Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет Информатики и систем управления
Кафедра Прикладная математика и информатика

Лабораторная работа №4 (РК1)
по курсу «Машинное обучение»

«Классическое компьютерное зрение. Распознавание лиц на фото»

Выполнил:
Студент группы ИУ9-31М
Беляев Антон

Проверила:
Криндач О.

Москва, 2019

Содержание

Постановка задачи

Необходимо реализовать алгоритм компьютерного зрения с использованием алгоритмов «классического» компьютерного зрения, без использования нейронных сетей для решения задачи классификации лиц.

Необходимо протестировать алгоритм на выбранном наборе данных и привести результаты.

Используемые инструменты

В качестве набора данных было выбрано подмножество известного датасета LFW, содержащего изображения лиц знаменитостей, а именно - изображения 3х людей:

- Вильям (Билл) Саймон
- Дженнифер Энистон
- Леброн Джеймс

В качестве используемых инструментов были выбраны

- Язык Python 3.7
- Библиотека компьютерного зрения OpenCV
- KNN-классификатор из библиотеки sklearn
- Распознавание лиц с помощью каскадов Хаара

Ход работы

На листинге ниже представлена программа

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

with zipfile.ZipFile('data 2.zip', 'r') as zip_ref:
    zip_ref.extractall('.')

# xml with classifier can be found at venv at cv2/data
clf = cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')


class Face:
    def __init__(self, img_path: str, clazz: str = None):
        self.clazz = clazz
        self.img = cv2.imread(img_path, cv2.COLOR_BGR2GRAY)
        self.detected_face = None
        self.embedding = None
        self._detect_face()
```

```
    def _detect_face(self):
        faces = clf.detectMultiScale(self.img, scaleFactor=1.2, minNeighbors=5)
        if 0 != len(faces):
            x, y, w, h = faces[0]
            self.detected_face = self.img[y:y + w, x:x + h]
            self.embedding = face_recognition.face_encodings(self.img, [[x, y, w, h]])[0]

    def __repr__(self):
        return self.__str__()

    def __str__(self):
        return f'[{self.clazz}] -> embedding[{len(self.embedding)}]'


jen = Face('/content/data/Jennifer_Aniston/Jennifer_Aniston_0006.jpg', 'Jen')

print('image:')
cv2_imshow(jen.img)

print('detected face:')
cv2_imshow(jen.detected_face)

print(jen)
```

Результат работы кода выше для демонстрации определения лица (и построение embedding'а размером в 128 элементов) на одном изображении:

```
image:
```

```
detected face:
```

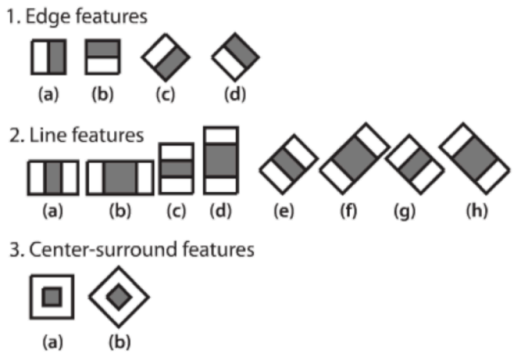
```
[Jen] -> embedding[128]
```

```
IMG_DATA_ROOT = 'data'

faces = []
for person in os.listdir(IMG_DATA_ROOT):
    for person_img_name in os.listdir(IMG_DATA_ROOT + "/" + person):
        img_path = IMG_DATA_ROOT + "/" + person + "/" + person_img_name
        face = Face(img_path=img_path, clazz=person)

        if face.detected_face is None:
            print('could not detect face on ', img_path)
            continue

        faces.append(face)

print('faces detected: ', len(faces))
# —> 40
```

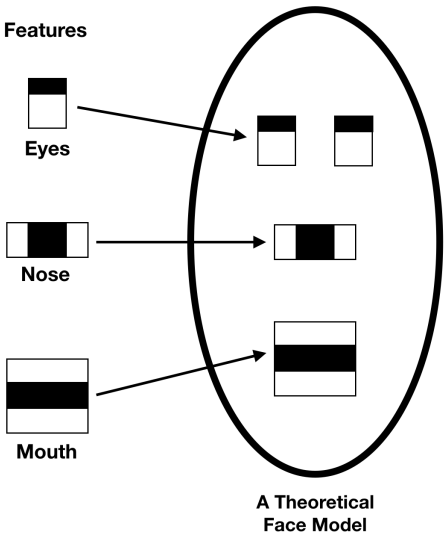
Распознавание лиц

Для распознавания лиц в библиотеке OpenCV доступны следующие методы:

- **Каскады Хаара**
- LBP (Local Binary Pattern)



Каскады Хаара используют алгоритм машинного обучения AdaBoost, который среди большого числа незначительных features с помощью «слабых классификаторов» (weak classifiers) отбирает самые важные, превращая их в «сильные классификаторы».



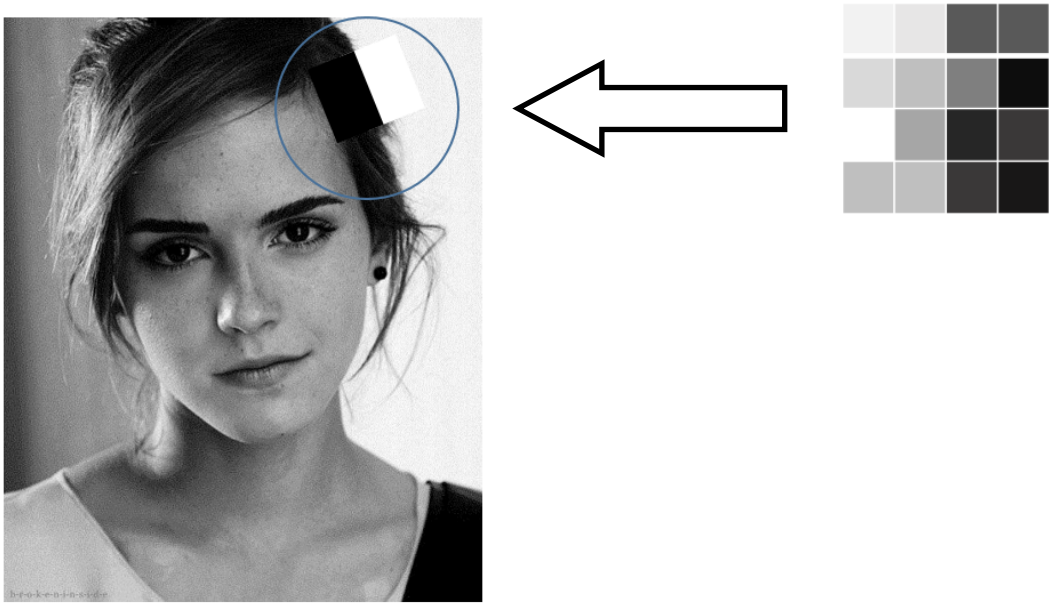
Таким образом, из этого набора выбираются самые важные характеристики лица.

В случае OpenCV, подобные предобученные каскады данных уже поставляются в формате XML «из коробки», вместе с библиотекой.

Каскады имеют свою спецификацию (глаза, рот) при распознавании конкретного изображения. В данном случае были выбраны каскады для распознавания лица в целом.

Каскады подобны фильтрам. Идея их прохождения в следующем:

- Обрабатывается лишь небольшой участок изображения за один раз (окно)
- Для каждого окна суммируется интенсивность темных и светлых пикселей
- Разность получившихся значений - и есть итоговое значение извлекаемой «фичи»
- Чем больше значение, тем значительнее фича.



Тестирование

При тестировании было выбран изображение, которого не было среди тренировочных данных и его embedding был подан в вход KNN-классификатору.

```
x_train = list(map(lambda face: face.embedding, faces))
y_train = list(map(lambda face: face.clazz, faces))

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)

# image not contained in test data
test_face = Face('/content/sample.jpg')
cv2_imshow(test_face.img)

# predict class of image
clazz_predicted = knn.predict([test_face.embedding])
print(clazz_predicted)
# —> Jennifer Aniston
```

Полученный класс - ‘Jennifer_Aniston’. Таким образом, поданное на вход изображение **классифицировано верно**!

['Jennifer_Aniston']

Вывод

В ходе работы была выполнена задача классификации изображения лиц.

Классификатор, построенный без применения нейросетей показал хорошие результаты.