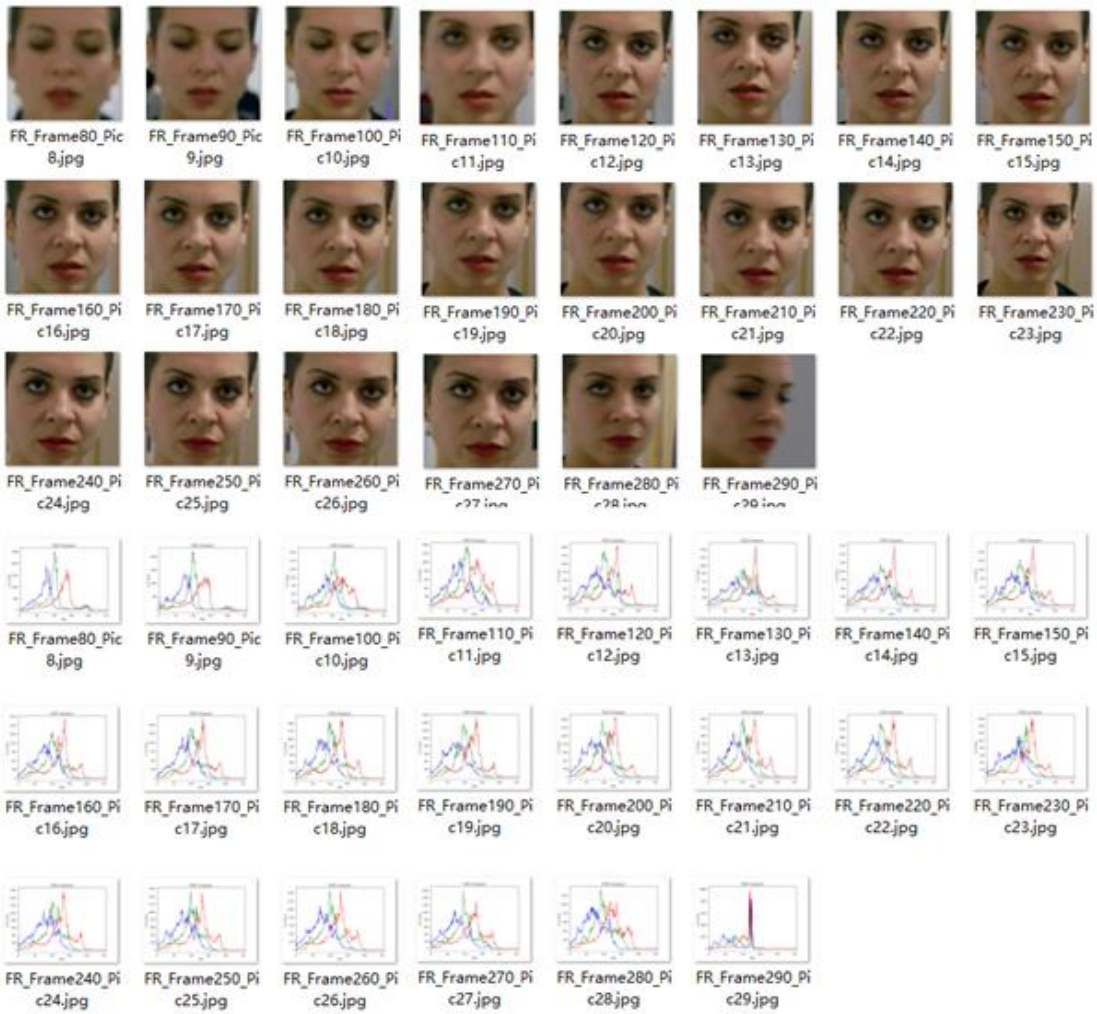


图 4-8 DFD 原始图像及其颜色直方图

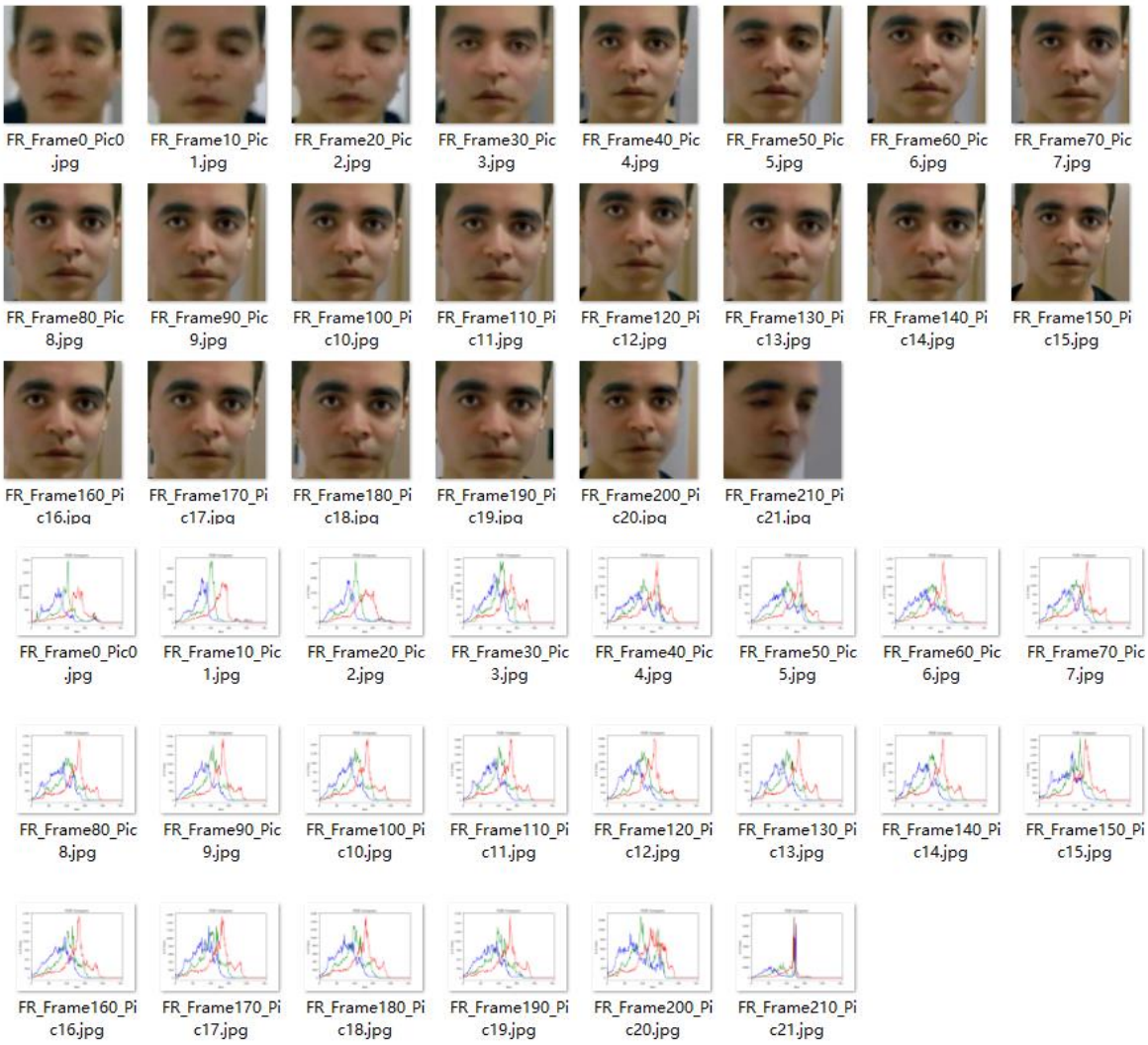


图 4-9 DFD 假脸图像及其颜色直方图

	A	B	C	D	E
1	b	g	r		
2	19	43	49		
3	19	43	49		
4	19	43	49		
5	20	44	50		
6	20	46	53		
7	22	48	55		
8	24	49	59		
9	25	50	60		
10	26	52	64		

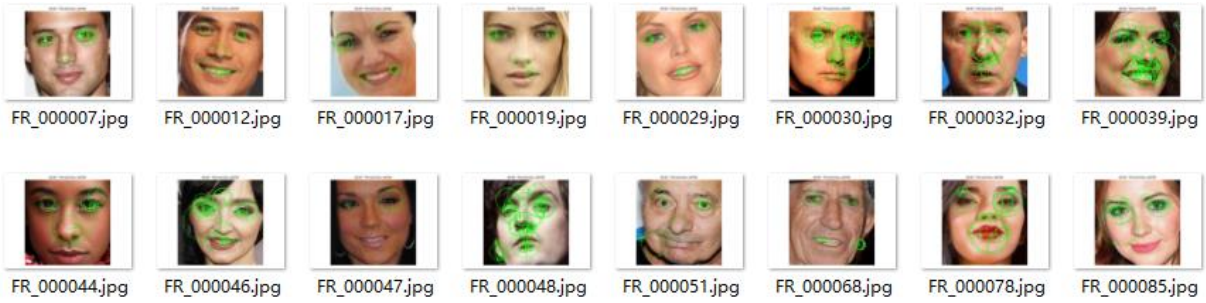
FR_Frame0_Pic0 FR_Frame100_Pic10 ...

图 4-10 保存了三原色信息的 Excel 表格

4.4 SURF 特征提取结果展示

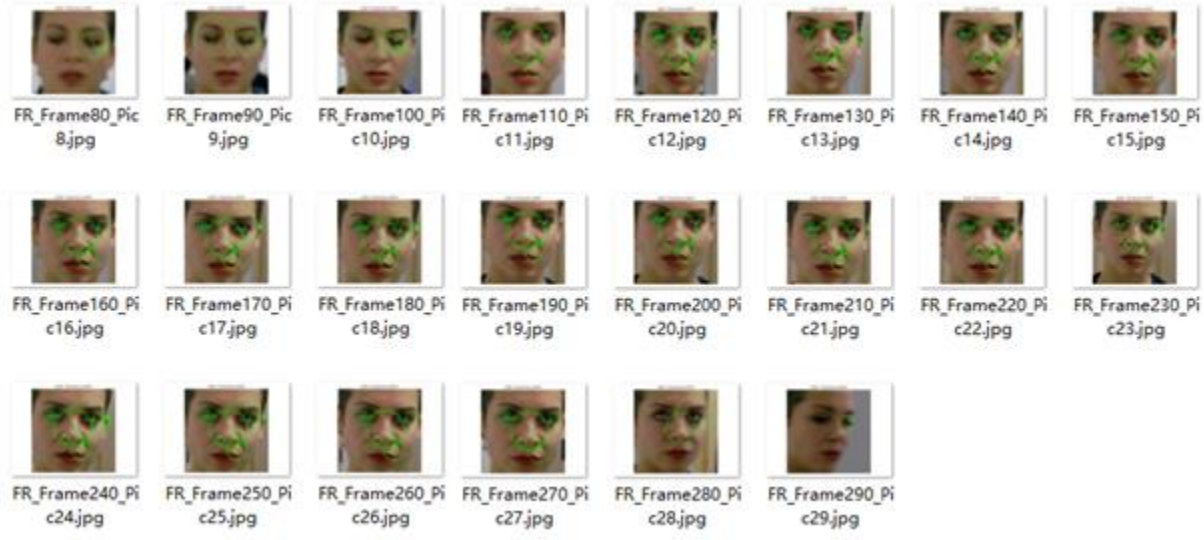
SURF 特征就是在设定特定的阈值情况下，找出图像中的特征向量，特征向量包含有位置、角度、尺寸等信息。为了将其可视化地展现出来，我们将特征点在原图中以有向圆圈的形式画出来，圆心表示向量位置，绘制出的一条圆心到圆弧的半径方向表示向量角度方向，圆的直径大小表示向量尺寸。绘制了 SURF 特征的图像如图 4-11 所示。为了后面训练 SVM 分类器所保存了 SURF 特征的 Excel 文件如图 4-12 所示。

从图 4-11 的(a)(b)的对比可以看出，真脸的 SURF 特征点明显较少且尺寸较小，而假脸在眉毛、眼睛、鼻头、嘴巴、耳朵等处有较多尺寸较大的特征点，表现在图像上便是绿色区域较多，颜色鲜艳十分明显。而图 4-11 的(c)(d)比较而言，(d)颜色稍微鲜艳一点点，这是由于 Deepfake 合成的假脸与真脸十分接近。

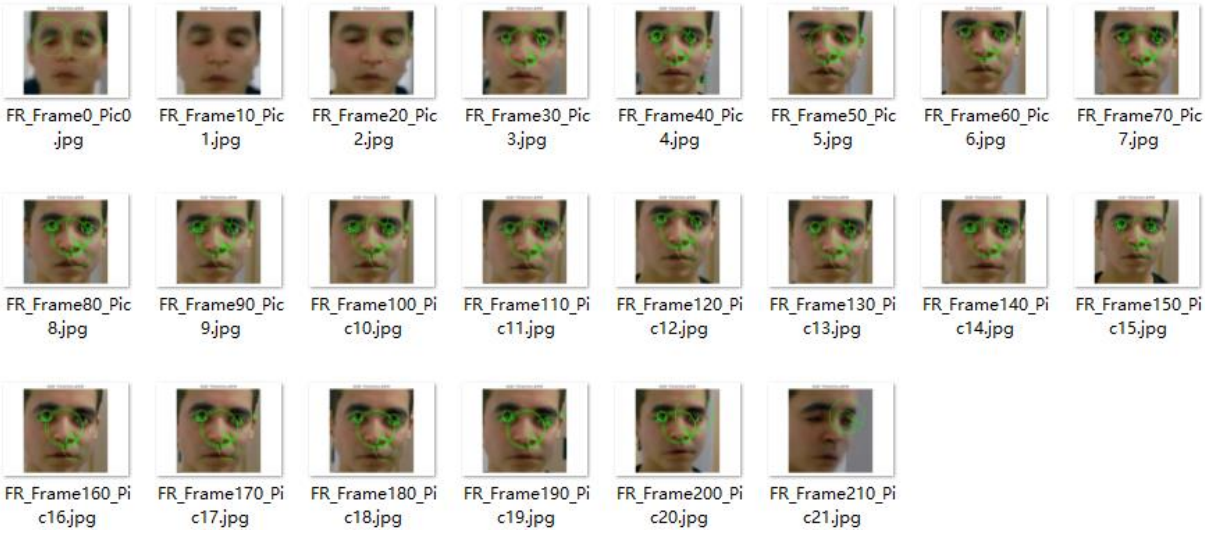


(a) Celaba 真脸 SURF 特征图像

(b) PGGAN 假脸 SURF 特征图像



(c) DFD 真脸 SURF 特征图像



(d) DFD 假脸 SURF 特征图像

图 4-11 SURF 特征图像

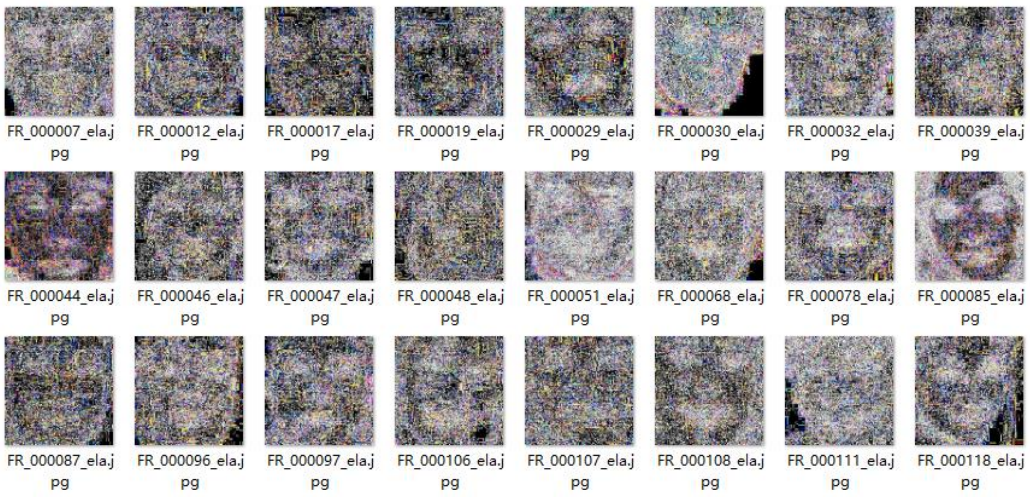
	A	B	C	D	E
1	x	y	angle	diameter	
2	86.50747	156.5885	337.5877	87	
3	127.3797	188.6003	182.5776	55	
4	215.393	81.33787	26.72819	47	
5	129.0494	195.1072	232.1191	44	
6	200.9802	123.9391	253.6454	39	
7	200.8668	127.0375	259.7556	32	
8	27.15483	129.783	8.463839	30	
9	21.4221	28.46	29.17863	29	
10	26.81833	129.734	2.147694	27	
11	34.25446	143.6423	159.6582	27	
12	133.1191	193.0221	253.6192	22	
13	95.08981	121.2082	80.65968	19	
14	93.56763	113.0749	271.5166	19	
15	41.38394	165.6736	290.2957	16	
16	24.41984	127.8976	5.803071	16	
		000000	000004	000005	00001: ...

图 4-12 SURF 特征数据 Excel

4.5 ELA 特征提取结果展示

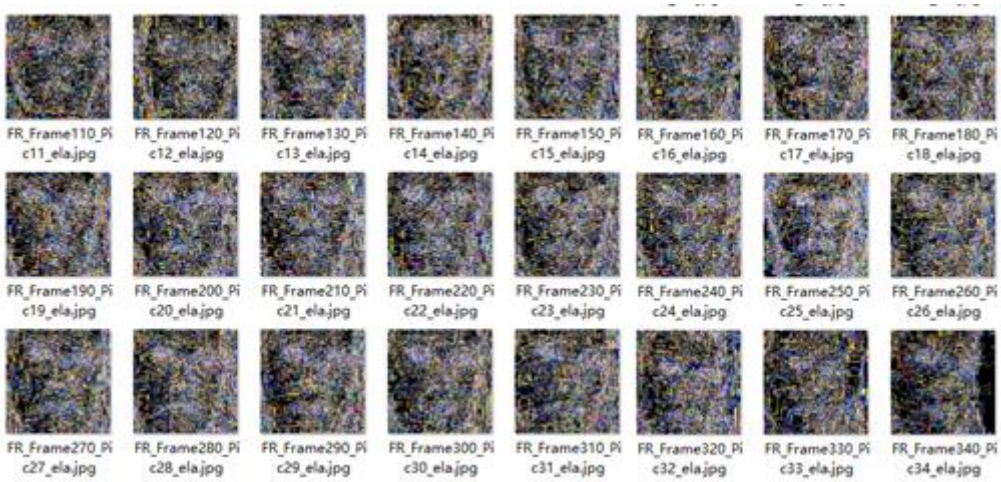
ELA 是将作差后的数据绘图，效果如图 4-13，保存的 Excel 如图 4-14。

通过图 4-13 的(a)(b)两图对比，我们可以看到 Celeba 真脸的 ELA 图上颜色大部分趋于一致，而 PGGAN 的眼睛、嘴巴、耳朵等区域与其他区域差异较大，可断定为该图像的这几处有被修改过的嫌疑。然而，对于 DFD 的真假脸而言，两者 ELA 图像（图 4-13 的(c)(d)）中，眼睛与其他区域颜色有差异，但都不是太大。

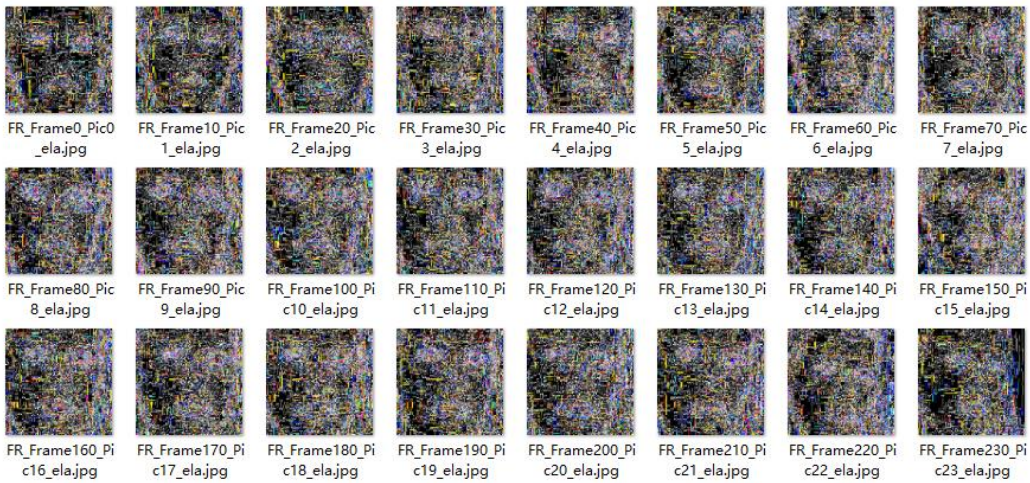


(a) Celaba 真脸 ELA 特征图像

(b) PGGAN 假脸 ELA 特征图像

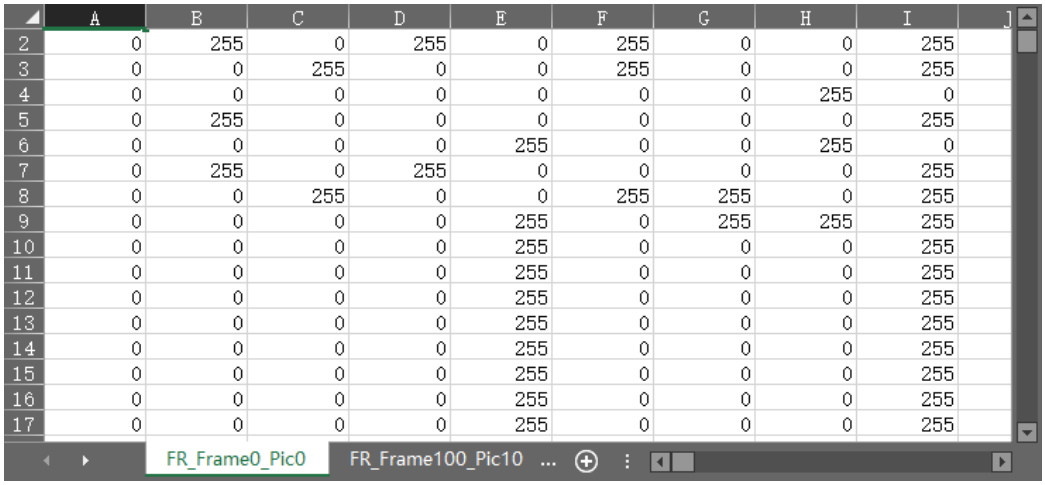


(c) DFD 真脸 ELA 特征图像



(d) DFD 假脸 ELA 特征图像

图 4-13 ELA 特征图



	A	B	C	D	E	F	G	H	I
2	0	255	0	255	0	255	0	0	255
3	0	0	255	0	0	255	0	0	255
4	0	0	0	0	0	0	0	255	0
5	0	255	0	0	0	0	0	0	255
6	0	0	0	0	255	0	0	255	0
7	0	255	0	255	0	0	0	0	255
8	0	0	255	0	0	255	255	0	255
9	0	0	0	0	255	0	255	255	255
10	0	0	0	0	255	0	0	0	255
11	0	0	0	0	255	0	0	0	255
12	0	0	0	0	255	0	0	0	255
13	0	0	0	0	255	0	0	0	255
14	0	0	0	0	255	0	0	0	255
15	0	0	0	0	255	0	0	0	255
16	0	0	0	0	255	0	0	0	255
17	0	0	0	0	255	0	0	0	255

图 4-14 ELA 特征数据 Excel

4.6 SVM 分类效果展示

由于首先数据库本身就很大，有很多图像，其次视频分帧之后图像会变多，再者就是每幅图像的特征数据大小分别是 color:3×65536,SURF:4×15;ELA:256×256，所以，总的下来会有大量的数据。为了看到实验效果，我们提取了其中的一部分数据，分别用来训练和测试 SVM 分类器。同时，为了实验的完整性，抽样的数据涉及到了各种类型：Celeba，PGGAN，DFD 的 original 和 attack。

训练和测试 SVM 分类器效果如图 4-15 所示。由图中数据可以看出，由于数据量的原因，color 的 SVM 分类器的训练、测试时间最长，ELA 次之，SURF 时间最短，也就最快。而在准确率方面，经过多次实验，我们可以得出 color 分类器大约 61%，SURF 分类器大约 64%，ELA 分类器大约 65%的结论。观察测试数据的实际真假和预测真假，我们发现 SVM 分类器在判别 Celeba 和 PGGAN 数据时，识别准确率较高，能够达到 70%~80%；而在识别 DFD 数据时，常常出错，准确率只有 55%左右。

```
Running Time of 训练 color 特征 SVM 分类器 : 0:02:31.862390
测试数据实际真假:
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
测试数据预测真假:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0]
color_clf 预测准确率: 0.6164383561643836
Running Time of 测试 color 特征 SVM 分类器 : 0:02:21.229064
Running Time of 训练 SURF 特征 SVM 分类器 : 0:00:00.309207
测试数据实际真假:
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
测试数据预测真假:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 0 1 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0]
SURF_clf 预测准确率: 0.6438356164383562
Running Time of 测试 SURF 特征 SVM 分类器 : 0:00:00.271234
Running Time of 训练 ELA 特征 SVM 分类器 : 0:00:36.909247
测试数据实际真假:
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
测试数据预测真假:
[1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0
 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
ELA_clf 预测准确率: 0.6575342465753424
Running Time of 测试 ELA 特征 SVM 分类器 : 0:00:33.487409
```

图 4-15 color，SURF，ELA 的 SVM 分类器训练测试结果

参考文献

[1] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.

[2] Bregler, Christoph; Covell, Michele; Slaney, Malcolm (1997). Video Rewrite: Driving Visual Speech with Audio. Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. 24: 353–360.

[3] Suwajanakorn, Supasorn; Seitz, Steven M.; Kemelmacher-Shlizerman, Ira (July 2017). Synthesizing Obama: Learning Lip Sync from Audio. ACM Trans. Graph. 36 (4): 95:1–95:13.

[4] Thies, Justus; Zollhöfer, Michael; Stamminger, Marc; Theobalt, Christian; Nießner, Matthias (June 2016). Face2Face: Real-Time Face Capture and Reenactment of RGB Videos. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE: 2387–2395.

[5] Farquhar, Peter (27 August 2018). An AI program will soon be here to help your deepfake dancing – just don't call it deepfake. Business Insider Australia. Retrieved 27 August 2018.

[6] Cole, Samantha; Maiberg, Emanuel; Koebler, Jason (26 June 2019). This Horrifying App Undresses a Photo of Any Woman with a Single Click. Vice. Retrieved 2 July 2019.

[7] Romano, Aja (18 April 2018). Jordan Peele's simulated Obama PSA is a double-edged warning against fake news. Vox. Retrieved 10 September 2018.

[8] Swenson, Kyle (11 January 2019). A Seattle TV station aired doctored footage of Trump's Oval Office speech. The employee has been fired. The Washington Post. Retrieved 11 January 2019.

[9] Holubowicz, Gerald (15 April 2020). Extinction Rebellion s'empare des deepfakes. Journalism.design (in French). Retrieved 21 April 2020.

[10] Katerina Cizek, William Uricchio, and Sarah Wolozin: Collective Wisdom | Massachusetts Institute of Technology [1]

[11] Amerini I, Galteri L, Caldelli R, et al. Deepfake Video Detection through Optical Flow

Based CNN[C]//Proceedings of the IEEE International Conference on Computer Vision Workshops. 2019: 0-0.

[12] Yang X, Li Y, Lyu S. Exposing deep fakes using inconsistent head poses[C]//ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019: 8261-8265.

[13] Yang X, Li Y, Qi H, et al. Exposing GAN-synthesized Faces Using Landmark Locations[J]. arXiv preprint arXiv:1904.00167, 2019.

[14] Anwar S, Milanova M, Anwer M, et al. Perceptual Judgments to Detect Computer Generated Forged Faces in Social Media[C]//IAPR Workshop on Multimodal Pattern Recognition of Social Signals in Human-Computer Interaction. Springer, Cham, 2018: 38-48.

[15] Bappy J H , Simons C , Nataraj L , et al. Hybrid LSTM and Encoder-Decoder Architecture for Detection of Image Forgeries[J]. IEEE Transactions on Image Processing, 2019:1-1.

[16] Karras T , Aila T , Laine S , et al. Progressive Growing of GANs for Improved Quality, Stability, and Variation[J]. 2017.

[17] 汪启伟. 图像直方图特征及其应用研究[D]. 中国科学技术大学, 2014.

[18] 孙艺珊, 李晓洁, 赵凯. 改进 LBP 和 HSV 颜色直方图相结合的地表状态识别[J]. 测绘通报 (2): 29.

[19] 娄强. 颜色直方图识别新技术研究[D]. 天津大学, 2007.

[20] BAY H. SURF : Speeded Up Robust Features[J]. Computer Vision & Image Understanding, 2006, 110(3):404-417.

[21] 储蓄. 基于改进 SURF 算法图像匹配方法研究[D]. 2017.

[22] 张亚娟. 基于 SURF 特征的图像与视频拼接技术的研究[D]. 西安电子科技大学.

[23] Warif N B A , Idris M Y I , Wahab A W A , et al. An evaluation of Error Level Analysis in image forensics[C]// 2015 5th IEEE International Conference on System Engineering and Technology (ICSET). IEEE, 2015.

[24] Jeronymo D C , Borges Y C C , Coelho L D S . Image Forgery Detection by Semi-Automatic Wavelet Soft-Thresholding with Error Level Analysis[J]. Expert Systems with

Applications, 2017:S0957417417303664.

[25]Zhang W , Zhao C , Li Y . A Novel Counterfeit Feature Extraction Technique for Exposing Face-Swap Images Based on Deep Learning and Error Level Analysis[J]. Entropy, 2020, 22(2):249.

[26]丁世飞, 齐丙娟, 谭红艳. An Overview on Theory and Algorithm of Support Vector Machines%支持向量机理论与算法研究综述[J]. 电子科技大学学报, 2011, 040(001):1-10.

[27]张萍, 王琳, 游星. 基于 SVM 分类的边缘提取算法[J]. 成都理工大学学报:自然科学版, 2017, 044(002):247-252.

[28]徐晓明. SVM 参数寻优及其在分类中的应用[D]. 大连海事大学, 2014.

[29]周绍磊, 廖剑, 史贤俊. RBF-SVM 的核参数选择方法及其在故障诊断中的应用[J]. 电子测量与仪器学报, 2014(03):20-26.

[30]王功鹏, 段萌, 牛常勇. 基于卷积神经网络的随机梯度下降算法[J]. 计算机工程与设计, 2018, 39(2): 441-445.

[31]金钊. 基于改进随机梯度下降算法的 SVM[D]. 河北大学, 2017.

[32]林雯. 新型基于帧间差分法的运动人脸检测算法研究[J]. 计算机仿真, 2010(10):248-251.

[33]高逸飞, 胡永健, 余泽琼,等. 5 种流行假脸视频检测网络性能分析和比较[J]. 应用科学学报, 2019, 37(5): 590-608.

[34]石雅笋. 改进的 SURF 图像配准算法研究[D]. 电子科技大学.

[35]李岩, 刘念, 张斌,等. 图像镜像复制粘贴篡改检测中的 FI-SURF 算法[J]. 通信学报, 2015, 036(005):54-65.

[36]Rublee E , Rabaud V , Konolige K , et al. ORB: An efficient alternative to SIFT or SURF[C]// International Conference on Computer Vision. IEEE, 2012.

[37]Ramadhani E . Photo splicing detection using error level analysis and laplacian-edge detection plugin on GIMP[J]. Journal of Physics Conference, 2019, 1193(1):012013.

[38]张学工. 关于统计学习理论与支持向量机[J]. 自动化学报, 2000(01):36-46.

[39]李建民, 张钹, 林福宗. 支持向量机的训练算法[J]. 清华大学学报(自然科学版),

2003, 043(001):120-124.

[40]陈永义, 俞小鼎, 高学浩,等. 处理非线性分类和回归问题的一种新方法(Ⅰ)——支持向量机方法简介[J]. 应用气象学报, 2004 , 15(3): 345-354.

[41]汪宝彬,汪玉霞. 随机梯度下降法的一些性质[J]. 数学杂志(6):1041-1044.