Шмаков Данила 181-351

Задание 6

В ходе выполнения работы использовалась библиотека Pillow PIL

```
from PIL import Image
```

Обучение происходило путем «скармливания» RGB пикселей с кожей, для это загружалась и конвертировалась в многомерный массив, а потом и в list картинка кожи на белом фоне

```
skin = Image.open('Skin.png').convert('RGB')
dataskin = np.array(skin)
dataskin = dataskin.reshape(skin.size[0] * skin.size[1], 3)
```

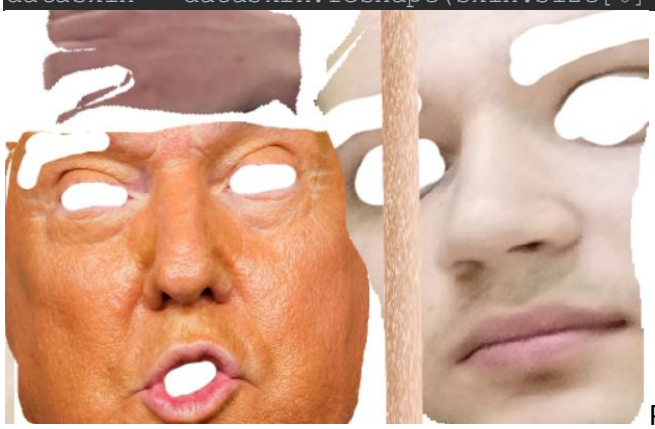

Рис 1. Фрагмент картинки с кожей

Если пиксель не белый, то добавляем его RGB- значения и метку кожа (1) в списки

```
for item in dataskin:
    if (item != listBack).any():
        listR.append(item.tolist()[0])
        listG.append(item.tolist()[1])
        listB.append(item.tolist()[2])
        listLabel.append(1)
```

Аналогично делаем с не кожей, присваивая метку «0».

Создаем датафрейм из этих листов и удаляем дубликаты

```
df = pd.DataFrame({'R':listR,'G':listG,'B':listB, 'label':listLabel})
print(df)

df = df.drop_duplicates(subset=['R', 'G', 'B'], keep=False)
print(df)
```

При подборе обучающих картинок я старался, чтобы количество меток кожи не кожи были примерно равны.

```
print(len(df[df["label"]==0]))
print(len(df[df["label"]==1]))
```

Создаем и обучаем классификатор

```
points_train, points_test, labels_train, labels_test =
train_test_split(df.iloc[:, :-1], df['label'],

test_size=0.25,random_state=0)
gnb = GaussianNB()
gnb.fit(points_train, labels_train)
prediction = gnb.predict(points_test)
```

Предсказываем значения меток для каждого пикселя картинки и создаем лист меток

```
listpredict = gnb.predict(datatest.reshape(test.size[0]*test.size[1],3))
```

Перекрашиваем все пиксели с меткой 1 в красный цвет

```
img = np.array(test)
row = 0
column = 0
for item in img:
    for i in item:
        if listpredict[row* len(item) + column] == 1:
            img[row][column] = replacement_color
        column += 1
    column=0
    row+=1

img2 = Image.fromarray(img, mode='RGB')
img2.show()
```

Получившиеся результаты:

HSV

Для HSV я пошел по пути оптимизации и использовал немного другое решение

Открываем обучающие картинки кожи/не кожи в HSV формате.

```
skin_hsv = Image.open('Skin.png').convert('HSV')
no_skin_hsv = Image.open('NoSkin2.jpg').convert('HSV')
```

Переводим данные в вид массива, удобного для работы

```
#перевод данных в удобный формат
skin_colors_hsv = np.array(skin_hsv).reshape(skin_hsv.size[0]*skin_hsv.size[1],3)
no_skin_colors_hsv = np.array(no_skin_hsv).reshape(no_skin_hsv.size[0]*no_skin_hsv.size[1],3)
```

Создание датафреймов:

```
#создание датафреймов
df_skin = pd.DataFrame(skin_colors_hsv, columns=['H','S','V']).drop_duplicates().reset_index(drop=True)
df_no_skin = pd.DataFrame(no_skin_colors_hsv, columns=['H','S','V']).drop_duplicates().reset_index(drop=True)

ones = pd.DataFrame(1, index=np.arange(len(df_skin)), columns=['label'])
zeros = pd.DataFrame(0, index=np.arange(len(df_no_skin)), columns=['label'])

df_skin = df_skin.assign(label=ones)
df_no_skin = df_no_skin.assign(label=zeros)

df_hsv = pd.concat([df_skin, df_no_skin]).reset_index(drop=True)
```

Изначально мы создаем два датафрейма со столбцами ‘h’ ‘s’ ‘v’ с соответствующими значениями. Далее создаем датафреймы метки: только нули и только единицы со значениями длинны равными df_no_skin и df_skin, после этого добавляем эти датафреймы-метки как метки (столбец ‘label’) и соединяем датафрейм в один.

Обучаем классификатор

```
points_train_hsv, points_test_hsv, labels_train_hsv, labels_test_hsv = train_test_split(df_hsv.iloc[:, :-1],
    df_hsv['label'], test_size=0.25, random_state=0)
gnb.fit(points_train_hsv, labels_train_hsv)
prediction_hsv = gnb.predict(points_test_hsv)
```

Тест классификатора:

Открываем тестовую картинку, предсказываем все метки кожа/не кожа. После чего в цикле перерисовываем все кожи в красный

```
#тест классификатора
me_hsv = Image.open('man.jpg').convert('HSV')
data_hsv = np.array(me_hsv).reshape(me_hsv.size[0]*me_hsv.size[1], 3)

prediction = gnb.predict(data_hsv)

img = np.array(me_hsv)

for i in range(len(img)):
    n = len(img[i])
    for j in range(n):
        if prediction[i * n + j] == 1:
            img[i][j]=(0,255,255)

Image.fromarray(img,mode="HSV").show()
```