# **9.2. Распознавание человека на фотографии**

( <https://www.asozykin.ru/courses/nnpython> )

Вводные замечания

[Глубокие нейронные сети](https://www.asozykin.ru/courses/nnpython) можно использовать для проведения биометрической идентификации человека по лицу. Такая идентификация удобна, потому что она выполняется быстро и не требует от человека каких-либо специальных действий. Кроме того, лицо у человека всегда с собой, его нельзя забыть дома или потерять.

Идентификация по лицу применяется во многих системах безопасности и обслуживания. Давайте рассмотрим пример из банковской отрасли. Клиент приходит в банк и показывает паспорт. Сотруднику банка нужно определить, что паспорт принадлежит именно этому человеку. Эта задача называется *верификацией*. Она не так проста, как может показаться, потому что люди в жизни иногда выглядят совсем не так, как на фотографии в паспорте. Например, я с возрастом изменился (хотя и не очень сильно):



Примеры более серьезных изменений можно найти на сайте [adme.ru](https://www.adme.ru/tvorchestvo-fotografy/foto-na-pasporte-i-v-zhizni-1222610/).

Задачу верификации можно решать автоматически, если сфотографировать клиента на web-камеру (это сейчас делают многие банки) и сравнить полученное изображение с фотографией в паспорте. Давайте посмотрим как это можно сделать с помощью глубоких нейронных сетей.

Библиотека машинного обучения dlib

Для распознавания человека на фотографии мы будем использовать библиотеку машинного обучения [dlib](http://dlib.net/), которая содержит удобные средства [распознавания лиц](http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html). Библиотека написана на C++, но у нее есть Python API, который мы и будем использовать.

Установить dlib проще всего вместе с [Anaconda](https://www.continuum.io/downloads). Установите Anaconda для своей операционной системы, а затем выполните команду:

conda install -c conda-forge dlib

Если вы не используете Anaconda, то можно установит dlib вручную по [инструкции](http://dlib.net/compile.html). Однако этот процесс значительно дольше и сложнее.

Алгоритм верификации

Для решения задачи верификации мы будем использовать [сверточную нейронную сеть](https://youtu.be/52U4BG0ENiM), а именно предварительно обученную нейронную сеть [ResNet](https://arxiv.org/abs/1512.03385). От сети отрезаются слои, отвечающие за классификацию, и остаются только сверточные слои, которые извлекают ключевые признаки из изображения. Результат работы - набор чисел, который называется *дескриптором*. Такие дескрипторы мы извлечем из фотографии клиента в паспорте и с web-камеры.

dlib использует модифицированный вариант сети ResNet34. Эта сеть выдает дескрипторы из 128 чисел. Сеть обучена специальным образом так, чтобы дескрипторы фотографий одного человека находились рядом друг с другом, а дескрипторы фотографий разных людей - далеко друг от друга.

Чтобы оценить близость дескрипторов в dlib используется [Евклидово расстояние](https://ru.wikipedia.org/wiki/%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%BE%D0%B2%D0%B0_%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B0). Если значение Евклидова расстояния между дескрипторами меньше 0.6, то считается, что на фотографиях один и тот же человек. С использованием такой метрики dlib обеспечивает точность 99.38% на тесте распознавания лиц [Labeled Faces in the Wild](http://vis-www.cs.umass.edu/lfw/).

Загрузка предварительно обученных моделей

Для работы нам понадобятся предварительно обученные модели для выделения лица на фотографии и извлечения дескрипторов из лица. Эти модели можно скачать с сайта dlib:

* [Модель для выделения лиц по 68 ключевым точкам](http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2).
* [Нейронная сеть ResNet34 для распознавания лиц](http://dlib.net/files/dlib_face_recognition_resnet_model_v1.dat.bz2).

Скачанные файлы нужно разархивировать и скопировать в каталог с программой распознавания лиц.

Реализация верификации человека на фотографии

Давайте рассмотрим, как реализовать верификацию человека на фотографии с помощью библиотеки dlib. Сначала нам необходимо подключить нужные модули:

import dlib

from skimage import io

from scipy.spatial import distance

Подключаем библиотеку dlib, модуль io из библиотеки [scikit-image](http://scikit-image.org/) для загрузки фотографий из файлов, а также модуль distance из библиотеки [SciPy](https://www.scipy.org/), который будет использоваться для рассчета Евклидова расстояния.

Создаем модели для распознавания лиц на основе ранее загруженных файлов:

sp **=** dlib**.**shape\_predictor('shape\_predictor\_68\_face\_landmarks.dat')

facerec **=** dlib**.**face\_recognition\_model\_v1('dlib\_face\_recognition\_resnet\_model\_v1.dat')

detector **=** dlib**.**get\_frontal\_face\_detector()

Загружаем и показываем первую фотографию. Это будет фотография из паспорта:

img **=** io**.**imread('sozykin\_passport.jpg')

win1 **=** dlib**.**image\_window()

win1**.**clear\_overlay()

win1**.**set\_image(img)

Фотография выглядит следующим образом:



Находим на фотографии лицо по 68 ключевым точкам с помощью алгоритма, описанного в статье [One Millisecond Face Alignment with an Ensemble of Regression Trees](http://www.csc.kth.se/~vahidk/papers/KazemiCVPR14.pdf):

dets **=** detector(img, 1)

**for** k, d **in** enumerate(dets):

**print**("Detection {}: Left: {} Top: {} Right: {} Bottom: {}"**.**format(

k, d**.**left(), d**.**top(), d**.**right(), d**.**bottom()))

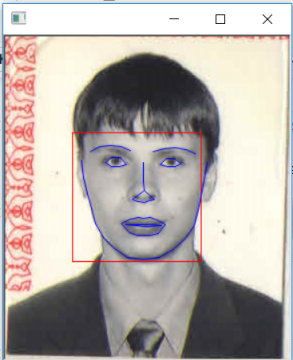
shape **=** sp(img, d)

win1**.**clear\_overlay()

win1**.**add\_overlay(d)

win1**.**add\_overlay(shape)

Найденное лицо и ключевые точки на нем будут показаны на фотографии:



Извлекаем дескриптор из найденного лица:

face\_descriptor1 **=** facerec**.**compute\_face\_descriptor(img, shape)

Можно напечатать дескриптор и посмотреть, как он выглядит:

**print**(face\_descriptor1)

**Пример дескриптора лица из 128 чисел**

-0.143361 -0.010574 -0.0205466 -0.0760989 -0.0817859 0.0557212 -0.0598148 -0.104499 0.156351 -0.113137 0.2541 -0.0255717 -0.294706 -0.0657252 0.00546183 0.0197356 -0.119519 -0.0928954 -0.0307344 -0.025405 0.0452379 0.027095 -0.0301305 0.0345449 -0.133859 -0.27474 -0.159836 -0.18149 -0.0380562 -0.0625319 0.00485351 0.0269533 -0.0954231 -0.0492104 0.031918 0.0857472 -0.0927711 -0.0259642 0.233257 -0.00325606 -0.121691 -0.079394 0.0188353 0.197803 0.130644 0.0386678 -0.0112381 -0.0870395 0.141996 -0.205454 0.0560226 0.253123 0.206587 0.118089 0.0277112 -0.154927 0.0465367 0.162008 -0.245842 0.0605535 0.048827 -0.0922293 -0.0161505 -0.0251943 0.197178 0.106469 -0.167589 -0.182694 0.208899 -0.130925 -0.081063 0.127503 -0.199762 -0.283384 -0.185639 0.0758053 0.370921 0.135176 -0.126596 -0.00810343 -0.170073 -0.130998 -0.0382377 0.0440755 0.00773637 -0.111487 -0.0522793 -0.0378758 0.171189 -0.0571423 0.025922 0.188256 0.00952435 0.0605275 0.0295069 0.0615573 -0.0917479 0.0212216 -0.057545 0.0380124 -0.0730892 -0.134272 0.0256359 0.063667 -0.145198 0.228549 0.0148294 -0.0463585 -0.057713 -0.0879126 -0.10461 -0.0139646 0.219559 -0.319542 0.197508 0.113558 0.116419 0.194342 0.0509606 0.105459 -0.0153985 -0.0616571 -0.264749 -0.0923162 0.00994077 -0.000828513 -0.0394463 0.0146801

Теперь сделаем все то же самое для фотографии с web-камеры:

img **=** io**.**imread('sozykin\_webcam.jpg')

win2 **=** dlib**.**image\_window()

win2**.**clear\_overlay()

win2**.**set\_image(img)

dets\_webcam **=** detector(img, 1)

**for** k, d **in** enumerate(dets\_webcam):

**print**("Detection {}: Left: {} Top: {} Right: {} Bottom: {}"**.**format(

k, d**.**left(), d**.**top(), d**.**right(), d**.**bottom()))

shape **=** sp(img, d)

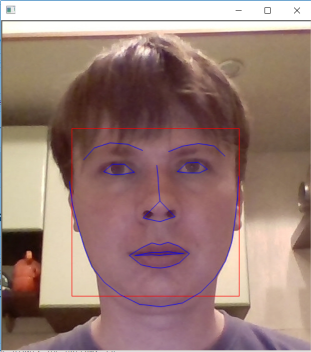
win2**.**clear\_overlay()

win2**.**add\_overlay(d)

win2**.**add\_overlay(shape)

face\_descriptor2 **=** facerec**.**compute\_face\_descriptor(img, shape)

Вот так выглядит фотография с выделенным лицом и ключевыми точками на нем:



Чтобы фото быстро не закрывалось, можно вставить команду:

wait = input(**"PRESS ENTER TO CONTINUE."**)

Теперь у нас есть два дескриптора лиц с разных фотографий face\_descriptor1 и face\_descriptor2. Чтобы определить, один и тот же человек на фотографиях, или нет, нам нужно найти Евклидово расстояние между дескрипторами:

a **=** distance**.**euclidean(face\_descriptor1, face\_descriptor2)

**print**(a)

Результат рассчета расстояния:

0.47269267125645975

Евклидово расстояние меньше 0.6, значит две фотографии принадлежат одному человеку.

Фотографии разных людей

Давайте попробуем сравнить фотографии разных людей. Загружаем фотографию другого человека и выделяем ее дескриптор:

img **=** io**.**imread('foto.jpg')

win2 **=** dlib**.**image\_window()

win2**.**clear\_overlay()

win2**.**set\_image(img)

dets\_webcam **=** detector(img, 1)

**for** k, d **in** enumerate(dets\_webcam):

**print**("Detection {}: Left: {} Top: {} Right: {} Bottom: {}"**.**format(

k, d**.**left(), d**.**top(), d**.**right(), d**.**bottom()))

shape **=** sp(img, d)

win2**.**clear\_overlay()

win2**.**add\_overlay(d)

win2**.**add\_overlay(shape)

face\_descriptor2 **=** facerec**.**compute\_face\_descriptor(img, shape)

Вот результат обнаружения лица:



Рассчитываем Евклидово расстояние:

a **=** distance**.**euclidean(face\_descriptor1, face\_descriptor2)

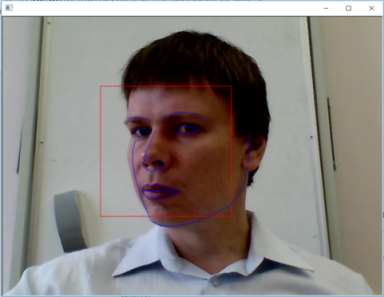
**print**(a)

0.8440052856492404

Расстояние больше 0.6, значит на фотографиях разные люди.

Изменение позы человека на фотографии

Может возникнуть вопрос: зачем выделять ключевые точки на лице? Почему нельзя просто извлекать дескриптор сразу из фотографии лица? Дело в том, что человек может смотреть не прямо в камеру, а в сторону. Например, так:



Если извлекать дескриптор из фотографии повернутого лица, он может сильно отличаться от дескриптора фотографии лица в фас. Чтобы решить эту проблему, dlib использует [афинное преобразование](https://ru.wikipedia.org/wiki/%D0%90%D1%84%D1%84%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) фотографии с использованием ключевых точек. Производится перенос ключевых точек в такую позицию, как будто бы человек смотрит прямо в камеру. Дескрипторы извлекаются только после афинного преобразования изображения.

Расстояние между дескрипторами на фотографии из паспорта и фотографии, на которой я смотрю в сторону от камеры:

0.47269267125645975

Расстояние больше, чем для фотографии, на которой я смотрю прямо в камеру, но все равно меньше 0.6. Значит на фото один и тот же человек.

Тестирование точности верификации

Пока все работает хорошо. Давайте протестируем более сложный случай: сравним фотографии различных похожих людей. Я нашел фотографию похожего человека с помощью сервиса [FindFace](https://findface.ru/):



Видно, что сходство действительно большое. Расстояние между дескриптором из этой фотографии и дескриптором из фотографии моего паспорта:

0.5564709089830993

Расстояние меньше рекомендуемого dlib значения 0.6, хотя на фотографиях разные люди. Это означает, что не всегда можно использовать значения расстояния, рекомендованные dlib или другой аналогичной библиотекой. Возможно для вашей системы, чтобы добиться необходимой точности, придется снизить границу расстояния до 0.5 или другого значения.

Итоги

Итак, мы научились проводить верификацию человека на фотографии с помощью библиотеки dlib. На первом шаге мы находим на фотографии лицо и его ключевые точки. Если человек смотрит в сторону, то ключевые точки переносятся таким образом, чтобы лицо было направлено прямо в камеру. Затем из выделенного лица извлекается дескриптор из 128 чисел с помощью предварительно обученной сети ResNet34. Для определения близости дескрипторов используем Евклидово расстояние.

По рекомендации dlib, если Евклидово расстояние меньше 0.6, значит на фотографиях один и тот же человек. Однако, как мы убедились, если использовать значение 0.6, то возможны ложные положительные срабатывания на фотографиях похожих людей. Поэтому вам, возможно, придется самостоятельно подбирать значение границы Евклидова расстояния для своих данных, чтобы обеспечить необходимую точность верификации.