计算机视觉与模式识别

Sliding Window

2021 年 04 月 26 日

一、实验内容

（1） 利用特征训练一个分类器（线性分类器、SVM 等均可）。我最终采用 SVM 作为人脸分类器。

（2） 按照 sliding window 思想编写代码，提取 HOG、Color Histogram 并生成候选框；

（3） 使用分类器进行分类（Scoring each proposal）

（4） 编写非极大值抑制算法，去除一些置信度较低的候选框，得到预测结果，实现人脸检测。

（5） 对实验结果进行分析。

二、实验原理

2.1 做法和原理

首先是训练一个人脸分类器，由于是二分类，类别少，所以只需要采用简单的分类器，我采用了 SVM，提取 HOG 特征进行训练。

然后是采用滑动窗口，这部分没有什么特别的难点，我是采用了循环的方式，设置了不同的窗口大小 win_size，以及设置一个步长 strides，然后就是类似于卷积一样每次用训练好的 svm 模型进行分类。

分类完所有的窗口后，进行非极大值抑制，这样做的目的是找出最大可能的人脸，并且去掉冗余的重复识别的框。主要是根据计算 IOU 两个框的 IOU，如果 IOU 大于设定的阈值，则把概率高的保留，其余的则去掉。这里我还加了一个是人脸的概率阈值，例如设置阈值 80%，如果分类器得到的结果只有 54%可能是人脸，则丢弃掉这些框。

2.2 非极大值抑制的流程：

- 根据置信度得分进行排序

- 选择置信度最高的边界框添加到最终输出列表中，将其从边界框列表中删除
- 计算所有边界框的面积
- 计算置信度最高的边界框与其它候选框的 IoU。
- 删除 IoU 大于阈值的边界框
- 重复上述过程，直至边界框列表为空

2.2 我的实现

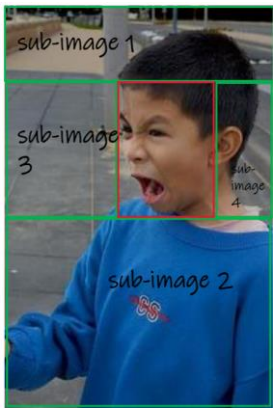
2.2.1 数据集的处理

所有图片不管是人脸还是非人脸，均采用 100*100 的分辨率，如果不是，则用插值法 resize 成 100*100 分辨率。

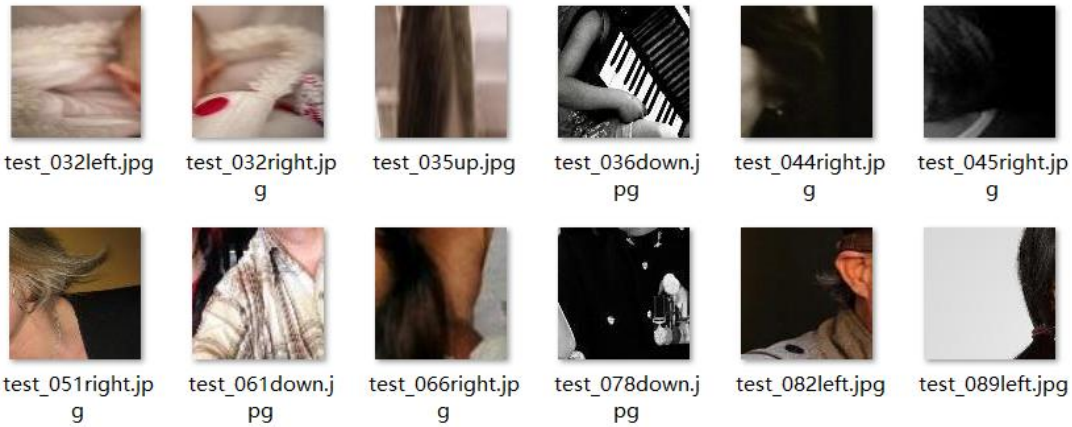
数据集部分，人脸数据集采用了 Real-world Affective Faces Database(RAF-DB：Real-world Affective Faces (RAF) Database (whdeng.cn)。包括了共 3954 个人脸照片，由于数据集提供了人脸的框位置，所以我提取了数据集原图中人脸框以外的位置作为一部分负样本（不过需要手动检查，因为这样有一些图片仍然含有人脸）；对于非人脸部分，采用了 Natural Images：https://www.kaggle.com/datasets/prasunroy/natural-images，这是 kaggle 上的一个数据集，包含了 8 个类别：狗、猫、人、花、水果、摩托车、汽车、飞机，我把人的类别去掉了，其余 7 类共 5913 个图片作为负样本，加上从前面说过的提取人脸框外的图片作为补充的负样本，最后负样本数有 8368 个图片。

图片 1：人脸数据集 Real-world Affective Faces Database

如何自己提取一些非人脸呢，对于 Real-world Affective Faces Database，提取给定人脸框（bounding box），将其左、右、上、下的图片作为一些负例，但是，仍然有一些原图是几个人脸的，所有还得手动检查，去掉不符合要求的脸，鉴于手动检查的工作量比较大，最后只手动检查得到了 2400 多张提取的负类图片；此外，我发现原图有很多白边，所以我还对图片进行了去白边处理。提取示意图如下：



图片 2：提取原图上下左右作为非人脸数据的一部分，绿色框部分为提取的子图



图片 3：提取人脸数据集中框出的人脸以外的图片作为一部分负例，我的提取方法是：提取给出人脸框以外的上下左右四张图片

图片 4：kaggle 的 Natural Images 数据集

2.2.2 模型训练与保存

SVM 的训练其实更简单，因为是二分类，至少比处理数据集需要的时间少。我采用 HOG 提取特征，图像全部被 resize 成 100×100 的大小，然后提取 HOG 特征，并将数据集按 9:1 的比例作为训练集和验证集。HOG 特征，我采用的是每个 block 含有 4 个 cell，相邻的两个 block 之间有一个 cell 的长度重叠，而每个 cell

则是一个 10×10 的子图，然后采用的分箱是 10 个 bins，这样的话，每个图片的特征数就是 blocks×cells×bins=(100/10 - 1)^2×4×10=3240 个。

然后人脸标签为 0，其他为 1，为了保证训练的鲁棒性，我还进行了洗牌 shuffle，减少一连串样本都是正例或者负例的情况。进行训练，我分别用了不同的核函数进行测试，包括 linear、poly、rbf 三个核函数。并保存模型，方便后续滑动窗口检测使用。

2.2.3 滑动窗口检测

滑动窗口也比较简单，我采用了多种框大小，以及可选的步长，根据框大小和步长依次提取图片中的子图进行分类，循环完整个图片，如果是人脸概率大于阈值，则添加作为候选框，否则就认为不是人脸。唯一的问题就是，框的可能实在太多了，成千上万个，即使 SVM 模型比较简单，分类很快，但是面对这么多框的分类，速度是一个大问题，想要分的越好，框的大小和数量最好更多，步长最好更小，这样计算量就非常的大。

2.2.4 非极大值抑制

将刚刚得到的候选框按概率从大到小进行排序，选出候选框概率最大的提出来，然后和剩下的框计算 IoU，将 IoU 大于阈值的去掉；反复计算，直到候选框为空。这样就得到了人脸检测的全部结果。就只放一个非极大值抑制的核心代码就行了，其他代码感觉 没必要放。下面代码就是对按概率排序后的框，取出最大的，和剩下的计算 IOU，将 IOU 大于阈值的去掉，重复此过程直到候选框为空。

```
157   IoU_threshold = 0.05
158   # non maximum suppression
159   while detected_box_sort != []:
160       # choose max probability and its corresponding box, then remove the max one
161       final_boxes.append(detected_box_sort[0])
162       final_proba.append(detected_probability_sort[0])
163       box = detected_box_sort.pop(0)
164       proba = detected_probability_sort.pop(0)
165
166       # calculate IoU
167       boxes_iou = getIoU(box, detected_box_sort)
168
169       # remove boxes whose IoU is larger than threshold
170       removed_number = 0
171       for i in range(len(boxes_iou)):
172           if boxes_iou[i] >= IoU_threshold:
173               detected_box_sort.pop(i-removed_number)
174               detected_probability_sort.pop(i-removed_number)
175               removed_number += 1
```

图片 5：非极大值抑制的核心代码

三、实验结果与分析

3.1 实验结果

3.1.1 训练 SVM 模型

因为是二分类，所以训练的准确率很高，很多时候是 99%，有时候是 100%，这取决于我设置的训练参数。

```
Data preproceing cost 0:00:09.573032 s

Start SVM...
Penalty coefficient:  1  === kernel:  poly

HOG SVM validation set accuracy: 1.0 ==> Training set accuracy 1.0
SVM model has been saved in './model/svm.model'!
Svm training and saving cost 0:03:02.399061 s
```

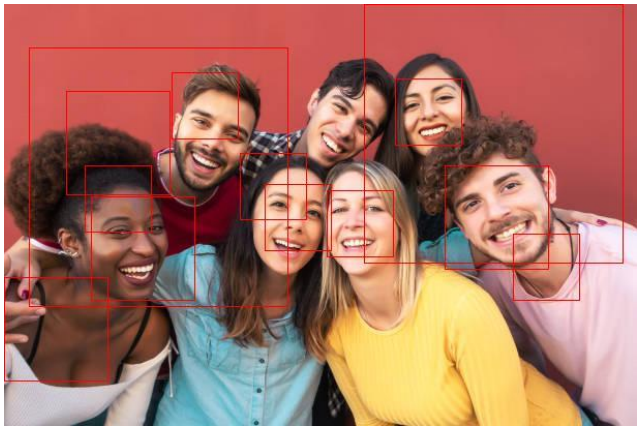
图片 6：采用 poly 核函数训练的二分类 SVM 模型；图中展示了验证集准确率和训练集准确率

3.1.2 视觉效果

对于不同的 IOU 阈值 a 以及认为是人脸的阈值 b，a 和 b 的值会对结果的影响比较大，会对最终框的数量以及准确率有较大的影响。图片 5 是单张人脸的测试，可以看到有些不是人脸的地方也被识别成了人脸.

图片 7：检测视觉效果

下面的图 7 则是阈值设置的不够好时得到的结果，可以看到有不少冗余的框。



图片 8：IOU 阈值设置的太大，以及认为是人脸的阈值太小时，得到的框会过多

调整阈值后，可以看到冗余框明显减少（图 8），7 个人脸有 6 个都被检测出来了，还有三个框识别错了地方。经过我测试，第七个人脸也是可以识别出来的，但是需要更多的框，以及更小的步长。为了减少时间，我采用的都是正方形框，所以相比训练时有很多人脸被 resize 变形后的识别结果会有所降低。

图片 9：经过调整后，可以看到效果明显好了不少

3.1.3 评价指标

我将圈出的框以及人脸作为总数（所以你会看到 TN 都是 0），识别正确的人脸作为正例，反之是负例，构建混淆矩阵，得到检测的准确率、召回率、F1-SCORE 等。

可以看到一开始准确率为(6+0)/14=43%，错误率为 1-43%=57%，精度为 6/(6+7)=46%，召回率为 6/7=86%，F1-SCORE 为 2×46%×86%/(46%+86%)=60%.

	预测人脸	预测非人脸	总数
实际人脸	6	1	7
实际非人脸	7	0	7
总数	13	1	14

表格 1：阈值设置的不好时的混淆矩阵

经过调整阈值和训练核函数等，结果明显改善，改善后的准确率为 6/10=60%，错误率则下降到了 40%，精度为 6/(6+3)=67%，召回率为 6/7=86%，F1-SCORE=2×67%×86%/(67%+86%)=75%.

	预测人脸	预测非人脸	总数
实际人脸	6	1	7
实际非人脸	3	0	3
总数	9	1	10

表格 2：阈值较好时的表现

可以看到，相比上面的混淆矩阵，调整阈值后的各项参数都变得更好。为了更直观，我做了个对比表格，设置为粗体的表示该指标效果更好。

	阈值设置的不好时	阈值设置得较好时
准确率	43%	**60%**
错误率	57%	**40%**
精度	46%	**67%**
召回率	86%	86%
F1-SOCRE	60%	**75%**

表格 3：表格 1 和表格 2 得到的精度、召回率等指标的对照表

3.2 分析

图片 7 可以看到单张人脸识别效果既识别出了正确的人脸,也得到了不存在的人脸,经过分析,是因为训练的人脸图片都是经过 resize 的图像,人物脸部会有或多或少的变形,而我检测时为了减少计算量,都是采用的方形框,所以不存在人物脸部变形的干扰,加上负例的训练数据种类还是比较少,对于陌生的图片,如果训练数据中没有这种负例,则很可能归类为人脸,故实际会有识别错误的情况.

另外,后面的图片看到有个别人脸没有被识别出来.这是因为滑动窗口的计算量实在太大了,故我设置了如下 14 中窗口大小,可以看出是明显不够的,所以有些人脸恰好不能被窗口精确包括,故识别不到也是正常的.要想精确识别全部,窗口的大小得更多种,并且还得是长宽也得有不同,才能达到最优效果.但是那样计算量太大,得到一个图片结果所需要的时间是不太可以忍受的,故我没有这样做.

```
window_sizes = [1024, 900, 800, 700, 600, 512, 400, 300, 250, 200, 164, 128, 100, 64]
```

最后就是训练集中的负类了,这是个难点,因为按理说负例是越多越好,因为实际中有数不清的非人脸物体,而我的非人脸训练集只有 8000 多个,有不少图片还是类似的物体,故实际分类人脸效果并不是完美的.

四、总结

滑动窗口的思想比较简单,实现起来也比较容易,但是缺点也比较明显,就是需要的框太多了,计算量非常大,所以速度很慢.不过滑动窗口的思想确实比较有意思,仔细想和动物的视觉机制还是有类似的地方的,眼睛也是需要扫视不同的地方来辨别物体,只不过速度比滑动窗口快很多.

很有意思,后面想个办法优化滑动窗口的策略,不然检测时间过长是无法忍受的;另外训练集上我也花了不少功夫,处理得到负类训练集等.