**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc516502661)

[ГЛАВА 1. АКТУАЛЬНОСТЬ И ПОСТАНОВКА ЗАДАЧИ 7](#_Toc516502662)

[1.1. Актуальность 7](#_Toc516502663)

[1.2. Цели 7](#_Toc516502664)

[1.3. Используемые средства для выполнения работы 7](#_Toc516502665)

[1.3.1. Язык программирования Python 7](#_Toc516502666)

[1.3.2. Библиотека NumPy 8](#_Toc516502667)

[1.3.3. Библиотека OpenCV 8](#_Toc516502668)

[1.3.4. 8](#_Toc516502669)

[ГЛАВА 2. ОБЗОР СОВРЕМЕННЫХ МЕТОДОВ РАСПОЗНАВАНИЯ ЛИЦ 9](#_Toc516502670)

[2.1. Метод гибкого сравнения на графах (Elastic graph matching) 9](#_Toc516502672)

[2.2. Нейронные сети 12](#_Toc516502673)

[2.3. Скрытые Марковские модели (СММ, HMM) 14](#_Toc516502674)

[2.4. Метод главных компонент или principal component analysis (PCA) 15](#_Toc516502675)

[2.5. Active Shape Models (ASM) 20](#_Toc516502676)

[2.6. Метод Виолы-Джонса (Viola-Jones) 23](#_Toc516502677)

[2.7. Недостатки существующих систем 35](#_Toc516502678)

[ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ С ПОМОЩЬЮ БИБЛИОТЕКИ OPENCV И НЕЙРОННОЙ СЕТИ 37](#_Toc516502679)

[3.1. Создание обучающей выборки при помощи библиотеки OpenCV 37](#_Toc516502681)

[3.2. Выбор типа нейронной сети 38](#_Toc516502682)

[3.3. Разработка нейронной сети 39](#_Toc516502683)

[ЗАКЛЮЧЕНИЕ 46](#_Toc516502684)

[СПИСОК ЛИТЕРАТУРЫ 47](#_Toc516502685)

[ПРИЛОЖЕНИЕ 1 49](#_Toc516502686)

[ПРИЛОЖЕНИЕ 2 50](#_Toc516502687)

ВВЕДЕНИЕ

Одним из важнейших векторов в развитии искусственного интеллекта является распознавание образов, которое уже не первое десятилетие продолжает использовать все вычислительные мощности, предоставляемые ему человечеством. Определение лиц – развитая подветвь распознавания образов. Системы распознавания лиц все чаще становятся нужны в мире, будь то электронные замки, которые пропускают только определенных людей или пароли на компьютерах. Определение людей на фотографиях в социальных сетях, да что там, любой современный фотоаппарат или веб-камера имеет в своем функционале какой-либо алгоритм для этого.

Целью работы является разработка программы локализации области лица на фото при помощи сверточной нейронной сети.

В первой главе проведен обзор основных методов распознавания лиц, рассмотрены недостатки современных методов, обоснована актуальность проблемы.

Во второй главе была разработана программа распознавания лиц c помощью библиотеки OpenCV и сверточной нейронной сети.

ГЛАВА 1. АКТУАЛЬНОСТЬ И ПОСТАНОВКА ЗАДАЧИ

* 1. Актуальность

Технологии распознавания лиц являются на сегодняшний день одним из наиболее активно развиваемых направлений компьютерного зрения. И хотя рамки использования данных технологий давно вышли за пределы задач технических средств обеспечения безопасности, наибольшую актуальность они по прежнему имеют именно в этой сфере. Ведь грамотно организованная система, использующая распознавание лиц позволяет качественно решать целый спектр задач — от подсчета уникальных посетителей в торговом заведении до организации автоматизированных пунктов пропуска на режимных предприятиях и содействия в выполнении каждодневных задач органов внутренних дел и силовых структур. Ввиду постоянно растущей террористической угрозы и общей нестабильности межгосударственных отношений использование данных технологий становится неотъемлемой частью решения задач государственной важности.

* 1. Цели

При выполнении бакалаврской работы были поставлены следующие задачи:

* Проанализировать современные методы распознавания лиц
* Написать программу для распознавания лиц при помощи нейронной сети
  1. Используемые средства для выполнения работы
     1. Язык программирования Python

Причины выбора этого языка:

* Он не связан с какой-либо одной операционной системой или машиной;
* Этот язык поддерживает основные парадигмы программирования, которые нужны для выполнения данной работы;
* Легко читаемый код;
* Множество полезных библиотек.
  + 1. Библиотека NumPy

Библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых и математических функций для операций с этими массивами, которые работают достаточно быстро за счет использования вставок на языках: C, C++ и Fortran.

* + 1. Библиотека OpenCV

Библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Используемые средства из этой библиотеки в данной работе:

* Методы распознавания лиц (EigenFaces, FisherFaces, LBPH);
* Алгоритм Лукаса-Канаде для отслеживания лиц;
* Алгоритм Виолы-Джонса для детектирования лиц;
* И другие разные функции для преобразования изображений.

ГЛАВА 2. ОБЗОР СОВРЕМЕННЫХ МЕТОДОВ РАСПОЗНАВАНИЯ ЛИЦ

Несмотря на большое разнообразие представленных алгоритмов, можно выделить общую структуру процесса распознавания лиц:

На первом этапе производится детектирование и локализация лица на изображении. На этапе распознавания производится выравнивание изображения лица (геометрическое и яркостное), вычисление признаков и непосредственно распознавание – сравнение вычисленных признаков с заложенными в базу данных эталонами. Основным отличием всех представленных алгоритмов будет вычисление признаков и сравнение их совокупностей между собой.

1. 1. Метод гибкого сравнения на графах (Elastic graph matching)

Суть метода сводится к эластичному сопоставлению графов, описывающих изображения лиц. Лица представлены в виде графов со взвешенными вершинами и ребрами. На этапе распознавания один из графов – эталонный – остается неизменным, в то время как другой деформируется с целью наилучшей подгонки к первому. В подобных системах распознавания графы могут представлять собой как прямоугольную решетку, так и структуру, образованную характерными (антропометрическими) точками лица (рис.2.1), (рис. 2.2).

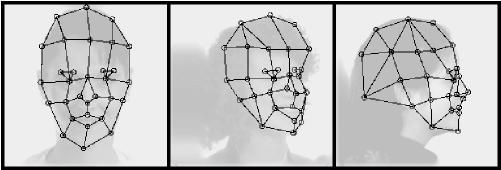


Рисунок 2.1 Граф на основе антропометрических точек лица.

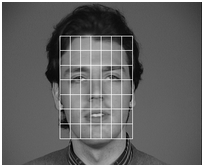


Рисунок 2.2 Регулярная решетка.

В вершинах графа вычисляются значения признаков, чаще всего используют комплексные значения фильтров Габора или их упорядоченных наборов – Габоровских вейвлет (строи Габора), которые вычисляются в некоторой локальной области вершины графа локально путем свертки значений яркости пикселей с фильтрами Габора (рис. 2.3), (рис. 2.4).

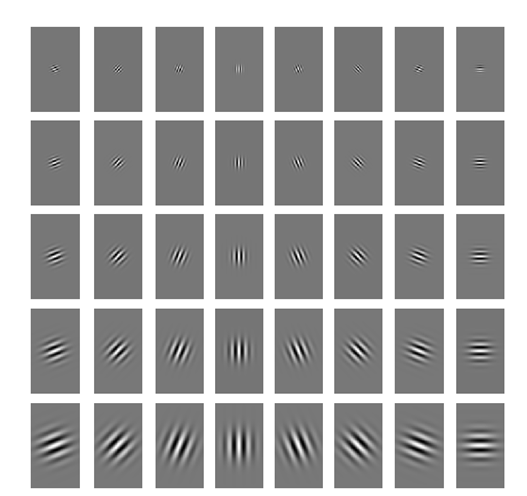


Рисунок 2.3 Набор (банк, jet) фильтров Габора.



Рисунок 2.4 Свертка изображения лица с двумя фильтрами Габора.

Ребра графа взвешиваются расстояниями между смежными вершинами. Различие (расстояние, дискриминационная характеристика) между двумя графами вычисляется при помощи некоторой ценовой функции деформации, учитывающей как различие между значениями признаков, вычисленными в вершинах, так и степень деформации ребер графа [1].

Деформация графа происходит путем смещения каждой из его вершин на некоторое расстояние в определённых направлениях относительно ее исходного местоположения и выбора такой ее позиции, при которой разница между значениями признаков (откликов фильтров Габора) в вершине деформируемого графа и соответствующей ей вершине эталонного графа будет минимальной. Данная операция выполняется поочередно для всех вершин графа до тех пор, пока не будет достигнуто наименьшее суммарное различие между признаками деформируемого и эталонного графов. Значение ценовой функции деформации при таком положении деформируемого графа и будет являться мерой различия между входным изображением лица и эталонным графом. Данная «релаксационная» процедура деформации должна выполняться для всех эталонных лиц, заложенных в базу данных системы. Результат распознавания системы – эталон с наилучшим значением ценовой функции деформации (рис. 2.5).

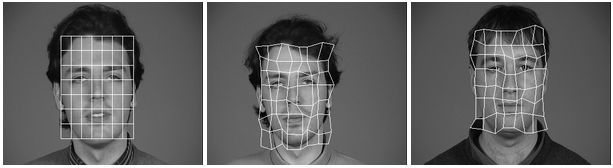


Рисунок 2.5 Пример деформации графа в виде регулярной решетки.

В отдельных публикациях указывается 95-97%-ая эффективность распознавания даже при наличии различных эмоциональных выражениях и изменении ракурса лица до 15 градусов. Однако разработчики систем эластичного сравнения на графах ссылаются на высокую вычислительную стоимость данного подхода. Например, для сравнения входного изображения лица с 87 эталонными тратилось приблизительно 25 секунд при работе на параллельной ЭВМ с 23 транспьютерами. В других публикациях по данной тематике время либо не указывается, либо говорится, что оно велико [2].

Недостатки: высокая вычислительная сложность процедуры распознавания. Низкая технологичность при запоминании новых эталонов. Линейная зависимость времени работы от размера базы данных лиц.

* 1. Нейронные сети

В настоящее время существует около десятка разновидности нейронных сетей (НС). Одним из самых широко используемых вариантов являться сеть, построенная на многослойном перцептроне, которая позволяет классифицировать поданное на вход изображение/сигнал в соответствии с предварительной настройкой/обучением сети.

Обучаются нейронные сети на наборе обучающих примеров. Суть обучения сводится к настройке весов межнейронных связей в процессе решения оптимизационной задачи методом градиентного спуска. В процессе обучения НС происходит автоматическое извлечение ключевых признаков, определение их важности и построение взаимосвязей между ними. Предполагается, что обученная НС сможет применить опыт, полученный в процессе обучения, на неизвестные образы за счет обобщающих способностей.

Наилучшие результаты в области распознавания лиц (по результатам анализа публикаций) показала Convolutional Neural Network или сверточная нейронная сеть (далее – СНС), которая является логическим развитием идей таких архитектур НС как когнитрона и неокогнитрона. Успех обусловлен возможностью учета двумерной топологии изображения, в отличие от многослойного перцептрона [3].

Отличительными особенностями СНС являются локальные рецепторные поля (обеспечивают локальную двумерную связность нейронов), общие веса (обеспечивают детектирование некоторых черт в любом месте изображения) и иерархическая организация с пространственными сэмплингом (spatial subsampling). Благодаря этим нововведениям СНС обеспечивает частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям (рис. 2.6).

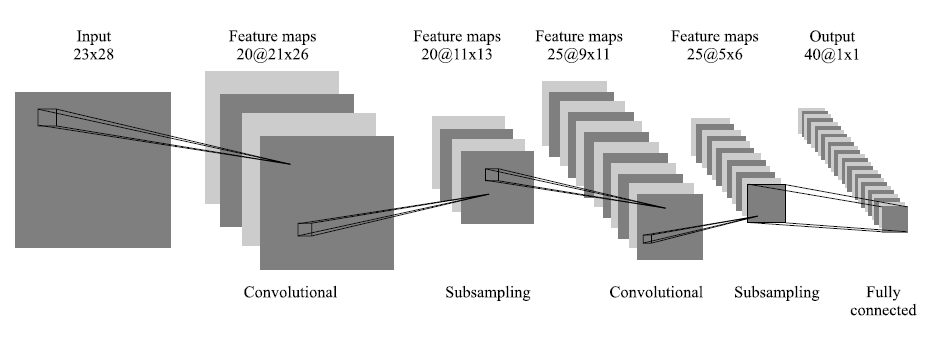


Рисунок 2.6 Схематичное изображение архитектуры сверточной нейронной сети.

Тестирование СНС на базе данных ORL, содержащей изображения лиц с небольшими изменениями освещения, масштаба, пространственных поворотов, положения и различными эмоциями, показало 96% точность распознавания.

Свое развитие СНС получили в разработке DeepFace (рис. 2.7), которую приобрел Facebook для распознавания лиц пользователей своей соцсети. Все особенности архитектуры носят закрытый характер.

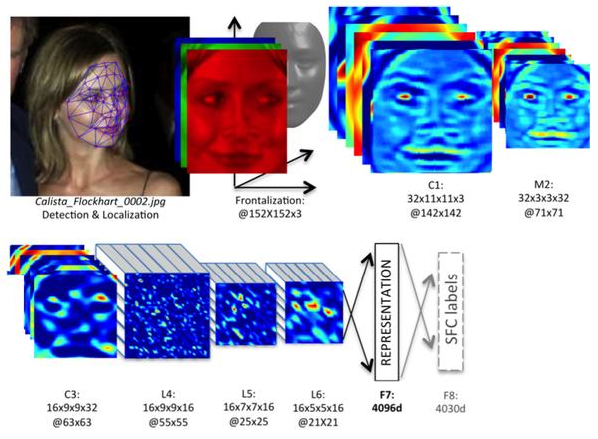


Рисунок 2.7 Принцип работы DeepFace.

Недостатки нейронных сетей: добавление нового эталонного лица в базу данных требует полного переобучения сети на всем имеющемся наборе (достаточно длительная процедура, в зависимости от размера выборки от 1 часа до нескольких дней). Проблемы математического характера, связанные с обучением: попадание в локальный оптимум, выбор оптимального шага оптимизации, переобучение и т. д. Трудно формализуемый этап выбора архитектуры сети (количество нейронов, слоев, характер связей). Обобщая все вышесказанное, можно заключить, что НС – «черный ящик» с трудно интерпретируемыми результатами работы [4].

* 1. Скрытые Марковские модели (СММ, HMM)

Одним из статистических методов распознавания лиц являются скрытые Марковские модели (СММ) с дискретным временем. СММ используют статистические свойства сигналов и учитывают непосредственно их пространственные характеристики. Элементами модели являются: множество скрытых состояний, множество наблюдаемых состояний, матрица переходных вероятностей, начальная вероятность состояний. Каждому соответствует своя Марковская модель. При распознавании объекта проверяются сгенерированные для заданной базы объектов Марковские модели и ищется максимальная из наблюдаемых вероятность того, что последовательность наблюдений для данного объекта сгенерирована соответствующей моделью [5].

На сегодняшний день не удалось найти примера коммерческого применения СММ для распознавания лиц.

Недостатки:

* необходимо подбирать параметры модели для каждой базы данных;
* СММ не обладает различающей способностью, то есть алгоритм обучения только максимизирует отклик каждого изображения на свою модель, но не минимизирует отклик на другие модели.
  1. Метод главных компонент или principal component analysis (PCA)

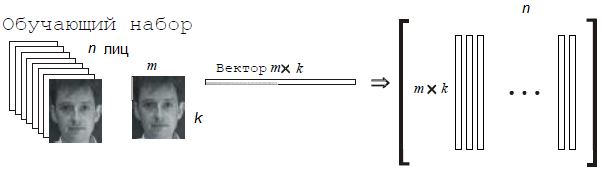
Одним из наиболее известных и проработанных является метод главных компонент (principal component analysis, PCA), основанный на преобразовании Карунена-Лоева.

Первоначально метод главных компонент начал применяться в статистике для снижения пространства признаков без существенной потери информации. В задаче распознавания лиц его применяют главным образом для представления изображения лица вектором малой размерности (главных компонент), который сравнивается затем с эталонными векторами, заложенными в базу данных.

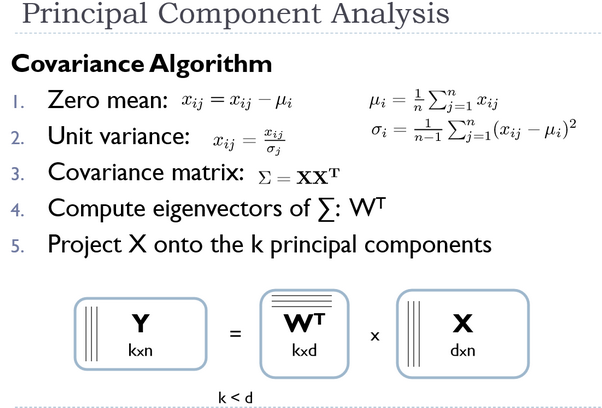
Главной целью метода главных компонент является значительное уменьшение размерности пространства признаков таким образом, чтобы оно как можно лучше описывало «типичные» образы, принадлежащие множеству лиц. Используя этот метод можно выявить различные изменчивости в обучающей выборке изображений лиц и описать эту изменчивость в базисе нескольких ортогональных векторов, которые называются собственными (eigenface).

Полученный один раз на обучающей выборке изображений лиц набор собственных векторов используется для кодирования всех остальных изображений лиц, которые представляются взвешенной комбинацией этих собственных векторов. Используя ограниченное количество собственных векторов можно получить сжатую аппроксимацию входному изображению лица, которую затем можно хранить в базе данных в виде вектора коэффициентов, служащего одновременно ключом поиска в базе данных лиц [6].

Суть метода главных компонент сводится к следующему. Вначале весь обучающий набор лиц преобразуется в одну общую матрицу данных, где каждая строка представляет собой один экземпляр изображения лица, разложенного в строку. Все лица обучающего набора должны быть приведены к одному размеру и с нормированными гистограммами (рис. 2.8).

  
Рисунок 2.8 Преобразования обучающего набора лиц в одну общую матрицу X.

Затем производится нормировка данных и приведение строк к 0-му среднему и 1-й дисперсии, вычисляется матрица ковариации. Для полученной матрицы ковариации решается задача определения собственных значений и соответствующих им собственных векторов (собственные лица). Далее производится сортировка собственных векторов в порядке убывания собственных значений и оставляют только первые k векторов по правилу (рис 2.9):

https://habrastorage.org/files/077/2e6/884/0772e6884e704d4db05c2d35297669d5.png  
  
Рисунок 2.9 Алгоритм РСА.

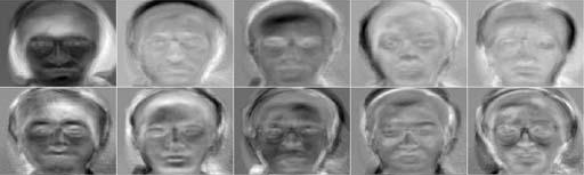


Рисунок 2.10 Пример первых десяти собственных векторов (собственных лиц), полученных на обучаемом наборе лиц.

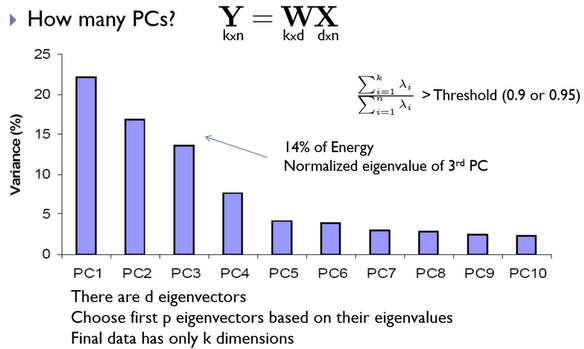
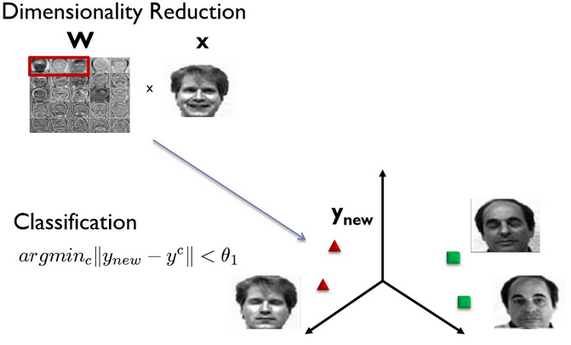
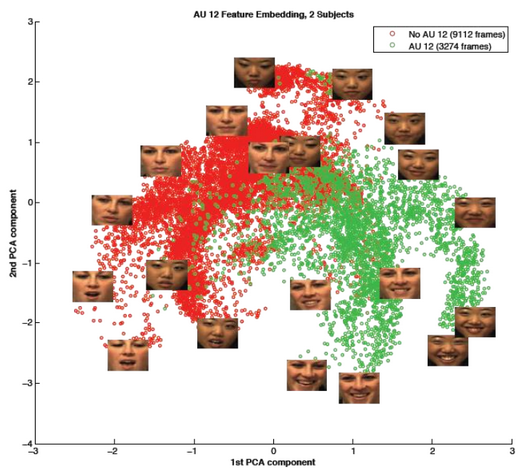
= 0.956\*-1.842\*+0.046 …  
  
Рисунок 2.11 Пример построения (синтеза) человеческого лица с помощью комбинации собственных лиц и главных компонент.  
  
  
Рисунок 2.12 Принцип выбора базиса из первых собственных векторов.  
Рисунок 2.13 Пример отображения лица в трехмерное метрическое пространство, полученном по трем собственным лицам и дальнейшее распознавание.  
  


Рисунок 2.14 Метод главных компонент.

Метод главных компонент (рис. 2.14) хорошо зарекомендовал себя в практических приложениях. Однако, в тех случаях, когда на изображении лица присутствуют значительные изменения в освещенности или выражении лица, эффективность метода значительно падает. Все дело в том, что PCA выбирает подпространство с такой целью, чтобы максимально аппроксимировать входной набор данных, а не выполнить дискриминацию между классами лиц [7].

Было предложено решение этой проблемы с использование линейного дискриминанта Фишера (в литературе встречается название “Eigen-Fisher”, “Fisherface”, LDA). LDA выбирает линейное подпространство, которое максимизирует отношение:

, (2.1)

где

(2.2)

матрица межклассового разброса, и

(2.3)

Матрица внутриклассового разброса; m – число классов в базе данных.

LDA ищет проекцию данных, при которой классы являются максимально линейно сепарабельны (см. рисунок ниже). Для сравнения PCA ищет такую проекцию данных, при которой будет максимизирован разброс по всей базе данных лиц (без учета классов). По результатам экспериментов в условиях сильного бакового и нижнего затенения изображений лиц Fisherface показал 95% эффективность по сравнению с 53% Eigenface.

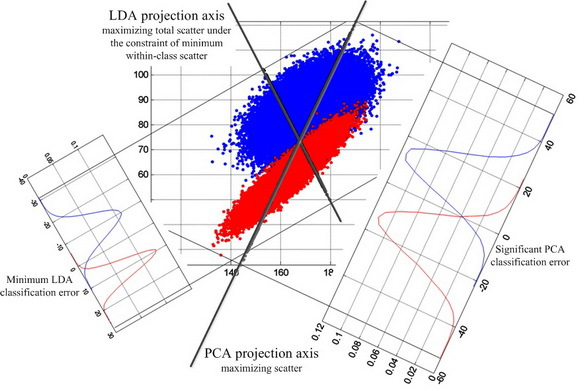


Рисунок 2.15 Принципиальное отличие формирования проекций PCA и LDA

* 1. Active Shape Models (ASM)

Суть метода ASM заключается в учете статистических связей между расположением антропометрических точек. На имеющейся выборке изображений лиц, снятых в анфас. На изображении эксперт размечает расположение антропометрических точек. На каждом изображении точки пронумерованы в одинаковом порядке.

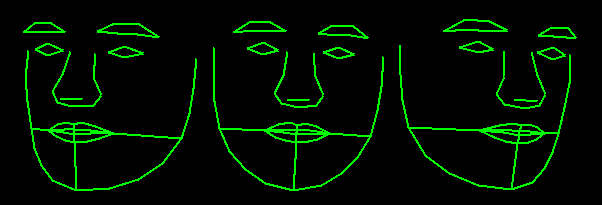
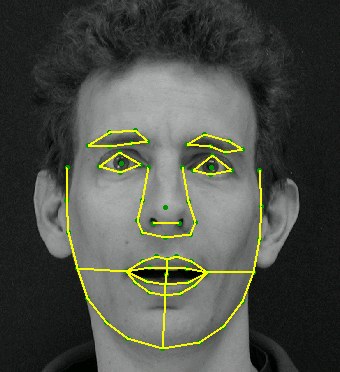


Рисунок 2.16 Пример представления формы лица с использованием 68 точек.

Для того чтобы привести координаты на всех изображениях к единой системе обычно выполняется т.н. обобщенный прокрустов анализ, в результате которого все точки приводятся к одному масштабу и центрируются. Далее для всего набора образов вычисляется средняя форма и матрица ковариации. На основе матрицы ковариации вычисляются собственные вектора, которые затем сортируются в порядке убывания соответствующих им собственных значений. Модель ASM определяется матрицей Φ и вектором средней формы s ̅.

Тогда любая форма может быть описана с помощью модели и параметров:

(2.4)

Локализации ASM модели на новом, не входящем в обучающую выборку изображении осуществляется в процессе решения оптимизационной задачи. Однако все же главной целью AAM и ASM является не распознавание лиц, а точная локализация лица и антропометрических точек на изображении для дальнейшей обработки. Практически во всех алгоритмах обязательным этапом, предваряющим классификацию, является выравнивание, под которым понимается выравнивание изображения лица во фронтальное положение относительно камеры или приведение совокупности лиц (например, в обучающей выборке для обучения классификатора) к единой системе координат. Для реализации этого этапа необходима локализация на изображении характерных для всех лиц антропометрических точек – чаще всего это центры зрачков или уголки глаз. Разные исследователи выделяют разные группы таких точек. В целях сокращения вычислительных затрат для систем реального времени разработчики выделяют не более 10 таких точек. Модели AAM и ASM как раз и предназначены для того чтобы точно локализовать эти антропометрические точки на изображении лица [8].

Основные проблемы, связанные с разработкой систем распознавания лиц [11]:

* Проблема освещенности (рис. 2.17)



Рисунок 2.17.

* Проблема положения головы (рис. 2.18).

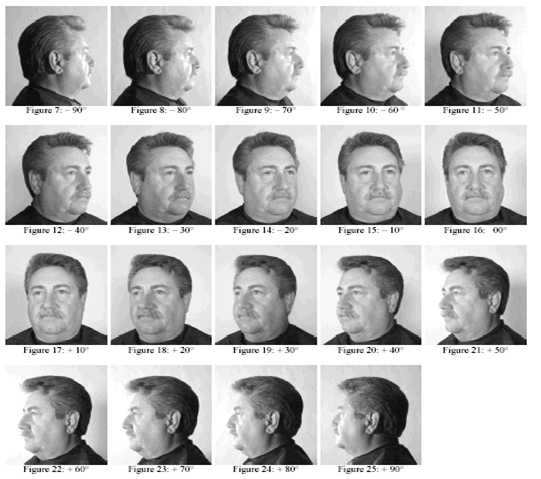


Рисунок 2.18.

С целью оценки эффективности предложенных алгоритмов распознавания лиц агентство DARPA и исследовательская лаборатория армии США разработали программу FERET (face recognition technology).

В масштабных тестах программы FERET принимали участие алгоритмы, основанные на гибком сравнении на графах и всевозможные модификации метода главных компонент (PCA). Эффективность всех алгоритмов была примерно одинаковой. В этой связи трудно или даже невозможно провести четкие различия между ними (особенно если согласовать даты тестирования). Для фронтальных изображений, сделанных в один и тот же день, приемлемая точность распознавания, как правило, составляет 95%. Для изображений, сделанных разными аппаратами и при разном освещении, точность, как правило, падает до 80%. Для изображений, сделанных с разницей в год, точность распознавания составило примерно 50%. При этом стоит заметить, что даже 50 процентов — это более чем приемлемая точность работы системы подобного рода[9].

Ежегодно FERET публикует отчет о сравнительном испытании современных систем распознавания лиц на базе лиц более одного миллиона. К большому сожалению в последних отчетах не раскрываются принципы построения систем распознавания, а публикуются только результаты работы коммерческих систем. На сегодняшний день лидирующей является система NeoFace разработанная компанией NEC.

* 1. Метод Виолы-Джонса (Viola-Jones)

Хотя метод был разработан и представлен в 2001 году Полом Виолой и Майклом Джонсом, он до сих пор является основополагающим для поиска объектов на изображении в реальном времени.

Основные принципы, на которых основан метод, таковы:

* используются изображения в [интегральном представлении](http://habrahabr.ru/blogs/algorithm/102919/), что позволяет вычислять быстро необходимые объекты;
* используются [признаки Хаара](http://en.wikipedia.org/wiki/Haar-like_features), с помощью которых происходит поиск нужного объекта (в данном контексте, лица и его черт);
* используется [бустинг](http://machinelearning.ru/wiki/index.php?title=%D0%91%D1%83%D1%81%D1%82%D0%B8%D0%BD%D0%B3) (от англ. boost – улучшение, усиление) для выбора наиболее подходящих признаков для искомого объекта на данной части изображения;
* все признаки поступают на вход [классификатора](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D0%B8), который даёт результат «верно» либо «ложь»;
* используются [каскады признаков](http://courses.graphicon.ru/main/vision/2011/lectures/7) для быстрого отбрасывания окон, где не найдено лицо.

Обучение классификаторов идет очень медленно, но результаты поиска лица очень быстры, именно поэтому был выбран данный метод распознавания лиц на изображении. Виола-Джонс является одним из лучших по соотношению показателей эффективность распознавания/скорость работы. Также этот детектор обладает крайне низкой вероятностью ложного обнаружения лица. Алгоритм даже хорошо работает и распознает черты лица под небольшим углом, примерно до 30 градусов. При угле наклона больше 30 градусов процент обнаружений резко падает. И это не позволяет в стандартной реализации детектировать повернутое лицо человека под произвольным углом, что в значительной мере затрудняет или делает невозможным использование алгоритма в современных производственных системах с учетом их растущих потребностей[10].

Требуется подробный разбор принципов, на которых основан алгоритм Виолы-Джонса. Данный метод в общем виде ищет лица и черты лица по общему принципу сканирующего окна (рис. 2.19).

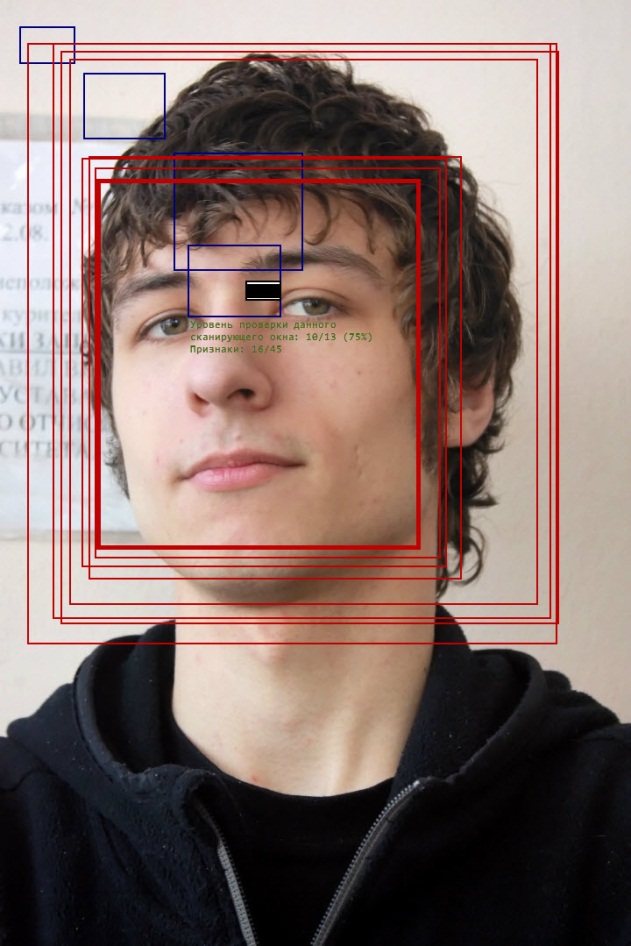


Рисунок 2.19 Визуализация сканирующего окна в программе.

В общем виде, задача обнаружения лица и черт лица человека на цифровом изображении выглядит именно так:

имеется изображение, на котором есть искомые объекты. Оно представлено двумерной матрицей пикселей размером w\*h, в которой каждый пиксель имеет значение:

-от 0 до 255, если это черно-белое изображение;

-от 0 до 2553, если это цветное изображение (компоненты R, G, B).

в результате своей работы, алгоритм должен определить лица и их черты и пометить их – поиск осуществляется в активной области изображения прямоугольными признаками, с помощью которых и описывается найденное лицо и его черты:

rectanglei ={x,y,w,h,a}, где x, y – координаты центра i-го прямоугольника, w – ширина, h – высота, a – угол наклона прямоугольника к вертикальной оси изображения.

Иными словами, применительно к рисункам и фотографиям используется подход на основе сканирующего окна (scanning window): сканируется изображение окном поиска (так называемое, окно сканирования), а затем применяется классификатор к каждому положению. Система обучения и выбора наиболее значимых признаков полностью автоматизирована и не требует вмешательства человека, поэтому данный подход работает быстро[11].

Алгоритм сканирования окна с признаками выглядит так:

* есть исследуемое изображение, выбрано окно сканирования, выбраны используемые признаки;
* далее окно сканирования начинает последовательно двигаться по изображению с шагом в 1 ячейку окна (допустим, размер самого окна есть 24\*24 ячейки);
* при сканировании изображения в каждом окне вычисляется приблизительно 200 000 вариантов расположения признаков, за счет изменения масштаба признаков и их положения в окне сканирования;
* сканирование производится последовательно для различных масштабов;
* масштабируется не само изображение, а сканирующее окно (изменяется размер ячейки);
* все найденные признаки попадают к классификатору, который «выносит вердикт».

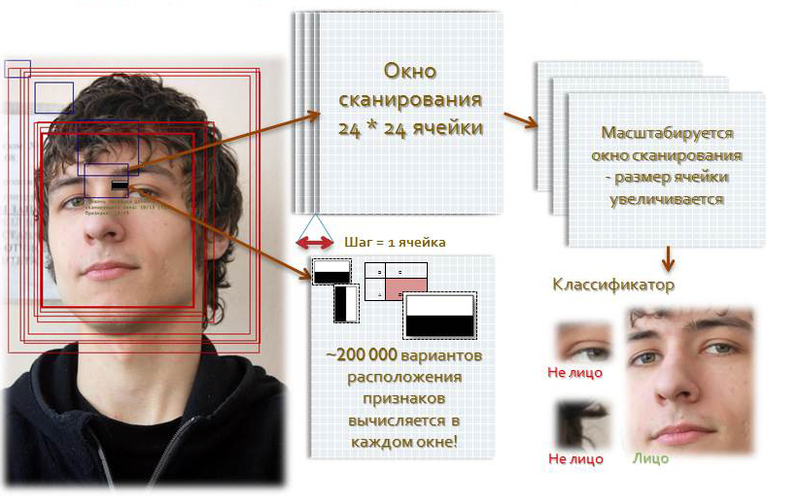


Рисунок 2.20.

В процессе поиска вычислять все признаки на маломощных настольных ПК просто нереально. Следовательно, классификатор должен реагировать только на определенное, нужное подмножество всех признаков. Совершенно логично, что надо обучить классификатор нахождению лиц по данному определенному подмножеству. Это можно сделать, обучая вычислительную машину автоматически [12].

Используемая в алгоритме модель машинного обучения

Обучение машины — это процесс получения модулем новых знаний. Есть признанное определение данному процессу:

«Машинное обучение (рис. 2.21) — это наука, изучающая компьютерные алгоритмы, автоматически улучшающиеся во время работы» (Michel, 1996).

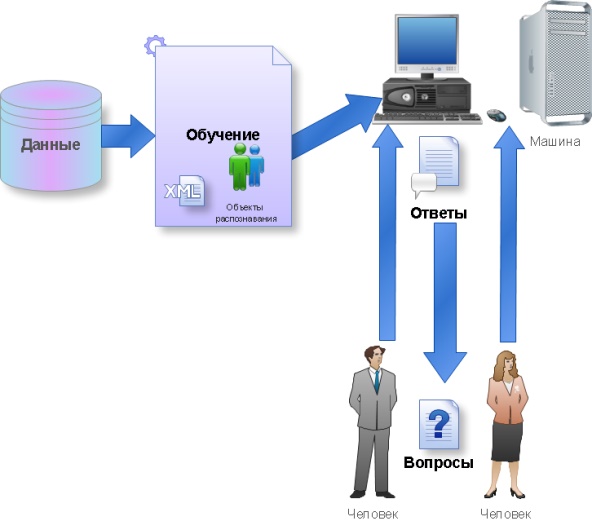


Рисунок 2.21 Обучение машины.

Данный процесс входит в концепцию и технологию под названием [Data mining](http://ru.wikipedia.org/wiki/Data_mining) (извлечение информации и интеллектуальный анализ данных), куда входят помимо Машинного обучения такие дисциплины, как Теория баз данных, Искусственный интеллект, Алгоритмизация, Распознавание образов и прочие.

Машинное обучение в методе Виолы-Джонса решает такую задачу как классификация.

Обучение классификатора в методе Виолы-Джонса

В контексте алгоритма, имеется множество объектов (изображений), разделённых некоторым образом на классы. Задано конечное множество изображений, для которых известно, к какому классу они относятся (к примеру, это может быть класс «фронтальное положение носа»). Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект — значит, указать номер (или наименование класса), к которому относится данный объект.

Классификация объекта — номер или наименование класса, выдаваемые алгоритмом классификации в результате его применения к данному конкретному объекту.

Классификатор(classifier) — в задачах классификации это аппроксимирующая функция, выносящая решение, к какому именно классу данный объект принадлежит.

Обучающая выборка – конечное число данных.

В машинном обучении задача классификации относится к разделу обучения с учителем когда классы поделены. Распознавание образов по сути своей и есть классификация изображений и сигналов. В случае алгоритма Виолы-Джонса для идентификации и распознавания лица классификация является двухклассовой.

Постановка классификации выглядит следующим образом:

Есть X – множество, в котором хранится описание объектов, Y – конечное множество номеров, принадлежащих классам. Между ними есть зависимость – отображение Y\*: X => Y. Обучающая выборка представлена Xm = {(x1,y1), …, (xm,ym)}. Конструируется функция f от вектора признаков X, которая выдает ответ для любого возможного наблюдения X и способна классифицировать объект x∈X. Данное простое правило должно хорошо работать и на новых данных.

Применяемый в алгоритме бустинг и разработка AdaBoost

Для решения проблемы данного, столь сложного обучения существует технология бустинга.

Бустинг — комплекс методов, способствующих повышению точности аналитических моделей. Эффективная модель, допускающая мало ошибок классификации, называется «сильной». «Слабая» же, напротив, не позволяет надежно разделять классы или давать точные предсказания, делает в работе большое количество ошибок. Поэтому бустинг (от англ. boosting – повышение, усиление, улучшение) означает дословно «усиление» «слабых» моделей – это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов [13].

Идея бустинга была предложена Робертом Шапиром (Schapire) в конце 90-х годов, когда надо было найти решение вопроса о том, чтобы имея множество плохих (незначительно отличающихся от случайных) алгоритмов обучения, получить один хороший. В основе такой идеи лежит построение цепочки (ансамбля) классификаторов, который называется каскадом, каждый из которых (кроме первого) обучается на ошибках предыдущего. Например, один из первых алгоритмов бустинга Boost1 использовал каскад из 3-х моделей, первая из которых обучалась на всем наборе данных, вторая – на выборке примеров, в половине из которых первая дала правильные ответы, а третья — на примерах, где «ответы» первых двух разошлись. Таким образом, имеет место последовательная обработка примеров каскадом классификаторов, причем так, что задача для каждого последующего становится труднее. Результат определяется путем простого голосования: пример относится к тому классу, который выдан большинством моделей каскада.  
Бустинг представляет собой [жадный алгоритм](http://ru.wikipedia.org/wiki/%D0%96%D0%B0%D0%B4%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) построения композиции алгоритмов (greedy algorithm) — это алгоритм, который на каждом шагу делает локально наилучший выбор в надежде, что итоговое решение будет оптимальным. Бустинг над решающими деревьями считается одним из наиболее эффективных методов с точки зрения качества классификации. Во многих экспериментах наблюдалось практически неограниченное уменьшение частоты ошибок на независимой тестовой выборке по мере наращивания композиции. Более того, качество на тестовой выборке часто продолжало улучшаться даже после достижения безошибочного распознавания всей обучающей выборки. Это перевернуло существовавшие долгое время представления о том, что для повышения обобщающей способности необходимо ограничивать сложность алгоритмов. На примере бустинга стало понятно, что хорошим качеством могут обладать сколь угодно сложные композиции, если их правильно настраивать [14].

Математически бустинг объясняется так:

Наряду с множествами X и Y вводится вспомогательное множество R, называемое пространством оценок. Рассматриваются алгоритмы, имеющие вид суперпозиции a(x) = C(b(x)), где функция b: X → R называется алгоритмическим оператором, функция C: R → Y –решающим правилом.

Многие алгоритмы классификации имеют именно такую структуру: сначала вычисляются оценки принадлежности объекта классам, затем решающее правило переводит эти оценки в номер класса. Значение оценки, как правило, характеризует степень уверенности классификации.

Алгоритмическая композиция – алгоритм a: X → Y вида  
a(x) = C(F(b1(x), . . . , bT (x)), x ∈ X, составленный из алгоритмических операторов bt :X→R, t=1,..., T, корректирующей операции F: RT→R и решающего правила C: R→Y.

Базовыми алгоритмами обозначаются функции at(x) = C(bt(x)), а при фиксированном решающем правиле C — и сами операторы bt(x).

Суперпозиции вида F(b1,..., bT ) являются отображениями из X в R, то есть, опять же, алгоритмическими операторами.

В задачах классификации на два непересекающихся класса в качестве пространства оценок обычно используется множество действительных чисел. Решающие правила могут иметь настраиваемые параметры. Так, в алгоритме Виолы-Джонса используется пороговое решающее правило, где, как правило, сначала строится оператор при нулевом значении, а затем подбирается значение оптимальное. Процесс последовательного обучения базовых алгоритмов применяется, пожалуй, чаще всего при построении композиций.  
Критерии останова могут использоваться различные, в зависимости от специфики задачи, возможно также совместное применение нескольких критериев [15]:

- построено заданное количество базовых алгоритмов T;

- достигнута заданная точность на обучающей выборке;

- достигнутую точность на контрольной выборке не удаётся улучшить на протяжении последних нескольких шагов при определенном параметре алгоритма.

Развитием данного подхода явилась разработка более совершенного семейства алгоритмов бустинга AdaBoost (adaptive boosting – адаптированное улучшение), предложенная Йоавом Фройндом (Freund) и Робертом Шапиром (Schapire) в 1999 году, который может использовать произвольное число классификаторов и производить обучение на одном наборе примеров, поочередно применяя их на различных шагах.

Рассматривается задача классификации на два класса, Y = {−1,+1}. К примеру, базовые алгоритмы также возвращают только два ответа −1 и +1, и решающее правило фиксировано: C(b) = sign(b).

Функционал качества композиции Qt определяется как число ошибок, допускаемых ею на обучающей выборке

Для решения задачи AdaBoosting’а нужна экспоненциальная аппроксимация пороговой функции потерь [z<0], причем экспонента Ez = e-z.

Плюсы AdaBoost:

хорошая обобщающая способность. В реальных задачах практически всегда строятся композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться по мере увеличения числа базовых алгоритмов;

простота реализации;

собственные накладные расходы бустинга невелики. Время построения композиции практически полностью определяется временем обучения базовых алгоритмов;

возможность идентифицировать объекты, являющиеся шумовыми выбросами. Это наиболее «трудные» объекты xi, для которых в процессе наращивания композиции веса wi принимают наибольшие значения.

Минусы AdaBoost:

Бывает переобучение при наличии значительного уровня шума в данных. Экспоненциальная функция потерь слишком сильно увеличивает веса «наиболее трудных» объектов, на которых ошибаются многие базовые алгоритмы. Однако именно эти объекты чаще всего оказываются шумовыми выбросами. В результате AdaBoost начинает настраиваться на шум, что ведёт к переобучению. Проблема решается путём удаления выбросов или применения менее «агрессивных» функций потерь. В частности, применяется алгоритм GentleBoost [16];

AdaBoost требует достаточно длинных обучающих выборок. Другие методы линейной коррекции, в частности, бэггинг, способны строить алгоритмы сопоставимого качества по меньшим выборкам данных;

Бывает построение неоптимального набора базовых алгоритмов. Для улучшения композиции можно периодически возвращаться к ранее построенным алгоритмам и обучать их заново.

Бустинг может приводить к построению громоздких композиций, состоящих из сотен алгоритмов. Такие композиции исключают возможность содержательной интерпретации, требуют больших объёмов памяти для хранения базовых алгоритмов и существенных временных затрат на вычисление классификаций.

В наши дни подход усиления простых классификаторов является популярным и, вероятно, наиболее эффективным методом классификации за счёт высокой скорости и эффективности работы и относительной простоты реализации.

Принципы решающего дерева в разрабатываемом алгоритме

Дерево принятия решений — это дерево, в листьях которого стоят значения целевой функции, а в остальных узлах — условия перехода (к примеру, на Лице есть Улыбка), определяющие по какому из ребер идти. Если для данного наблюдения условие равно истине то осуществляется переход по левому ребру, если же ложь — по правому.

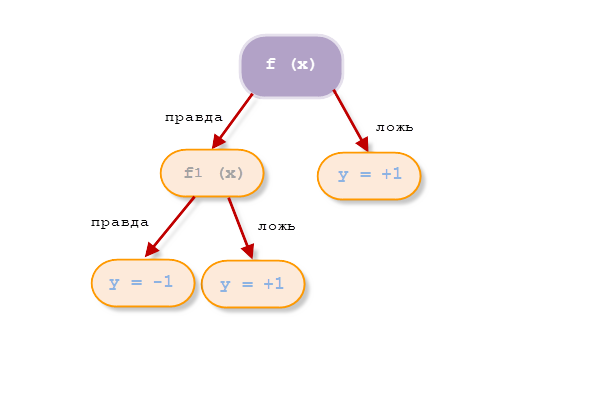


Рисунок 2.22 Решающее дерево.

Достоинствами таких решающих деревьев являются наглядность, легкость работы с ними, быстродействие. Также, они легко применяются для задач с множеством классов.

Задача поиска и нахождения лиц на изображении с помощью данного принципа часто бывает очередным шагом на пути к распознаванию характерных черт, к примеру, верификации человека по распознанному лицу или распознавания мимики лица.

Признаки Хаара

Признак — отображение f: X => Df, где Df — множество допустимых значений признака. Если заданы признаки f1,…,fn, то вектор признаков x = (f1(x),…,fn(x)) называется признаковым описанием объекта x ∈ X.

Признаковые описания допустимо отождествлять с самими объектами. При этом множество X = Df1\* …\* Dfn называют признаковым пространством.

Признаки делятся на следующие типы в зависимости от множества Df:

* бинарный признак, Df = {0,1};
* номинальный признак: Df — конечное множество;
* порядковый признак: Df — конечное упорядоченное множество;
* количественный признак: Df — множество действительных чисел.

Естественно, бывают прикладные задачи с разнотипными признаками, для их решения подходят далеко не все методы.  
В стандартном методе Виолы – Джонса используются прямоугольные признаки, изображенные на рисунке ниже, они называются примитивами Хаара (рис. 2.23):

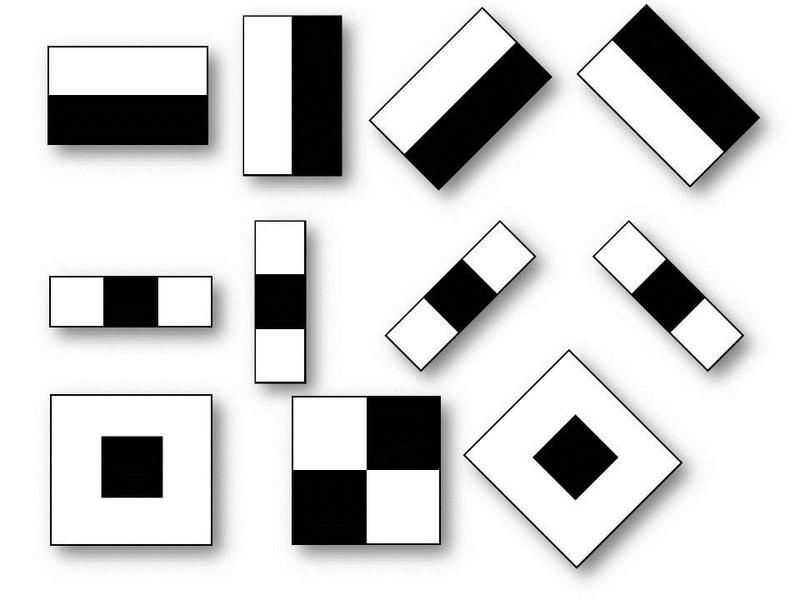


Рисунок 2.23 Примитивы Хаара.

В расширенном методе Виолы – Джонса, использующемся в библиотеке OpenCV используются дополнительные признаки (рис. 2.24):

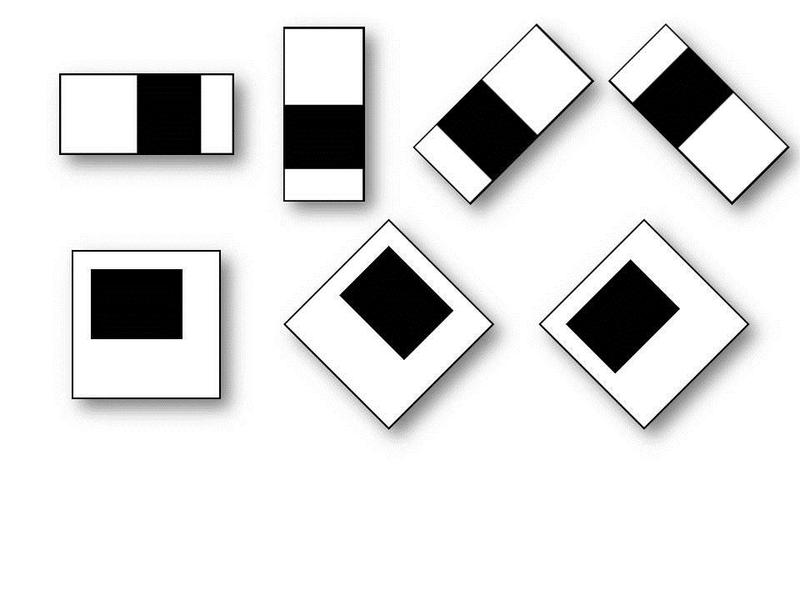


Рисунок 2.24 Дополнительные примитивы Хаара.

Вычисляемым значением такого признака будет  
F = X-Y, где X – сумма значений яркостей точек закрываемых светлой частью признака, а Y – сумма значений яркостей точек закрываемых темной частью признака. Для их вычисления используется понятие интегрального изображения, рассмотренное выше. Признаки Хаара дают точечное значение перепада яркости по оси X и Y соответственно[17].

Прямоугольные признаки Хаара

Простейший прямоугольный признак Хаара можно определить как разность сумм пикселей двух смежный областей внутри прямоугольника, который может занимать различные положения и масштабы на изображении. Такой вид признаков называется 2-прямоугольным. Виола и Джонс так же определили 3-прямоугольные и 4-прямоугольные признаки. Каждый признак может показать наличие (или отсутствие) какой-либо конкретной характеристики изображения, такой как границы или изменение текстур. Например, 2-прямоугольный признак может показать, где находится граница между темным и светлым регионами.

Наклонные признаки Хаара

Линхарт и Майд представили идею наклоненных (45 градусов) признаков Хаара. Это было сделано для увеличения размерности пространства признаков. Способ оказался удачным и некоторые наклонные признаки были способны лучше описывать объект. Например, 2-прямоугольный наклонный признак Хаара может показать наличие края, наклоненного на 45 градусов.

Мессом и Барзак дополнили концепцию наклонных признаков Хаара. Хоть идея и является математически верной, на практике при использовании признаков под разными углами возникают проблемы. Для ускорения вычислений, детектор использует изображения низкого разрешения, что приводит к ошибке округления. Исходя из этого, наклонные признаки Хаара обычно не используются[18].

* 1. Недостатки существующих систем

Среди основных недостатков существующих систем можно выделить следующие:

1. требуют огромных временных затрат;
2. обладают закрытой архитектурой;
3. являются запатентованными;
4. имеют значительную погрешность;
5. сложно реализуемы;
6. ни один из методов не решает вышеописанные задачи в полном объеме.

ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ С ПОМОЩЬЮ БИБЛИОТЕКИ OPENCV И НЕЙРОННОЙ СЕТИ

1. 1. Создание обучающей выборки при помощи библиотеки OpenCV

Для определения области лица на фотографии при помощи сверточной нейронной сети нам необходима выборка. Для этого можем воспользоваться очень удобной для этой задачи библиотекой OpenCV, которая определяет эти самые лица при помощи примитивов Хаара [18]. В общем виде, задача обнаружения лица и черт лица человека на цифровом изображении выглядит так: имеется изображение, на котором есть искомые объекты. Оно представлено двумерной матрицей пикселей размером w\*h, в которой каждый пиксель имеет значение: от 0 до 255, если это черно-белое изображение и от 0 до 255^3, если это цветное изображение (компоненты R, G, B). В результате своей работы алгоритм должен определить лица и их черты и пометить их - поиск осуществляется в активной области изображения прямоугольными признаками, с помощью которых и описывается найденное лицо и его черты:

(3.1)

где x, y - координаты центра i-го прямоугольника, w - ширина, h – высота. Иными словами, применительно к рисункам и фотографиям используется подход на основе сканирующего окна (scanning window): сканируется изображение окном поиска (так называемое, окно сканирования), а затем применяется классификатор к каждому положению. Система обучения и выбора наиболее значимых признаков полностью автоматизирована и не требует вмешательства человека, поэтому данный подход работает быстро.

Алгоритм сканирования окна с признаками выглядит так: есть исследуемое изображение, выбрано окно сканирования, выбраны используемые признаки. Далее окно сканирования начинает последовательно двигаться по изображению с шагом в 1 ячейку окна (допустим, размер самого окна есть 24\*24 ячейки). Сканирование производится последовательно для различных масштабов, масштабируется не само изображение, а сканирующее окно (изменяется размер ячейки). Все найденные признаки попадают к классификатору, который «выносит вердикт». Далее создается командный файл, при помощи которого записываются все необходимые данные в файл.

В итоге на выходе мы получаем csv файл с именем файлов и необходимыми координатами, фрагмент которого представлен на рис. 3.1. Листинг программы смотри в Приложение 1.

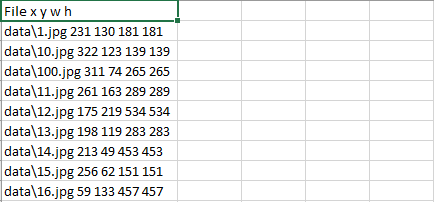


Рисунок 3.1. Фрагмент обучающей выборки.

* 1. Выбор типа нейронной сети

Наиболее часто в задачах распознавания и идентификации изображений используются классические нейросетевые архитектуры (многослойный персептрон, сети с радиально-базисной функцией и др.), но из анализа данных работ и экспериментальных исследований следует, что применение классических нейросетевых архитектур в данной задаче неэффективно по следующим причинам [19]:

* изображения имеют большую размерность, соответственно возрастает размер нейронной сети;
* большое количество параметров увеличивает вместимость системы и соответственно требует большей тренировочной выборки, увеличивает время и вычислительную сложность процесса обучения;
* для повышения эффективности работы системы желательно применять несколько нейронных сетей (обученные с различными начальными значениями синаптических коэффициентов и порядком предъявления образов), но это увеличивает вычислительную сложность решения задачи и время выполнения;
* отсутствует инвариантность к изменениям масштаба изображения, ракурсов съёмки камеры и других геометрических искажений входного сигнала.

Поэтому для решения этой задачи были выбраны сверточные нейронные сети, т.к. они обеспечивают частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям. Архитектура сверточной нейронной сети состоит из многих слоёв. Слои бывают двух типов: сверточные и подвыборочные, они чередуются друг с другом как показано на рис. 3.2. [20]

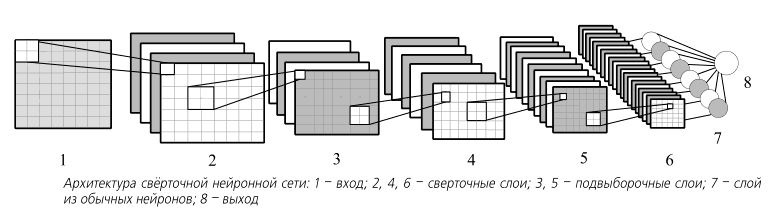


Рисунок 3.2. Архитектура сверточной нейронной сети.

* 1. Разработка нейронной сети

Разработана нейронная сеть, состоящая из 4 слоёв. Входными данными нейронной сети являются изображения размером 48 x 48 пикселей. В качестве обучения сети используется метод fit. Этому методу необходимо указать данные, на которых мы будем обучать сеть, правильные ответы, метод оптимизации и количество эпох (количество эпох указывает сколько раз мы выполняем обучение с помощью нашего набора данных). Для обучения описанной нейронной сети был использован алгоритм обратного распространения ошибки (backpropagation). Этот метод обучения многослойной нейронной сети называется обобщенным дельта-правилом. Метод был предложен в 1986 г. Румельхартом, Макклеландом и Вильямсом. Для выходного слоя корректировка весов интуитивна понятна, но для скрытых слоев долгое время не было известно алгоритма. Веса скрытого нейрона должны изменяться прямо пропорционально ошибке тех нейронов, с которыми данный нейрон связан. Вот почему обратное распространение этих ошибок через сеть позволяет корректно настраивать веса связей между всеми слоями. В этом случае величина функции ошибки уменьшается и сеть обучается. Величина ошибки определяется по формуле среднеквадратичной ошибки:

(3.2)

В качестве метода минимизации ошибки используется метод градиентного спуска, суть этого метода сводится к поиску минимума (или максимума) функции за счет движения вдоль вектора градиента. Для поиска минимума движение должно быть осуществляться в направлении антиградиента. Метод градиентного спуска в соответствии с рисунком 3.3.

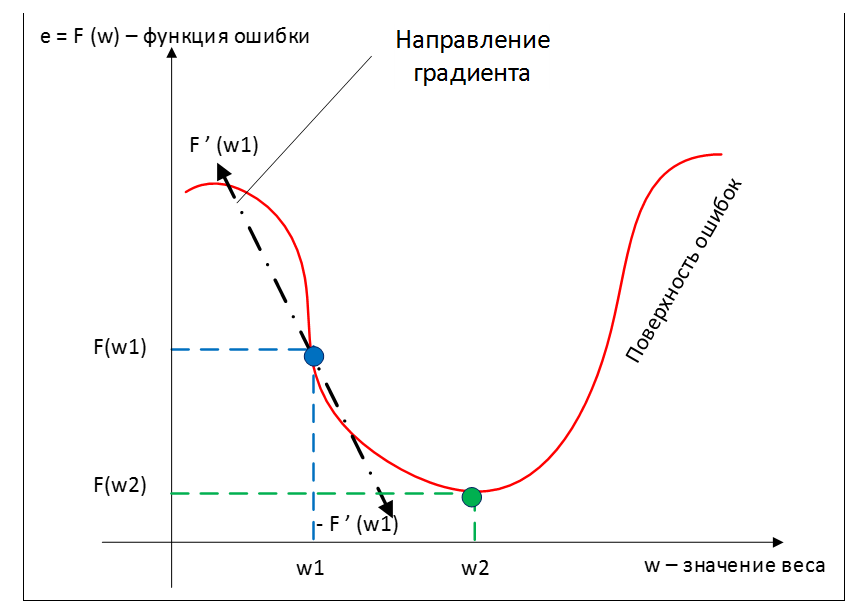


Рисунок 3.3. Метод градиентного спуска

Градиент функции потери представляет из себя вектор частных производных, вычисляющийся по формуле:

Производную функции ошибки по конкретному образу можно записать по правилу цепочки, формула:

Ошибка нейрона обычно записывается в виде символа (дельта). Для выходного слоя ошибка определена в явном виде, если взять производную от формулы 3.1, то получим t минус y, то есть разницу между желаемым и полученным выходом. Но как рассчитать ошибку для скрытых слоев? Для решения этой задачи, как раз и был придуман алгоритм обратного распространения ошибки. Суть его заключается в последовательном вычислении ошибок скрытых слоев с помощью значений ошибки выходного слоя, т.е. значения ошибки распространяются по сети в обратном направлении от выхода к входу.   
  
Ошибка для скрытого слоя рассчитывается по формуле 3.5:

где

Алгоритм распространения ошибки сводится к следующим этапам:

* прямое распространение сигнала по сети, вычисления состояния нейронов;
* вычисление значения ошибки для выходного слоя;
* обратное распространение: последовательно от конца к началу для всех скрытых слоев вычисляем по формуле 3.4;
* обновление весов сети на вычисленную ранее δ ошибки.

Таким образом наша сеть после обучения готова к использованию. На выходе получаем изображение с выделенной областью лица при помощи OpenCV (зеленым прямоугольником) и при помощи нейронной сети (красным прямоугольником). Сделано это для визуального сравнения работы нейронной сети и данной библиотеки. Примеры вывода показаны на рисунках ниже:

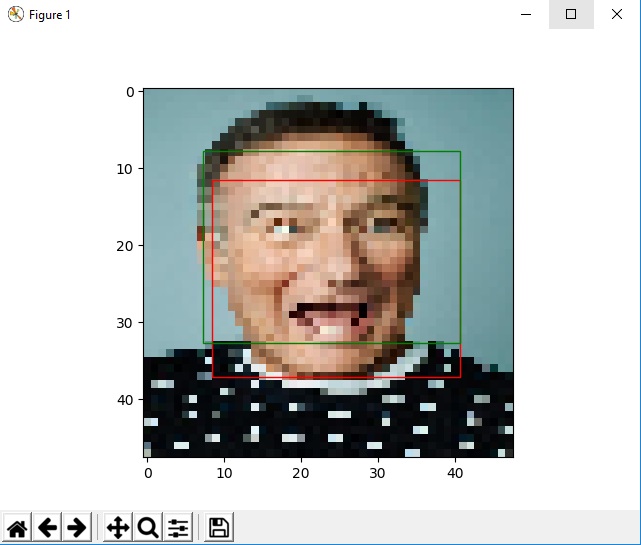


Рисунок 3.4. Пример 1

На мой взгляд на рисунке 3.4. нейронная сеть распознала область лица даже немного лучше, чем библиотека OpenCV, так как захватила область подбородка, чего не сделала библиотека OpenCV

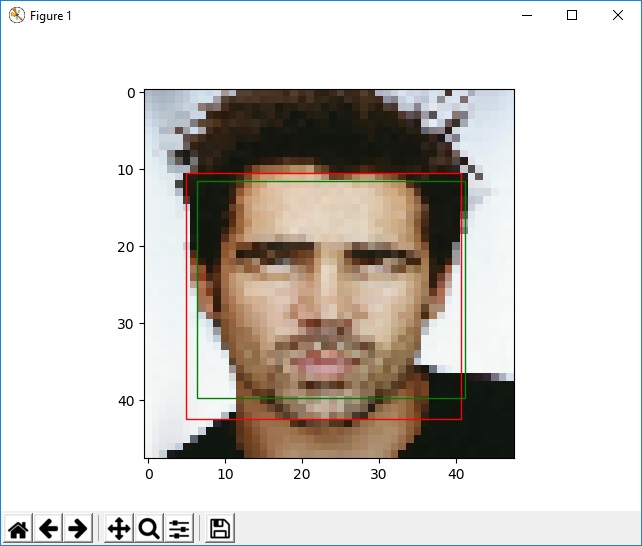


Рисунок 3.5. Пример 2

На рисунке 3.5. аналогичная ситуация как и на рисунке 3.4.

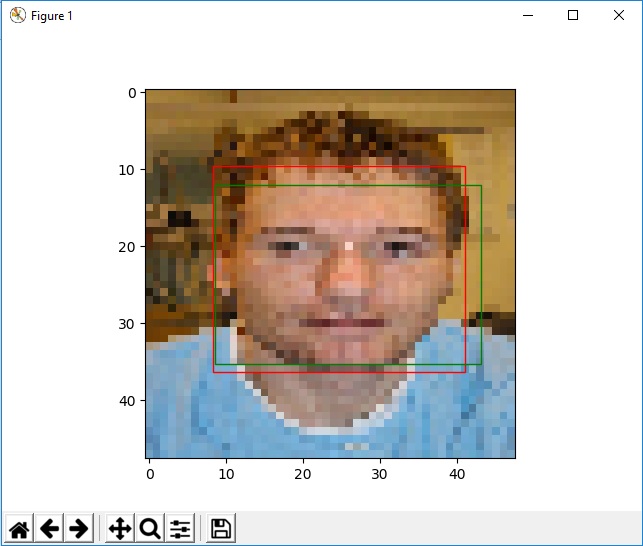


Рисунок 3.6. Пример 3

На рисунке 3.6. нейронная сеть определила лицо практически идентично с библиотекой OpenCV.

Итак, в заключении можно сказать, что сверточные нейронные сети хорошо подходят для задачи распознавания объектов на фото и в частности для локализации области лица. Хочется добавить, что при увеличении количества входных данных улучшится и обучаемость нашей сети в связи с чем распознавание лица будет еще более близким к идеалу (в нашем случае идеал – это результат работы библиотеки OpenCV).

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы были достигнуты все поставленные цели. Согласно поставленным целям были исследованы основные существующие методы распознавания лиц.

В работе рассмотрены основные классы задач распознавания человека по изображению лица. Указаны преимущества и перспективы нейросетевых методов. Отмечены архитектуры нейронных сетей, перспективных для данной задачи. Отмечены перспективные направления исследований в этой области для достижения намеченной цели. Приведены описание и результаты экспериментов по созданию системы контроля доступа на основе анализа изображения лица человека.

В настоящее время нейросетевой подход к задаче контроля доступа на основе анализа изображения лица нейросетевыми методами во многом остаётся недостаточно разработанным, что требует дальнейших исследований в этой области.

СПИСОК ЛИТЕРАТУРЫ

1. Форсайт Д., Понс Ж. "Компьютерное зрение. Современный подход" – М.: Издательский дом "Вильямс", 2004. – 928с.
2. Л. Шапиро, Дж. Стокман. "Компьютерное зрение"  – М.: [Бином. Лаборатория знаний](https://ru.wikipedia.org/w/index.php?title=%D0%91%D0%B8%D0%BD%D0%BE%D0%BC._%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%D0%B7%D0%BD%D0%B0%D0%BD%D0%B8%D0%B9&action=edit&redlink=1), 2006. – 752 с.
3. Д.В. Брилюк, В.В. Старвойтов "Нейросетевые методы распознавания изображений" – Минск, 2002. – 170с.
4. Хайкин С. "Нейронные сети: полное представление" – Издательский дом "Прентисс-Холл", 1999. – 1490с.
5. [Фомин Я. А.](https://ru.wikipedia.org/w/index.php?title=%D0%A4%D0%BE%D0%BC%D0%B8%D0%BD,_%D0%AF%D1%80%D0%BE%D1%81%D0%BB%D0%B0%D0%B2_%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B5%D0%B5%D0%B2%D0%B8%D1%87&action=edit&redlink=1) Распознавание образов: теория и применения. – 2-е изд. – М.: ФАЗИС, 2012. – 429 с.
6. [Фомин Я. А.](https://ru.wikipedia.org/w/index.php?title=%D0%A4%D0%BE%D0%BC%D0%B8%D0%BD,_%D0%AF%D1%80%D0%BE%D1%81%D0%BB%D0%B0%D0%B2_%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B5%D0%B5%D0%B2%D0%B8%D1%87&action=edit&redlink=1), [Тарловский Г. Р.](https://ru.wikipedia.org/w/index.php?title=%D0%A2%D0%B0%D1%80%D0%BB%D0%BE%D0%B2%D1%81%D0%BA%D0%B8%D0%B9,_%D0%93%D0%B5%D0%BD%D0%BD%D0%B0%D0%B4%D0%B8%D0%B9_%D0%A0%D1%83%D0%B4%D0%BE%D0%BB%D1%8C%D1%84%D0%BE%D0%B2%D0%B8%D1%87&action=edit&redlink=1) "Статистическая теория распознавания образов." – М.: Радио и связь, 1986. – 624 с.
7. Горелик А. Л., Скрипкин В. А. Методы распознавания. – 4-е изд. – М.: Высшая школа, 1984, 2004. – 262 с.
8. [Вапник В. Н.](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D0%BF%D0%BD%D0%B8%D0%BA,_%D0%92%D0%BB%D0%B0%D0%B4%D0%B8%D0%BC%D0%B8%D1%80_%D0%9D%D0%B0%D1%83%D0%BC%D0%BE%D0%B2%D0%B8%D1%87), [Червоненкис А. Я.](https://ru.wikipedia.org/wiki/%D0%A7%D0%B5%D1%80%D0%B2%D0%BE%D0%BD%D0%B5%D0%BD%D0%BA%D0%B8%D1%81,_%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B5%D0%B9_%D0%AF%D0%BA%D0%BE%D0%B2%D0%BB%D0%B5%D0%B2%D0%B8%D1%87) Теория распознавания образов. – М.: Наука, 1974. — 416 с.
9. Галушкин А.И., Томашевич Д.С., Томашевич Н.С. "Методы

реализации инвариантности к аффинным преобразованиям двумерных изображений" // Приложение к журналу "Информационные технологии". - 2001. - №1. - С.1-19.

1. Земцов А. "Алгоритмы распознавания лиц и их применение в системах биометрического контроля доступа. LAP Lambert Academic Publishing", 2011, 128с.
2. Закревский А.Д. "Логика распознавания", 2003, 144с.
3. [К. Верхаген](http://www.ozon.ru/person/23740498/), [Р. Дейн](http://www.ozon.ru/person/23740507/), [Ф. Грун](http://www.ozon.ru/person/23740514/), [Й. Йостен](http://www.ozon.ru/person/23740523/), [П. Вербек](http://www.ozon.ru/person/23740528/) "Распознавание образов. Состояние и перспективы. Радио и связь" 1985, 104с.
4. Применение нейросетей в распознавании изображений [электронный ресурс]. – https://habr.com/post/74326/
5. P. A. Viola and M. J. Jones, “Robust real-time face detection”, International Journal of Computer Vision, 137 - 154, 2004.
6. Распознавание лиц: нейронная сеть в действии [электронный ресурс]. – http://www.azoft.ru/blog/raspoznavanie-lic-svertochnaya-nejronnaya-set/
7. Галушкин А.И. Синтез многослойных нейронных систем распознавания образов. М.: Энергия, 1974. – 368с.
8. The CImg library [электорнный ресурс]. http://cimg.sourceforge.net/
9. Learning OpenCV 3 Computer Vision in C++ with the OpenCV Library By [Adrian Kaehler, Gary Bradski](http://shop.oreilly.com/product/0636920044765.do#tab_04_2) Publisher: O'Reilly Media Release Date: September 2015 Pages: 650с.
10. Хайкин С. "Нейронные сети. Полный курс" – М.: Издательский дом "Вильямс", 2006. – 1104 с.
11. Галушкин А.И. Нейронные сети: основы теории. М.: Горячая линия-Телеком, 2010. – 496 с.

ПРИЛОЖЕНИЕ 1

import cv2  
import sys  
  
*# Get user supplied values*imagePath = sys.argv[1]  
*# imagePath = "12.png"*cascPath = "haarcascade\_frontalface\_default.xml"  
  
*# Create the haar cascade*faceCascade = cv2.CascadeClassifier(cascPath)  
  
*# Read the image*image = cv2.imread(imagePath)  
gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
  
*# Detect faces in the image*faces = faceCascade.detectMultiScale(  
 gray,  
 scaleFactor=1.1,  
 minNeighbors=5,  
 minSize=(30, 30)  
 *#flags = cv2.CV\_HAAR\_SCALE\_IMAGE*)  
  
*# print("Found {0} faces!".format(len(faces)))  
  
# Draw a rectangle around the faces*for (x, y, w, h) in faces:  
 print(imagePath, x, y, w, h);  
 cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)  
 cv2.imwrite(imagePath + '.jpg', image)  
 break  
  
*# cv2.imshow("Faces found", image)  
# cv2.waitKey(0)*

ПРИЛОЖЕНИЕ 2

Листинг 1

import numpy as np  
import chainer  
from chainer.backends import cuda  
from chainer import Function, gradient\_check, report, training, utils, Variable  
from chainer import datasets, iterators, optimizers, serializers  
from chainer import Link, Chain, ChainList  
import chainer.functions as F  
import chainer.links as L  
  
  
from chainer.training import extensions  
def augment(x, y):  
 angles = np.random.choice([0, 1, 2, 3], len(x))  
 for i in range(1,4):  
 index = np.where(angles == i)[0]  
 x[index] = np.rot90(x[index], i, (2, 3))  
 if i%2 != 0:  
 c = y[index, 3]  
 y[index, 3] = y[index, 2]  
 y[index, 2] = c  
 theta = np.radians(90 \* i)  
 c, s = np.cos(theta), np.sin(theta)  
 R = np.array(((c, -s), (s, c)))  
 y[index, :2] = np.dot(y[index, :2], R)  
  
  
class FaceNet():  
 def \_\_init\_\_(self):  
 self.model = Chain(  
 conv1=L.Convolution2D(3, 20, 3, 1, 1),  
 conv2=L.Convolution2D(20, 20, 3, 1, 1),  
  
 conv3=L.Convolution2D(20, 40, 3, 1, 1),  
 conv4=L.Convolution2D(40, 40, 3, 1, 1),  
  
 linear1=L.Linear(None, 100),  
 linear2=L.Linear(100, 4)  
 )  
  
 self.optimizer = optimizers.Adam()  
 self.optimizer.setup(self.model)  
  
 def foward(self, x):  
 out = self.model.conv1(x)  
 out = F.elu(out)  
 out = self.model.conv2(out)  
  
 out = F.max\_pooling\_2d(out, 2)  
 out = F.elu(out)  
 out = self.model.conv3(out)  
 out = F.elu(out)  
 out = self.model.conv4(out)  
 out = F.elu(out)  
  
 out = F.average\_pooling\_2d(out, 6)  
 out = F.dropout(out)  
 out = self.model.linear1(out)  
 out = F.elu(out)  
 out = F.dropout(out)  
 out = self.model.linear2(out)  
  
  
 return out  
  
 def predict(self, X, step=100):  
 with chainer.using\_config('train', False):  
 with chainer.no\_backprop\_mode():  
 output = []  
 for i in range(0, len(X), step):  
 x = Variable(X[i:i + step])  
 output.append(self.foward(x).data)  
 return np.vstack(output)  
  
 def score(self, X, Y, step=100):  
 predicted = self.predict(X, step)  
 score = F.r2\_score(predicted, Y).data  
 return score  
  
  
  
 def fit(self, X, Y, batchsize=100, n\_epoch=20):  
 with chainer.using\_config('train', True):  
 learning\_curve = []  
 for epoch in range(n\_epoch):  
 print('epoch ',epoch)  
 index = np.random.permutation(len(X))  
 for i in range(0, len(index), batchsize):  
 self.model.cleargrads()  
 print(i)  
 x = X[index[i:i+batchsize]]  
 y = Y[index[i:i+batchsize]]  
 *#augment(x, y)* x = Variable(x)  
 y = Variable(y)  
  
 output = self.foward(x)  
 loss = F.mean\_squared\_error(y, output)  
 loss.backward()  
  
 learning\_curve.append(float(loss.data))  
  
 self.optimizer.update()  
 return learning\_curve

Листинг 2

import numpy as np  
from matplotlib import pyplot as pl  
import pandas  
from sklearn.linear\_model import LinearRegression  
from neural\_network import FaceNet  
from PIL import Image  
import matplotlib.patches as patches  
  
data = pandas.read\_csv('data.csv', delimiter=' ')  
pathes = data[['File']].as\_matrix()[:, 0]  
rects = data[['x', 'y', 'w', 'h']].as\_matrix()  
size = 48  
imgX = []  
rectY = []  
  
  
for file, rect in zip(pathes, rects):  
 img = Image.open(file)  
 x = rect[0] / img.size[0]  
 y = rect[1] / img.size[1]  
 w = rect[2] / img.size[0]  
 h = rect[3] / img.size[1]  
 img = (np.array(img.resize((size, size)))/128.0 - 1).astype(np.float32).T  
 r = np.array([x, y, w, h], np.float32)  
 imgX.append(img)  
 rectY.append(r)  
  
imgX = np.array(imgX)  
rectY = np.array(rectY)  
rectY = (2 \* rectY - 1).astype(np.float32)  
  
index = np.random.permutation(len(imgX))  
train\_index = index[:int(len(index)\*0.8)]  
test\_index = index[int(len(index)\*0.8):]  
  
  
  
"""predicted\_rect = np.zeros(rectY[test\_index].shape)  
for i in range(4):  
 print('start fit %s'%i)  
 L = LinearRegression()  
 L.fit(imgX[train\_index].reshape(len(train\_index), -1), rectY[train\_index, [i]])  
  
 print(L.score(imgX[train\_index].reshape(len(train\_index), -1), rectY[train\_index, [i]]))  
 print(L.score(imgX[test\_index].reshape(len(test\_index), -1), rectY[test\_index, [i]]))  
 predicted\_rect[:, i] = L.predict(imgX[test\_index].reshape(len(test\_index), -1))"""  
  
  
model = FaceNet()  
print('start fit')  
learning\_curve = model.fit(imgX[train\_index], rectY[train\_index], 20, 10)  
print('end fit')  
print(model.score(imgX[train\_index], rectY[train\_index]))  
print(model.score(imgX[test\_index], rectY[test\_index]))  
  
predicted\_rect = model.predict(imgX[train\_index])  
  
pl.figure()  
"pl.plot(learning\_curve)"  
pl.show()  
  
rectY = 0.5 \* (rectY + 1)  
predicted\_rect = 0.5 \* (predicted\_rect + 1)  
  
  
  
for img, rect, rect\_true in zip(imgX[test\_index], predicted\_rect, rectY[test\_index]):  
 image = (img.T + 1)/2  
 fig, ax = pl.subplots(1)  
 ax.imshow(image)  
 rect\_patch\_predicted = patches.Rectangle((rect[0]\*size, rect[1]\*size), rect[2] \* size, rect[3] \* size,  
 linewidth=1, edgecolor='r', facecolor='none')  
 rect\_patch\_true = patches.Rectangle((rect\_true[0] \* size, rect\_true[1] \* size), rect\_true[2] \* size, rect\_true[3] \* size,  
 linewidth=1, edgecolor='g', facecolor='none')  
  
 ax.add\_patch(rect\_patch\_predicted)  
 ax.add\_patch(rect\_patch\_true)  
 pl.show()