به نام خدا

درس دید کامپیوتری

گزارش تمرین شماره دوم

علیرضا بانشی

95101185

تمارین کامپیوتری

قسمت اول

gammaCorrection

bool cv::HOGDescriptor::gammaCorrection

Flag to specify whether the gamma correction preprocessing is required or not.

histogramNormType

int cv::HOGDescriptor::histogramNormType

histogramNormType

L2HysThreshold

double cv::HOGDescriptor::L2HysThreshold

L2-Hys normalization method shrinkage.

nbins

int cv::HOGDescriptor::nbins

Number of bins used in the calculation of histogram of gradients. Default value is 9.

nlevels

int cv::HOGDescriptor::nlevels

Maximum number of detection window increases. Default value is 64.

Examples:

samples/tapi/hog.cpp.

oclSvmDetector

UMat cv::HOGDescriptor::oclSvmDetector

coefficients for the linear SVM classifier used when OpenCL is enabled

signedGradient

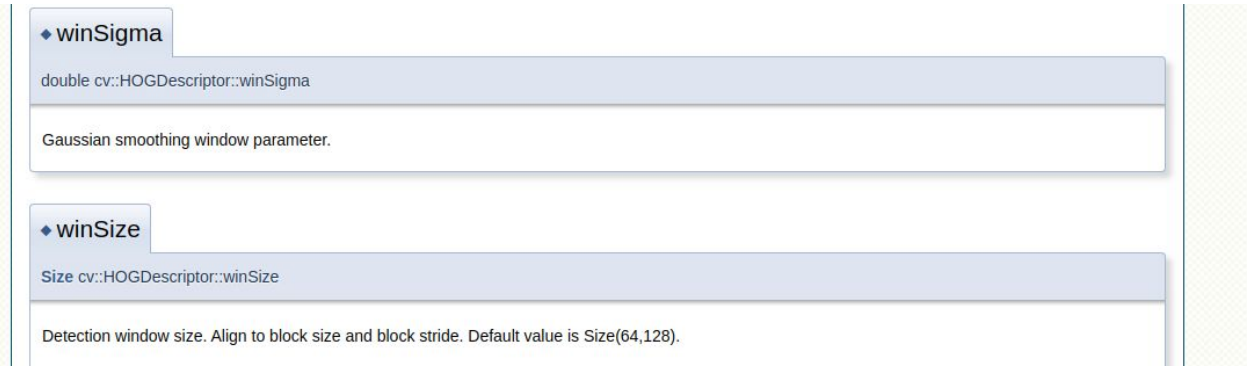
bool cv::HOGDescriptor::signedGradient

Indicates signed gradient will be used or not.

svmDetector

std::vector<float> cv::HOGDescriptor::svmDetector

coefficients for the linear SVM classifier.



این مقادیر در مقاله به شرح زیر است:

$$\text{Winsize} = 128\text{x}64$$

$$\text{Cellsize} = 4\text{x}4$$

$$\text{Blocksize} = 16\text{x}16$$

$$\text{Blockstride} = 4\text{x}4$$

$$\text{Nbins} = 9$$

قسمت دوم

برای این که HOG یک تصویر را محاسبه کنیم ابتدا به کمک HOG descriptor پارامتر ها را ست می کنیم و سپس به کمک دستور compute آن را محاسبه می‌کنیم.
کد این بخش در قسمت section2 آمده است و نتایج در فایل hog feature ذخیره و پیوست شده است.

قسمت سوم

در این قسمت با استفاده از توضیحات داده شده در صورت تمرین patch های مثبت و منفی را لود می‌کنیم. کد این قسمت در section3 آمده است.

قسمت چهارم

در این قسمت می‌خواهیم داده ها را لیبل بزنیم.
برای این کار ابتدا تمامی patch ها را چه مثبت چه منفی در X1 می ریزیم و سپس به ازای هر المان که در X1 داریم ویژگی های HOG آن را استخراج می کنیم و در X می‌ریزیم.حالا X مجموعه تمامی داده های ماست.
حالا لازم داریم بردار y را تشکیل دهیم که طول آن به اندازه X و هر المان آن با 0 و 1 بودن به ترتیب نگاتیو یا پوزیتیو بودن را تعیین می‌کند.
برای این کار دو np.array با طول های متناسب با پچ های مثبت و منفی تشکیل می‌دهیم و آن ها را با 1 و 0 پر می کنیم و در نهایت این دو بردار را کانکت می‌کنیم:

```
5 Y = np.array([1
6                    for im in positive_patches])
7 Y1 = np.array([0
8                    for im in negative_patches])
9
0 y = np.concatenate((Y,Y1),axis = 0)
```

حالا داده های ما آماده است و می‌توانیم یادگیری را آغاز کنیم.
کد این قسمت در section4 آمده است.

قسمت پنجم

ابتدا باید داده های train و test را جدا کنیم. برای این کار از کد زیر استفاده می‌کنیم:

```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

با این کار 80 درصد داده ها را به trainو 20 درصد را به تست اختصاص می‌دهیم.
برای یادگیری از دو الگوریتم استفاده کرده‌ایم:

1. اول svm خطی و با استفاده از متد GridSearchCV

به کد زیر دقت کنید:

```
svclassifier = GridSearchCV(LinearSVC(), {'C': [1.0, 2.0, 4.0, 8.0]})
svclassifier.fit(X_train, y_train)
y_pred_svm_linear_test = svclassifier.predict(X_test)
y_pred_svm_linear_train = svclassifier.predict(X_train)
```

ابتدا در خط اول مدل را تشکیل می‌دهیم.سپس مدل را با داده هاید train فیت می‌کنیم.
حالا داده های train و test را به کمک مدل predict می کنیم تا دقت روی هر دو را بدست بیاوریم.
حالا به کمک کد زیر دقت را محاسبه می‌کنیم.

```
print("linear : Trainig data Accuracy:",metrics.accuracy_score(y_train, y_pred_svm_linear_train))
print("linear : Testing data Accuracy:",metrics.accuracy_score(y_test, y_pred_svm_linear_test))
```

نتیجه به قرار زیر می‌باشد:

```
linear : Trainig data Accuracy: 0.9973399641473428
linear : Testing data Accuracy: 0.9932924713773563
```

پارامتر های بهینه ای که به جواب بالا ختم شده است به قرار زیر است:

```
print(svclassifier.best_params_)

{'C': 2.0}
```

2. دو svm غیر خطی با کرنل rbf

در این قسمت برای تشکیل مدل کافیست از کد زیر استفاده کنیم:

```
svclassifier1 = SVC(kernel='rbf')
```

بقیه مراحل را مانند قسمت قبل انجام می‌دهیم نتیجه به قرار زیر می‌باشد:

```
rbf :Training data Accuracy: 0.9992771641704736
```

```
rbf : Testing data Accuracy: 0.9973401179599861
```

مشاهده می‌شود که با این متد دقت کمی بهتر است.
زیرا دیتاها گستردگی زیادی دارند و svm غیر خطی به دلیل این‌که در فضای با بعد بالاتر داده ها را جدا می‌کند عملکرد بهتری دارد.
کد های این قسمت در section 5 آمده است.

قسمت ششم

ابتدا 5 تصویر تست داده شده را می‌خوانیم و آن‌ها را با ضریب .5 rescale می‌کنیم سپس قسمت هایی از آن را کراپ می‌کنیم تا محاسبات در ادامه سریعتر شود و نتایج بهتری بگیریم.

```
test_image1 = cv2.imread('test_image1.jpg',0)
test_image1 = transform.rescale(test_image1, 0.5)
test_image1 = test_image1[20:300, 80:230]
```

کد این قسمت در ابتدای section 6 ودر بخش Reading Test Images آمده است.
خروجی تصاویر را مشاهده می‌کنیم:

حالا باید برای هر تصویر pyramid ها را حساب کنیم.برای این کار از دو روش زیر استفاده کردم .
یک روش آماده با کمک تابع آماده موجود در کتابخانه skimage داده شده در تمرین:

```
pyramid = np.asarray(tuple(pyramid_gaussian(image)))
```

دو روش مشابه به کمک opencv

```
layer = image
gaussian_pyramid = [layer]
for i in range(7):
    layer = cv2.pyrDown(layer)
    gaussian_pyramid.append(layer)
```

حالا سعی کردم برای تصاویر بدست آمده از هر تصویر ویژگی های HOG را استخراج کنم که هر بار با ارور عجیب زیر مواجه شدم:

```
                                    1 frames
/usr/local/lib/python3.6/dist-packages/skimage/feature/_hog.py in hog(image, 
    273     n_blocks_col = (n_cells_col - b_col) + 1
    274     normalized_blocks = np.zeros((n_blocks_row, n_blocks_col,
--> 275                                   b_row, b_col, orientations))
    276
    277     for r in range(n_blocks_row):

ValueError: negative dimensions are not allowed

SEARCH STACK OVERFLOW
```

سپس با مشاوره از دوستان راه دیگری را انتخاب کردم.

روی تصاویر پرمایش می‌کنیم و پنجره های 62*47 را انتخاب می‌کنم.
البته همه پنجره ها را ن و برای سادگی 16 بار 16 بار حرکت می‌کنیم.تابعی که این کار را انجام می دهد به شرح زیر است که بسیار ساده است:

```
def pyramid_fun(img):
    for i in range(0, img.shape[0] - 62, 16):
        for j in range(0, img.shape[1] - 47, 16):
            # extracting window
            patch = img[i:i + 62, j:j + 47]
            yield (i, j), patch
```

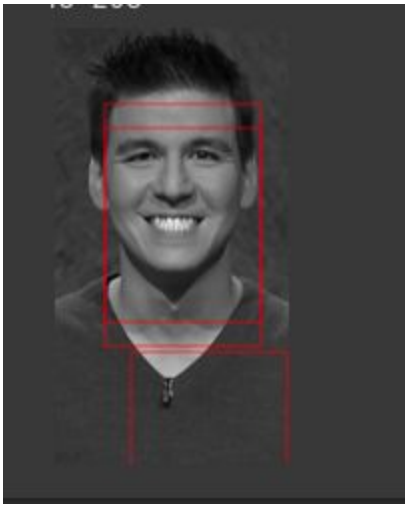
صرفا یک سری تصویر تشکیل می‌دهد و در دیکشنری می‌ریزد.
حالا مصابق زیر دیکشنری تصاویر مرتبط با هر تصویر تست را بدست می‌آوریم.سپس ویژگی های hog هر ست را تشکیل می‌دهیم و نتایج را در یک np.array ذخیره می‌کنیم.مطابق زیر:

```
i1, p1 = zip(*pyramid_fun(test_image1))
i2, p2 = zip(*pyramid_fun(test_image2))
i3, p3 = zip(*pyramid_fun(test_image3))
i4, p4 = zip(*pyramid_fun(test_image4))
i5, p5 = zip(*pyramid_fun(test_image5))



test1_hog = np.array([feature.hog(patch) for patch in p1])
test2_hog = np.array([feature.hog(patch) for patch in p2])
test3_hog = np.array([feature.hog(patch) for patch in p3])
test4_hog = np.array([feature.hog(patch) for patch in p4])
test5_hog = np.array([feature.hog(patch) for patch in p5])
```

حال وقت آن رسیده است که این داده های جدید را به کمک مدل بدست آمده در قسمت قبل پیشبینی کنیم.
ابتدا مدل را پردیکت می‌کنیم و سپس روی اندیس ها میگردیم و جاهایی که چهره تشخیص داده شده یعنی out1 برابر یک است یک مستطیل می‌کشیم.
گوشه چپ بالا به کمک پنجره های پیش بینی شده توسط مدل بدست می‌آید و ابعاد مستطیل تقریبا مطابق با ابعاد صورت در تصویر تست است.
کد این قسمت به قرار زیر است:

```
out1 = svclassifier.predict(test1_hog)



fig, ax = plt.subplots()
ax.imshow(test_image1, cmap='gray')
ax.axis('off')

Ni, Nj = positive_patches[0].shape
indices = np.array(i1)
t=[]

for i, j in indices[out1 == 1]:

    ax.add_patch(plt.Rectangle((j, i), 100, 140, edgecolor='red',
                               alpha=0.3, lw=2, facecolor='none'))
    print(j,i)
    t.append((j,i,j+100,i+140))
tt = np.array(t)
```

همینطور محل همه مستطیل ها را در tt ذخیره می‌کنیم بعدا از آن استفاده می‌کنیم:
خروجی برای تصویر به قرار زیر است:

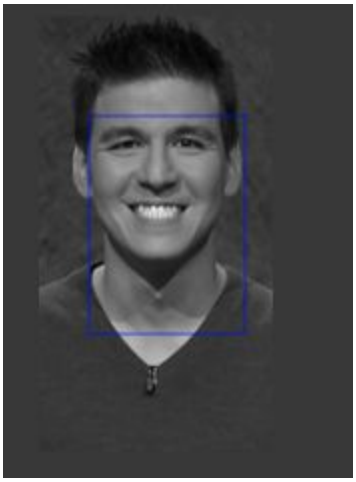


مشاهده می‌شود که نتیجه قابل قبول است.
حالا باید از Non.Maximum.Suppression استفاده کنیم که بهترین مستطیل را انتخاب کنیم.
با جست و جو اینترنت تابع های آماده ای پیدا کردم که این کار را انجام می‌دهد.
لینک :https://github.com/bruceyang2012/nms_python

کد های این قسمت در بخش Non.Maximum.Suppression آورده شده است.
حالا به این تابع همه مستطیل ها را ورودی می‌دهیم (در قسمت قبل در tt ذخیره کردیم و آن بهترین مستطیل را خروجی می‌دهد.
کد این قسمت مطابق زیر است:

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow

import time
pick = non_max_suppression_fast(tt, probs=None,overlapThresh=0.3)

fig, ax = plt.subplots()
ax.imshow(test_image1, cmap='gray')
ax.axis('off')

Ni, Nj = positive_patches[0].shape
indices = np.array(indices4)
ax.add_patch(plt.Rectangle((pick[1][0], pick[1][1]), 100, 140, edgecolor='blue',
                           alpha=0.3, lw=2, facecolor='none'))

```

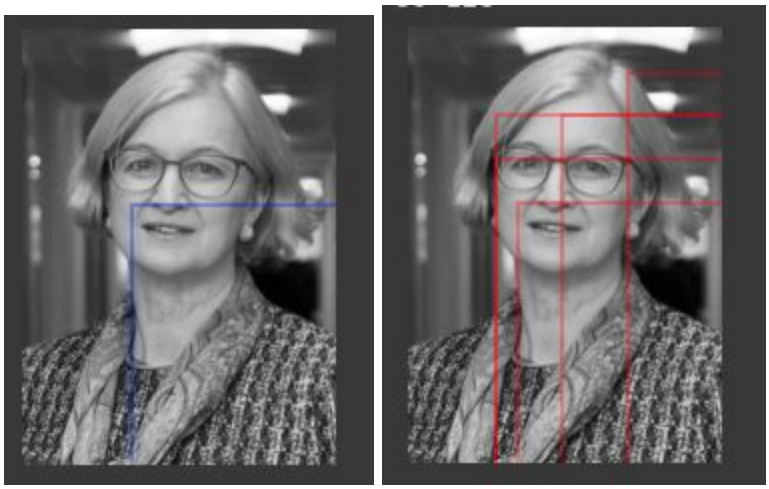
نتیجه برای اولین تصویر به قرار زیر است:



حالا همین کار را برای بقیه تصاویر انجام می‌دهیم و نتایج را مشاهده می‌کنیم:

تصویر2

تصویر3



تصویر4

تصویر5

همه کد های مربوط به این قسمت ها به تفکیک تصویر تست زیر قسمت section 6 آمده است.