Amirreza Vishteh

HW0

25/11/1401

Operation System

(1

الف)Q1.ipynb

با opencv

```
img = cv2.imread("Q1.jpg")
#load image in grey
gimg = cv2.imread("Q1.jpg",0)
#Displaying images
print(img.shape)
cv2.imshow("bird", img)
cv2.imshow("birdgray", gimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Matplotlib:

```
imageplt = plt.imread('Q1.jpg')
plt.imshow(imageplt)

plt.axis('off')
plt.show()
```

✓ 0.5s

تفاوتشان :

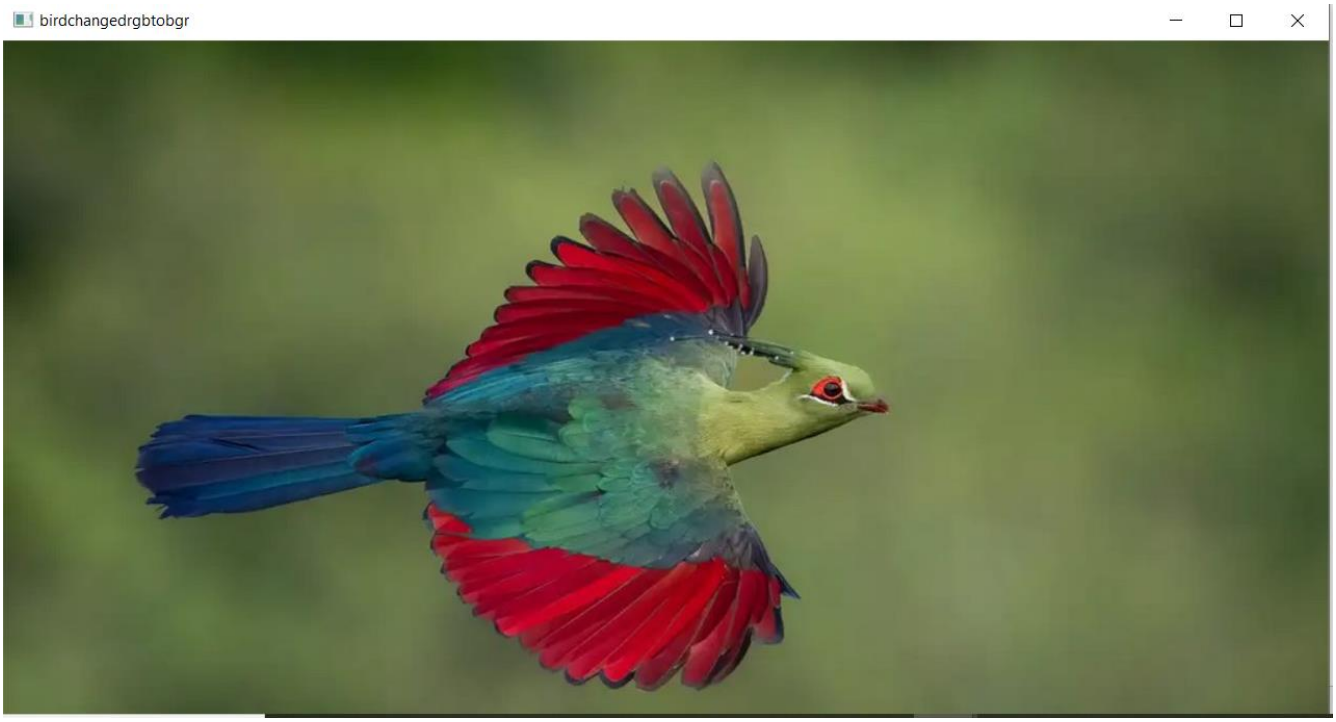
1)

در این بود که در matplotlib دور تصویر یک scale است که انرا با axis پاک کردم

2)در matplotlib عکس ها rgb اند ولی در cv2 به صورت bgr اند بنابراین برای نمایش در cv2عکسی که با که با matplotlib خوانده شده باید تبدیل این ترکیب رنگی صورت بگیرد در غیر این صورت رنگ ها همانطور که در پنجره باز شده با نام don’t change مبینید ترکیب رنگی برعکس خواهد بود

```
converted = cv2.cvtColor(imageplt,cv2.COLOR_BGR2RGB)
cv2.imshow("dontchange",imageplt)
cv2.imshow("birdchangedrgbtobgr", converted)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

عکس درست:



ب)

Img.shape یک tuple سه تایی است که اول تعداد ردیف و بعدی تعداد ستون و بعدی تعداد کانال های عکس است کانال منظور برای این تصویر rgb است. یعنی سه تا

```
(680, 1020, 3)
```

پ)

```
folder_path = "./Q1/NEW"
if os.path.exists("Q1/NEW"):

    test = os.listdir(folder_path)
    for images in test:
        if images.endswith(".jpeg") or images.endswith(".txt"):
            os.remove(os.path.join(folder_path, images))
    os.rmdir("Q1/NEW")
```

در اینجا چک میکنیم اگر فولدر مدنظر موجود بود ابتدا محتویات انرا حذف و بعد خود دایرکتوری را حذف میکنیم

سپس:

```
new_dir="./Q1/NEW"
os.mkdir(new_dir)
folder_dir = "./Q1"
folder_path = "./Q1/NEW"
```

فولدر را مسازیم وبعد:

```
# print(os.listdir(folder_dir))
for images in os.listdir(folder_dir):

    if (images.endswith(".jpeg")):
        pathme=folder_path+ "/"+ images.split('.')[0]+".txt"
        grayimage= cv2.imread(folder_dir + "/" + images,cv2.IMREAD_GRAYSCALE)
        # print(grayimage)
        cv2.imwrite(f"Q1/NEW/{images}",grayimage)
        f=open(pathme,"w")
        f.write(str(grayimage.shape))
        f.close()
```

به ازای تمام عکس های موجود یک فایل txt میسازیم وبعد نسخه سیاه و سفید عکس را به دست میاوریم و در فولدر ذخیره میکنیم.

(2

ابتدا تابع crop را پیاده سازی میکنیم :

```
images = []
xstart=0
ystart=0
for i in range(0,MainSize[0],CropSize[0]):
  for j in range(0,MainSize[1],CropSize[1]):
    if np.sum(image[i:i+CropSize[0], j:j+CropSize[1]]):
      print(f"{i}:{i+CropSize[0]}, {j}:{j+CropSize[1]}")
      images.append(image[i:i+CropSize[0], j:j+CropSize[1]])

return images
```

در این تابع لیستی از image های slice شده ذخیره میشود که تاریک نیستند یعنی sum(pixels)!=0 است که index انها به این شکل است:

```
0:16, 0:16
0:16, 16:32
0:16, 32:48
0:16, 48:64
0:16, 64:80
0:16, 80:96
0:16, 96:112
0:16, 112:128
0:16, 128:144
0:16, 144:160
0:16, 160:176
0:16, 176:192
0:16, 192:208
0:16, 208:224
0:16, 224:240
0:16, 240:256
0:16, 256:272
0:16, 272:288
0:16, 288:304
0:16, 304:320
0:16, 320:336
0:16, 336:352
0:16, 352:368
0:16, 368:384
0:16, 384:400
```

تابع resize هم از توابع پیش فرض استفاده میکند و عکس را resize میکند:

```
def resize(image, size):
    '''
      calculate the resized image
      input(s):
        image (ndarray): input image
        size (tuple): size of output image
      output(s):
        images (ndarray): the resized image
    '''

    resized_image = cv2.resize(image, size)
    return resized_image
```

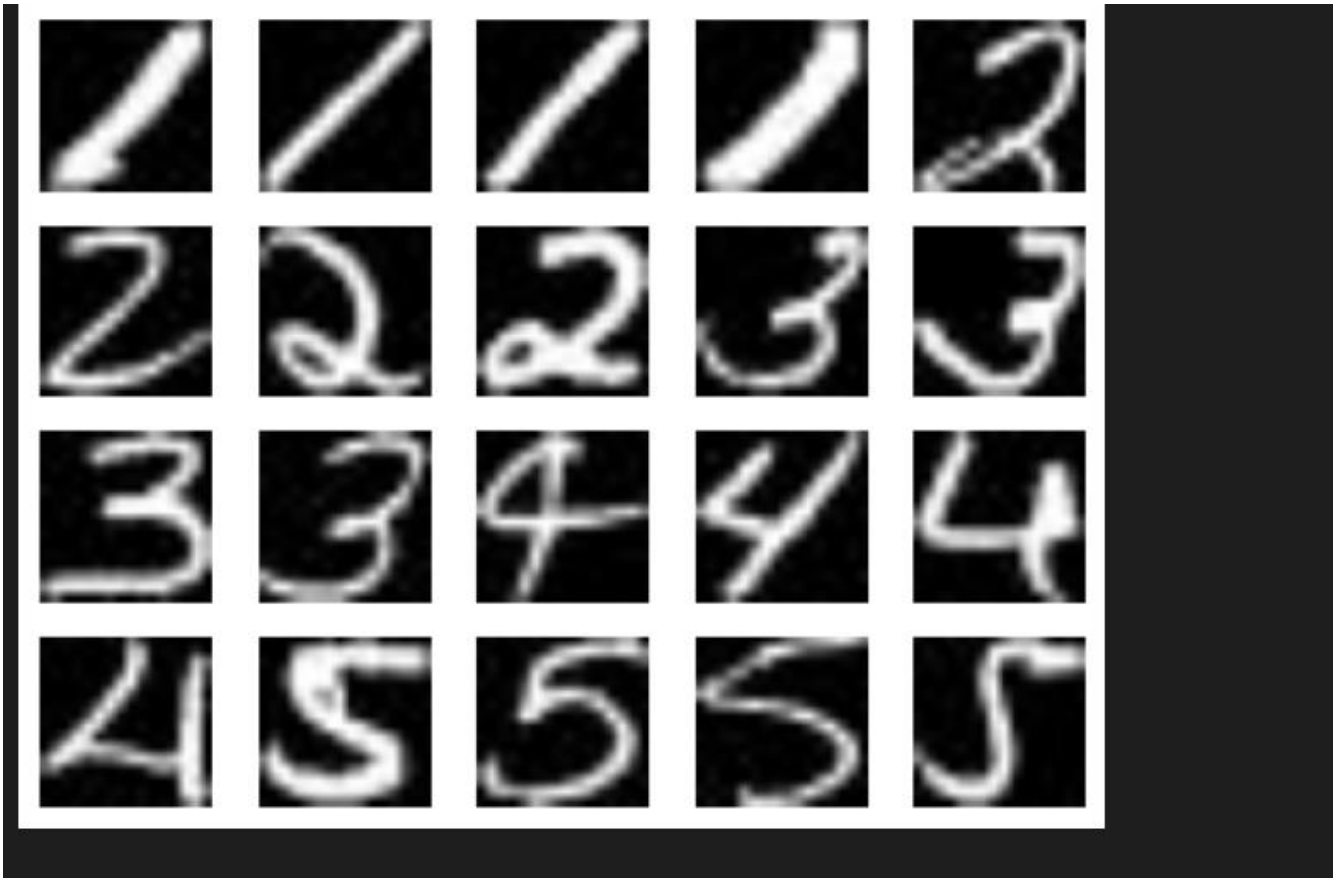
در بخش بعدی عکس های دایرکتوری Q2-images را میخوانیم و ذخیره میکنیم:

```
images_path = ["usps_1.jpg","usps_2.jpg","usps_3.jpg","usps_4.jpg","usps_5.jpg"]
images = []
foldername="Q2_images/"
for i in images_path:

  image = cv2.imread(foldername+i,cv2.IMREAD_GRAYSCALE)
  images.append(image)
```

در بخش اخر هم کار های گفته شده را انجام دادم مثل: resize و انتخاب عدد رندوم و در نهایت برای نشان دادن عکس ها در فرمت خواسته شده از تابع plt.subplot استفاده کردم

```
k=1
for i in range(0,5):
    for j in range(0,4):
        x=np.random.randint(0,1100)
        new=resize(All_cropped_images[i][x],(50,50))
        plt.subplot(4,5,k)#satr,soton,chandomin shekle
        plt.imshow(new,cmap='gray')

        plt.axis('off')
        k=k+1
plt.show()
```

عکس نهایی:



(3

ابتدا یک عدد از ورودی میگیریم و بعد ماتریس خواسته شده را میسازیم.

در تابع create_matrix صرفا یک ماتریس با درایه های n تا100+n مسازد:

```
matrix = np.random.randint(n, n+100, size=(n, n))
################################################

return matrix
```

0.0s

در تابع شمردن ارقام هم صرفا عدد را برای شمردن ارقامش به string تبدیل کردم

```
##################################################
############ YOUR CODES GO HERE ##############

digits = {}
for i in range(10):
  digits[i] = 0
for i in range(len(matrix)):
  for j in range(len(matrix)):
    strdeg= str(matrix[i][j])
    for k in range(len(strdeg)):
      digits[int(strdeg[k])] += 1

##################################################

return digits
```

در تابع traversematrix هم صرفا با الگو یا بی مسیر خواسته شده را به دست اوردم

(4

بخش اول صرفا محاسبه عملیات ها با توابع اماده است

```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])

b=np.array([[1,2,3],[4,5,6],[7,8,9]])
c=np.array([[1],[-1],[1]])

newtranspose=np.transpose(a)
# print(newtranspose)
res=np.dot(newtranspose,a)

# print(res)
reverse=np.linalg.inv(res)
res2=np.dot(reverse,newtranspose)
res3=np.dot(res2,c)
final=np.add(res3,a)
print(final)
```

در بخش دوم روی ماتریس بزرگتر حرکت و مربع هایی با سایز ماتریس کوچک تر خارجو حاصل ضرب را در ماتریس resultمیریزیم:

```
n=A.shape[0]
m=B.shape[0]
res=np.zeros((n-m+1,n-m+1))
for i in range(0,n-m+1):
  for j in range(0,n-m+1):
    res[i,j]=np.multiply(A[i:i+m , j:j+m], B).sum()

##################################################

return res
```

(5

```
image = cv2.imread("Q5.png")
h, w, c = image.shape
print(f"image:height={h} width={w} chanels={c}")
mtype=image.dtype
print(f"Type {mtype}")
print("All Pixels:")
av=np.mean(image)
mi=np.min(image)
ma=np.max(image)
print(f"average is {av}")
print(f"minimum is {mi}")
print(f"maximum is {ma}")
# mincolorchanal=np.amin(np.amin(image,axis=0),axis=0)[0]
# maxchanal=np.amax(np.amax(image,axis=0),axis=0)[0]
minchanal=np.min(image[:,:,0:1])
maxchanal=np.max(image[:,:,0:1])
averagechanel=np.mean(image[:,:,0:1])
print("for chanel 0:")
print(f"average is {averagechanel}")
print(f"minimum is {minchanal}")
print(f"maximum is {maxchanal}")
```

بخش اول مجدد با slicing , search است

بخش دوم:

اینجا ما سه ارایه نظیر به هم classes, boxes ,scores داریم که در boxes ابتدا مختصات گوشه سمت چپ و بعد ابعاد ان شکل است و کلاس هم کلاس ان شی و اون یکی هم ضریب اطمینان است حال در تابع visualize دور اشیا با تابع rectangle مستطیل میکشیم در detections که boxes قرار دارد و ما مختصات را به ان داده و ان میکشد

```
generated_img =frame.copy()
# cv2.rectangle()
for i in range(len(detections)):
  cv2.rectangle(generated_img,(detections[i][0][0],detections[i][0][1]),(detections[i][0][0]+de
cv2.imwrite('result.png', generated_img)
return generated_img
```

بعد عکس را در imwrite result.png میکنیم:

```
img = visualize(image, detections)
cv2.imwrite('result.png', img)
plt.imshow(img)
plt.show()
```